Implementing and Enhancing minBERT for Optimized Performance on Multiple Downstream Classification Tasks

Stanford CS224N Default Project

Priti Rangnekar
Department of Computer Science
Stanford University
pritir@stanford.edu

Abstract

The Bidirectional Encoder Representations from Transformers (BERT), a language model that is based on the transformer architecture and generates contextual word representations, has proven to be promising in its performance on a number of natural language processing tasks since its introduction by Google in 2018. In this project, we seek to identify and assess the effectiveness of approaches for obtaining robust and generalizable sentence embeddings that can perform well on multiple tasks, which is increasingly crucial for developing models that can be used in diverse contexts and settings in the real world. To do so, we first implement minBERT and perform fine-tuning for the model to perform sentiment classification. We then seek to improve minBERT's ability to perform multiple downstream tasks - namely, sentiment analysis, paraphrase detection, and semantic textual similarity. Specifically, we incorporate and evaluate the techniques of cosine-similarity fine-tuning for semantic textual similarity and paraphrase detection, multitask fine-tuning with gradient surgery, and the effect of ordering of tasks in round robin training on sentiment analysis. Cosine similarity fine-tuning for semantic textual similarity and multitask fine-tuning with gradient surgery are shown to be effective, resulting in accuracy of 0.512 for sentiment classification, 0.719 for paraphrasing, and correlation of 0.629 for semantic textual similarity on the test set. We also identify that cosine similarity is more effective for semantic textual similarity than for paraphrase detection, and that a modified round robin training approach, which trains on semantic analysis in both the beginning, middle, and end of a single cycle, can improve performance on semantic analysis. Our work enables future investigation into these approaches as a means for enhancing performance on individual tasks as well as combinations of tasks, a crucial area of study as the research community aims to make models more generalizable and robust.

1 Key Information to include

- Mentor: Aditya Agrawal
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Natural language processing (NLP) has been characterized by rapid advancements in recent years, particularly in light of the rise of attention mechanisms and the transformer architecture. As articulated

in the 2017 paper "Attention Is All You Need" by Google (Vaswani et al., 2017), attention, which assigns varying degrees of importance to different parts of a sentence, can be relied upon to develop a transformer. In the transformer architecture, text is converted to a token - a numerical representation - and then to a vector using a word embedding table. Tokens are then contextualized within a broader context window using a multi-head attention mechanism which amplifies important tokens and diminishes less important tokens. The BERT model, one such example of a transformer, has especially been compelling (Devlin et al., 2018). An encoder-only model primarily used for understanding and encoding text as opposed to generating text, BERT is characterized by its consideration of bidirectional - both left and right - context in order to build representations of text. Furthermore, having been trained on masked-language modeling and next sentence prediction, BERT is well-equipped to perform real-world natural language processing tasks and can be fine-tuned on further tasks. Thus, in our paper, we seek to understand how to enhance a minBERT implementation such that it can simultaneously perform well on 3 different tasks: sentiment analysis, paraphrase, and semantic textual similarity by using the techniques of cosine-similarity fine-tuning and multitask fine-tuning with gradient surgery. These techniques are of interest since cosine similarity is a popular approach to computing similarity between embeddings, and multitask fine-tuning with gradient surgery combines the multiple losses across tasks, while accounting for the fact that gradient directions may conflict with one another. Therefore, we believe that such approaches have potential for improving minBERT's performance on individual tasks as well as overall performance.

3 Related Work

Since BERT's initial development, a variety of research has been conducted as to how to improve BERT's performance, of which we highlight two. As Bi et. al. explain in "MtRec: Multi-Task Learning over BERT for News Recommendation," traditional methods for training BERT for news representations have treated additional fields of information as additional features and simply combined their feature vectors. However, these shallow feature encodings are not compatible with BERT's deeper encodings. Thus, the authors found that a multi-task method that adds together the losses on different tasks - in their case, category classification and named entity recognition, rather than fine-tuning on individual tasks one at a time, can boost model performance (Bi et al., 2022). The researchers also draw upon a technique known as gradient surgery, which recognizes that the gradients for the different tasks may differ and resolves these conflicts by projecting the gradient of each task onto the normal plane of another conflicting task's gradient, enabling the model to learn more smoothly. As for cosine similarity, Reimers and Gurevych in 2019 presented Sentence-BERT, a modification of the pretrained BERT network that derives "semantically meaningful sentence embeddings that can be compared using cosine similarity" (Reimers and Gurevych, 2019) through siamese and triplet network structures. They found that this approach achieved significant improvements over state-of-the-art sentence embeddings methods on the semantic textual similarity task. Thus, in our research, we seek to apply and evaluate these methods in an alternate context, namely that of performance on three different tasks.

4 Approach

4.1 Initial minBERT architecture

Our approach begins with implementing minBERT. In doing so, we implement multi-head self attention and the transformer layer, as inspired by the description in the original BERT paper and in "Attention Is All You Need." We also utilize an Adam Optimizer, which computes adaptive learning rates for different parameters based on estimates of the moments of the gradient, as described in "Decoupled Weight Decay Regularization" (Loshchilov and Hutter, 2017) and "Adam: A Method for Stochastic Optimization" (Kingma and Ba, 2014). Furthermore, our minBERT model uses the same embeddings for our three different tasks, but with 3 different classification heads - one for each task, as explained next.

Sentiment Analysis (SST) To predict sentiment of an input, we first use the input ids and attention mask to acquire a pooled representation using BERT embeddings. Next, we apply a dropout layer with probability 0.1. Finally, we use a linear classifier, with an input layer size equivalent to the model's hidden layer size and an output layer size equivalent to the number of possible sentiment classes. For our loss function, we use cross entropy loss between the model's logits and the true

sentiment labels.
Paraphrase Detection (PARA) To predict if 2 inputs are paraphrases, for each of the 2 input ids and attention masks, we acquire a pooled representation and then apply dropout with probability 0.1. Then, we concatenate the 2 embeddings and pass the concatenation into a linear classifier with an input layer size of twice the model's hidden layer size (since we concatenate 2 embeddings), and an output layer size of 1 (since we acquire a single logit). We choose to concatenate embeddings in order to retain and incorporate information about both of the sentences. For our loss function, we use binary cross entropy loss with logits.
Semantic Textual Similarity (STS) To predict semantic textual similarity between 2 inputs, we use an approach very similar to paraphrase detection, observing that both of these tasks are related to the similarity between 2 inputs. For each of the 2 input ids and attention masks, we acquire a pooled representation and then apply dropout with probability 0.1. Then, we concatenate the 2 embeddings and pass the concatenation into a linear classifier with an input layer size of twice the model's hidden layer size, since we concatenate 2 embeddings, and an output layer size of 1, since we acquire a single logit. For our loss function, however, we use mean squared error loss based on the difference between our predicted logit and the true similarity value. This choice is due to the fact that semantic textual similarity values are continuous, whereas paraphrase detection is binary.

4.2 Multi-task Fine-Tuning Methods

Our initial approach to training does not perform multi-task finetuning and simply utilizes initial BERT embeddings. We then investigate the following training methods.

4.2.1 Round Robin Training Orders

For our initial training method, we use round robin training (Stickland and Murray, 2019), in which in each epoch, we train on a single dataset. We then cycle through the datasets. We then investigate different orders in which the datasets are cycled through with this round robin approach.

4.2.2 Multitask Fine-Tuning with Gradient Surgery

As an alternate approach, we investigate multi-task learning by training on all 3 datasets in each epoch, adding together the losses on each of our tasks - weighted equally - as suggested by (Bi et al., 2022) to have our loss function be $L_{total} = L_{SST} + L_{PARA} + L_{STS}$.
Furthermore, we implement gradient surgery based on the PCGrad library by (Yu et al., 2020), which projects task gradients as follows.

$$\mathbf{g_i} = \mathbf{g_i} - \frac{\mathbf{g_i} \cdot \mathbf{g_j}}{||\mathbf{g_j}||^2} \cdot \mathbf{g_j}$$

4.3 Incorporating Cosine Similarity

4.3.1 Cosine Similarity for Semantic Textual Similarity

To improve the quality of our embeddings and performance on the semantic textual similarity task, we investigate fine tuning on this task by calculating the cosine similarity between the pooled representation of each of our 2 inputs, as opposed to simply concatenating them (PyTorchContributors, 2023b). We then use mean squared error loss as before, making our overall approach a variation on the use of CosineEmbeddingLoss ((PyTorchContributors, 2023a)). This refinement is motivated by the idea that rather than simply concatenating representations of inputs, we can more directly calculate a value representing the similarity between two sentences by computing the cosine similarity between the two.

$$\text{similarity} = \frac{x_1 \cdot x_2}{\max(\|x_1\|_2 \cdot \|x_2\|_2, \epsilon)}$$

4.3.2 Cosine Similarity for Paraphrase Detection

Based on the insight that like semantic textual similarity, paraphrase detection also involves some representations of sentence similarity - since paraphrased sentences may be similar to one another, we also investigate cosine similarity in our paraphrase classifier. Specifically, we calculate the cosine similarity between our 2 pooled representations, concatenate the cosine similarity with our pooled representations, and apply dropout with probability 0.1 before using the linear classifier. This is based on the notions that paraphrasing is not entirely a notion of similarity and that preserving information about the sentences themselves may be important for the model.

5 Experiments

5.1 Data

For fine-tuning our initial minBERT implementation's embeddings for sentiment analysis, we use the Stanford Sentiment Treebank (SST) dataset and the CFIMDB dataset. The SST dataset consists of phrases with a label of either negative (0), somewhat negative (1), neutral (2), somewhat positive (3), or positive (4). The CFIMDB dataset contains movie reviews, each with a binary label of negative or positive. For fine-tuning our minBERT implementation on downstream tasks, we again use the SST dataset for sentiment analysis, as well as the Quora dataset for paraphrase detection, and the SemEval dataset for semantic textual analysis. The Quora dataset consists of question pairs with binary labels indicating whether particular instances are paraphrases of one another. The SemEval STS Benchmark Dataset consists of sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). All of our datasets are publicly available and are obtained from the default project.

Dataset	Task	Train Set	Dev Set	Test Set
SST	Sentiment	8,544	1,101	2,210
CFIMDB	Sentiment	1701	245	488
Quora (PARA)	Paraphrase	141,506	20,215	40,431
SemEval STS Benchmark	Semantic Textual Similarity	6,041	864	1,726

5.2 Evaluation method

For sentiment analysis, since the labels are discrete, the evaluation metric is accuracy, the ratio of predicted values that are exact matches with the true values to the total number of predictions. For paraphrase detection, since the labels are binary, we again simply use accuracy as the evaluation metric. For semantic textual similarity, the evaluation metric is the Pearson correlation of the true similarity values against the predicted similarity values. We also use a combined evaluation metric - the sum of the accuracy for sentiment analysis, the accuracy for paraphrase detection, and the Pearson correlation for semantic textual similarity - for deciding when to save our model checkpoints, as a proxy for the combined score calculated by the leaderboard.

5.3 Experimental details

5.3.1 Experiment 0: Baseline

We first implemented the following baselin model using our initial minBERT architecture and without multitask training:

Model	Epochs	Learning Rate	Batch Size
Baseline Finetune	9	1e-5	16

5.3.2 Experiment 1: Pretrain vs. Finetune

We then implement a round robin training algorithm that trains for 9 epochs in the order (SST, PARA, STS) x 3 and run it on both pretrain and finetune modes. This is the only experiment in our study in which we test for pretraining mode, before pursuing fine-tuning mode only.

	Learning Rate	**Epochs**	**Batch Size**
RR-SSTFIRST-Pretrain	1e-3	9	16
RR-SSTFIRST-Finetune	1e-5	9	16

5.3.3 Experiment 2: Round Robin (RR) vs. Multitask Fine-Tuning with Gradient Surgery (PCG)

To run a model with PCGrad, we are training on all 3 datasets simultaneously and combining the loss. To reduce overfitting, we use a form of downsampling by only training on the minimum number of batches across all datasets. With a batch size of 8, we find that STS, PARA, and SST have 1068, 17,688, and 755 batches respectively. Thus, we are capped at 755 batches of size 8 each. Then, for more efficient training during this experimentation phase, we further constrain ourselves to utilizing 200 batches per epoch. We do the same for our round robin training for more consistent comparison.

	Learning Rate	Epochs	Batch Size
RR-SSTFIRST-200	1e-5	9	8
PCG-200	1e-5	9	8

5.3.4 Experiment 3: Cosine Similarity for STS

Next, we evaluate cosine similarity for STS. We perform ablation studies to evaluate the effect of cosine similarity both alongside and without PCGrad. Thus, we continue using 200 batches as was done in the previous section.

	Learning Rate	Epochs	Batch Size
RR-SSTFIRST-COSSTS-200	1e-5	9	8
PCG-COSSTS-200	1e-5	9	8

5.3.5 Experiment 4: Cosine Similarity for PARA

Next, we evaluate cosine similarity for PARA. We utilize PCGrad for all models in this section, based on the results of the previous section. Due to increased GPU limits during the time of training for this experiment, we also increased our batch size to 16 for all models in this section.

	Learning Rate	Epochs	Batch Size
PCG-COSSTS-200-16	1e-5	9	16
PCG-COSPARA-200-16	1e-5	9	16
PCG-COSSTS-COSPARA-200-16	1e-5	9	16

5.3.6 Experiment 5: Effect of Round Robin Training Orders on SST

Lastly, having tested enhancements targeted towards both STS and PARA, we investigate whetherwe can improve performance on SST. We investigate modifying our original round robin algorithm, which previously trained on SST first in each round robin cycle, to explore the orders below. For a consistent basis for comparison with our previous results, we use a batch size of 8, 200 batches, and cosine similarity for STS. For RR-SSTLAST-COSSTS, we train in the reverse order as we did for RR-COSSTS-200; in other words, we train in the order (STS-PARA-SST) x 3. For RR-SUM12-COSSTS, we investigate a modified round robin, in which SST is neither at the beginning of each cycle, nor at the end of each cycle. Since the total sum of epoch numbers is 36, we assign datasets to epochs such that the total sum of epoch numbers for each dataset is the same, namely 12. Thus, we train our 9 epochs in the order 0: SST, 1: PARA, 2: STS, 3: STS, 4: SST, 5: PARA, 6: PARA, 7: STS, 8: SST.

	Learning Rate	Epochs	Batch Size
RR-SSTLAST-COSSTS-200	1e-5	9	8
RR-SUM12-COSSTS-200	1e-5	9	8

5.3.7 Experiment 6: Overall Model

Lastly, based on our results to this point, we hypothesized that a model utilizing both PCGrad and cosine similarity for STS will have the best performance overall and thus train it for the full minimum number of batches across 3 datasets, with a batch size of 8 due to GPU considerations.

	Learning Rate	Epochs	Batch Size
PCGrad-COSSTS-MinBatches	1e-5	9	8

	SST	PARA	STS	Sum
Baseline Finetune	0.144	0.380	-0.007	0.517
RR-SSTFIRST-Pretrain	0.292	0.647	0.250	1.189
RR-SSTFIRST-Finetune	0.522	0.778	0.358	1.658
RR-SSTFIRST-200	0.460	0.667	0.320	1.447
PCG-200	0.482	0.702	0.351	1.535
RR-SSTFIRST-COSSTS-200	0.452	0.630	0.564	1.646
PCG-COSSTS-200	0.479	0.678	0.603	1.760
PCG-COSSTS-200-16	0.500	0.693	0.641	1.834
PCG-COSPARA-200-16	0.495	0.696	0.369	1.805
PCG-COSSTS-COSPARA-200-16	0.503	0.691	0.614	1.808
RR-SSTLAST-COSSTS-200	0.396	0.625	0.459	1.480
RR-SUM12-COSSTS-200	0.487	0.625	0.458	1.570
PCGrad-COSSTS-MinBatches	0.505	0.719	0.656	1.880

5.4 Results

Our final model PCGrad-COSSTS-MinBatches, shown above, earns an overall score of 0.684 on the dev leaderboard; it earns scores of 0.512, 0.719, and 0.629 on the test leaderboard, with an overall score of 0.682.

Finetuning mode outperformed pretraining mode: RR-SSTFIRST-Finetune significantly outperforms RR-SSTFIRST-Pretrain on all 3 tasks, which aligns with our expectations, since finetune mode adjusts BERT parameters in addition to the weights of the task layers, thus making the model more influenced by our datasets and training routine on the tasks.

Cosine Similarity for STS (COSSTS) and PCGrad (PCG) improved results on all 3 tasks, both when used in isolation and when used together: PCG-COSSTS-200, using both cosine similarity for STS and PCGrad, achieves scores of 0.479, 0.678, and 0.603. In contrast, RR-STSFIRST-200, using neither, achieves scores of only 0.460, 0.667, and 0.320. Thus, these 2 methods in combination improve SST, PARA, and STS by 0.019, 0.011, and 0.283 respectively. While the significant improvement to STS is to be expected, we see that using COSSTS leaves performance on the other tasks relatively unchanged or slightly diminished. For example, with round robin training, using cosine similarity decreased SST performance from 0.460 to 0.452 (-0.008) and PARA performance from 0.667 to 0.630 (-0.037). With PCGrad, the decreases are less stark, with SST performance decreasing by only 0.003 and PARA by 0.024. This may be due to the fact that PCGrad combines losses across tasks, and gradient surgery helps mitigate conflicitng directions, making the model truly learn multiple tasks at once, rather than excelling at one task at the expense of another. In contrast, PCGrad in isolation improves performance on all 3 tasks, with PCG-200 outperforming RR-SSTFIRST-200 on all 3 tasks, and PCG-COSSTS-200 outperforming RR-SSTFIRST-COSSTS-200 on all 3 tasks. However, the overall improvement is less noticeable, with PCG-200 improving over RR-SSTFIRST-200 by 0.088, while RR-SSTFIRST-COSSTS-200 improves over RR-SSTFIRST-200 by 0.199, largely due to the boost in STS performance by 76.25 percent of its original value.

Cosine Similarity for PARA (COSPARA) did not significantly improve performance. We first observe that PCG-COSSTS-COSPARA-200-16, which uses cosine similarity for both PARA and STS, performs slightly worse than PCG-COSSTS-200-16, which only uses cosine similarity for STS, with a diference of -0.029. Specificallly, both the PARA and STS values slightly decline, but not to an extent that we can conclusively say that COSPARA has a negative effect. However, the model that used only COSPARA had the highest PARA accuracy of all - 0.696, as opposed to 0.693 when only COSSTS was used, and 0.691 when COSPARA and COSSTS were used. That being said, this model had a very low correlation for STS, that of only 0.369. Thus, we see that for overall model performance, COSSTS appears to be far more crucial as it significantly boosts STS performance, which is otherwise very low. This behavior of COSPARA may be due to the fact that the cosine value constitutes only a small part of the overall concatenated representation - our paraphrase linear layer was increased only increased by 1 from (hidden size * 2) to (hidden size * 2 + 1), and thus does not have a major effect, even though in principle it makes sense to consider some form of similarity. Although we do apply dropout to randomly reduce the role that other portions of the concatenated embedding may play relative to the cosine value, our dropout probability being only 0.1 likely means that the dropout does not significantly affect our findings.

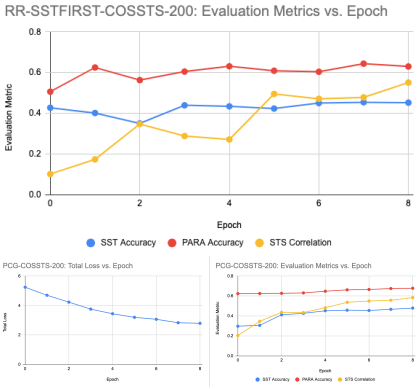

Figure 1: In Round Robin (top), evaluation values plateau or drop for tasks not actively being trained on their own datasets; PCGrad (Bottom), shows more consistent improvement across tasks.

Within round robin scheduling, STS performance is highest when STS is scheduled to be trained on both first and last. We compare the results of RR-SSTFIRST-COSSTS-200, RR-SSTLAST-COSSTS-200, and RR-SUM12-COSSTS-200. Across all 3, the PARA accuracy is roughly similar, with 0.630 for the first and 0.625 for the second too. This is likely because across all 3 runs, PARA is predominantly trained on in the middle epochs. RR-SSTFIRST-COSSTS-200, with the (SST-PARA-STS) x 3 ordering, has higher values for SST and STS compared to RR-SSTLAST-COSSTS-200, with the (STS-PARA-SST) x 3 ordering. Thus, given the nature of the 3 tasks, it may be that SST performs better when trained on first and STS generally performs better when trained later on. This finding is slightly different from our expectations, which were that SST would perform better when trained on in the later portions of a cycle, as it would be able to utilize existing embeddings and learnings from being trained on PARA and STS, and would not have its embeddings overwritten by training on PARA and STS. However, it may be that being trained on early allows the initial embeddings to be more influenced by SST, or that further training on PARA and STS allow the model to learn semantic information, actually improving performance on SST too. Based on these findings, our novel RR-SUM12-COSSTS-200 variation of round robin training, which attempts to allow SST training to take place in the beginning, middle, and end across our 3 cycles, appears to produce the best SST performance of 0.487, in contrast to 0.452 for RR-SSTFIRST-COSSTS-200 and RR-SSTLAST-COSSTS-200.

6 Analysis

Having explained our experiment results in the previous section, we know discuss our final model's performance on each of the three tasks.

Sentiment Analysis: From the normalized confusion matrix below, we see the darkest regions are along the diagonal corresponding to correct predictions, indicating strong performance. The non-diagonal squares with high values are still close to the diagonal, indicating slight inaccuracies - for example, predicting an input as neutral instead of somewhat negative. The model also has 0 cases in which it predicts an input of 0 as 4, or an input of 4 as 0, indicating that it has learnt the core of the sentiment analysis task well enough to not make egregious errors. The model correctly predicts for a variety of short and long inputs, including "Very bad"

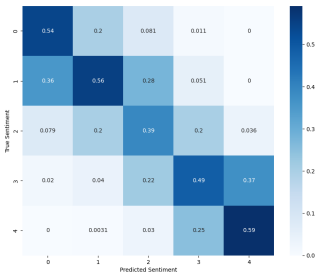

Figure 2: Confusion Matrix for SST

(0), "A moody , multi-dimensional love story and sci-fi mystery , Solaris is a thought-provoking , haunting film that allows the seeds of the imagination to germinate" (4), and "Chilling but uncommercial look into the mind of Jeffrey Dahmer , serial killer" (2), indicating that the attention mechanism does allow it to pick up on sentiment-related terms even earlier in the sentence. However, it fails to capture nuanced context or sarcasm, as indicated by its prediction of "A great ensemble cast can't lift this heartfelt enterprise out of the familiar" as somewhat positive instead of negative, likely due to the words "great" and "heartfelt."

Paraphrase Detection On paraphrase detection, the model achieves precision of 0.641 and recall of 0.569, indicating that it is more prone to false negatives than false positives. This may be due to the imbalanced nature of the dataset, with 89225 instances of non-paraphrases and only 52273 instances of paraphrases.
The model produces false positives for inputs that have many exact words in common but have subtle differences that make the questions non-duplicates. For example, the model classifies "What are the best and profitable ways for saving money?" and "What are your best ways to save money?" as duplicates. The model also fails to detect paraphrases in which one input provides extra context, such as "I want to travel to Dubai. What are the best hotels to stay?" and "What are the best hotels on Dubai?," or inputs that have the same big picture idea but give different examples, such as "Why are Facebook, Google, and others not allowed in China?" and "Why are Google, Facebook, YouTube and other social networking sites banned in China?"

Semantic Textual Similarity: The cosine similarity based model generally predicts higher similarity values than the original concatenation-based model. For example, for the input "Some guy sitting on a couch watching television" and "A guy is sitting on the couch watching TV," which has a true STS value of 5 (same meaning) the original model only predicted 2.033 (share some details), while the cosine model predicted 4.82. This may be due to cosine similarity's ability to capture notions of similarity - such as synonyms - present in the input's embeddings, that pure concatenation may not grasp. This also means that the cosine similarity model can overpredict values for sentences that are about the same big picture idea but differ in focus or details. For example, for the inputs "Government bonds sold off sharply after Greenspan told Congress on Tuesday that the U.S. economy "could very well be embarking on a period of extended growth"" and "Greenspan told Congress on Tuesday the U.S. economy "could very well be embarking on a period of sustained growth"." which have a true similarity value of 3.2, the cosine similarity model predicts 4.59, while the original model predicted 3.127, a closer value to truth.

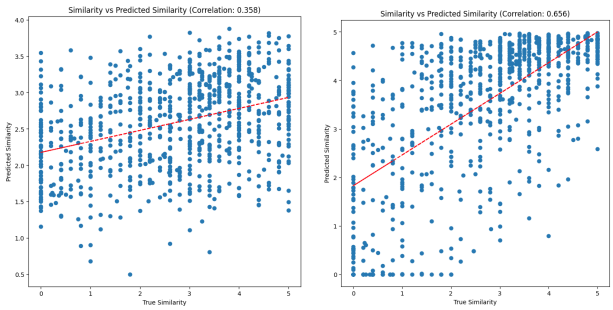

Figure 3: STS Correlation Without (Left) and With (Right) Cosine Similarity.

7 Conclusion

Overall, we found that we were able to significantly improve upon our baseline through our enhancements of multitask finetuning with gradient surgery for sentiment analysis, paraphrase detection, and semantic textual similarity, as well as incorporation of cosine similarity for the semantic textual similarity task. Notably, with all other hyperparameters the same, we improved our model from 0.460 SST, 0.667 PARA, 0.320 to 0.479 SST, 0.678 PARA, and 0.603 STS through these 2 techniques. We also make additional research contributions. Cosine similarity for the STS task was more effective than incorporating it for the paraphrase detection task. The SST task also performed best when trained in the beginning, middle, and end portions of a single cycle as opposed to always being the first task in cycles, or always the last task in cycles. A limitation in our approach is regarding the amount of data used and amount of time trained for. Thus, our research provides areas for further investigation, and our existing experiments show that through changes such as increasing batch sizes, number of batches, or epochs, accuracy and correlation values would be higher. We also need to extend our investigations by testing further combinations of configurations beyond our sequential enhancement approach, to explore potentially unexpected interactions between different training schemas, such as PCGrad, or model architectures, such as whether both PARA and STS use cosine similarity. By doing so, we would be able to garner further insights regarding the interactions between tasks in multitask training and raise follow-up questions regarding which tasks should be prioritized, or balancing learning across tasks, for BERT-like models.

8 Ethics Statement

We believe that an understanding of ethical challenges, possible societal risks, and mitigation strategies is crucial, especially at a time when natural language processing is field that is rapidly growing, both in academic research and human impact.

The first ethical challenge that we consider is in the datasets we use. We utilized a variety of datasets for training our 3 tasks, including movie reviews and Quora questions. Although these datasets are commonly used for artificial intelligence research and have been professionally curated, it is important to be aware of the privacy and consent concerns surrounding scraping social media or review posting sites. While we are not aware of any such concerns at this point in time regarding our data, we are proud to be consciously using datasets for training that have been vetted. We will also continue to monitor the projects and sites from which we draw our datasets and take appropriate action to not utilize any datasets we find to be from unethical sources.

Another ethical challenge that we consider is in the training of our model. In order to improve our overall performance on all 3 tasks, we heavily utilized compute through Google Colaboratory. Although our footprint is relatively small, we are cognizant of the fact that training models does have a non-negligible impact on the environment, and thus sought to only train models for the durations necessary to answer our underlying research questions, while training for longer periods of time or on larger datasets only for model configurations we had found to be promising before.

We also consider the societal impacts of utilizing our model once trained. Transformer models such as BERT, as well as the explosion of foundation models in the past 2 years, demonstrates that training powerful models is not to be taken lightly.

For one, a model of this size and discerning capability could be misused, either as a means for more generative purposes - such as to generate hate speech - or for automation that often takes the form of a black box. For example, consider that in our Quora dataset, our model incorrectly classified inputs with relatively specific cultural terms, such as "In what ways can Ganesh chaturthi Festival can be celebrated in a more eco friendly way?" and "What are some of the unique ways of celebrating Holi Festival?" as paraphrases. The model's attention mechanism, though powerful, may pay more attention to well-known terms such as "ways," "Festival," and "celebrat[ed/ing]" when classifying the input, while not realizing that "Holi" and "Ganesh chaturthi" are entirely distinct, since those terms may be less frequent in our datasets. Thus, if deployed on a site such as Quora to identify and remove paraphrases, members of non-English communities may find their posts being removed or flagged incorrectly, exacerbating global digital inequalities. Similarly, for semantic textual similarity, our model gave "UN chief welcomes peaceful presidential elections in Guinea" and "UN chief condemns attack against peacekeepers in Mali" a similarity score of 2.6 instead of 1.0. A classifier making such errors may cause posts of a certain political or geopolitical nature to be mistakenly flagged or recommended to users based on a false notion of similarity, again alienating marginalized groups.

We identify 3 mitigation techniques for this concern. The first is to take a human-centric approach during deployment and to ensure that users have recourse for trust and safety abuses due to the model - for example, being able to get a human to manually review their post. The second is to utilize more diverse datasets - for example, incorporating data labeled or provided by non-English speakers, or other social media platforms worldwide. Lastly, as academic researchers, we can take the time to better understand the models that we build and investigate their specific inputs and outputs, and study whether we can develop any technical approaches towards fairness and explainability in our AI systems.

In doing so, we can strive towards a system in which AI models such as ours can be used for expanding our understanding of computer science and for assisting humans in real life, while minimizing ethical and societal harms.

9 Acknowledgements

I would like to thank my mentor, Aditya Agrawal, as well as the entire CS224N teaching team for their guidance and support throughout this course and in brainstorming for and enhancing this project.

References

Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. MTRec: Multi-task learning over BERT for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669, Dublin, Ireland. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL Anthology*, Online. ACL Anthology.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

PyTorchContributors. 2023a. Cosineembeddingloss.

PyTorchContributors. 2023b. Cosinesimilarity.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Conference on Empirical Methods in Natural Language Processing*.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, Online. Advances in Neural Information Processing Systems.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

A Appendix (optional)

A.1 Datasets

Examples of entries in the datasets can be seen below:

	id	sentence	sentiment
0	32a4f146782cbde1b	The Rock is destined	3
1	6e0fb51a6fefc7c1b6	The gorgeously elab	4
2	9b1d25f272afcc21d4	Singer/composer Br	3
3	40ad03c7e8989942	You 'd think by now /	2

	id	sentence1	sentence2	is_duplicate
321296	a4da64a4e943bf8b0	How can I master my	How can I master ge	0
373075	f965ab2d29d51008a	Can deleted pictures	How do I delete a pi	0
213557	5aca073f4a977d551	Who is the best prime	Who is the best prime	1
209343	a8abbd08d3401287	What qualifications c	What qualifications c	1

	id	sentence1	sentence2	similarity
8615	6d0ec4e8a34ac432	13 killed in Afgan ter	6 killed in Afghan ca	2
7087	9aaabdce8cfff4c060	China to resume US	China and Taiwan h	0.8
8201	a70c8ad3f82ac2967	But in the first 30 sec	But in the first 30 sec	4.2
7821	a89ed469c8f2f8131	A yellow vested pers	A person is doing ver	0.2
201	9ee4d44188c33f4e7	Someone is slicing t	Someone is riding a	0
3538	252ebec695ca7efc2	capital offenses in se	capital offences in ir	3.2

Figure 4: SST, PARA, and STS datasets

A.2 Confusion Matrix for Paraphrase Detection

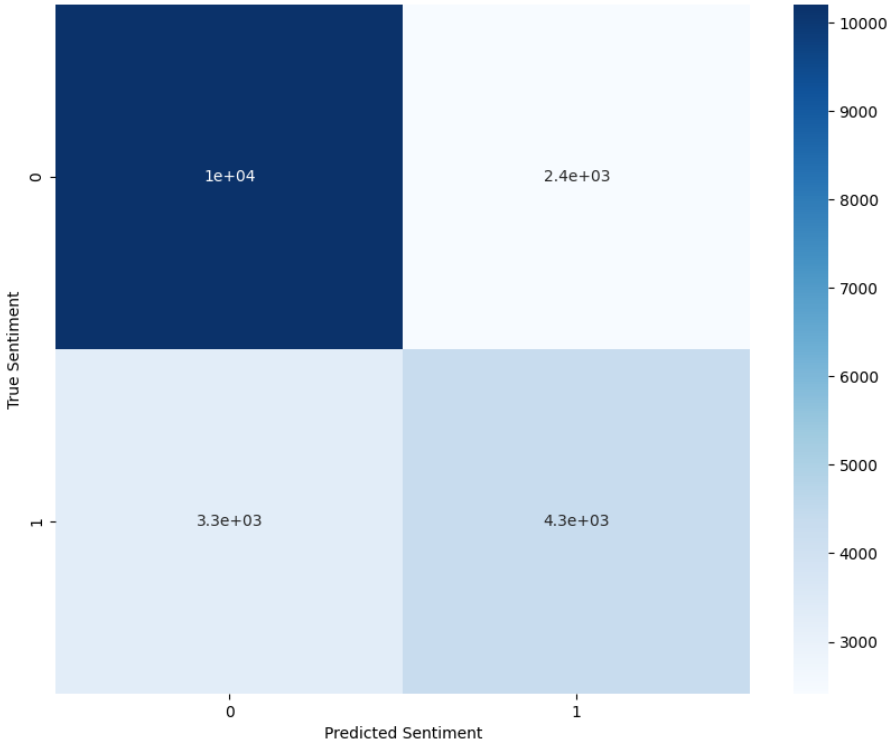

Figure 5: PARA Confusion Matrix.png