Enhancing BERT for NLP Tasks: Pretraining, Fine-tuning, and Model Augmentation

Stanford CS224N {Default} Project

Name
Department of Computer Science
Stanford University
`dylanrd@stanford.edu, bperk25@stanford.edu`

Abstract

This project aims to enhance BERT's performance in Semantic Textual Similarity (STS), Paraphrase Detection, and Sentiment Analysis tasks through additional pre-training, fine-tuning with cosine similarity, and model size expansion. Inspired by "Sentence-BERT," our goal is to assess whether these enhancements significantly improve accuracy and efficiency. We used diverse datasets for training and evaluation, comparing against existing benchmarks like BERT, STSb: STS benchmark, STS12-STS16, and SICK-R: SICK relatedness dataset. Through this work, we seek to progress NLP research by improving deep learning model effectiveness in real-world applications.

1 Introduction

Sentiment analysis is an important task in natural language processing, having to accurately identify the underlying sentiment or opinion expressed within textual data. This task has previously been approached through various methods including lexicon-based methods, relying on predefined sentiment lexicons to assign sentiment scores to words or phrases, and machine learning-based methods, employing algorithms to classify text into predefined sentiment categories. In this project, we aim to enhance sentiment analysis leveraging a multitask BERT model, a versatile adaptation of the original BERT architecture designed to concurrently address sentiment classification, paraphrase detection, and semantic textual similarity tasks. By fine-tuning this multitask model, we seek to improve its capability to discern subtle sentiment distinctions in text, potentially reducing computation time by sharing representations across tasks. Our approach draws inspiration from recent advancements in sentiment analysis research , particularly those utilizing SBERT-based models, to develop and evaluate our proposed enhancements (Reimers and Gurevych, 2019).

2 Related Work

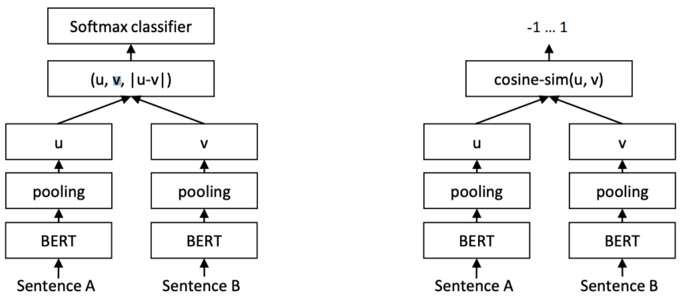
Previous work have aimed to solve a persistent issue encountered when utilizing BERT and RoBERTa models, where processing both sentences concurrently causes substantial computational overhead. This challenge interferes with the creation of independent sentence embeddings, requiring alternative solutions. Researchers have explored passing individual sentences through these models and deriving fixed-size vectors using methodologies such as averaging the outputs or leveraging the distinctive [CLS] token. In response to this issue, some propose Sentence-BERT (SBERT), an enhanced iteration of BERT designed to streamline the generation of sentence embeddings. SBERT implements siamese and triplet network architectures, which play pivotal roles in facilitating efficient computation and ensuring the preservation of semantic meaning. Through rigorous experimentation, SBERT significantly truncates the time required to identify the most similar pair of sentences, drastically reducing it from a staggering 65 hours with traditional BERT/RoBERTa models to a mere 5 seconds, all while maintaining the high accuracy standards set by BERT. The siamese network architecture

embedded within SBERT proves instrumental in deriving fixed-size vectors for input sentences, thereby enabling the identification of semantically akin sentences using established measures such as cosine similarity or diverse distance metrics. SBERT's effectiveness stems from its integration of a pooling operation into the output of BERT/RoBERTa. This feature allows for the exploration of multiple pooling strategies, including leveraging the [CLS] token output, computing the mean of all output vectors, or determining a max-over-time of the output vectors, with the mean strategy emerging as the default choice.
Although most previous works share some limitations - such as the reliance on English-only training data which may restrict SBERT's applicability in multilingual settings, and the absence of in-depth analysis on model mistakes, leaving room for uncertainty regarding SBERT's robustness - the achieve their primary goal of enhancing efficiency without compromising accuracy. Therefore, SBERT remains a valuable tool in natural language processing tasks, yet future research should address the identified limitations and delve deeper into the model's performance.

3 Approach

3.1 Architecture

The architecture of our MultitaskBERT model is structured with BERT at its core, followed by a shared fully connected (FC) layer that maintains the original BERT hidden size (768 dimensions for bert-base-uncased). This shared layer is crucial as it acts as a bridge between the generic representations learned by BERT and the task-specific adaptations required for our diverse set of tasks. On top of this shared layer, we introduce task-specific heads: a linear layer for sentiment classification that maps to the number of sentiment classes, a linear layer for binary paraphrase detection, and another linear layer for computing a continuous similarity score in semantic textual similarity tasks. Each of these heads is designed to interpret the shared representations in the context of its respective task, allowing for specialized learning without losing the benefits of cross-task knowledge transfer. After calculating the sentence embeddings, cosine similarity is used to measure the similarity between pairs of sentences. This computes the cosine of the angle between two vectors, providing a measure of their similarity regardless of their magnitude. Our architecture remains similar to the graphic below, except that our cosine similarity is within a range [0, 1] so we can assign sentiment scores easily to the resulting similarities.



3.2 Baselines

The best extractive baselines for Semantic Textual Similarity (STS) tasks, focusing on both Avg. BERT embeddings and BERT CLS-vector models exhibit, vary across different benchmarks. The Avg. BERT embeddings model achieves scores ranging from 38.78 to 63.15 on STS12 to STS15. The BERT CLS-vector model demonstrates lower scores across all benchmarks, ranging from 20.16 to 42.63, having lower performance in capturing relationships between sentence pairs.

Model	STS12	STS13	STS14	STS15	STS16	STSb	SICK-R	Avg.
Avg. BERT embeddings	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT CLS-vector	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19

Table 1: Spearman rank correlation ρ between the cosine similarity of sentence representations and the gold labels for various Textual Similarity (STS) tasks. Figure comes from Reimers and Gurevych (2019)

4 Experiments

4.1 Data

We will be using the SST dataset, IMDb movie review dataset, and the Stanford Sentiment Treebank for sentiment analysis, the Quora dataset for paraphrase detection, and the SemEval dataset for semantic textual analysis. These are all preexisting datasets that we will access open source.

4.2 Evaluation

For evaluation, we will use the Pearson correlation coefficient as a well-defined, numerical metric to assess the performance of our enhanced BERT model in tasks such as Semantic Textual Similarity (STS) and Paraphrase Detection. Specifically, we will compare against the performance achieved by the original BERT model and other models, such as InferSent, Quick-Thoughts, and USE, on these tasks. Additionally, for Sentiment Analysis, we will utilize accuracy as an evaluation metric and compare our model's performance against existing benchmarks like the IMDb movie review dataset or the Stanford Sentiment Treebank. For qualitative evaluation, we conduct error analysis to identify patterns and potential areas for improvement by examining sample predictions made by our model and comparing them with human-labeled data.

4.3 Experimental details

For all of the tasks, our settings remained the same: Learning Rate: $1e-5$ Batch Size: 8 Epochs: 10 Optimizer: AdamW with default parameters. Regularization: Dropout with a probability of 0.3

However, we experimented with a model using Adagrad optimizer while the rest of the settings remained the same.

4.4 Results

4.4.1 Extensions

We first implemented a naive implementation of the multitask classifier by leveraging the existing BERT architecture. This implementation had 3 separate linear layers, simple neural networks, for each of the downstream tasks: sentiment classification, paraphrase detection, and semantic text similarity. The embeddings were gathered for each of the three tasks by processing them through BERT and the respective linear layer, and output predictions for that task.

After gathering results for this naive approach using the base BERT, we explored our first extension of using cosine similarity for semantic similarity between words. We gathered the embeddings from the forward pass and then normalized the embeddings before calculating the cosine similarity and scaling the results by a factor of (cosine similarity + 1) * 2.5. This is due to cosine similarity being calculated in a range [0, 1] so we scale this result up to match the 5 sentiment classes 0-5.

Our second extension involved additional finetuning on the ids-sst-train, quora-train, and sts-train datasets. While in the "pretrain" mode the parameters of the BERT model are frozen and only the layers added for the specific tasks are trained, in the "finetune" mode, these BERT parameters are unfrozen and updated during training. This fine-tuning allows the model to adjust the pre-trained BERT embeddings more closely to the specific characteristics and nuances of the tasks at hand. We saw a substantial increases in accuracy in the paraphrase and STS tasks as shown in the tables below.

Our third extension involved sharing a layer between our tasks. The shared layer in consists of a linear layer followed by a ReLU activation function and dropout, which is designed to process the embeddings from the BERT base model before task-specific prediction. The aim of this shared layer is to create a common representation space that captures features relevant to all the tasks the model is being trained on, namely sentiment classification, paraphrase detection, and semantic textual similarity. By using a shared layer, the model leverages the commonalities among these tasks, which can lead to more robust and generalized feature extraction. This approach also helps in parameter efficiency, as it reduces the need for separate, task-specific feature extraction layers for each task, thereby streamlining the model architecture and potentially improving its learning capabilities across different tasks. We saw a great increase in the paraphrase accuracy, however the other STT task suffered a significant hit to overall accuracy.

Furthermore we experimented with a version of the model using the Adagrad optimizer instead of AdamW with a shared layer. The Adagrad optimizer is an algorithm for gradient-based optimization that adapts the learning rate for each parameter, giving larger updates to infrequent parameters and smaller updates to frequent ones. This feature makes Adagrad particularly suitable for dealing with sparse data and features, as it can dynamically adjust how much each feature contributes to the learning process based on its frequency. However, this led rather subpar results with no improvements in any category.

4.4.2 Quantitative Analysis

Based on our our second extension (finetuning) to our final extension (shared layer), we saw these changes: Sentiment Classification: Minimal Change: This suggests that the shared layer does not significantly impact the sentiment classification task. It's possible that sentiment classification, being potentially more reliant on specific lexical cues than the other tasks, gains little from the shared contextual embeddings that are influenced by the needs of the other tasks.

Paraphrase Detection: Significant Improvement: The accuracy for paraphrase detection sees a substantial increase from 0.626 to 0.737 with the shared layer. This improvement indicates that paraphrase detection benefits greatly from the shared representations. The task of detecting paraphrases likely shares more commonalities with the STS task, as both involve assessing the relationship between pairs of sentences. The shared layer helps the model to better generalize across these tasks by learning representations that capture semantic similarities effectively.

Semantic Textual Similarity (STS) Decrease in Performance: The STS task experiences a decrease in performance, with the correlation dropping from 0.546 to 0.437. The drop in STS performance suggests that the shared representations, while beneficial for paraphrase detection, may not align well with the specific requirements of the STS task. This could be due to the STS task's reliance on fine-grained distinctions in similarity, which might be diluted when the model learns to accommodate the needs of the other tasks within the shared layer. The shared layer's representations, optimized also for sentiment classification and paraphrase detection, might not capture the nuanced similarities and differences as effectively for STS.

We were expecting a increase for both STS and Paraphrase due to their commonalities but our results differed from this expectation. However, we did not expect a huge improvement in Sentiment Classification due to less overlap with the other two tasks. It seems that targeted upgrades at specific tasks i.e. Cosine Similarity for Sentiment, and finetuning for all three were generally more adept at increasing the accurarcy score rather than trying to generalize upgrades to each as we did in the shared layer. This intuitively makes sense, but it was promising to see an improvement in 2/3 tasks with the shared albeit a incredibly small one for Sentiment.

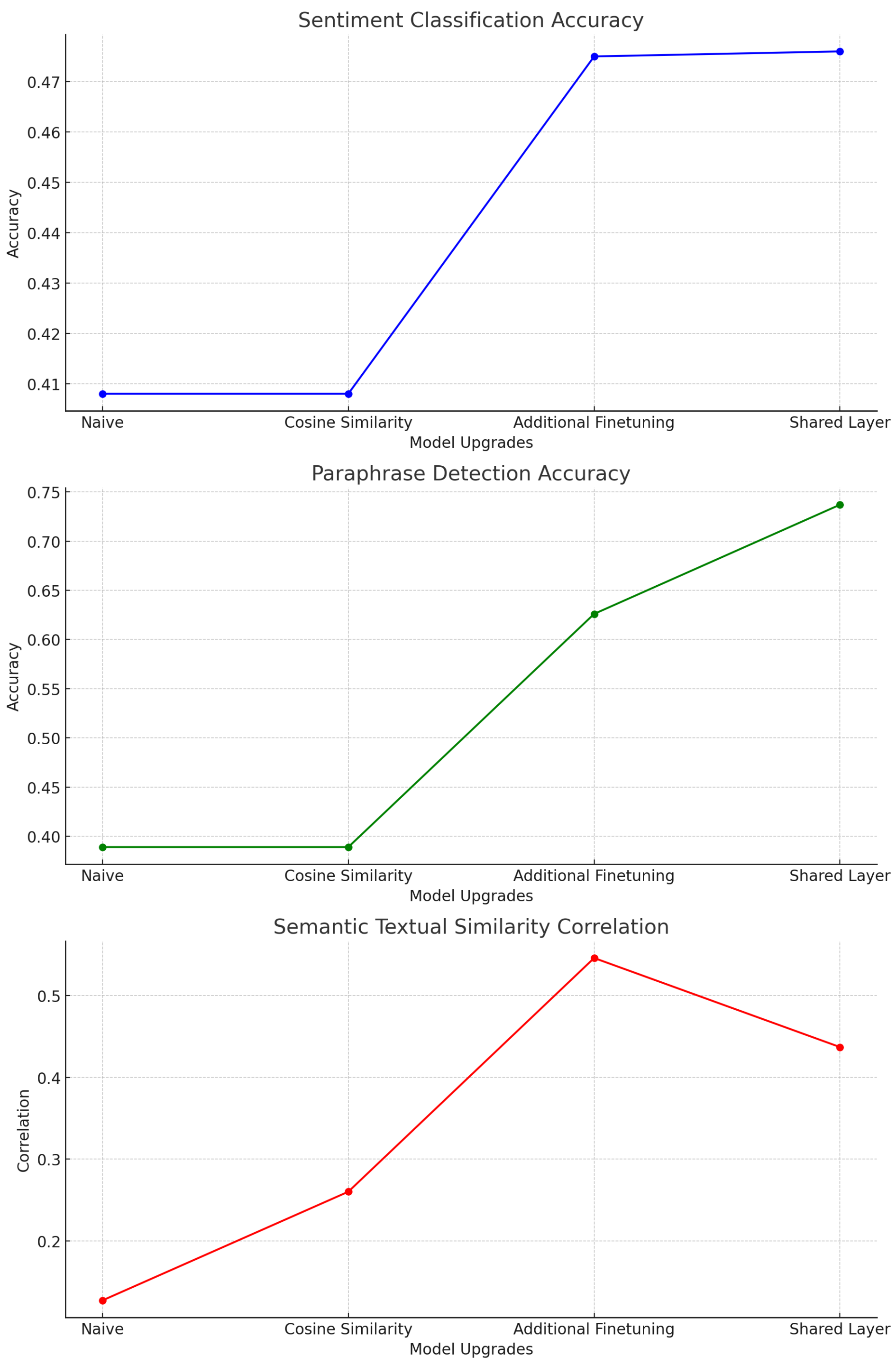


Figure 1: Multitask Comparison

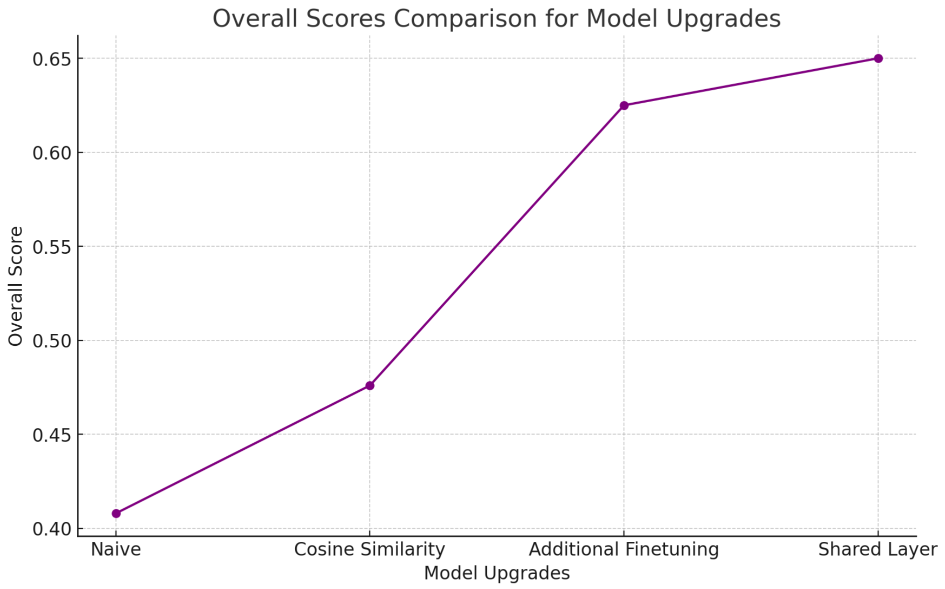


Figure 2: Overall Scores Comparison

5 Analysis

We observed both many successes and areas for improvement during our exploration of the system and its results. Notably, the model excels in capturing nuanced semantic relationships and accurately identifying paraphrased expressions, demonstrating its proficiency in understanding some textual nuances. This success can be attributed to the incorporation of additional fine-tuning along with the shared learning layer with STS. However, despite these enhancements, the model faces challenges in interpreting subtle nuances in language, particularly in sentences containing sarcasm, irony, or ambiguous sentiments. This is seen in sentences that used deep figurative language such as "Maneuvers skillfully through the plot 's hot brine – until it 's undone by the sogginess of its contemporary characters , and actors". Additionally, its performance is hindered by cultural references or idiomatic expressions not well-represented in the training data. This is seen some incorrectly predicted sentences that included phrases such as "the speedy wham-bam", "mothball-y", "slash-n-hack" and "girl-power". Furthermore, the model struggles with complex sentence structures and non/adjacent words with opposite connotations, which may obscure the intended sentiment or semantic relationship between text pairs. An example of this is the sentence "Corny, schmaltzy, and predictable, but still manages to be kind of heartwarming, nonetheless." Overall there is still ample room for growth and potential future improvements to address the identified challenges and further enhance the model's capabilities.

6 Conclusion

6.1 Main Findings

Task Synergy: The results underscore the potential for task synergy in multitask learning frameworks. Paraphrase detection and STS, both involving semantic analysis of sentence pairs, were expected to benefit mutually from shared representations. However, the shared layer's impact was more pronounced and positive for paraphrase detection than for STS.

Task-Specific Requirements The varied impact of the shared layer on different tasks highlights the importance of considering task-specific requirements. While shared representations can enhance learning by leveraging commonalities across tasks, they may also obscure or dilute task-specific features critical for specific task performance.

6.2 Primary Limitations

One-Size-Fits-All Approach: The use of a single shared layer across all tasks may not optimally address the unique requirements of each task, suggesting a limitation in the flexibility of the current model architecture.

6.3 Avenues for Future Work

Given the complexity of multitask learning, more extensive hyperparameter tuning and experimentation with different optimizer settings (including revisiting the choice of AdaGrad vs. AdamW) could yield improvements in task performances. Furthermore, we could exploring alternative shared structures. Investigating alternative architectures for shared learning could offer new ways to enhance task synergy while preserving task-specific learning. Also, implementing pretraining to each task could significantly improve model performance similar to how finetuning led to increased accuracies for all three.

References

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.