2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application

Evaluating BERT on cloud-edge time series forecasting and sentiment analysis via prompt learning

Qizhi Li[1], Xianyong Li[1]*, Yujia Song[1], Maolin Zhang[1], Longqi Chen[1], Gang Wang[1], Yajun Du[1]
[1]School of Computer and Software Engineering, Xihua University, Chengdu, China
*Corresponding author. Email: lixy@mail.xhu.edu.cn

***Abstract*—Existing pre-trained language models (PTLMs), like BERT, have shown their powerful capabilities in many natural language processing tasks. In sequence analysis, such as time series forecasting, anomaly detection, and sentiment analysis, PTLMs have also achieved new state-of-the-art results. However, does this mean that PTLMs know sequence analysis? This paper explores whether BERT pre-trained on a large amount of data contains knowledge of sequence analysis. Specifically, we adopt prompt learning to see whether BERT will achieve good results on cloud-edge time series forecasting and sentiment analysis tasks. For the cloud-edge time series forecasting task, we give BERT some regular cloud-edge data and let it predict the features of the next time step; For the sentiment analysis task, we give BERT some sentence with sentiment and ask it what sentiment these sentences carry. Our experimental results reveal that: (1) BERT performs not well on the cloud-edge time series forecasting task, which means the logical reasoning of BERT is not good; (2) for sentiment analysis task, BERT with the prompt template performs poorly on both English and Chinese datasets; and (3) for sentiment analysis task, BERT appears to be more likely to perceive the text as carrying positive sentiment.**

***Index Terms*—pre-trained language models, BERT, time series forecasting, sentiment analysis, prompt learning**

I. Introduction

Since could-edge systems contain rich time series data, scholars usually use a large number of time series data with multiple features in the analysis of cloud computing [2], [19], [20], [25]–[27]. Meanwhile, due to the complexity of cloud-edge servers, many sequence analysis tasks, such as anomaly detection [21], [28], mobile edge computing [3], [4], and time series forecasting [22], have arisen. Sentiment analysis is another classical sequence analysis task that needs models to classify the sentiment of sentences. There also have many sentiment analysis tasks, such as aspect sentiment analysis [1], and implicit sentiment analysis [5].

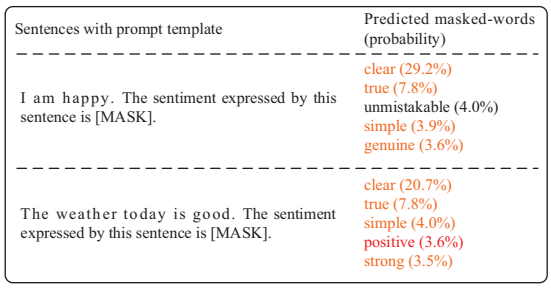

Fig. 1. Using BERT to predict word probabilities for [MASK] using prompt learning. The orange and black words represent the words with positive and neutral sentiments, respectively. The red words represent that they fit the context and sentiment. The sentiment orientations of words are obtained from Subjectivity Clue Lexicon [7].

To deal with these sequence analysis tasks, many models, such as the RNN-based model [29] and the transformer-based model [23], [24], have emerged. In addition, the pre-trained transformer-based models, i.e., pre-trained language models (PTLMs), have achieved many state-of-the-art (SOTA) results in various tasks [15], [22]. The PTLMs have achieved such excellent results because they are trained on large-scale datasets, which made them learn excellent language knowledge and representations [6].

However, few researchers considered whether PTLMs know sequence analysis. In Fig. 1, if we give BERT [15] sentences with sentiment and ask it what sentiment these sentences carry, the answers provided by BERT are out of context[1].

This paper studies whether BERT knows how to analyze sequence. Specifically, by adopting prompt learning [6], we design two tasks for BERT:

This work is partially supported by the Sichuan Science and Technology Program (Nos. 2022YFG0378, 2021YFQ0008), the National Natural Science Foundation of China (Nos. 61802316, 61872298, 61902324), the Key Scientific Research Fund of Xihua University (No. z1422615) and the Innovation Fund of Postgraduate, Xihua University (Grant No. YCJJ2021025).

[1]Experiments can reproduce at https://huggingface.co/bert-base-uncased.

979-8-3503-1993-4/22/$31.00 ©2022 IEEE
DOI 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00051

1) We let BERT forecast on the cloud-edge time series data. This task aims to judge whether BERT can forecast the features of the next time step based on the previous regular cloud-edge data. This task can tell us whether BERT has logical reasoning ability. For example, if we give BERT "*1, 2, 3, 4, [MASK]*", we expect BERT can fill the slot with "*5*".
2) We also let BERT answer the sentiment of sentences. This task aims to judge whether BERT can understand human sentiment. For example, if we give BERT a sentence "*The weather today is good. The sentiment expressed by this sentence is [MASK].*" and an answer space "*positive, negative*", we expect BERT can fill the slot with "*positive*".

For the forecasting task, we evaluate BERT on the Server Machine Dataset (SMD) [29]; for the sentiment analysis task, we evaluate BERT on two datasets: (1) a relatively simple English sentiment analysis dataset, i.e., the IMDb dataset [8]; and (2) a relatively complex Chinese implicit sentiment analysis dataset, i.e., the SMP2019-ECISA dataset[2].

Our findings reveal that: (1) for cloud-edge time series forecasting task, BERT achieves the highest mean square error, which means the logical reasoning ability of BERT is relatively weak; (2) for sentiment analysis task, BERT seems prone to interpreting the text as carrying positive sentiment, which means BERT cannot accurately understand human sentiment.

II. Experimental results on time series forecasting

In this section, we explore whether BERT can infer the information of the next time step based on the regularity of the previous N time steps. Our goal is to give BERT a set of regular time series and see if BERT can find the pattern to predict the features of the next time step.

A. Model architecture

The architecture of the time series forecasting model is shown in Fig. 2. The left part is the time series encoding process; The right part is the next time step forecasting process. Let the input features at time i be $\boldsymbol{x}_i = \{x_i^1, x_i^2, ..., x_i^n\}$, where x_i^k is the $k-$th feature ($k \in [1, n]$) in time i, and n is the number of features. In each sliding window, which contains m timestamps, is defined as $\boldsymbol{X}_j = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_m\}$, where j is the $j-$th timestamp, and $i \in [1, m]$. $\boldsymbol{X}_j$ is a two-dimensional matrix, denoted as $\boldsymbol{X}_j \in \mathbb{R}^{m \times n}$. Moreover, we denote the features for the next step corresponding to the $j-$th sliding window be $\boldsymbol{y}_j = \{x_j^1, x_j^2, ..., x_j^n\}$, where $\boldsymbol{y}_j$ is a vector, with the shape of $\boldsymbol{y}_j \in \mathbb{R}^{1 \times n}$.

At the time series encoding stage, we first use a multi-head self-attention layer to encode $\boldsymbol{X}_j$, since multi-head self-attention can capture the dependencies between each feature. The output hidden state, denoted as $\boldsymbol{h}_{att}$, is a two-dimensional matrix $\boldsymbol{h}_{att} \in \mathbb{R}^{m \times n}$, which is calculated as follow,

$$\boldsymbol{h}_{att} = \mathrm{MultiHeadAttn}(\boldsymbol{X}_j), \tag{1}$$

where $\mathrm{MultiHeadAttn}(\cdot)$ is the multi-head self-attention function. Then we adopt BiLSTM to capture the dependencies between each timestamp. The last hidden state output by BiLSTM, denoted as $\boldsymbol{h}_{lstm} \in \mathbb{R}^{1 \times d_l}$ is a vector (d_l is the hidden size of BiLSTM), carries the dependencies of each feature in every timestamp as follows.

$$\boldsymbol{h}_{lstm} = \mathrm{BiLSTM}(\boldsymbol{h}_{att}), \tag{2}$$

where $\mathrm{BiLSTM}(\cdot)$ is the BiLSTM function. However, d_l is smaller than the dimension of the BERT embedding d_b. Then, we use a fully connected layer to map the hidden state of BiLSTM output to BERT embedding as the following formula.

$$\boldsymbol{h}_s = \boldsymbol{W}^d \boldsymbol{h}_{lstm} + \boldsymbol{b}^d, \tag{3}$$

where $\boldsymbol{h}_s$ is the hidden state of every feature in the sliding window, with the dimension $\mathbb{R}^{1 \times d_b}$; $\boldsymbol{W}^d$ and $\boldsymbol{b}^d$ are two trainable parameters in the fully connected network.

At the forecasting stage, we feed $\boldsymbol{h}_s$ into BERT and ask it what the features of the next time step are. Let the hidden state of previous m timestamps be $[S]$ (where $[S]$ is $\boldsymbol{h}_s$), the prompt template be $[X]$, and the answer slots be $[F]$. We choose '下一个时刻的特征为' ('The features of next step are') as $[X]$, and we use special tokens <feature i> as $[F]$, where $i \in [1, n]$ is the $i-$th feature of the next step.

We first convert $[CLS]$, $[SEP]$, $[X]$ and $[F]$ to word embedding $[\boldsymbol{CLS}]$, $[\boldsymbol{SEP}]$, $[\boldsymbol{X}]$ and $[\boldsymbol{F}]$. Then we concatenate these word embeddings and $[S]$ and feed the concatenated word embeddings into BERT.

Since BERT can capture the dependencies of the entire input sequence, the hidden state $\boldsymbol{h}_{[F]}$ of $[F]$ can carry information of the previous m timestamps and questions. Then we use average pooling to extract the feature maps of $\boldsymbol{h}_{[F]}$. The feature maps are the predicted features $\hat{\boldsymbol{y}}_j$ of BERT.

B. The SMD dataset

The SMD dataset is a real-time dataset of 28 cloud platform servers, each with 38 features, i.e., $n = 38$ (see Table I). We choose the data of the first 8, the last 11, and the rest 9 servers as the training set, the testing set, and the developing set, respectively.

[2]http://sa-nsfc.com/evaluation/ecisa/scheme

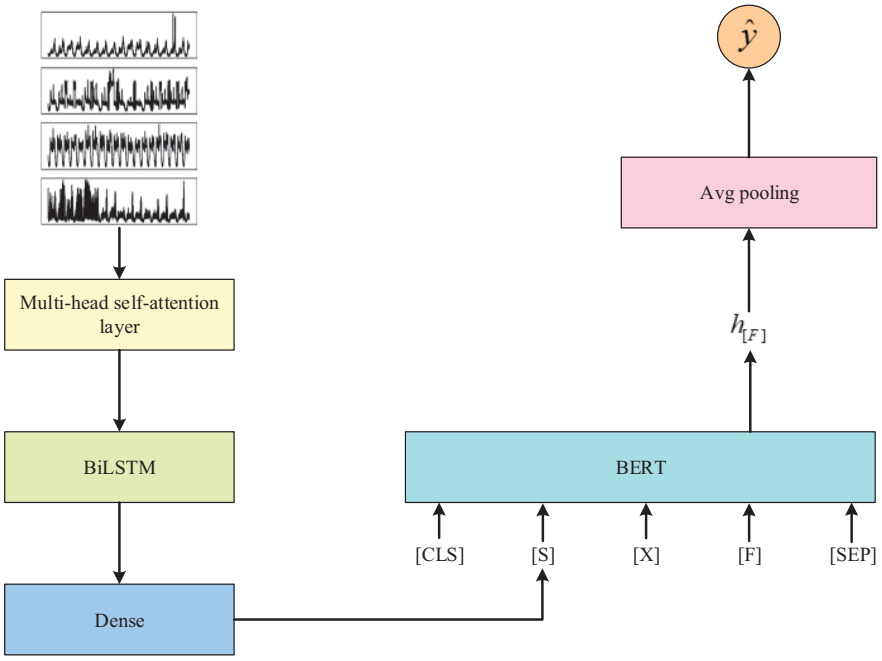

Fig. 2. The architecture of the time series forecasting model. The left part is the time series encoding process; The right part is the next time step forecasting process. In the architecture, [CLS] and [SEP] are the special tokens of BERT; [S] is the hidden state for the previous m time steps; [X] is the prompt template; [F] is the slot that needs to be filled with the forecasted feature for the next step.

TABLE I
DETAIL OF THE SMD DATASET.

	Train	Dev	Test
The amount of data	227,832	256,311	313,269

C. Experiment set-up

The experiments will conduct on the SMD dataset for the following models.

a) ***BERT with prompt learning****:* BERT with prompt learning for time series forecasting, denoted as $\mathrm{BERT}^{\mathrm{t}}_{\mathrm{prompt}}$, is detailed in II-A. The attention heads of $\mathrm{BERT}^{\mathrm{t}}_{\mathrm{prompt}}$ are 2 , $d_l = 150$, $d_b = 768$, the sequence length is 56, the learning rate is $1e-5$, the weight decay is $1e-4$, the batch size is 32, and each sliding window contains 100 timestamps (which means $m = 100$). We adopt AdamW [16] as the optimizer and use the pre-trained Chinese BERT parameters[3] as the start point of BERT.

b) ***OmniAnomaly****:* OmniAnomaly [29] is an anomaly detection model which reconstructs the input data by learning the dependencies and data distribution features of multivariate time series based on GRU and variational auto-encoder (VAE). OmniAnomaly contains two parts: $qnet$ and $pnet$. Specifically, $qnet$ is an encoder that learns the features of the input data and extracts the hidden states of these features. $pnet$ is a decoder that decodes the hidden states and reconstructs the input data.

c) ***GDN*** *[32] :* It adopts a graph structure to represent dependence relationships between different cloud-edge servers and forecast features of the next step by using the graph attention function.

D. Experimental results

The mean squared error (MSE) is selected as the evaluation metric for time series forecasting.

$$\mathcal{L} = \mathrm{MSE}(\hat{\boldsymbol{y}}, \boldsymbol{y}), \tag{4}$$

where $\mathcal{L}$ is the MSE loss, and $\mathrm{MSE}(\cdot)$ is the MSE function.

From Table II, $\mathrm{BERT}^{\mathrm{t}}_{\mathrm{prompt}}$ achieves the highest MSE loss (6.74%), while GDN has the lowest MSE loss (4.50%). They imply that BERT with prompt learning cannot predict the features of the next time step well based on the information from the previous sliding window. Therefore, BERT does not have good logical reasoning ability. This suggests that: (1) using a more powerful model to extract the features in the sliding window; (2) improving the logical reasoning ability of BERT, for example, adding the training

[3]https://huggingface.co/bert-base-chinese.

TABLE II
THE EXPERIMENTAL RESULTS OF $\mathrm{BERT}^{\mathrm{t}}_{\mathrm{prompt}}$, OMNIANOMALY, AND GDN ON THE SMD DATASET (%). BOLD DATA INDICATES THE LOWEST MSE.

	MSE
OmniAnomaly	5.83
GDN	**4.50**
$\mathrm{BERT}^{\mathrm{t}}_{\mathrm{prompt}}$	6.74

TABLE III
DETAIL OF THE IMDB DATASET.

	Train	Dev	Test
positive	11,250	1,250	12,500
negative	11,250	1,250	12,500

objective of logical reasoning in the pre-training stage of BERT.

III. EXPERIMENTAL RESULTS ON SENTIMENT ANALYSIS

In this section, we explore whether BERT can really understand human sentiment. Our goal is to give BERT a sentence with sentiment, and ask BERT to determine which sentiment the sentence carries.

A. Experimental results on IMDb

1) The IMDb dataset [8]: It is an English binary sentiment classification dataset containing 25,000 highly polar movie reviews for training and 25,000 for testing (see Table III)). The training and testing sets have both 12,500 positive and 12,500 negative reviews. We extract 2,500 pieces of data from the training set as the developing set, of which 1,250 positive reviews and 1,250 negative reviews.

2) Experiment set-up: We choose the following models to conduct experiments on the IMDb dataset.

a) ***BERT with prompt learning*** *(*$\mathrm{BERT}_{\mathrm{prompt}}$*):* It does not fine-tune on the training set. It only does cloze tests on the testing set. For a review $[X]$, the template with an answer slot $[MASK]$, the answer space, and the positive and negative sentiment orientations are denoted by temp, $[Z]$, $+$ and $-$, respectively. Table IV shows one example of the *input*, *template*, and *answer* of $\mathrm{BERT}_{\mathrm{prompt}}$.

b) ***BERT*** *[15]:* It is a classical Transformer-based model [10]. When BERT does text classification tasks, the first token $[CLS]$ output by BERT will carry the information of the entire sentence, and BERT will send it to a classifier for classification. When we fine-tune BERT on the training set, we choose two modes: (1) only fine-tuning the parameters of the classifier (denoted as $\mathrm{BERT}_{\mathrm{cls}}$); and (2) fine-tuning the whole BERT (denoted as $\mathrm{BERT}_{\mathrm{whole}}$).

TABLE IV
ONE EXAMPLE OF THE *input*, *template*, AND *answer* OF $\mathrm{BERT}_{\mathrm{prompt}}$ IN THE IMDB DATASET.

$[X]$	The acting was bad.
temp	Totally, it was $[MASK]$.
$[Z]$	{good: $+$, fascinating: $+$, perfect: $+$, bad: $-$, horrible: $-$, terrible: $-$}

TABLE V
THE HYPERPARAMETER SETTINGS OF BERT AND BIGRU ON THE IMDB DATASET.

	BERT	BiGRU
epochs	100	100
batch size	32	64
hidden size	768	150
dropout	-	0.5
learning rate	1e-6	1e-3
weight decay	1e-4	1e-4
sequence length	512	512

c) ***RoBERTa*** *[11]:* It is a more robust BERT. Similar to BERT, RoBERTa feeds the hidden state of the token *[CLS]* into a classifier to output the sentiment label. We also use the following two modes to fine-tune RoBERTa: (1) $\mathrm{RoBERTa}_{\mathrm{cls}}$ stands for only fine-tuning the classifier of RoBERTa; and (2) $\mathrm{RoBERTa}_{\mathrm{whole}}$ stands for fine-tuning all the parameters of RoBERTa.

d) ***BiGRU*** *[12]:* It is a traditional RNN-based model. We use pre-trained 300-dimensional Word2Vec [13] as the word embedding. Then we feed the hidden states that are output by a layer of BiGRU into one classifier to analyze sentiment.

We conduct our experiments on one A40 (48GB) and use AdamW [16] as the optimizer. We use pre-trained English parameters as the start point of $\mathrm{BERT}_{\mathrm{prompt}}$, $\mathrm{BERT}_{\mathrm{cls}}$, and $\mathrm{BERT}_{\mathrm{whole}}$[4]. The hyperparameter settings of the above models are shown in Table V.

3) Experimental results: From Table VI, $\mathrm{BERT}_{\mathrm{whole}}$ achieves the highest accuracy, precision, and F1-score (93.10%, 92.19%, and 93.17% respectively); $\mathrm{BERT}_{\mathrm{prompt}}$ gets the lowest accuracy, precision, and F1-score (61.61%, 56.84%, 71.55%). However, $\mathrm{BERT}_{\mathrm{prompt}}$ achieves the highest recall score (96.54%). It can conclude that: (1) Without

[4]https://huggingface.co/bert-base-uncased.

TABLE VI
THE EXPERIMENTAL RESULTS OF BIGRU, $\text{RoBERTa}_{\text{cls}}$, $\text{RoBERTa}_{\text{whole}}$, BERT_{cls}, $\text{BERT}_{\text{whole}}$, AND $\text{BERT}_{\text{prompt}}$ ON THE IMDB DATASET (%). BOLD DATA INDICATES THE HIGHEST SCORES, WHILE * STANDS FOR THE LOWEST SCORES.

	Accuracy	Precision	Recall	F1-score
BiGRU	86.08	88.22	83.28*	85.68
$\text{RoBERTa}_{\text{cls}}$	87.40	85.99	89.36	87.64
$\text{RoBERTa}_{\text{whole}}$	86.60	88.09	84.64	86.33
BERT_{cls}	92.74	92.13	93.47	92.80
$\text{BERT}_{\text{whole}}$	**93.10**	**92.19**	94.18	**93.17**
$\text{BERT}_{\text{prompt}}$	61.61*	56.84*	**96.54**	71.55*

fine-tuning, BERT does not perform as well as the fine-tuned baseline model on the sentiment analysis task. (2) Although the performance of $\text{BERT}_{\text{prompt}}$ is inferior, we can observe that it is still trying to answer the question instead of guessing the answer. Because the numbers of positive and negative reviews are equal, $\text{BERT}_{\text{prompt}}$ can achieve 61.61% accuracy, indicating that it still has some ability to perceive human sentiment on this dataset. (3) $\text{BERT}_{\text{prompt}}$ achieves the highest recall score, but it gets the lowest precision score, which means that $\text{BERT}_{\text{prompt}}$ is easier to classify sentences as positive examples. It demonstrates that without fine-tuning, BERT performs poorly for the negative human sentiment analysis.

B. Experimental results on SMP2019-ECISA

Since BERT achieves good results on the IMDb dataset with apparent sentiment orientations, we increase the difficulty of the experiment. We adopt the SMP2019-ECISA dataset as the more complicated experimental dataset.

The Chinese implicit sentiment analysis task is more complex than the English sentiment analysis task. (1) A token in English can usually express a specific sentiment. In contrast, Chinese requires multiple tokens to form a word to express sentiment. (2) A sentence with implicit sentiment is hard to detect, for it expresses subjective sentiment but lacks explicit sentiment clues [5], [9].

1) The SMP2019-ECISA dataset: It is a Chinese implicit sentiment dataset including positive, negative, and neutral implicit sentiment sentences. The data topics cover many fields from public forums (such as Weibo and Mafengwo), including tourism, smog, etc. The details of the SMP2019-ECISA dataset are shown in Table VII.

2) Experiment set-up: Since BERT adopts WordPiece embeddings [14], it uses tokens as the basic unit when segmenting Chinese tokens. When we let BERT fill the $[MASK]$ slot, it can only output one token. So it is not easy to construct a suitable three-classification prompt template in natural language. Therefore, we split this task into two subtasks. We first let $\text{BERT}_{\text{prompt}}$ classify whether sentences contain positive or negative sentiments. If BERT answers yes, we let it answer whether the sentences carry positive or negative sentiments. Fig. 3 shows one example of the *inputs*, *templates*, and *answers* of $\text{BERT}_{\text{prompt}}$ in the SMP2019-ECISA dataset. For Question 1, if BERT answers *no*, we assign *neutral* to the sentence; otherwise, we ask it Question 2. We use the pre-trained Chinese parameters as the start point of $\text{BERT}_{\text{prompt}}$, BERT_{cls}, and $\text{BERT}_{\text{whole}}$.

The other hyperparameter settings of all models on SMP2019-ECISA are the same as in Table V, except that the learning rate of BERT is 1e-5.

3) Experimental results: From Table VIII, BERT_{cls} achieves the highest macro-P, macro-R, and macro-F1 scores (78.12%, 78.45%, and 78.27%

TABLE VII
DETAIL OF SMP2019-ECISA DATASET.

	Train	Dev	Test
positive	3,799	1,229	919
negative	3,934	1,353	979
neutral	6,956	2,550	1,902

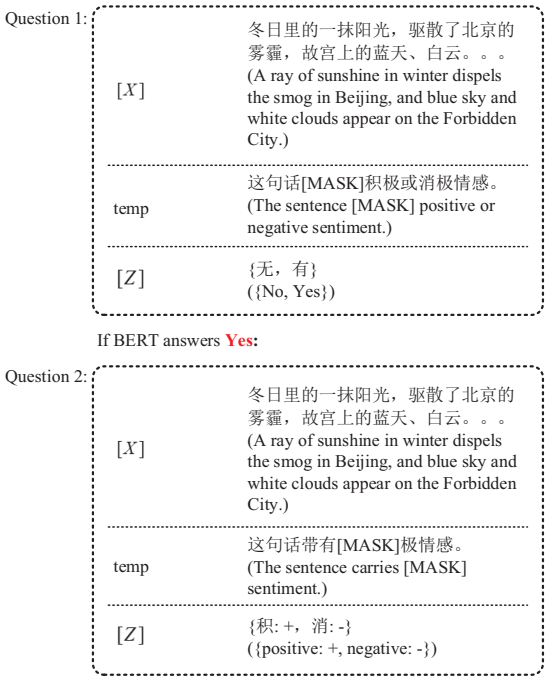

Fig. 3. One example of the *inputs*, *templates* and *answers* of $\text{BERT}_{\text{prompt}}$ in SMP2019-ECISA dataset.

TABLE VIII
THE EXPERIMENTAL RESULTS OF $\mathrm{BERT}_{\mathrm{prompt}}$, $\mathrm{BERT}_{\mathrm{cls}}$, $\mathrm{BERT}_{\mathrm{whole}}$, AND BIGRU ON SMP2019-ECISA DATASET. BOLD DATA INDICATES THE HIGHEST SCORES, WHILE * STANDS FOR THE LOWEST SCORES.

	Macro-P	Macro-R	Macro-F1
BiGRU	68.67	70.40	69.29
$\mathrm{BERT}_{\mathrm{cls}}$	**78.12**	**78.45**	**78.27**
$\mathrm{BERT}_{\mathrm{whole}}$	77.80	78.01	77.89
$\mathrm{BERT}_{\mathrm{prompt}}$	32.92*	44.41*	30.42*

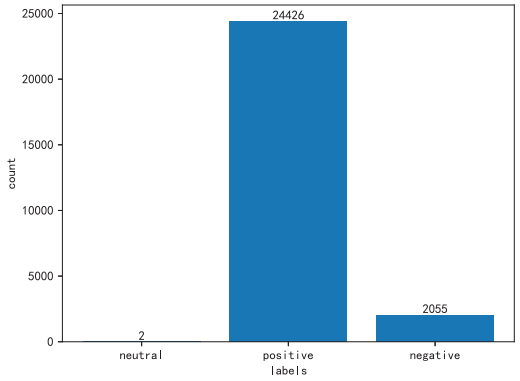

Fig. 4. The number of predicted labels.

respectively), while $\mathrm{BERT}_{\mathrm{prompt}}$ gets the lowest macro-P, macro-R, and macro-F1 scores (32.92%, 44.41%, and 30.42%). Since the SMP2019-ECISA dataset is three-classified, we cannot directly analyze why $\mathrm{BERT}_{\mathrm{prompt}}$ performs so poorly from macro-P, macro-R, and macro-F1 scores. Therefore, on the test set, we count the number of predicted labels[5] output by $\mathrm{BERT}_{\mathrm{prompt}}$, as shown in Fig. 4.

These results show that when $\mathrm{BERT}_{\mathrm{prompt}}$ answers Question 1, it only considers two texts that do not carry positive or negative sentiments. For the remaining 26481 texts, $\mathrm{BERT}_{\mathrm{prompt}}$ thinks that 92.24% of them contain positive sentiment. Therefore, we can conclude that: (1) BERT cannot tell whether a sentence carries positive or negative sentiment. When encountering such questions, it only answers yes. (2) Like the experiments in Table VI, BERT is very kind and seldom thinks that sentences contain negative sentiments. (3) From these results in Table VI and VIII, BERT thinks that most sentences express positive sentiments. It implies that BERT does not understand human sentiment very well.

[5]The reason that the data volume on Fig. 4 mismatches the data volume in Table VII is that there are 22683 obfuscated data, which will not participate in the final evaluation. The website gives the final evaluation results after uploading the file, so we do not know which data is the obfuscated data.

TABLE IX
THE EXPERIMENTAL RESULTS OF $\mathrm{BERT}_{\mathrm{prompt}}$, $\mathrm{BERT}^{1}_{\mathrm{prompt}}$, $\mathrm{BERT}^{2}_{\mathrm{prompt}}$, $\mathrm{BERT}^{1,2}_{\mathrm{prompt}}$, $\mathrm{BERT}_{\mathrm{cls}}$, $\mathrm{BERT}_{\mathrm{whole}}$ AND BIGRU ON THE SMP2019-ECISA DATASET (%). BOLD DATA INDICATES THE HIGHEST SCORES, WHILE THE NUMBER WITH A STAR STANDS FOR THE LOWEST SCORE.

	Macro-P	Macro-R	Macro-F1
BiGRU	68.67	70.40	69.29
$\mathrm{BERT}_{\mathrm{cls}}$	**78.12**	**78.45**	**78.27**
$\mathrm{BERT}_{\mathrm{whole}}$	77.80	78.01	77.89
$\mathrm{BERT}_{\mathrm{prompt}}$	32.92	44.41*	30.42*
$\mathrm{BERT}^{1}_{\mathrm{prompt}}$	69.54	69.47	67.84
$\mathrm{BERT}^{2}_{\mathrm{prompt}}$	31.31*	58.66	40.14
$\mathrm{BERT}^{1,2}_{\mathrm{prompt}}$	69.94	68.17	66.31

4) Experimental results of fine-tuning $\mathrm{BERT}_{\mathrm{prompt}}$*:* Considering that BERT especially likes to output positive sentiment in previous experiments, we use the data in the training set to fine-tune it and observe how it performs.

In the fine-tuning stage, we set three fine-tuning modes:

a) ***Fine-tuning*** $\mathrm{BERT}_{\mathrm{prompt}}$ ***with only Question 1 on SMP2019-ECISA*** *(*$\mathrm{BERT}^{1}_{\mathrm{prompt}}$*):* We only train BERT to answer Question 1 when we fine-tune it. We extract all the sentences with sentiment orientations from the training set and train BERT to answer whether the sentences carry positive or negative sentiments.

b) ***Fine-tuning*** $\mathrm{BERT}_{\mathrm{prompt}}$ ***with only Question 2 on SMP2019-ECISA*** *(*$\mathrm{BERT}^{2}_{\mathrm{prompt}}$*):* We only train BERT to answer Question 2 when we fine-tune it. We extract all the positive and negative sentences from the training set and train BERT to classify positive and negative sentiments.

c) ***Fine-tuning*** $\mathrm{BERT}_{\mathrm{prompt}}$ ***with both Question 1 and Question 2 on SMP2019-ECISA*** *(*$\mathrm{BERT}^{1,2}_{\mathrm{prompt}}$*):* We train BERT to answer Questions 1 and 2. We adopt the strategies of $\mathrm{BERT}^{1}_{\mathrm{prompt}}$ and $\mathrm{BERT}^{2}_{\mathrm{prompt}}$ to train BERT to answer Questions 1 and 2, respectively.

From Table IX, $\mathrm{BERT}^{1}_{\mathrm{prompt}}$, $\mathrm{BERT}^{2}_{\mathrm{prompt}}$ and $\mathrm{BERT}^{2}_{\mathrm{prompt}}$ are 37.42%, 9.72% and 35.89% higher than $\mathrm{BERT}_{\mathrm{prompt}}$ in macro-F1 scores, respectively. It concludes that $\mathrm{BERT}_{\mathrm{prompt}}$ with fine-tuning can significantly improve the model's performance. However, without fine-tuning the BERT classifier and only fine-tuning the parameters in the model, the performance of BERT is still lower than BiGRU. We count the number of predicted labels output by each mode on the test set as shown in Fig. 5.

From these results, we can observe that: (1) when we adopt Question 1 to train BERT, the ability that

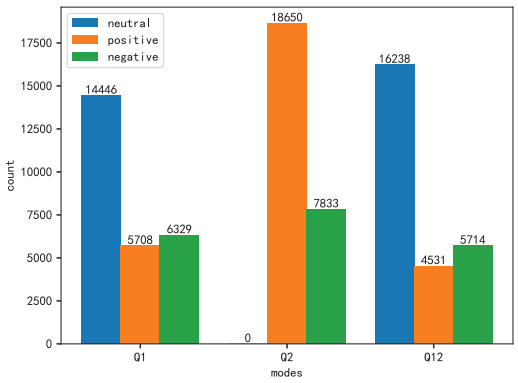

Fig. 5. The number of predicted labels in every mode. Q1, Q2 and Q12 stand for $\text{BERT}^{1}_{\text{prompt}}$, $\text{BERT}^{2}_{\text{prompt}}$ and $\text{BERT}^{1,2}_{\text{prompt}}$, respectively.

BERT identifies whether a sentence carries positive or negative sentiment is significantly improved; and (2) when we only let BERT classify positive and negative sentences, BERT treats each sentence as a positive or negative sentence.

From these observations, we can conclude that, without fine-tuning, the pre-trained BERT cannot understand human sentiment; and it is likelier to perceive sentences carrying positive sentiments.

IV. Related work

A. Time series forecasting models

Time series forecasting aims to extend the forecasting time, it can provide early warning of could-edge servers network crashes and long-term energy consumption planning of cloud server [22]. Time series forecasting models using deep learning can be divided into three categories: (1) RNN-based models. These models use RNN to capture the dependencies at each time step [30]. (2) Attention-based models. These models adopt temporal attention to capture long-term dependencies [31]. And (3) transformer-based models. Since transformer-based language models achieve many SOTA results on natural language processing tasks, many transformer-based time series forecasting models [31] have emerged.

B. Sentiment analysis

Sentiment analysis, also known as opinion mining, aims to mine the sentimental information in texts, speeches, images, and so on [5]. With the continuous research of scholars, sentiment analysis has become more and more challenging, such as implicit sentiment analysis [5], [9] and aspect-level sentiment analysis [1].

C. Pre-trained language models

The PTLMs are expected to obtain rich language knowledge by training them on the large-scale corpus. Since Word2Vec [13] was proposed, many neural network-based word embedding models have emerged. These models train word embeddings on large-scale corpora. These models can only train word embeddings and feed them into the downstream tasks. However, these models do not participate in the training of the downstream tasks. Since Transformer [10] was proposed, the PTLMs have become the fixed architecture. They could fine-tune their model and task-specific parameters in the downstream tasks [11], [15]. These models have a massive amount of parameters, but they can achieve great results by only fine-tuning task-specific parameters in downstream tasks. However, some huge models, such as GPT-3 [18], are almost impossible to fine-tune task-specific models on downstream tasks.

D. Prompt learning

Prompt learning tries to mine the language knowledge inside the PTLMs [6]. It concatenates a prompt template with [MASK] to texts and uses masked language modeling [15] to make the PTLMs complete the cloze. Prompt learning achieves promising results on few- and zero-shot tasks [17]. Furthermore, prompt learning does not require fine-tuning any additional task-specific parameters.

V. Conclusion

This paper evaluated BERT on cloud-edge time series forecasting and sentiment analysis tasks via prompt learning. For cloud-edge time series forecasting tasks, we fed BERT some sequence information of m timestamps and let it predict the features at the next time step. For sentiment analysis, we gave BERT some sentences with sentiments and asked it to answer the sentiments of these sentences. The experimental results showed that: (1) For cloud-edge time series forecasting tasks, BERT with prompt learning could not achieve good results, which means the logical reasoning ability of BERT is not good. (2) BERT thinks almost every sentence carries sentiment for sentiment analysis tasks. BERT is likelier to perceive sentences carrying positive sentiments, meaning BERT cannot understand human sentiments.

In the future, we will continue this research in the following aspects: (1) making BERT possess the logical reasoning ability in the pre-training stage; (2) introducing prompt learning into other cloud-edge systems analysis tasks, such as anomaly detection, diagnosis, and fault detection [33]; (3) making BERT learn sentiment knowledge during pre-training; and (4) finding a new architecture that can better understand human sentiments.

REFERENCES

[1] S. Chen, Y. Wang, J. Liu, et al., "Bidirectional Machine Reading Comprehension for Aspect Sentiment Triplet Extraction," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*. AAAI Press, 2021, pp. 12666-12674.

[2] C. Dai, X. Liu, W. Chen, et al., "A Low-Latency Object Detection Algorithm for the Edge Devices of IoV Systems," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 11169-11178, 2020.

[3] X. Wang, L. T. Yang, L. Ren, et al., "A Tensor-Based Computing and Optimization Model for Intelligent Edge Services," *IEEE Network*, vol. 36, no. 1, pp. 40-44, 2022.

[4] W. Chien, H. Weng, and C. Lai, "Q-learning based collaborative cache allocation in mobile edge computing," *Future Generation Computer Systems*, vol. 102, pp. 603-610, 2020.

[5] J. Liao, M. Wang, X. Chen, et al., "Dynamic commonsense knowledge fused method for Chinese implicit sentiment analysis," *Information Processing and Management*, vol. 59, no. 3, p. 102934, 2022.

[6] P. Liu, W. Yuan, J. Fu, et al., "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing," *CoRR*, 2021.

[7] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, HLT/EMNLP 2005*, 2005, pp. 347-354.

[8] A. L. Maas, R. E. Daly, P. T. Pham, et al., "Learning Word Vectors for Sentiment Analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, ACL 2011*. Association for Computational Linguistics, 2011, pp. 142-150.

[9] Q. Li, X. Li, Y. Du, et al., "ISWR: An Implicit Sentiment Words Recognition Model Based on Sentiment Propagation," in *Natural Language Processing and Chinese Computing - 10th CCF International Conference, NLPCC 2021*, vol. 13029. Springer, 2021, pp. 248-259.

[10] A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention is All you Need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NIPS 2017*, 2017, pp. 5998-6008.

[11] Y. Liu, M. Ott, N. Goyal, et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *CoRR*, 2019.

[12] K. Cho, B. van Merrienboer, C. Gülçehre, et al., "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*. ACL, 2014, pp. 1724-1734.

[13] T. Mikolov, K. Chen, G. Corrado, et al., "Efficient Estimation of Word Representations in Vector Space," in *1st International Conference on Learning Representations, ICLR 2013*, 2013.

[14] Y. Wu, M. Schuster, Z. Chen, et al., "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," *CoRR*, 2016.

[15] J. Devlin, M. Chang, K. Lee, et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*. Association for Computational Linguistics, 2019, pp. 4171-4186.

[16] I. Loshchilov, and F. Hutter, "Decoupled Weight Decay Regularization," in *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net, 2019.

[17] F. Petroni, T. Rocktäschel, S. Riedel, et al., "Language Models as Knowledge Bases?," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019*. Association for Computational Linguistics, 2019, pp. 2463-2473.

[18] T. B. Brown, B. Mann, N. Ryder, et al., "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

[19] T. S. Dillon, C. Wu, and E. Chang, "Cloud Computing: Issues and Challenges," in *24th IEEE International Conference on Advanced Information Networking and Applications, AINA 2010*. IEEE Computer Society, 2010, pp. 27-33.

[20] D. Rosendo, A. Costan, P. Valduriez, et al., "Distributed intelligence on the Edge-to-Cloud Continuum: A systematic literature review," *Journal of Parallel and Distributed Computing*, vol. 166, pp. 71-94, 2022.

[21] K. Hundman, V. Constantinou, C. Laporte, et al., "Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*. ACM, 2018, pp. 387-395.

[22] H. Wu, J. Xu, J. Wang, et al., "Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, 2021, pp. 22419-22430.

[23] S. Huang, Y. Liu, C. J. Fung, et al., "HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2064-2076, 2020.

[24] S. Tuli, G. Casale, and N. R. Jennings, "TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data," *Proceedings of the VLDB Endowment*, vol. 15, no. 6, pp. 1201-1214, 2022.

[25] P. Chen, Y. Xia, S. Pang, et al., "A probabilistic model for performance analysis of cloud infrastructures," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 17, pp. 4784-4796, 2015.

[26] T. Long, P. Chen, Y. Xia, et al., "A Novel Fault-Tolerant Approach to Web Service Composition upon the Edge Computing Environment," in *2021 IEEE International Conference on Web Services, ICWS 2021*, ser. Lecture Notes in Computer Science, vol. 12994. Springer, 2021, pp. 15-31.

[27] P. Chen, Y. Xia, and C. Yu, "A Novel Reinforcement-Learning-Based Approach to Workflow Scheduling Upon Infrastructure-as-a-Service Clouds," *International Journal of Web Services Research*, vol. 18, no. 1, pp. 21-33, 2021.

[28] P. Chen, H. Liu, R. Xin, et al., "Effectively Detecting Operational Anomalies in Large-scale IoT Data Infrastructures by using a GAN-based Predictive Model," *The Computer Journal*, 2022.

[29] Y. Su, Y. Zhao, C. Niu, et al., "Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*. ACM, 2019, pp. 2828-2837.

[30] R. Yu, S. Zheng, A. Anandkumar, et al., "Long-term Forecasting using Tensor-Train RNNs," *CoRR*, 2017.

[31] S. Shih, F. Sun, and H. Lee, "Temporal pattern attention for multivariate time series forecasting," *Machine Learning*, vol. 108, no. 8-9, pp. 1421-1441, 2019.

[32] A. Deng, and B. Hooi, "Graph Neural Network-Based Anomaly Detection in Multivariate Time Series," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*. AAAI Press, 2021, pp. 4027-4035.

[33] X. Li, Y. Du, and Y. Fan, "Fault Tolerance of Optical Hypercube Interconnection Networks with r-Communication Pattern," *Wireless Communications and Mobile Computing*, vol. 2021, p. 5358760, 2021.