minBERT and Downstream Tasks

Stanford CS224N Default Project

Shouzhong Shi
Department of Computer Science
Stanford University
sshi@stanford.edu

Abstract

This project aims to delve into the intricacies of BERT's architecture, assess its efficacy across diverse datasets and tasks, and investigate techniques for refining its performance. Initially, the BERT architecture is implemented for sentiment classification, laying the groundwork for the project's foundational analysis. Subsequently, the model is extended to a multi-task classifier to assess its capability across sentiment classification, paraphrase detection, and semantic textual similarity tasks. Addressing challenges such as conflicting gradients and imbalanced datasets inherent in multi-task classification, various methods including weighted loss and gradient surgery are employed. Through the execution of six experiments encompassing individual extensions and combinations thereof, the best-performing model showcases notable enhancements over the baseline, achieving accuracies of 0.498 for sentiment analysis, 0.778 for paraphrase detection, and 0.465 for semantic textual similarity. Particularly noteworthy is the substantial improvement in semantic textual similarity over the baseline. Overall, the fine-tuned model demonstrates promising performance across the three tasks, with a pronounced advantage observed in textual similarity, underscoring its potential as a robust and adaptable BERT-based multi-task model, contingent upon further refinement and exploration.

1. Key Information

. mentor: Yuan Gao
. external collaborator: NA
. sharing project: NA

2. Introduction

The rise of powerful pretrained large language models (LLMs) like BERT and GPT has significantly transformed the landscape of Natural Language Processing (NLP) research. Previously, research primarily focused on building isolated models tailored to specific language tasks from scratch, with limited knowledge sharing across different models and tasks. However, the advent of large attention-based language models, extensively trained on basic tasks using vast amounts of text data, has yielded highly effective token embeddings that offer substantial benefits across various downstream language tasks. Consequently, researchers have increasingly leveraged these pretrained models as a foundation to develop cutting-edge models for diverse language tasks.

Given that the rich token embeddings generated by LLMs contain valuable information applicable to multiple tasks, the concept of multitask language models naturally arises. These models aim to utilize the same set of embeddings for multiple downstream tasks, streamlining the overall process. In this paper, a multitask classifier is employed to tackle these downstream tasks. Multitask deep learning models, a class of neural network models,

are capable of simultaneously handling multiple interconnected tasks using a single network architecture featuring multiple output layers, each dedicated to a distinct task. While shared layers of the model extract features beneficial for all tasks, task-specific layers focus on making predictions for each task. However, despite its efficacy, this approach poses challenges that need to be addressed. For instance, since multitask classifiers address related tasks concurrently, there's a risk of interference among models, potentially leading to reduced accuracy. Moreover, discrepancies in the amount of data available for different tasks may introduce biases favoring tasks with more data, thereby adversely affecting performance on other tasks. Thus, the design of the multitask classifier architecture and overall approach must carefully navigate these challenges.

In this project, I try to enhance and fine-tune a minBERT model to handle three distinct downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity analysis. This involves implementing several extensions, such as gradient surgery, layer-wise learning rate decay, employing cosine similarity fine-tuning for semantic textual similarity, incorporating additional fine-tuning datasets, and applying weighted loss to enhance the performance of the multitask classifier and address the multi-task challenges.

3. Related Work

When LLMs became available, extensive research and experimentation have been conducted to enhance their performance across various downstream language tasks. The abundant research in this field presents numerous intriguing and promising ideas for enhancing any model that employs a pretrained LLM as its foundation. For instance, Sun et al. [1] systematically explored methods to fine-tune the BERT model for diverse downstream tasks.

Another notable study by Yu et al. in 2020 [2] introduced gradient surgery, a technique aimed at resolving gradient conflicts in multi-task learning. It hypothesizes that a significant challenge in optimizing multi-task learning lies in conflicting gradients from different tasks hindering progress.. In this context, gradients are considered conflicting if they exhibit negative cosine similarity. The proposed gradient surgery seeks to adjust gradients for each task to minimize negative conflicts with other task gradients, while fostering positive interactions between them, without making assumptions about the model's structure. Thus, only conflicting gradients undergo modification, while non-conflicting ones remain unchanged.

4. Approach

4.1 minBert

The backbone of the model that I use in this default project is BERT, which convert sentence input into tokens before it performs any additional processing. The BERT model first breaks down the input text into individual tokens and assigns each token a unique ID. Then, a trainable embedding layer is applied to each token. Besides embedding layer, BERT also includes transformer layer, which comprises multi-head attention, a residual connection with an additive and normalization layer, a feed-forward layer, and another residual connection

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O$$
$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

figure 1 Multi-head attention

To establish the backbone of our baseline, I utilize the original BERT model, which incorporates multi-head self-attention and Transformer layers, and call it minBERT. I employ this minBERT model to conduct

sentiment classification on the Stanford Sentiment Treebank dataset (SST) and the movie reviews dataset (CFIMDB).

The accuracy of both pretraining and fine-tuning the minBERT model on SST and CFIMDB dataset are shown in Table 1.

Table 1. minBERT results on SST and CFIMDB datasets

	Pretrain Dev Accuracy	finetune Dev Accuracy
SST	0.390	0.522
CFIMDB	0.78	0.976

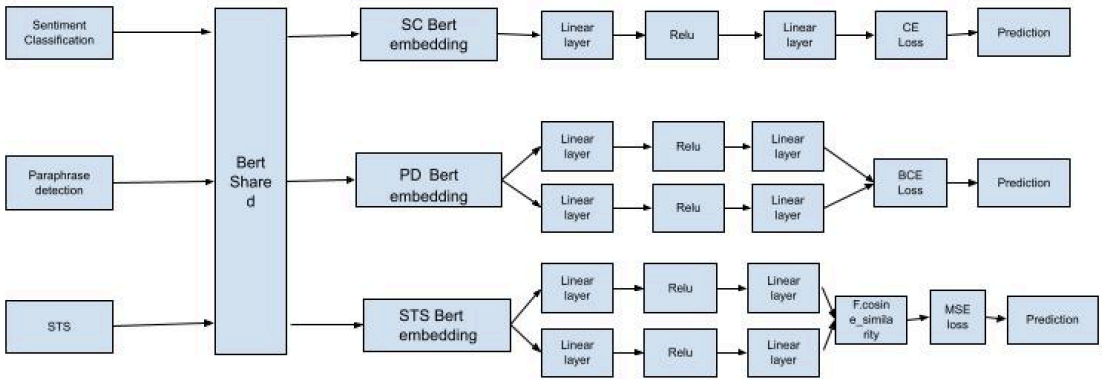
4.1 baseline

The baseline is a simple implementation of minBERT that retains key BERT architectures such as multi-head self-attention, a transformer layer, and pre-training and uses the Adam optimizer. In addition, the multitask classifier was constructed using a dropout layer, and utilized one linear layer per task. The performance evaluation on the SST, Quora, and SemEval datasets, and using the default parameters serves as our baseline to compare the effectiveness of fine-tuning and other modifications on the model.

Table 2 Multi-task accuracy

Overal Score	Sentiment Classiification Acc	Paraphrase Detection Acc	Semantic Textual Similarity Acc
0.508	0.521	0.384	0.237

4.2 Model Architecture



4.3 Extensions

4.3.1 Gradient Surgery

In Section 2, it was noted that during the fine-tuning of a pre-trained model across multiple tasks, gradients may vary greatly in magnitude, potentially hampering the learning process. In our model, due to the

disparate sizes of the datasets, particularly with the Quora dataset being significantly larger than the others, backpropagation from the Quora task can detrimentally affect the gradients of the other tasks. To mitigate this issue, I integrated PCGrad, a gradient surgery technique, into the model. Following the methodology outlined in the original PCGrad paper [3], I assess whether the gradients of the current task conflict with those of the other tasks by computing their cosine similarity. If the similarity is negative, indicating conflict, I replace the current task's gradient with its projection onto the normal plane of the conflicting gradient. This process is repeated for each of the three tasks. Specifically, GS computes the gradient magnitude of two gradients

$$\Phi(\mathbf{g}_i, \mathbf{g}_j) = \frac{2\|\mathbf{g}_i\|_2 \|\mathbf{g}_j\|_2}{\|\mathbf{g}_i\|_2^2 + \|\mathbf{g}_j\|_2^2}$$

and use it to decide whether they are conflicting, if the similarity is negative, then we should replace g to be

$$\mathbf{g}_i = \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_j\|^2} \mathbf{g}_j.$$

Otherwise g is untouched.

4.3.2 Cosine Similarity

In one of our approaches to fine-tuning, I integrate cosine similarity into the training process for the STS task. This metric evaluates the likeness between two embeddings within a high-dimensional space, ranging from -1 to 1, with values nearing 1 indicating strong similarity. Specifically, during training iterations, I refine our multi-task model using cosine embedding loss, which hinges on cosine similarity. This approach endeavors to refine the embeddings of words or phrases that share semantic meaning, aiming to reduce the gap between embeddings of similar items while widening the gap between those of dissimilar items. Ultimately, this process yields a scalar value denoting the loss, which is minimized throughout training.

4.3.3 Layer-wise learning rate decay

I incorporated a decreasing decay rate strategy across the layers, employing specific parameters of $\xi = 0.95$ and lr=1.5e-5. Essentially, given that our BERT model comprises 12 layers, the learning rate for the last layer is set to 1.5e-5, while for the first layer, it's calculated as 1.5e-5 multiplied by 0.9512, resulting in 8.1e-6. The choice of the decay factor of 0.95 stems from Sun's paper [1], which identified it as the most effective parameter. I aim to maintain an average learning rate across the layers close to the default learning rate provided, which is 1.0e-5. By implementing this extension, the model prioritizes retaining crucial information from the more relevant last layers with higher learning rates, while gradually reducing the retention of information from the initial layers to mitigate interference among different tasks

4.3.4 Additional Finetuning Datasets

While the provided training datasets are probably most useful for model training, additional related datasets could also improve performance. Since the provided training datasets for SST and STS are much smaller than the training dataset for PARA, I adapted outside datasets and added them to the provided datasets. Please see the Data section for more details.

4.3.5 Layers

For each of our three tasks - paraphrase identification, semantic textual similarity, and sentiment analysis - I incorporated a two-step process into our BERT classifier. First, I implemented a linear layer followed by a ReLU activation function. This initial linear layer elevates the input data into a higher-dimensional space, enabling the model to discern intricate relationships between inputs and outputs. The ReLU activation function introduces non-linearity into the decision-making process by converting negative values in the linear output to zero while leaving positive values unchanged. This mechanism empowers the model to capture non-linear patterns within the input data, a crucial aspect for achieving optimal performance across

the three tasks. Secondly, a second linear layer is employed to map the abstract representations derived from the ReLU activation function to the output layer. This step facilitates the model in making predictions based on the intricate features gleaned from the input data

5. Experiments

Data: Sentiment analysis is performed using two main datasets: the Stanford Sentiment Treebank (SST) and the CFIMDB dataset, comprising 11,855 and 2,434 examples respectively. For paraphrase detection, the Quora dataset (Quo) with 400,000 examples is utilized. Additionally, the SemEval STS Benchmark Dataset, consisting of 8,628 examples, is employed for the STS task. Concerns regarding potential overfitting due to limited data in SST and SemEval led to the decision to fine-tune the model using the SICK_train.txt dataset. This dataset contains 10,000 sentence pairs with varying similarity levels, ranging from 1 (unrelated) to 5 (equivalent meaning), thereby providing supplementary data for the semantic textual similarity task. To ensure compatibility with the STS data, which uses a continuous similarity score between 0 and 5, the scores from the SICK_train.txt dataset were rescaled from [1, 5] to [0, 5] using the formula: newScore = (oldScore − 1) × 1.25

Evaluation method: I employ two evaluation metrics across the three tasks. For the SST and paraphrase tasks, I utilize accuracy as the evaluation metric. Accuracy reflects the proportion of correctly classified instances within the dataset, making it a straightforward and intuitive measure of a model's classification capability. In the context of the STS task, I opt for the Pearson correlation coefficient to assess performance, as stipulated by the default project starter code. This choice is rational because it furnishes a singular value summarizing the overall linear correlation between the ground truth labels and the model's predicted similarity scores. I assess the performance of our fine-tuned multitask models using the metrics mentioned above, comparing their scores against those of our baseline model.

Experimental details:

In the experiments, I train the model for a total of 10 epochs and apply a learning rate of 1e − 5 for pre-training. Also, I set the hidden_dropout_prob to 0.3 for all experiments. I trained the above 4 extensions under the method section individually, and I combined some of them to explore a best-result model.

No	Model	SST Accuracy	PARA Accuracy	STS Accuracy	Average Accuracy
1	Base Bert	0.301	0.625	0.212	0.501
2	Bert + Cosine Similarity	0.301	0.625	0.260	0.518
3	Bert + layers	0.299	0.625	0.243	0.515
4	Bert + gradient surgery	0.332	0.641	0.264	0.535
5	Bert + rt decay	0.348	0.641	0.269	0.541
6	Bert + fine tune addition data + gradient surgery + rt decay	0.506	0.742	0.504	0.663
7	Bert + fine tune addition data +	0.498	0.778	0.465	0.670

	layers + cosine similarity + rt decay				

I found that adding additional task-specific layers and just using cosine similarity had not a lot of effect on the baseline performance as individual extension to the minBert baseline implementation. The big improvement in the performance was due to finetuning on additional data sets combined with a few extensions. Finally, the best performance was achieved with cosine similarity, learning rate decay, additional layers for paraphrase detection, and additional training on the SICK datasets It outperformed the baseline with accuracies of 0.498 for sentiment analysis, 0.778 for paraphrase detection, and 0.465 for STS.

6. Analysis

6.1 task analysis

The paraphrase detection task demonstrates the highest accuracy across all models. Particularly noteworthy is the significant improvement in STS performance from the baseline, particularly evident when fine-tuning with additional SICK datasets. The overall accuracy sees substantial enhancement. Additionally, I conducted error analysis on each task, examining some model outputs. It was observed that the model excelled in accurately predicting sentiment for concise sentences with evident positive or negative language. However, it was noted that the model heavily relies on such language and struggles to grasp contextual nuances, comprehend sarcasm, irony, idiomatic expressions, or figurative language. In the realm of paraphrase identification, I've observed numerous instances of false positives arising from sentences sharing a high number of identical words arranged similarly. This could stem from the model's limited exposure to variations of similar yet non-identical sentence structures. Regarding semantic textual similarity, my research suggests that the model excels with concise sentences featuring straightforward grammatical constructs but encounters difficulties when confronted with more intricate language patterns.

6.2 Extensions analysis

6.2.1 Inference between different tasks

When comes to the multitask learning, one of the challenges is to handle interference between different tasks. E.g the gradient/target may be in different directions. In an attempt to mitigate this challenge, I implemented gradient surgery. When two gradients are working against each other, the method is to change them by making each one point in a direction that’s perpendicular to the other. This stops the interfering parts of the gradients from affecting the network. it selectively adjusting gradients to prioritize learning tasks that are more important or challenging, thus enhancing overall prediction accuracy across multiple tasks. From the experiment 4 as we can see, it improved all the 3 tasks.

6.2.2 Additional Dataset

From the provided data we can see that Quora dataset is more than ten times larger than the other two datasets. In addition to the provided data, I added additional datasets, SICK dataset in this case, which contains about 10k rows of data in total, help improved the overall performance quite a lot. Additional datasets introduce a broader range of language patterns, styles, and domains, enriching the model's understanding of various linguistic nuances

6.2.3 Fine tuning

As demonstrated in the experiments above, fine-tuing is crucial in multi-task learning. Fine-tuning allows the

model to adapt to the specifics of each task. In MTL, a single model is trained on multiple tasks, each with its own unique characteristics, data distributions, and objectives. Fine-tuning enables the model to adjust its parameters to better address the nuances of each individual task. Fine-tuning leverages the pre-trained representations and adapts them to the particular tasks at hand. This transfer of knowledge from pre-training helps the model to generalize better

7. Conclusion

This project presented numerous captivating challenges, both in terms of conceptualization and implementation/engineering. I immersed myself in understanding the intricate mechanisms of the renowned BERT model, constructing pipelines for pre-training and fine-tuning, and customizing the model for three distinct tasks. Incorporating gradient surgery and cosine similarity into our multi-task framework resulted in superior performance compared to the baseline and other fine-tuned models. This underscores the effectiveness of gradient surgery in addressing conflicting gradients arising from disparate tasks and the appropriateness of cosine similarity for similarity-based tasks. While there were many other experiments I wished to explore, such as "Multiple Negative Ranking Loss" and more exhaustive hyperparameter tuning, constraints in compute resources and time limited my endeavors. Future iterations of this project will involve delving deeper into hyperparameter experimentation. Furthermore, there is a need for future research to investigate the effectiveness of utilizing modular and hierarchical architectures for weight sharing in multitask learning

References

[1] Ch iSun,Xipeng Qiu,Yige Xu, andXuanjing Huang. How to fine-tune bert for text classification? In China National Conference on Chinese Computational Linguistics, pages 194–206. Springer, 2019.

[2] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. Advances in Neural Information Processing Systems, 33:5824– 5836, 2020.

[3] Wei-Cheng Tseng. Weichengtseng/pytorch-pcgrad, 2020.

[4] Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. Mtrec: Multi-task learning over bert for news recommendation. In Findings of the Association for Computational Linguistics: ACL 2022, pages 2663–2669, 2022.