Losing to Win: Evaluating the Success of Various Loss Functions Within Multitask Settings

Stanford CS224N Default Project

Jadon Geathers
Department of Computer Science
Stanford University
geathers@stanford.edu

Abstract

BERT, a pretrained transformer model, has shown promising results in multitask learning, where it can be fine-tuned on multiple NLP tasks in parallel. Importantly, multitask learning augments the efficacy, efficiency, and generalizability of the model's contextualized embeddings, which motivates the further exploration of multitask fine-tuning. In this project, I implement and extend core components of the BERT model and its companion AdamW optimizer. These fine-tuning extensions are accomplished by investigating various loss functions, including a cumulative multitask loss, cosine similarity loss, multiple negatives ranking loss, and mean squared error loss. With said extensions, I further fine-tune the model's contextualized embeddings to perform well on multiple sentence-level tasks, being sentiment analysis, paraphrase detection, and the analysis of semantic textual similarity. We find that the joint use of the cumulative multitask and mean squared error loss functions yield the best results on the three tasks, with a sentiment analysis accuracy of 0.514, and paraphrase detection accuracy of 0.789, and a semantic textual similarity coefficient of 0.589. These results illustrate the effectiveness of multitask fine-tuning, which can extend the range of tasks over which BERT can successfully operate.

1 Key information

This project was mentored by Sauren Khosla. There are no external collaborators for this project, and the project is not shared with another class.

2 Introduction

The Bidirectional Encoder Representations from Transformers (BERT) model is a pretrained, transformer-based natural language processing model that has demonstrated a high degree of robustness in its generated contextual word embeddings. Given the model's ability to capture subtle nuance in language across several individual tasks, a natural direction of research is to investigate the plausibility of applying BERT within multitask settings. That is, we ponder the ability of BERT to simultaneously perform well across multiple sentence-level tasks. In this project, we specifically look towards the tasks of sentiment analysis of single sentences, paraphrase detection of pairs of sentences, and the analysis of the semantic textual similarity of pairs of sentences. Upon using BERT across these three tasks in parallel, we ideally generate more robust and generalizable sentence embeddings that perform well in a wide array of settings.

Because tasks may vary drastically in their analysis of sentence embeddings, applying a generalized, one-size-fits-all model is highly challenging. Yet, current methods for addressing this problem of achieving generalizability are wide in range. For instance, Jiang et al. (2020) use an adversarial regularization technique to keep the complexity of the model low; Sun et al. (2019) use additional pretraining to enhance the performance of weaker tasks; Yu et al. (2020) use gradient surgery to project the task gradients onto the normal planes of conflicting task gradients. Typically, these approaches tend to work well over one subset of tasks yet may exhibit subpar results over another subset–in all cases, we see that generalization is difficult simply because there are many varied tasks to consider. Thus, to contribute to the conversation, I decided to try my hand at improving the robustness of the BERT model within multitask settings by using a combination of various loss function based techniques.

My approach to the problem first required me to implement the BERT model and its corresponding AdamW optimizer to perform the sentence classification tasks of interest. I then extended the default multitask training functionality by introducing various loss functions: a cumulative multitask loss function, cosine similarity-based loss, multiple negatives ranking loss, and mean squared error loss. I use these specialized loss functions in hopes of improving the sentence embeddings for use across the three tasks at hand. In the results, we see that some loss functions and combinations of loss functions perform very well, like the mean squared error function and the cumulative multitask loss function; others conflict with the overall global performance, like multiple negatives ranking loss.

3 Related Work

My choice of loss functions for the three tasks comes from existing work. For instance, Reimers and Gurevych (2019) compare sentence embeddings using cosine similarity, so I similarly computed the cosine similarity for all sentence embeddings corresponding to the semantic textual similarity task and computed the cosine embedding loss. In a similar light, Henderson et al. (2017) use paired sentence embeddings to compute the multiple negatives ranking (MNR) loss, which enhances the BERT model's performance on a smart email reply task. Recognizing the parallel structure of the data between this paper and the Quora dataset that is used in this project, I used my paired sentence embeddings to compute the MNR loss and improve performance on the paraphrase detection task. Lastly, my idea for accumulating the loss functions over each of the three tasks per batch stems from Bi et al. (2022), where the total loss function is computed as the sum of loss functions specific to individual tasks. Whereas the authors employ this loss over the tasks of named entity recognition and category classification, I use it over sentiment analysis, paraphrase detection, and semantic textual similarity. Ultimately, I seek to gain valuable insight into the interactions between loss functions by combining the approaches in the outlined papers, hoping to understand more about how their combinations beneficially or detrimentally impacts performance.

4 Approach

4.1 BERT model and AdamW optimizer implementation

I first completed my implementations of the BERT model and the AdamW optimizer by completing the provided code skeleton. For the BERT model, this involved creating the architecture for multi-headed attention, an add-norm function, an embedding function, and a forward function; for the AdamW optimizer, I implemented the step function according to the Adam algorithm delinated in the project handout. Lastly, I implemented the forward functions of the BERT sentiment classifier and the multitask BERT classifier, alongside the prediction functions that output predictions for each of the three tasks of interest, being sentiment analysis, paraphrase detection, and semantic textual similarity.

With the implementation complete, I was then able to approach the sentence classification tasks at hand. As the central baseline, I use the predefined multitask classifier to generate results for the three tasks together. All testing was done via the existing evaluation script provided with the project.

4.2 Cumulative multitask loss

As opposed to fine-tuning my model on individual tasks, I decided to update the BERT model by using multitask learning; in doing so, I aimed to prevent the model from being biased towards any particular task, making it easier for the model to perform successfully over other tasks. Specifically, I used a round-robin approach to cycle through each dataset and accumulate the loss accordingly, subsequently using the cumulative loss to update the model. This can be illustrated mathematically

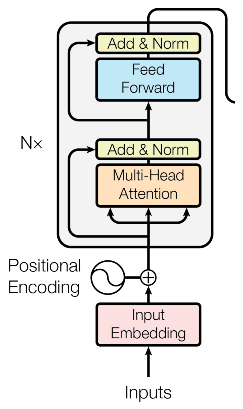

Figure 1: Encoding layer of the BERT model from Vaswani et al. (2017)

via the equation

$$J_{\text{total}} = J_{\text{sentiment}} + J_{\text{paraphrase}} + J_{\text{similarity}}$$

where each subscripted J denotes the appropriate loss function.

4.3 Cosine similarity loss

To enhance the performance of the model on the task of analyzing semantic textual similarity, I used a loss function dependent on the cosine similarity between two sentence embeddings. This cosine similarity loss is mathematically identical to the existing `CosineEmbeddingLoss` function provided by PyTorch, yet my loss function was hand-implemented and modified so as to more easily adapt to the structure of my model.

Now, take two sentence embeddings x and y, and denote their similarity label by $L(x, y)$. Note that, for the similarity task, the labels are on an integer scale from 0 to 5. For these two sentence embeddings, the loss is computed as

$$J_{\text{similarity}}(x, y) = \begin{cases} \max(0,\ \text{similarity}(x, y)) & 0 \leq L(x, y) \leq 2 \\ 1 - \text{similarity}(x, y) & 3 \leq L(x, y) \leq 5 \end{cases}$$

where similarity denotes the numerically stable cosine similarity between the embeddings, given by

$$\text{similarity}(x, y) = \frac{x \cdot y}{\max(||x||_2||y||_2, 10^{-8})}.$$

4.4 Multiple negatives ranking loss

As the paraphrase detection class consists of sentence pairs with binary labels, I saw the opportunity to apply multiple negatives ranking loss learning to this task to further improve my model. In this loss function, we aim to minimize the distance between positive pairs (in this case, paraphrases) while maximizing the distance between negative pairs (in this cases, non-paraphrases).

To implement this loss function, I used the extracted batches of the Quora dataset (used for the paraphrase task–see section 4.1) to obtain one positive pair and at least four negative pairs per batch. Let the number of negative pairs be $K - 1$. Then the data consists of K sentence pairs $[(a_+, b_+), (a_+, b_1), (a_+, b_2), \ldots, (a_+, b_{K-1})]$ where (a_+, b_+) is the positive pair and (a_+, b_i) is the ith negative pair. I calculate the loss as

$$J_{\text{paraphrase}} = \frac{1}{K}\left[\text{ReLU}(1 - a_+^\top b_+) + \text{ReLU}(1 - a_+^\top b_+ + \sum_{i=1}^{K-1} a_i^\top b_i)\right].$$

Here, we see that the loss is lower when the positive term $a_+^\top b_+$ is larger, and the loss is higher when the negative terms $a_i^\top b_i$ are larger.

4.5 Mean squared error loss with cosine similarity

I sought to improve BERT's performance on the semantic textual similarity task in a different fashion than by just using the cosine similarity loss by using the mean squared error loss, which is provided as the `MSELoss` function in PyTorch. I used this function unmodified. The paired sentence embeddings, however, are still first passed through the cosine similarity function before being sent as input into the loss function. For sentence embeddings (x_i, y_i) and their similarity label $L(x_i, y_i)$ for $i \in \{1, 2, \dots, K\}$, we calculate the mean squared error loss as

$$J_{\text{similarity}} = \frac{1}{K} \sum_{i=1}^{K} (L(x_i, y_i) - \text{similarity}(x_i, y_i))^2.$$

5 Experiments

5.1 Data

For the task of sentiment analysis with the BERT model, I am using the Stanford Sentiment Treebank (SST) dataset, which consists of 11,855 single sentences, each of which is a sentence from a movie review. Additionally, I am using the CFIMDB dataset, which consists of 2,434 polarized, possibly multiple-sentence movie reviews where each review either has a positive label or a negative label. For the additional downstream task of paraphrase detection, I am using the Quora dataset, which consists of a subset of 400,000 question pairs whose labels indicate whether the elements of the pair are each other's paraphrase. Lastly, for the task of determining semantic textual similarity, I am using the SemEval STS Benchmark Dataset, which consists of 8,628 sentence pairs, where each pair has an associated label indicated the two sentences' similarity on a scale from 0 (maximally dissimilar) to 5 (maximally similar). All datasets can be found via the Default Project Handout. The specific dataset breakdowns are as follows:

	SST	CFIMDB	Quora	STS
`train`	8,544	1,701	141,506	6,041
`dev`	1,101	245	20,215	864
`test`	2,210	488	40,431	1,726

5.2 Evaluation method

Following my implementation of the baseline BERT (+ AdamW) model, I sought to evaluate the performance of my model on the task of sentiment analysis by comparing my computed accuracies on the SST and CFIMDB `dev` datasets to the already provided scores on the same datasets given in section 4.4 in the Default Project Handout. Then, upon obtaining results for this single task, I began the core work of evaluating the model's performance on multitask classification with the tasks of sentiment analysis, paraphrase detection, and semantic textual similarity.

For sentiment analysis and paraphrase detection, I evaluate the performance by computing the accuracy, and for semantic textual similarity, I do so by computing the Pearson correlation coefficient. I compute one set of results by using the provided baseline implementation of the multitask training method, which trains only using the SST and CFIMDB datasets. I then compute the accuracies and correlation coefficients using the extended models, which train on all datasets and are extended by the various loss functions outlined in Section 3.

5.3 Experimental details

In order to verify the proper functionality of my implemented baseline BERT model on the single task of sentiment analysis, I ran my sentiment analysis classification experiment for pretraining and finetuning using the default hyperparameters suggested by the project handout. That is, for pretraining and finetuning, I set the following hyperparameters:

- Number of epochs: 10
- Batch size: 8

- Hidden dropout probability: 0.3
- Pretrain learning rate: 10^{-3}
- Fine-tune learning rate: 10^{-5}

These hyperparameters allowed me to compare my results to the expected baseline accuracies and correlation coefficient. Once I determined that the model was functional, I then approached experiments in the multitask domain. For multitask classification on sentiment analysis, paraphrase detection, and semantic textual similarity, I am again using the same default hyperparameters. I ran an ablation study on the model, taking various combinations of loss functions to analyze the impact of each component on the overall performance. For the multiple negatives ranking loss function, I required that each positive pair be associated with at least four negative pairs. Lastly, all computation was performed using a g5.2xlarge instance via Amazon's AWS computing services, and training took approximately 6-10 hours per model.

5.4 Results

I report the accuracy and correlation scores for both the `dev` and `test` datasets; scores for the `test` dataset are solely provided for the best performing model.

Multitask Classifications

BERT Model	Sentiment Task Dev Acc.	Paraphrase Task Dev Acc.	Similarity Task Dev Corr.
Default	0.406	0.380	-0.010
Multitask	0.425	0.785	0.335
Multitask + Cos	0.479	**0.787**	0.240
Multitask + MNR	0.493	0.377	0.531
Multitask + MSE	**0.504**	0.785	**0.624**
Multitask + MSE + MNR	0.460	0.510	0.617
	Sentiment Task Test Acc.	Paraphrase Task Test Acc.	Similarity Task Test Corr.
Multitask + MSE	0.514	0.789	0.589

Out of all models, we observe the best overall performance with the model that uses both the cumulative multitask loss function and the mean squared error (MSE) loss function for the similarity task. Interestingly, the MSE loss function led to a nearly twofold increase in the correlation score from the next best model (0.335 versus 0.624). Despite its calculation relying on the cosine similarity as does the cosine similarity loss function, MSE loss drastically outperforms the cosine similarity loss (0.624 versus 0.240) in the similarity task. In fact, the ensemble cosine similarity and cumulative multitask loss model performs worse on the similarity task than the multitask loss model alone. The MSE loss function did better than I expected, but the poor performance of the cosine similarity loss function was not at all expected. Despite two different programmatic approaches to using the cosine similarity loss, the observed performance was still subpar.

I now state the best results for the individual tasks on the `dev` datasets, which are bolded in the table. The model that performs best on the sentiment task is the cumulative multitask and MSE loss function model (acc: 0.504). For the paraphrase task, the best performing model is the cumulative multitask and cosine loss function model (acc: 0.787). Lastly, for the similarity task, the best model is again the cumulative multitask and MSE loss function model (corr: 0.624).

The cumulative multitask loss function led to an increase in performance across the three tasks and an especially large performance with the paraphrase and similarity tasks when compared with the default multitask training function. This demonstrates the value of the using the cumulative loss via a round-robin approach to generalize the model and lead to a global improvements across a wide range of tasks. On the other hand, the multiple negatives ranking loss performed poorly (acc: 0.377), and I propose that this is likely to be implementation-based, as the architecture of the implemented loss function may conflict with existing code.

6 Analysis

I refer to the ablation study in Section 5.4 to perform a qualitative analysis. The models were successively shrinked from the largest ensemble model, consisting of three simultaneously active loss functions, to the smallest model, consisting of one loss function. From the study, we notice that te best performance in each task arises when the cumulative loss function is applied in conjuction with another task-specific loss function (cosine/MSE). Yet, not all combinations of loss functions work well.

One potential reason for this is that the loss functions are not applied on the same scale. For instance, the cosine similarity loss is always on a scale of 0 to 1, whereas MSE loss is theoretically unbounded. Consequently, this could lead to the model favoring one loss function over another, which leads to poor performance overall. Another reason that may cause the unexpected performance of the ensembled loss functions is that the loss functions may be operating over conflicting objectives, which makes optimizing the model difficult. Perhaps the aforementioned technique of gradient surgery may provide some resolve to this issue.

I notice that, throughout the ablation study, the models' performance on the sentiment task varies less drastically than in the other tasks when the multitask loss is introduced and when the model is trained on all datasets. I found this interesting, as I would expect that the cumulative multitask loss function would lead to more robust sentence embeddings that generalize well to all three tasks and lead to a very large boost in performance in every task, but this is not the case for only the sentiment task. However, this unexpected result could be an issue not intrinsic to the model but rather an issue intrinsic to the data. Sentiment analysis is a largely subjective task, and when constructing datasets that rely heavily on subjective interpretation, there is a greater possibility of mislabels or conflicting examples in the dataset.

7 Conclusion

Ultimately, multitask finetuning via the cumulative loss function and the round-robin approach has proven highly effective at producing a generalized model. However, loss functions applied to individual tasks led to varying degrees of success with respect to their overall performance in the model. Whereas cosine similarity loss performed in a subpar fashion for the semantic textual similarity task, the mean squared error loss performed very well. Then, multiple negatives ranking loss did not perform as well as expected in the paraphrase detection task, although it was expected to lead to large improvements. These results provide insight into the complex and often unexpected interactions between various loss functions. With more time and possibly more people working on this problem besides myself, my implementations of the loss functions could be refined, and the objectives could be redesigned in a non-conflicting manner, which would potentially allow the loss functions to behave more compatibly.

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.

Matthew 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. *arXiv preprint arXiv:1705.00652*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online. Association for Computational Linguistics.

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

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583.

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

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