Improving the Performance of BERT Using Contrastive Learning and Meta-Learning

Stanford CS224N **Default** Project

Akash Gupta, Justin Shen, Peter Westbrook
Department of Computer Science, Stanford University
apgupta3@stanford.edu, jshen3@stanford.edu, petewest@stanford.edu

TA mentor: Jingwen Wu (Stanford CS224N Default Project)

Abstract

This project explores novel approaches to improving the multitask performance of BERT on tasks like sentiment classification, paraphrase detection and semantic textual similarity (STS). We first experiment with various training procedures including sequential training, round-robin training, annealed sampling, and square-root sampling. Additionally, we experiment with architectural changes in our output layer in order to improve the final performance. Lastly, we experiment with various loss functions such as the baseline cross entropy, MSE, and SimCSE, a contrastive learning approach. While improving the baseline model, we selected the most promising procedures, from each category (training, architecture, loss) to combine into a final model that significantly outperforms the baseline. We achieve an average final dev set performance of **74.3%**.

1 Introduction

The broad research goal of engineering powerful natural language systems is important for a variety of downstream tasks. In 2018, Devlin et. al made a significant step in this field in their paper that introduced Bidirectional Encoder Representations from Transformers (BERT) as a powerful baseline model that could be easily fine-tuned for a myriad of downstream tasks Devlin et al. (2019). With state-of-the-art results on many of the key natural language benchmarks (i.e. GLUE, Multi-NLI, SQuAD, etc.), it served as a cornerstone for natural language processing research in the coming years.

Although serving as a powerful baseline model, many papers have since come out proposing novel methods to improve the baseline BERT architecture on natural language inference (NLI) tasks. In our paper, we seek to explore these methods, and extend their applicability to a multi-task NLI setting. We experiment with methods like multi-task learning methods, gradient surgery, various loss functions, contrastive learning, and ensembling, yielding significant improvements over the baseline BERT architecture.

2 Related Work

More recently, newer papers have been published, proposing novel methods that build on Devlin et. al's work and expanding the capabilities of the baseline BERT model. In 2021, Gao et. al published SimCSE, a constrastive learning framework for learning more robust sentence embeddings Gao et al. (2021). Gao et. al's paper demonstrated that their SimCSE framework generated a significant improvement in natural language inference tasks over the baseline BERT model Gao et al. (2021). The core idea behind contrastive learning is given an anchor (the original example), a positive example (similar in the context of the task) and a negative example (not similar in the context of the task), we learn a contrastive representation in the embedding space. This pushes positive examples closer together while pushing negative examples farther apart, allowing for better downstream results in natural language inference tasks.

In 2020, Yu et. al, published another key paper that improved BERT's ability to learn in a multi-task setting Yu et al. (2020). Through *gradient surgery*, Yu et. al proposed a method that avoids conflicting gradient updates while training for multiple tasks Yu et al. (2020). This is important as oftentimes when learning in a multi-task environment, there may exist tasks that are "conflicting" meaning that an improvement in performance on one task may erode the performance for another task Yu et al. (2020). *Gradient surgery* solves this problem by projecting one task's gradient onto a normal plane of another conflicting task's gradient, allowing substantial improvements in the results of multi-task learning Yu et al. (2020). Stickland & Murray also published a paper that highlighted the importance that the training loop and sampling procedure have on downstream multitask learning capability Stickland and Murray (2019). Jiang et. al also introduced Smoothness-inducing (SMART) regularization, a method incorporates the minimization of the difference in output between similar inputs as part of the loss function, in order to reduce over fitting Jiang et al. (2020).

3 Approach

Our goal is to effectively train a multi-task BERT classification model to learn on multiple tasks. A naive approach to multi-task learning would be sequentially training our model on each task (i.e. sentiment classification, paraphrase detection, semantic textual similarity).

We will first fine-tune a baseline BERT model on each task using the above datasets using task-wise sequential training. To further refine our baseline BERT model, we can use round robin training loop which will allow the model to simultaneously learn various tasks. To deal with conflicting gradients in multi-task learning, we'll use a gradient surgery. For gradient surgery, we used the open-source Pytorch-PCGrad package Yu et al. (2020).

We'll then experiment with various training loop procedures (i.e. annealed sampling, round-robin, proportional sampling, etc.). Next, we'll test the efficacy of various training loss functions (i.e. SimCSE, MSE, cross entropy). Lastly, we'll evaluate the efficacy of SMART regularization.

All methods were originally coded outside of gradient surgery.

3.1 Baseline

To evaluate the efficacy of our final model, we used a simple BERT multitask model to encode our input with an architecture that is described in the paper "Attention is all you need" (Vaswani et al., 2018) and depicted in (Fig. 1) Vaswani et al. (2017). We then built different task specific layers on top of the base BERT model for our baseline, which we display in the appendix: Fig. 2), Fig. 3), Fig. 4).

To train our baseline model, we truncated our datasets to the length of the smallest dataset to get uniform dataset sizes and used task wise sequential training, which would train all batches in sentiment classification first, then all batches in paraphrase detection, then all batches in similarity score. This will serve as our point of iteration, allowing us to evaluate how our model additions alter or improve the final model's results.

3.2 Round Robin

To improve from our baseline approach, we first wanted to address the issue of "unlearning" the performance from old tasks when training on the new task and over fitting on later tasks compared to earlier tasks. Therefore, we employed a round robin training procedure, which allowed the model to learn and update the gradients of all three tasks in each step. In each training step, we sample a batch from each of the three datasets and sequentially train the model on each batch, optimizing the gradients of all three tasks. This is advantageous for multi-task learning as we are aiming to optimize a joint-optimization problem between all three tasks.

3.2.1 Truncated Triple-Batch Training

To solve the issues of over fitting on smaller datasets, we truncated all the datasets to the size of the smallest dataset, the SemEval dataset. While this approach meant that we did not use a large portion of our data for the Quora dataset, it was advantageous because no training example was seen more than once per epoch by the model, so it could avoid over fitting on smaller datasets.

3.2.2 Gradient Surgery

Although a baseline loss function for round robin training could be the sum of the loss functions for each task, conflicting gradients can make this approach sub-optimal. To address this, we implemented gradient surgery, which projects conflicting gradients onto the normal plane of other gradients to reduce inconsistent gradient updates in the process of optimizing multiple loss functions. This allows us to more efficiently optimize the parameters of our models to perform well in the multi-task setting (Fig. 6).

3.3 Sampling Methods

In addition to round robin training procedure, we tested 3 different sampling methods to train our model: Proportional Sampling, Annealed Sampling, and Square Root Sampling. These sampling methods aim to improve off round robin by preventing the over fitting of smaller datasets. For all three sampling methods, we took 2400 training steps per epoch, as proposed arbitrarily by Stickland and Murray (2019).

3.3.1 Proportional Sampling

Proportional sampling samples a batch from each task i with a probability proportional to the size of the dataset, namely:

$$p_i \propto N_i \tag{1}$$

where N_i is the number of training for task i.

3.3.2 Square Root Sampling

Square root sampling samples a batch from each task i with a probability proportional to the square root of the size of the dataset, namely:

$$p_i \propto N_i^{\frac{1}{2}} \tag{2}$$

where N_i is the number of training for task i. Square root sampling aims to help minimize interference in the training process, which occurs when one task gets sampled too many times in a row and the performance of other tasks drops. By using the square root probabilities, we reduce the disparity of sampling probabilities between the larger and smaller datasets, which we hypothesized would be helpful, given the SST training dataset is over $45x$ larger than the STS dataset.

3.3.3 Annealed Sampling

Annealed sampling samples a batch from each task i with a probability proportional to:

$$p_i \propto N_i^{\alpha} \tag{3}$$

where N_i is the number of training for task i. Following Stickland and Murray (2019), we set α to be:

$$\alpha = 1 - 0.8(\frac{e-1}{E-1}) \tag{4}$$

where e is the current epoch of training we are on and E is the total number of epochs we are training for. As seen in the equation, annealed sampling starts off with sampling from a distribution similar to proportional sampling, but as the training progresses the sampling between tasks becomes more evenly split. Stickland and Murray (2019) noticed that it became even more important to avoid interference later in the training process, which is why sampling from a more evenly split distribution among tasks would be beneficial.

3.4 Loss functions

In training our model, we experimented with various loss functions that could improve our performance. Initially, we tried cross entropy across all tasks and this served as our baseline. However, since STS is graded based on the logits' similarity with the ground truth score, we hypothesized that an MSE loss could outperform cross entropy for the STS task.

3.4.1 SimCSE Loss

SimCSE leverages a contrastive training procedure to learn more robust embeddings that are easily separable in the embedding space. In our paper, we leveraged unsupervised SimCSE in order as a pre-training methodology before fine-tuning the model again afterward. In unsupervised SimCSE, for each training example i, we run two forward passes from the MultitaskBERT to create two embeddings, v_i and v_i' where v_i and v_i' have slight variations from dropout noise. We then use the SimCSE objective function to maximize the similarity between these two examples and minimize the similarity across the other training examples. We run this method for 5 epochs, observing monotonically decreasing loss at each epoch.

$$\ell_i = -\log \frac{e^{\mathrm{sim}(\mathbf{h}_i^{z_i}, \mathbf{h}_i^{z_i'})/\tau}}{\sum_{j=1}^{N} e^{\mathrm{sim}(\mathbf{h}_i^{z_i}, \mathbf{h}_j^{z_j'})/\tau}} \tag{5}$$

3.5 Architecture

Although our architecture for predicting sentiment stayed the same throughout our research, we explored other alternative architectures from our baseline model for paraphrase detection and sentence similarity. For these two tasks, we tried concatenating the input ids and attention masks together before passing them into the BERT model and then passing that output through a linear layer as depicted in (Fig. 5). Throughout our paper, we refer to our baseline architecture as old and our modified architecture as new.

3.6 Smoothness-Inducing Adversarial Regularization

To prevent our model over fitting on training data, we also implemented Smoothness-Inducing Adversarial Regularization (SMART). As proposed by Jiang et al. (2020), SMART aims to add a regularization term to the loss function as such:

$$\min_{\theta} \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta), \tag{6}$$

where $\mathcal{L}(\theta)$ is the original loss function, λ_s is the regularization coefficient and

$$\mathcal{R}_s(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_p \leq \epsilon} \ell_s(f(\tilde{x}_i; \theta), f(x_i; \theta)), \tag{7}$$

For classification tasks such as the sentiment prediction and paraphrase detection, Jiang et al. (2020) proposed ℓ_s to be symmetric KL divergence.

$$\ell_s(P, Q) = \mathcal{D}_{\mathrm{KL}}(P \parallel Q) + \mathcal{D}_{\mathrm{KL}}(Q \parallel P); \tag{8}$$

For regression tasks, such as similarity score, Jiang et al. (2020) proposed ℓ_s to be squared loss, so we used mean squared error.

By minimizing the loss between very similar inputs as part of our loss functions, we are penalizing drastic changes in the outputs for small changes in inputs, which prevents over fitting. Given there was a bigger discrepancy between training loss and test loss for sentiment compared to paraphrase detection and similarity score, we implemented SMART with a higher lambda for sentiment than the other two tasks.

4 Experiments

4.1 Data

For the sentiment classification task, we fine-tuned our pre-trained BERT model on the Stanford Sentiment Treebank dataset (11,855 total examples | 8,544 train examples | 1,101 dev examples | 2,210 test examples), which consists of labeled, multi-class sentiment analysis examples. In order to fine-tune our model on paraphrase detection examples, we also used the provided Quora dataset (404,298 total examples | 283,010 train examples | 40,439 dev examples | 80,859 test examples) consisting of question pairs. Lastly, for the semantic textual similarity task, we fine-tuned our model on the SemEval STS Benchmark dataset (8,628 total examples | 6,040 train examples | 863 dev examples | 1,725 test examples) consisting of sentence pairs.

4.2 Evaluation method

In evaluating the efficacy of our methods, we used a variety of different evaluation metrics. For sentiment classification and paraphrase detection, we used classification metrics such as accuracy, precision, recall, and F-1 score. These gave us a holistic understanding of our model's strengths and weaknesses. For STS we evaluated our model's outputs using Pearson's correlation.

4.3 Experimental details

To fine tune the entire model, we used of $\alpha = 1 \times 10^{-5}$ with a batch size of 8. We trained everything on a NVIDIA Tesla T4 GPU. The cumulative training time for the full model ranged from 6 hours for a baseline to 8 hours for SimCSE. For our sampling taining procedures, we used 2400 steps per epoch.

4.4 Results

4.4.1 Round Robin Results

Table 1 shows how round robin improves on the baseline model, which was sequential task training on truncated dataset. Grad means gradient surgery was applied on this layer.

Table 1: Round Robin Table

	Sentiment Acc	Paraphrase Acc	Similarity Corr	Avg Acc
Baseline	0.469	0.713	0.540	0.574
RR (Truncated)	0.486	0.717	0.559	0.587
RR (Full)	0.476	0.706	0.577	0.586
RR (Trunc + Grad)	0.496	0.728	0.572	0.599

Table 1 shows that round robin improved off the sequential task training, as average accuracy improved from 0.574 to 0.587. It also shows that there was minimal difference between truncating the dataset (0.587) and running round robin all the way through the longest dataset (0.586). Finally, it shows that gradient surgery was able to improve the model from 0.587 to 0.599.

4.4.2 Sampling Methods Results

All tests in Table 2 were run on the baseline architecture, with the only variable being how we chose which batches would be trained on.

Table 2: Baseline Sampling Methods Table

	Sentiment Acc	Paraphrase Acc	Similarity Corr	Avg Acc
RR (Trunc + Grad)	0.496	0.728	0.572	0.599
Proportional	0.460	0.808	0.515	0.594
Annealed	0.480	0.804	0.581	0.622
Square Root	0.500	0.810	0.582	0.631

Table 2 shows that proportional sampling performed the worst as 0.594 was the lowest average accuracy of the four tests. Meanwhile, both annealed sampling (0.622) and square root sampling (0.631) did better than the best round robin test (0.599).

4.4.3 Loss Function Results

Table 3 tests cross entropy (CE) and mean square error (MSE) on the similarity score task. We tried these three tests on the two best sampling methods from table 2. All tests in table 3 still use the baseline model architecture.

Table 3: Loss Functions Table

	Sentiment Acc	Paraphrase Acc	Similarity Corr	Avg Acc
SQRT + CE	0.500	0.810	0.582	0.631
SQRT + MSE	0.480	0.783	0.437	0.567
Annealed + CE	0.480	0.804	0.581	0.622
Annealed + MSE	0.458	0.786	0.412	0.522

Table 3 shows that for both annealed sampling and square root sampling, cross entropy loss outperformed the mean squared error for the similarity task loss function as our average accuracies were higher for both. This went against we expected, as we thought that switching to MSE for a regression task would improve performance.

4.4.4 Model Architecture Results

Table 4 test our differing model architectures for the paraphrase detection and similarity score tasks. All tests in this table used all cross entropy loss functions.

Table 4: Model Architecture Table

	Sentiment Acc	Paraphrase Acc	Similarity Corr	Avg Acc
SQRT + Old	0.500	0.810	0.582	0.631
SQRT + New	0.481	0.861	0.795	0.712
Annealed + Old	0.480	0.804	0.581	0.622
Annealed + New	0.510	0.873	0.770	0.718

The main takeway from table 4 was that concatenating the input ids before passing them through BERT significantly improved average accuracy. We can see this as the square root sampling with this architecture (0.712) outperformed the square root sampling architecture (0.632) by 8 percent. Similarly, the annealed sampling with this architecture (0.718) outperformed the square root sampling architecture (0.622) by 9.6 percent.

4.4.5 SimCSE Pretraining Results

Table 5 compares our baseline architectures (new, old) with square root sampling with and without SimCSE.

Table 5: SimCSE Table

	Sentiment Acc	Paraphrase Acc	Similarity Corr	Avg Acc
Baseline Old + SQRT	0.500	0.810	0.582	0.631
SimCSE Old + SQRT	0.507	0.817	0.710	0.678
Baseline New + SQRT	0.481	0.861	0.795	0.712
SimCSE New + SQRT	0.492	0.867	0.801	0.720

We find that SimCSE boosts performance by up to 4.6% when compared to the baselines.

4.4.6 SMART Results

Table 6 test varying lambda values on the best model from table 4. Given sentiment exhibited the biggest discrepancies between train loss and test loss, we made those lambda values higher than the other two tasks i.e $\lambda = 3, 1, 1$ means a λ value of 3 for sentiment and 1 for the other two tasks.

Table 6: SMART Table

	Sentiment Acc	Paraphrase Acc	Similarity Corr	Avg Acc
SQRT, $\lambda = 0, 0, 0$	0.481	0.861	0.795	0.712
SQRT, $\lambda = 3, 1, 1$	0.522	0.874	0.815	0.737
SQRT, $\lambda = 5, 3, 3$	**0.522**	**0.877**	**0.830**	**0.743**
Ann, $\lambda = 0, 0, 0$	0.510	0.873	0.770	0.718
Ann, $\lambda = 3, 1, 1$	0.511	0.876	0.806	0.731
Ann, $\lambda = 5, 3, 3$	0.509	0.878	0.823	0.737

Table 6 shows that adding regularization to our model can improve performance by preventing over fitting. From this table, we can see that increasing lambda to 3 increased our performance in all 3 tasks, but when we changed lambda to 5 for sentiment analysis, the accuracy stayed at 0.522 for sqrt and went down from 0.511 to 0.509 for annealed. The bolded model is our final model with the best performance, which we have provided in the appendix (Table. 8).

5 Analysis

5.1 Sampling Method Analysis

Through our tests, we were able to come to the conclusion that square root sampling worked better than proportional sampling, the best round robin test, and annealed sampling. This is likely due to the training datasets we were using, which had sizes of $8,544$, $280,010$, and $6,040$. The results exemplify how square root sampling is able to strike a balance between not sampling with too much discrepancies and not over fitting on smaller datasets. We can see this because it has the biggest improvements on proportional sampling in sentiment accuracy and similarity correlation, which makes sense as those are the two smallest datasets, which get under sampled in proportional sampling. Meanwhile, square root sampling has the largest improvements on the round robin training on paraphrase accuracy because it still samples paraphrase examples more than the other tasks, which make sense, as that is the biggest dataset. Finally, it is interesting to note that while square root sampling had the biggest improvements in these areas, it beat round robin and proportional sampling in all three tasks.

5.2 Loss Function Analysis

The first additional loss function that we experimented with outside of cross entropy was MSE. We thought that for the regression task of STS where we were graded on Pearson's correlation, it could make more sense to use a loss function more native to regression analysis like MSE. Despite this intuition, our results were worse when training a model with MSE loss on the STS task. These results are reported in Table 3.

The major contribution in our loss function analysis was our implementation of unsupervised SimCSE from scratch. Here, we trained over the training examples using a contrastive SimCSE objective. We found that pre-training our models with the SimCSE objective yielded improvements in model performance. This is likely due to the SimCSE objective segmenting the embedding space into more separable classes, pushing together the v_i and v_i' while separating v_i and v_j, resulting in better downstream multitask performance. By pre-training with SimCSE and then training again using square-root sampling (our best performing training method), we are able to learn a more robust representation of the embeddings, resulting in better downstream multitask classificant accuracy.

5.3 Model Architecture Analysis

It was noteworthy that the most drastic improvement we made in the model was changing the architecture of the similarity score task and paraphrase detection. Once we passed the concatenated input ids and attention masks into BERT instead of passing them in separate and concatenating them after, our model improved a lot, especially in similarity score task. This is likely due to the fact that we were able to use the power of the BERT model to start learning relationships between the two inputs rather than solely relying on a linear layer after passing them individually through the BERT model to pick up on these complex relationships. This makes sense for why the similarity task would improve the most, as that is more complex than the binary paraphrase task. Therefore, while this change in architecture improved the paraphrase detection task, it's ability to leverage the BERT model to pickup complex relationships between the words was most heavily seen in the sentiment correlation improvements.

5.4 SMART Analysis

Given the discrepancy between the training and testing accuracies for sentiment was higher than the other two tasks, we had anticipated to see the most improvement in sentiment prediction. However, even with higher lambda values, it did not improve much more than the other two tasks. Additionally, it was interesting that annealed sampling performed better in the absence of SMART, but square root sampling performed better with SMART. Finally, switching the lambda value from 3 to 5 for the sentiment task did not show any improvements, which is noteworthy because the sentiment still was over fitting, as the training accuracy was above 0.75. Going forward, exploring other options to reduce over fitting could be beneficial for the sentiment task, such using datasets with more training data.

5.5 Final Model Analysis

We elected to evaluate our final model using SST since our final model struggled the most on SST compared to the other tasks. Despite solely looking at a few of the outputs, these will give us a rough idea of how our model tends to overestimate/underestimate sentiment. Looking at the outputs for the final model, we can generally see that the model's score closely represents the label's score. This is good as it indicates that our model's predictions are generally well-calibrated to the ground-truth answer distribution.
We found that generally the model can be confused by words such as "fun" causing it to over-inflate the sentiment score from the actual value of 2 to 3 for the fourth example. It's also possible that a similar thing is occuring in the third example as the word "like" causes the model to inflate the score from an actual value of 1 to 3. Although upon further inspection, this example is quite hard, even for a human, to properly measure.

Table 7: Final Model Analysis on SST

Input Text	Model Score	Actual Score
It's a lovely film with lovely performances by Buy and Accorsi	3	4
Enormously entertaining for moviegoers of any age	4	4
You won't like Roger, but you will quickly recognize him	3	1
Makes even the claustrophobic on-board quarters seem fun	3	2

6 Conclusion

Throughout the course of this project, we experimented with a variety of methods in order to boost the baseline performance of the multitask BERT model. We took four main approaches to boosting the performance of our model: training loop methods, architectural improvements, varying the loss functions, and SMART regularization. We then combined the best methods together to form our final model, resulting in a final average accuracy of **0.743**.

Despite these improvements, one potential limitation of our work is that we only experimented with unsupervised SimCSE. In order to improve our results, we could use supervised SimCSE, allowing our model to learn more nuanced differences between positive and negative training examples. Future research should focus on implementing supervised SimCSE as a method to boost the improvement of the multitask BERT model. Furthermore, we didn't get a chance to experiment with ensembling or meta-learning methods that could potentially improve the generalization, accuracy, and robustness of our final model. Additionally, we can run more hyperparameter optimizations in order to go beyond the given hyperparameters of the model.

7 Ethics Statement

One major ethical concern is the potential for biased outputs, as these models may inherit and amplify biases present in the training data, leading to unfair or discriminatory results. Additionally, as we saw through our results, our model is not perfect, so there is a risk of the spread of misinformation. To mitigate these risks, it is essential that the training data is as unbiased and representative as possible. To do this, we need to both filter out any training examples that could reinforces biases and ensure that we have enough examples from a diverse set of sub populations. To mitigate the risk of misinformation, it is important that we understand the limitations of the model before using it in the real world. For example, if it is crucial that we achieve 100 percent accuracy on a sentiment task, then our model might not be sufficient for that use case. Finally, continuous monitoring and evaluation of the models' performance and impact on different populations are crucial to address and solve any unintended consequences.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Association for Computational Linguistics (ACL)*.

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

Haoming Jiang, Xiaodong Liu, Jianfeng Gao, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Association for Computational Linguistics*.

Asa Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the International Conference on Machine Learning*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Conference on Neural Information Processing Systems*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Conference on Neural Information Processing Systems*.

A Appendix (optional)

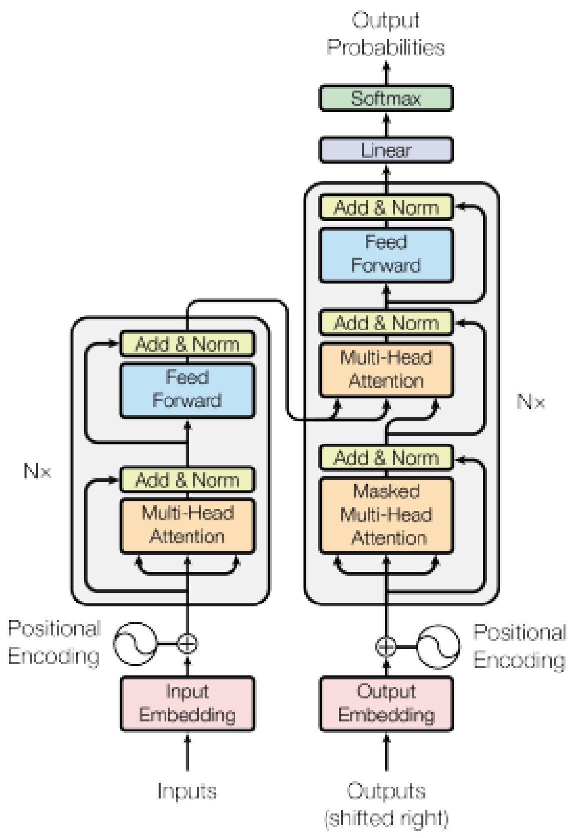

Figure 1: BERT Model Architecture

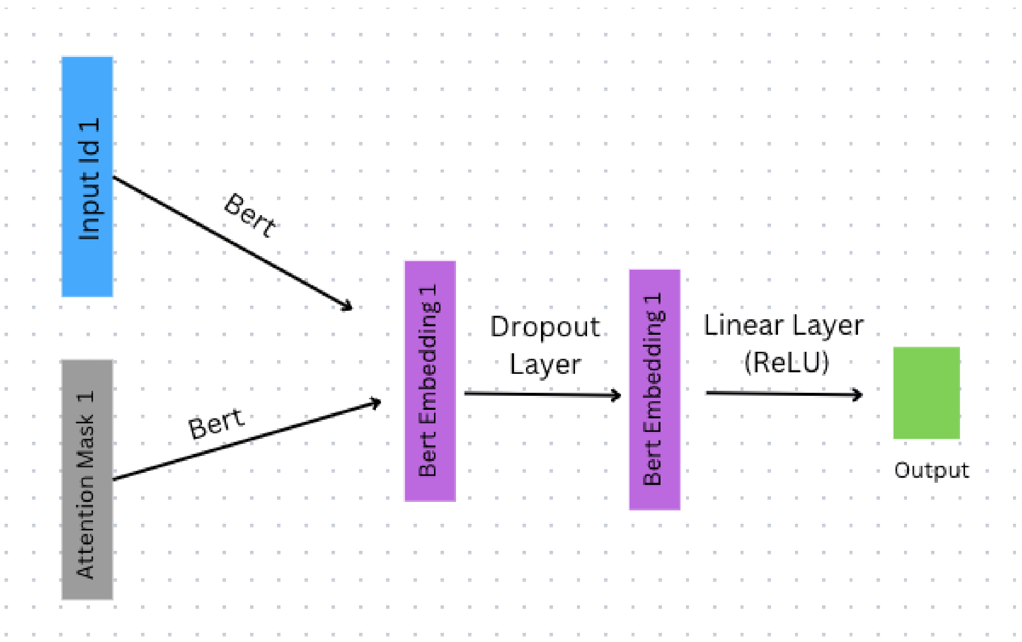

Figure 2: Sentiment Prediction Architecture

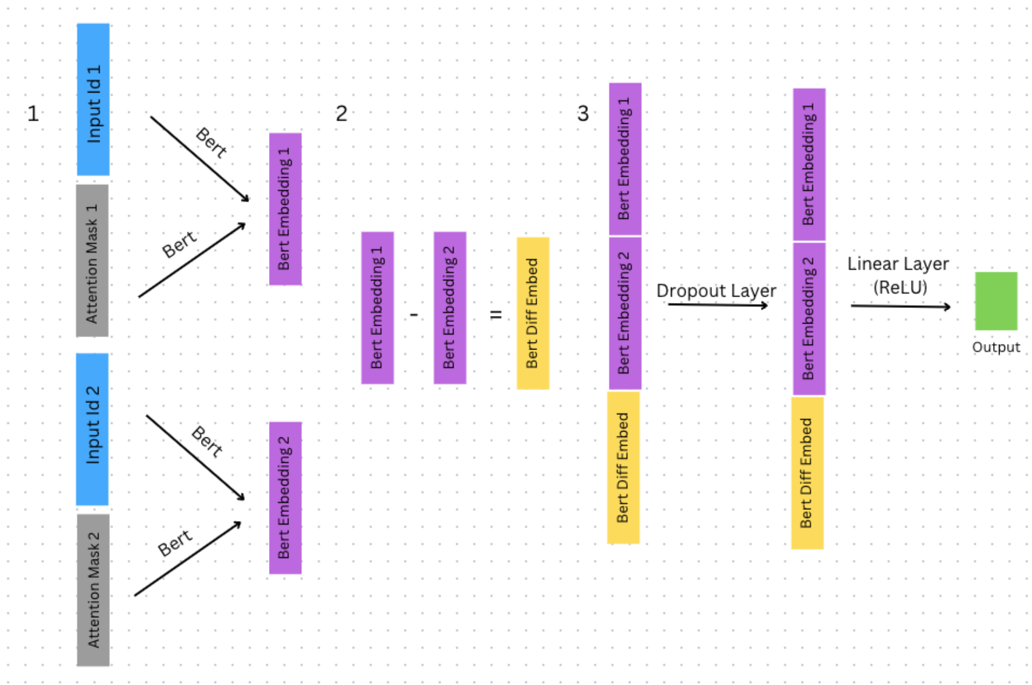

Figure 3: Paraphrase Detection Baseline Architecture

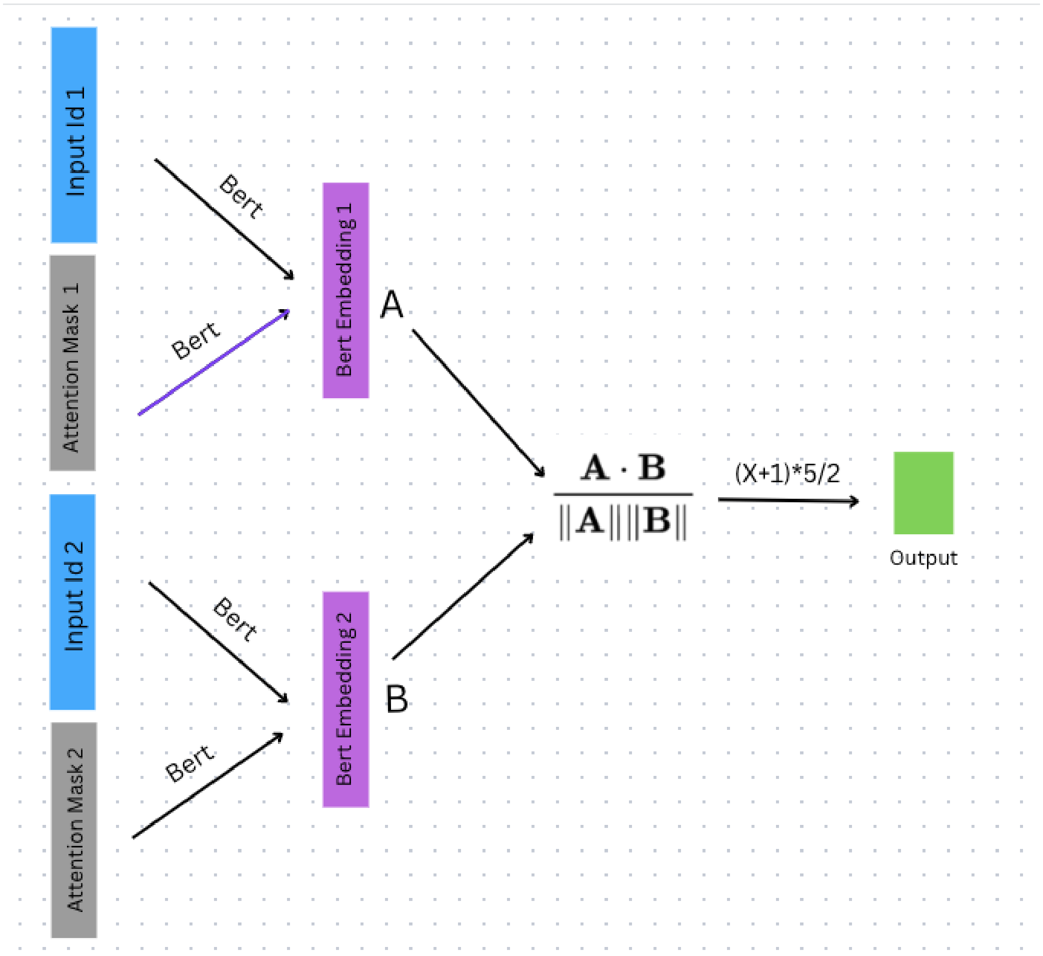

Figure 4: Sentence Similarity Baseline Architecture

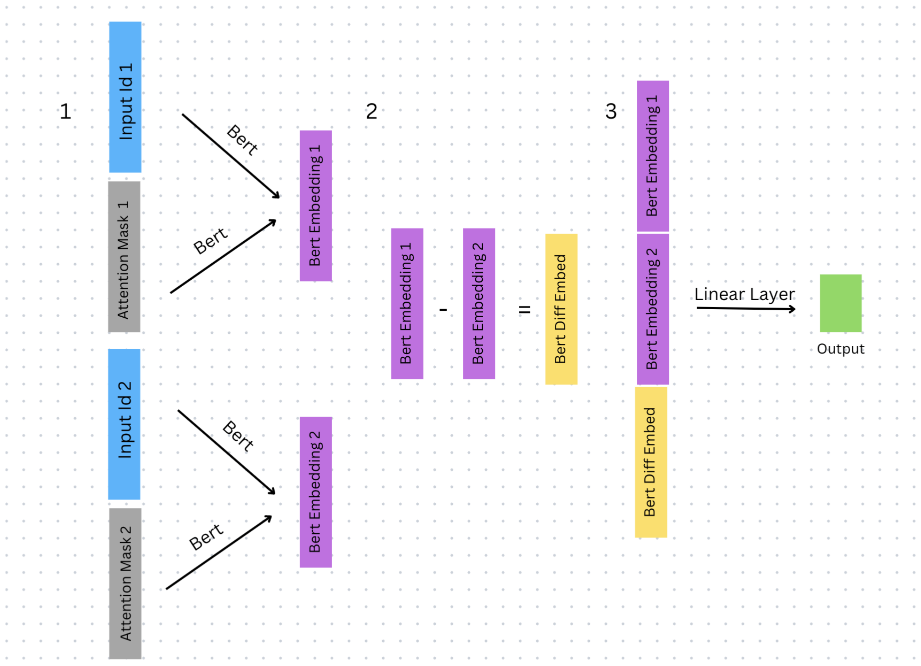

Figure 5: Unsupervised SimCSE visualization

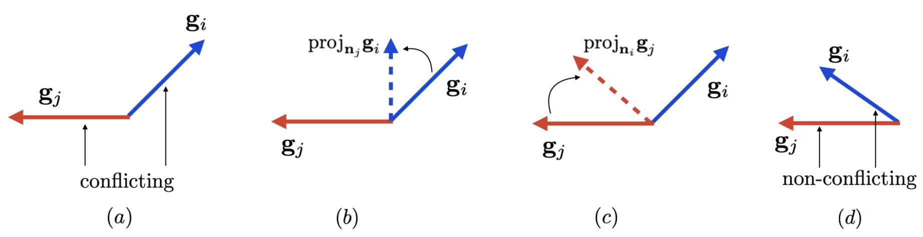

Figure 6: Gradient surgery visualization

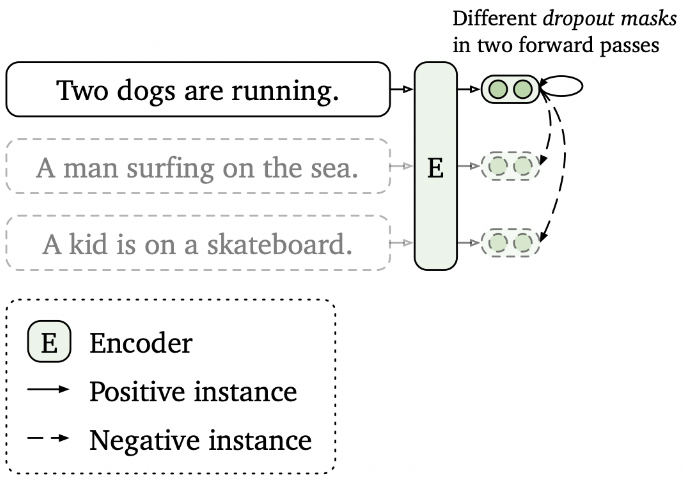

Figure 7: Unsupervised SimCSE visualization

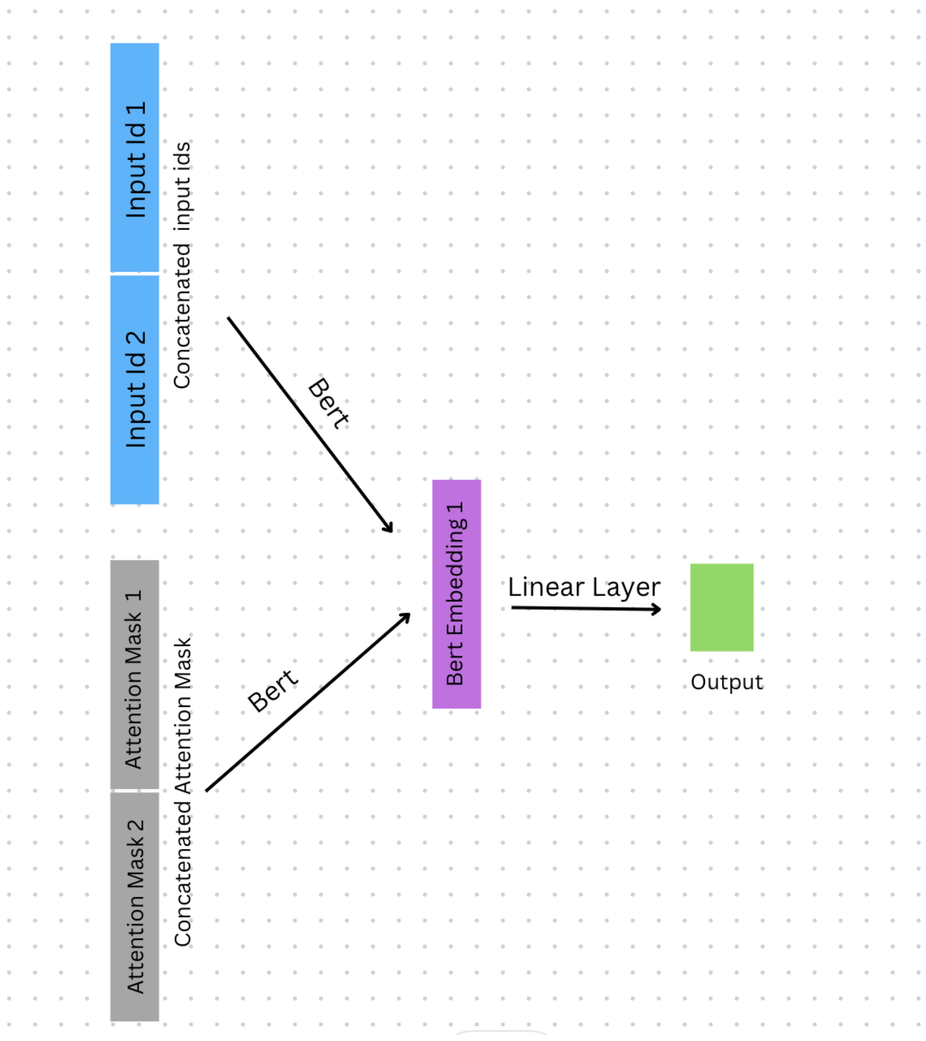

Figure 8: Paraphrase Detection and Sentence Similarity New Architecture

Table 8: SMART Table

	Sentiment	Paraphrase
Accuracy	0.522	0.877
Precision	0.532	0.532
Recall	0.498	0.878
F-1 Score	0.496	0.870