Exploring Methods to Improve Robustness of Downstream Tasks for the BERT Language Model

Stanford CS224N Default Project

Kenny Dao
Department of Computer Science
Stanford University
`kdao@stanford.edu`

Viraj Mehta
Department of Computer Science
Stanford University
`viraj28m@stanford.edu`

Jeremy Tian
Department of Computer Science
Stanford University
`jtian25@stanford.edu`

Abstract

In this project, we first implement the minBERT model, which utilizes features of the original BERT model, an established large language model to understand human language. For developing and testing this baseline, we perform sentiment classification on two datasets, the Stanford Sentiment Treebank (SST) and the CFIMDB dataset. Following this baseline implementation, we implemented three main extensions to improve the minBERT performance: cosine-similarity/concatenation fine-tuning, unsupervised contrastive learning, and multi-task fine-tuning. Our model saw greatly improved performance over the baseline with these novelties. After extensive testing, we found the most improvement with sentiment analysis using contrastive learning, with paraphrase detection using concatenation, and with semantic textual similarity with cosine similarity, with overall results improving with multitask fine-tuning.

1 Introduction

The Bidirectional Encoder Representations from Transformers (BERT) model, released in 2018, saw state-of-the-art improvements on many important tasks in natural language processing, including question-answering and language inference. BERT's architecture entails a multi-layer bidirectional transformer encoder; the original authors pre-trained BERT on two tasks (masked LM and next sentence prediction), followed by finetuning on additional downstream tasks.

However, despite the advancements offered by BERT, the fundamental issue of deriving the most optimal sentence embeddings is still an ongoing, widely-researched problem in the sphere of natural language processing. BERT embeddings are pretrained to predict word tokens given surrounding context; therefore, it is difficult to build BERT sentence embeddings, especially given little communication between the pre-trained encoder and a task-specific layer.

In current literature, there have been many approaches towards solving this problem. One approach from Cer et al[2]. in 2017 attempted to construct sentence embeddings by applying pooling on the last layers of BERT, which is a common benchmark approach. Cer et al[2]. employed various combinations of pooling layers, and tested them on Semantic Textual Similarity (STS) dataset, seeing suboptimal results suggesting that the current approach of simply using pooling is a limitation of the current BERT model. Other approaches have seen more success, such as the use of skip-though vectors by Kiros et al.[4], in which they abstract the skip-gram model to the sentence level, seeing generic sentence representations that perform well in practice. However, this approach has not been applied to such a widely-used language model like BERT.

In this project, we decided to approach the problem of sentence embeddings by using a combination of cosine-similarity fine-tuning, unsupervised contrastive learning, and multitask fine-tuning. We first established a baseline minBERT model, implementing key features of the BERT model for the task of sentiment analysis. We then applied cosine-similarity to the downstream tasks of paraphrase detection and semantic textual similarity, finding improvements for the dev set accuracy for semantic textual similarity, although performance for paraphrase detection remained the same.

We also implemented unsupervised contrastive learning to learn sentence representations, a methodology that involves maximizing the agreement between different views of the same example sentence, while minimizing the agreement between different example sentences. We followed the contrastive learning framework outlined by Gao et al[3]. in the SimCSE paper, applying our own independently-sampled dropout masks and hyperparameter tuning. The use of unsupervised contrastive learning demonstrated improvements over the baseline minBERT model on sentiment analysis but not for paraphrase detection and semantic textual similarity. Finally, we tested out an approach using multitask fine-tuning, improving the performance of our model over all downstream tasks compared to the baseline.

2 Related Work

One of the key papers involving developing sentence embeddings using BERT is "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks" by Nils Reimers and Iryna Gurevych[6], in which they modified the original BERT architecture to use siamese and triplet network structures to create sentence embeddings that can be compared using cosine similarity. This model was mainly applied to the task of semantic textual similarity, outperforming other models in searching for semantically similar sentence pairs. In the SBERT architecture, a pooling layer was applied as a last layer to the BERT model, followed by a computation of cosine similarity between two sentences.

In addition, the relevant paper for unsupervised contrastive learning that influenced our work is "SimCSE: Simple Contrastive Learning of Sentence Embeddings," published in 2021 by Gao et al.[3] This paper first tested an unsupervised contrastive learning approach, in which the input sentence itself was used as a positive contrastive object, with dropout applied as noise, which greatly improved the quality of the sentence embeddings. Gao et al.[3] also tested a supervised contrastive learning approach, in which they used a labeled dataset to incorporate similarity between a premise sentence, a positive entailment sentence, and a negative contradiction hypothesis in their loss function. The supervised model demonstrated further improvement over the baseline.

It is also important in the space of multitask fine-tuning to understand the previous work by Yu et al. [7] in "Gradient Surgery for Multi-Task Learning." In this paper, they describe an approach for sharing structure across multiple tasks to enable more efficient learning. In the prior work by Bi et al.[1], "MTRec: Multi-Task Learning over BERT for News Recommendation," they add together the losses on the separate tasks of category classification and named entity recognition. However, Yu et al.[7] approach the problem of separate gradient directions for different tasks via gradient surgery, which involves modifying the gradients of shared parameters during backpropagation based on the relative importance of each task.

3 Approach

3.1 Baseline

For the first half of this project, we completed the baseline of implementing the minBERT model and testing it on sentiment analysis. In this approach, we first implemented the multi-head attention layer of the BERT transformer model, which maps a query and a set of key-value pairs to an output of the weighted sum of the values, which is calculated by the function:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

where the K, V, and Q are the key, value, and query matrices respectively, and d_k is the dimension of the queries and keys. This function is essentially identical to dot-product attention except for the scaling factor of $\sqrt{d_k}$. We additionally completed implementation for the BERT transformer layer consisting of multi-headed attention fed into an add & norm step, then a feed-forward layer, then finally put through another add & norm step. For the last part of the transformer layer, we also implemented the embedding layer for the embeddings of each token and the Adam optimizer based on decoupled weight decay regularization. In terms of the other downstream tasks, we implemented a random classifier for paraphrase detection and semantic textual similarity evaluation.

3.2 Extensions

Following the implementation of our baseline minBERT model, we implemented additional improvements to our model for the tasks of sentiment analysis, paraphrase detection, and semantic textual similarity.

Cosine-Similarity & Concatenation Fine-Tuning

First, we decided to use cosine similarity to improve our model's predictions. Given sentence embeddings for two sentences, u and v, we compute similarity scores prior to applying cross entropy loss, as shown in Figure 1 and 2 below.

We tested this approach on the pairwise downstream tasks: paraphrase detection and semantic textual similarity, finding improvement in the semantic textual similarity task, but relatively little improvement in paraphrase detection.

Thus, we also decided to test a classification objective function, in which we concatenate the sentence embeddings u and v with the element-wise difference $|u - v|$ and multiply it with a trainable weight before feeding it into the softmax function, as follows: $o = \text{softmax}(W_t(u, v, |u - v|))$. Following testing with each of our downstream tasks, we saw

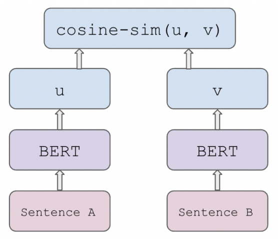

Figure 1: Cosine-similarity fine-tuning applied to sentence pairs

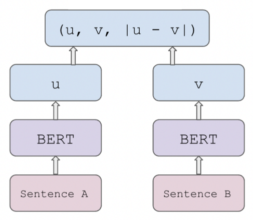

Figure 2: The classification objective function model, based on the SBERT architecture

improvement on the paraphrase detection task, but did not apply this approach to the other two, which did not see the most improvement.

Contrastive Learning

We next implemented an unsupervised contrastive learning approach to create better sentence embeddings via minBERT. The unsupervised contrastive learning approach was detailed as follows.

In each batch of size N, we have a collection of sentences $\{x_i\}_{i=1}^m$, and we use $x_i^+ = x_i$. This means that we pass the same sentence to the pre-trained encoder twice, which makes up our "positive pair" of embeddings that we want to minimize the distance between. The key to this is that each sentence has an independently-sampled dropout mask applied to it, thus creating two separate embeddings for the same sentence. This dropout acts as the sole form of data augmentation.

Following the application of these random dropout masks z and z', we denote $h_i^z = f_\theta(x_i, z)$ and $h_i^{z'} = f_\theta(x_i, z)$. Now we have our positive pair. To obtain our negative pairs, we simply apply a dropout mask to each of the other $N-1$ sentences in the mini-batch, obtaining $h_j^{z_j'}$. Thus, for each sentence in the mini-batch, our loss is given as follows:

$$\ell_i = -\log \frac{e^{sim\left(\mathbf{h}_i^{z_i}, \mathbf{h}_i^{z_i'}\right)/\tau}}{\sum_{j=1}^{N} e^{sim\left(\mathbf{h}_i^{z_i}, \mathbf{h}_j^{z_j'}\right)/\tau}} \tag{2}$$

where τ is the temperature hyperparameter, which is used to control the randomness of predictions prior to applying the softmax. Here, sim is computed via cosine similarity between the two embeddings. The above loss is for a single sentence chosen as the premise; however, we apply this loss to every sentence in each mini-batch as a premise, so our total loss is given as follows:

$$L = \sum_{i=1}^{N} \ell_i = -\log \frac{e^{sim\left(\mathbf{h}_i^{z_i}, \mathbf{h}_i^{z_i'}\right)/\tau}}{\sum_{j=1}^{N} e^{sim\left(\mathbf{h}_i^{z_i}, \mathbf{h}_j^{z_j'}\right)/\tau}} \tag{3}$$

A visualization for this approach can be shown above in Figure 3, demonstrating the positive sentence pair with different embeddings and each of the negative pairs.

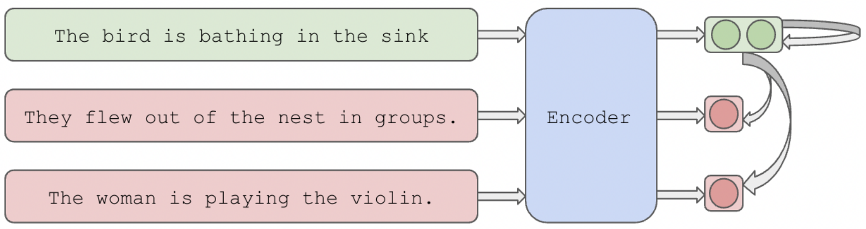
The bird is bathing in the sink
They flew out of the nest in groups.
The woman is playing the violin.
Encoder

Figure 3: Unsupervised Contrastive Learning: For each sentence denoted as the premise, different dropout masks are applied to obtain two embeddings.

We applied this contrastive learning approach to each of the three tasks outlined above. We found it to significantly outperform the baseline model on the task of sentiment analysis; however, it did not yield significant improvements for sentiment analysis or paraphrase detection over the classification objective function and cosine-similarity approaches outlined above, respectively. These results and analyses are detailed further below.

Multitask Fine-Tuning

Finally, we decided to try multi-task fine-tuning with a round robin approach to see if we could achieve better performance. This approach relied on the idea that optimizing the loss functions on each of the three tasks simultaneously would increase performance across the board. Thus, we defined our new loss function as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{sst} + \mathcal{L}_{p} + \mathcal{L}_{sts} \tag{4}$$

where $\mathcal{L}_{sst}$ is our sentiment analysis loss, $\mathcal{L}_{p}$ is our paraphrase detection loss, and $\mathcal{L}_{sts}$ is our semantic textual similarity loss.

However, one limitation of this approach is that there may exist gradient conflicts between different tasks, in which gradients in opposite directions harm each other, or the multi-task gradient is dominated by one gradient. Therefore, to resolve this issue, we implemented gradient surgery, a method involving altering the gradients by projecting each gradient onto the normal plane of the other, thus preventing interference. Given the gradients for two tasks, g_i and g_j, the new resulting gradient for g_i is given as follows:

$$g_i = g_i - \frac{(g_i \cdot g_j)}{\|g_j\|^2} \cdot g_j \tag{5}$$

We applied gradient surgery using the PCGrad (Project Conflicting Gradients) approach outlined by Yu et al.[7] We made use of a PyTorch 1.11 reimplementation by Nzeyimana[5].

4 Experiments

4.1 Data

We evaluated our baseline minBERT model using the Stanford Sentiment Treebank dataset, which consists of 11,855 sentences from movie reviews with labels on a scale of degree of negativity (0)/positivity (4), and the CFIMDB dataset, which consists of 2,434 highly polar movie reviews. We trained our model using these datasets for the task of sentiment classification on the pre-train and finetune modes of the model.

We evaluated our improved minBERT model on the three downstream tasks using the following datasets. We used the aforementioned Stanford Sentiment Treebank to evaluate on the task of sentiment analysis; we used the Quora dataset, which consists of 400,000 question pairs labeled with paraphrase indications, to evaluate on the task of paraphrase detection, and the SemEval STS Benchmark dataset, which consists of 8,628 sentence pairs of varying similarity, to evaluate on the task of semantic textual similarity.

4.2 Evaluation method

The evaluation metrics we used include accuracy for sentiment classification and paraphrase detection and Pearson correlation coefficient for semantic textual similarity. Accuracy measures the number of correct predictions in relation to the total number of predictions. The Pearson correlation coefficient, which is a measure of linear correlation based on the ratio of the covariance between variables, will be used to compare the true similarity values against the predicted similarity values.

4.3 Experimental details

Baseline

We evaluated our baseline minBERT model using various learning rates, dropout probabilities, and batch sizes to optimize these hyperparameters. We found the best results given the following hyperparameter values. We trained our model using a learning rate of 1e-3 for pretrain and 1e-5 for finetuning, using an Adam optimizer with weight decay regularization. Our batch size was 8 for both the SST dataset and the CFIMDB dataset. We also employed dropout with a probability of 0.3. Using a local machine with an NVIDIA GeForce RTX 3080 Ti, the model was trained for 10 epochs, with a total training time of 5 minutes and 6 seconds for the SST dataset and 5 minutes and 34 seconds for the CFIMDB dataset for the pre-trained model, and 20 minutes and 2 seconds for the SST dataset and 20 minutes and 34 seconds for the CFIMDB dataset for the finetuned model.

Extensions

For each of our extensions, we ran our model using a learning rate of 1e-3 for pretraining and 1e-5 for finetuning. For the tasks of cosine-similarity fine-tuning and contrastive learning, our average training times per epoch for pretraining were as follows, rounded to the nearest second: 63 seconds for sentiment analysis, 14 minutes and 32 seconds for paraphrase detection, and 127 seconds for semantic textual similarity. For finetuning, the average training times per epoch were 122 seconds for sentiment analysis, 41 minutes and 37 seconds for paraphrase detection, and 156 seconds for semantic textual similarity.

After implementing contrastive learning for sentiment analysis, we tested various dropout probabilities, using grid search to settle on the optimal one. Grid search is a method for performing hyperparameter optimization by considering various parameter combinations. Additionally, we used a fixed value for the temperature hyperparameter of 0.05, as this was the value used in the original SimCSE paper. We found the best performance with dropout probability of 0.8, with an accuracy of 0.527 for sentiment analysis. The below graph shows the results of our grid search for dropout values in increments of 0.1. For multitask learning, however, we saw an average training time of 1.1 hours per epoch for pretraining and 2.3 hours per epoch for finetuning.

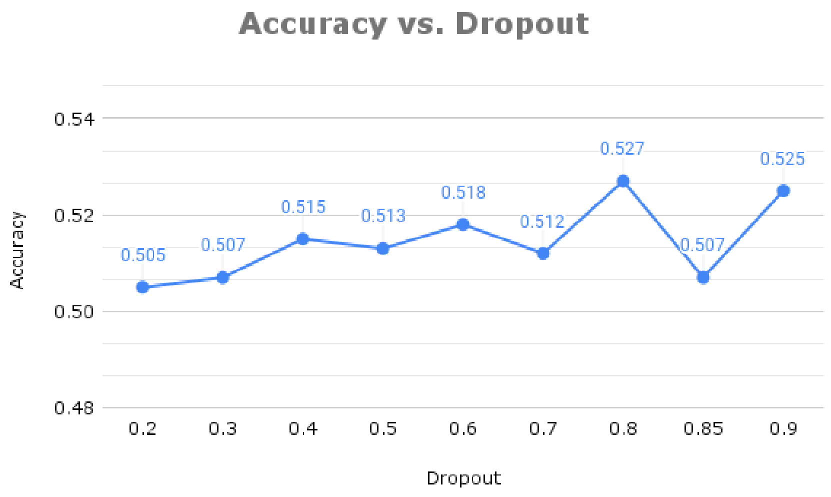

Figure 4: Effect of dropout on sentiment analysis accuracy with contrastive learning.
Until p = 0.8, the accuracy increased slowly.

We also applied contrastive learning to semantic textual similarity with a fixed dropout value of 0.3, but did not see improvement over the semantic textual similarity model based on cosine-similarity fine-tuning. As we increased the dropout probability, we saw worse results.

For our loss functions, we used cross entropy with our contrastive loss function as defined in the approach section above. We tested out the use of both MSE loss and binary cross entropy loss for paraphrase detection, but found better results using binary cross entropy, as paraphrase detection was framed as a binary classification task. However, after testing out cross entropy and MSE loss functions, we found best performance with MSE loss for semantic textual similarity.

For our implementation of multitask fine-tuning, we faced the limitation that the Quora dataset for paraphrase detection was much larger than the SST dataset for sentiment analysis and the SemEval STS benchmark dataset for semantic textual similarity by many orders of magnitude. To resolve this issue, we decided to resample data from the SST and STS datasets to match the size of the Quora dataset. We decided to use resampling rather than simply taking the first x number of examples in the Quora dataset, which would essentially truncate the dataset, in order to broaden the type of examples seen by our model. In addition to resampling, we also tried simply sampling from the Quora dataset in order to equalize the dataset sizes in an attempt to improve the total computation time. We tested out different numbers of samples from the Quora dataset including 1/10 of the Quora dataset size, 1/5 its size, and 1/2 its size. We found best results given 1/2 its size, as even though training took longer, the model was able to see more examples.

4.4 Results

For our baseline minBERT model results, we achieved a pre-train accuracy of 0.399 on the dev set and finetune accuracy of 0.532 on the dev set for the SST dataset. We also achieved a pre-train accuracy of 0.792 on the dev set and a finetune accuracy of 0.971 on the dev set for the CFIMDB dataset, as depicted in Figure 5 below.

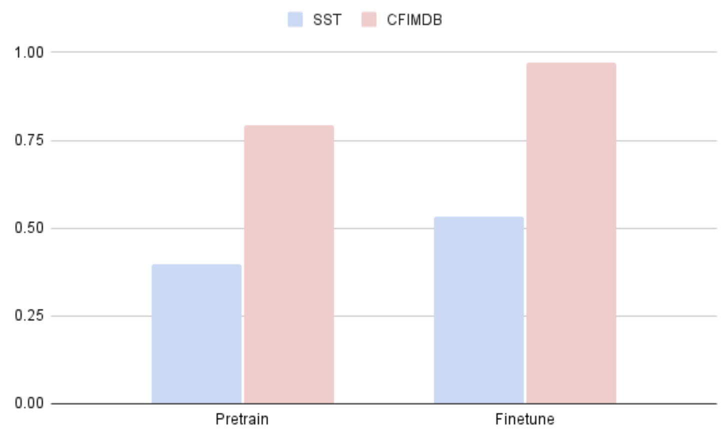

Figure 5: Results of baseline minBERT model on the SST and CFIMDB datasets for both pretraining and finetuning.

In terms of our Dev Set Leaderboard results, we achieved an SST dev accuracy of 0.405, paraphrase dev accuracy of 0.375, and STS dev correlation of 0.042 with an overall dev score of 0.274 for our baseline model, as shown in Figure 5 above.

For our extenstions, the below tables (Table 1 and Table 2) show the evaluation metrics we used (accuracy and Pearson coefficient) for each of the three downstream tasks given each of the different methodologies we outlined in our approach section above. We additionally show a comparison with our baseline results for each of the tasks.

	Baseline	**Concatenation**	**Cosine-Similarity**	**Contrastive Learning**
Sentiment Analysis	0.400	N/A	N/A	0.530
Paraphrase Detection	0.370	0.780	0.631	0.551
Semantic Textual Similarity	0.04	0.532	0.585	0.428

Table 1: Best results of the various approaches on the three downstream tasks when trained **individually**.

	Sequential Fine-Tuning	**Multitask Fine-Tuning**
Sentiment Analysis (CL)	0.334	0.510
Paraphrase Detection (C)	0.673	0.503
Semantic Textual Similarity (CS)	0.502	0.519
Overall	0.503	0.511

Table 2: End results of the two training techniques on the downstream tasks.
Sequential is trained in order of Sentiment, Paraphrase, then Semantic; Multitask utilized a Round Robin approach.
**CL = Contrastive Learning, C = Concatenation, CS = Cosine-Similarity*

As shown in the tables above, the best results achieved using sequential fine-tuning were with using unsupervised contrastive learning for sentiment analysis (accuracy = 0.334), concatenation for paraphrase detection (accuracy = 0.673), cosine similarity for semantic textual similarity (STS) (Pearson coefficient = 0.502). However, once multitask fine-tuning was implemented, the results improved for both sentiment analysis using contrastive learning and STS using cosine similarity, with an accuracy of 0.510 and a Pearson coefficient of 0.519, respectively. Additionally, our overall average results for multitask fine-tuning demonstrated greater improvement over the baseline than any of our individual sequential fine-tuning efforts.

Below, we plotted our train loss given the number of epochs for multitask fine-tuning.

Based on these result statistics, we feel that many of our expectations were met. For one, we expected that the addition of contrastive learning would lead to improved accuracy for sentiment analysis. This is because we built contrastive learning on top of our baseline implementation and contrastive learning improved how our model comprehended the attitude of sentences. As for our training techniques, it was not surprising that paraphrase detection had decreased performance with multi-task fine-tuning as the other tasks increased. Since the Quora dataset used to train paraphrase was so large (over 10x the size of the other datasets) and given our resampling approach, we expected that paraphrase detection performance would decrease as much as it did. In terms of multi-task leading to a slight edge over sequential,

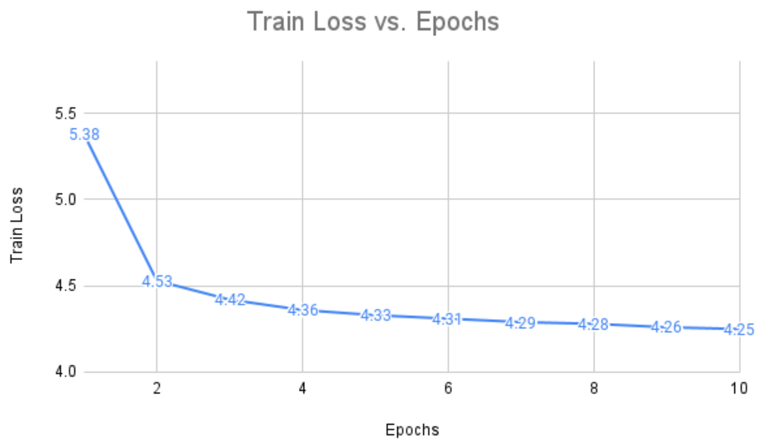

Figure 6: Train loss as a function of the number of epochs trained for multitask fine-tuning.

this was not too surprising as we expected the two to have approximately similar average/overall performance with individual tasks' performances converging towards the overall performance.

However, there were a few results that we did not expect and found surprising. For one, we were surprised to find that the cosine-similarity and contrastive learning did not lead to increased performance over concatenation for paraphrase detection. We initially expected the two extensions to be better at discovering the semantic differences between sentence pairs, but based on these results, it appears that concatenation's retaining of the original embeddings are more beneficial. Secondly, it was surprising that contrastive learning led to lower accuracy than concatenation for STS, although we did expect cosine-similarity to produce the best result. Similar to before, it appears that retaining the original embeddings and incorporating more information in the logits produces a more accurate semantic evaluation than contrastive learning. It is also surprising that paraphrase detection and semantic textual similarity had different best-performing approaches given that they both require a high level of semantic understanding of sentence pairs. A possible reason for this is the differences in task outputs where paraphrase returns a binary yes/no prediction and STS outputs a 1-5 equivalence rating. It is plausible that concatenation can achieve a brief idea of which binary output is appropriate but is not as capable as cosine similarity of producing polarizing outputs on a scale.

5 Analysis

Ablation Study

To understand the improvements offered by multitask fine-tuning over sequential finetuning, we conducted an ablation study. For each of our tasks, we isolated and removed the loss component, and then tested how well the multitask model would perform on the other two tasks. For example, we could remove the loss component $\mathcal{L}_{sts}$ associated with semantic textual similarity, and then see how our model performed on the other two tasks, sentiment analysis and paraphrase detection. This way, we could see if incorporating a task's loss actually significantly affected the performance on other tasks. The results of our ablation study are shown in the table below.

	$\mathcal{L}_{sst}$ **(Sentiment Analysis)**	$\mathcal{L}_{p}$ **(Paraphrase Detection)**	$\mathcal{L}_{sts}$ **(STS)**
Sentiment Analysis	N/A	0.494	0.491
Paraphrase Detection (C)	0.490	N/A	0.465
STS (CS)	0.505	0.486	N/A

Table 3: Results of evaluation on each task for removing each loss where each column header denotes which loss component was removed
**STS = Semantic Textual Similarity, C = Concatenation, CS = Cosine Similarity*

Upon evaluating these results, we see that the task of semantic textual similarity (STS) performed much worse when the paraphrase detection loss was removed, as opposed to when the sentiment analysis loss was removed. This likely has to do with the greater similarity between the tasks of paraphrase detection and STS, which both rely on semantic understanding of a sentence. The same is shown to be true of the task of paraphrase detection, which performs significantly worse when the STS loss is removed as opposed to when the sentiment analysis loss is removed.

Overall, the evaluation metrics show that the model performed worse on all tasks when a singular loss was removed. This can be attributed to a decreased understanding of both the English language and sentences as a whole given less tuning and learning for semantic and sentiment understanding.

Qualitative Analysis

For sentiment analysis, as shown in Table 4 below, we see that unsupervised contrastive learning provided an improvement over our baseline model on the example sentence, as it correctly predicted a 3 (somewhat positive) rating whereas the baseline predicted a 4 (positive). A possible reason for this is that the baseline is improperly emphasizing the importance of various polarizing words that may push its prediction too far in one direction. In this case, it might be over-emphasizing the word, “exceed”, which is associated with a positive review. On the other hand, contrastive learning may have produced a better prediction because it teaches the model to differentiate between similar and dissimilar word pairs, which can be used to better capture the relationships between words in a sentence.

	Example	**Label**	**Baseline**	**CS/C**	**CL**
Sentiment Analysis	"Extreme Ops" exceeds expectations.	3	4	N/A	3
Paraphrase Detection	Sentence 1: Where’s a good university to study Computer Science in the UK? Sentence 2: What are some good UK universities in computer science?	1.0	0.0	0.74 (C)	0.14
Semantic Textual Similarity	Sentence 1: Syrian forces move to retake Aleppo. Sentence 2: Syrian Regime Bids To Retake City Of Aleppo.	4.4	0.0	4.21 (CS)	2.85

Table 4: Example sentences & model predictions for downstream tasks
**CS = Cosine Similarity, C = Concatenation, CL = Contrastive Learning*

From the above table, we can see that for the example given for paraphrase detection, head concatenation resulted in a more accurate prediction compared to contrastive learning. The contrastive learning approach predicted a 0.14, indicating that the sentences were not paraphrases, while the concatenation approach predicted 0.74, which is closer to the true prediction. We hypothesize that the reason for the poor performance from the contrastive learning approach is that it placed too much emphasis on contrasting the "Where" with the "What," as these are typically two fundamentally different questions, causing the model to think that the two questions were not paraphrases.

We also noticed this same pattern with semantic textual similarity predictions when analyzing the incorrect predictions from the contrastive learning model. We found that there were quite a few of the aforementioned examples that the model got wrong: two sentences were clearly paraphrases of each other, but there was one word/phrase that might’ve normally indicated a different semantic meaning; however, in this particular pair of sentences it didn’t change the meaning. We believe that this was one source of decreased performance due to contrastive learning, as we did not notice this result in head concatenation for paraphrase detection or for cosine similarity in semantic textual similarity.

6 Conclusion

BERT is a powerful language model that has a good grasp of sentence meanings. With this being said, there are a few areas to improve on, namely achieving better sentence embeddings. To this end, our project addressed this primary issue by using a combination of cosine-similarity, unsupervised contrastive learning, and multi-task fine-tuning. Using these methods we were able to achieve much better results as compared to our baseline model.

Some limitations of our results came from the dataset as the Quora dataset was much larger than the other datasets, potentially making our model be more fine-tuned to that task. This was supported by the fact that our model performed the best on paraphrase detection by a large margin when sequentially fine-tuning. When we attempted to equalize the dataset sizes through resampling/sampling, we saw the performances on all the tasks cluster together more, supporting this hypothesis. Although we were able to find optimal dropout values, another limitation of our model was that we were not able to explore other hyperparameters, such as tweaking the temperature and learning rates, because the computational time to run 10 full epochs on all the training and dev set was very high.

Although these limitations are potential avenues of future work to further explore, we were ultimately able to greatly improve our model’s performance on the three downstream tasks (sentiment classification, paraphrase detection, and semantic textual similarity) through cosine-similarity, unsupervised contrastive learning, and multi-task fine-tuning.

References

[1] Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. Mtrec: Multi-task learning over bert for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2663–2669, 2022.

[2] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semantic textual similarity-multilingual and cross-lingual focused evaluation. In *Proceedings of the 2017 SEMVAL International Workshop on Semantic Evaluation (2017). https://doi. org/10.18653/v1/s17-2001*, 2017.

[3] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

[4] Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. *Advances in neural information processing systems*, 28, 2015.

[5] Antoine Nzeyimana. Pytorch-pcgrad-gradvac-amp-gradaccum/antoine nzeyimana, 2022.

[6] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[7] 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*, 33:5824–5836, 2020.