Multi-Task Learning for Language Model Fine-Tuning

Stanford CS224N Default Project

Anonto Zaman
Department of Aeronautics and Astronautics
Stanford University
`anontoz@stanford.edu`

Abstract

Language models have shown impressive performance across a number of tasks. When applying a particular model to multiple objectives, however, performance often suffers, owing in-part to complexities with individual training and stringent data requirements. Oftentimes, it may be desirable to optimize a network over multiple tasks simultaneously, but doing so is difficult and may lead to worse performance. In this work, we develop a simple multi-task language model for sentiment analysis, paraphrase detection, and semantic textual similarity tasks, based on the BERT architecture. At train time, we develop models that (1) train only the task specific layers, (2) train the entire model across all tasks simultaneously, and (3) utilize gradient surgery so task gradients do not conflict with one another. We find that multi-task training across the full model achieves the best performance. Though gradient surgery achieves adequate performance for semantic textual similarity, its performance for other tasks suffers, even compared to the model trained on the last task-specific layers. Though the multi-task model achieves reasonable performance, it may be limited by its relative simplicity. Ultimately, this project was a useful exercise in learning how to use and develop language models.

1 Key Information to include

- Mentor: Olivia Lee
- External Collaborators (if you have any): N/A
- Sharing project: No

2 Introduction

In recent years, advances in deep learning have yielded rapid developments in the performance of language models. Despite their impressive capabilities, modern language models still require large amounts of training data and may struggle to generalize across different tasks. To address this issue, a variety of open source language models have been developed. Most notably, the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. (2019)) architecture is a flexible, pretrained language model that is suitable for a variety of tasks. Though some large language models such as LLaMA (Touvron et al. (2023)) and GPT-3 (Brown et al. (2020)) have been successfully used across a variety of language tasks, BERT remains popular for its comparatively lightweight architecture.

In this paper, we explore how gradient surgery may be used to improve the performance of language models for downstream tasks. We leverage a minimalist version of the BERT architecture, minBERT, for sentiment analysis, paraphrase detection, and semantic textual similarity tasks. The gradient surgery method is benchmarked standard multi-objective gradient descent and individual task-specific training approaches.

3 Related Work

Multi-task learning has previously been explored for news recommendations using BERT (Bi et al. (2022)). The authors trained their model across a main task of news recommendation and auxiliary tasks of category classification and named entity recognition. The authors found that their model achieved improved performance over the standard BERT architecture. They additionally found that combining multi-task learning with attentive multi-field learning was most effective for improving performance.

Yu et al. (2020) proposes a method for multi-objective optimization called projecting conflicting gradients (PCGrad). When gradients from different tasks conflict, each is projected onto the plane of the other and then passed to the optimizer. By directly altering the gradients, the algorithm prevents interfering components of the gradient from being applied. The authors provide a theoretical proof of the optimization conditions when PCGrad outperforms traditional multi-task optimization. In their empirical analysis, they assessed PCGrad's performance on a variety of multi-task problems, such as scene understanding and goal-conditioned RL. The results showed that their method yielded improvements in optimization speed, data efficiency, and performance. Additionally, their method can be implemented alongside prior state-of-the-art methods.

Though previous literature has explored using PCGrad for optimizing multi-task models, this work aims to specifically measure its benefit over simple, multi-objective optimization.

4 Approach

The original BERT model consists of a multi-layer bidirectional transformer architecture. The model outputs the contextualized embedding for each word piece of the sentence (i.e. the encoder output) and the classification ([CLS]) token. The model was pretrained on two unsupervised masked language modelling and next sentence prediction tasks using Wikipedia articles (Devlin et al. (2019)).

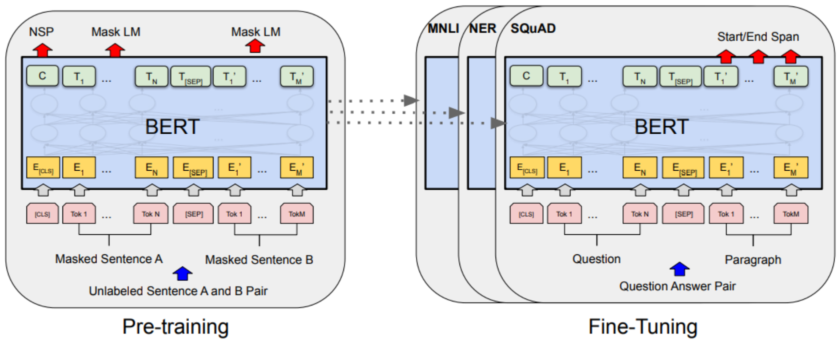

Figure 1: Visualization of BERT pre-training and fine tuning architectures (Devlin et al. (2019)). In the original paper, BERT was fine-tuned on question-answer tasks.

In developing the multi-task language model, one of the key considerations was model complexity. For each task, a single layer was appended to the original BERT model for later tasks. Though additional layers may have improved performance, the single-layer approach minimizes the number of parameters and improves training speed. Note that the BERT [CLS] tokens were used as the inputs for all downstream tasks. All [CLS] tokens were passed through a dropout layer before being fed to the subsequent layers.

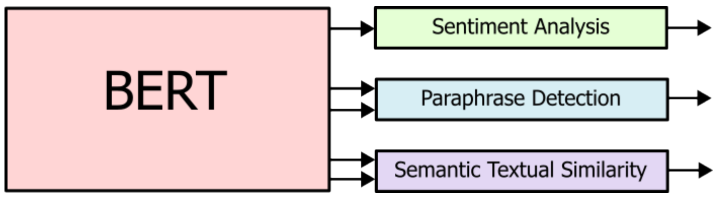

Figure 2: Visualization of multi-task model architecture.

4.1 Sentiment Analysis

For the sentiment analysis task, the objective is to rate the sentiment of a phrase on a discrete, numerical scale. To do so, the BERT [CLS] tokens were passed through a single linear layer, outputting five logits, as shown in Equation 1, where $\mathbf{x}$ is the [CLS] embedding from BERT. The loss function for this task was chosen as the cross-entropy between the predicted labels and true labels, as shown in Equation 2, where x_i is the logit of the true label and C is the number of possible sentiment values.

$$SA(\mathbf{x}) = W_{SA}\mathbf{x} + b_{SA} \tag{1}$$

$$\ell_{SA}(\mathbf{x}) = -\log\left(\frac{\exp(x_i)}{\sum_{c=1}^{C}\exp(x_c)}\right) \tag{2}$$

4.2 Paraphrase Detection

The paraphrase detection task takes two sentences and outputs a label over whether one is a paraphrase of the other. To accomplish this goal, the [CLS] tokens of two phrases were passed through a bilinear layer, outputting a single logit over the probability that they are similar, as seen in Equation 3. The bilinear layer was chosen in this case because it seemed like a simple and efficient way to compare two phrases. As with the sentiment analysis task, the loss was computed by taking the binary cross-entropy between the true and predicted label.

$$PD(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_2^{\mathrm{T}} W_{PD}\mathbf{x}_1 + b_{PD} \tag{3}$$

4.3 Semantic Textual Similarity

Like the paraphrase detection task, the semantic textual similarity (STS) task takes two phrases and computes their similarity on a numerical scale. To accomplish this, we again utilize a single bilinear layer with the two [CLS] tokens as inputs. The bilinear layer outputs a discrete probability distribution over the similarity values. For training, the cross-entropy is used between the predicted distribution and the true label.

$$STS(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_2^{\mathrm{T}} W_{STS}\mathbf{x}_1 + b_{STS} \tag{4}$$

4.4 Gradient Surgery

For multi-task learning, one approach is to take the overall loss as the sum of losses from each individual task:

$$\ell = \ell_{SA} + \ell_{PD} + \ell_{STS} \tag{5}$$

When training, however, the gradients from the losses may conflict with one another, making it difficult to efficiently optimize over the model parameters. To address this issue, we leverage the PCGrad algorithm, proposed by Yu et al. (2020). When the gradients for task i and j, denoted by $\mathbf{g}_i$ and $\mathbf{g}_j$ respectively, conflict with one another (i.e. $\mathbf{g}_i \cdot \mathbf{g}_j < 0$), then we subtract the projection of $\mathbf{g}_i$ onto $\mathbf{g}_j$, as shown in Equation 6. By completing this projection, we hope to reduce the influence of conflicting gradients when training. Note that the code implementation of PCGrad was obtained from (Tseng (2020)).

$$\mathbf{g}_i^{PC} = \mathbf{g}_i - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{||\mathbf{g}_j||^2}\mathbf{g}_i \tag{6}$$

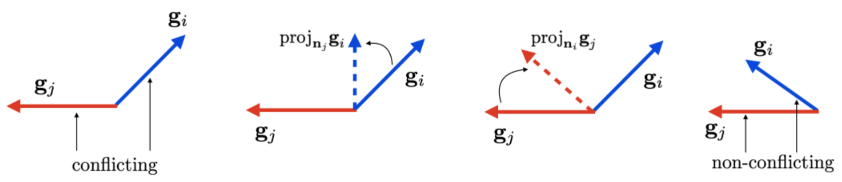

Figure 3: Illustration of gradient surgery method(Yu et al. (2020)). The conflicting gradient is projected onto the same direction as the other task.

5 Experiments

We trained a variety of models, some of which were optimized over the last linear layers, while others were optimized over the full model architecture. All model training and evaluations were completed using the Google Cloud Platform with a single NVIDIA T4 GPU.

5.1 Data

For the sentiment analysis task, we utilized the Stanford Sentiment Treebank (SST) dataset (Socher et al. (2013)), consisting of single sentences from movie reviews. Each sentence was ranked by human judges and has a label corresponding to its positiveness. The dataset was split into 8,544 training examples, 1,101 development examples, and 2,210 test examples.

Our dataset for the paraphrase detection task was obtained form Quora, consisting of pairs of questions and a label as to whether they are paraphrases of each other (Iyer et al. (2017)). The dataset consists of 283,010 training examples, 40,429 development examples, and 80,859 test examples.

The STS task was trained using the SemEval STS Benchmark Dataset, ranking whether pairs of sentences are similar according to a 5-point scale (Agirre et al. (2013)). The dataset consists of 6,040 training examples, 863 development examples, and 1,725 training examples.

5.2 Evaluation method

The performance of the sentiment analysis and paraphrase detection tasks were evaluated using the accuracy, simply dividing the number of correct detections by the number of total examples. For the STS task, performance was evaluated using the Pearson correlation of the true similarity values against the predicted similarities.

5.3 Experimental details

For training, we utilized the Adam optimizer with a learning rate of 1e-5. The following model configurations were considered:

- **Random**: Complete no additional training, simply evaluate performance of random parameters in the task-specific layers. Used as a baseline to evaluate performance of subsequent models.
- **Last Linear Layer**: Fix the parameters in the BERT model and train only the task-specific linear and bilinear layers. This model is used as the baseline for the test sets, as it is the simplest-possible model implementation and requires the least training effort.
- **Full Model Multi-Task**: Train the entire model simultaneously, using the loss function defined in Equation 5.
- **Full Model PCGrad**: Train the entire model using the gradient surgery approach defined in Equation 6.

Each model was trained for 10 epochs with a batch size of eight samples per iteration. Because the Quora dataset for paraphrase detection included so many training examples, we randomly selected a subset of 8,000 examples per epoch, rather than looping through the entire training set. This decision was made in the interest of increasing training speed. Each model took approximately 1-1.5 hours to train.

5.4 Results

The results from the development set are shown in Table 5.4 whereas the test results are shown in Table 5.4. As expected, all models achieve better performance than the random model in the development set. We find that the best performing model across all metrics is the full model trained with the sum of losses. Surprisingly, the PCGrad model achieves worse performance than the last linear layer model on the sentiment and paraphrase accuracy tasks across both the development and test sets. Because the gradients were propagated through the entire model, we would expect the overall performance to be better than training over only the last layer. Instead, it appears that we achieve only modest improvements in performance on the STS task.

One potential reason for PCGrad's poor performance relative to the simultaneous model is the choice of optimizer. Because Adam stores a moving estimate of the first and second moments of the gradient, adjusting the gradient direction may lead to poor estimates of the moments. This may cause the optimizer to produce steps that are suboptimal for learning.

Model	Sentiment Acc.	Paraphrase Acc.	STS Pearson
Random	0.144	0.407	-0.016
Last Linear Layer	0.395	0.631	0.254
Full Model Multi-Task	**0.480**	**0.753**	**0.380**
Full Model PCGrad	0.371	0.574	0.327

Table 1: Model performance on development set.

Model	Sentiment Acc.	Paraphrase Acc.	STS Pearson
Last Linear Layer	0.412	0.631	0.175
Full Model Multi-Task	**0.515**	**0.756**	**0.292**
Full Model PCGrad	0.352	0.576	0.290

Table 2: Model performance on test set.

6 Analysis

Even though we used only a subset of training examples for the paraphrase task, the model still achieves impressive levels of performance. This result makes sense considering the relative simplicity of the task - even though approximately 8000 examples were used in each training epoch for the sentiment and paraphrase tasks, the paraphrase task requires only a single output, whereas the sentiment task requires a distribution over multiple outputs. We would expect that the model's performance would achieve additional gains if we optimized over the full training set.

A major design choice in the model was its lack of depth - each task feature only a single layer appended to the BERT model. This choice was made deliberately to reduce the number of parameters and speed up training time. In practice, one would expect that additional layers may lead to better performance, though the number of layers and dimension of hidden states is not immediately obvious.

One of the key challenges in this work was training for the STS task. One of our early approaches involved directly outputting an estimate of the score, normalized between 0 and 5 using a sigmoid function. We found that this approach yielded poor performance (< 0 Pearson score). By instead optimizing over a probability distribution of scores, we hoped to improve the model's performance and achieve a higher STS score. It is possible that with enough training examples, a single-output model optimized over the L2-loss with the true similarity value would achieve better performance than the chosen architecture.

The difficulty of optimizing over the STS task may also illuminate the poor performance of the PCGrad model. In the test set, we note that the PCGrad model achieves similar performance to the multi-task model for the STS task. This may indicate that the PCGrad method biases the gradients towards the task with the highest gradients, leading to improved performance for the STS task at the detriment of the other objectives.

7 Conclusion

In this work we developed a multi-task language model built off of the BERT architecture for sentiment analysis, paraphrase detection, and semantic textual similarity. Our model consists of single-layer multi-headed outputs for each task. We found that training over the sum of losses for each objective achieved the best performance. Though the gradient surgery method performed better than the random baseline, it failed to improve over the standard multi-task training method and was outperformed by the last linear layer method in two of the tasks.

Beyond the technical contributions of this work, a large portion of this project was spent getting familiar with the engineering principles behind language models. Much of the effort for this project involved tuning loss functions and model hyperparameters, and getting familiar with how design choices impact the system's overall performance. As such, this endeavor was incredibly useful as a learning exercise for writing language models.

Future avenues of research could include using the cosine similarity to compute the STS scores, rather than the bilinear layer. This method was used by Reimers and Gurevych (2019) and achieved strong performance compared to baselines.

8 Ethics Statement

As with all machine learning tasks, there are ethical considerations with regard to bias in the provided dataset. It is possible that any model trained on the datasets will perpetuate these biases and present them in any output. Biases are especially important to consider in language tasks, as human-written sentences are inherently informed by the authors' cultural background. To address this limitation, we could try and leverage a variety of datasets from different sources, especially those with known diversity in terms of authorship and viewpoint (i.e. datasets written by non-native English speakers).

Another ethical concern is that any training paradigm requires significant computational resources, which is not accessible to all NLP developers. This project utilized Google Cloud GPUs and would have financially taxing without the credits provided by the CS224N teaching staff. To address this concern, we could publish the final models online and open-source, making them accessible for other researchers to test and further develop.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

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.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. 2017. First quora dataset release: Question pairs.

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.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Wei-Cheng Tseng. 2020. Weichengtseng/pytorch-pcgrad.

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