Freeze Your Layers

Stanford CS224N Default Project

Gautham Gorti
Department of Statistics
Stanford University
ggorti3@stanford.edu

Alex Thiesmeyer
Department of Statistics
Stanford University
thiesmey@stanford.edu

Abstract

To achieve high performance on natural language processing tasks requiring supervised learning, empirical work has demonstrated the success of pretraining a language model on a large amount of text data from a general domain and then finetuning the model on a smaller amount of data from the problem domain. Within a multi-task learning framework, the model is simultaneously finetuned on multiple different datasets for different tasks. With a variation of the widely used BERT model serving as our representation learner, we investigate the best way to use multi-task learning to achieve high performance on three distinct tasks. We find that a simple implementation of such a model performs moderately well on all tasks, and we are only able to improve performance in relation to such an implementation by freezing layers of our version of BERT during training.

1 Introduction

The BERT model architecture was introduced in Devlin et. al, 2019 [1]. BERT is a multi-layer Transformer with bidirectional attention, derived from the work of Vaswani et. al, 2017 [2]. BERT takes a sequence of tokens as input, with a special `[CLS]` token prepended at the start of each sequence. As output, BERT returns the final hidden state of each token. The final hidden state of the `[CLS]` token serves as an embedding representation of the sequence as a whole. As a training objective, BERT uses masked language modeling, wherein certain input tokens are masked and the model is tasked with predicting the true value of the masked tokens.

Devlin et. al, 2019 demonstrated that high-performing models for a variety of tasks could be developed by pretraining BERT on a large corpus and then finetuning the model, with some additional layers, on a task-specific dataset [1]. The intuition behind this process is that, through pretraining, BERT learns how to output high-quality general sentence representations which can be used as input in supervised learning settings.

The creators of BERT finetuned separate BERT models for each specific task they addressed. Effectively, they built a toolset, where each tool in the set had a particular purpose. Ideally, rather than carry around a large, cumbersome box of tools, we would like to develop some kind of multitool. That is, a single model which can achieve high performance on multiple different downstream tasks.

We use a variation of BERT, minBERT, and the pretraining-finetuning paradigm, but by working within the framework of multi-task learning, we develop a single model for use on three different tasks. In particular, we adapt minBERT for sentiment analysis, paraphrase detection, and semantic textual similarity evaluation.

2 Related Work

After Devlin et. al, 2019, the utility of finetuning BERT for a variety of text classification tasks was explored in Sun et. al, 2019 [1][3]. The authors used multi-task learning as an extra step after pretraining to encourage BERT to learn robust embeddings, but proceeded to finetune the model to each task individually.

Liu et. al, 2019 used multi-task learning with BERT to achieve state-of-the-art performance on ten text classification tasks, including sentiment analysis and semantic textual similarity [4]. Our work can be seen as a minimalist version of theirs. Indeed, our dataloader for multi-task learning is derived from theirs, and we implement the SMART regularization technique from the related work Jiang et. al, 2020 [5].

Various other work has explored how to use BERT to address the specific tasks we focus on. In particular, Reimers and Gurevych, 2019 showed that for semantic textual similarity tasks, rather than having a sentence pair separated by a `[SEP]` token as a single input to BERT, the two sentences could be input individually into BERT and their output could be compared via cosine-similarity [6].

3 Approach

3.1 Task-Specific Heads and Dataloader

As a representation of each input sentence, minBERT puts the final hidden state of the `[CLS]` token through a pooling layer consisting of a multi-layer perceptron with tanh activation. Let y be the output of this multi-layer perceptron.

Our task-specific layers, or head, for sentiment analysis returns

$$W_2^{\mathrm{SA}}(\mathrm{GELU}(W_1^{\mathrm{SA}}y + b_1^{\mathrm{SA}})) + b_2^{\mathrm{SA}} \in \mathbb{R}^5$$

A dropout layer is applied after the activation function. Cross entropy loss is used for training.

For paraphrase detection, if y_1 and y_2 are the pooled outputs of minBERT for two sentences, then we return

$$(W_1^{\mathrm{Para}}y_1 + b_1^{\mathrm{Para}})^\top(W_2^{\mathrm{Para}}y_2 + b_2^{\mathrm{Para}}) \in \mathbb{R}$$

Cross entropy loss is used during training.

Our head for STS is similar. Let

$$z_1 = W_1^{\mathrm{STS}}y_1 + b_1^{\mathrm{STS}}$$
$$z_2 = W_2^{\mathrm{STS}}y_2 + b_2^{\mathrm{STS}}$$

We return $5 \cdot \max(0, \cos(z_1, z_2))$, where $\cos(z_1, z_2)$ is the cosine similarity between z_1 and z_2. Mean-squared error loss is used during training.

For training without gradient surgery, our dataloader is directly adapted from the algorithm given in Liu et al., 2019 [4]. Each dataset is separated into mini-batches and then pooled with the mini-batches of the other datasets. During training, a mini-batch is randomly chosen from this set of mini-batches, and the task corresponding to the selected mini-batch is used to calculate the model's gradients.

3.2 Gradient Surgery

For training with gradient surgery, we used a different dataloader. The three datasets for each task are consolidated to form one dataset. When sampling batches, the dataloader samples uniformly from the consolidated dataset, without replacement. During training, samples within the batch are grouped by task and then fed into the model to produce up to three gradient vectors—one for each task present in the batch. It may be the case that gradients backpropagated from different tasks point in conflicting directions; if so, then the descent direction for one task may be a poor direction for other tasks. To alleviate this issue, we implement an intuitive "gradient surgery" procedure from Yu et. al, 2020 that projects gradients onto the subspace orthogonal to the other gradients [7]. In our implementation, we select the dominant task gradient and project out conflicting components of the other gradients.

Let $g_{sa}, g_{para}, g_{sts} \in R^{N_{bert}}$ denote the averaged, flattened gradient vectors of the minBERT backbone from each task in a given batch (N_{bert} denotes number of parameters in the minBERT model). WLOG, suppose $\|g_{sa}\| \geq \|g_{para}\|,\ \|g_{sts}\|$. We calculate

$$g'_{para} = g_{para} - \mathbb{1}\{g_{para} \cdot g_{sa} < 0\}\frac{g_{para} \cdot g_{sa}}{\|g_{sa}\|^2}g_{sa},$$

$$g'_{sts} = g_{sts} - \mathbb{1}\{g_{sts} \cdot g_{sa} < 0\} \frac{g_{sts} \cdot g_{sa}}{\|g_{sa}\|^2} g_{sa},$$

and the new descent direction is

$$g_{sa} + g'_{para} + g'_{sts}.$$

3.3 Regularization

The primary issue we faced during experimentation was that our model's performance on the validation set was substantially lower than on the training set. This problem indicated that the model was overfitting. To correct for this, we first attempted to reduce the number of parameters in each of our heads, for example, by replacing the multi-layer perceptron in the sentiment classifier with a single linear layer. Such changes had a marginal effect on performance. We turned to two more powerful methods of regularization.

First, we tested the effect of keeping some of minBERT's layers frozen during finetuning. We assumed that if the heads were not responsible for overfitting, then likely minBERT had too much flexibility to change from its pretrained parameters.

We also experimented with a more advanced form of regularization, called SMART, introduced in Jiang et. al, 2020 [5]. Their method of regularization has two parts. The first is a smoothness-inducing adversarial regularizer, which they define as

$$\mathcal{R}_{\mathrm{s}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_p \leq \epsilon} \ell_{\mathrm{s}}(f(\tilde{x}_i; \theta), f(x_i; \theta))$$

where the x_i are the embeddings of the input sentences, ℓ_s is some loss function, and ϵ and p are hyperparameters. This penalty restricts the ability of f to vary widely in a small neighborhood around each input, hence smoothing the function, as the terminology suggests.

The second part of the regularization is a Bregman proximal point optimization method. Specifically, for two sets of model parameters θ_1 and θ_2 we define the Bregman divergence

$$\mathcal{D}_{\mathrm{Breg}}(\theta_1, \theta_2) = \frac{1}{n} \sum_{i=1}^{n} \ell_{\mathrm{s}}(f(x_i; \theta_1), f(x_i; \theta_2))$$

While the smoothness-inducing adversarial regularizer works directly against overfitting by incentivizing simpler model geometry, the Bregman divergence addresses the issue by penalizing weight updates which have a large impact on the model.

If $\mathcal{L}(\theta)$ is the original loss function, and $\theta_{\mathrm{current}}$ is the current set of model parameters, then the new loss is given by

$$\mathcal{L}(\theta) + \lambda_{\mathrm{s}} \mathcal{R}_{\mathrm{s}}(\theta) + \mu \mathcal{D}_{\mathrm{Breg}}(\theta, \theta_{\mathrm{current}})$$

for hyperparameters λ_s and μ.

For our implementation, we primarily followed the example of Jiang et. al, 2020, using their description of their algorithm as well as their code [1][5]. θ was updated with momentum, as described in their work. Symmetric Kullback-Leibler divergence was used as the loss ℓ_{s} for sentiment analysis and paraphrase detection, and mean-squared error loss was used for semantic textual similarity. p was set to be infinity. A direct adaptation was not possible though, as their method supposes that the model f is a function of a single sentence x. Two of our model heads are functions of sentence pairs instead, taking the form $f(x_i, y_i; \theta)$. To resolve this difference, for these tasks we set

$$\tilde{x}_i = \operatorname*{argmax}_{\|\tilde{x}_i - x_i\|_p \leq \epsilon} \ell_{\mathrm{s}}(f(\tilde{x}_i, y_i; \theta), f(x_i, y_i; \theta))$$

$$\tilde{y}_i = \operatorname*{argmax}_{\|\tilde{y}_i - y_i\|_p \leq \epsilon} \ell_{\mathrm{s}}(f(x_i, \tilde{y}_i; \theta), f(x_i, y_i; \theta))$$

and return

$$\mathcal{R}_{\mathrm{s}}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell_{\mathrm{s}}(f(\tilde{x}_i, \tilde{y}_i; \theta), f(x_i, y_i; \theta))$$

[1] `https://github.com/namisan/mt-dnn`

3.4 Baseline

We developed a baseline model for minimal functionality. Our baseline model used the same sentiment analysis and paraphrase detection heads as our final model, as well as the same dataloader. For semantic similarity, the baseline used a head identical to the paraphrase detection head. The baseline did not make use of any of the other methods described above.

4 Experiments

4.1 Data

The Stanford Sentiment Treebank (SST), Quora Question Pairs dataset [2], and SemEval STS Benchmark datasets were used for the sentiment analysis, paraphrase detection, and semantic textual similarity tasks, respectively [8][9].

The SST dataset consists of 11,855 sentences from movie reviews, each with a discrete sentiment label from 0 (negative) to 4 (positive). The Quora paraphrase dataset contains 400,000 question pairs with binary labels denoting whether the questions are paraphrases of each other. The SemEval STS dataset contains 8,628 sentence pairs, each with a continuous label from 0 to 5 denoting their semantic similarity.

4.2 Evaluation method

For evaluating the performance of our model on sentiment analysis and paraphrase detection, we used the proportion of correct label predictions. For semantic textual similarity, we used the Pearson correlation between our predictions and the true score values.

To understand the performance of our model within a wider context, these scores were compared with scores submitted to the class leaderboard.

4.3 Experimental process

Our experimental process was guided by attempting to modify the model to improve on our baseline's performance. We found this surprisingly difficult to achieve. Most of our experimental work focused on testing different model configurations.

Different heads were tested for each task. Both increasing and decreasing complexity of model heads had marginal impact on performance.

Gradient surgery did not have a substantial impact on performance, and was discarded. SMART regularization improved performance on the semantic textual similarity head when that head was finetuned in isolation. There was no improvement for the other tasks in isolation, and no improvement when doing multi-task finetuning. SMART was discarded.

When finetuning heads in isolation, freezing minBERT layers had marginal effect on the sentiment analysis and paraphrase detection tasks, but improved performance on semantic textual similarity. This pattern continued to hold true when doing multi-task learning as well, so freezing layers was deemed beneficial. We tried freezing layers just when updating the model with respect to semantic textual similarity as well as freezing layers for all tasks, and found the latter approach more successful.

For our final model configuration, we froze all the layers of minBERT except the final three and the pooling layer. A learning rate of 1e-4 was used for 9 epochs.

[2] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

4.4 Results

Task	Baseline (Dev)	Final (Dev)	Final (Test)
Sentiment Analysis	0.508	0.498	0.518
Paraphrase Detection	0.811	0.817	0.819
Semantic Similarity	0.551	0.690	0.643
Overall	0.623	0.668	0.660

Final results on validation and test sets

We were only able to noticeably improve the model's performance on semantic textual similarity in relation to the baseline. The most important reason behind the improvement for this task was the decision to freeze layers of minBERT during training. Unfortunately, despite the fact that the model seemed to be overfitting for sentiment analysis and paraphrase detection as well, freezing layers had little impact on performance for these tasks. We were disappointed to find that none of the other approaches we implemented were able to affect performance on these two tasks in a meaningful way.

5 Analysis

We reviewed the qualitative performance of our model across the three tasks on the validation dataset.

5.1 Sentiment Analysis

For sentiment analysis, we were interested in three aspects of our model: which sentences it classifies well, which sentences it struggles with, and how prediction accuracy varies across classes. Out of correctly classified sentences, the one the model predicted with the highest assigned probability was the positive, class 4 review

> a gorgeous, high-spirited musical from india that exquisitely blends music, dance, song, and high drama.

which includes three strongly positive adjectives/adverbs. In contrast, the sentence it assigned the lowest probability to the true class to was

> hilariously inept and ridiculous.

This phrase is labeled as class 3, but the model classified it as a 0. The phrase only has three words from which sentiment can be inferred. On their own, the word "inept" is negative, "ridiculous" is slightly negative, and "hilariously" is positive. Inferring a mostly positive overall sentiment, then, is quite challenging. Even for a human evaluator, with this little context the phrase could easily be interpreted as a class 1 negative review.

The model achieved the following per class accuracies

Class	0	1	2	3	4
Accuracy	0.42	0.73	0.07	0.66	0.47

Sentiment classifier per class accuracies

The model performs best on classes 1 and 3, but is strikingly bad at correctly classifying neutral sentences. We note that classes 1 and 3 combined make up about 50% of the training data, so that may explain the higher performance on those labels, but the rest of the training data is approximately evenly split between the other three classes. Hence we cannot ascribe the poor performance on class 2 as a consequence of the model not seeing enough such examples as it learned. Rather, we suppose that neutral sentences may be hard to classify. The model may be able to minimize loss by treating the five-class classification problem more as a two-stage process: determine whether the sentence is mostly positive or negative, then determine the degree of sentiment in the chosen category.

5.2 Paraphrase Detection

We first looked at cases where the model correctly classified sentence pairs. The pair for which the model assigned the highest probability to their being paraphrases of each other was

> why are saltwater taffy candy imported in the philippines?
> why is saltwater taffy candy imported in the philippines?

This pair differs only on the point of whether candy is a plural or singular noun, so we are glad that the model performs well here. Similarly, the pair for which model assigned the highest probability to their not being paraphrases of each other was

> which is the best c programming institute in patna?
> how much do castles and manors costs?

two completely unrelated questions.

For failure cases, we looked at the pair that the model incorrectly classified as paraphrases with the highest confidence, and the pair that the model incorrectly classified as not being paraphrases with the highest confidence. The former pair was

> why spotify is not available in india?
> is spotify not available in india?

and the latter pair was

> do native speakers of spanish ever confuse ser and estar?
> what are the ways to quickly learn the use of the two spanish verbs "ser" and "estar"? do native hispanophones ever confuse their use?

In the first case, the sentences are almost lexically identical. The model is not able to recognize the importance of the word "why" to the overall meaning of the question. The failure of the model in the second case is more interesting. Only the second part of the second question is a paraphrase of the first question. The first part is a separate question altogether. It is not suprising that the model misclassified this pair, but it is surprising it did so with high confidence, given the high degree of lexical overlap.

5.3 Semantic Textual Similarity

We fit a linear regression between our model's predicted values and the true similarity scores and calculated the residuals. The sentence pair with the smallest residual was

> a man and woman walking past a record shop
> a man and woman kissing in front of a crowd of people.

This pair had a true score of 0.2. The model was smart enough to not be confused by the identical start to each sentence. The man and woman are doing two highly different activities in each. The sentence pair with the largest residual was

> non-proliferation expert at the international institute for strategic studies mark fitzpatrick stated that the iaea report had an unusually strong tenor.
> senior fellow at the international institute for strategic studies mark fitzpatrick stated that the international atomic energy agency plan is superficial.

This pair had a true score of 4. This score is surprising—the sentences differ acutely in their description of the agency's report, which is key to each sentence's meaning. Poor performance on this example may actually reflect good judgement by the model.

6 Conclusion

Our model's performance on all three tasks is reasonable, especially when the simplicity of the final configuration is taken into account. However, performance does not compare well to other models. We were disappointed in our inability to make a significant leap in any of the three tasks after we developed our baseline model. Our baseline was created with the sole intention of having a functional model. Hence, we imagined that large improvements in performance would be made immediately when more carefully considered methods were introduced. We were mostly proven wrong. We were surprised that SMART regularization was not the answer to our overfitting issues, and we question the validity of our implementation.

We heard from other groups during the poster presentation session that concatenating the sentence pairs for the paraphrase detection and semantic textual similarity tasks before inputting into minBERT had a large impact on performance. After a five minute alteration to our code implementing this change, without any optimization, our scores on the validation set were 0.494, 0.881, and 0.865, respectively. We are frustrated that this simple change was so instrumental to model performance, in particular because Reimers and Gurevych, 2019, a work which strongly informed our own, advises against this approach in favor of the two pass approach that we use [6].

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, June 2019. Association for Computational Linguistics.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[3] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer, 2019.

[4] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 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, July 2019. Association for Computational Linguistics.

[5] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 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, July 2020. Association for Computational Linguistics.

[6] Nils Reimers and Iryna Gurevych. 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, November 2019. Association for Computational Linguistics.

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

[8] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 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, 2013.

[9] Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *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, June 2013. Association for Computational Linguistics.