# Improving MiniBERT's Semantic Performance with Semantic-rich Sentence Embeddings

Stanford CS224N Default Project

**Melvin Orichi Socana**
SUNet ID: msorichi
Electrical Engineering
Stanford University
msorichi@stanford.edu

## Abstract

We have developed an Sentence-BERT (SBERT) transformer model that achieves performance on sentiment classification, paraphrase detection, and semantic textual similarity (STS) tasks comparable to BERT. SBERT is a method for generating high-quality sentence embeddings to provide a way for machines to understand the meaning of sentences and to compare them with one another. Theoretically, SBERT should garner higher performance on tasks that require a model to view multiple sentence embeddings such as in paraphrase detection and STS. In addition to SBERT, we have implemented various significant changes both to the model and training algorithms to increase performance and to experiment with more advanced techniques. Experiments with our implementation of SBERT and subsequent additions have shown slightly better performance on tasks that can benefit from sentence embeddings. Other additions to the standard baseline BERT model include experimenting with L2 regularization, adversarial regularization, additional datasets, and loss functions for their respective tasks.

## 1 Information

- Mentor: Cathy Yang
- Sharing project: Yes

## 2 Introduction

Pre-trained transformer-based deep language models have revolutionized natural language processing (NLP) since their introduction in 2018. Among these models, BERT has achieved state-of-the-art performance on a wide range of NLP benchmarks and tasks. However, while BERT and its variants have shown impressive results on many tasks, there is still much room for improvement, particularly on more complex downstream NLP tasks.

In recent years, researchers have focused on optimizing and building on top of BERT to achieve even better performance. One promising approach is Sentence-BERT (SBERT), a method for generating high-quality sentence embeddings that can be used to compare and contrast sentences [1]. SBERT has become a popular library for NLP practitioners, as it can be applied to a wide range of tasks and languages. Theoretically, SBERT should garner higher performance on tasks that require a model to view multiple sentence embeddings such as in paraphrase detection and STS.

In this paper, we present a novel approach that combines state-of-the-art model architecture with several optimization strategies to achieve better performance on three important NLP tasks: sentiment

analysis, paraphrase detection, and semantic similarity classification. Our approach includes SBERT, adversarial regularization, L2 regularization, and the inclusion of an additional dataset to improve performance. We evaluate our approach using the Stanford Sentiment Treebank [2], Quora dataset[3], and SemEval STS Benchmark dataset [4].

Our work builds upon prior research that has focused on improving BERT-based models for complex NLP tasks. By combining state-of-the-art model architecture with a range of optimization strategies, we aim to push the limits of performance even further. Our results demonstrate that our approach outperforms our baseline miniBERT model on the paraphrase detection, and semantic similarity classification tasks. While our study is not the first to explore optimization strategies across different fields, our approach contributes to the ongoing effort to improve NLP models and highlights the potential of combining multiple strategies for better performance. Our work contributes to ongoing efforts to improve NLP models and demonstrates the potential of combining optimization strategies for better performance.

## 3 Related Work

### 3.1 Sentence-BERT (SBERT)

Recent years have seen significant developments in pre-trained, transformer-based deep language models for natural language processing (NLP) tasks. Among them, BERT has been particularly successful, achieving state-of-the-art performance on various NLP benchmarks [5]. However, while BERT models have shown impressive results, their high computational cost limits their practical use in real-world applications.

To address this issue, Reimers and Gurevych proposed a method called Sentence-BERT (SBERT) that uses trained sentence-level embeddings to reduce the computational overhead [1]. The SBERT model utilizes a siamese architecture, where two parallel BERT models share weights and produce embeddings for each sentence in a pair. These embeddings are then processed by a pooling layer to obtain semantically-rich sentence-level embeddings, which can be used for tasks such as classification or similarity comparison. The downside of this method is that training takes considerably longer due to the introduction of an additional BERT layer.

Since its introduction, SBERT has gained significant attention in the NLP community and has been successfully applied to various tasks such as paraphrase detection, information retrieval, and sentiment analysis. Several studies have investigated the effectiveness of different pooling strategies and have proposed various modifications to the SBERT model to improve its performance on specific tasks [6, 7].

Overall, SBERT represents a significant advancement in NLP research and has shown promising results in various applications, highlighting the importance of using semantically-rich sentence embeddings for reducing computational overhead while maintaining high performance.

### 3.2 Adversarial Regularization

Adversarial regularization has been applied to various deep learning models, including BERT and its variants, to improve their generalization performance on various NLP tasks. One notable application is the "Adversarial Dropout Regularization" proposed by Li et al., where dropout is replaced by a stochastic adversarial perturbation of the input during both training and testing [8]. This method is shown to be effective in improving the robustness of BERT models against adversarial attacks, while maintaining or even improving their performance on clean data.

In addition, Xu et al. propose a method called "Adversarial Learning with Semantic Perturbation" (ALSP) to enhance the semantic robustness of BERT models [9]. The method introduces a semantic

perturbation module that generates adversarial examples by incorporating word-level synonyms and antonyms, which are designed to preserve the original semantics of the input. The authors demonstrate that ALSP can significantly improve the robustness of BERT models on various semantic-level attacks.

Furthermore, some recent studies have explored the combination of adversarial regularization with other techniques, such as learning rate decay and weight decay, to further improve the performance of BERT models.[10, 11] These studies demonstrate the effectiveness of the combined approach in achieving state-of-the-art performance on various NLP tasks.

# 4 Approach

## 4.1 Baseline Model

For our baseline model, we are using a basic minBERT model. Designed to be a simplified implementation of BERT, it includes a tokenizer and base embedding layer, which includes both token embeddings and positional embeddings, followed by a stack of 12 BERT transformer layers. Each BERT layer consists of a multi-headed attention layer followed by an add-norm layer, then a feed-forward layer followed by a second add-norm layer. The model is trained with an AdamW optimizer, an extension of Adam that introduces weight decay as a form of regularization. This allows for more fine-grained control over the amount of weight decay applied to each layer of the network, which can improve performance on some tasks. More details can be found in the 2023 CS224n default project handout.[12]

## 4.2 Implemented Model

Starting with the baseline minBERT model as the core of our implementation, we integrated various additions to the baseline model. The architecture for sentiment analysis has been largely left untouched from the baseline. This is because the SBERT architecture does not have a great effect on tasks that only require a single sentence embedding. As such, the sentiment analysis task is simply composed of a single BERT model, a dropout layer, and 2 dense layers.

**SBERT Implementation:**

We refer directly to the SBERT paper [1] to develop our model. In SBERT, for tasks that require a model to examine multiple sentence embeddings a separate BERT model is used for each embedding. For our case, two parallel BERT models share weights and generate embeddings for each sentence in a pair. This allows the model to learn a more robust representation of each sentence, as it is forced to encode information from each sentence independently, while still being able to capture any underlying relationship between the sentences. The output of these parallel models is then fed into a pooling layer to generate a single, fixed-size representation for each sentence. We implement this architecture for both paraphrase detection and semantic textual similarity. For sentiment analysis, a single BERT model is used that is separated from the other tasks to prevent conflict.

A slightly different architecture is required between paraphrase detection and STS due to paraphrase detection being a classification task and STS being treated as a regression task. For paraphrase detection, we concatenate both sentence embeddings and their absolute difference to capture information relating to each sentence and their relation to each other. Here, we deviate from the original proposed architecture. Rather than applying the softmax, we use a binary cross entropy loss instead as there are only two classes. This is a valid approach because the binary cross entropy takes in the logit output of the model and applies a sigmoid function to map the output to the range [0,1], which can be interpreted as the probability of the positive class. For STS, we follow the original SBERT paper for a regression task where the cosine similarity between both pooled outputs is outputted between the values [-1,1] and then we subsequently apply an MSE loss as our loss function.

Lastly, we tack on some dense layers with a ReLU activation functions and a few dropout layers to improve the robustness of the model.

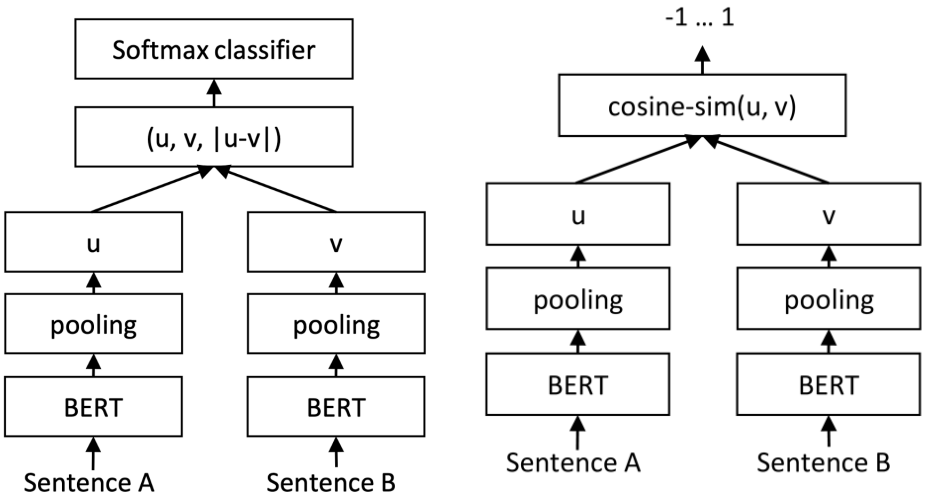


Figure 1: Visualizations of the original SBERT architecture with classification objection function (left) and regression objective function (right). Note that in our architecture the softmax classifier is not used. Taken from the SBERT paper [1].

**Adversarial Regularization:** To implement adversarial regularization, we added a new function to create perturbed version of the current batch of data. Adversarial regularization works based on the following equation:

$$\min_{\theta} \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_{\mathrm{s}} \mathcal{R}_{\mathrm{s}}(\theta) \tag{1}$$

Here, $\mathcal{L}(\theta)$ is the loss term for the given task, $\mathcal{R}_{\mathrm{s}}(\theta)$ is the adversarial regularization term, and $\lambda_{\mathrm{s}}$ is a scaling factor hyperparameter). The adversarial regularization term is computed by taking the loss of the model on perturbed data points. Perturbations are generated by adding a uniformly randomly generated number to existing data points. By forcing our model to handle inputs slightly outside the training examples, adversarial regularization is achieved.

In addition to adversarial regularization, we also apply a small L2 regularization during training which has consistently been used to prevent overfitting in models.

**Learning Rate Decay:** In addition to the learning rate scaling done as part of the AdamW optimizer (see the AdamW paper [13] for more details), we implemented our own form of learning rate scaling and early stopping mechanisms. After the model has failed to improve on the validation set for a few epochs, learning rate is halved. If the validation set continues to fail, the model is stopped. This halving of the learning rate is required because if the model continues to train while not improving on the validation set, it indicates that overfitting is occurring and the model is not generalizing well.

# 5 Experiments

## 5.1 Data

The datasets we used are as follows:

| Data Source | Train | Dev | Test | Set Source |
|---|---|---|---|---|
| Stanford Sentiment Treebank | 8,544 | 1,101 | 2,210 | Default Project |
| Quora Dataset | 141,506 | 20,215 | 40,131 | Default Project |
| SemEval STS Dataset | 6,041 | 864 | 1,726 | Default Project |
| SICK Dataset | 4,439 | 495 | 4,906 | Marelli et al., 2014 |

Table 1: Information on the datasets used in this project.

Further information about the first 4 datasets can be found in the 2023 CS224n default project handout.[12] The SICK dataset consists of labeled sentence pairs where each sentence pair is given a relatedness score in the range 1-5. This is different from the SemEval STS dataset where each sentence pair is given a relatedness score in the range 0-5. As such we scaled the STS dataset to be in a range similar to the SemEval set. Unfortunately, due to the very different distribution compared to the SemEval dataset, it did not improve results for the STS task. As such, we used it to supplement training on the semantic similarity task by treating it more as an additional method to prevent overfitting by scaling its loss down via a scaling hyperparameter. We originally wanted to integrate this dataset due to the few training examples in the SemEval dataset. This dataset was obtained from the Hugging Face datasets library.[14]

## 5.2 Evaluation method

We evaluated our model quantitatively, using a series of scores across the various datasets. For the SST, and Quora dataset, our metric was accuracy. The SST task required classification between 5 varying positive/negative ratings, and the accuracy metric determined how many of the predicted labels matched the actual label. For the semantic similarity task on the STS dataset, performance was measured based on the Pearson correlation of the predicted labels compared to the true value, ranging from -1 (total negative correlation) to +1 (total positive correlation). In addition, we computed more advanced metrics, like precision, recall, and F1 score for sentiment analysis and STS tasks. For STS F1 score because it is treated as a regression task, we use a threshold of $\pm$ 0.2 to denote a true

positive. Note that these metrics were imported from sklearn [**?** ] and integrated into the evaluation code ourselves so they may not be accurate. For paraphrase detection these metrics failed to appear properly and as such we will not include them in our results.

The completed model was submitted to the 2023 CS224n Default Project leaderboard, wherein all default project teams would submit and compare performance across the three tasks.

### 5.3 Experimental details

In our final implementation of the SBERT model:

the adversarial regularization scaling parameter $\lambda_A = 0.01$

the l2 regularization scaling parameter $\lambda_{L2} = 0.01$,

learning rate $l_r = 10^{-3}$, learning rate halving mark = 5 epochs no improvement

batch size = 16

Dropout layer probability = 0.3

Epochs = 10

After obtaining the concatenated pooled outputs for paraphrase detection, the output is fed through two dense layers of sizes (BERT HIDDEN SIZE*3, 256) and (256, 1). Paraphrase detection took a very long time to train (1 hour 30 minutes per epoch) and as such we were only able to train on paraphrase detection for a few epochs. The other tasks were very fast to train, together they took about 3 minutes per epoch

### 5.4 Results

| | Paraphrase Accuracy | Sentiment Accuracy | Similarity Accuracy |
|---|---|---|---|
| Baseline Dev | 0.446 | 0.517 | 0.007 |
| SBERT Test | **0.745** | 0.520 | **0.348** |

Table 2: We compare results on the three tasks with the minBERT baseline and our implemented multiclassifier model. The SBERT model results are from the test leaderboard.

There was a significant improvement in the performance of paraphrase detection and STS scores. Sentiment accuracy remained the same as no modifications were made to target that task. Ideally, we hoped these results would be higher. We believe that the lack of stronger results is due to the lack of pretraining of the model and lack of tuning hyperparameters. Validation accuracy for the STS

| | Precision | Recall | F1 Score |
|---|---|---|---|
| SBERT noReg Sentiment Analysis | 0.512 | 0.490 | 0.477 |
| SBERT noReg STS | 0.786 | 0.886 | 0.833 |
| SBERT AdvReg Sentiment Analysis | 0.506 | 0.502 | 0.502 |
| SBERT AdvReg STS | 0.868 | 0.932 | 0.899 |

Table 3: We examine precision, recall, and F1 score on both sentiment analysis and STS. The STS evaluation has a threshold=$\pm$0.2

.

## 6 Analysis

Our model significantly improves on the baseline for both paraphrase detection and STS, but does not provide a significant improvement in sentiment analysis from the baseline despite our numerous

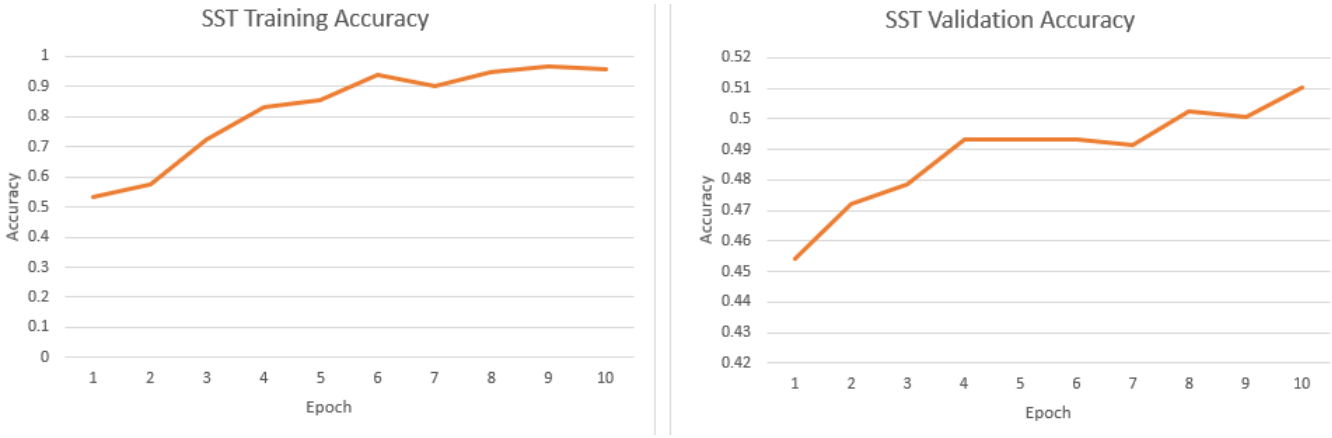


Figure 2: SST Training/Validation Accuracy

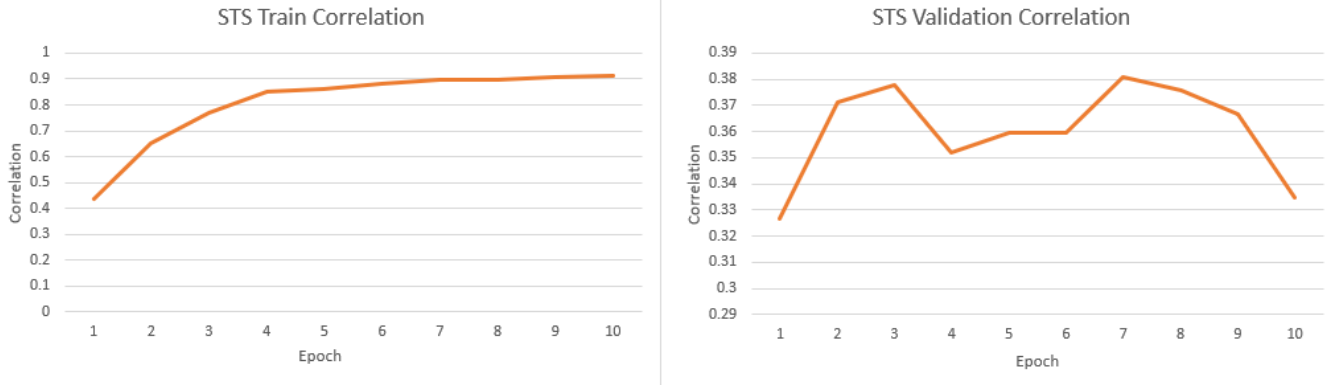


Figure 3: STS Training/Validation Correlation

modifications. This has resulted because we did not make any modifications to the sentiment analysis training or the layers that the sentiment analysis task interfaces other than a few dense layers and a dropout.

Our training and validation graphs for the STS task are strange. The validation correlation appears to bounce up and down which could be the result of various factors. Some things to consider are: the small dataset used for STS, overfitting, training data not being very indicative of the validation data, or a misconfiguration of hyperparameters. We had not considered it before, but it would have been pertinent to give each task a different learning rate depending on this issue. The very small dataset we were given also was likely very damaging to our results. We attempted to integrate more data like the SICK dataset, but unfortunately, this did not result in a higher correlation.

With a threshold of 0.2 for STS, we received a significant increase to metrics. This indicates that the actual evaluation code for accuracy has a very strict threshold compared to our chosen value. Although these metrics are not officially used in this project, we thought it would be interesting to examine them.

Although SBERT should theoretically greatly improved performance on tasks that require a model to examine multiple sentence embeddings, our results were not as impressive as we anticipated. We believe the major key reason as to why our results did not greatly improve on STS and paraphrase detection was due to not conducting any pretraining on our model. Unfortunately, we ran out of time to implement a pretraining function with an MLM objective. We additionally did not have time to introduce a hyperparameter tuning algorithm and from our previous experiments with different learning rates the values of each hyperparameter has drastic effects on our model implementation.

## 7 Conclusion

We learned many useful ways to increase performance of a model such as regularization. We also were able to gain a greater understanding of the various loss functions and the appropriate times to use each one. Lastly, it was very educational to implement a different form of the BERT architecture and dive deeper into how it functions and garners higher performance.

Overall, SBERT did have some significant performance increases compared to the baseline BERT model trained on sentiment analysis; however, there are many improvements that could have been made to increase the performance of our model. Additional pretraining, ensembling, potentially trying out different loss functions, and hyperparameter tuning were all significant ways to increase robustness of the model.

Our primary limitation for this project was the time it took to train on the paraphrase dataset. In future projects, we would recommend trying out a slightly smaller version of this model to make training faster.

## References


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

[2] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, pages 1631–1642, 2013.

[3] Shankar Iyer and et al. First quora dataset release: Question pairs. In *Proceedings of the 2nd workshop on data science for social good*, pages 3–10, 2017.

[4] Iris Hendrickx and et al. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 33–38, 2010.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *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, 2019.

[6] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525, 2020.

[7] Urvashi Khandelwal, Kushagra Khanna, Manohar Kaul, and Partha Talukdar. Sentilare: Generating contrastive sentences for analyzing embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3599–3608, 2020.

[8] Chunyuan Li, Hao Liu, Changyou Chen, Yunchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. Dropout improves generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[9] Zeyu Xu, Shuang Zhang, Hao Zhang, and Wei Lu. Adversarial learning with semantic perturbation for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4956–4966, 2020.

[10] Xiaojie Wang, Yizhe Zhang, Zhe Wang, Jiawei Zhang, Zhe Gan, Zhe Lin, Xiaodong Sun, and Jingjing Liu. Adversarial training with decayed adversarial examples. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1125–1135, 2019.

[11] Yang Gao, Xiaojie Wang, Shouling Ji, Yisen Wang, and Yu Gong. Adversarial weight decay regularization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7465–7476, 2020.

[12] Cs 224n: Default final project: minbert and downstream tasks, 2023. Accessible via `web.stanford.edu/class/cs224n/project/default-final-project-bert-handout.pdf`.

[13] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[14] Huggingface sick dataset. 2023. Accessible via `huggingface.co/datasets/sick`.