Multi-task Learning and Fine-tuning with BERT

Stanford CS224N Default Project

Mel Guo
Department of Computer Science
Stanford University
`melguo@stanford.edu`

Abstract

With the advent of BERT, the capacity for NLP models to understand human language has significantly improved, yet challenges remain in fine-tuning these models for specific tasks without falling victim to issues such as catastrophic forgetting. This research contributes to the ongoing exploration of how large pre-trained language models like BERT can be effectively fine-tuned and adapted for a range of NLP tasks, offering insights into multitask learning, data sharing strategies, and the optimization of sentence representation methods for enhanced model performance. My project investigates various extensions to maximize the utilization of BERT on sentiment classification, paraphrase detection, and semantic textual similarity. Notably, round-robin multi-task learning, cosine similarity fine-tuning, shared relational layer for similar tasks, and the appropriate pooling method enhance BERT's performance when combined.

1 Introduction

Since the incipience of NLP as a field, computers have struggled to understand human language, particularly language context. The advent of pre-trained language models like Bidirectional Encoder Representations from Transformers (BERT) revolutionized the field of NLP by solving for numerous tasks with state-of-the-art accuracy. Unlike prior language representation models, BERT can pre-train deep bidirectional representations from unlabeled text to achieve a deeper sense of language context and understanding. The goal of my project is to utilize BERT to classify sentence sentiment, predict paraphrase pairs, and detect semantic textual similarity while sharing BERT word embeddings.

The state-of-the-art approach solves for multiple tasks via transfer learning, in which large language models gain general semantic and syntactic knowledge from pre-training before being fine-tuned on downstream tasks (Liu et al., 2019). However, this approach can be susceptible to catastrophic forgetting (Mccloskey and Cohen, 1989) when fine-tuning on multiple tasks one after the other causes the model to forget the weights learned from a previous task. Thus, this project implements multi-task learning (MTL), which allows models to aggregate training samples over multiple tasks and share knowledge together. Therefore, models can avoid catastrophic forgetting and tasks with smaller dataset sizes can benefit from the linguistic knowledge gained in richer data environments.

In this project, I investigate how the BERT's performance on three downstream tasks are impacted by multi-task learning with task-specific data, cosine similarity fine-tuning, shared relational layers between similar tasks, and [CLS] vs. MEAN pooling.

2 Related Work

Since the 1990s, multitask learning (MTL) (Caruana 1997) has been training paradigm in machine learning research for improving a model's generalization across related tasks. In recent years, researchers have been utilizing multitask learning (Crenshaw 2020) to help alleviate some of the most notorious obstacles in deep learning: large-scale data requirements and computation demand.

Multitask learning leverages shared representations, which can improve data efficiency, reduce overfitting through shared representations, and increase learning speed. Motivated by prior work on multitask learning (Stickland et al., 2020), this project aims to experiment with multi-task models that learn features general enough to enhance performance on tasks simultaneously. However, during multitask learning, gradients from different tasks can conflict with one another that can lead to detrimental results. BERT researchers have adopted MTL frameworks (Liu et al., 2019) to improve a model's encoding capabilities and tackled some optimization challenges through gradient surgery (Yu et al., 2020). Gradient surgery employs projecting conflicting gradients (PCGrad), which projects each conflicting gradient onto the normal plane of the other, preventing the interfering components from being applied to the network.

In Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, Reimers and Gurevych (2019) modify a pretrained BERT network by using siamese and triplet network structures to derive semantically meaningful sentence embeddings, which are compared by their cosine similarity. To generate sentence embeddings, Reimers and Gureych use MEAN pooling rather than using the [CLS] token embeddings. They also utilize cosine similarity fine-tuning, which helps compare sentence embeddings and determine their similarity. Their SBERT model outperforms other sentence=level models on STS and other transfer learning tasks, and I'll be taking after some of their approach in this project.

3 Approach

My project first evaluates upon the pre-trained minBERT and its SST-fine-tuned model. Then, in the spirit of transfer learning, I evaluate the model for paraphrase detection and semantic textual similarity (STS). The bulk of this research focuses on fine-tuning the model for paraphrase detection and semantic textual similarity by implementing round-robin multitask learning, shared relational layer for similar tasks, MEAN pooling, and cosine similarity fine-tuning.

3.1 Single-Tasking Approach

In this single-tasking approach, I used the pre-trained SST-fine-tuned minBERT model and tested how it generalized to the three tasks individually. For sentiment classification, I generated BERT embeddings, applied a dropout layer, and used a linear layer to output logits for the five sentiment classes. For the paraphrase detection and semantic textual similarity tasks, I generated BERT embeddings for both inputs, applied dropout to them, concatenated their embeddings, and used a linear layer to output a singular logit. For loss functions, I used cross entropy loss for sentiment classification, binary cross entropy loss for paraphrase detection, and mean squared error loss for semantic textual similarity. Since this is a single tasking approach, I computed gradients of each of the tasks' losses individually.

3.2 Round-Robin Multitask Fine-tuning

To generalize better across tasks, I decided to experiment with a round-robin multitask learning approach. Following in the footsteps of Bi et al, I create a multitask loss function that is the sum of the losses from each of the three tasks.

$$L_{total} = L_{sent} + L_{para} + L_{sts} \tag{1}$$

In each training epoch, I utilized the pre-trained BERT weights, cycled through a batch of SST, paraphrase, and STS at a time, calculated losses for each task, summed the losses, and computed the gradient of the total loss. Overall, this round-robin procedure ensures that the model gets exposure to datasets from all tasks.

3.3 Shared Relational Layer for Similar Tasks

Since paraphrase detection and semantic textual similarity are both tasks evaluating the similarity of sentence pairs, it's possible that using a shared relational layer between them could improve their overall learning. Especially since semantic textual similarity's STS dataset is much smaller than that of paraphrase detection, sharing a relational layer could STS learn from the other rich data source. I shared a nonlinear Leaky ReLU layer between them, with hopes that exposure to the richer Quora dataset would improve semantic textual similarity learning.

3.4 Weighted Multitask Fine-tuning

In my experiments integrating the shared relational layer into the round-robin multi-task learning model, results for paraphrase detection and semantic textual similarity improved, at the cost of sentiment classification. In attempt to allow the features learned from sentiment classification to feature more prominently alongside those from the shared relational layer, I decided to experiment with weighted multi-task learning with a few variations on the multitask loss function:

$$L_{total} = 1.12 * L_{sent} + L_{para} + L_{sts} \tag{2}$$

$$L_{total} = 1.25 * L_{sent} + L_{para} + L_{sts} \tag{3}$$

3.5 Cosine Similarity Fine-Tuning

One of my enhancements to minBERT is introducing cosine similarity to my fine-tune on the paraphrase detection and STS tasks. Cosine similarity measures the cosine of the angle between two vectors in inner product space, and it is a commonly used similarity metric in machine learning. Cosine similarity returns values from -1 (completely different) to 1 (the same).

$$\text{cosinesimilarity}(\mathrm{x}_1, \mathrm{x}_2) = \frac{\mathbf{x}_1 \cdot \mathbf{x}_2}{\max(\|\mathbf{x}_1\|_2, \varepsilon) \cdot \max(\|\mathbf{x}_2\|_2, \varepsilon)} \tag{4}$$

For each pair of embeddings, I calculated the cosine similarity between them. Then, I concatenated the cosine similarity value with the sentence embeddings to form a combined feature vector. I applied dropout to this feature vector and then passed it through a linear layer to produce a single logit for paraphrase detection and semantic textual similarity.

3.6 Pooling Methods

A common approach for BERT is to use the special classification token [CLS] embedding; however, this is often not a good summary of the semantic content of the input. Thus, to form a more representative sentence-level embedding, I experiment with MEAN pooling, which takes the average of the word embedding in the sequence.

4 Experiments

4.1 Data

- For sentiment analysis classification, I use movie review data from the Stanford Sentiment Treebank (SST) dataset and CFIMDB dataset. The SST dataset consists of 11,855 sentences extracted from movie reviews that were parsed with the Stanford parser. The dataset includes 215,154 unique phrases, with each phrase labeled as negative, somewhat negative, neutral, somewhat positive, or positive. The CFIMDB dataset consists of 2,434 highly polar movie reviews. Each review has a binary label of negative or positive.
- For the paraphrase detection task, I utilize the Quora dataset, which has 400,000 question pairs with binary labels of whether particular pairs are paraphrases of one another.
- Lastly, I use The SemEval Benchmark dataset for semantic textual similarity, which has 8,628 different sentence pairs of varying similarity on a scale of 0 (unrelated) to 5 (equivalent meaning).

4.2 Evaluation method

Paraphrase detection and sentiment classification are evaluated based on accuracy using the number of correct predictions. Since the STS label is continuous, I evaluate the semantic textual similarity task using the Pearson Correlation coefficient, a value between -1 and 1 that measures the linear correlation between true and predicted labels.

4.3 Experimental details

All experiments are implemented in Pytorch v2.2.1 and conducted on a workstation with one A100 or V100 Nvidia GPU. Unless otherwise specified, each experiment is run in 5 epochs with a fine-tune learning rate of 1e-5 and a hidden-layer dropout probability of 0.3. All models use the AdamW optimizer. Due to limited computing resources, I performed hyperparameter tuning on my models sporadically. See results below for more experimental details.

4.4 Results

4.4.1 Baseline Results

Table 1: Dev accuracies of baseline BERT model (pretrain and fine-tune) vs. Benchmarks

Model type	**Sentiment (SST)**		**Sentiment (CFIMDB)**	
	Accuracy	**Benchmark**	**Accuracy**	**Benchmark**
Pretrain default	0.416	0.390	0.788	0.780
Fine-tune default	0.522	0.515	0.963	0.966

Above is the dev set performance on the baseline minBERT model fine-tuned for sentiment class-fication, alongside the benchmarks from the project handout. My results are very similar to the benchmarks, suggesting the likely "correctness" of this implementation.

4.4.2 Single-task Learning Results

Table 2: Dev accuracies from single-tasking approach

SST Acc	**Para Acc**	**STS Corr**
0.316	0.640	0.067

The single-task BERT model performs sub-optimally. However, the accuracy for paraphrase detection was higher than I had anticipated, likely due to the larger size of the Quora dataset.

4.4.3 Multitask Learning Results

Table 3: Dev accuracies from multitask learning

Pooling	**SST Acc**	**Para Acc**	**STS Corr**
CLS	0.514	0.706	0.381
MEAN	0.511	0.728	0.383

As expected, multitask learning results are much better than the single-tasking results across all tasks. Since the SST and STS datasets are smaller, the model likely benefited from exposure to more data and more diverse data, enhancing its capacity to generalize to new, unseen data. Overall, this improved multi-task model performance over the single-task model shows the transfer learning potential among these three tasks.

Between the pooling methods, the differences are mostly trivial except for paraphrase detection – MEAN pooling outperforms [CLS] pooling. This is expected since paraphrase detection is the most complex of the tasks and likely benefits from the more comprehensive representation of a sentence that MEAN pooling provides.

4.4.4 Cosine Similarity Fine-tuning + Multitask Learning Results

Table 4: Dev accuracies from multi-task learning and cosine similarity fine-tuning

Pooling	**SST Acc**	**Para Acc**	**STS Corr**
CLS	0.513	0.724	0.348
MEAN	0.525	0.727	0.365

Next, I experimented with multi-task models by fine-tuning them with cosine-similarity and changing their pooling methods. Cosine similarity fine-tuning improves paraphrase detection dev accuracy across both pooling methods. Across all three tasks, mean pooling outperforms CLS pooling as expected. This suggests that averaging over all the BERT word embeddings is more effective than using the [CLS] token. One hypothesis is that MEAN pooling better encapsulates the essence of longer sentences where semantic meaning is more spread out across the sequence.

4.4.5 Shared Relational Layer + Cosine Similarity Fine-tuning + Multitask Learning Results

Table 5: Dev accuracies MTL, shared relational layer and cosine similarity fine-tuning

MTL	**Pooling**	**Learning Rate**	**Epochs**	**SST Acc**	**Para Acc**	**STS Corr**
Round-Robin	MEAN	1e-5	5	0.508	0.720	0.473
Round-Robin	MEAN	2e-5	10	0.503	0.754	0.493
Weighted (1.12)	MEAN	1e-5	5	0.516	0.709	0.408
Weighted (1.25)	MEAN	1e-5	5	0.513	0.712	0.419

The integration of a shared relational layer between the paraphrase detection and STS tasks brought dramatic improvements in the STS task, which was previously suffering from a scarcer dataset. This confirms my hypothesis that bringing correlations together from the Quora and STS data into a shared layer would boost performance. However, as expected, the model did prioritize these two similarity tasks at the slight expense of sentiment classification performance. The optimal feature representations gained from this shared layer of similarity detection was likely sub-optimal for sentiment classification, which is a different type of task that focuses on detecting the sentiment or emotion in a text.

To address the decrease in SST dev accuracy from integrating a shared relational layer, I decided to run two experiments with a weighted multi-task learning approach in which I scaled the SST loss by a factor of 1.12 and 1.25 before adding it to my multi-task loss (See Section 4.4). While this weighted multi-task approach did increase the SST accuracy score, it did so at a larger expense of both of the other tasks. Interestingly, the weighted multi-task approach in which SST loss was scaled by a factor of 1.12 worsened the accuracies of paraphrase detection and STS more than a factor of 1.25 did.

Since the round-robin multi-task model with a shared relational layer and cosine similarity fine-tuning generated the best results thus far, I decided to tune its hyperparameters in another experiment. Increasing the learning from 1e-5 to 2e-5 and the number of epochs from 5 to 10 caused a slight decrease in SST accuracy but notably increased in paraphrase detection (+0.34) and STS (+0.20) dev accuracies. I was slightly concerned that increasing the number of epochs would lead to more overfitting, but that didn't seem to happen. I suspect that the combination of a higher learning rate and more training epochs likely helped the shared relational layer mature. The higher learning rate could have also benefited cosine similarity fine-tuning by helping the model more efficiently adjust the direction of embedding vectors.

5 Analysis

Table 6: SST Sample of Erroneous Predictions

Sentence	**True**	**Predicted**
It's a lovely film with lovely performances by Buy and Accorsi.	4	3
Uses sharp humor and insight into human nature to examine class conflict, adolescent yearning, the roots of friendship and sexual identity.	3	4
Nothing's at stake, just a twisty double-cross you can smell a mile away – still, the derivative nine queens is lots of fun.	2	3
Entertains by providing good, lively company.	4	3
No one goes unindicted here, which is probably for the best.	1	2

Table 6 showcases a sample of some of my model's erroneous predictions. For the sentiment classification task (scale from 0 (negative) to 4 (positive)), many of the incorrect predictions are off by 1 class. It seems the model hasn't completely learned the nuances of the extent of a positive or negative sentiment. For instance, positive reviews with a label of 4 such as "It's a lovely film with lovely performances..." and "Entertains by providing good, lively company" receive a prediction of 3 (somewhat positive). However, in other examples such as "Uses sharp humor and insight into human nature to examine class conflict, adolescent yearning, the roots of friendship and sexual identity" where the label is 3 (somewhat positive) and the model predicted it to be a 4 (positive), some people might even agree more with the model's prediction. Sentences that are labeled as 2 (neutral), however, seem to be more ambiguous though, which perhaps is not necessarily at fault of the model but rather the ambiguous design of the task's labeling system from a scale of 0 to 4. In the example, "Nothing's at stake, just a twisty double-cross you can smell a mile away – still, the derivative nine queens is lots of fun", the former half indicates neutral indifference while the latter half is a more positive addition. This receives a label of 2 (neutral), perhaps because a combination of indifference and slight positive still balances out to a neutral ranking. However, the model likely interprets the phrase "lots of fun" as somewhat positive, which explains its prediction of 3. The last example in Table 6, "No one goes unindicted here, which is probably for the best" is a more difficult sentence to understand the sentiment of. The sentence contains a double negative and expresses negative sentiment in a rather ironic manner; it suggests that everyone being indicted would be a positive outcome. The model predicted this sentence to be a 2 (neutral), which is definitely too generous of a ranking, but not surprising since irony is difficult to learn.

Table 7: Paraphrase Detection Sample of False Positives

Sentence 1	**Sentence 2**	**Is Paraphrase**	**Predicted**
What can you get as a customer of Star Alliance?	What are some ways to register with Star Alliance?	0	1
"Which is correct grammar: I graduated from university of xxxx or I graduated from the University of xxxx? Do I need "the" here?"	"Grammar: What are the most common English language and grammatical errors made by people from India?"	0	1
...	...		
Why do I fall asleep when sitting?	Why do I keep falling asleep?	0	1

Table 7 exhibits a sample of false positives from the paraphrase detection task. A common theme among the false positives is interpreting similar sentence structure as an indication of a paraphrase. The first and last example are both questions, and each of their sentence pairs share similar syntax and topics with one another. The first question is about Star Alliance and the other question is about falling asleep. However, the model seems to miss out on important details that determine the sentence pairs to not be paraphrases. In the sentence pairs about grammar, the similarities in punctuation such as the colon seem to lead the model to think these questions are paraphrases when they are not.

Table 8: Paraphrase Detection Sample of False Negatives

Sentence 1	**Sentence 2**	**Is Paraphrase**	**Predicted**
Where can I find statistics for a certain disease?	I'm looking for disease stats, where should I search?	1	0
What are some signs that you're in a toxic relationship?	How do I know if my relationship is harmful?	1	0
...	...		
Can you recommend a good productivity tool?	What's a productivity app you would suggest using?	1	0

Table 8 exhibits a sample of false negatives from the paraphrase detection task. It seems that some of the same reasons for the model's false positives are also responsible for its false negatives. Overall, the model seems to be conflating syntax for semantics. In Table 8, it appears the model interprets different sentence structure or different question words as indications of not being paraphrases. It's also possible that the model is not picking up on synonyms as well, which is also being used in these sentence pairs (i.e. "statistics" and "stats", "toxic" and "harmful", "tool" and "app").

6 Conclusion

In this project, I investigated how multi-task learning and fine-tuning techniques can improve the efficacy of BERT on sentiment classification, paraphrase detection, and semantic textual similarity. My results showed that the round-robin multi-task model generalizes better to new, unseen data than the single-task model. Cosine similarity proved to be a useful for fine-tuning model capacity to detect sentence similarity. The improved results that MEAN pooling brought over the [CLS] token embedding for a more complex task like paraphrase detection showed the importance of choosing the appropriate sentence representation. Notably, implementing a shared relational layer between paraphrase detection and semantic textual similarity allowed the model to better learn from the richer, shared data environment. Overall, my best model, with an overall score of 0.668 on the dev and test set, utilizes round-robin multitask learning, cosine similarity fine-tuning, and shared relational layer for similar tasks. One of the primary limitations in my process was limited computing resources, which prevented me from further experimenting with different dropout rates, learning rates, epochs, and batch sizes. In future work, I am interested in exploring the effects of additional domain-adaptive pre-training on my model.

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*.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *CoRR*, abs/2009.09796.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, 24(C):109–165.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Asa Cooper Stickland and Iain Murray. 2019. BERT and pals: Projected attention layers for efficient adaptation in multi-task learning. *CoRR*, abs/1902.02671.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *How to Fine-Tune BERT for Text Classification?* Chinese Computational Linguistics.

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

A Appendix (optional)

If you wish, you can include an appendix, which should be part of the main PDF, and does not count towards the 6-8 page limit. Appendices can be useful to supply extra details, examples, figures, results, visualizations, etc. that you couldn't fit into the main paper. However, your grader *does not* have to read your appendix, and you should assume that you will be graded based on the content of the main part of your paper only.