Multitask BERT

Stanford CS224N Default Project

Caroline Zanze
Department of Computer Science
Stanford University
ckzanze@stanford.edu

Drew Wadsworth
Department of Computer Science
Stanford University
swads@stanford.edu

Abstract

In this project, we implement a vanilla minBERT to generate embeddings, and then expand upon this model to simultaneously perform three tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. First, we created a model which individually trains each of the three tasks. Then, we implemented true multi-task learning by summing the three tasks' loss functions for the minBERT portion of our architecture. Lastly, we used gradient surgery atop this multi-task architecture to address the problem of conflicting gradients. Ultimately, we were able to create a model which achieved 49.9% accuracy on the 5-class sentiment analysis task, 70.0% accuracy on the binary classification task of paraphrase detection, and a Pearson correlation of 0.352 on semantic textual similarity.

1 Introduction

Multitask learning can boost efficiency due to its ability to share parameters, and, in theory, produces a more general linguistic model. The goal of our project was to explore and evaluate a variety of multi-task learning methods for the three tasks of sentiment analysis, paraphrase detection, and semantic textual similarity. We started by implementing a model which simultaneously learns and performs the three tasks but does not share any layers. Then, we moved on to fine-tuning and sharing BERT between the three tasks, and, finally, we implemented gradient surgery, which theoretically can improve multi-task learning by resolving gradient conflicts.

Discussing these three approaches is the main goal of this report, but we also implemented weight decay regularization, experimented with multiple approaches for addressing dataset imbalance, and tried both subtracting and concatenating the inputs for the paraphrase and semantic textual similarity tasks.

2 Related Work

Bi, et al. use multi-task learning on top of BERT for three tasks in the domain of news recommendation [1]. Their 3-task multitask architecture is similar to ours in that they use BERT as the transfer learning model, and they learn 3 related tasks. Bi, et al. compute a multitask loss M_{MTRec} by summing the individual task losses, as follows: $\mathcal{L}_{MTRec} = \mathcal{L}_{Main} + \mathcal{L}_{category} + \mathcal{L}_{NER}$, where Main, category, and NER are the three tasks learned by Bi, et al. We use this loss-summing approach for our multitask learning experiments.

Yu, et al. introduce the idea of Gradient Surgery, which, for any two gradients that are conflicting, where conflicting is defined as having an angle between the gradients of greater than $\pi/2$ radians, one of the gradients (randomly chosen) is projected onto the normal plane of the other gradient [2]. This is designed to improve learning speed and overall model accuracy. We use this approach in some of our multitask learning experiments. Yu et al. evaluated gradient surgery on five multi-tasked supervised learning datasets: MultiMNIST, CityScapes, CelebA, multi-task CIFAR-100 and NYUv2.

Notably, although we are implementing the same methods as Bi et al. and Yu at al., our approach is—as far as we are aware—novel in tasks we apply them to; there is no overlapping dataset use.

Loshchilov, et al. introduce a mechanism of regularization called decoupled weight decay, which differentiates weight decay from L2 regularization when using an adaptive optimizer such as Adam. We implement decoupled weight decay in our AdamW implementation and utilize it to regularize in some of our experiments [3]

3 Approach

We conducted a variety of experiments to iteratively expand upon the vanilla minBERT model.

First, we implemented the most basic type of "multitask" model, which consists of three simultaneous, yet entirely independent models for the three tasks, with no weight sharing. That is, this initial model has one BERT model and one final linear layer per task. For the sentiment classification task, we use dropout and a linear layer that takes BERT embeddings as input and outputs 5 logits. For the paraphrase task, we use dropout and a linear layer that takes as input the concatenated embeddings of the two input sentences and outputs a single binary logit. For the similarity task, we use dropout and a linear layer that takes as input the concatenated embeddings of the two input sentences and outputs a single continuous logit that indicates how similar the sentences are. We had initially subtracted the embeddings before feeding them into the paraphrase and semantic textual similarity heads, but we found that performance on semantic textual similarity was far better with concatenation. Our idea behind subtracting the embeddings is that we are interested in the difference between two sentences for both paraphrase and semantic textual similarity. However, concatenating the two embeddings before feeding them to the linear layer produced a more accurate model, so we chose to concatenate the embeddings for the remainder of our experiments. We train this initial model in both "pretrain" and "finetune" modes, where "pretrain" mode only trains the last linear layer of each of the models, and "finetune" mode trains the entire model, including minBERT. For this experiment, our baseline for comparison is random guessing for each of the tasks. The breakdown of this baseline is as follows: sentiment classification accuracy: 0.2, paraphrase detection accuracy: 0.5, semantic textual similarity correlation: 0.0.

For all following experiments, our baseline is the initial model described above. We train all of the following models in finetune mode and with concatenation rather than subtraction.

The second experiment consists of an architecture where BERT is shared between the three tasks, and the final linear layers are not shared between the models. The multitask loss is computed by summing the losses for a single batch of each task. The idea behind this model is that word embeddings should intuitively be general—they should not need to be tailored to different tasks. Therefore, we can finetune the BERT model on all tasks and hopefully achieve a better result than using three separate models . At the very least, we should achieve a similar level of accuracy with a much smaller model than using separate BERT models across tasks. The largest issue encountered with this model is that the number of batches per epoch is different for the different tasks, making it hard to define a multitask batch over which to compute a multitask loss. We initially solved this issue by upsampling the smaller datasets such that the number of batches per epoch were equivalent across the tasks. This functionally works, but it takes a very long time to train due to the inflated amount of training data.

The third experiment attempts to reduce both training time by upsampling the smaller datasets by no more than 12 times their original size and downsampling the largest dataset—Quora—by around 10 times its original size. Originally, we sampled a tenth of the Quora dataset and used that same sample on each epoch, but then we thought of a better way to optimize the need for a smaller amount of Quora data per epoch and the benefits of exposing the model to a large range of data: on each epoch, we select a different tenth of the Quora dataset to train on.

The fourth experiment is designed to improve the quality of the multitask BERT embeddings by applying the idea of gradient surgery. As described by Yu, et al., many multitask models suffer from the problem of gradients that point in different directions, causing the summed gradient to point in a suboptimal direction [2]. Therefore, as Yu, et al. propose, if the gradients from the two tasks point in opposing directions — that is, if the angle between the gradients is greater than $\pi/2$, it can be beneficial to project one of the conflicting gradients onto the vector subspace that is orthogonal to the other gradient. This "gradient surgery" removes the conflicting component of one of the vectors. We

incorporate an existing, general implementation of gradient surgery authored by Wei-Cheng Tseng into our model [4].

The fifth experiment adds regularization (beyond dropout, which we have used throughout) to the model in an attempt to reduce overfitting. Our implementation of the AdamW optimizer for minBERT implements decoupled weight decay, a simple but powerful regularization technique [3].

4 Experiments

4.1 Data

After implementing minBERT, we trained and evaluated it on sentiment analysis using two datasets: Stanford Sentiment Treebank (SST), and CFIMDB. SST contains 215,154 unique phrases from 11,855 single sentences from movie reviews. Each phrase is labeled as negative, somewhat negative, neutral, somewhat positive, or positive [5]. Thus, sentiment analysis is a multi-class classification problem with respect to the SST dataset. CFIMDB contains 2,434 polarized movie reviews which are labeled with a binary positive or negative [6].

We extended minBERT to simultaneously perform sentiment analysis, paraphrase detection, and semantic textual similarity. For the sentiment analysis task, we use the SST dataset described above.

We used the Quora dataset for the paraphrase detection task, which is the binary classification problem in which, given two input sentences, we determine whether the second sentence is a paraphrase of the first. The Quora dataset contains 400,000 question pairs from quora.com, labeled with whether different questions are paraphrases of each other [7].

Finally, we use the SemEval STS Benchmark Dataset for the semantic textual similarity objective. Given 2 input sentences, the objective is to predict the similarity of the sentences on a continuous scale from 0 (not at all related) to 5 (same meaning). The STS dataset contains 8,628 labeled sentence pairs [8].

4.2 Evaluation method

For the sentiment analysis and paraphrase detection tasks, we evaluated our model with a simple accuracy-based metric: $\frac{Q_{correct}}{Q_{total}}$, the fraction of the total queries that are classed correctly.

For the semantic textual similarity task, we calculated the Pearson correlation of the true similarity values against our predicted similarity values [8].

4.3 Experimental details

The model details for the main experiments are described in the Approach section above. For each of the experiments, we trained the model for 10 epochs, with a batch size of 64 for the sentiment classification task, with a batch size of 8 for the paraphrase detection task, and with a batch size of 8 for the semantic textual similarity task. For the majority of the experiments, the model was run in finetune mode, as it became clear early on that finetuning BERT results in better performance. In finetune mode, the learning rate used was always 1×10^{-5} (whereas we used 0.001 for pretrain mode). Dropout is used for the final layers for all tasks, with dropout probability = 0.3.

4.4 Results

Figure 1 shows 9 of our models which we trained and evaluated on the development datasets. The green section (Models 1-3) constitutes the "separate" models which do not share any layers. Models 1 and 2 concatenate the two embeddings output by BERT for the paraphrase and semantic textual similarity tasks before feeding them to the rest of the model (model 3 subtracts the embeddings), and models 2 and 3 fine-tune BERT (model 1 simply uses the pre-trained BERT). The fine-tuned models performed best, which is unsurprising because fine-tuning adjusts the BERT embeddings to specifically serve the given task. Model 2, which uses concatenation, performed significantly better than Model 3, which uses subtraction, on the paraphrase detection task. These two models performed essentially the same on semantic analysis. Thus, we decided to use concatenation and fine-tuning for the remainder of our experiments.

Model	Sentiment Analysis - Accuracy		Paraphrase Detection - Accuracy		Semantic Textual Similarity - Pearson Correlation	
	Train	Dev	Train	Dev	Train	Dev
1. Separate, pre-train, concat	0.301	0.291	0.655	0.648	0.155	0.171
2. Separate, fine-tune, concat	0.977	0.507	0.997	**0.786**	0.963	0.358
3. Separate, fine-tune, subtract	0.990	**0.518**	0.990	0.602	0.875	0.013
4. Shared BERT, fine-tune, concat, upsample	1.000	0.510	0.986	0.780	0.978	**0.371**
5. Shared BERT, fine-tune, concat, downsample	1.000	0.500	0.958	0.699	0.967	0.366
6. Shared BERT, fine-tune, concat, sample para each epoch	0.998	0.510	0.787	0.760	0.982	0.367
7. Grad surgery, fine-tune, concat, sample para each epoch, no weight decay	0.998	0.501	0.787	0.707	0.982	0.326
8. Grad surgery, fine-tune, concat, sample para each epoch, weight decay = 0.001	0.998	0.483	0.787	0.759	0.982	0.367
9. Grad surgery, fine-tune, concat, sample para each epoch, weight decay = 0.05	0.996	0.496	0.788	0.761	0.982	0.369

Figure 1: Train and dev performance across 9 models

The blue section (Models 4-6) consists of the multi-task models which share weights during the BERT portion of the architecture. Model 4 upsamples the SST and STS datasets so that they can have the same number of batches as the Quora dataset, which is much larger. Model 5 still upsamples these datasets, but to a lesser degree, and it also downsamples the Quora dataset. Lastly, Model 6 upsamples SST and STS and draws a new sample of data from Quora on each epoch. Models 4, 5, and 6 perform similarly on the sentiment analysis and semantic textual similarity tasks. Model 4 performs best on paraphrase detection, with an accuracy of 78.0%. Model 6 performs next best, with an accuracy of 76.0%, and model 5 performs worst, with an accuracy of 69.9%. It makes sense that model 5 performed worst because it sees significantly less of the total training Quora dataset. We would have expected Model 6 to do worse than Model 4 because Model 4 sees all of the Quora train data on each epoch.

Model 6 is notably not overfit—there is still more room for training. Of course, if we were to train for longer, we would eliminate the point of model 6, which was to save time training. In the future, we would like to increase the learning rate to see how much better Model 6 can perform without overfitting the train data. Although Model 4 performed best, we implemented our gradient surgery experiments with the Model 6 technique of sampling a new chunk of Quora. We did this because the technique of purely upsampling was simply impractical, as it took 24 hours to train our model.

Gradient surgery constituted our final set of experiments (Models 7-9). Model 7 uses no weight decay regularization, Model 8 uses a weight decay of 0.001, and Model 9 uses a weight decay of 0.05. Comparing Model 7 to Model 6, we see that gradient surgery was not helpful; it actually very slightly hurt our performance on the paraphrase and semantic textual similarity tasks. This may be because the gradient directions are, in general, in relatively similar directions, so the normal plane projection may happen infrequently. Another explanation is that the gradient projections are sometimes obscuring useful information. In models 8 and 9, we see that regularization minimally improved upon the non-regularized Model 7 for the paraphrase and semantic textual similarity tasks.

Future steps include attempting other weight decay and dropout rate values as well as other types of regularization, such as Smoothness-Inducing Adversarial Regularization [9].

The best performances per task are bolded. Shared BERT with upsampling (Model 6) performs best on the semantic textual similarity task, although the performances are very similar across all models except the separate subtraction model (Model 3), which performs very poorly. Performances on sentiment analysis and paraphrase detection are also quite similar across many models, but the best scores come from Models 3 and 2, respectively.

We evaluated Model 6 on the test data and achieved results on par with our dev set performance (see Figure 2 below).

Sentiment Analysis - Accuracy	Paraphrase Detection - Accuracy	Semantic Textual Similarity - Pearson Correlation
0.499	0.700	0.352

Figure 2: Model 6's performance on the test datasets

5 Analysis

We will now take a fine-grained look at the performance of model 6—the fine-tuned, shared BERT model in which we randomly sampled data from the paraphrase dataset each epoch. Despite the best results for the sentiment analysis and paraphrase detection accuracies being highest in the "separate" models, we analyze model 6 because 1) the impetus of our project was to implement and better understand multitask learning, and 2) Model 6 does achieve the best score on semantic textual similarity.

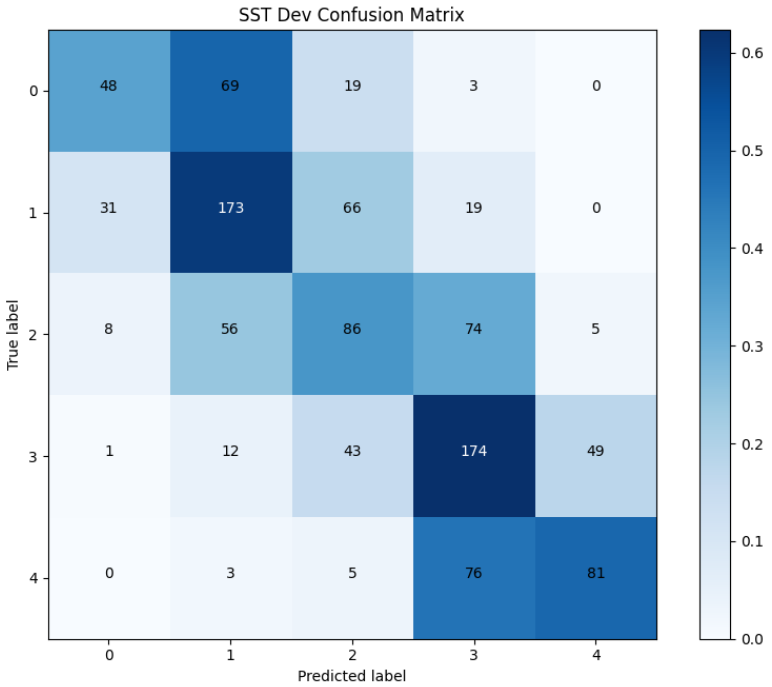

Figure 3: Confusion matrix of Model 6's sentiment analysis performance on the development SST dataset

Figure 3 shows the confusion matrix for the paraphrase detection task on the dev dataset. We see from the confusion matrix that the positive class accuracy (84.4%) is much higher than the negative class accuracy (67.5%). The model is somewhat biased toward the positive class, which makes sense because there are more negative examples in the training data, so the loss is more affected by this class. If we had more time, we would have explicitly mitigated this bias by implementing a weighted cross-entropy loss or modifying our upsampling technique to only upsample from the positive class.

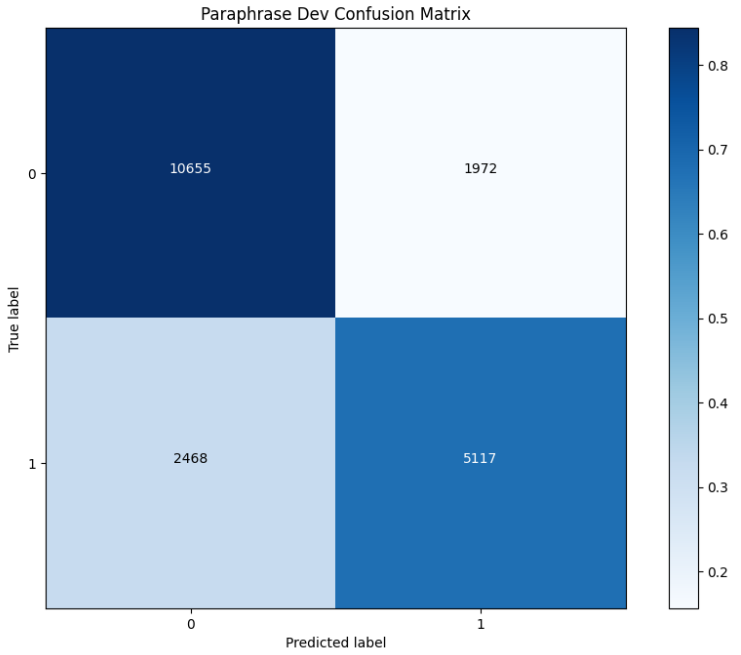

Figure 4: Confusion matrix of Model 6's paraphrase detection performance on the development Quora dataset

Figure 4 shows the confusion matrix for the sentiment analysis task on the dev dataset. The per class accuracies (in order from class 0 to class 4) are 34.5%, 59.9%, 37.6%, 62.4%, and 49.1%. It makes sense that the class 2 accuracy is relatively low, since it is the "neutral" sentiment class, which intuitively has the most room for ambiguity. It is somewhat surprising that the 0 and 4 classes do not perform the best, for intuitively it seems easiest to detect the extreme cases. The superior performances of classes 1 and 3 may have to do with class imbalance and/or many of the true class 2 examples being categorized one class off. It is promising that, for each class, the densest category aside from the true class is an adjacent class.

Lastly, we have the semantic textual similarity scatter plot in figure 5. The general shape of the plot and the line of best fit show that there is a positive correlation between the labels and the predictions. Interestingly, the graph looks pretty regular; the model does not appear to have that much harder of a time assigning a similarity score to one type of sentence pair over another (i.e. it does not perform much better on more negatively related sentences than positively related ones).

6 Conclusion

We ultimately achieved a model that performs multitask learning that is on par with individually fine-tuning on each task of the sentence-analysis tasks. To do so, we learned about and implemented a variety of techniques, and we also developed our own technique of sampling different parts of the Quora dataset per epoch to address the dataset imbalance.

There is ample room for future work, which has been mentioned throughout this report. Most importantly, our model overfits on the training data despite our implementation of AdamW weight decay. Therefore, we could try additional regularization methods like Smoothness-Inducing Regularization

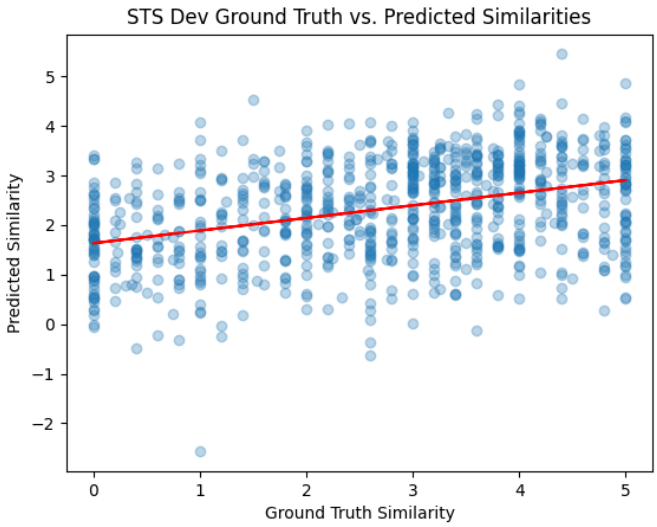

Figure 5: Scatter plot of Model 6's semantic textual similarity performance on the development SemEval STS Benchmark dataset

and experiment with adjusting our dropout rate. Another avenue of future work would be modifying our architecture after the BERT portion of our model; we would add more layers and experiment with sharing some of them.

References

[1] Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. Mtrec: Multi-task learning over bert for news recommendation. 2022.

[2] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea. Finn. Gradient surgery for multi-task learning. In *dvances in Neural Information Processing Systems*, 2020.

[3] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. 2017.

[4] Wei-Cheng Tseng. Weichengtseng/pytorch-pcgrad, 2020.

[5] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, and Christopher potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013.

[6] Chris Manning and 224N teaching staff. Cs 224n: Default final project: minbert and downstream tasks. 2023.

[7] Kornél Csernai, Shankar Iyer, and Nikhil Dandekar. First quora dataset release: Question pairs., 2017.

[8] Eneko Agirre, Diab Mona Cer, Daniel, Aitor Gonzalez-Agirre, and Weiwei. Guo. *sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, 2013.

[9] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiadong Liu, Jianfeng Gao, and Tuo. Zhao. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *arXiv*, 2021.