Improving BERT for Downstream Tasks

Stanford CS224N Default Project

Kevin Phan
Department of Computer Science and Biology
Stanford University
kevphan@stanford.edu

Abstract

In the realm of natural language processing, multitask learning stands as a prominent approach to tackle various language understanding tasks concurrently. Our study delves into the application of the Bidirectional Encoder Representations from Transformers (BERT) model across three significant tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Leveraging BERT's contextual word representations, we address the challenges of multitask learning while highlighting the implications of employing different methodologies for enhancing model performance. Through experiments and evaluations, we explore strategies such as multiple negatives ranking loss, cosine-similarity fine-tuning, and cumulative loss round-robin training. Our findings shed light on the efficacy of these approaches, providing insights into their impact on model performance across diverse tasks.

1 Key Information to include

- Mentor: Yuan Gao
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

BERT, short for Bidirectional Encoder Representations from Transformers, constitutes a transformer-based model designed to create contextual word representations. It leverages surrounding text context to decipher ambiguous language in texts (Devlin et al., 2018). By harnessing bidirectional word representations, BERT has emerged as a fundamental component for large language models, particularly in various natural language processing tasks. Specifically, we examine sentiment analysis, paraphrase detection, and semantic textual similarity.

Sentiment analysis involves identifying the polarity within language and categorizing it along a discrete spectrum from negative (labeled 0) to positive (labeled 5). Paraphrase detection entails identifying text that rephrases another piece of text. This task is binary classification, where two sentences are labeled jointly as 0 (not a paraphrase) or 1 (paraphrase). Semantic textual similarity (STS) aims to gauge the likeness in meaning between two sentences, represented on a continuous scale from 0 (not similar) to 5 (identical).

Addressing these three tasks within a single model exemplifies multitask learning, which can be notably challenging due to differences in objectives and formats across tasks. Prior approaches to multitask models have recognized the complexities inherent in using a single model for multiple tasks, as each task may necessitate different optimal parameters. Consequently, achieving a balance between the requirements and preferences of each task can be challenging. As noted by Yu et al. (2020), it is not uncommon for a multitask model to exhibit lower overall performance compared to a model tailored to a specific task. Nonetheless, since multitask models employ the same structure for different problems, they offer promising avenues for efficiently tackling multitask challenges. Therefore, there is a significant incentive to comprehend the limitations and enhance the performance of multitask models.

In this paper, we aim to improve minBERT's performance on multitask classification utilizing various extensions.

3 Related Work

Multitask Models, which undertake various tasks within a unified framework, face numerous optimization challenges due to the distinct optimal parameters required for each task. Consequently, they often exhibit inferior performance compared to models tailored for specific tasks (Yu et al., 2020). Currently, multiple methodologies aim to strike a balance between implementing efficient multitask models and enhancing their performance. To address this challenge, we extended the framework of our baseline BERT through fine-tuning to enhance its performance (Stickland and Murray, 2019). Multitask Fine-tuning is a technique that involves fine-tuning a model on multiple tasks simultaneously, leveraging the shared structure to facilitate more effective learning.

Round Robin Training for Combined Loss: One notable approach involves training different tasks concurrently and then combining their losses into a final loss for gradient descent, which has demonstrated superior performance compared to the baseline BERT model itself (Bi et al., 2022).

Cosine Similarity as an Additional Feature: Reimers and Gurevych (2019) conducted a comparative analysis of sentence embeddings using cosine similarity. Our study proposes calculating the cosine similarity between sentence pair and utilizing the output as an additional featre in the prediction head of the STS and paraphrase detection task.

Multiple Negatives Ranking Loss: Henderson et al. (2017) utilized paired sentence embeddings to compute the multiple negatives ranking (MNR) loss, thereby enhancing the performance of the BERT model in a task involving smart email replies.

4 Approach

4.1 Baseline BERT and Adam Optimizer Implementation

For BERT, we developed the architecture for multiheaded attention, an add-norm function, an embedding function, and a forward function.

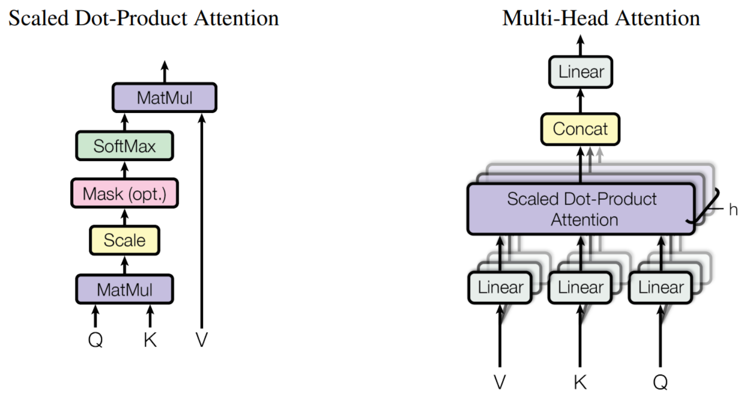

Figure 1: Implementation of scaled dot product attentions and multi-head attention (Vaswani et al., 2017)

Similarly, for the Adam optimizer, we implemented the step function in accordance with the Adam algorithm outlined in the project guidelines.

Subsequently, We proceeded to implement the forward functions of the BERT sentiment classifier and the multitask BERT classifier, as well as the prediction functions that yield predictions for each of the three targeted tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. As a result, the architecture of the multitask classifier in which our experiments built on is as follows:

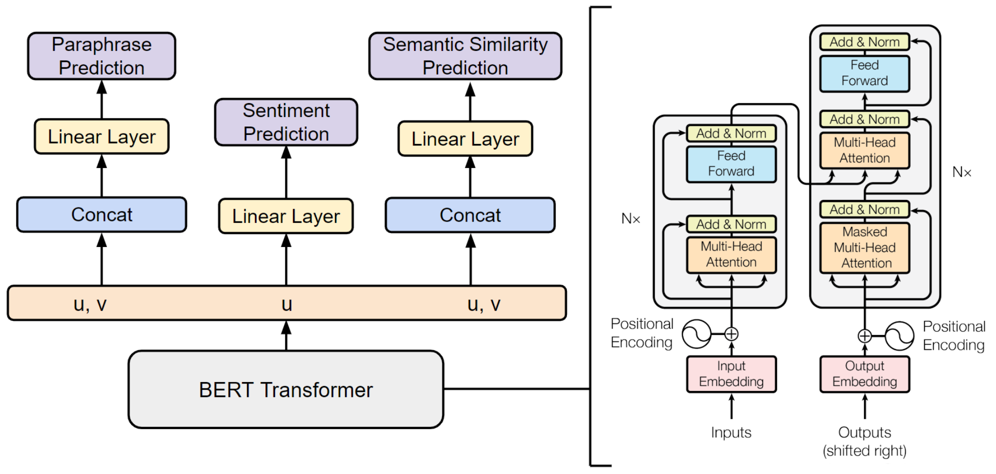

Figure 2: Multitask Classifier

4.2 Combined Loss and Round Robin Batches

For this extension, we implemented a method to cycle through each dataset for every batch, as opposed to training on one dataset/task at a time for every epoch. This allowed us to calculate the loss as follows:

$$J_{Combined} = J_{Sentiment} + J_{Paraphrase} + J_{Semantic}$$

for every batch. For every batch, we used this combined loss when calculating the gradient and updating the model weights accordingly. By cycling through each task and their corresponding datasets for each batch, we hope to prevent the model from becoming too biased towards a specific task. For the loss functions the STS task we used MSE, the sentiment classification task used cross entropy, and the paraphrase detection task used binary cross entropy with logits.

4.3 Multiple Negatives Ranking Loss

Given that the paraphrase detection class comprises sentence pairs labeled with binary values, we wanted to enhance the model by implementing multiple negatives ranking loss learning for this task.

$$J_{Paraphrase}(x, y, \theta) = -\frac{1}{K}\sum_{i=1}^{K} \log P_{\approx}(y_i|x_i)$$

This loss function endeavors to minimize the distance between positive pairs (i.e., paraphrases) while simultaneously maximizing the distance between negative pairs (i.e., non-paraphrases).

4.4 Cosine-Similarity Fine-Tuning

Cosine-Similarity Fine-Tuning is a method utilized to gauge the resemblance between two word embeddings, denoted as u and v, through cosine similarity computation:

$$similarity = \frac{u \cdot v}{\max(||u||^2 \cdot ||v||^2, \epsilon)},$$

where ϵ represents a threshold value. For every sentence pair, we calculated the similarity score according to the above formula then concatenated the value the word embeddings before inputting into the prediction linear layers. By doing so, we wanted the model to learn to utilize the cosine similarity scores when making predictions as well as increasing the expressive power of the embeddings. We did this for both the STS and paraphrase detection tasks. The modified architecture is as follows:

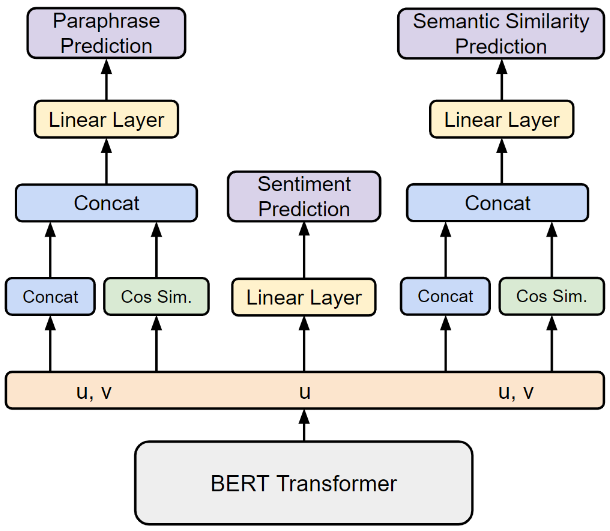

Figure 3: Modified architecture to use cosine similarity to enhance embeddings.

5 Experiments

5.1 Data

For sentiment analysis, we utilize the Stanford Sentiment Treebank (SST) dataset comprising 11,855 individual sentences, each derived from movie reviews (Socher et al., 2013). For paraphrase detection, we employ the Quora dataset, a subset of 400,000 question pairs annotated to indicate whether they are paraphrases of each other. Lastly, for the task of determining semantic textual similarity, we rely on the SemEval STS Benchmark Dataset, which consists of 8,628 pairs of sentences, each pair assigned a label indicating their similarity on a scale from 0 (maximally dissimilar) to 5 (maximally similar).

Table 1: Dataset tasks and splits

	SST	SemEval	Quora
Task	Sentiment Analysis	Semantic Textual Similarity	Paraphrase Detection
Total size	11,855	8,628	202,152
Train size	8,544	6,041	141,506
Dev size	1,101	864	20,215
Test size	2,210	1,726	40,431

5.2 Evaluation method

We employ accuracy as our evaluation metric for sentiment analysis and paraphrase detection, given that these tasks entail classification. In contrast, for semantic textual similarity, we utilize Pearson Correlation to gauge the efficacy of the predicted similarity against the actual similarity. We will compare our scores to our baseline models, as well as the GradeScope leaderboard.

5.3 Experimental details

For each experiment, we rand the following hyperparameters:

- Learning Rate = 1e-5 for finetuning, 1e-3 for pretraining
- Batchsize = 8
- Epochs = 0

- Dropout = 0.3

Overall, we conducted six experiments as follows on GCP with a P100 GPU:

1. Baseline Pretraining ~5 hrs training
2. Baseline Finetuning ~8 hrs training
3. Multiple Negatives Ranking Loss for Paraphrase Task ~8 hrs training
4. Cos Sim Embeddings ~19 hrs training
5. Cumulative Loss Round Robin ~6 hrs training
6. Cumulative Loss Round Robin + Cos Sim Embeddings ~18 hrs training

5.4 Results

Table 2: Model Performances

Model	Sentiment Accuracy	Paraphrase Accuracy	SemEval Correlation
Baseline Multitask Pretraining	0.488	0.643	0.342
Baseline Multitask Finetuning	0.503	0.681	0.354
Multiple Negatives Ranking Loss	**0.515**	0.432	0.362
Cosine Similarity Embeddings	0.506	0.731	0.351
Cumulative Loss RR	0.505	0.702	0.365
Cumulative Loss RR + Cos Sim	0.506	**0.785**	**0.376**

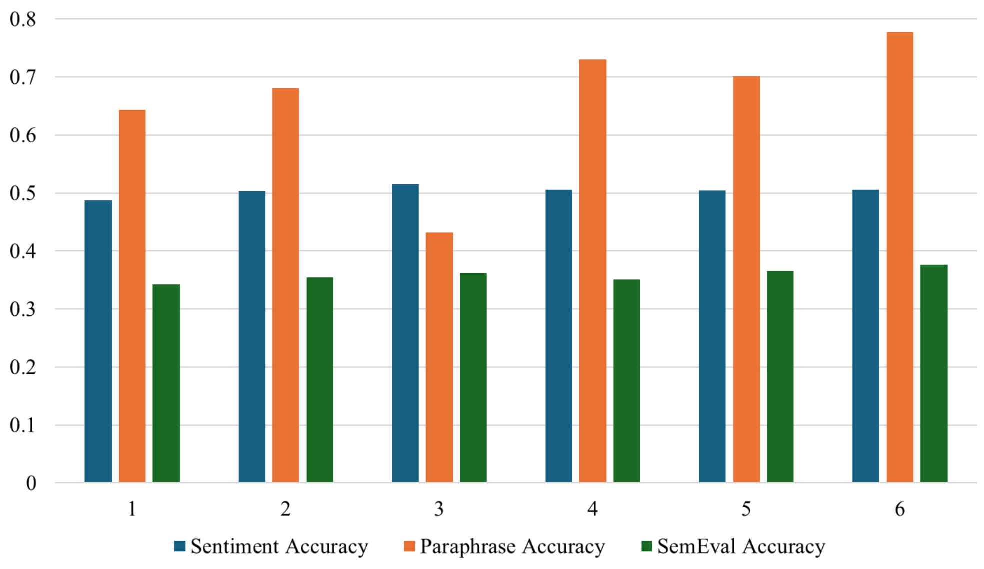

Figure 4: Experiment performance on Dev Set, for numbering refer to 5.3

Based on the results on the dev set, we decided to submit experiment six to the leaderboard (Cumulative Loss Round Robin with Cosine Similarity Embeddings). Overall, we received a test score of 0.653 and the following metrics, placing 73rd on the leaderboard.

Table 3: Best Model Performance on Test Set

Model	Sentiment Accuracy	Paraphrase Accuracy	SemEval Correlation
Cumulative Loss RR + Cos Sim	0.504	0.790	0.332

6 Analysis

6.1 Multiple Negative Ranking Loss

A notable observation was the unexpected ineffectiveness of Multiple Negatives Ranking Loss (MNRL) in enhancing the paraphrase task results. One potential explanation for this outcome could be attributed to the necessity of upscaling the calculated MNRL loss values due to their relatively smaller magnitudes compared to losses from other tasks. Consequently, the model may have allocated diminished emphasis on optimizing the paraphrase detection component. For future investigations, we intend to explore scaling the MNRL loss to ascertain its potential impact on enhancing metrics for the paraphrase task. Intriguingly, sentiment classification accuracy demonstrated superior performance in this experiment, likely owing to the sentiment classification task's proportionally greater contribution to the overall loss.

6.2 Multitask Round Robin for Cumulative Loss

Our hypothesis suggests that the round-robin cumulative loss contributed to the enhancement of results by mitigating bias during training towards specific tasks. In general, we observed a notable improvement in evaluation metrics compared to the baseline when employing this cumulative loss function. Moreover, alongside the improved metrics, we noted faster runtimes. This expedited processing can be attributed to the computation of the loss once for every batch across all three tasks, as opposed to three separate computations (once for each task) for every batch. Consequently, fewer calls to the step function are made during this procedure, thereby reducing computational overhead.

6.3 Cosine Similarity Embeddings

Furthermore, we posit that integrating cosine similarity into the prediction layers contributed to the enhancement of metrics. Overall, employing this method resulted in a notable improvement in metrics compared to the baseline multitask fine-tuning performance. However, it is worth mentioning that the runtime increased, likely due to the heightened complexity of the model. For each batch, computing cosine similarity twice becomes necessary to input into both the paraphrase and semantic evaluation tasks. Additionally, the embedding size is increased as the cosine similarity is concatenated to the embedding. Remarkably, this enhancement predominantly affected the paraphrase metrics rather than the semantic similarity task. We speculate that introducing additional layers or complexity may be necessary to further enhance the latter metric.

7 Conclusion

In this study, we investigated the multitask learning capabilities of the BERT model across sentiment analysis, paraphrase detection, and semantic textual similarity tasks. By implementing various methodologies such as multiple negatives ranking loss, cosine-similarity fine-tuning, and cumulative loss round-robin training, we aimed to enhance model performance and explore the dynamics of multitask learning. Our experiments revealed notable improvements in model performance, particularly in paraphrase detection but showed limited improvement in the sentiment classification and semantic evaluation tasks. The incorporation of cosine similarity embeddings and cumulative loss round-robin training showcased promising results, indicating the potential of these techniques in advancing multitask learning frameworks. Moving forward, further research into scaling loss functions and refining training methodologies could offer deeper insights and contribute to the continual evolution of multitask learning paradigms in natural language processing.

References

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Matthew L. Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *CoRR*, abs/2001.06782.