

Received October 31, 2019, accepted November 13, 2019, date of publication November 18, 2019,
date of current version December 23, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2953990

Improving BERT-Based Text Classification With Auxiliary Sentence and Domain Knowledge

SHANSHAN YU^{ID}^{1,2}, JINDIAN SU^{ID}³, AND DA LUO^{ID}³

¹College of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou 510640, China

²Guangdong Province Precise Medicine and Big Data Engineering Technology Research Center for Traditional Chinese Medicine, Guangdong Pharmaceutical University, Guangzhou 510640, China

³College of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China

Corresponding authors: Shanshan Yu (susyu@139.com) and Jindian Su (sujd@scut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61936003, in part by the Research and Development Program in Key Areas of Guangdong Province under Grant 2018B010109004, in part by the Applied Scientific and Technology Special Project of Department of Science and Technology of Guangdong Province under Grant 20168010124010, in part by the Natural Science Foundation of Guangdong Province under Grant 2015A030310318, in part by the Medical Scientific Research Foundation of Guangdong Province under Grant A2015065, and in part by the Innovation Project of Guangdong Pharmaceutical University under Grant 52159433.

ABSTRACT General language model BERT pre-trained on cross-domain text corpus, BookCorpus and Wikipedia, achieves excellent performance on a couple of natural language processing tasks through the way of fine-tuning in the downstream tasks. But it still lacks of task-specific knowledge and domain-related knowledge for further improving the performance of BERT model and more detailed fine-tuning strategy analyses are necessary. To address these problem, a BERT-based text classification model BERT4TC is proposed via constructing auxiliary sentence to turn the classification task into a binary sentence-pair one, aiming to address the limited training data problem and task-awareness problem. The architecture and implementation details of BERT4TC are also presented, as well as a post-training approach for addressing the domain challenge of BERT. Finally, extensive experiments are conducted on seven public widely-studied datasets for analyzing the fine-tuning strategies from the perspectives of learning rate, sequence length and hidden state vector selection. After that, BERT4TC models with different auxiliary sentences and post-training objectives are compared and analyzed in depth. The experiment results show that BERT4TC with suitable auxiliary sentence significantly outperforms both typical feature-based methods and fine-tuning methods, and achieves new state-of-the-art performance on multi-class classification datasets. For binary sentiment classification datasets, our BERT4TC post-trained with suitable domain-related corpus also achieves better results compared with original BERT model.

INDEX TERMS Natural language processing, text classification, bidirectional encoder representations from transformer, neural networks, language model.

I. INTRODUCTION

Text classification has been a classic task and heated research hotspot in the field of natural language processing (NLP), aiming to assign pre-defined categories to a given text sequence. Previous works in the past few years tried to use various neural network models to learn text presentations for classification, i.e. convolutional neural network (CNN) models [1], [2], recurrent neural network (RNN) models [3], attentional models [3]–[5] and adversarial

The associate editor coordinating the review of this manuscript and approving it for publication was Seyedali Mirjalili^{ID}.

models [6], which significantly outperform many canonical statistics-based methods. In these work, word vectors pre-trained over large scale of unsupervised text corpus are usually adopted as the features of sequences, which are usually trained via word2vec tool [7], [8] or Glove [9] algorithm based on the assumption that the words with similar meanings tend to appear in similar contexts. Consequently, the semantic vector of each word is learned according to its concurrence information with other words within a certain window size, which makes it a kind of context-independent representations that tends to encounter some following challenges:

(1) Polysemous challenge. In traditional word vector representation, each word is assigned to a fixed vector no matter whatever contexts it appears. However, most of the time, a word might have completely different meanings with respect to different contexts. For example, given two different sentences "I like eating apple" and "I like using apple phone", the first "apple" denotes a kind of fruits, while the second "apple" refers to a famous company. For text classification task, the precise meaning of a word might sometimes play a crucial role for the final prediction, especially for those indicative words.

(2) Task-specific structure dependent challenge. For most of previous neural network-based models, the classification performance relies heavily on task-specific neural structures, which usually require elaborate designs and careful training. At the same time, some complex syntactic features or advanced mechanisms often need to be adopted, i.e. dependency tree or attention.

To address the polysemous challenge, some scholars propose the notion of contextualized word vectors, i.e. CoVe [10] and ELMo (Embeddings from Language Models) [11], to learn multiple vectors for a word according to its different appearing contexts. CoVe uses a deep LSTM (Long Short-term Memory) encoder from an attentional sequence-to-sequence model trained for machine translation (MT) to contextualize word vectors, and demonstrates that adding these context vectors can improve performance over using only unsupervised word and character vectors on a wide variety of common NLP tasks. ELMo uses vectors derived from a bidirectional LSTM that is trained with a coupled language model (LM) objective on a large text corpus and integrates these contextual word vectors with existing task-specific supervised neural architectures. Both CoVe and ELMo successfully generalize traditional word vectors to contain context-sensitive features, but these learned representations are still typically used as features in a downstream model [12].

In the past two years, in order to avoid heavily-engineered task-specific structures and greatly decrease the parameters to be learned from scratch, some scholars contributed along another direction by pre-training general language model on large-scale unsupervised text corpus and fine-tuning it for the downstream tasks. For example, Howard and Ruder propose an effective transfer learning method ULM-FiT (Universal Language Model Fine-tuning) in [13] for learning common language representations on general domain corpus, and optimizes it in the downstream task with the proposed techniques of discriminative fine-tuning and slanted triangular learning rates (STLR). The experimental results show that ULM-FiT significantly outperforms the state-of-the-art on six widely-studied text classification tasks. The scholars from OpenAI group propose the Generative Pre-trained Transformer (OpenAI GPT) in [14] by using a left-to-right multi-layer Transformer [15] architecture to learn the general language presentations from large-scale texts. Based on the work of OpenAI GPT, the scholars from Google company further

propose a new language representation model BERT (Bidirectional Encoder Representations from Transformers) in [12], aiming to address the unidirectional constraints in GPT and extend the model to multi-layer bidirectional Transformer. They also propose two new pre-training objectives, "masked language model" (MLM) and "next sentence prediction" (NSP) for BERT.

Currently, although BERT has already achieved many amazing results in a couple of NLP tasks and obviously outperforms most of feature-based representation methods, e.g., word2vec, Glove, CoVe and ELMo, its potential has yet to be fully explored, which at least includes:

(1) More detailed analyses about fine-tuning strategies for pre-training BERT in text classification tasks. The authors in [12], [16] already present some typical fine-tuning strategies for BERT from different perspectives, e.g. text length, transformer layer selection and learning rate. But they only consider the situation of long-text dataset whose sentences contain more than 512 tokens. So, it is necessary to further discuss fine-tuning strategies for short-text datasets (less than 512 tokens). Moreover, There still lack of discussions about the selections of different hidden state vectors from BERT encoder.

(2) Task-awareness challenge when BERT meets the limited number of fine-tuning examples in the downstream task, which is insufficient to fine-tune BERT to ensure full task-awareness of the model. Since large scale of high-quality training data are usually unavailable because manual labelling of training data is very time-consuming and human intensive, how to make full use of the limited supervised training data for fine-tuning BERT in the downstream task is of great importance.

(3) Leveraging BERT along still leaves the domain challenges unresolved (as BERT is trained on the concatenation of BooksCorpus and English Wikipedia articles and has no understanding of text classification knowledge). Especially, When sufficient training data for fine-tuning are unavailable, external domain-related corpus might provide another effective way for alleviating the task-awareness challenge.

In this paper, we propose a BERT-based model for text classification BERT4TC by constructing auxiliary sentence and converting the task into a sentence-pair one, which aims to better incorporate task-specific knowledge into pre-training BERT and address the task-awareness challenge. Furthermore, we also propose a post-training approach to utilize domain-related knowledge for addressing the domain challenge. Extensive experiments on several widely-used text classification datasets are conducted to validate the effectiveness of our model. The rest of this paper is organized as follows: Section II briefly review some related work about text classification and pre-training language model. Section III presents the structure of BERT4TC model and its implementations, as well as the post-training approach. Section IV gives extensive experiments over seven widely-studied text classification datasets and in-depth analyses. The conclusions

and the directions for the future research are summarized in Section V.

II. RELATED WORK

Many works have been contributed to text classification task by using various canonical neural networks. Some typical works include CNN-based models (e.g., VDCNN [1] and DPCNN [2]), RNN-based models (e.g., SANN [3]) and various attention-based models (e.g., HAN [5] and DiSAN [3]). All these works used pre-trained word embeddings ([7]–[9]) as an important component of downstream models to offer significant improvements over embeddings learned from scratch. Although many state-of-the-art results have been achieved, the polysemous and task-specific structure dependent problems have brought many restrictions for further improvement of the models' performance. Even with the help of recently successful application and developments of contextualized word vectors, i.e. CoVe [10] and ELMo [11], the model structures still need to be elaborately designed and trained.

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 a couple of NLP tasks, such as natural language inference, text classification and textual entailment. Howard and Ruder [13] propose ULMFiT and achieves state-of-the-art results in the text classification task. Alec et al. [14] propose OpenAI GPT by using a left-to-right multi-layer Transformer architecture to learn the general language presentations from large-scale texts. To address the unidirectional constraints of OpenAI GPT and improve the power of the pre-trained representations, Jacob et al. [12] propose a new fine-tuning based approaches BERT by enabling the pre-trained deep bidirectional representations, which as a result can avoid heavily-engineered task-specific architectures and greatly decrease the parameters to be learned from scratch. Compared with other previous pre-trained representations, BERT adopts a fine-tuning approach that requires almost no specific architecture for each end task and has achieved great success in many NLP tasks.

Based on the innovated work of BERT, many related researches have been done from different aspects. For example, Sun et al. [16] investigate different fine-tuning methods of BERT on text classification tasks, including pre-process of long text, layer selection, layer-wise learning rate, catastrophic forgetting and low-shot learning problems. But they only consider the long-text datasets and neglect the situations of short-text datasets and hidden vector selection. Xu et al. in [17] proposed a novel post-training approach on BERT to enhance the performance of fine-tuning of BERT for review reading comprehension. They also applied the proposed post-training to some other review-based tasks such as aspect extraction and aspect sentiment classification. In this paper, we also propose a post-training approach for further training BERT with domain-related knowledge and consider the different conditions of training objectives.

Sun et al. [18] propose to construct an auxiliary sentence from the aspect and convert the aspect-based sentiment analysis to a sentence-pair classification task. Inspired by the work of [18], we extend it to text classification task by constructing auxiliary sentence to address the task-awareness challenges and conduct exhaustive experiments to illustrate its effectiveness for multi-label text classification tasks. While for binary classification tasks, we demonstrate that post-training BERT with suitable domain-related corpus would be helpful for addressing the domain challenge and improving the performance of the classifier.

III. BERT4TC: BERT-BASED MODEL FOR TEXT CLASSIFICATION

Given an input text sequence x , the classification model can be regarded as a function, i.e. $f(x) = y$, for measuring the conditional probability distributions over all possible labels in the pre-defined category set $y = \{y_1, \dots, y_c\}$.

A. MODEL STRUCTURE

In this section, we propose a BERT-based model for text classification, BERT4TC, via constructing auxiliary sentence and incorporating domain-related knowledge. See FIGURE 1 for the structure of BERT4TC.

BERT4TC consists of three parts as follows:

(1) Input layer. It aims to build an input sequence for the model via constructing auxiliary sentence and turn the task into a sentence-pair one. After that, the WordPiece [19] embeddings with a 30,000 token vocabulary are used for segmenting the input sequence and the split word pieces are denoted with ##. The position embeddings, word embeddings and segmentation embeddings for each token are then summed to yield the final input representations.

(2) BERT encoder. It consists of 12 Transformer blocks and 12 self-attention heads by taking an input of a sequence of no more than 512 tokens and outputting the representations of the sequence. The representations might be a specific hidden state vector or a time-step sequence of hidden state vectors.

(3) Output layer. It consists of a simple softmax classifier on the top of BERT encoder for calculating the conditional probability distributions over pre-defined categorical labels.

B. INPUT LAYER

Let x be a token input sequence consisting of k words, denoted as $x_{1:k} = x_1 x_2 \dots x_i \dots x_k$, where $x_i (1 \leq x_i \leq k)$ refers to the i^{th} word in the sequence. For BERT model, an input sequence can unambiguously represent both a simple text sequence or a pair of text sequences in one token sequence (i.e. [Question, Answer]), in which the first token is always [CLS] that contains the special classification embedding and another special token [SEP] is used for separating segments or denoting the end of the sequence.

Existing BERT-based researches on text classification task (e.g. [12], [16]) all build input sequence as a simple text sequence and take the final hidden state vector of [CLS] as the representations of the whole sequence. The model is

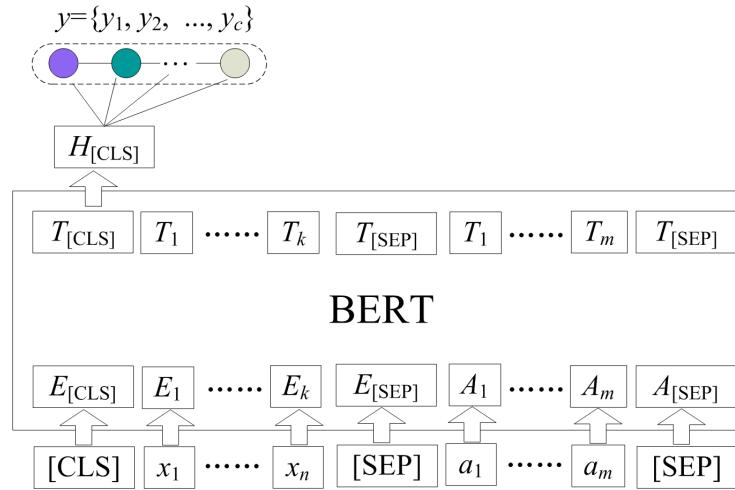


FIGURE 1. Structure of BERT4TC model.

then fine-tuned for learning downstream task-specific knowledge. Inspired by the work of [18], we propose a method of constructing auxiliary sentence and converting the original classification task into a sentence-pair one with a binary set of new categorical labels. As shown in the latter experiment, this is a necessary operation that can significantly improve the performance of the task. To facilitate the comparisons with other BERT-based classification models in [12], [16] whose input only consists of a simple text sequence, we propose several different ways of constructing auxiliary sentences in BERT4TC. Taking the input sequence $x_{1:k}$ as an example, the BERT4TC without/with auxiliary sentence can be defined as:

(1) **BERT4TC-S**. It denotes that the input is a simple text sequence, e.g., [CLS] $x_1 \dots x_i \dots x_k$ [SEP]. BERT4TC-S is just as the right canonical BERT model used in [12], [16].

(2) **BERT4TC-AQ**. The auxiliary sentence to generate is a pre-defined pseudo-sentence without categorical label information, e.g., [CLS] $x_1 \dots x_i \dots x_k$ [SEP] $a_1 \dots a_j \dots a_m$ [SEP], which can be pseudo-question sentence, e.g., "What is the result?". The target categorical labels for BERT4TC-AQ are consistent with the original task.

(3) **BERT4TC-AA**. The auxiliary sentence to generate from the labels is a sequence of tokens that only consist of a label from the original pre-defined set $y = \{y_1, \dots, y_j, \dots, y_c\}$, e.g., [CLS] $x_1 \dots x_i \dots x_k$ [SEP] y_j [SEP]. The task also needs to be converted into a binary classification problem for obtaining the probability distributions over the new categorical label set $\{0, 1\}$, in which the probability value of 1 is used as the matching score. We take the class of the sequence with the highest matching score for the predicted categorical label.

(4) **BERT4TC-AWA**. The auxiliary sentence to generate from the labels is a pseudo-sentence that consists of a categorical label and some other words, e.g., [CLS] $x_1 \dots x_i \dots x_k$ [SEP] $a_1 \dots a_m$ [SEP]. In the same

TABLE 1. Examples of input sequence constructions.

Model	Input Sequence	Label
BERT4TC-S	[CLS] I like this film. [SEP]	{negative, positive}
BERT4TC-AQ	[CLS] I like this film. [SEP] What is the result? [SEP]	{negative, positive}
BERT4TC-AA	[CLS] I like this film. [SEP] positive [SEP]	{0, 1}
	[CLS] I like this film. [SEP] negative [SEP]	{0, 1}
BERT4TC-AWA	[CLS] I like this film. [SEP] The result is positive. [SEP]	{0, 1}
	[CLS] I like this film. [SEP] The result is negative. [SEP]	{0, 1}

way as BERT4TC-AA, the original classification task should also be converted into $\{0, 1\}$.

For BERT4TC-AQ, we only construct an auxiliary sentence for each training sample with the original category information. While for BERT4TC-AA and BERT4TC-AWA, total c number of sentence-pairs would be built for each sample. We present an algorithm get_inputData for constructing auxiliary sentence and the corresponding sentence-pair as follows:

Given a sample, e.g. "I like this film." with a "positive" label from the set {"positive", "negative"}, different input sequences without or with auxiliary sentence are constructed as shown in TABLE 1 (we boldface the correct label).

We follow the same input sequence treatments of BERT by taking WordPiece embeddings [19] with a 30,000 token vocabulary to segment input sequence and denote the split word pieces with ##. Each segmented input sequence should be no more than 512 tokens. For BERT4TC-AQ, BERT4TC-AA and BERT4TC-AWA, if the sentence-pair $(x_{1:k}, a_{1:m})$ satisfies $k+m+3 > 512$, where 3 means one [CLS] token plus two [SEP] tokens, then only at most 509 tokens can

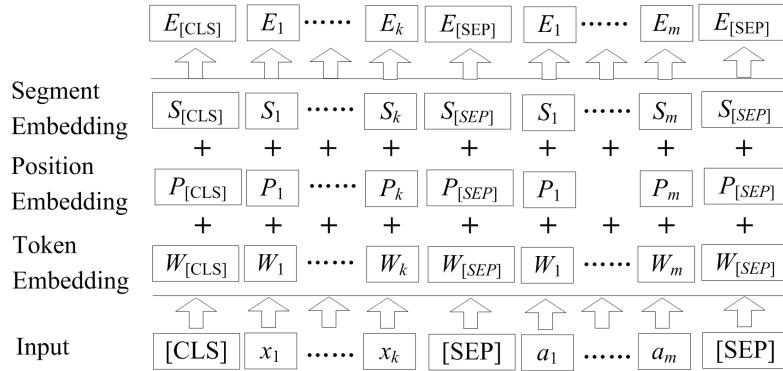


FIGURE 2. Construction of input sequence representations for BERT4TC.

Algorithm 1 get_inputData

Input: data, labels, dType and auxType
Output: examples #a BERT structure that encapsulates the sentence-pair and the correct label

```

1: function get_inputdata(data, labels, dType, auxType)
2:   assert dType in {"train", "test", "dev"}
3:   assert auxType in {"AQ", "AA", "AWA"}
4:   for (i, sent) in enumerate(data):
5:     label = labels[i]
6:     sents, newLabels = buildAuxSent(label, auxType)
7:     for j in range(len(sents)):
8:       guid = "%d-%d-%d" % (dType, i, j)
9:       examples.append(InputExample(guid=guid,
10:         text_a=sent, text_b=sents[j], label=newLabels[j]))
11: function buildAuxSent(label, auxType): #y={y1 ... yc}
12:   newSents = []
13:   newLabels = []
14:   if auxType == "AQ": # for BERT4TC-AQ
15:     newSents.append("What is the result?")
16:     newLabels.append(label)
17:   elif auxType in ("AA", "AWA"):
18:     for i in range(len(y)):
19:       auxSent = y[i] if auxType == "AA"
20:         else insert(y[i], auxSentMode)
21:       # auxSentMode is a temple, e.g. "The result is {}."
22:       # where {} will be replaced by y[i].
23:       newLabel = 1 if y[i] == label else 0
24:       newSents.append(auxSent)
25:       newLabels.append(newLabel)
26:   return newSents, newLabels

```

be kept. We choose to keep the auxiliary sentence and only shorten the original input text. The pre-treatment formula is shown in the follow way:

$$(x_{1:k}, a_{1:m}) = \begin{cases} [\text{CLS}] x_1 \dots x_k [\text{SEP}] a_1 \dots a_m [\text{SEP}] \\ \quad \text{if } k + m < 509 \\ [\text{CLS}] x_1 \dots x_{k+m-509} [\text{SEP}] a_1 \dots \\ \quad a_m [\text{SEP}] \quad \text{if } k + m \geq 509 \end{cases} \quad (1)$$

The input representation for each token E is obtained by summing its token embeddings W , segment embeddings S and position embeddings P . See FIGURE 2 for the visualization of this construction.

C. BERT ENCODER

The model architecture of BERT is a multi-layer bidirectional Transformer encoder based on the original implementations described in [15]. It consists of 12 layers (Transformer blocks) and 12 self-attention heads. See FIGURE 3 for the basic structure of Transformer.

The BERT encoder can output a hidden state vector or a time-step sequence of hidden state vectors. In BERT4TC, we only use the final hidden state vector $H_{[\text{CLS}]} \in R^h$ of the special [CLS] as the aggregate representations of the sequence, where h is the dimension with a default value of 768.

D. OUTPUT LAYER

The output layer is a simple softmax classifier on the top of BERT encoder. Let θ be the set of all trainable parameters for BERT4TC, the output layer turns the vector $H_{[\text{CLS}]}$ into the conditional probability distributions $P(y_i|H_{[\text{CLS}]}, \theta)$ over all categorical labels $y = \{y_1, \dots, y_c\}$ (or $y = \{0, 1\}$) as follows:

$$P(y_i|H_{[\text{CLS}]}, \theta) = \frac{\exp(P(y_i|H_{[\text{CLS}]}, \theta))}{\sum_{j=1}^c \exp(P(y_j|H_{[\text{CLS}]}, \theta))} \quad (2)$$

where $V \in R^{c \times h}$ is the trainable task-specific parameter matrix and c is the number of labels.

Let t be the true label of the input sequence x , we take the label with the largest $y_x = \text{argmax}(P(y_i|H_{[\text{CLS}]}, \theta))$ value as the predicted result and compute a standard calculation loss $J(x, \theta)$ based on the canonical cross-entropy function as follows:

$$J(x, \theta) = \begin{cases} -t \ln P(y_x) - (1-t) \ln(1 - P(y_x)) & if c = 2 \\ -\ln P(y_x) & if c > 2 \end{cases} \quad (3)$$

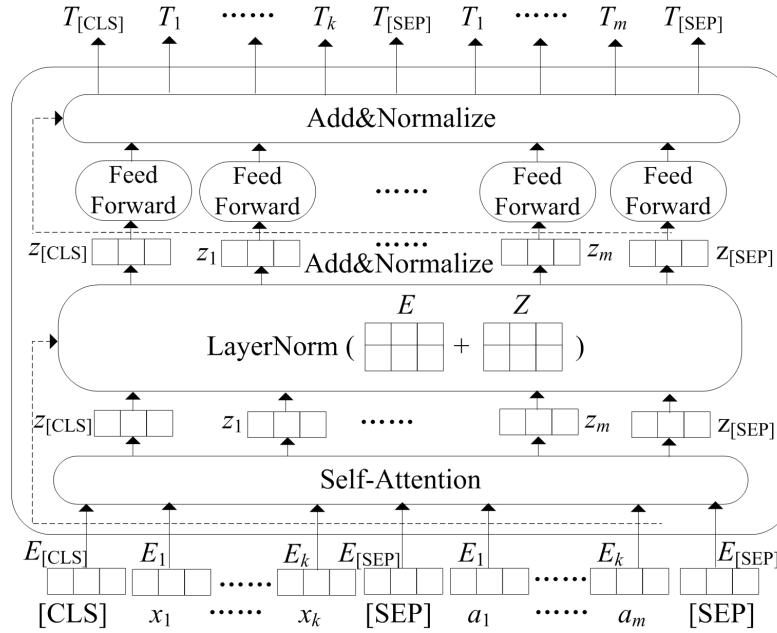


FIGURE 3. Basic structure of transformer.

We use the parameter `batch_size` to denote the number of each training batch. To avoid over-fitting, the regularization strategy Dropout is adopted and the value is always kept at 0.1. BERT4TC uses the default Adam optimizer with $\text{beta}_1 = 0.9$ and $\text{beta}_2 = 0.999$. The parameter of learning rate is denoted as `lr` for short, which might vary in different situations. We fine-tune all trainable parameters from BERT4TC as well as V jointly by maximizing the log-probability of the correct label.

E. POST-TRAINING FOR BERT

In latter experiments, we can see that constructing auxiliary sentence for multi-label text classification task can allow BERT to better utilize limited supervised training data and address task-awareness challenge. However, since BERT is pre-trained on the common BooksCorpus and English Wikipedia text corpus without understanding of text classification knowledge, only fine-tuning BERT in the downstream task might still leave the domain challenge unresolved. Especially, When sufficient training data for fine-tuning are unavailable, external existing domain-related corpus might provide another effective way for alleviating the task-awareness challenge. To enhance the performance of text classification task, we may need to reduce the bias introduced by domain-unrelated knowledge (e.g., from Wikipedia corpora) and fuse domain-related knowledge (from unsupervised domain data) via post-training BERT and task-specific knowledge (from supervised task-targeted training data but out-of-domain data) via fine-tuning.

To post-training on domain knowledge, we continue to use two pre-training objectives proposed in BERT: masked language model (MLM) and next sentence prediction (NSP).

The former tries to predict randomly masked words and the latter tries to detect whether two sides of the input are from the same document or not. Just as what has been pointed out in [12] and [17], MLM is crucial for injecting domain knowledge and for alleviating the bias of the knowledge from Wikipedia, while NSP encourages BERT to learn contextual representation beyond word-level. In the latter experiments, we use domain-related corpus to further post-training pre-trained BERT and compare these two different training objectives.

We define the loss function of MLM as J_{MLM} and the loss function of NSP as J_{NSP} , then the total joint loss of the domain knowledge post-training is $J_{DK} = J_{MLM} + J_{NSP}$.

Algorithm 2 Post-Training Algorithm for BER

Input: T_{DK} : one batch of domain knowledge training data
trainType: type of data

- 1: **function** `post_training_bert(T_{DK} , trainType)`
- 2: assert `trainType` in {"MLM", "MLM-NSP"}
- 3: $\nabla_{\theta} J_{DK} \leftarrow 0$; $u = |T_{DK}|$
- 4: for (i, sent) in enumerate(T_{DK}):
- 5: if `trainType` == "MLM": $J_{partial} = J_{MLM}(\text{sent})$
- 6: if `trainType` == "MLM-NSP":
- 7: $J_{partial} = J_{MLM}(\text{sent}) + J_{NSP}(\text{sent})$
- 8: $\nabla_{\theta} J_{DK} \leftarrow \nabla_{\theta} J_{DK} + \text{BackProp}(\frac{J_{partial}}{u})$
- 9: $\theta \leftarrow \text{ParameterUpdates}(\nabla_{\theta} J_{DK})$

Algorithm 2 describes one training step by taking one batch of data on domain knowledge T_{DK} to update the parameters θ of BERT. The gradients ∇_{θ} of all parameters are firstly initialized as 0 to prepare gradient computation. The partial

TABLE 2. Statistics about seven datasets.

Dataset	Type	Label Number	Average Length	Max Length	Average Token Length	Max Token Length	Train	Dev	Test
rt	Sentiment	2	22	61	25	76	7676	853	2133
sstb2	Sentiment	2	10	56	12	64	67349	872	1821
imdb	Sentiment	2	234	2494	250	2732	2.25k	2500	2.5k
sstb5	Sentiment	5	20	55	22	61	7680	854	2210
TREC	Question	6	10	37	11	39	4906	546	500
AGnews	Topic	4	38	172	52	1014	110k	10k	7.6k
DBPedia	Topic	14	52	1588	59	3827	550k	10k	70k

loss $J_{partial}$ is computed with different training objectives, e.g., "MLM" and "MLM-NSP", and accumulated, which finally yields the gradients by backpropagation. We detail some typical hyper-parameter settings of this algorithm in the following experiments.

IV. EXPERIMENTS

In this section, we evaluate our model on seven widely-studied datasets used in many related works (e.g., [6], [12]–[14], [16]). These datasets have varying numbers of sequence length and can be divided into three types:

Sentiment Analysis For sentiment analysis, we evaluate our model on the binary movie review Rotten Tomatoes (rt for short) [20], binary and five-class versions of the Stanford Sentiment Treebank (sstb2 for binary-class and sstb5 for five-class) [21], and on the binary Internet Movie Database (imdb for short) [22].

Question classification For question classification, we evaluate our model on the six-class version of the TREC dataset [23], which is a dataset for question classification consisting of open-domain, fact-based questions divided into broad semantic categories. TREC dataset is sentence-level, and there are fewer training samples for it.

Topic classification For topic classification, we use large-scale AG's News (denoted as AGnews) and DBPedia created by Xiang *et al.* [24]. AGnews is an internet news article dataset. We choose the 4 largest classes, World, Entertainment, Sports and Business, to construct the dataset, using only the title and description fields. The number of training samples for each class is 30,000 and testing 1900. DBPedia is a crowd-sourced community effort to extract structured information from Wikipedia, and is constructed by picking 14 non-overlapping classes from DBpedia 2014. From each of these 14 ontology classes, we randomly choose 40,000 training samples and 5,000 testing samples. The fields we used for DBPedia come from the title and abstract of each Wikipedia article.

Some statistic information about these seven dataset is listed in TABLE 2, where the Average Length and Max Length columns respectively refer to the average and maximum non-tokenized sequence lengths of training samples, while the Average Token Length and Max Token Length columns respectively refer to the average and maximum tokenized lengths after segmentation by WordPiece.

According to whether the max tokenized length exceeds 512 or not, we regard rt, sstb2, sstb5 and TREC as short-text datasets, and imdb, AGnews and DBPedia as long-text datasets in our experimental situations.

Due to the limitation of machine resources and the insensitivity to word case for text classification task, we only choose the pre-trained uncased BERT (BERT-base) model for fine-tuning in all later experiments. The number of Transformer blocks is 12, the hidden layer size is 768, the number of self-attention heads is 12, and the corpus for training is BooksCorpus+ Wikipedia with total 3.3 billions tokens.

A. EXPERIMENT 1: FINE-TUNING ANALYSIS FOR SHORT-TEXT DATASETS

In this section, we use BERT4TC-S as the default model to evaluate different fine-tuning strategies on the above short-text datasets from the perspectives of learning rate lr, sequence length and hidden state vector selection. The parameter batch_size is uniformly set to 24, and the maximum sequence length is set to the corresponding Max Length values given in TABLE 2.

(1) Comparisons of learning rates

The results of accuracy (Acc) and macro F1 (F1) metrics under the conditions of different lr are reported in TABLE 3. For ease of comparisons, we emphasize the best results in boldface.

From the results in TABLE 3, we can see that learning rate settings have significant impacts on the performance of the model. In most of the cases, lower lr value yields better results. Especially when lr is set to 2e-05, BERT4TC-S achieves the highest Acc and macro F1 values on all short-text datasets. As the value of lr increases, the model fails to converge and the performance decreases. Sun *et al.* in [16] also found that a lower learning rate, such as 2e-05, is necessary to make BERT overcome the catastrophic forgetting problem, while aggressive learning rate of 5e-04 or 4e-04 will tend to cause the training set fail to converge. In this experiment, we find similar situations and come to the same conclusions. Therefore, in all following experiments lr is uniformly set to 2e-05 for short-text datasets without further explicit mention.

(2) Comparisons of sentence lengths

For short-text datasets, the maximum sequence length is far lower than 512. So, we consider three kinds of different

TABLE 3. Results of BERT4TC-S on short-text datasets with different lr.

Datasets	Learning Rate lr	Acc	F1
rt	5e-05	0.8678	0.8674
	4e-05	0.8748	0.8746
	3e-05	0.8659	0.8658
	2e-05	0.8748	0.8747
	1e-05	0.8706	0.8705
sstb2	5e-05	0.9325	0.9324
	4e-05	0.9379	0.9379
	3e-05	0.9357	0.9357
	2e-05	0.9385	0.9385
	1e-05	0.9286	0.9286
sstb5	5e-05	0.5394	0.5288
	4e-05	0.50308	0.5212
	3e-05	0.5267	0.5192
	2e-05	0.5425	0.5343
	1e-05	0.5376	0.5242
TREC	5e-05	0.964	0.947
	4e-05	0.966	0.9621
	3e-05	0.974	0.9697
	2e-05	0.974	0.9698
	1e-05	0.97	0.9554

TABLE 4. Results of BERT4TC-S on short-text datasets with different sequence lengths.

Datasets	Sequence Length	Acc	F1
rt	2/3*maxlen=41	0.8669	0.8668
	maxlen=61	0.8748	0.8747
	token_maxlen=76	0.8795	0.8795
sstb2	2/3*maxlen=37	0.9319	0.9319
	maxlen=56	0.9385	0.9385
	token_maxlen=64	0.9385	0.9385
sstb5	2/3*maxlen=37	0.5231	0.5151
	maxlen=55	0.5425	0.5343
	token_maxlen=61	0.5443	0.5379
TREC	2/3*maxlen=26	0.966	0.9401
	maxlen=37	0.97	0.9581
	token_maxlen=39	0.976	0.9793

sequence lengths in the following experiments: maximum sequence length (maxlen for short), maximum token length (token_maxlen for short) and part of maximum sequence length (e.g., two-thirds of maxlen). The Acc and macro F1 results are shown in TABLE 4.

From the results in TABLE 4, we can see that:

(1) When using token_maxlen as the maximum sequence length, BERT4TC-S obviously outperforms the models with other lengths on all datasets. More smaller than token_maxlen the maximum length is set, more worse the model will perform, which can be seen from the results on rt and sstb5. Since using token_maxlen allows to preserve all tokens in

the sequences without losing any information, as a result the model can better capture the complete meanings of the sequences and thus learn more task-specific knowledge, which finally achieves better results. When the length is smaller, the model has to discard some tokens, thus decrease the performance of the model.

(2) For sstb2, the model achieves the same performance under the conditions of maxlen and token_maxlen, which implies that they make no difference for fine-tuning and ensuring the full task-awareness of the model. According to the statistics about sstb2, we can find that in its total 67349 training samples there are only 26 sentences with the lengths exceeding 56. Since sstb2 has relatively large amounts of training data, the model can be fully trained and thus using maxlen produces minor impacts. While for rt, sstb5 and TREC datasets, although there are also very few samples with the lengths exceeding token_maxlen, the model with token_maxlen yields better results due to the corresponding limited training data.

(3) When only part of maxlen is considered, e.g., two-thirds of maxlen, the performance of the model decreases quickly, which implies that choosing suitable maximum sequence length is of great importance. Obviously, larger maximum length allow the BERT model to yield more complete aggregated representations and better performance, especially when sufficient training data are unavailable. Of course, it comes with the expense of quickly increasing trainable parameters and training cost. However, if sufficient supervised training data are available, then larger maximum length might not always guarantee better performance after exceeding some certain threshold value.

B. EXPERIMENT 2: FINE-TUNING ANALYSIS FOR LONG-TEXT DATASETS

Different from the short-text datasets mentioned above, training samples in the long-text datasets might contain more than 512 tokens, which as a result have to be shorten so as to meet the input length restriction of BERT. Experimental results from [16] show that BERT performs best on imdb when the input is built by concatenating the first 128 tokens and the final 352 tokens. In the same way, we apply such sequence treatment on all other long-text datasets and then report the Acc and macro F1 results of BERT4TC-S with different learning rates in TABLE 5.

Similarly, we can see that a lower learning rate, e.g., 1e-05 in this experiment, makes the model achieve the highest Acc and macro F1 values, which are a little better than using the setting of 2e-05. Therefore, we use lr of 1e-05 as the default setting in the following experiments on long-text datasets.

Since Sun et al. in [16] already compared different length fine-tuning strategies of BERT on imdb, we only focus on the remaining AGnews and DBPedia in the following experiments. See the Acc and macro F1 results as TABLE 6 shows.

From results in TABLE 6, we can see that the numbers of training samples with the length smaller than 100 on AGnews and DBpedia accounts for 98% and 94.1% of the

TABLE 5. Results of BERT4SC-S on long-text datasets with different learning rates.

Datasets	Learning Rate lr	Acc	F1
imdb	5e-05	0.9379	0.9379
	4e-05	0.9398	0.9398
	3e-05	0.9449	0.9448
	2e-05	0.9456	0.9456
	1e-05	0.9458	0.9458
AGnews	5e-05	0.9404	0.9404
	4e-05	0.9431	0.9421
	3e-05	0.9459	0.9459
	2e-05	0.9471	0.9471
	1e-05	0.9475	0.9475
DBPedia	5e-05	0.9903	0.9903
	4e-05	0.9918	0.9918
	3e-05	0.9926	0.9926
	2e-05	0.9926	0.9926
	1e-05	0.9932	0.9932

TABLE 6. Results of BERT4TC-S with different sequence lengths on AGnews and DBPedia.

Datasets	Max Sequence Length	Percent	Acc	F1
AGnews	52	57.60%	0.8158	0.8158
	70	91.10%	0.9231	0.9231
	100	98.00%	0.9475	0.9475
	512	98.00%	0.9475	0.9475
DBPedia	59	50.30%	0.9928	0.9928
	70	61.90%	0.9928	0.9928
	100	94.10%	0.9932	0.9932
	512	94.10%	0.9932	0.9932

total samples respectively. As the maximum length increases, the Acc and macro F1 values remain unchanged even if the length is up to 512. That means that increasing the maximum length from 100 up to 512 makes no sense for the performance of the model, except for more training costs. Consequently, we use 100 as the default maximum length setting for both AGnews and DBPedia. We also find that as the maximum length decreases if it is smaller than 100, the performance of the model also drops accordingly. But AGnews drops more quickly than DBPedia. We think the main reason might be due to the facts that the sequences in DBPedia are made up of the title and abstract fields of the article, which as a result contain more informative tokens than AGnews whose sequences consist of the title and description fields, because the abstract of the article is usually more concise and informative than the description part.

C. EXPERIMENT 3: COMPARISONS OF HIDDEN STATE VECTOR SELECTIONS

To further extend the work of [16] that only focuses on layer selection and imdb, we compare the impacts

TABLE 7. Results of BERT4TC-S with different hidden state vectors.

Datasets	Models	Acc	F1
rt	BERT4TC-S	0.8795	0.8795
	BERT4TC-S-last	0.8725	0.8725
	BERT4TC-S-mean	0.8776	0.8776
	BERT4TC-S-max	0.8753	0.8753
sstb2	BERT4TC-S	0.9385	0.9385
	BERT4TC-S-last	0.9379	0.9379
	BERT4TC-S-mean	0.9374	0.9374
	BERT4TC-S-max	0.9368	0.9368
imdb	BERT4TC-S	0.9364	0.9364
	BERT4TC-S-last	0.934	0.934
	BERT4TC-S-mean	0.9308	0.9308
	BERT4TC-S-max	0.9319	0.9311
sstb5	BERT4TC-S	0.5443	0.5379
	BERT4TC-S-last	0.5285	0.5228
	BERT4TC-S-mean	0.5348	0.5291
	BERT4TC-S-max	0.5389	0.5291
TREC	BERT4TC-S	0.976	0.9793
	BERT4TC-S-last	0.970	0.9659
	BERT4TC-S-mean	0.974	0.969
	BERT4TC-S-max	0.970	0.9592
AGnews	BERT4TC-S	0.9475	0.9475
	BERT4TC-S-last	0.9468	0.9468
	BERT4TC-S-mean	0.9467	0.9467
	BERT4TC-S-max	0.9471	0.9471
DBPedia	BERT4TC-S	0.9932	0.9932
	BERT4TC-S-last	0.9918	0.9918
	BERT4TC-S-mean	0.9929	0.9929
	BERT4TC-S-max	0.9926	0.9926

of selecting different hidden state vectors of BERT's last layer on all datasets in this section. Let BERT4TC-S and BERT4TC-S-last respectively denote the conditions that the sequences are represented as the hidden state vectors of the [CLS] and [SEP] tokens, and BERT4TC-S-mean and BERT4TC-S-max respectively denote the situations that sequences are presented as the average and maximum values of all outputs from the last layer. See TABLE 7 for the Acc and macro F1 results.

From TABLE 7, we can see that BERT4TC-S achieves highest Acc and macro F1 values on all datasets, which implies that the hidden vector of the [CLS] can better catch the semantics of the corresponding sequence. When selecting other hidden state vectors, e.g., last, average or maximum ones, the model performs differently and unsteadily, but all not as good as BERT4TC-S.

D. EXPERIMENT 4: COMPARISONS OF DIFFERENT AUXILIARY SENTENCES.

In this section, we compare BERT4TC-S with other BERT4TC models that contain auxiliary sentence. We report the Acc, macro F1 and Precision (P) metrics in TABLE 8.

TABLE 8. Comparisons of BERT4TC without/with auxiliary sentence.

Datasets	Models	Acc	F1	P
rt	BERT4TC-S	0.8795	0.8795	0.8797
	BERT4TC-AQ	0.8701	0.8701	0.8701
	BERT4TC-AA	0.8683	0.8683	0.8683
	BERT4TC-AWA	0.8654	0.8654	0.8654
sstb2	BERT4TC-S	0.9385	0.9385	0.9397
	BERT4TC-AQ	0.9352	0.9351	0.9368
	BERT4TC-AA	0.9357	0.9357	0.9357
	BERT4TC-AWA	0.9344	0.9344	0.9344
imdb	BERT4TC-S	0.9364	0.9364	0.9364
	BERT4TC-AQ	0.9301	0.9301	0.9301
	BERT4TC-AA	0.9285	0.9285	0.9284
	BERT4TC-AWA	0.9276	0.9276	0.9276
sstb5	BERT4TC-S	0.5443	0.5379	0.5395
	BERT4TC-AQ	0.543	0.5349	0.5423
	BERT4TC-AA	0.8294	0.7274	0.7395
	BERT4TC-AWA	0.8256	0.7257	0.7331
TREC	BERT4TC-S	0.976	0.9793	0.9819
	BERT4TC-AQ	0.974	0.9685	0.98
	BERT4TC-AA	0.991	0.9838	0.9834
	BERT4TC-AWA	0.99	0.982	0.9813
AGnews	BERT4TC-S	0.9475	0.9475	0.9475
	BERT4TC-AQ	0.9405	0.9405	0.9403
	BERT4TC-AA	0.9612	0.9612	0.961
	BERT4TC-AWA	0.9607	0.9607	0.9607
DBPedia	BERT4TC-S	0.9932	0.9932	0.9932
	BERT4TC-AQ	0.9901	0.9901	0.9901
	BERT4TC-AA	0.9987	0.9987	0.9987
	BERT4TC-AWA	0.9981	0.9981	0.9981

From the results in Table 8, we can see that:

(1) For all datasets, BERT4TC-AQ with question auxiliary sentence doesn't bring any performance improvements to the model, but decreases it on the contrary. The main reason according to our analyses is because the auxiliary sentence doesn't introduce any informative messages or relation semantics to the original target input sequence. Although the model can still learn the representations of each sentence-pair sequence via fine-tuning, the performance is decreased slightly due to the newly added meaningless and noisy auxiliary sentence.

(2) For binary datasets, e.g., rt, sstb2 and imdb, all BERT4TC models with auxiliary sentences perform worse than BERT4TC-S, no matter whether the auxiliary sentence contains categorical label information or not. The new objective of the sentence-pair task is converted to judge whether two sentences are relevant or not. As a result, the reasons of worse performance might be mainly due to the lost of some semantic information during the process of conversion, since it forces the model to learn the relation between the sentence pair, instead of the target sentiment polarity for the sentence

directly. More importantly, the model can't learn more task-specific knowledge because of the not increasing number of training data.

(3) For multi-class datasets, e.g., sstb5, TREC, AGnews and DBpedia, both BERT4TC-AA and BERT4TC-AWS significantly outperform BERT4TC-S and BERT4TC-QA. Especially, BERT4TC-AA achieves the accuracy improvements by 28.64%, 1.5%, 1.37% and 0.51% respectively over BERT4TC-S that the input sequence is a single sentence. This definitely illustrates the effectiveness of auxiliary sentence. For sstb5 and TREC, the numbers of training samples are relatively limited with respect to each label, which as a result might not be insufficient for fine-tuning BERT to ensure full task-awareness of the model. However, by constructing auxiliary sentences and converting the multi-class classification task into the binary one, we can make full use of the limited supervised training data and increase them several times, which as a result allows the model to be more fully trained and learn more task-specific knowledge. Additionally, the new binary objectives might also be helpful for reducing the uncertainties caused by some easily confusing labels in the multi-class situations, e.g., very negative and negative in sstb5, or World and Business in AGnews.

(4) On all multi-class datasets, BERT4TC-AA obviously outperforms BERT4TC-AWA. Different from the auxiliary sentence proposed in [18] in which both aspect and label might vary with respect to different sentence-pairs, the experimental results of BERT4TC-AA and BERT4TC-AWA on various multi-class datasets demonstrate that only including the categorical label in the auxiliary sentence would be enough, since other meaningless tokens might bring noisy information on the contrary.

E. EXPERIMENT 5: COMPARISONS TO PREVIOUS MODELS

In this section, we compare our method to some recently typical works, especially some famous baseline models such as ELMo [11], GPT [14], ULM-FiT [13] and BERT-base [12]. See TABLE 9 for the accuracy results on the above seven datasets. The results of these baselines come from their original works and “-” means not reported.

As what is shown in TABLE 9, we can see that:

(1) BERT4TC-S outperforms various feature-based models, including Virtual Adversarial, DPCNN and ELMo, on almost all datasets except for sstb5 with a slightly lower results, which confirms the effectiveness of pre-training BERT and fine-tuning approach. Compared with GPT, the accuracy on sstb2 is improved by 2.2% for BERT-base and 2.55% for BERT4TC-S due to their different training parameter settings. It is evident that bidirectional Transformer encoder provides better capability to capture the semantic of sequence than the one-way left-to-right Transformer encoder.

(2) Both BERT4TC-S and ULM-FiT adopt pre-trained general language models and fine-tune them in the downstream task, which is proved to be effective from the better results compared with those feature-based models.

TABLE 9. Accuracies on seven datasets.

Models	rt	sstb2	imdb	sstb5	TREC	AGnews	DBPedia
Virtual Adversarial [6]	0.809	-	0.9409	-	-	-	0.9924
DPCNN [2]	-	-	-	-	-	0.9313	0.9912
ELMo [11]	-	-	-	0.547	-	-	-
GPT [14]	-	0.913	-	-	-	-	-
ULM-FiT [13]	-	-	0.954	-	0.964	0.9499	0.992
BERT-base [12]	-	0.935	0.946	-	0.972	0.9475	0.9929
BERT4TC-S	0.8795	0.9385	0.9458	0.5443	0.976	0.9475	0.9932
BERT4TC-AA	0.8683	0.9357	0.9285	0.8294	0.991	0.9612	0.9987
BERT4TC-AWA	0.8654	0.9344	0.9276	0.8256	0.99	0.9607	0.9981

The performance of BERT4TC-S and ULM-FiT are comparable due to their different training mechanisms and fine-tuning methods. More concretely, ULM-FiT propose discriminative fine-tuning, slanted triangular learning rates and gradual unfreezing techniques to retrain previous knowledge and avoid catastrophic forgetting during fine-tuning, which are quite different from MLN and NSP. They also have different pre-training corpus. Both BERT4TC-S and BERT-base have the same model structures except for different fine-tuning parameter settings and their results are very close.

(3) Although BERT4TC-AA and BERT4TC-AWA don't perform as well as BERT-based or BERT4TC-S on rt and sstb2, they gain much better results than Virtual Adversarial and GPT. However, they perform worse than Virtual Adversarial and ULM-FiT on imdb, which we think is mainly due to the lack of sentence segmentations in imdb and the using of different length pre-treatment methods.

(4) Both BERT4TC-AA and BERT4TC-AWA outperform significantly than all other models on sstb5, TREC, AGnews and DBPedia, including both feature-based methods and fine-tuning methods. Typically, BERT4TC-AA pushes the current state-of-the-art result on sstb5 from 0.547 up to 0.8294 with an absolute improvement by 0.2824. It also achieves state-of-the-art results on other multi-class datasets. Plus the results from Experiment IV-D, we can conclude that constructing auxiliary sentence with label information is indeed effective for multi-class classification task, which not only significantly improves the classification performance by increasing the number of supervised training data to more fully pre-train BERT and ensure task-awareness of the model, but also simplifies the task by converting it into a binary one for directly learning the relations between the target text sequence and the corresponding categorical label with the help of NSP training objective.

F. EXPERIMENT 6: POST-TRAINING

From Experiment IV-E, we found that constructing auxiliary sentence for binary classification task is not helpful. As a result, in this section we conduct experiments to discuss whether post-training BERT with domain-related knowledge

TABLE 10. Information about post-training corpus.

Post-training Corpus	Sample Number	Training Objective
rt	10662	MLM
sstb2	70042	MLM
imdb	50000	MLM
sstb5	10742	MLM
scaledata	301358	MLM+NSP
scaledata_mlm	1087540	MLM

can provide another effective way for addressing domain challenge or not. In the following experiment, we only focus on binary classification tasks for simplification.

We select the multi-class movie review dataset scaledata from [20] as a post-training corpora because it is a paragraph-level dataset consisting of multiple consecutive sentences and different from the above sentence-level datasets, e.g., rt, sstb2, imdb and sstb5. As a result, we can compare the different effects of MLM and NSP training objectives. Before post-training, we use spaCy tool to split each paragraph of scaledata into independent sentences and keep their original orders in the paragraph. Let scaledata and scaledata_mlm denote that the post-training objective is MLM+NSP and MLM respectively. For scaledata, each training sample consists of a couple of sentences with their original sequence orders, which also allows the NSP training objective to learn each sentence and its contexts. While for scaledata_mlm, each training sample only includes a simple sentence and neglects its relations with the previous or next sentence in the paragraph. In addition, since rt, sstb2, imdb and sstb5 are all datasets for movie review texts, we can also use them as the cross post-training corpus. The information about the training sample numbers and the corresponding objectives is listed in Table 10.

We take the above corpus to respectively post-train the public pre-trained uncased BERT model, and then use it to continue to fine-tune BERT4TC in the downstream task with the same settings as the previous experiments. When post-training, we use the maximum sequence length of 128, the learning rate lr of 5e-05, and the training epoch number

TABLE 11. Experiments results of BERT4TC on binary sentiment classification without/with post-training corpus.

Dataset	Post-training Corpus	BERT4TC-S			BERT4TC-AA		
		Acc	F1	P	Acc	F1	P
rt	-	0.8795	0.8795	0.8749	0.8683	0.8683	0.8683
sstb2	-	0.9385	0.9385	0.9397	0.9357	0.9357	0.9357
imdb	-	0.9364	0.9364	0.9364	0.9285	0.9285	0.9285
rt	sstb2	0.8678	0.8678	0.8678	0.8626	0.8626	0.8626
	imdb	0.8801	0.8807	0.8801	0.8705	0.8705	0.8705
	scaledata	0.8823	0.882	0.883	0.8712	0.8712	0.8713
	scaledata_mlm	0.8847	0.8847	0.8848	0.8708	0.8708	0.8708
sstb2	rt	0.9396	0.9396	0.9396	0.936	0.936	0.936
	sstb5	0.9407	0.9407	0.9418	0.9336	0.9336	0.9336
	scaledata	0.9412	0.9412	0.9414	0.9349	0.9349	0.9349
	scaledata_mlm	0.9395	0.9395	0.9395	0.9327	0.9327	0.9327
imdb	rt	0.9296	0.9296	0.9297	0.9281	0.9281	0.9280
	sstb2	0.9213	0.9213	0.9213	0.9210	0.9210	0.9210
	scaledata	0.9316	0.9316	0.9316	0.9315	0.9315	0.9315
	scaledata_mlm	0.9351	0.9351	0.9351	0.9348	0.9348	0.9348

of 3. For easy to compare, we also include the previous results of BERT4TC-S and BERT4TC-AA on rt, sstb2 and imdb. The Acc, macro F1 and P (precision) results are reported in Table 11, where “-” in the “Post-training Corpus” column means no post-training.

From the results in Table IV-F, we can see that:

(1) For rt, when using sstb2 as the post-training corpora, the performance isn't increased as we expected, but decreases on the contrary. According to our analysis, we find that the main reason might be due to the relatively low quality of samples in sstb2. There are more than 67% of training data in sstb2 have the lengths of no more than 5 tokens (including the meaningless punctuation symbol). As a result, the post-training with sstb2 will bring noisy data and decrease the performance of the model. When post-training with imdb, the performance of the model is only slightly improved, which might be due to the lack of structure and segment information in imdb. When post-training with scaledata or scaledata_mlm datasets, the performance of the model is improved obviously, since the review texts in rt and scaledata or scaledata_mlm are highly related, which as a result can offer more domain-related knowledge and better address domain challenge.

(2) For sstb2, the performance of the model is improved no matter using rt, sstb5, scaledata or scaledata_mlm as post-training corpora. Different from the post-training impacts of sstb2 on rt, the performance is improved because of the high quality of training data in rt, most of which contain complete sentence meanings, instead of several tokens. Although sstb5 is a five-class version dataset, the performance is still improved because of their highly related domain and task knowledge between sstb2 and sstb5. When post-training with scaledata or scaledata_mlm, the performance is also improved obviously.

(3) For imdb, the performance of both BERT4TC-S and BERT4TC-AA decreases after post-training with each corpora. We believe that the main reason mainly lies in imdb itself. Each sample in imdb is a text sequence that covers a couple of sentences without any segment information and is also pre-processed by only considering part of the sequence, which as a result constrains the model to better utilize the sentence-level domain knowledge in the post-training corpus, as well as to learn the complete semantics or meanings for the sequence.

(4) No matter using post-training approach or not, BERT4TC-S outperforms BERT4TC-AA with similar reasons as what we have discussed in Experiment IV-D. According to the results of scaledata and scaledata_mlm on rt and sstb2, both BERT4TC-S and BERT4TC-AA perform better when post-training with scaledata, since scaledata uses MLM+NSP as the training objective, which as a result allows to utilize the additional relationship between sentences for enhancing the understandings of the target sentence. Since imdb lacks of explicit sentence segmentation information, post-training with scaledata_mlm would perform better than that of scaledata by simply considering the word-level meanings and relationships.

V. CONCLUSION

Using general language model BERT pre-trained on large scale of unlabeled corpus and fine-tune it in the downstream tasks has achieves many state-of-the-art results in multiple NLP tasks. In this paper, we propose a BERT-based text classification model by constructing auxiliary sentence to turn the task into a sentence-pair one, aiming to incorporate more task-specific knowledge and address task-awareness challenge. We also propose a post-training approach to utilize domain-related corpus for addressing domain challenge.

Extensive experiments over seven widely-used text classification datasets are firstly conducted to analyze some typical fine-tuning strategies for BERT from the perspectives of learning rate, sequence length and hidden state vector selection, which can provide a supplement to the existing work. After that, we analyze our model with different auxiliary sentences and post-training objectives in detail, as well as to other recently typical works. The experimental results show that BERT4TC with suitable auxiliary sentence significantly outperforms both typical feature-based methods and fine-tuning methods, and achieves new state-of-the-art performance on multi-class classification datasets. While for binary classification datasets, our BERT4TC post-trained strategy on suitable domain-related corpus also achieves better results compared with original BERT model.

In the future, we would like to further explore the better ways of incorporating task-specific and domain-related knowledge into BERT with in-domain and cross-domain pre-training. We also try to design more effective neural networks on the top of BERT encoder instead of a single softmax classifier.

REFERENCES

- [1] S. Holge, B. Loic, C. Alexis, and L. Yann, “Very deep convolutional networks for text classification,” in *Proc. 15th Conf. Eur. Chapter Assoc. for Comput. Linguistics*, vol. 1, 2017, pp. 1107–1116.
- [2] R. Johnson and T. Zhang, “Deep pyramid convolutional neural networks for text categorization,” in *Proc. 55th Annu. Meeting Assoc. Comput. Linguistics*, Vancouver, BC, Canada, vol. 1, 2017, pp. 562–570.
- [3] K. Filippou and P. Alexandros, “Structural attention neural networks for improved sentiment analysis,” in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2017, pp. 586–591.
- [4] S. Tao, Z. Tianyi, L. Guodong, J. Jing, P. Shirui, and Z. Chengqi, “DISAN: Directional self-attention network for rnncnn-free language understanding,” in *Proc. 32nd AAAI Conf. Artif. Intell. (AAAI)*, Apr. 2018, pp. 5446–5455.
- [5] Y. Zichao, Y. Dify, D. Chris, H. Xiaodong, S. Alex, and H. Eduard, “Hierarchical attention networks for document classification,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, Jun. 2016, pp. 1480–1489.
- [6] T. Miyato, A. M. Dai, and I. Goodfellow, “Adversarial training methods for semi-supervised text classification,” in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–12.
- [7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 3111–3119.
- [8] M. Tomas, C. Kai, C. Greg, and D. Jeffrey, “Efficient estimation of word representations in vector space,” in *Proc. Workshop (ICLR)*, Jan. 2013.
- [9] P. Jeffrey, S. Richard, and M. D. Christopher, “Glove: Global vectors for word representation,” in *Proc. Empirical Methods Natural Lang. Process. (EMNLP)*, vol. 12, 2014, pp. 1532–1543.
- [10] B. McCann, J. Bradbury, and C. Xiong, “Learned in translation: Contextualized word vectors,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6294–6305.
- [11] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proc. North Amer. Assoc. Comput. Linguistics (NAACL)*, 2018, pp. 2227–2237.
- [12] D. Jacob, C. Ming-Wei, L. Kenton, and T. Kristina, “Bert: Pre-training of deep bidirectional transformers for language understanding,” Tech. Rep., 2018.
- [13] H. Jeremy and R. Sebastian, “Universal language model fine-tuning for text classification,” Tech. Rep., 2018.
- [14] R. Alec, N. Karthik, S. Tim, and S. Ilya, “Improving language understanding with unsupervised learning,” Tech. Rep., 2018.
- [15] V. Ashish, S. Noam, P. Niki, U. Jakob, J. Llion, G. N. Aidan, K. Lukasz, and P. Illia, “Attention is all you need,” in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 5998–6008.
- [16] C. Sun, X. Qiu, Y. Xu, and X. Huang, “How to fine-tune BERT for text classification?” May 2019, *arXiv:1905.05583*. [Online]. Available: <https://arxiv.org/abs/1905.05583>
- [17] H. Xu, B. Liu, L. Shu, and P. S. Yu, “Bert post-training for review reading comprehension and aspect-based sentiment analysis,” Apr. 2019, *arXiv:1904.02232*. [Online]. Available: <https://arxiv.org/abs/1904.02232>
- [18] C. Sun, H. Luyao, and Q. Xipeng, “Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence,” Mar. 2019, *arXiv:1903.09588*. [Online]. Available: <https://arxiv.org/abs/1903.09588>
- [19] W. Yonghui et al., “Google’s neural machine translation system: Bridging the gap between human and machine translation,” Sep. 2016, *arXiv:1609.08144*. [Online]. Available: <https://arxiv.org/abs/1609.08144>
- [20] B. Pang and L. Lee, “Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales,” in *Proc. ACL*, Jun. 2005, pp. 115–124.
- [21] S. Richard, P. Alex, W. Jean Y. C. Jason, and D. Christopher, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Oct. 2013, pp. 1631–1642.
- [22] M. L. Andrew, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, Jun. 2011, pp. 142–150.
- [23] V. M. Ellen and T. M. Dawn, “The TREC-8-question answering track evaluation,” in *Proc. 8th Text Retr. Conf.*, Nov. 1999, p. 82.
- [24] Z. Xiang, Z. Junbo, and L. Yann, “Character-level convolutional networks for text classification,” in *Proc. Neural Inf. Process. Syst.*, 2015, pp. 1–9.



SHANSHAN YU was born in 1980. She received the Ph.D. degree. She is currently a Lecturer. She is also a Senior Member of China Computer Federation. Her main research interests include machine learning, big data, and semantic Web.



JINDIAN SU was born in 1980. He received the Ph.D. degree. He is currently an Assistant Professor. He is also a member of CCF. His main research interests include natural language processing, artificial intelligence, and machine learning.



DA LUO is currently pursuing the master’s degree with the South China University of Technology, Guangzhou, China. His current research interests include knowledge graph and natural language processing.