Implementation of minBERT and contrastive learning to improve Sentence Embeddings

Stanford CS224N Default Project

Akshit Goel
Department of Electrical Engineering
Stanford University
akshit@stanford.edu

Linyin Lyu
Department of Computer Science
Stanford University
llyu@stanford.edu

Nourya Cohen
Department of Computer Science
Stanford University
nacohen@stanford.edu

Mentor: David No External Collaborators No shared project

Abstract

A language model's performance on various NLP tasks can be judged by the quality of sentence embeddings that it generates. We take inspiration from the SimCSE paper (Gao et al., 2021) and perform additional pretraining using contrastive learning to improve the sentence representation for multitask performance. We also integrate gradient surgery in the multitask learning process based on the paper (Yu et al., 2020) which highlighted the importance of non-conflicting gradients for improved multitask performance. We further conducted multiple experiments to improve our multitask training strategy. We observe that training through randomized distribution of batches amongst the different tasks help a lot in improving the multitask performance. We achieve a 45% improvement in the STS performance using multitask learning as compared to our baseline for single task finetuning. We analyse our results and identify areas for further improve the performance of multi-task learning.

1 Introduction

Multitask Learning takes a cue from how humans tend to learn, as people frequently transfer knowledge gained from prior experiences to facilitate the acquisition of new skills. Likewise, it is beneficial for several related tasks to be learned together so that knowledge learned from one task can benefit others. For the default project, we extend BERT model to improve its performance on multitask learning. First, we pretrained BERT with contrastive learning objective. It encourages the model to produce embeddings such that the similarity between positive pairs is higher than the similarity between negative pairs. With better sentence representation, we tried different strategies to process the three tasks (Sentiment Analysis, Paraphrase Detection and Semantic Textual Similarity) and balance their loss functions. We implemented gradient surgery based on the paper (Yu et al., 2020) to improve the multitask performance by having non-conflicting projections of gradients with respect to different tasks. After concluding all the experiments we choose the combination of the best strategies amongst all our experiments to get the final results.

2 Related Work

The three works that mainly informed our paper was the (Devlin et al., 2018) paper which established the BERT model, the (Gao et al., 2021) paper on Contrastive Learning, and the (Yu et al., 2020) paper on gradient surgery to improve multitask performance.

The (Devlin et al., 2018) paper establishes the BERT paper that we are currently using as the leading base model for word embeddings. More specifically, BERT pre-trains on unlabeled texts on both the Masked Language Model and Next Sentence Prediction objectives. By stacking biderectional transformer encoder layers (Biderectional Encoder Transformer Layers or BERT) and conducting the afformentioned pre-training, BERT needed only one additional output layer for finetuning any range of tasks and produced state of the art results.

The (Gao et al., 2021) paper extended this by using contrastive learning as additional training for BERT embeddings. Their results showed that contrastive learning regularized the pre-trained embeddings' anisotropic space to be more uniform. The unsupervised approach which we implemented, takes in an input sentence and predicts itself in a contrastive object, with the only noise being because of standard dropout. The results for the unsupervised approach were 4.2% higher on the Spearman's correlation then the previous best results.

The (Yu et al., 2020) paper on gradient surgery for multi-task learning further extends this research. While deep learning research often produced great results on single task learning, multi-task learning, which is far more efficient data wise, is more challenging from an optimization perspective lowering the overall efficiency. This paper suggests that one reason for the lower quality results with multi-task learning is gradient interference caused by the different tasks within the multi-task optimization landscape. It suggested that projecting gradients for the tasks such that they no longer conflict can lead to improved efficiency and performance on these tasks in a multitask training setting.

3 Approach

We use the pretrained BERT as the base model. On top of the base model, we add additional feed forward layers to adapt to specific tasks. The prediction head for the sentiment analysis task is a dropout layer followed by a linear layer which outputs the appropriate classification label. For the paraphrase detection and semantic textual analysis tasks, we first generate the embeddings for both the input sentences. These embeddings are concatenated together and passed through some shared layers like dropout, linear and GELU layers. The output of these shared layers is passed through a task-specific output layer which makes the prediction based on the expected output type for that task. With this architecture (as shown in Figure 1), we explore the below extensions hoping to improve the performance on the multitask learning. These extensions were tried in isolation of each other on a smaller dataset. Specifically, we took about $1/16^{th}$ of the paraphrase dataset, so that all the 3 datasets for SST, Paraphrase detection and STS were of similar size. This allowed us to iterate quickly and try multiple experiments. The experiments which lead to improved results were combined together into our final run on complete dataset.

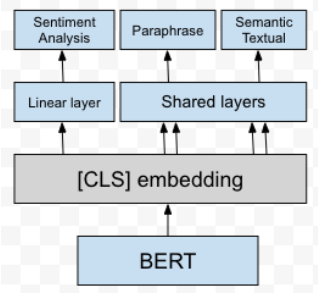

Figure 1: Architecture of the multitask learning.

3.1 Contrastive learning

We implement the unsupervised contrastive learning approach based on the paper (Gao et al., 2021). We pass a sentence to the BERT model twice to get two embeddings. The embeddings are different because BERT apply dropout in the attention and feed forward layers. This way we get a positive pair. The other sentences within the same mini-batch are considered as negatives pairs. We pretrain the BERT model on this contrastive learning objective (Equation 1) to increase the similarity of positive pair compared to negative paris. After the pretraining, we use the saved model to further finetune on the three tasks - Sentiment analysis, Paraphrase detection and Semantic Textual Similarity in the multitask context.

$$L = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{N} \exp(sim(z_i, z_k)/\tau)} \tag{1}$$

In the above equation, sim is cosine similarity, z_i and z_j are the embeddings of the positive pair, z_k are embeddings of other sentence within the batch, and τ is a temperature parameter to adjust how "sharp" the distribution is.

Our idea behind using contrastive learning based on (Gao et al., 2021) was that the performance of a model on different sentence level tasks depend upon the quality of the sentence embeddings. Using contrastive learning can help a model understand the similarities and differences between different sentences and reflect that in the embeddings it generate. We believe this can lead to better quality sentence embeddings and help on the downstream NLP tasks. Further, since we use unsupervised contrastive learning, it eliminates the need to get a separate labelled data. We combined all the sentences from the three tasks and used that as the training dataset for contrastive learning in the hope that the model performs better on these datasets for the three tasks as part of multitask learning.

3.2 Gradient surgery

In parallel to working on contrastive learning, we worked on the idea of gradient surgery based on the paper (Yu et al., 2020). We worked on this idea since we noticed that our multitask performance based on our initial experiments was lower than the performance of individual tasks. Therefore, our hypothesis was that there might be some gradient conflicts that were preventing the model to perform well on multitask learning and gradient surgery might help in removing these conflicts to improve the model's multitask performance.

The gradient surgery is based on the idea that in a multitask training environment, different tasks might interfere in the optimization process of each other. This might lead the model to have conflicting gradients which prevents them from performing well due to destructive interference. To detect if the two gradients are conflicting, we use inner product between them. A negative dot product reveals that the gradients are conflicting and we must subtract the projection to obtain non-conflicting gradients.

We denote the gradient with respect to task 1 as $grad\_task_1$ and gradient with respect to a second task as $grad\_task_2$. These 2 gradients conflict with each other if the projection direction between them as indicated by equation 3 is negative. If its negative it implies destructive interference between them and the $grad\_task_1$ is modified as indicated by equation 4.

$$inner\_product = \langle grad\_task_1, grads\_task_2 \rangle \tag{2}$$

$$proj\_direction = \frac{inner\_product}{\langle grads\_task_2, grads\_task_2 \rangle} \tag{3}$$

$$grad\_task_1 = grad\_task_1 - \min(proj\_direction, 0) \cdot grads\_task_2 \tag{4}$$

In order to train using gradient surgery, we organized the datasets for all the 3 tasks in what we call as super-batches. Each super batches contains 1 batch each from the three tasks. We experiment with two different orders within a super batch. The fixed order consists of the batches within a superbatch in the order - Sentiment Analysis (SST), Paraphrase Detection and Semantic Textual Similarity (STS). While for randomized order we distribute the tasks randomly in a superbatch.

3.3 Different training order approaches

In addition to contrastive learning and gradient surgery we conducted the following experiments in isolation to find out the best training strategy that we must adopt. In this class of experiments we studied the impact of different training order strategies as described below:

1. **Fixed Order**: This is the default approach that we assumed for multitask training. Here we processed the tasks in a fixed order and also completed training all the batches of a task before moving to the next task. we sequentially process each task following the fixed order - Sentiment Analysis (SST), Paraphrase detection and Semantic Textual Similarity (STS) for every epoch (as shown in figure 2 (a)). Also note that each task was performed for its entire dataset before moving to the next task.
2. **Randomized Order**: In this approach, we make a list of batches for all the tasks and randomly shuffle this list. Then we iteratively process the batches from this list as shown in figure 2 (b)) Therefore, basically in this approach neither the order of tasks is fixed nor the entire batch of tasks is trained at once.

   The idea behind using this approach was that we felt that the model might be overfitting to the task order during training and thus have lower performance on dev set. Randomizing the order, prevented this overfitting and improved the performance of our model as indicated in the results section.

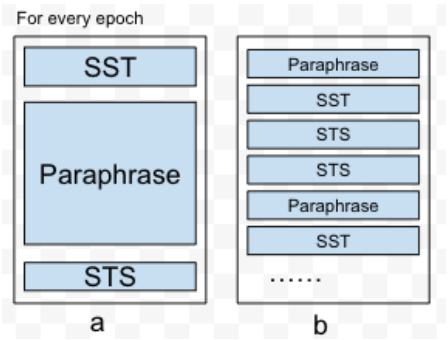

Figure 2: (a) Fixed order to process each task. (b) Interleave batches from different tasks with random order. (Note: The box size is proportional to the data size for different task)

3.4 Different loss update approaches

Since we noticed from the experiments based on section 3.3 that the random order performs better, we conducted the experiments for the strategies in this section only on the randomized order. We experiment with two strategies to update the model under this approach:

1. **Individual Loss Update**: Under this strategy, we process the batch for each task independently. This means that we compute the loss separately using the task-specific predictions and the ground truth labels. The mode is updated based on each task's loss independently.
2. **Combined Loss Update**: We tried this approach because we felt that since the overall goal is to have higher multitask performance, we must combine the losses from all the tasks and update the model based on the gradients with respect to this combined loss. Under this approach we adopted a batching strategy of super-batches similar to gradient surgery.

4 Experiments

4.1 Data

For pretraining with contrastive learning, we take the sentences from all the 3 provided datasets for Sentiment Analysis (SST), Paraphrase Detection and the Semantic Textual Similarity. We combine all the sentences into a single text file and use it as the dataset for contrastive learning. For multitask

learning, we use the provided three datasets: Stanford Sentiment Treebank (SST) for Sentiment Analysis, Quora for Paraphrase Detection and SemEval STS Benchmark dataset for Semantic Textual Similarity.

4.2 Evaluation method

For sentiment analysis and paraphrase detection task, we use accuracy to evaluate the performance. For the semantic textual analysis, we use Pearson correlation to compare the predicted similarity scores with the true similarity values. This is in line with the metrics provided in the default project handout.

4.3 Baseline

Our baselines are formed by finetuning on each task and dataset separately. The baselines for SST, Paraphrase and STS are 0.524, 0.810, 0.381 respectively.

4.4 Experimental details

1. **Data size**: We first sampled 8843 examples from the Quora dataset to shorten the training time for paraphrase detection task. By doing so, the three datasets have similar data sizes, and the problem is simplified to learning multitask for relatively similar size datasets. This also helps us experiment rapidly and reduce costs. After we compare the results of all experiments, we'll pick the best experiment and use the whole dataset without sampling to run the experiment again and make the submission.
2. **BERT**: We used all the default parameters of BERT. For all of our experiments, we take the [CLS] representation as the sentence embedding.
3. **Multitask learning**: For finetuning multitask we used the learning rate of 1e-5 and batch size of 8. The finetuning here refers to using the '–option finetune' flag to update all the parameters including BERT and additional layers. We did not use the '–option pretrain' flag, since the BERT parameters are frozen under this flag.
4. **Contrastive learning**: To pretrain BERT using contrastive learning objective, we used a learning rate of 3e-5 and temperature $\tau = 0.05$ (Gao et al., 2021).
5. **Epochs**: When finetuning multitask, we found that development accuracy plateaus early even though the training loss continues decreasing. This indicates that the model starts overfitting after 4 epochs (Figure 3). Thus, we adopted early stopping and concluded that using a smaller number of epochs (epochs = 5) is sufficient for finetuning BERT for our experiments.

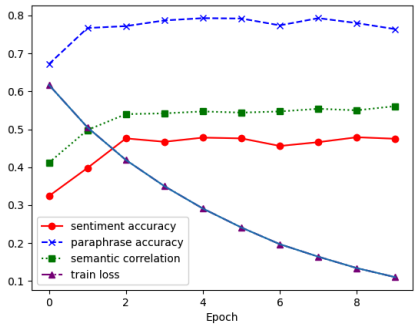

Figure 3: Training loss and Dev accuracy

4.5 Results

The table below shows the results from our different experiments on multitask learning with the same sampled dataset mentioned in 4.4.1:

Multitask			
Tasks	Sentiment Analysis	Paraphrase Detection	Semantic Textual Similarity
Fixed order	0.499	0.675	0.465
Random order	0.518	0.725	0.509
Gradient surgery(fixed)	0.514	0.690	0.429
Gradient surgery(random)	0.510	0.732	0.370
Combined loss	0.503	0.725	0.408
Contrastive Learning	0.490	0.719	0.498

As expected, we realize that the performance improves when the batches are randomized. We believe this is because in a fixed order the model overfits to the given order of the tasks and is not able to generalize well across different tasks. In case of gradient surgery, we used super-batches as indicated in the approach section. We believe since using superbatches automatically distributes the batches of different tasks, it does not happen that one task completes all its batches before the other tasks even starts. This prevents the model to overfit on one particular task and probably is the reason why for gradient surgery the fixed and random results look similar where fixed order performs better on STS, random order performs better on paraphrase. For SST both the orders perform almost similar.

In general, contrary to our expectations gradient surgery does not help improve the performance consistently across all tasks. In case of fixed order it helped improve the performance on Sentiment Analysis and Paraphrase Detection while degrades the performance on Semantic Textual Similarity (STS). For random order, it helps improve the performance on Paraphrase Detection, but degrades the performance on Sentiment Analysis. For STS, it remains almost similar and just degrades slightly. As per our understanding, we believe that STS performance degraded upon gradient surgery probably because of the following 2 reasons:

- The STS dataset is the smallest amongst all and is probably the most difficult task amongst the three tasks as its a regression problem as compared to the other tasks which are classification problems with a lot less possible values of the outputs.
- The gradient surgery most probably reduces the magnitude of the gradients due to the operations as indicated in equation 4. This might lead to smaller updates for STS. We must have been able to confirm this behaviour by using larger learning rates for STS, but since the run takes a long time, due to limited compute resources, we have not been able to conduct further tests.

We also notice that the Combined loss experiment which was conducted on superbatches did not perform well as compared to different losses for each task and thus we have not considered it in our final run.

Also, contrary to our hypothesis we noticed that the performance degrades with contrastive learning. We'll discuss more about it in the analysis section.

Based on these results, we conducted our final experiment with randomized order multitask learning on full dataset across the 3 tasks. Our test results have been compared below against the baseline results.

	Baseline	Multitask
SST	0.524	0.522
Paraphrase	0.810	0.816
STS	0.381	0.552

Table 1: Final result

We observe that while the performance remained similar for SST and paraphrase, STS benefited a lot from multitask learning - an improvement of nearly 45%. We believe this improvement happened because of the small dataset for STS. Since the task is inherently harder, and also has a smaller dataset the model is not able to learn the nuances well and is not able to perform better with finetuning for just STS. However, with multitask learning, the dataset for all the three tasks taken together helped the model improve its understanding of the task. Also, we believe that since the paraphrase detection task holds a lot of similarity with STS, the latter was able to benefit through learning on the former task.

5 Analysis

As mentioned in the results section, the degradation of the results with contrastive learning was contrary to our expectations. To understand this behaviour we conducted further experiments on contrastive learning and studied the impact of contrastive learning outside the multitask learning setting. We evaluated the performance of only STS task with and without contrastive learning. In the latter situation we pretrained our model on contrastive learning before finetuning it on STS. With this additional pretraining on the contrastive learning objective, we observed an increase in the Pearson's correlation by nearly 22% (comparison show in table 2). This is in line with our expected behaviour because contrastive learning trains the model to identify similar vs different sentences and improves the sentence embeddings. Since the contrastive learning objective is quite similar to STS, this improved understanding of the task probably helps the model perform better on STS.

However, from our experiments, we observe that contrastive learning does not benefit in the case of multitask learning. While the results for paraphrase detection and STS are almost similar to that of without contrastive learning (random order results in the table), there is a greater drop on Sentiment Analysis. We believe that the performance degradation for Sentiment Analysis happen as the contrastive learning objective is different from the Sentiment Analysis, and thus the improvement in sentence embeddings does not translate to a better performance on Sentiment Analysis. A similar explanation is also reflected in the (Gao et al., 2021) paper which suggests using additional masked language modeling. The paper achieves it through adding a MLM loss function to the contrastive learning objective and trains the model on the joint loss. However due to limited time and compute resources, we could not verify this through further experiments.

	w/o contrastive learning	w/ contrative learning
STS	0.382	0.467

Table 2: Finetuning STS task with or without contrastive learning pretraining

Next, if we plot the training results of our best performing multitask experiment (Random order), we observe potential overfitting. The training accuracy of the model almost reaches 1 for all 3 tasks (Figure 4). This overfitting is most likely because the training dataset used for finetuning is much smaller compared to the size of model. This is potentially another reason why we didn't perform significantly better with multitask learning as compared to the baseline. We tried to address this though use of regularization techniques as described in the (Jiang et al., 2019) paper. The paper makes a similar suggestion by describing how due to limited data for finetuning tasks as compared to the extremely large capacity of pre-trained models, aggressive fine-tuning often leads to overfitting. It suggests adding 'smoothness-inducing regularization' to address the above issue. We tried to incorporate this technique as an additional experiment taking inspiration from a github repo[1] on this paper. The training results are presented in figure 5. We did not observe any significant benefit through the regularization. We believe this might have been most likely due to some issue with our implementation. However, we could not conduct more experiments to fix it because of limited time and compute resources. (Figure 4)

(Figure 5)

6 Conclusion

In this default project, we implemented the minBERT and adamW optmizier followed by a detailed exploration of several extensions for multitask learning. We initally thought that contrastive learning could help us improve the performance of multitask learning, but our experiments and analysis indicated that though it does significantly improve the STS performance on single-task finetuning, it does not give similar results for multi-task learning. We further tried to improve the multitask performance through the use of gradient surgery and experimented with different training strategies. We observe a significant improvement in the STS task compared to our baseline through the use of these techniques, but the performance on the other tasks does not improve much. We identified areas for further improvements like - better regularization, resolving task interference and better sentence embeddings to improve the performance on multitask learning.

[1] https://github.com/archinetai/smart-pytorch

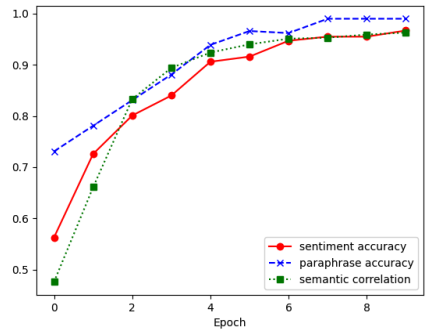

Figure 4: Training Accuracy for 'Random Order' multitask experiment

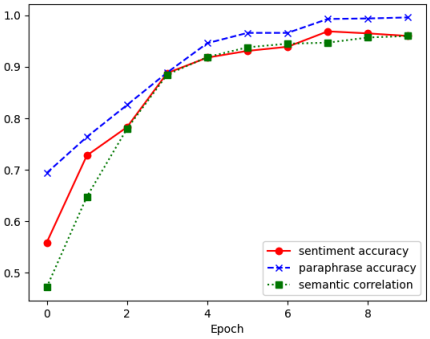

Figure 5: Training Accuracy for SMART

References

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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. SMART: robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *CoRR*, abs/1911.03437.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc.