minBERT and Downstream Tasks

Stanford CS224N Default Project

Jiajing Luo
Department of Computer Science
Stanford University
jiajingl@stanford.edu

Abstract

I pretrained robust embeddings with good down-stream performance across multi-tasks when fine-tuning BERT. I explored the extensions like multi-task finetuning, cosine similarity, simCSE and LoRA on my BERT model for sentiment analysis, paraphrase detection and semantic textual similarity. I optimized both on training time cost and accuracy by selecting the extensions widely used in industry. I also added a feed-forward layer before the linear projection to fine-tuning SST task.

Ultimately we find that full-mode with unsupervized simple contrastive learning of sentence embeddings has the highest overall accuracy score of 0.740 on test leaderboard. By setting r = 32 and $\alpha = 16.0$ for LoRA, I cut the training time from 14 hours to 11 hour.

1 Key Information to include

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

2 Introduction

BERT(Devlin et al., 2019) is a transformer-based model that significantly improved upon existing language models at its time of release, producing robust embeddings that could be fine-tuned to achieve state-of-the-art performance on a variety of downstream tasks.

My experiments utilized a suite of industry-popular methods to fine-tune a pre-trained BERT model, aiming to enhance generalization performance on three tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Additionally, I applied cosine similarity, supervised SimCSE, and unsupervised SimCSE for multi-task fine-tuning. To improve computational efficiency, I employed the LoRA model. In industrial applications, where models like ChatGPT can have billions of parameters, computational efficiency and memory usage can become bottlenecks in model training. As a result, parameter efficient fine tuning methods are critical even if it does not help with accuracy scores on the leader board. The report explained all the four extensions:(cosine similarity, SimCSE, multi-task, LoRA fine-tuning) applied in the project.

3 Related Work

3.1 Multitask Fine-Tuning

In the project, the BERT output needs to be fine-tuned on three tasks.

Rather than fine-tuning BERT on individual tasks, I can alternatively make use of multi-task learning to update BERT. For example, Bi et al. [1], use multi-task learning adding together the losses on the tasks of category classification and named entity recognition.

3.2 Cosine-Similarity Fine-Tuning

One potential way to improve my embeddings is to use CosineEmbeddingLoss while fine-tuning on the SemEval dataset.
The SemEval (Semantic Evaluation) dataset is widely used in natural language processing (NLP) for evaluating semantic understanding tasks, such as sentence similarity, semantic textual similarity (STS), and more. The dataset often includes pairs of sentences that are labeled with their degree of similarity or equivalence.
Some tasks in SemEval focus on identifying paraphrases, where two sentences are different in wording but identical or nearly identical in meaning. have a cosine similarity of 1, while unrelated sentences have a cosine similarity score of 0.

It's applied in STS task for calculating similarity and do multi-classification.

3.3 Contrastive Learning

Gao et al [3] proposed a simple contrastive learning framework called SimCSE that works with both unlabeled and labeled data. Unsupervised SimCSE simply takes an input sentence and predicts itself in a contrastive learning framework, with only standard dropout used as noise. In contrast, supervised SimCSE incorporates annotated pairs from NLI datasets into contrastive learning by using entailment pairs as positives and contradiction pairs as hard negatives.
Unsupervised SimCSE can be applied in all three tasks because they apply with the theory that within the same input, the BERT output should be the same.

3.4 Parameter Efficient Fine-tuning Methods: LoRA

Several techniques exist to reduce the memory usage/running time required to finetune language models, while preserving performance. One such technique is LoRA [4], which is motivated by the hypothesis that even during full fine-tuning, the updates to the weight matrices of the language model are essentially low rank.
LoRA (Low-Rank Adaptation of Large Language Models) LoRA is based on the hypothesis that, even during full fine-tuning, the updates to the weight matrices of the language model are predominantly low rank.
LoRA can help me train the BERT faster and reduce the over-fitting.

4 Approach

The baseline for the model is last-linear-model-training with warm up. It does not have any of the following extension. The baseline is for both accuracies and testing time cost.

4.1 Multitask Fine-Tuning

MTRec: Instead of fine-tuning BERT on individual tasks, you can alternatively use multi-task learning to update BERT. I used multi-task learning adding together the losses on the tasks of category classification and named entity recognition.

$$L_{total} = L_{task1} + L_{task2} \tag{1}$$

The total loss is calculated as the sum of SST, Para and STS tasks loss and then do the backward. If the unsupervised_simcse is enabled, I will add the total loss for unsupervised training result. If the supervised_simcse is enabled, I will add the total loss for unsupervised training result.
After that, I will take the backward on the total loss.

4.2 Cosine-Similarity Fine-Tuning

Cosine-Similarity is used in SST tasks. I applied cosine similarity funtion on the two BERT output given two different inputs and masks. I got the loss by mean square error loss instead of linear_loss given in the paper because mean square error penalizes larger errors more severely because of the squaring of the error term. It will benefit me to classify the results in 5 categories.

4.3 Contrastive Learning

I applied unsupervised simCSE in all three tasks and only supervised simCSE in paraphrase detection because only paraphrase detection gives similar context that can be tagged as being equal.
The main idea is to calculate the BERT output for the same input with different dropout two times. I got two different outputs and I wanted the outputs to be the same, so I calculated the loss on the two outputs and do the backward on it.
When calculating the loss, since the two matrixes should be the same, after their multiplication, I expected to get an identity matrix. I used cross_entropy on the difference between the multiplication result and the identify matrix as the loss.

4.4 Parameter Efficient Fine-tuning Methods

I used LoRA as the Parameter Efficient Fine-tuning Methods. It can efficiently improve the training time and reduce the over-fiting risk.

LoRA fine tune rank - r and LoRA scale α are passed by the command line to train the model.

Given a pre-trained model, the referred paper proposed re-parameterizing the weight matrices W of the model as $W_0 + BA$, where W_0 denotes the weight matrix obtained from pre-training, and A,B are parameters newly added during fine-tuning. Here, if W_0 is of shape $\times$ k, then B and A will be of shapes d$\times$ r and r$\times$ k respectively, for some r $\ll$ d,k.
A significant advantage of this technique is that less memory is required for the optimizer (which needs to store some state for each parameter being optimized). According to my test result, LoRA performs a little worse than standard fine-tuning in the accuracies performance. I need to balance between accuracies and training time by tuning r

5 Experiments

5.1 Data

I used Stanford Sentiment Treebank (SST) dataset and CFIMDB dataset for sentiment analysis, Quora dataset for paraphrases tasks and SemEval STS Benchmark Dataset for STS tasks.

5.2 Evaluation method

I used last-linear-layer-model with warm up as the baseline. I evaluated the performance from tasks accuracies and training time cost for multi-task, cosine similarity, simCSE fine tuning.

For LoRA fine tuning, it's only applied in unsupervised simCSE due to the time reason. I added LoRA in the last two days.

5.3 Experimental details

- GPU:NVIDIA GeForce RTX4080
- default learning rate = 1e-5
- LoRA_r = 32
- LoRA_α = 16.0
- default_batch_size = 8, also tried with 16
- weight_decay = 1e-3

- steps_per_epoch = 2000
- command example to train the model with supervised_simcse flag

```
python multitask_classifier.py --use_gpu --fine-tune-mode=full-model --lr=1e-5
--weight_decay=1e-3 --epoch=100 --batch_size=16
--supervised_simcse --unsupervised_simcse
--model_file_suffix=both_simcse_batch_16
```

- command example to train the model with LoRA flag:

```
python multitask_classifier.py --use_gpu
 --fine-tune-mode=LoRA --LoRA_r=4 --lr=1e-4
 --weight_decay=1e-2 --epoch=100 --batch_size=16
 --supervised_simcse --unsupervised_simcse
 --model_file_suffix=both_simcse_batch_16_LoRA_4
```

5.4 Results

Callout: multi-tasks fine tuning is added in all strategies except for the baseline. Supervised SimCSE paraphrase detection results are not as good as unpervisedSimCSE, so I do not submitted it on the Dev leaderboard.

- Baseline last linear layer with warm up.
- Cosine: STS task are trained with cosine similarity. Multi-tasks fine-tuning is enabled.
- Cosine + SimCSE: SST and paraphrase are trained with cosine similarity. Unsupervised SimCSE is applied in STS. Multi-tasks fine-tuning is enabled.
- SimCSE + LoRA: All tasks are trained with unsupervised SimCSE with LoRA. I also added a FFN before the linear project for SST tasks to fine-tune the SST tasks. Multi-tasks fine-tuning is enabled.
- SimCSE + full mode: All tasks are trained with unsupervised SimCSE with full-mode. I also added a FFN before the linear project for SST tasks to fine-tune the SST tasks.

Fine-tuning strategies	Overall dev score	SST accuracy	Paraphrase accuracy	STS dev correlation
Baseline	0.471	0.318	0.383	0.427
Cosine	0.662	0.517	0.805	0.329
SimCSE	0.703	0.517	0.800	0.586
SimCSE + LoRA	0.727	0.467	0.801	0.829
SimCSE + full-mode	0.732	0.482	0.808	0.814

Table 1: Dev Accuracy Table

Fine-tuning strategies	Training Time costs in Hour
Baseline	0.507
Cosine	11.76
SimCSE	14.18
SimCSE + LoRA	11.01
SimCSE + full-mode	14.47

Table 2: Training Time Costs

The result on Test leader board is from the combination of SimCSE + LoRA:
SST test accuracy: 0.511 (+0.511)
Paraphrase test accuracy: 0.805 (+0.805)
STS test correlation: 0.806 (+1.806)
Overall test score: 0.740 (+0.740)

5.5 Comment on your quantitative results

- **Testing Result followed my expectation**, for the overall performance, SimCSE + full-mode has the highest score and doing well in paraphrase detection. SimCSE + LoRA has like less then 1% accuracies reduction but the training times has been cut by 21%.
- **Testing Result followed my expectation**,The SimCSE + LoRA strategy achieves a higher overall test score than the development leaderboard, indicating LoRA's effectiveness in reducing overfiting risks. LoRA achieves this by reducing the number of trainable parameters and applying low-rank decomposition to weight matrices, thereby constraining the model to learn more generalized patterns and prevent overfit.
- **Testing Result not followed my expectation**, after applying unsupervised SimCSE, the SST accuracy went down. The reason can be that SimCSE is optimized for tasks that require understanding and comparing the semantic content of sentences. Sentiment analysis focuses on detecting the emotional tone or sentiment polarity (positive, negative, neutral) of a sentence. This task often requires fine-grained understanding of specific words and phrases that indicate sentiment, which might not be the primary focus of embeddings generated by SimCSE. In order to solve this issue, I added a FFN layer before the linear project for SST tasks to fine-tune the SST tasks in the following up strategy.
- **Testing Result not following my expectation**, for paraphrase detection tasks, all strategies performance are close to each other except for the baseline. The baseline is to get calculate the multiplication of two matrixes to get similarity. It can catch direction affectively. SimCSE multiplies two BERT output matrix to get the similarity, it also effectively catch the vector direction. They use different ways to capture the direction for the sentence structure structure for get the similar performance.
- **Testing Result not followed my expectation**, for STS tasks, even baseline had a higher STS accuracies than Cosine Similarity. Cosine similarity only considers the angle between two vectors but not considers their magnitudes. This can lead to issues when the magnitude of the embeddings contains valuable information about sentence similarity. Semantic Textual Similarity (STS) task focuses on the semantic analysis, the magnitude of embeddings can carry semantic significance, such as the intensity or importance of certain words. Ignoring this information can lead to less accurate similarity measurements.

6 Analysis

6.1 Selected examples for error analysis for SimCSE + LoRA strategy

- **Paraphrase Detection Error Examples**

Examples	Predicted Result	Ground Truth
Is it biologically good or bad to marry other caste? Is it good or bad to marry other caste?	0	1
What is addiction? What exactly is addiction?	0	1
Why do Quorans answer questions that are already answered? Why do Quorans downvote questions they cannot answer?	1	0
How do I upload photos to Quora with a pc? How do you include a photo with your post on Quora?	1	0

Table 3: Paraphrase Detection Error Examples

For the first two examples, if one sentence contains an intensifying adverb while the other maintains the same context, the paraphrase detection result may be inaccurate. The addition of an intensifying adverb can subtly change the sentence's meaning or emphasis, leading to semantic drift, which affects the paraphrase detection parameters.

For the second two examples, their semantic look closer with a lot of duplicate words but actually not the same meaning, so the paraphrase detection results are also impacted by the semantic analysis.

- **SST Error Examples**

Examples	Predicted Result	Ground Truth
A coda in every sense , The Pinochet Case splits time between a minute-by-minute account of the British court 's extradition chess game and the regime 's talking-head survivors .	2	4
Jaglom ... put -LRB- s -RRB- the audience in the privileged position of eavesdropping on his characters	1	3
A delirious celebration of the female orgasm	4	2
If you 're in the mood for a Bollywood film, here 's one for you .	4	2

Table 4: SST Detection Error Examples

The predicted results for the first two examples were neutral, contrary to the positive ground truths. Certain key terms like 'Pinochet Cases', 'LRB', and 'RRB' pose challenges for sentiment prediction when lacking context or training data. To enhance accuracy, including the paragraph as input and segmenting it into sentences for the feed-forward network (FFN) would provide contextual information for better estimation

In the second pair, predicted results were positive, but with the neutral ground truth. Positive terms like 'celebration' and 'mood' can sway sentiment judgment. Semantic training impacts STS detection akin to paraphrase detection, revealing potential conflicts in gradient directions between tasks.

- **STS Error Examples**

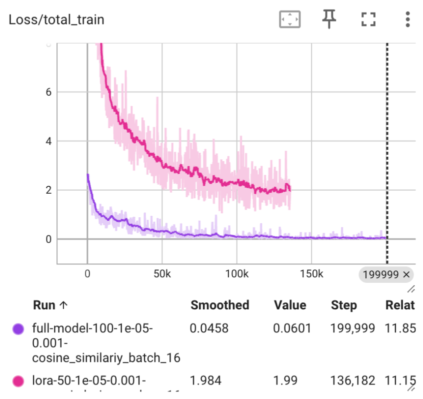
Examples	Predicted Result	Predicted Result * 5	Ground Truth
US drone strike kills eight in Waziristan US drone strike kills 11 in Pakistan	0.749	3.75	2.0
many of those executed in iran are hanged in public. many of the people executed are hanged in public.	0.923	4.16	3.4
Israel conducts airstrike on Syria Israel launches new airstrike against Syria	0.695	3.475	4.4
Sevens World Cup within Kenya,Äôs reach. Clash of Styles in Court Opens Trial in Young Man,Äôs Death	0.322	1.265	3.3

Table 5: STS Detection Error Examples

For errors 1 and 2, although the paraphrases for the given pairs are similar, differing key information such as 'eight in Waziristan' versus '11 in Pakistan' significantly alters the context. In example 2, 'in Iran' is present in sentence 1 but not in sentence 2. The prediction model prioritizes paraphrasing over missing or unmatched context, possibly influenced by paraphrase detection.

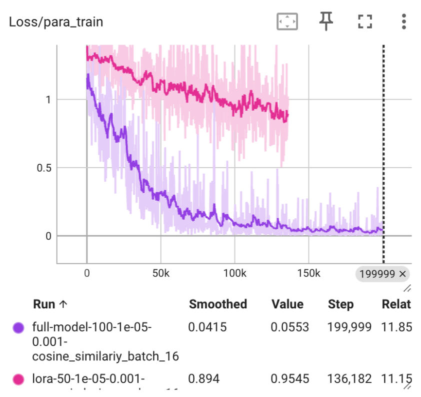
For error 3 and 4, the sentences have different paraphrases so even the semantic meanings are similar, the predicted scores are not that high. They also seemed to be influenced by paraphrase detection.

6.2 Loss Comparision Between Cosine Similarity and SimCSE with LoRA



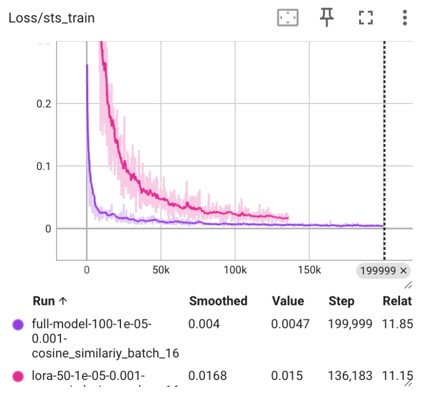
Run ↑	Smoothed	Value	Step	Relat
● full-model-100-1e-05-0.001-cosine_similariy_batch_16	0.0458	0.0601	199,999	11.85
● lora-50-1e-05-0.001-	1.984	1.99	136,182	11.15

Figure 1: Overall loss



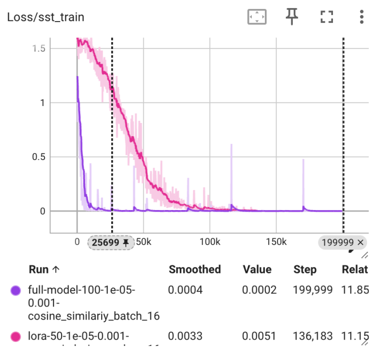
Run ↑	Smoothed	Value	Step	Relat
● full-model-100-1e-05-0.001-cosine_similariy_batch_16	0.0415	0.0553	199,999	11.85
● lora-50-1e-05-0.001-	0.894	0.9545	136,182	11.15

Figure 2: Paraphrase loss



Run ↑	Smoothed	Value	Step	Relat
● full-model-100-1e-05-0.001-cosine_similariy_batch_16	0.004	0.0047	199,999	11.85
● lora-50-1e-05-0.001-	0.0168	0.015	136,183	11.15

Figure 3: STS loss



Run ↑	Smoothed	Value	Step	Relat
● full-model-100-1e-05-0.001-cosine_similariy_batch_16	0.0004	0.0002	199,999	11.85
● lora-50-1e-05-0.001-	0.0033	0.0051	136,183	11.15

Figure 4: SST loss

These are the graphs from the dev training process from my local leader-board.
Cosine similarity in STS + multi-task fine tuning can reduce the loss and finish the training much faster than the SimCSE with LoRA in every tasks. However, from **5.4 Results**, we get that cosine similarity's score on test leaderboard can not beat the dev leader-board like SimCSE with LoRA does.

The reason could be on the model complexity. The cosine similarity builds a straightforward computation of cosine similarity between embeddings produced by BERT only in STS tasks. SimCSE involves a contrastive learning objective, which requires the model to distinguish between positive and negative sentence pairs, adding complexity in all three tasks. LoRA introduces low-rank adaptations that need to be fine-tuned, adding more parameters and complexity to the training process. Additionally, both SimCSE and LoRA introduced additional parameters.

7 Conclusion

The main findings for my projects of pretraining BERT on downstream tasks are as follows:

- Multi-Task Fine-tuning can be widely applied in various multi-tasks model training, it's not specific to task types. Gradient directions of different tasks may conflict with one another, so the tasks can impact each other's training result.
- Cosine similarity does well in capturing structure information but does not capture semantic information efficiently.
- SimSCE can capture both structure information and semantic information efficiently, but not good at sentiment analysis
- LoRA greatly helps both in both reducing the training time cost and reducing the overfiting risks.

One of the primary limitations for my work can be the dataset. I've not trained on the extension data set. The avenues for future work could be:
1. Further tuning on LoRA_r and LoRA_α to balance between training time cost and accuracies.
2. Yu et al. [5] recommend a technique called Gradient Surgery that projects the gradient of the i-th task gi onto the normal plane of another conflicting task's gradient g. It can mitigate the issue I found in **Analysis 6.1**

8 Ethics Statement

8.1 Ethical Challenges and Societal Risks

- To sentiment analysis, my model can be utilized to judgement all negative statement and deleting them from the social media. It can be utilized to control the political event and impact the election.
- Paraphrase detection, my model can be utilized to detect the historical online statements for a person, if the statments are judged to be like a threaten or against government multiple times, the person could be monitored and the account will be closed directly.
- Semantic textual similarity. My model can be applied in detecting the topics/context which government does not like by comparing the online statement and the sentences or phrases that are listed as banned.
- During the pandemic, I observed these risks materializing. The government suppressed statements that contradicted their lockdown strategies. They graded individuals' social media history to determine personal credit. People with low credit scores were barred from public places requiring credit checks.

8.2 Mitigation

- We require legal regulations for AI model usage, ensuring transparency in privacy protection and enforcing penalties for rule violations. Additionally, technical measures for mitigation and counteracting attacks are essential.
- We can assess bias using benchmark datasets designed to uncover biases in various areas like politics, ethnicity, and age. For example, an input such as 'I do not support close-down strategy' should be identified as a neutral statement rather than negative

References

A Appendix (optional)

A.1 Referred Papers

1 BERT and PALs: Projected Attention Layers for Efficient Adaptation in Multi-Task Learning

2 MTRec: MultiTask Learning over BERT for News Recommendation

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

4 Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022.

5 Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. Advances in Neural Information Processing Systems, 33:58245836, 2020.