Multitasking with BERT

Stanford CS224N { Default} Project

Jiacheng Hu
Department of Computer Science
Stanford University
jchu0822@stanford.edu

Jack Hung
Department of Computer Science
Stanford University
jjhung66@stanford.edu

Abstract

Models take a lot of resources to train. Thus it is often desirable to train one single model that can perform a variety of tasks, which is what we are trying to achieve on this default final project. Our goal is to implement a BERT model that is able to multitask: after training, this one single model would be able to perform sentiment analysis, paraphrase detection, and semantic textual similarity. After finishing implementing a minBERT model and the Adam optimizer, we achieve this multi-task goal by implementing a multi-task finetuning framework, which optimizes the model over all three different tasks at the same time instead of one after another. To avoid clashing between objectives, we implemented gradient surgery on our Adam optimizer and explored whether assigning different weights to objectives could lead to a potential increase in model performance. The answer we get from this particular model is no. We take full advantage of the Sentence-BERT framework for paraphrase detection and semantic textual similarity tasks, which captures deeper information and yields a much higher model performance than the conventional concatenation approach. We also propose using a uniform loss function across multi-task learning, and we discovered that this provides uniformity to the losses of distinct objectives and empirically yields better performance on the multitask-fine tuned model.

1 Key Information to include

- Mentor: Cathy Yang
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

In this paper, we tackle the challenge of building a multitask learning model which fine tunes on top of BERT in order to perform on 3 key tasks: sentiment classification, paraphrase detection, and semantic textual similarity. The main goal of our project is to implement a handful of potential improvements to our baseline model, which directly uses BERT representations without additional fine tuning, in order to achieve significant increases in the overall performance.

While there are various challenges to this task, we narrowed down our focus to a handful which we attempted to address:

1. **Gradient updates to the same model from different loss functions can often clash**: to combat this, we use a technique called gradient surgery proposed by Yu et al. which has potential to reconcile these differences.
2. **Favored performance towards certain tasks over others:** The ultimate goal of our project is to improve our model's scores on all three main tasks. This means that improvement to

any task is beneficial, ideally no one task outshines the rest. To combat this imbalance, we modify gradient surgery to prioritize some gradient updates over others so we can shift and balance the improvements, an approach proposed by Bi et al. in order to train a "main" task and an "auxiliary" (less important) task.

3. **Prediction Functions for Different Tasks:** The three tasks that our model needs to handle, being sentiment analysis (one sentence input, one output score from 1-5), paraphrase detection (two sentences input, binary output), and semantic textual similarity (two sentences input, one similarity score as output), vary in input and output formats quite drastically. This means that the prediction functions for different tasks cannot be defined uniformly across all three tasks. Furthermore, as this is one single model performing three different tasks at the same time, we need to consider whether the objectives we set up eventually clash with each other, as this would greatly hinder the performance of the model.

4. **Loss Functions for Multitask Fine-tuning:** Aside from the gradient surgery, we are, in short, adding the loss functions of the three gradients from different objectives together, and optimizing the model from the summed loss as a whole. On paper this sounds promising, but can get troublesome in implementation: when there is a single task, the optimizer trains the model with the goal to minimize the loss, which should lead to an increase in performance of that given task; but with three losses added together and being optimized, it could cause uneven weighting, as there is no guarantee that multiple gradients contribute to the total loss with the extent to improve on the performance of each task. Thus specifically for our goal to finetune the model with multitask learning, it is crucial to maintain uniformity across different loss functions.

3 Related Work

Bidirectional Encoder Representations from Transformers (BERT) by Devlin et al. (2018). is a technique for creating language representations through pre-training on massive corpuses. The large amount of data used means that the word embeddings are incredibly robust, giving it great promise in being able to "understand" the meaning of words in a text and thus making it a good candidate for use in various downstream applications when used in conjunction with fine-tuning. Our project explores this concept, applying various fine-tuning techniques to build a representation on top of the baseline BERT model that can perform decently on a selected set of tasks.

Gradient surgery is a technique proposed by Yu et al. (2020). in which during gradient updates, the gradient of the i-th task g_i is projected to the normal plane of some conflicting gradient g_j. In the original paper, Yu et al. claims that such manipulation of gradients improves the performance of models during multi-task learning, as the method can alleviate conflicts in gradient updates to the same parameter which can often be the case with unrelated tasks. We apply this exact technique via the PCGrad optimizer written by Tseng (2020) to produce a successful improvement to our baseline model.

The Sentence BERT or SBERT by Reimers and Gurevych (2019) is a modified BERT network which uses Siamese network structures to improve performance on semantic similarity tasks. In their paper, Reimers and Gurevych propose that two BERT networks should be used to encode pairs of sentences, after which cosine similarity is applied to the two embeddings for tasks like semantic similarity. Additionally, they demonstrate that SBERT can be more widely used in other downstream NLP applications. We take inspiration from SBERT's architecture and adapt it to multitask learning, particularly drawing from their classification objective function and using a similar equation in our paraphrase detection and semantic similarity tasks.

4 Approach

1. Multi-task Learning with Gradient Surgery: As mentioned in the related work, Yu et al. (2020)'s gradient surgery relies on projecting conflicting gradients, so that the gradient of the i-th task g_i is projected to the normal plane of some conflicting gradient g_j:

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

Much richer details of gradient surgery and why it performs well on a theoretical level are covered in the original paper. The pseudocode of the PCgrad update is illustrated below:

Algorithm 1 PCGrad Update Rule

Require: Model parameters θ, task minibatch $\mathcal{B} = \{\mathcal{T}_k\}$

$\mathbf{g}_k \leftarrow \nabla_\theta \mathcal{L}_k(\theta) \ \ \forall k$

$\mathbf{g}_k^{\text{PC}} \leftarrow \mathbf{g}_k \ \ \forall k$

for $\mathcal{T}_i \in \mathcal{B}$ **do**

  for $\mathcal{T}_j \overset{\text{uniformly}}{\sim} \mathcal{B} \setminus \mathcal{T}_i$ in random order **do**

    if $\mathbf{g}_i^{\text{PC}} \cdot \mathbf{g}_j < 0$ **then**

      // Subtract the projection of $\mathbf{g}_i^{\text{PC}}$ *onto* $\mathbf{g}_j$

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

return update $\Delta\theta = \mathbf{g}^{\text{PC}} = \sum_i \mathbf{g}_i^{\text{PC}}$

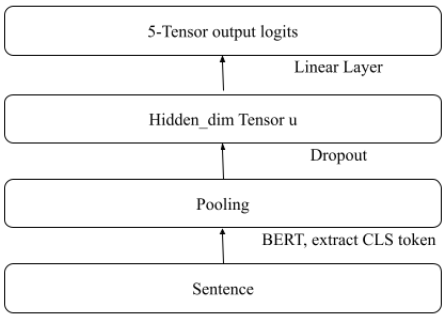
Fortunately, there are complete implementations of PCgrad updating rules online written by WeiChengTseng (Tseng, 2020). We used this implementation in our training, renaming the original pcGrad class by Tseng as MGrad in our multitask_optimizer.py file, which is just a copy of Tseng's implementation. As this is proven theoretically from the original paper as a robust technique that would not hinder the performance of multitask training, we applied the pcGrad update rule on our Adam optimizer for all other experiments.

2. **Main + Auxiliary Tasks:** In Bi et al. (2022) paper MTRecc: Multi-task Learning over BERT for News Recommendation, Bi et al. propose a modified version of gradient surgery by introducing the concepts of main and auxiliary tasks: the gradients of the auxiliary tasks are added and applied a tunable hyperparameter weight λ, then feeded with the gradient of the main task to perform gradient surgery. Specifically, we declare sentiment analysis as our main task and the other two as auxiliary tasks. That being said, we claim that paraphrase and semantic textual similarity are "helping" tasks that will help improve on sentiment analysis, because making the computer learn paraphrases and textual similarities will logically improve its efficacy in classifying sentiments. The equation of our auxiliary task computation goes as follows:

$$g_{aux} = \lambda(g_{para} + g_{sts})$$

In our implementation, we added two new arguments in the argument parser: loss_type and lbd. Declaring loss_type as "main+aux" will apply the modified gradient surgery as shown in the equation above with $\lambda = args.lbd.$ λ is empirically set to 0.3, as declared in Bi et al.'s original paper.

3. **Sentiment Classification:** We keep the neural framework of sentiment classification to be relatively simple, not giving it any modifications from the given code that finetunes sentiment classification alone. The neural network for sentiment classification goes as such:



A cross-entropy loss is computed between the yielded 5-Tensor logits and the labels as the loss function of the sentiment classification objective.

4. **Paraphrase Detection:** We have iterated through various methods for the model to learn the paraphrase detection method. At first, we have attempted to rely on cosine similarity on the two transformed sentence embeddings to generate output, which yields an okay result on the model. The final method that we eventually adopt is inspired by the Sentence-BERT structure proposed by Reimers and Gurevych (2019) Their original SBERT architecture is presented as below:

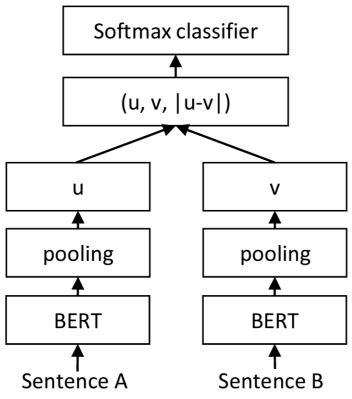
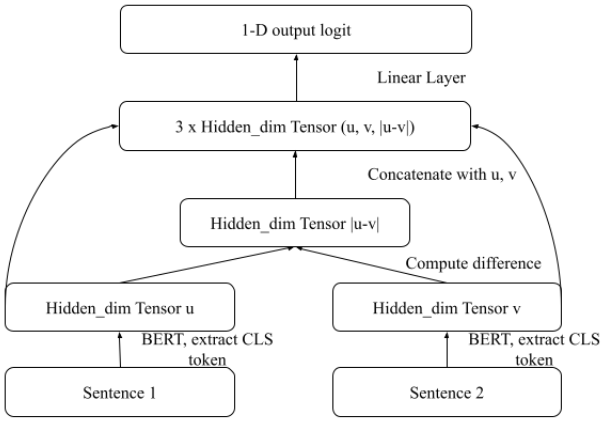


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

Note that originally, this classifier is used for classification, and a softmax classifier layer on the top is used to perform the task. As our output for each pair of sentences is just one single logit, we applied some modifications on the above architecture, implementing a neural network with architecture shown as below:



To maintain the consistency of gradients, we also apply a cross entropy loss to the output logit and the actual label to compute the loss function of this objective.

5. **Hyperparameter Tuning**: We also explore various hyperparameters to see if adjusting them would lead to a significant improvement in model converging speed or model performance. Specifically, we explored batch size, learning rate, and investigated the correlation between them.

5 Experiments

5.1 Data

We use the Stanford Sentiment Treebank dataset(**[1]**) for training and evaluating our model on the Sentiment Classification Task. This dataset consists of 11,855 sentences from movie reviews each labeled negative, somewhat negative, neutral, somewhat positive, or positive by 3 human reviewers. The dataset is partitioned into training which has 8,544 examples, dev which has 1,101 examples, and test which has 2,210 examples.

We also use the Quora question pairs dataset(**[2]**) for training and evaluating our model on the paraphrase detection task. This dataset contains quora questions labeled as either paraphrases (1) of each other or not (0). It contains over 400,000 lines of possibly duplicate question pairs. The training set will contain 141,506 of these examples, the dev set contains 20,215 examples, and the test set contains 40,431 examples.

Finally, we are training and evaluating our model on the Semantic Textual Similarity Task using the SemEval STS Benchmark Dataset(**[3]**), which consists of 8,628 sentence pairs each with a similarity score between 0 (unrelated) and 5 (equivalent). The training dataset has 6,041 examples, the dev set has 864 examples, and the test set has 1,726 examples.

5.2 Evaluation method

For evaluation, we ran the default evaluation function for our model over the three tasks and their associated datasets, gathering the accuracy of our model's responses based on each dataset's ground truth responses and averaging across the three to get our overall scores.

5.3 Experimental details

1. **Model experiments:** for all model experiments, we keep a constant set of hyperparameter settings. Specifically, we maintain batch size 32, learning rate 1e-4, and 8 epochs. This

was decided through some hyperparameter experiments in which we found that this set of hyperparameters results in fastest convergence without compromising performance.

2. **Hyperparameter Experiments:** we tried all sorts of combinations of hyperparameter values performed using the base BERT network model. We tried the following values for these hyperparameters:

   - Learning rate: 1e-3, 1e-4, 1e-5, 1e-6
   - Batch size: 8, 16, 32, 64
   - Epochs: 3, 6, 8

5.4 Results

Our final results for the test dataset were: **SST: 0.524, Paraphrase: 0.797, STS: 0.68**8.

The results from our model experiments are shown in **(Table 1)**. The main difference across each trial lies in the prediction function used. We found that applying cosine similarity to the input embeddings of the Paraphrase detection and semantic similarity tasks offered some improvement to scores, but the greatest improvement occurred when applying the SBERT classification objective function to semantic similarity and Paraphrase detection.

Table 2 contains results pertaining to our hyperparameter experiments, as described in the experimental details section. We found that although larger learning rates corresponded to worse scores when using smaller batch sizes, larger batch sizes benefitted from the increased learning rate.

Table 3 shows a comparison between using MSE loss and Cross entropy for the semantic similarity task in the context of our multitask learning architecture. Despite MSE loss being a common option for the semantic similarity task, we found that using cross entropy actually improves the score, a result which we believe may be related to congruence across the three loss functions used during simultaneous multitask learning.

Model	SST Dev	Paraphrase Dev	STS Dev
minBERT (baseline)	0.322	0.489	0.040
Multitask BERT	0.501	0.543	0.060
Multitask BERT with cosine similarity on paraphrase only	0.498	0.589	0.473
Multitask BERT with cosine similarity on sts and difference on paraphrase	0.498	0.589	0.473
Multitask BERT with SBERT on paraphrase and sts	0.514	0.642	0.613

Table 1: Model Experiments

Hyperparameters	SST Dev	Paraphrase Dev	STS Dev
Batch size 8, lr=1e-5 (baseline)	0.501	0.724	0.706
Batch size 8, lr=1e-3	0.475	0.642	0.612
Batch size 32, lr=1e-5	0.511	0.638	0.580
Batch size 32, lr=1e-4 (BEST)	0.509	0.767	0.723

Table 2: Hyperparameter Experiments

Model	SST Dev	Paraphrase Dev	STS Dev
MSELoss on sts	0.483	0.732	0.528
CrossEntropyLoss on sts	0.509	0.767	0.723

Table 3: Loss Function experiments

6 Analysis

Here we analyze the results from our extensions and offer explanations for why we think we got those results.

1. **Multitask Learning with Gradient Surgery:** Since the overarching goal of our project was to train an additional representation on top of the existing BERT model that can be used to perform on three different tasks, implementing multitask learning was the logical first step. Additionally, since gradient updates have a good chance of being conflicting, we decided to include gradient surgery to reconcile these differences.

   In **table 4**, we see that just by implementing multitask fine tuning with gradient surgery on top of the baseline model, we achieved a substantial increase in performance across all three tasks as expected. This should make sense, since adding any sort of fine tuning mechanism to the baseline is likely to give it improved performance. Interestingly, the model's performance increased by a similar proportion on all three tasks and showed that the gradient update mechanism favors no single task, even when gradient updates for paraphrase and sts were down weighted so sentiment classification would be treated as the main task. The main + auxiliary task approach ends up not working ideally in implementation for training our model.

Model	SST Dev	Paraphrase Dev	STS Dev
minBERT (baseline)	0.322	0.489	0.040
Multitask BERT (even)	0.501	0.543	0.060
Multitask BERT (main+aux)	0.497	0.528	0.060

Table 4: Different Multitask Techniques

2. **SBERT inspired classification function:** Our most significant improvement came from redefining our paraphrase detection and semantic similarity prediction functions (**Table 5**). This is a reasonable result since the new prediction functions take into account not only the individual inputs, but also the differences between them. Consequently, we would expect the model to better understand whether two sentences are the same as each other.

   A surprising result we also found while experimenting with the prediction and loss functions of our models was that while MSE loss is well used for training semantic similarity tasks, our model actually functioned better when equipped with cross entropy loss across all tasks. We believe that this is the case because we are implementing multitask-learning, and thus keeping the gradients to be on the same scale is crucial. MSE loss and cross entropy loss are drastically different functions: if each of them is evaluated individually, decreasing any of them does theoretically lead to a better performance of the model. However, such an intuition is not true when the two loss functions are added together. In our case, we see that MSE loss and cross entropy loss synchronize very poorly when being added together, breaking the congruence of gradients across the three different objectives and yielding a much worse result in return. This is primarily why we adopted cross entropy loss on training the sts task, as counterintuitive as it sounds.

Model	SST Dev	Paraphrase Dev	STS Dev
minBERT (baseline)	0.322	0.489	0.040
Multitask BERT with SBERT on paraphrase and sts, MSELoss on sts	0.483	0.732	0.528
Multitask BERT with SBERT and cross entropy loss on paraphrase and sts	0.516	0.797	0.710

Table 5: Different Prediction Functions

3. **Hyperparameter Tuning:** As our final improvement, we decided to perform hyperparameter tuning to achieve the optimal score via our model (**Table 6**). We found that while the performance of the model did not increase massively across different hyperparameters, using a larger batch size from 8 to 32 increased the speed to convergence, allowing us to reach

optimal performance by epoch 3 when using a batch size of 32 in contrast to convergence at epoch 6 when using a batch size of 8. We believe that this increased speed to convergence is due to better gradient estimation with larger batch sizes, since there is less likely to be as much noise and therefore variance across updates.

Also, although larger batch sizes and learning rates have the potential to lead to greater overfitting on the training set, we did not observe significant differences in performance between the training set and the dev set with a batch size of 32 and an increased learning rate of 1e-4. As we did not apply any additional regularization techniques to our model, it is possible that the use of a large quantity of training data was able to effectively prevent any possible overfitting that would have otherwise occurred.

Model	**SST Dev**	**Paraphrase Dev**	**STS Dev**	**SST Train**	**Paraphrase Train**	**STS Train**
Batch size 8, lr=1e-5 (baseline)	0.501	0.724	0.706	0.512	0.730	0.712
Multitask BERT with SBERT on paraphrase and sts, Batch size 32, lr=1e-4 (BEST)	0.509	0.767	0.723	0.524	0.781	0.725

Table 6: Different Hyperparameters

7 Conclusion

We implemented various fine tuning techniques on top of the pretrained BERT model to perform on three tasks: Sentiment Analysis (SST), Paraphrase Detection, and Semantic Textual Similarity (SST). We find that simultaneous multitask learning with uniform loss functions, along with modified, SBERT based prediction functions end up yielding decent results, achieving test accuracies of:
SST: 0.524 Paraphrase: 0.797 STS: 0.688.

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. *CoRR*, abs/1810.04805.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Wei-Cheng Tseng. 2020. Weichengtseng/pytorch-pcgrad.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *CoRR*, abs/2001.06782.

[1] https://nlp.stanford.edu/sentiment/treebank.html

[2] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

[3] default project handout (https://web.stanford.edu/class/cs224n/project/default-final-project-bert-handout.pdf)