How to Fine-Tune BERT for Text Classification?

Chi Sun, Xipeng Qiu[*], Yige Xu, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{sunc17,xpqiu,ygxu18,xjhuang}@fudan.edu.cn

Abstract

Language model pre-training has proven to be useful in learning universal language representations. As a state-of-the-art language model pre-training model, BERT (Bidirectional Encoder Representations from Transformers) has achieved amazing results in many language understanding tasks. In this paper, we conduct exhaustive experiments to investigate different fine-tuning methods of BERT on text classification task and provide a general solution for BERT fine-tuning. Finally, the proposed solution obtains new state-of-the-art results on eight widely-studied text classification datasets.

1 Introduction

Text classification is a classic problem in Natural Language Processing (NLP). The task is to assign predefined categories to a given text sequence. An important intermediate step is the text representation. Previous work uses various neural models to learn text representation, including convolution models (Kalchbrenner et al., 2014; Zhang et al., 2015; Conneau et al., 2016; Johnson and Zhang, 2017; Zhang et al., 2017; Shen et al., 2018), recurrent models (Liu et al., 2016; Yogatama et al., 2017; Seo et al., 2017), and attention mechanisms (Yang et al., 2016; Lin et al., 2017).

Alternatively, substantial work has shown that pre-trained models on large corpus are beneficial for text classification and other NLP tasks, which can avoid training a new model from scratch. One kind of pre-trained models is the word embeddings, such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), or the contextualized word embeddings, such as CoVe (McCann et al., 2017) and ELMo (Peters et al., 2018). These word embeddings are often used as additional features for the main task. Another kind of pre-training models is sentence-level. Howard and Ruder (2018) propose ULMFiT, a fine-tuning method for pre-trained language model that achieves state-of-the-art results on six widely studied text classification datasets. More recently, pre-trained language models have shown to be useful in learning common language representations by utilizing a large amount of unlabeled data: e.g., OpenAI GPT (Radford et al., 2018) and BERT (Devlin et al., 2018). BERT is based on a multi-layer bidirectional Transformer (Vaswani et al., 2017) and is trained on plain text for masked word prediction and next sentence prediction tasks.

Although BERT has achieved amazing results in many natural language understanding (NLU) tasks, its potential has yet to be fully explored. There is little research to enhance BERT to improve the performance on target tasks further.

In this paper, we investigate how to maximize the utilization of BERT for the text classification task. We explore several ways of fine-tuning BERT to enhance its performance on text classification task. We design exhaustive experiments to make a detailed analysis of BERT.

The contributions of our paper are as follows:

- We propose a general solution to fine-tune the pre-trained BERT model, which includes three steps: (1) further pre-train BERT on within-task training data or in-domain data; (2) optional fine-tuning BERT with multi-task learning if several related tasks are available; (3) fine-tune BERT for the target task.
- We also investigate the fine-tuning methods for BERT on target task, including preprocess of long text, layer selection, layer-wise learning rate, catastrophic forgetting, and low-shot learning problems.

[*] Corresponding author

- We achieve the new state-of-the-art results on seven widely-studied English text classification datasets and one Chinese news classification dataset.

2 Related Work

Borrowing the learned knowledge from the other tasks has a rising interest in the field of NLP. We briefly review two related approaches: language model pre-training and multi-task Learning.

2.1 Language Model Pre-training

Pre-trained word embeddings (Mikolov et al., 2013; Pennington et al., 2014), as an important component of modern NLP systems can offer significant improvements over embeddings learned from scratch. The generalization of word embeddings, such as sentence embeddings (Kiros et al., 2015; Logeswaran and Lee, 2018) or paragraph embeddings (Le and Mikolov, 2014), are also used as features in downstream models.

Peters et al. (2018) concatenate embeddings derived from language model as additional features for the main task and advance the state-of-the-art for several major NLP benchmarks. In addition to pre-training with unsupervised data, transfer learning with a large amount of supervised data can also achieve good performance, such as natural language inference (Conneau et al., 2017) and machine translation (McCann et al., 2017).

More recently, the method of pre-training language models on a large network with a large amount of unlabeled data and fine-tuning in downstream tasks has made a breakthrough in several natural language understanding tasks, such as OpenAI GPT (Radford et al., 2018) and BERT (Devlin et al., 2018). Dai and Le (2015) use language model fine-tuning but overfit with 10k labeled examples while Howard and Ruder (2018) propose ULMFiT and achieve state-of-the-art results in the text classification task. BERT is pre-trained on *Masked Language Model Task* and *Next Sentence Prediction Task* via a large cross-domain corpus. Unlike previous bidirectional language models (biLM) limited to a combination of two unidirectional language models (i.e., left-to-right and right-to-left), BERT uses a Masked Language Model to predict words which are randomly masked or replaced. BERT is the first fine-tuning based representation model that achieves state-of-the-art results for a range of NLP tasks, demonstrating the enormous potential of the fine-tuning method. In this paper, we have further explored the BERT fine-tuning method for text classification.

2.2 Multi-task learning

Multi-task learning (Caruana, 1993; Collobert and Weston, 2008) is another relevant direction. Rei (2017) and Liu et al. (2018) use this method to train the language model and the main task model jointly. Liu et al. (2019) extend the MT-DNN model originally proposed in Liu et al. (2015) by incorporating BERT as its shared text encoding layers. MTL requires training tasks from scratch every time, which makes it inefficient and it usually requires careful weighing of task-specific objective functions (Chen et al., 2017). However, we can use multi-task BERT fine-tuning to avoid this problem by making full use of the shared pre-trained model.

3 BERT for Text Classification

BERT-base model contains an encoder with 12 Transformer blocks, 12 self-attention heads, and the hidden size of 768. BERT takes an input of a sequence of no more than 512 tokens and outputs the representation of the sequence. The sequence has one or two segments that the first token of the sequence is always `[CLS]` which contains the special classification embedding and another special token `[SEP]` is used for separating segments.

For text classification tasks, BERT takes the final hidden state $\mathbf{h}$ of the first token `[CLS]` as the representation of the whole sequence. A simple softmax classifier is added to the top of BERT to predict the probability of label c:

$$p(c|\mathbf{h}) = \text{softmax}(W\mathbf{h}), \qquad (1)$$

where W is the task-specific parameter matrix. We fine-tune all the parameters from BERT as well as W jointly by maximizing the log-probability of the correct label.

4 Methodology

When we adapt BERT to NLP tasks in a target domain, a proper fine-tuning strategy is desired. In this paper, we look for the proper fine-tuning methods in the following three ways.

1) **Fine-Tuning Strategies**: When we fine-tune BERT for a target task, there are many ways to

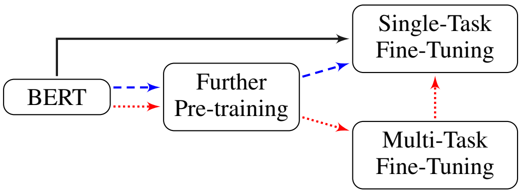


Figure 1: Three general ways for fine-tuning BERT, shown with different colors.

utilize BERT. For example, the different layers of BERT capture different levels of semantic and syntactic information, which layer is better for a target task? How we choose a better optimization algorithm and learning rate?

2) **Further Pre-training**: BERT is trained in the general domain, which has a different data distribution from the target domain. A natural idea is to further pre-train BERT with target domain data.

3) **Multi-Task Fine-Tuning**: Without pre-trained LM models, multi-task learning has shown its effectiveness of exploiting the shared knowledge among the multiple tasks. When there are several available tasks in a target domain, an interesting question is whether it still bring benefits to fine-tune BERT on all the tasks simultaneously.

Our general methodology of fine-tuning BERT is shown in Figure 1.

4.1 Fine-Tuning Strategies

Different layers of a neural network can capture different levels of syntactic and semantic information (Yosinski et al., 2014; Howard and Ruder, 2018).

To adapt BERT to a target task, we need to consider several factors: 1) The first factor is the preprocessing of long text since the maximum sequence length of BERT is 512. 2) The second factor is layer selection. The official BERT-base model consists of an embedding layer, a 12-layer encoder, and a pooling layer. We need to select the most effective layer for the text classification task. 3) The third factor is the overfitting problem. A better optimizer with an appropriate learning rate is desired.

Intuitively, the lower layer of the BERT model may contain more general information. We can fine-tune them with different learning rates.

Following Howard and Ruder (2018), we split the parameters θ into $\{\theta^1, \cdots, \theta^L\}$ where θ^l contains the parameters of the l-th layer of BERT. Then the parameters are updated as follows:

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta), \tag{2}$$

where η^l represents the learning rate of the l-th layer.

We set the base learning rate to η^L and use $\eta^{k-1} = \xi \cdot \eta^k$, where ξ is a decay factor and less than or equal to 1. When $\xi < 1$, the lower layer has a lower learning rate than the higher layer. When $\xi = 1$, all layers have the same learning rate, which is equivalent to the regular stochastic gradient descent (SGD). We will investigate these factors in Sec. 5.3.

4.2 Further Pre-training

The BERT model is pre-trained in the general-domain corpus. For a text classification task in a specific domain, such as movie reviews, its data distribution may be different from BERT. Therefore, we can further pre-train BERT with masked language model and next sentence prediction tasks on the domain-specific data. Three further pre-training approaches are performed:

1) Within-task pre-training, in which BERT is further pre-trained on the training data of a target task.

2) In-domain pre-training, in which the pre-training data is obtained from the same domain of a target task. For example, there are several different sentiment classification tasks, which have a similar data distribution. We can further pre-train BERT on the combined training data from these tasks.

3) Cross-domain pre-training, in which the pre-training data is obtained from both the same and other different domains to a target task.

We will investigate these different approaches to further pre-training in Sec. 5.4.

4.3 Multi-Task Fine-Tuning

Multi-task Learning is also an effective approach to share the knowledge obtained from several related supervised tasks. Similar to Liu et al. (2019), we also use fine-tune BERT in multi-task learning framework for text classification.

All the tasks share the BERT layers and the embedding layer. The only layer that does not share is the final classification layer, which means that each task has a private classifier layer. The experimental analysis is in Sec. 5.5.

Dataset	Classes	Type	Average lengths	Max lengths	Exceeding ratio	Train samples	Test samples
IMDb	2	Sentiment	292	3,045	12.69%	25,000	25,000
Yelp P.	2	Sentiment	177	2,066	4.60%	560,000	38,000
Yelp F.	5	Sentiment	179	2,342	4.60%	650,000	50,000
TREC	6	Question	11	39	0.00%	5,452	500
Yahoo! Answers	10	Question	131	4,018	2.65%	1,400,000	60,000
AG's News	4	Topic	44	221	0.00%	120,000	7,600
DBPedia	14	Topic	67	3,841	0.00%	560,000	70,000
Sogou News	6	Topic	737	47,988	46.23%	54,000	6,000

Table 1: Statistics of eight text classification datasets. The exceeding ratio means the percentage of the number of samples with a length exceeding 512.

5 Experiments

We investigate the different fine-tuning methods for seven English and one Chinese text classification tasks. We use the base BERT models: the uncased BERT-base model[1] and the Chinese BERT-base model[2] respectively.

5.1 Datasets

We evaluate our approach on eight widely-studied datasets. These datasets have varying numbers of documents and varying document lengths, covering three common text classification tasks: sentiment analysis, question classification, and topic classification. We show the statistics for each dataset in Table 1.

Sentiment analysis For sentiment analysis, we use the binary film review IMDb dataset (Maas et al., 2011) and the binary and five-class version of the Yelp review dataset built by Zhang et al. (2015).

Question classification For question classification, we evaluate our method on the six-class version of the TREC dataset (Voorhees and Tice, 1999) and Yahoo! Answers dataset created by Zhang et al. (2015). TREC dataset is dataset for question classification consisting of open-domain, fact-based questions divided into broad semantic categories. Compared to other document-level datasets, TREC dataset is sentence-level, and there are fewer training examples for it. Yahoo! Answers dataset is a big dataset with 1,400k train samples.

Topic classification For topic classification, we use large-scale AG's News and DBPedia created by Zhang et al. (2015). To test the effectiveness of BERT for Chinese text, we create the Chinese training and test datasets for Sogou news corpus. Unlike Zhang et al. (2015), we use the Chinese character directly rather than Pinyin. The dataset is a combination of the SogouCA and SogouCS news corpora (Wang et al., 2008). We determine the category of the news based on the URL, such as "sports" corresponding to "http://sports.sohu.com". We choose 6 categories – "sports", "house", "business", "entertainment", "women" and "technology". The number of training samples selected for each class is 9,000 and testing 1,000.

Data preprocessing Following Devlin et al. (2018), we use WordPiece embeddings (Wu et al., 2016) with a 30,000 token vocabulary and denote split word pieces with ##. So the statistics of the length of the documents in the datasets are based on the word pieces. For further pre-training with BERT, we use spaCy[3] to perform sentence segmentation in English datasets and we use "。","？" and "！" as separators when dealing with the Chinese Sogou News dataset.

5.2 Hyperparameters

We use the BERT-base model (Devlin et al., 2018) with a hidden size of 768, 12 Transformer blocks (Vaswani et al., 2017) and 12 self-attention heads. We further pre-train with BERT on 1 TITAN Xp GPU, with a batch size of 32, max squence length of 128, learning rate of 5e-5, train steps of 100,000 and warm-up steps of 10,000.

We fine-tune the BERT model on 4 TITAN Xp GPUs and set the batch size to 24 to ensure that the GPU memory is fully utilized. The dropout probability is always kept at 0.1. We use Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We use *slanted triangular learning rates* (Howard and Ruder, 2018), the

[1] https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip

[2] https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip

[3] https://spacy.io/

base learning rate is 2e-5, and the warm-up proportion is 0.1. We empirically set the max number of the epoch to 4 and save the best model on the validation set for testing.

5.3 Exp-I: Investigating Different Fine-Tuning Strategies

In this subsection, we use the IMDb dataset to investigate the different fine-tuning strategies. The official pre-trained model is set as the initial encoder[4].

5.3.1 Dealing with long texts

The maximum sequence length of BERT is 512. The first problem of applying BERT to text classification is how processing the text with a length larger than 512. We try the following ways for dealing with long articles.

Truncation methods Usually, the key information of an article is at the beginning and end. We use three different methods of truncate text to perform BERT fine-tuning.

1. **head-only**: keep the first 510 tokens[5];
2. **tail-only**: keep the last 510 tokens;
3. **head+tail**: empirically select the first 128 and the last 382 tokens.

Hierarchical methods The input text is firstly divided into $k = L/510$ fractions, which is fed into BERT to obtain the representation of the k text fractions. The representation of each fraction is the hidden state of the `[CLS]` tokens of the last layer. Then we use mean pooling, max pooling and self-attention to combine the representations of all the fractions.

Table 2 shows the effectiveness of the above methods. The truncation method of **head+tail** achieves the best performance on IMDb and Sogou datasets. Therefore, we use this method to deal with the long text in the following experiments.

Method	IMDb	Sogou
head-only	5.63	2.58
tail-only	5.44	3.17
head+tail	**5.42**	**2.43**
hier. mean	5.89	2.83
hier. max	5.71	2.47
hier. self-attention	5.49	2.65

Table 2: Test error rates (%) on IMDb and Chinese Sogou News datasets.

5.3.2 Features from Different layers

Each layer of BERT captures the different features of the input text. We investigate the effectiveness of features from different layers. We then fine-tune the model and record the performance on test error rates.

Table 3 shows the performance of fine-tuning BERT with different layers. The feature from the last layer of BERT gives the best performance. Therefore, we use this setting for the following experiments.

Layer	Test error rates(%)
Layer-0	11.07
Layer-1	9.81
Layer-2	9.29
Layer-3	8.66
Layer-4	7.83
Layer-5	6.83
Layer-6	6.83
Layer-7	6.41
Layer-8	6.04
Layer-9	5.70
Layer-10	5.46
Layer-11	**5.42**
First 4 Layers + concat	8.69
First 4 Layers + mean	9.09
First 4 Layers + max	8.76
Last 4 Layers + concat	5.43
Last 4 Layers + mean	5.44
Last 4 Layers + max	**5.42**
All 12 Layers + concat	5.44

Table 3: Fine-tuning BERT with different layers on IMDb dataset.

5.3.3 Catastrophic Forgetting

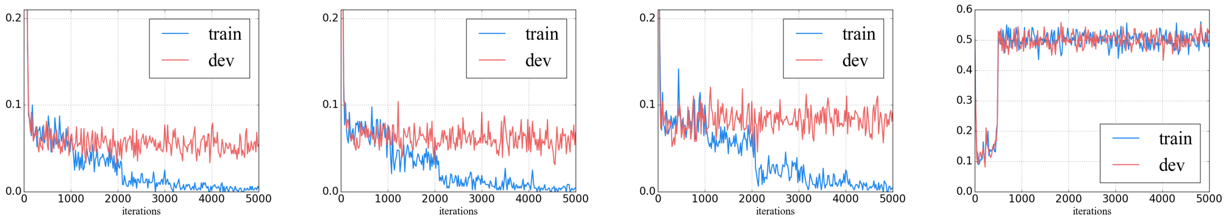
Catastrophic forgetting (McCloskey and Cohen, 1989) is usually a common problem in transfer learning, which means the pre-trained knowledge is erased during learning of new knowledge. Therefore, we also investigate whether BERT suffers from the catastrophic forgetting problem.

We fine-tune BERT with different learning rates, and the learning curves of error rates on IMDb are shown in Figure 2.

We find that a lower learning rate, such as 2e-5, is necessary to make BERT overcome the catastrophic forgetting problem. With an aggressive learn rate of 4e-4, the training set fails to converge.

[4] https://github.com/google-research/bert

[5] 512 to subtract the `[CLS]` and `[SEP]` tokens.



(a) lr=2e-5 (b) lr=5e-5 (c) lr=1e-4 (d) lr=4e-4

Figure 2: Catastrophic Forgetting

5.3.4 Layer-wise Decreasing Layer Rate

Table 4 show the performance of different base learning rate and decay factors (see Eq. (2)) on IMDb dataset. We find that assign a lower learning rate to the lower layer is effective to fine-tuning BERT, and an appropriate setting is ξ=0.95 and lr=2.0e-5.

Learning rate	Decay factor ξ	Test error rates(%)
2.5e-5	1.00	5.52
2.5e-5	0.95	5.46
2.5e-5	0.90	**5.44**
2.5e-5	0.85	5.58
2.0e-5	1.00	5.42
2.0e-5	0.95	**5.40**
2.0e-5	0.90	5.52
2.0e-5	0.85	5.65

Table 4: Decreasing layer-wise layer rate.

5.4 Exp-II: Investigating the Further Pretraining

Besides, fine-tune BERT with supervised learning, we can further pre-train BERT on the training data by unsupervised masked language model and next sentence prediction tasks. In this section, we investigate the effectiveness of further pre-training. In the following experiments, we use the best strategies in Exp-I during the fine-tuning phase.

5.4.1 Within-Task Further Pre-Training

Therefore, we first investigate the effectiveness of within-task further pre-training. We take further pre-trained models with different steps and then fine-tune them with text classification task.

As shown in Figure 3, the further pre-training is useful to improve the performance of BERT for a target task, which achieves the best performance after 100K training steps.

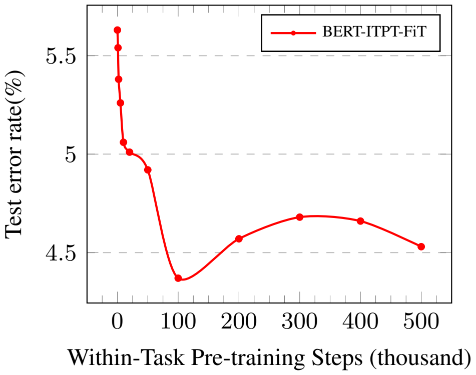


Figure 3: Benefit of different further pre-training steps on IMDb datasets. BERT-ITPT-FiT means “BERT + withIn-Task Pre-Training + Fine-Tuning”.

5.4.2 In-Domain and Cross-Domain Further Pre-Training

Besides the training data of a target task, we can further pre-train BERT on the data from the same domain. In this subsection, we investigate whether further pre-training BERT with in-domain and cross-domain data can continue to improve the performance of BERT.

We partition the seven English datasets into three domains: topic, sentiment, and question. The partition way is not strictly correct. Therefore we also conduct extensive experiments for cross-task pre-training, in which each task is regarded as a different domain.

The results is shown in Table 5. We find that almost all further pre-training models perform better on all seven datasets than the original BERT-base model (row ‘w/o pretrain’ in Table 5). Generally, in-domain pretraining can bring better performance than within-task pretraining. On the small sentence-level TREC dataset, within-task pre-training do harm to the performance while in-domain pre-training which utilizes Yah. A. corpus can achieve better results on TREC.

Cross-domain pre-training (row ‘all’ in Table 5)

Domain	sentiment			question		topic	
Dataset	IMDb	Yelp P.	Yelp F.	TREC	Yah. A.	AG's News	DBPedia
IMDb	**4.37**	2.18	29.60	2.60	22.39	5.24	0.68
Yelp P.	5.24	1.92	29.37	2.00	22.38	5.14	**0.65**
Yelp F.	5.18	1.94	29.42	2.40	22.33	5.43	**0.65**
all sentiment	4.88	**1.87**	29.25	3.00	22.35	5.34	0.67
TREC	5.65	2.09	29.35	3.20	22.17	5.12	0.66
Yah. A.	5.52	2.08	29.31	**1.80**	22.38	5.16	0.67
all question	5.68	2.14	29.52	2.20	**21.86**	5.21	0.68
AG's News	5.97	2.15	29.38	2.00	22.32	**4.80**	0.68
DBPedia	5.80	2.13	29.47	2.60	22.30	5.13	0.68
all topic	5.85	2.20	29.68	2.60	22.28	4.88	**0.65**
all	5.18	1.97	**29.20**	2.80	21.94	5.08	0.67
w/o pretrain	5.40	2.28	30.06	2.80	22.42	5.25	0.71

Table 5: Performance of in-domain and cross-domain further pre-training on seven datasets. Each was further pre-trained for 100k steps. The first column indicates the different further pre-training dataset. "all sentiment" means the dataset consists of all the training datasets in sentiment domain. "all" means the dataset consists of all the seven training datasets. Note that some of the data in Yelp P. and Yelp F. are overlapping, e.g., part of the data in the test set of Yelp P. will appear in the training set of Yelp F., so we remove this part of data from the training sets during further pre-training.

does not bring an obvious benefit in general. It is reasonable since BERT is already trained on a general domain.

We also find that IMDb and Yelp do not help each other in sentiment domain. The reason may be that IMDb and Yelp are two sentiment tasks of movie and food. The data distributions have a significant difference.

5.4.3 Comparisons to Previous Models

We compare our model with the following a variety of different methods: CNN-based methods such as Char-level CNN (Zhang et al., 2015), VDCNN (Conneau et al., 2016) and DPCNN (Johnson and Zhang, 2017); RNN-based models such as D-LSTM (Yogatama et al., 2017), Skim-LSTM (Seo et al., 2017) and hierarchical attention networks (Yang et al., 2016); feature-based transfer learning methods such as rigion embedding (Qiao et al., 2018) and CoVe (McCann et al., 2017); and the language model fine-tuning method (ULMFiT) (Howard and Ruder, 2018), which is the current state-of-the-art for text classification.

We implement BERT-Feat through using the feature from BERT model as the input embedding of the biLSTM with self-attention (Lin et al., 2017). The result of BERT-IDPT-FiT corresponds to the row of 'all sentiment', 'all question', and 'all topic' in Table 5, and the result of BERT-CDPT-FiT corresponds to the row of 'all' in it.

As is shown in Table 6, BERT-Feat performs better than all other baselines except for ULMFiT. In addition to being slightly worse than BERT-Feat on DBpedia dataset, BERT-FiT outperforms BERT-Feat on the other seven datasets. Moreover, all of the three further pre-training models are better than BERT-FiT model. Using BERT-Feat as a reference, we calculate the average percentage increase of other BERT-FiT models on each dataset. BERT-IDPT-FiT performs best, with an average error rate reduce by 18.57%.

5.5 Exp-III: Multi-task Fine-Tuning

When there are several datasets for the text classification task, to take full advantage of these available data, we further consider a fine-tuning step with multi-task learning. We use four English text classification datasets (IMDb, Yelp P., AG, and DBP). The dataset Yelp F. is excluded since there is overlap between the test set of Yelp F. and the training set of Yelp P., and two datasets of question domain are also excluded.

We experiment with the official uncased BERT-base weights and the weights further pre-trained on all seven English classification datasets respectively. In order to achieve better classification results for each subtask, after fine-tuning together, we fine-tune the extra steps on the respective datasets with a lower learning rate.

Table 7 shows that for multi-task fine-tuning based on BERT, the effect is improved. However, multi-task fine-tuning does not seem to be help-

Model	IMDb	Yelp P.	Yelp F.	TREC	Yah. A.	AG	DBP	Sogou	Avg. Δ
Char-level CNN(Zhang et al., 2015)	/	4.88	37.95	/	28.80	9.51	1.55	3.80*	/
VDCNN (Conneau et al., 2016)	/	4.28	35.28	/	26.57	8.67	1.29	3.28	/
DPCNN (Johnson and Zhang, 2017)	/	2.64	30.58	/	23.90	6.87	0.88	3.48*	/
D-LSTM (Yogatama et al., 2017)	/	7.40	40.40	/	26.30	7.90	1.30	5.10	/
Standard LSTM (Seo et al., 2017)	8.90	/	/	/	/	6.50	/	/	/
Skim-LSTM (Seo et al., 2017)	8.80	/	/	/	/	6.40	/	/	/
HAN (Yang et al., 2016)	/	/	/	/	24.20	/	/	/	/
Region Emb. (Qiao et al., 2018)	/	3.60	35.10	/	26.30	7.20	1.10	2.40	/
CoVe (McCann et al., 2017)	8.20	/	/	4.20	/	/	/	/	/
ULMFiT (Howard and Ruder, 2018)	4.60	2.16	29.98	3.60	/	5.01	0.80	/	/
BERT-Feat	6.79	2.39	30.47	4.20	22.72	5.92	0.70	2.50	-
BERT-FiT	5.40	2.28	30.06	2.80	22.42	5.25	0.71	2.43	9.22%
BERT-ITPT-FiT	**4.37**	1.92	29.42	3.20	22.38	**4.80**	0.68	**1.93**	16.07%
BERT-IDPT-FiT	4.88	**1.87**	29.25	**2.20**	**21.86**	4.88	**0.65**	/	**18.57%**
BERT-CDPT-FiT	5.18	1.97	**29.20**	2.80	21.94	5.08	0.67	/	14.38%

Table 6: Test error rates (%) on eight text classification datasets. The results without * of previous models are the results reported on their papers. / means not reported. * means the results are from our implementation since the Sogou dataset is different from theirs. BERT-Feat means "BERT as features". BERT-FiT means "BERT + Fine-Tuning". BERT-ITPT-FiT means "BERT + withIn-Task Pre-Training + Fine-Tuning". BERT-IDPT-FiT means "BERT + In-Domain Pre-Training + Fine-Tuning". BERT-CDPT-FiT means "BERT + Cross-Domain Pre-Training + Fine-Tuning".

Method	IMDb	Yelp P.	AG	DBP
BERT-FiT	5.40	2.28	5.25	0.71
BERT-MFiT-FiT	5.36	2.19	5.20	0.68
BERT-CDPT-FiT	5.18	**1.97**	**5.08**	**0.67**
BERT-CDPT-MFiT-FiT	**4.96**	2.06	5.13	**0.67**

Table 7: Test error rates (%) with multi-task fine-tuning.

ful to BERT-CDPT in Yelp P. and AG. Multi-task fine-tuning and cross-domain pre-training may be alternative methods since the BERT-CDPT model already contains rich domain-specific information, and multi-task learning may not be necessary to improve generalization on related text classification sub-tasks.

5.6 Exp-IV: Few-Shot Learning

One of the benefits of the pre-trained model is being able to train a model for downstream tasks within small training data. We evaluate BERT-FiT and BERT-ITPT-FiT on different numbers of training examples. We select a subset of IMDb training data and feed them into BERT-FiT and BERT-ITPT-FiT. We show the result in Figure 4.

This experiment result demonstrates that BERT brings a significant improvement to small size data. Further pre-trained BERT can further boost its performance, which improves the performance from 17.26% to 9.23% in error rates with only 0.4% training data.

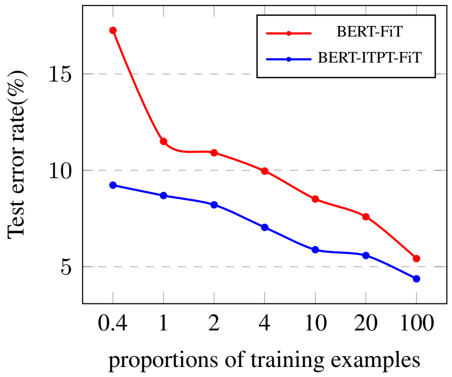


Figure 4: Test error rates(%) on IMDb dataset with different proportions of training examples.

5.7 Exp-V: Further Pre-Training on BERT Large

In this subsection, we investigate whether the BERT$_{\text{LARGE}}$ model has similar findings to BERT$_{\text{BASE}}$. We further pre-train Google's pre-trained BERT$_{\text{LARGE}}$ model[6] on 1 Tesla-V100-PCIE 32G GPU with a batch size of 24, the max sequence length of 128 and 120K training steps. For target task classifier BERT fine-tuning, we set the batch size to 24 and fine-tune BERT$_{\text{LARGE}}$ on 4 Tesla-V100-PCIE 32G GPUs with the max sequence length of 512.

[6] https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-24_H-1024_A-16.zip

As shown in Table 8, ULMFiT performs better on almost all of the tasks compared to BERT$_{\text{BASE}}$ but not BERT$_{\text{LARGE}}$. This changes however with the task-specific further pre-training where even BERT$_{\text{BASE}}$ outperforms ULMFiT on all tasks. BERT$_{\text{LARGE}}$ fine-tuning with task-specific further pre-training achieves state-of-the-art results.

Model	IMDb	Yelp P.	Yelp F.	AG	DBP
ULMFiT	4.60	2.16	29.98	5.01	0.80
BERT$_{\text{BASE}}$	5.40	2.28	30.06	5.25	0.71
+ ITPT	4.37	1.92	29.42	4.80	0.68
BERT$_{\text{LARGE}}$	4.86	2.04	29.25	4.86	0.62
+ ITPT	**4.21**	**1.81**	**28.62**	**4.66**	**0.61**

Table 8: Test error rates (%) on five text classification datasets.

6 Conclusion

In this paper, we conduct extensive experiments to investigate the different approaches to fine-tuning BERT for the text classification task. There are some experimental findings: 1) The top layer of BERT is more useful for text classification; 2) With an appropriate layer-wise decreasing learning rate, BERT can overcome the catastrophic forgetting problem; 3) Within-task and in-domain further pre-training can significantly boost its performance; 4) A preceding multi-task fine-tuning is also helpful to the single-task fine-tuning, but its benefit is smaller than further pre-training; 5) BERT can improve the task with small-size data.

With the above findings, we achieve state-of-the-art performances on eight widely studied text classification datasets. In the future, we will probe more insight of BERT on how it works.

References

Rich Caruana. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on Machine Learning*.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2017. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv preprint arXiv:1711.02257*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Alexis Conneau, Holger Schwenk, Loıc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*, 2.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

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*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 562–570.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Chao Qiao, Bo Huang, Guocheng Niu, Daren Li, Daxiang Dong, Wei He, Dianhai Yu, and Hua Wu. 2018. Anew method of region embedding for text classification. In *International Conference on Learning Representations*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. *arXiv preprint arXiv:1704.07156*.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Neural speed reading via skimrnn. *arXiv preprint arXiv:1711.02085*.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2018. Deconvolutional latent-variable model for text sequence matching. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ellen M Voorhees and Dawn M Tice. 1999. The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82. Citeseer.

Canhui Wang, Min Zhang, Shaoping Ma, and Liyun Ru. 2008. Automatic online news issue construction in web environment. In *Proceedings of the 17th international conference on World Wide Web*, pages 457–466. ACM.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*, pages 4169–4179.