Combining Improvements for a Better BERT

Stanford CS224N Default Project

Diego Ahmad-Stein
Department of Computer Science
Stanford University
`dahmad1@stanford.edu`

Emily Wesel
Department of Computer Science
Stanford University
`ewesel@stanford.edu`

Abstract

With the constant churn of new papers being published on novel techniques, there is little academic time dedicated to assessing whether better results can be obtained by combining these strategies. In this paper, we synthesized the results of several of these papers discussing improvements to the baseline BERT strategy to see whether we can train a single BERT model to achieve a greater accuracy on several text classification tasks. We implement additional pretraining, multi-task fine-tuning, and gradient surgery strategies, and run them both separately and together. We found that a round robin, multi-task fine-tuning approach provided a solid foundation that achieved strong results, but that the further augmentations provided only modest improvements, as did the combination of these strategies.

1 Key Information to Include

- Mentor: Shai Limonchik
- External Collaborators (if you have any): N/A
- Sharing project: No

2 Introduction

One of the perennial problems of the natural language processing field is the struggle to create a language model that can answer questions across a variety of fields accurately, rather than training a model to be highly specialized in a single field. Recent large language models such as GPT-3 have made large strides in this area, but they still have areas of weakness. Further improvements in multi-objective learning could help to shore up the weaknesses of these models in the future. In this paper, we implement several strategies that are known to help with multi-objective learning on BERT, a smaller precursor to GPT. Each of these strategies on their own causes an increase in the accuracy of one or more of our learning objectives. The objectives in question were accuracy on sentiment classification, paraphrase detection, and similarity correlation. The techniques included multi-task fine-tuning as discussed in Sun et al. (2020). and using a similarity head as described in Reimers and Gurevych (2019), additional pretraining as outlined in Sun et al. (2020), and gradient surgery as described by Yu et al. (2020). The rest of this paper outlines our work on these approaches and the related works that inspired them, discusses our experiments and results, and analyzes why some of the approaches may have fallen short. We conclude with suggestions for further work.

3 Related Work

3.1 Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

Two of the tasks we were training on involved deciding how similar two sentences were. Reimers and Gurevych (2019)'s paper outlined a useful technique for designing a head to decide on the similarity between two BERT embeddings. If the embedding were *u* and *v*, this approach created the vector {*u*, *v*, *abs(u - v)*} and passed it through a linear layer and a nonlinearity.

3.2 How to Fine-Tune BERT for Text Classification?

This paper by Sun et al. (2020) describes several techniques for improving BERT's performance on mult-task learning for text classification objectives. We focused on two of these in particular. The first, additional pretraining, meant doing more of the same type of training BERT was originally trained with (masked next-token prediction) using in-domain data for our final objectives. The other, multi-task fine-tuning, outlines the strategy of having each task share an underlying model with only a separate classification head for each task.

3.3 Gradient Surgery for Multi-Task Learning

This paper by Yu et al. (2020) explains how in multi-objective learning scenarios, the gradients of a parameter with respect to the losses for the two different tasks can be conflicting in direction, preventing effective progress. To avoid this, they developed a method of gradient surgery, which involves "surgically" removing parts from each of the gradients to project them on to each other's normal planes, allowing the model to take larger steps in a "mutually agreeable" direction while avoid conflict. We re-implemented a simple version of this method as one of our techniques for training our multi-task BERT.

4 Approach

4.1 Baseline BERT

The backbone of our model is a single MultitaskBERT model as described in the Default Final Project Handout. We used AdamW as our optimizer. We used attention heads as described in Vaswani et al. (2017).

4.2 Additional Pretraining

Discussed in Sun et al. (2020), additional pretraining is the concept of training the base BERT model further using the same masked language modelling task with which it was originally trained, but using training data from the target domains rather than general language data. For each of our three target domains, we take a datasets, mask out 15% of the words in each input sentence, and require the BERT model to attempt to infer these words. The model predicts a probability distribution along the entire vocabulary (about 30,000 words). Computing the cross-entropy with the one-hot vector of the actual word gives us the probability assigned to the correct word, from which we compute a loss and update the model.

According to the literature, using in-domain data is most effective, so we chose to reuse our existing datasets from pretraining. We were also influenced by Gururangan et al. (2020), which argues for in-domain pretraining. Although the pretraining loss did not completely plateau, we chose not to continue training, since we were already experiencing overfitting.

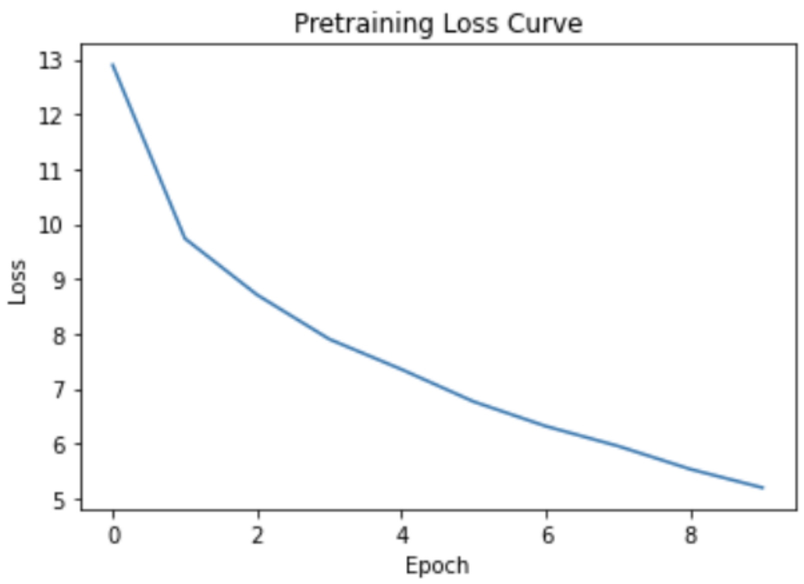

Figure 1: Loss curve during pretraining

4.3 Individual Heads

Our model implements a different head for each of its three objectives.

For sentiment analysis, the [CLS] (end-of-sentence) token embedding that BERT gives for the sentence in question is passed through a dropout layer (for training only) and a linear layer projecting it down to the dimensionality of the number of sentiment classes (five).

For both the paraphrase detection and similarity score heads, the [CLS] token embeddings for each of the two sentences (represented *u* and *v*) were concatenated into a vector {*u*, *v*, *abs(u - v)*}, which was then passed into a dropout layer (for training only) and a linear layer projecting it down to a single logit representing how similar the sentences were.

4.4 Multi-task Fine-tuning

We trained our model (tuning both the parameters of BERT and the heads) on each of our three datasets simultanteously using a round robin approach. Rather than drawing a batch from a single dataset, our training loop drew a batch of the same size from each of the three training datasets, ran forward inference and got a loss for each of them from their respective heads, then summed these to get a total loss. We then ran a backwards pass and and gradient update step on this combined loss. Since the datasets were vastly different sizes, an epoch was defined by one pass through the largest dataset. During one of these epochs, when a smaller dataset was exhausted, we would simply loop back to the start of that set.

4.5 Gradient Surgery

To address the fact that these gradients could be in conflicting directions, we employed gradient surgery to alter each of the gradient components so that they conflicted less. Specifically, for each pair of gradients $\mathbf{g}_i$ and $\mathbf{g}_j$ that were already conflicting (i.e. $\mathbf{g}_i \cdot \mathbf{g}_j < 0$), we would set

$$\mathbf{g}_i = \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{||\mathbf{g}_j||^2}\mathbf{g}_j$$

, an update which projects $\mathbf{g}_i$ onto the normal plane of $\mathbf{g}_j$. The class handling this operation wraps around our regular optimizer (AdamW) and intercepts backwards() calls to update the gradients on each parameter in this manner. We implemented our own simplified version of the a "gradient surgeon" after studying Yu et al. (2020)'s work.

4.6 Hyperparameter Tuning

As is common in training a language model with a limited dataset, we encountered the issue of overfitting. Our model quickly memorized the sentiment classification dataset and got near-perfect accuracy on the training set, but accuracy for the dev set remained low. We explored several mitigations for this issue: increasing the dropout rate, increasing batch size, and decreasing the learning rate. Although the loss curve does not completely plateau, we decided not to increase the number of epochs as we believed that this would exacerbate overfitting. Increasing the batch size allows the model to train on a more balanced dataset during each iteration, making it less likely for the model to encounter a batch impacted by a bottleneck effect that caused noise in a certain direction. Additionally, decreasing the learning rate makes the model less sensitive to new data it encounters, hopefully making it more robust to noise.

Our model employs dropout, randomly abandoning hidden nodes to ensure that neurons learn to represent individually useful information rather than being heavily codependent on other nodes in their layer and makes them less sensitive to noise. An appropriate dropout strikes a middle ground between a naive Bayesian approach, where all features are independent, and logistic regression, where all features are interdependent. Our STS and SST datasets were much smaller than paraphrase, making dropout a necessity to prevent severe overfitting. Of course, excessive dropout must also be avoided, lest no signal make it through the model.

5 Experiments

5.1 Data

Dataset	Possible Outputs	Type	Train Samples	Dev Samples	Test Samples
Stanford Sentiment Treebank	5	Sentiment	8,544	,101	2,210
Quora Dataset	2	Paraphrase	141,506	20,215	40,431
SemEval STS Benchmark Dataset	[0.0, 5.0]	Regression	6,041	864	1,726

All these datasets and their associated tasks are well documented in the final project handout, but we will reproduce a brief description here for convenience.

5.1.1 Stanford Sentiment Treebank

This dataset has sentences from movie reviews, labeled with five labels ranging from negative to positive. The model takes in one of these sentences and outputs five logits corresponding to its belief that each of these five labels is the correct one for this sentence.

5.1.2 Quora Dataset

This dataset has pairs of Quora question titles, and labels each of them as paraphrases or not paraphrases of one another. The model takes in two of these pairs and outputs a similarity score corresponding to its belief that these two questions are paraphrases.

5.1.3 SemEval STS Benchmark Dataset

This dataset has pairs with semantic similarity ranging on a scale from 0 (unrelated) to 5 (equivalent meaning). The model takes in one of these pairs and outputs a logit corresponding to how similar it believes they are.

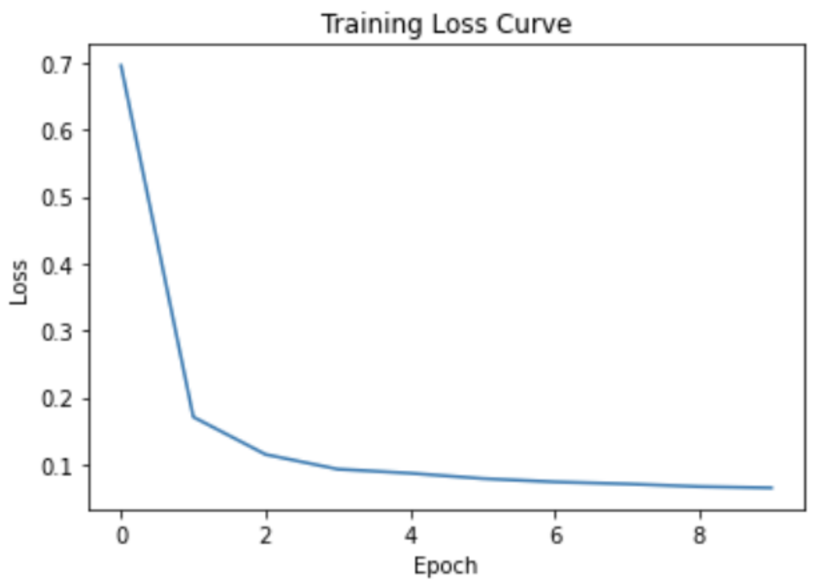

Figure 2: Loss curve during training

5.2 Evaluation method

We evaluated each improvement we attempted in terms of its effect on the accuracy scores achieved on the dev set for each of these three datasets after 10 epochs of training. Our "bare minimum" baseline statistics are those generated by a run trained only on Stanford Sentiment Treebank Data.

5.3 Experimental details

Our experiments were run in fine-tune mode (meaning the parameters of the BERT model were able to be altered). Learning rate and batch size were 1e-5 and 8 respectively, except in the "hyperparameter tuning" experiment, in which they were changed (to 5e-6 and 16). On an 8 vCPU GPU, training for 10 epochs typically took about 10 hours.

Since the loss converged, we did not train beyond ten epochs.

5.4 Results

Test	SST Dev Accuracy	Paraphrase Dev Accuracy	STS Dev Accuracy	Overall Sco
Baseline	.416	.376	.019	0.271
Three Heads, Round-Robin (3HRR)	.485	**.853**	.635	.658
3HRR + Additional Pretraining (AP)	**.509**	.815	**.677**	**.667**
3HRR + Gradient Surgery (GS)	.485	.840	.660	.662
3HRR + AP + GS	.469	.844	.650	.654
Tuned Hyperparameters	.506	.796	.632	.645

Dev leaderboard results: {SST dev: .509, para dev: .815, STS dev: .677, Overall dev: .667}
Test leaderboard results: {SST test: .521, para test: .812, STS test: .635, Overall test: .656}

As evidenced above, the implementation of a simple round robin approach to multi-task learning already led to significant improvements. From there, adding additional pretraining and gradient surgery separately each improved some accuracies by a few points, while causing slight declines in others. Both resulted in an increase in total accuracy, though the effect was small (between 0.5%-1% for each).

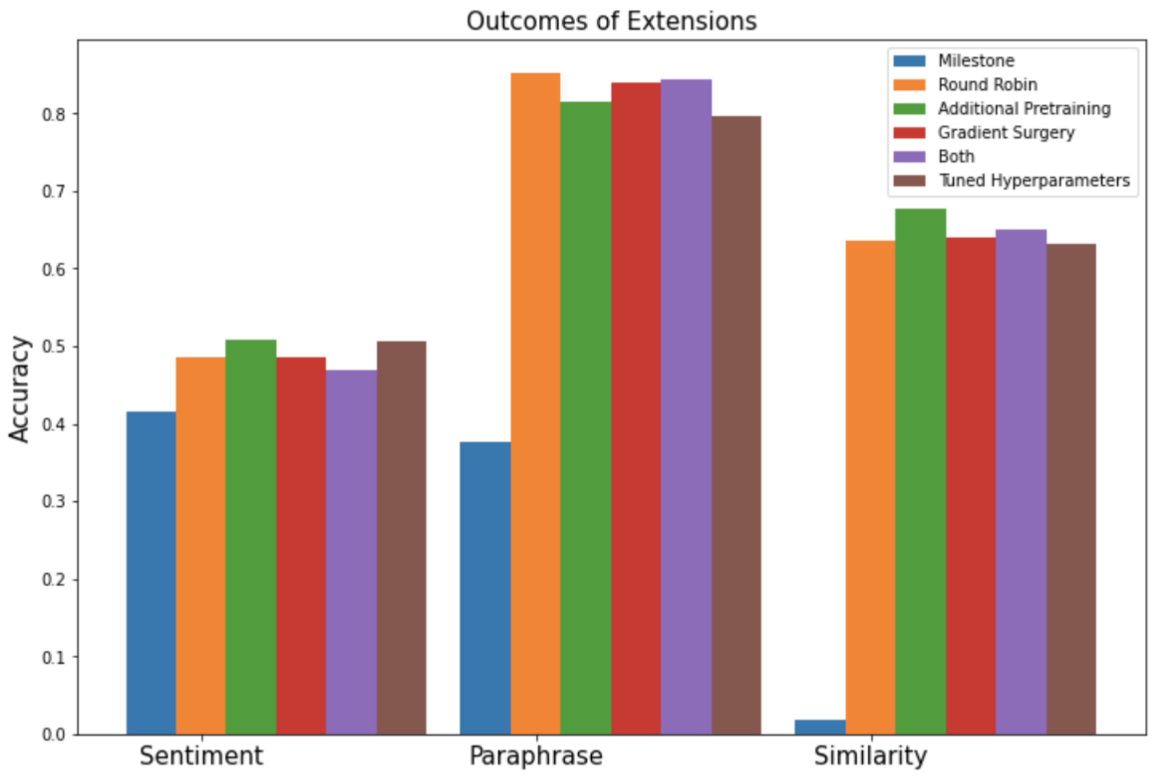

Figure 3: Accuracies Across Various Models

Disappointingly, the combination of these two augmentations into one final model seemed to adopt the weaknesses of each rather than the strengths, and resulted in a model with worse performance than either of its two constituent parts.

Overall, the improvements for individual changes were somewhat smaller than we expected, and the combination of approaches did not result in improvements. As we discuss further in the upcoming analysis section, it is possible that the conditions for these additions to be helpful were not met, or that the additions did not address the actual limiting factors holding the model back from a higher score.

6 Analysis

6.1 Overfitting and Insufficient Data

One phenomenon our model was experiencing was overfitting. Throughout the training of various models, accuracy on the SST and STS train datasets would frequently go to >99%. This is likely due to the way we were handling the different sizes of the datasets. Since we would loop back to the beginning of a smaller set when it ran out, we would iterate over the entire SST dataset 16.5 times for every epoch and the STS set 23.4 times for every epoch. We explored an alternative approach to correct this where we loop through each dataset once per epoch, taking bigger batches from the bigger sets to keep the pace identical. However, this was untenable because the batch size on the Quora dataset would be very large, and long inputs (which force the whole input batch to have their max sequence length) would run out of GPU RAM.

6.2 Further Pretraining

We employed further within-task pretraining. Our finding agree with Sun et al. (2020), that within-task pretraining can improve accuracy by about one percentage point. Sun et al. (2020) found that generally, within-domain pretraining was more effective, but this result broke down in the case of attempting to pretrain movie reviews with a dataset of restaurant reviews. Understanding when and why within-domain outperforms within-task pretraining is an area of active research.

6.3 Gradient Surgery

The addition of gradient surgery was responsible for a roughly .5% increase in the overall score relative to our standard round robin multi-task fine-tuning approach. While a non-negligible improvement, this was not the significant change we were hoping for. Yu et al. (2020) describes the three conditions necessary for gradient surgery to have a significant impact. They are: the gradients for the different tasks conflict, there is a large difference in the magnitudes of the gradients, and the curvature of the optimization landscape is high. Of these, we believe that the second did not always hold in our case. The model trained in each of the tasks at roughly equal rates, and our inspection of some gradients for each problem during development showed that they appeared to be on the same order of magnitude. This may explain why the addition of gradient surgery to our base approach did not result in a significant improvements in the overall score.

6.4 The [CLS] Token Bottleneck and Single-Layer Heads

Each of our three heads bases its analysis on the [CLS] token generated at the very end of the BERT embedding of a sentence. While this will encode information from the whole sentence to some extent, it does represent an inherent bottleneck of information flow between the model and the head. Perhaps by making better use of the full sentence embedding generated by BERT, by averaging them for example, we could have garnered more information for use in our classification heads. These heads were also relatively simple, consisting only of a single linear layer to project the [CLS] token down to a result. A deeper, more complex head may also have been necessary to extract more information from the BERT embeddings.

6.5 Tuning Hyperparameters

Ultimately, our idea of increasing batch size, decreasing learning rate, and increasing dropout rate did not increase the accuracy on the dev set. We tried to infer what hyperparameters might be successful based on inspection of early epochs, but the 12-hour training time necessary for each experiment prevented us from extensively exploring the hyperparameter space. However, we believe that further research in this area with greater time and compute resources could still yield improvements.

7 Conclusion

Our round robin multi-task learning strategy achieves strong performance on each of the three objectives we set out to learn. However, while each of our added optimizations tweaked the accuracy of each different task somewhat, we did not find any of them to represent a significant overall improvement over a straightforward approach using round robin multi-task fine-tuning. It is possible that we were simply running up against a natural limit of how much information our single-layer classification heads can extract from a BERT model that has to maintain representations general enough to work for all three tasks. Time and compute constraints limited us from doing further experimentation to solve these problems, but further areas of research might include more complex classification heads to extract more data from the generated BERT embeddings, or heads which looked at all generated BERT embedding tokens for a sentence, rather than the limited representation of the full sentence state found in the [CLS] token.

References

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification?

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning.