Improving BERT Fine-tuning with Embedding Normalization

Wenxuan Zhou, Junyi Du, Xiang Ren
University of Southern California
{zhouwenx, junyidu, xiangren}@usc.edu

Abstract

Large pre-trained sentence encoders like BERT start a new chapter in natural language processing. A common practice to apply pre-trained BERT to sequence classification tasks (e.g., classification of sentences or sentence pairs) is by feeding the embedding of [CLS] token (in the last layer) to a task-specific classification layer, and then fine tune the model parameters of BERT and classifier jointly. In this paper, we conduct systematic analysis over several sequence classification datasets to examine the embedding values of [CLS] token before the fine tuning phase, and present the biased embedding distribution issue—i.e., embedding values of [CLS] concentrate on a few dimensions and are non-zero centered. Such biased embedding brings challenge to the optimization process during fine-tuning as gradients of [CLS] embedding may explode and result in degraded model performance. We further propose several simple yet effective normalization methods to modify the [CLS] embedding during the fine-tuning. Compared with the previous practice, neural classification model with the normalized embedding shows improvements on several text classification tasks, demonstrates the effectiveness of our method.

1 Introduction

Transformer-based sentence encoders including BERT (Devlin et al., 2018) and its variants (Joshi et al., 2019; Liu et al., 2019) have achieved tremendous success on a wide range of natural language processing tasks. This demonstrates the power of transfer learning from large pre-trained language model to the downstream tasks with task-specific training data. In the current practice of applying BERT, two model transfer strategies, namely feature extraction and model fine-tuning, are widely adopted. The former one represents the input sequence as a linear combination of all layers in the Transformer model, where the weights are learnable, while the parameters in transformer remain freezed during the training stage of the downstream task. The latter one, as shown to be a more effective option (Peters et al., 2019), stacks a task-specific classifier on top of the pre-trained Transformer network, and updates the model parameters in both the classifier and the Transformer together, towards optimizing the downstream task with its training data. Specifically, for sequence classification, the embedding of the [CLS] token in the last layer of Transformer is fed into a fully connected network followed by a softmax classifier.

Despite the successes of fine-tuning pre-trained Transformers like BERT, the detailed mechanisms of how knowledge from pre-trained BERT are transferred to facilitate the downstream tasks is not yet well understood—e.g., whether the information stored in [CLS] token embedding can directly apply to the end task is still unclear. In the pre-training stage, the [CLS] token is either trained by next sentence prediction (NSP) or masked language model (MLM), which may introduce inductive bias that is irrelevant to end tasks. Also, during pre-training the input sentences are sampled from large-scale raw corpora, while in fine-tuning, the input sentences are sampled from training set of end tasks. This dismatch in training objectives and data distribution may lead to highly biased distribution in the [CLS] embedding—some dimensions have non-zero means or very large range of values. In the fine-tuning stage, these large embedding values may dominate the training process or lead to unstable gradients of model parameters in back propagation, which hinders parameter optimization or even cause gradient explosion.

In this paper, we analyze the bias in embedding distribution on BERT model, and propose to solve...
them by explicitly normalizing the \([\text{CLS}]\) embedding. We first estimate the distribution statistics of the \([\text{CLS}]\) embedding on the whole training set, then use these information to normalize each training example in fine-tuning. In this way, we fix the numeric issues while keep the original information in \([\text{CLS}]\) embedding. Experiments on four GLUE \cite{Wang2018} text classification tasks show that this simple method significantly boosts fine-tuning performance.

We briefly overview the fine-tuning process of BERT in Section 2, present our analysis of biased embedding distribution and its negative impact in Section 3, introduce a simple yet effective embedding normalization method in Section 4, and conduct experiments over several public datasets (as part of GLUE benchmark) in Section 5.

2 Fine-tuning BERT for Sequence Classification Tasks

For text classification, the task is to assign categories to text based on its content. It is one of the most fundamental problems in natural language processing. State-of-the-art methods in text classification take a pre-training fine-tuning process – they first train a text encoder on large-scale corpora in a self-supervised way, then adjust model parameters on end tasks in order to fit with their objectives.

In this paper, we use BERT as the base encoder. BERT takes an input sentence and outputs the contextual embedding of each token. A special token \([\text{CLS}]\) is padded at the beginning of the sentence in order to represent sentence-level information. During pre-training stage, the \([\text{CLS}]\) token is trained by next sentence prediction (NSP) objective, which is a binary classification task for predicting whether two segments follow each other. Specifically, given two segments \(s_1, s_2\) and final \([\text{CLS}]\) embedding \(h \in \mathcal{R}^{d_h}\), NSP is predicted by:

\[
P(s_2 \text{ follows } s_1) = \frac{e^{w_1 h + b_1}}{e^{w_1 h + b_1} + e^{w_2 h + b_2}},
\]

where \(w_1, w_2 \in \mathcal{R}^{d_h}\) and \(b_1, b_2 \in \mathcal{R}\) are trainable weights.

To adapt the \([\text{CLS}]\) embedding to end tasks, the pre-trained classifier is replaced by a new neural network to predict the task label in the fine-tuning stage:

\[
k = \text{FFN}(h),
\]

\[
P(c| h) = \text{Softmax}(w.k + b_c),
\]

where \(w_c \in \mathcal{R}^{d_k}, b_c \in \mathcal{R}\), \(\text{FFN} : \mathcal{R}^{d_h} \rightarrow \mathcal{R}^{d_k}\) are random initialized model parameters. In this work, we use a single layer fully connected layer followed by tanh activation:

\[
k = \tanh(W_f h + b_f),
\]

where \(W_f \in \mathcal{R}^{d_h \times d_k}, b_f \in \mathcal{R}^{d_h}\) are random initialized. In fine-tuning, both pre-trained and new model parameters are updated on text classification datasets by the cross entropy loss \(\mathcal{L}\).

3 Biased Embedding Distribution

We conduct systematic analysis over four sequence classification datasets from GLUE \cite{Wang2018} in this section to present the observations on biased embedding distribution and discuss its negative impact on the model performance.

To train neural networks from scratch, a common practice is to initialize model parameters by zero-centered i.i.d. random variables. It helps keep gradients of each layer in similar scale, which reduces numeric issues such as gradient exploding or vanishing. Previous work \cite{Sutskever2013} shows that if the model parameters are not carefully initialized, neural models would reach sub-optimal performance or even fail to learn. However, this problem does not get much attention in the context of fine-tuning neural models. During fine-tuning, the bottom network is initialized by a pre-trained neural network specialized in a different task and trained on different data distribution. Thus, the output embedding of the bottom network, which fed into the task-specific classifier, may be non zero-centered and have large range of value, and cause unstable gradients in the new-added classifier. For example, in BERT model, the gradients of the FFN layer are:

\[
\frac{\partial \mathcal{L}}{\partial W_f} = \frac{\partial \mathcal{L}}{\partial (W_f h + b_f)} \otimes h,
\]

\[
\frac{\partial \mathcal{L}}{\partial b_f} = \frac{\partial \mathcal{L}}{\partial (W_f h + b_f)}.
\]

If the value of a element in \(h\) is large, some dimensions of the gradient of \(W_f\) will be amplified, which leads to higher variance in gradient. Large variance is a severe problem especially when the model is trained with relatively small batch size. It causes numeric problems like gradients exploding and result in inefficient optimization of parameters \cite{Pascaru2012}.
Figure 1: Visualization of the distribution of pre-trained [CLS] embedding on the training set of different fine-tuning tasks by t-SNE. Different colors stand for different class labels.

Figure 2: Average value of each [CLS] embedding dimension on the training set. Some dimensions have very biased values.

Also, the non-zero centered distribution of output embedding is increasing the difficulty of fitting data. In fine-tuning, parameters in the new-added classifier are usually initialized by zero-centered i.i.d random variable. It requires more updates for a classifier with a biased embedding as input to converge than that with a normalized embedding. An intuitive example of is given in Figure 3. When the embedding \( h \) is biased, the optimal solution \( f^* \) also contains large bias, which is harder to learn than the normalized embedding \( \tilde{h} \).

To see whether BERT suffers from problems mentioned above, we visualize the distribution of [CLS] embedding of BERT-base on 4 text classification datasets, and show the average value of each embedding dimension in Figure 1 and Figure 2. We observe that the pre-trained [CLS] embedding of new datasets fall into two clusters, because the [CLS] embedding is trained with next sentence prediction (NSP) objective in BERT. This shows the distribution of [CLS] embedding is skewed by the NSP task in pre-training. In terms of each embedding dimension, we find that most dimensions has an average value within 0.5, while some dimensions have very large average values. In back propagation, these dimensions will dominate the gradients, and even lead to gradient explosion problem.

4 Proposed Method

To solve the above problem, we propose to explicitly normalize the embedding of [CLS] embedding in fine-tuning. Our major objectives are 1) eliminating large numeric values in embedding so as to stabilize backward gradients and 2) making the embedding distribution closer to 0 so as to facilitate learning the new classifier.

4.1 Embedding Normalization

We test the following widely-used data normalization methods:

- **Z-normalization.** Z-normalization (Goldin and Kanellakis, 1995) transforms the input vector into the output vector whose mean is near to 0 and standard deviation is near to 1. Specifically, given a set of input vectors \( X = \{x_1, x_2, ..., x_n\} \), Z-normalization computes the statistics of the whole dataset and transform each training example by:

  \[
  \hat{x}_i = x_i - \mu \frac{\sigma + \epsilon}{\sigma} ,
  \]

  where \( \mu \), \( \sigma \) are vectors representing the mean and standard deviation of each embedding dimension, \( \epsilon \) is a hyper-parameter to prevent small denominators. This method is also explored in fine-tuning setting (Varno et al., 2019), where it is theoretically proved to be effective for linear classifiers.

- **Min-Max Normalization.** Min-Max normalization linearly transforms each dimension of
Normalization $f^*$

Figure 3: An illustration of the non-zero centered distribution. The optimal solution on $h$ requires more updates than that on $\hat{h}$. It motivates us to normalize the embeddings in fine-tuning.

the input vectors to a range from -1 to 1:

$$\theta_1 = \min_{x \in \mathcal{X}} x$$

$$\theta_2 = \max_{x \in \mathcal{X}} x$$

$$\hat{x}_i = \frac{2x_i - \theta_1 - \theta_2}{\theta_2 - \theta_1 + \epsilon},$$

(2)

where $\min$ and $\max$ are element-wise minimum and maximum value of each dimension on $\mathcal{X}$. By stretching or compressing the embedding, this method ensures that each dimension is in the same scale and no extreme value occurs in the training set. This method is commonly used in computer vision to preprocess input images into unit scales.

- **L2 normalization.** L2 normalization transforms the L2 norm of each embedding dimension to an average value of 1:

$$\delta^2 = \mathbb{E}_{x \in \mathcal{X}} [x^2],$$

$$\hat{x}_i = \frac{x_i}{\delta + \epsilon}.$$  

(3)

This method constrains the L2 norm of each embedding not to be too large so as to ease the gradient exploding problem.

4.2 Model Fine-tuning and Prediction with Normalized Embedding

In fine-tuning, we first get the $[\text{CLS}]$ embeddings of each training example with pre-trained sentence encoder, then use them to calculate the normalization statistics ($\mu, \sigma$, etc.). In training phase, we use the normalization statistics to transform the $[\text{CLS}]$ embedding $h$ of input sentences to normalized embedding $\hat{h}$, which are taken as inputs for upper layers. For example, with L2 normalization, the output of the neural network becomes:

$$\hat{h} = \frac{h}{\delta + \epsilon},$$

$$k = \text{FFN}(\hat{h}),$$

$$P(c|h) = \text{Softmax}(w_c k + b_c),$$

where $\delta$ is a normalization statistics. Although the parameters of the pre-trained model is updated in fine-tuning, we do not update the normalization statistics for simplicity. In prediction phase, the $[\text{CLS}]$ embeddings of new examples are normalized in a similar way, using the normalization statistics of the training set.

5 Experiments

5.1 Experiment Settings

We implemented our methods on a BERT implementation. The model is optimized with AdamW optimizer using a learning rate of 5e-5 for BERT-base and 2e-5 for BERT-large. The learning rate is scheduled by a linear warmup for the first 6% of steps followed by a linear decay to 0. The model is fine-tuned for 10 epochs on each task. We apply early stopping according to task-specific metrics on the dev set. Other hyper-parameters are same as pre-training.

5.2 Main Results

The experiment results are presented in Table 1. Overall, our methods achieve consistent improvements over fine-tuning with the BERT-base encoder. Among three normalization methods, L2 normalization is the most effective one, while Min-Max normalization and Z-normalization sometimes given worse results. We think it is because that some dimensions have very small variance. Z-normalization and Min-Max normalization may greatly stretch the embedding distribution and amplify the noise in embedding. It can be solved with a more careful selection of $\epsilon$. With the BERT-large encoder, L2 normalization still helps, but the performance gains are much less. We also conduct experiments on MNLI (433k sentences) and QNLI (130k sentences) datasets with the BERT-base encoder. Results are presented in Table 2. Again, L2 normalization improves model performance on both datasets.

https://github.com/huggingface/transformers
Table 1: Results (mean ± std, median, and max) on the dev sets of GLUE from 5 runs with different random seeds.

| Method / Task | MRPC | RTE | SST-2 |
|---------------|------|-----|-------|
| BERT-base     | 88.2 ± 0.5 / 88.1 / 89.2 | 71.1 ± 1.1 / 71.5 / 72.2 | 91.9 ± 0.9 / 91.4 / 92.9 |
| BERT-base + Z-normalization | 88.6 ± 0.3 / 88.4 / 88.9 | 69.4 ± 2.0 / 69.7 / 72.2 | 92.2 ± 0.1 / 92.2 / 92.3 |
| BERT-base + L2 normalization | 88.7 ± 0.4 / 88.6 / 89.3 | 72.0 ± 0.5 / 71.8 / 72.9 | 92.2 ± 0.4 / 92.3 / 92.8 |
| BERT-large-wwm | 88.4 ± 0.8 / 88.9 / 89.1 | 73.4 ± 1.8 / 72.9 / 75.8 | 94.4 ± 0.2 / 94.4 / 94.6 |
| BERT-large-wwm + Z-normalization | 88.1 ± 1.3 / 88.6 / 89.3 | 74.4 ± 1.7 / 74.7 / 76.2 | 93.7 ± 0.3 / 93.7 / 93.9 |
| BERT-large-wwm + Min-Max normalization | 88.8 ± 0.5 / 88.9 / 89.5 | 73.7 ± 1.4 / 74.0 / 74.7 | 93.7 ± 0.3 / 93.5 / 94.0 |
| BERT-large-wwm + L2 normalization | 88.6 ± 1.0 / 88.8 / 90.0 | 74.8 ± 1.2 / 74.7 / 76.2 | 94.3 ± 0.2 / 94.4 / 94.5 |

Table 2: Results (median, max) on the dev sets from 3 runs with different random seeds. We use BERT-base as the sentence encoder.

| Method / Task | MNLI-m | MNLI-mm | QNLI |
|---------------|---------|---------|------|
| BERT-base     | 84.21 / 84.29 | 84.56 / 84.79 | 91.25 / 91.38 |
| Z-normalization | 84.01 / 84.05 | 84.67 / 84.91 | 91.14 / 91.34 |
| Min-Max normalization | 83.79 / 83.89 | 84.27 / 84.40 | 91.27 / 91.36 |
| L2 normalization | 84.52 / 84.66 | 84.91 / 85.40 | 91.38 / 91.58 |

6 Conclusion

In this paper, we analyzed the biased embedding distribution problem of [CLS] token in pre-trained sentence encoders. We showed that this problem may lead to numeric issues such as gradient exploding and increase the difficulty of fitting task-specific classifiers. To solve the problem, we proposed a simple yet effective method to regularize the embedding to smaller scales. Experiments on several text classification datasets proved the effectiveness of our method. Future work includes applying our methods to more datasets and encoders (such as RoBERTa and XLNet), and exploring more normalization methods.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Dina Q Goldin and Paris C Kanellakis. 1995. On similarity queries for time-series data: constraint specification and implementation. In International Conference on Principles and Practice of Constraint Programming, pages 137–153. Springer.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2019. Spanbert: Improving pre-training by representing and predicting spans. arXiv preprint arXiv:1907.10529.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019.

Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. CoRR, abs/1211.5063, 2.

Matthew Peters, Sebastian Ruder, and Noah A Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. arXiv preprint arXiv:1903.05987.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In International conference on machine learning, pages 1139–1147.

Farshid Varno, Behrouz Haji Soleimani, Marzie Saghayi, Lisa Di Jorio, and Stan Matwin. 2019. Efficient neural task adaptation by maximum entropy initialization. arXiv preprint arXiv:1905.10698.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461.