Investigating BERT Model's Abilities in Multi-Task Learning and Methods for Performance Improvement

Stanford CS224N {Default} Project

Haozhuo (Tommy) Li
Department of Statistics
Stanford University
tommy01@stanford.edu

Xi (Mac) Ya
Department of Computer Science
Stanford University
maaac@stanford.edu

Abstract

In recent years, multi-task learning (MTL) has emerged as a promising approach to enhance the performance of deep learning models by leveraging shared information across related tasks. However, the ability of popular large language models like BERT to handle multiple tasks simultaneously remains under-explored. In this paper, we investigate the performance of BERT (Devlin et al., 2019) on MTL and propose a series of methods to improve its effectiveness in this setting. We conduct experiments on three natural language processing tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Our proposed method demonstrates improvements over baseline BERT models and other MTL strategies, highlighting its potential for a wide range of applications. Meanwhile, we had findings contrary to our expectation, which outlines future work that should be conducted.

1 Introduction

The Bidirectional Encoder Representations from the Transformers (BERT) model (Devlin et al., 2019) has achieved state-of-the-art performance on various NLP tasks. However, despite its remarkable success, BERT's ability to handle multiple tasks simultaneously, and the methods to improve its performance on multi-task learning (MTL), remains largely unexamined. This paper aims to address this gap by investigating BERT's capabilities in MTL and proposing methods to enhance its performance.

The problem of multi-task learning is both exciting and challenging due to the potential benefits of transferring knowledge across different tasks and domains, leading to more efficient learning and improved generalization (Caruana, 1997). In natural language processing, MTL is particularly appealing as it allows for the exploitation of shared linguistic representations, which can result in more robust and accurate models (Ruder, 2017). However, this potential advantage also introduces difficulties in optimizing the shared model parameters across diverse tasks, which may have conflicting learning objectives or require distinct feature representations (Liu et al., 2019b).

The unique architecture of BERT, characterized by its deep bidirectional attention mechanism and pre-training on a massive corpus, may both facilitate and hinder its adaptation to MTL. On one hand, BERT's capacity to capture complex contextual information could promote knowledge transfer across tasks. On the other hand, the model's size and depth may exacerbate optimization challenges and contribute to catastrophic forgetting (Goodfellow et al., 2014) when training on multiple tasks simultaneously.

To address these concerns, this paper systematically investigates BERT's performance on MTL. We evaluate the model on a diverse set of NLP tasks and datasets, examining the impact of different MTL strategies and training techniques on its performance. Concretely, we adopted several methods to adapt BERT to MTL, including Gradient Surgery, SMART, further pretraining, etc, leveraging its inherent structure and exploiting task-specific information to facilitate knowledge transfer and mitigate potential conflicts among tasks.

In the end, we had findings different from our preliminary beliefs. Further pretraining, for example, brought little performance gain to our model, which is analyzed in the 5. Analysis section below. Although not all methods performed as anticipated, our multiple adaptations to the base BERT model yielded a net gain in overall performance, signifying the potential for further refinement and optimization through future research.

2 Related Work

2.1 A Closer Look at How Fine-tuning Changes BERT (Zhou and Srikumar, 2022)

As pretraining becomes prevalent in deep learning, we want to understand its effect on the model weights before applying it in our own project. This paper analyzes the effect of pretraining with probing methods. It shows that pretraining generally improves accuracies in classification tasks and simplifies the underlying embedding space, which makes each class more easily separable.
As our project contains multitask learning on three tasks, they all depend on the same embeddings generated from BERT. Our work are naturally an extention of the paper on multitask learning.

2.2 Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks (Reimers and Gurevych, 2019)

This paper is directly related to one of the three tasks: semantic texual similarity (STS). It proposes a new modelt that optimizes findind of similar sentence pairs. But we only utilize an idea in the paper that sentence similarity can be computed by the cosine similarity of their embeddings.
This idea inspires us to add cosine similarity into the prediction head of paraphrase detection and STS.

2.3 SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization(Jiang et al., 2020)

Due to the complexity of pretrained large language model, it is likely to overfit on the downstream task when finetuning. This paper proposes two regularization methods. We utilize one of them in our project to prevent our model from overfitting on any tasks: Smoothness-Inducing Adversarial Regularization. Therefore, our work is a generalization of the method on multitask learning.

2.4 Gradient Surgery for Multi-Task Learning (Yu et al.)

This paper identifies that multitask learning are challenging as gradients of multiple tasks may be conflicting and cause detrimental gradient updates. It proposes a general solution for this problem: gradient surgery. It simply projects the conflicting gradients onto the normal plane of other gradients. We utilize this approach as the fundamentals of our multitask training process.

2.5 Decoupled Weight Decay Regularization (Loshchilov and Hutter, 2017)

Kingma and Ba (2014) introduced the Adam algorithm, a novel optimization technique for training deep neural networks. Loshchilov and Hutter (2017) proposed a modification to the popular Adam optimization algorithm, called AdamW. The authors introduce a weight decay approach that is decoupled from the learning rate, resulting in improved performance.

2.6 How to Fine-Tune BERT for Text Classification? (Sun et al., 2019)

The authors propose a two-step approach for fine-tuning BERT for text classification tasks. The first step is called "re-pretraining," where they continue the masked language model (MLM) training using domain-specific data before fine-tuning BERT for the specific task. The re-pretraining step aims to adapt the BERT model to the domain-specific data, capturing the data's unique characteristics and linguistic patterns. This greatly inspired us to re-pretrain BERT to have better MTL results.

2.7 EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks (Wei and Zou, 2019)

This paper proposes Easy Data Augmentation (EDA), a set of four simple techniques for text classification tasks: synonym replacement, random insertion, random swap, and random deletion. The authors demonstrate that these techniques can improve model performance on various datasets.

3 Approach

3.1 Implementation of base BERT and Adam Optimizer

In the first part of our project, we implemented the following core parts of the original BERT (Devlin et al., 2019):

Adam Optimizer
For the optimizer, we implemented Decoupled Weight Decay Regularization and Adam: A Method for Stochastic Optimization.

Tokenizer `(tokenizer.py)`
Tokenization is a method that splits paragraphs and sentences into smaller units (tokens) before NLP models performing any additional processing. The BERT model utilizes a `WordPiece` tokenizer that splits sentences into individual words into word pieces. Word pieces that have previously not been seen will be set as the [UNK] token. To represent sentence embeddings with the hidden state of the first token, [CLS] is prepended to the token representation of each input sentence.

Embedding Layer
After converting each token to ids, the BERT model subsequently utilizes a trainable embedding layer across each token. In `bert.BertModel.embed`, we finished implementing the embedding layer by summing the token embeddings, the segmentation embeddings, and the position embeddings, which, after applying embed_layer_norm and dropout, are utilized later in the model to encode the position of different words within the input.

BERT Transformer Layer
The base BERT makes use of 12 Encoder Transformer layers. The Transformer layer of the BERT transformer consists of multi-head attention, followed by an additive and normalization layer with a residual connection, a feed-forward layer, and a final additive and normalization layer with a residual connection, which we implemented in `bert.BertLayer`

Multiheaded Self-Attention
Multi-head Self-Attention consists of a scaled-dot product applied across multiple different heads. We implemented the attention layer in `bert.BertSelfAttention.attention` according to the following function:

$$\texttt{Attention}(Q, K, V) = \texttt{softmax}(\frac{QK^\top}{\sqrt{d_k}})V$$

Baseline
For the baseline, we train the model only on the Stanford Sentiment Treebank Dataset for sentiment analysis and get predictions of all three tasks with separate prediction head on each task.

3.2 Multi-task Learning and Optimization

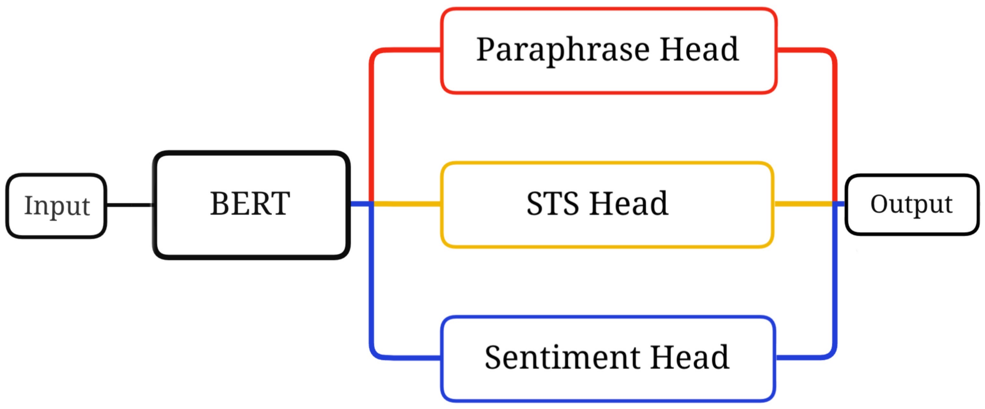
In the second part, we use the BERT model implemented previously and optimize it simultaneously on three tasks: sentiment analysis, paraphrase detection, and semantic textual similarity (STS), each with details in the next section.

Additional Pretraining
(Not used in the final model), following Sun et al. (2019), further pretraining was applied on our model. The process was done by assembling the BERT model (`bert-base-uncased`) with a masked language modeling head and training on datasets of the domain specific to our task, including SST, STS, IMDB and Quora dataset.

Data Augmentation for Pretraining Dataset
In order to improve the unsatisfying pretraining results, upon examining the four datasets, we observed a significant imbalance: the Quora dataset contains more sentences than the combined total of the other three. Consequently, we employed data augmentation techniques to enhance the size of the smaller datasets. Among various current NLP data augmentation methods, such as synonym replacement, random insertion, random swap, and random deletion, we examined synonym replacement, back translation, and contextual word embeddings augmentation. To implement those, we adopted nlpgaug, which is a library for textual augmentation in machine learning experiments. All of three showed promising augmentation results, especially back translation, which utilizes WMT19 English to German and German to English model (Ng et al., 2020). For example, a sentence in which colloquialisms or abbreviations are used can be back-translated into more formal words or grammar structures, which maintain its contextual meaning but with different word choices. Unfortunately, using the WMT19 model twice (English to German and German back to English) for each word leads to a slow inference speed, which we couldn't afford. As a result, synonym replacement is deployed twice on each sentence, in which words are replaced with their synonyms to create new, slightly different sentences. Lastly, contextual word embeddings augmentation is used, which leverages pre-trained contextual word embeddings, the GPT model in our case, to generate semantically similar sentences. This is achieved by replacing words in the original sentence with their nearest neighbors in the embedding space or by generating new words conditioned on the context. Eventually, the sizes of SST, STS, IMDB datasets were increased by three times. Note that data augmentation is only used on pretraining, not finetuning.

AdamaxW Optimizer
(Not used in the final model), following Liu et al. (2019a), we tried combining Adamax (Kingma and Ba, 2014) and AdamW (Loshchilov and Hutter, 2017) as AdamaxW to gain ultimate performance. However, as promising results were not achieved, we eventually used AdamW as our optimizer.

Multitask Learning
To learn the three tasks simultaneously, we share the BERT parameters among tasks and add separate prediction head for each task. The high-level model structure are shown in the figure below:



Prediction Head of Each Task
For the sentiment analysis head, we simply use a linear layer to transform the sentence embedding to a logit of size 5 (number of classes).
For the paraphrase detection head and STS head, the inputs are embeddings of a pair of sentences. The two heads consists of two parts. They first one compute the cosine similarity of the two embeddings and scale them to appropriate scale (original scale for paraphrase, 0-5 for STS). The second part simply concatenate these embeddings and apply linear layers to output a scalar logit.

Loss Function of Each Task
We use the categorical cross entropy loss for sentiment analysis, binary cross entropy loss for paraphrase detection, and (1 - pearson correlation) for SST.

Gradient Surgery (Yu et al.)
For the training process, we use the round-robin approach. In a batch, we first forward propagate one batch for each task to get three losses. Then, we use the gradient surgery method described in the previous section to perform backward propagation. Specifically, each conflicting gradient $\mathbf{g_i}$ is projected onto the normal plane of another task $\mathbf{g_j}$:

$$\mathbf{g_i} = \mathbf{g_i} - \frac{\mathbf{g_i} \cdot \mathbf{g_j}}{||\mathbf{g_j}||^2} \cdot \mathbf{g_j} \tag{1}$$

As the three datasets are of different sizes (details in the next section), we write wrappers for the two shorter datasets that starts from the beginning when reaching the end.

Smoothness-Inducing Adversarial Regularization (SMART) (Jiang et al., 2020)
In addition to the original loss of each task, we add a regularization loss term to each loss for better generalization ability of the model. Specifically, the loss function for each task is:

$$L(\theta) + \lambda_s R_s(\theta) \tag{2}$$

where $L(\theta)$ is the original loss of each task, and λ_s, $R_s(\theta)$ are the weight regularization strength and SMART loss, respectively.

$$R_s(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{||\tilde{x_i} - x_i||_p \leq \epsilon} l(f(\tilde{x_i}; \theta), f(x_i; \theta)) \tag{3}$$

where $\epsilon > 0$ is a tuning parameter and l is the divergence between two outputs. Intuitively, this regularization term favors the θ at stable locations, where a small perturbation would not change the output too much.

4 Experiments

4.1 Data

1. **Sentiment Analysis**
   Stanford Sentiment Treebank (SST) Dataset
   The SST dataset consists of 11855 single sentences extracted from movie reviews. It is parsed with the Stanford parser to generate phrases, which are then labeled by human judges to have **negative, somewhat negative, neutral, somewhat positive, or positive**. We will use categorical encodings for these 5 labels (from 0 to 4).
2. **Paraphrase Detection**
   Quora Dataset
   It contains 202,152 question pairs with labels indicating whether they are paraphrase of each other. This can be treated naturally as a binary classification problem.
3. **Semantic Textual Similarity (STS)**
   SemEval STS Benchmark Dataset
   It consists of 8628 sentence pairs with labels from 0 (unrelated) to 5 (equivalent meaning) indicating their semantic similarity. Note that we form it as a regression problem with continuous logits produced by the model.

4.2 Evaluation method

1. For **Paraphrase Detection**, the logits are fed into sigmoid and rounded to produce predictions, which are then used to calculate the accuracy.

2. For **Sentiment Analysis** using Stanford Sentiment Treebank Dataset, F1 score and accuracy score are calculated
3. For **Semantic Textual Similarity**, **Pearson product-moment correlation coefficients** are calculated, with formula shown below.

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

4. Then we take the mean of those evaluated scores and evaluate the multitask performances.

4.3 Experimental details

We use 10% of the Quora dataset to test different hyperparameters and extensions of our model based on its performance on the dev set. Finally, we use 0.00001 as our learning rate and 10 as the batch size (10 for each dataset, 30 in total). And for the SMART loss regularization strength, we use 0.5 on SST and STS, but no regularization on paraphrase as it is not overfitting. Once the setting is finalized, we used 50% and the whole dataset on training.

4.4 Results

Method	SST dev Acc	Paraphrase dev Acc	STS dev Corr	Overall dev score
Baseline	0.511	0.395	0.245	0.384
Simple multi-task	0.491	0.732	0.445	0.556
With pretraining (with SMART, gradient surery)	0.328	0.765	0.642	0.5783
Final (SMART, gradient surgery, No pretraining)	0.514	0.789	0.727	0.677

Table 1: Experiment results for different methods

In our study, we constructed a BERT model for multi-task learning (MTL) utilizing several advanced techniques, including Gradient Surgery, SMART Loss, additional pretraining. Both Gradient Surgery and SMART Loss demonstrated substantial performance gains by effectively balancing the optimization objectives across multiple tasks. However, the additional pretraining and AdamaxW did not yield significant improvements, as shown in the table above.

5 Analysis

As mentioned previously, pretraining poses detrimental effect on the model performance. We propose several possibilities on that:

1. BERT is already pretrained on a large corpus of text on general domain. This enables it to capture grammatic and textual information in its weights. Our pretraining data on movie reviews and Quora question may be already captured in the original BERT weights. So further pretraining may not help the model capture more domain-specific features, but harm its ability on general texts.
2. In addition, we can see while the paraphrase and STS results are almost the same as the final model, SST accuracy are very low. We see this phenomenon at first without augmenting the pretraining dataset. We suspect that this may be caused by extremely imbalanced dataset mostly on Quora questions. But after augmentation, the result of SST is still low. We suspect that simple data augmentation does not help the model capture textual complexity.

6 Conclusion

This paper systematically investigated BERT's performance on multi-task learning and proposed several methods to enhance its performance over three different tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Our multiple adaptations to the base BERT model yielded a net gain in overall performance, signifying the potential for further refinement and optimization through future research. Specifically, our study found that Gradient Surgery SMART and Pearson correlation loss were effective techniques for improving multi-task learning with BERT. Data augmentation also proved to be useful in increasing the size of smaller datasets. However, additional pretraining and AdamaxW optimizer did not yield significant improvements. Limitations of our work include the lack of exploration of other multi-task learning architectures and hyperparameters, as well as the relatively small size of some of our datasets. Future work could focus on investigating different combinations of pretraining and fine-tuning techniques, as well as exploring other multi-task learning models and methods.

7 Code from other sources

In this paper, in the process of implementing various approaches, a few existing repositories gave us inspirations and references.

1. Further Pretraining: We use this repository for the masked LM pretraining of our BERT model. But as this repository is poorly maintained, we need to do tremendous amount of modifications on it to fit our model and data and fix version issues.
2. Smart Pytorch: We use this repository for the SMART Loss during training. Significant amount of input formating is done in our code to make our input compatible to its expected input.
3. Gradient Surgery: We use this repository for the gradient surgery of multitask learning.
4. AdamaxW Optimizer: Use the implementation of AdamxW optimizer in this repository.
5. Data Augmentation for Pretraining: Used this package to implement the data augmentation in pretraining. The package only offers fundamental sentence-transforming functions. We implement our own pipeline for augmenting the datasets.

References

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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, Minneapolis, Minnesota. Association for Computational Linguistics.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2014. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

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.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2020. Facebook fair's wmt19 news translation task submission. In *Proc. of WMT*.

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.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

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

Yichu Zhou and Vivek Srikumar. 2022. A closer look at how fine-tuning changes BERT. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1046–1061, Dublin, Ireland. Association for Computational Linguistics.