IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 9, 2022, accepted March 9, 2022, date of publication March 11, 2022, date of current version March 22, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3158975

Injecting User Identity Into Pretrained Language Models for Document-Level Sentiment Classification

XINLEI CAO[1], JINYANG YU[2], AND YAN ZHUANG[3,4]

[1]School of Computer Science and Technology, East China Normal University, Shanghai 200062, China
[2]Center for ADR Monitoring of Guangdong, Guangzhou 510080, China
[3]National Engineering Laboratory for Medical Big Data Application Technology, Chinese PLA General Hospital, Beijing 100853, China
[4]Medical Innovation Research Division, Medical Big Data Research Center, Chinese PLA General Hospital, Beijing 100853, China

Corresponding author: Xinlei Cao (cyril.xlcao@gmail.com)

This work was supported by the Medical Big Data and AI Research Development Project of the People's Liberation Army General Hospital (PLAGH) (2019MBD-046).

ABSTRACT This paper mainly studies the combination of pre-trained language models and user identity information for document-level sentiment classification. In recent years, pre-trained language models (PLMs) such as BERT have achieved state-of-the-art results on many NLP applications, including document-level sentiment classification. On the other hand, a collection of works introduce additional information such as user identity for better text modeling. However, most of them inject user identity into traditional models, while few studies have been conducted to study the combination of pre-trained language models and user identity for even better performance. To address this issue, in this paper, we propose to unite user identity and PLMs and formulate User-enhanced Pre-trained Language Models (U-PLMs). Specifically, we demonstrate two simple yet effective attempts, i.e. embedding-based and attention-based personalization, which inject user identity into different parts of a pre-trained language model and provide personalization from different perspectives. Experiments in three datasets with two backbone PLMs show that our proposed methods outperform the best state-of-the-art baseline method with an absolute improvement of up to 3%, 2.8%, and 2.2% on accuracy. In addition, our methods encode user identity with plugin modules, which are fully compatible with most auto-encoding pre-trained language models.

INDEX TERMS Representation learning, document-level sentiment classification, personalized sentiment classification, pre-trained language models, attention mechanism.

I. INTRODUCTION

Document-level sentiment classification, as a common subtask of sentiment analysis and text classification, aims to extract the sentiment polarity in a piece of text document (e.g. reviews and tweets) [1].

Representative works of document-level sentiment classification are mainly based on neural networks [2]–[4]. Based on the bag-of-words assumption, typical models usually consist of the following steps. Firstly, they learn word embeddings to represent words as vectors. Secondly, they use CNNs or RNNs to capture dependencies between words. Then they apply attention mechanisms or simply take the vector sum or average over these vectors to constitute a representation of the whole document. Finally, they make predictions based on the text representation. However, these methods are based on static word embeddings such as word2vec [5] and GloVe [6] and cannot model words well.

Recent years have seen the rise of a large number of Pre-trained Language Models (PLMs) based on Transformers [7], such as BERT [8], RoBERTa [9], ALBERT [10], etc. These methods use self-attention and usually deep architectures to capture dependencies between words repeatedly and produce word representations that are well aware of the whole context in the text. After pre-trained on large-scale unlabeled text corpora, one can use a PLM by easily adding some task-specific layers and then fine-tuning the whole model with labeled data. BERT and its alternatives have achieved

The associate editor coordinating the review of this manuscript and approving it for publication was Victor S. Sheng.

state-of-the-art results on many NLP tasks, such as question answering, language inference, and text classification.

However, few studies have been conducted to apply PLMs in the task of personalized sentiment classification where apart from the text itself, its context information such as identities of the user (who is writing it) and the item (what it is describing) is also available and can help generate a better representation of the text. Most methods for this task are traditional CNN/RNN-based methods [11]–[16], whereas CNN and RNN both have their weakness. Therefore, it is a natural thought to ''upgrade'' the backbone models from CNN/RNN to the stronger PLMs. A recent work [17] tried to incorporate context information into PLMs by using historical reviews of user/item to help the prediction, but the target review text is still modeled without considering user or item in their work.

To this end, we propose **U**ser-enhanced **P**re-trained **L**anguage **M**odel**s** (U-PLMs) to take advantages of both PLMs and user data in the process of text modeling. To be specific, we propose two schemes of personalization, which inject user identities into different modules of a pre-trained language model.

The embedding-based personalization takes advantage of the design of PLMs [8]–[10] that they can accept the sum of multiple embeddings as input. We associate the user id of the text with an embedding vector, and then add it to all words at the input. With this document-level global bias, PLMs can encode correlations between words in a personalized way and output representations closer to users' actual intents.

The attention-based personalization is inspired by a common pattern of some traditional methods [14]–[16]. These methods are based on hierarchical RNN/CNNs, which construct a sentence with words in the lower level and then construct a document with sentences in the higher level. To incorporate context information, these methods add embedding vectors of users and items in attention functions of both levels to help select important words in a sentence and important sentences in a document. To transfer this idea into PLMs, we introduce user embeddings in self-attention modules in PLMs. For each text, we add the embedding of its user as a bias of the [CLS] token's query in self-attention, where [CLS] is a special token used to represent the whole text. With these enhanced self-attention modules, PLMs are capable of selecting user-specific important words in each layer and generating more accurate representations of the text.

Our main contributions are as follows:

1) We propose to apply PLMs in the task of personalized sentiment classification for better text representation, making use of both PLMs' modeling ability and user identity information in the process of text modeling.
2) We demonstrate two different attempts to inject user identities into different modules of a pre-trained language model, both of which can effectively improve the performance of sentiment classification.
3) Both of our attempts serve as plugins and are fully compatible with most auto-encoding PLMs such as BERT and RoBERTa.
4) Experiments on three datasets with two backbone PLMs show that our framework obtains remarkable improvements over vanilla PLMs and state-of-the-art models for personalized sentiment classification.

II. RELATED WORK

A. DOCUMENT-LEVEL SENTIMENT CLASSIFICATION

Document-level sentiment classification is a task aiming at inferring the overall sentiment polarity of a given document, which is usually a review that describes how the author thinks of a particular item (movie, goods, restaurant, etc.) or a tweet that expresses the author's mood or opinion at some moment.

Specifically, given a document consisting of multiple sentences, the problem is to determine whether this document conveys a positive/negative opinion or even to what extent the positive/negative sentiment is. In the first case, the problem is defined as a binary classification task, where 0 represents negative and 1 stands for positive. In the latter case, for the case of scores from 1 to 5 stars as an example, 1 and 5 correspond to greatly negative and greatly positive opinions respectively, and 3 means a neutral sentiment. Then the problem is usually defined as a 5-class classification task.

Based on the bag-of-words assumption, a common structure within typical methods for document-level sentiment classification usually consists of four parts: (1) a word embedding layer to represent words, (2) CNNs or RNNs to capture word dependencies, (3) attention or simple pooling functions to gather word-level information and represent the document, and (4) a classifier for sentiment classification.

Quite a few methods are proposed to make improvements to this structure. Xu *et al.* proposes CLSTM [18], i.e. a LSTM [19] with cache mechanism, to capture the overall semantic information in long texts. Reference [3] sees a document from three levels: word, sentence, and document. It applies CNN/LSTM to word embeddings within a sentence to construct the representation of this specific sentence, and then similarly represents the document with multiple sentences. Based on this hierarchical structure, reference [14] further discovers the different importance of words (sentences) within a sentence (document) and introduces attention mechanism into the procedure of sentence (document) composition with words (sentences).

However, these methods are all based on static word embeddings such as word2vec [5] and GloVe [6] which have limited expression ability.

One issue of these embedding approaches is that they only learn a single vector for a word but cannot handle the cases that one word can have different meanings in different contexts (such as the word ''chair'' with meanings of ''seat'' or ''leader''). On the contrary, pre-trained language models, which are pre-trained on a large corpus of unlabeled text, can generate context-aware word representations in the phase of fine-tuning for specific tasks. Therefore, there is a trend towards the adoption of pre-trained language models in place of traditional methods for many NLP tasks including document-level sentiment classification.

Another problem of these static word embeddings is that they learn representations of words according to contexts but ignore their sentiment polarities, which is problematic when applied to sentiment classification. For example, words ''good'' and ''bad'' with similar contexts are mapped into close word vectors in the embedding space, but they have opposite sentiment polarities. To solve this problem, there are some works [20]–[22] focusing on injecting sentiment knowledge into word embeddings for better performance for the task of sentiment classification.

B. PERSONALIZED SENTIMENT CLASSIFICATION

The task of personalized sentiment classification is a subtask of document-level sentiment classification which assumes that user and item ids of the text are also available.

For better performance of document-level sentiment classification, some researchers have conducted studies to introduce user representations and item representations in the analysis of review texts [13]–[16]. The intuition is that these representations can provide global information such as rating and language preferences of users and overall ratings of items [23]. Taking this information into account can help provide better text representations.

Chen *et al.* [14] make improvements over the hierarchical LSTM structure and introduce user and item embeddings into the word-sentence and sentence-document attention functions. Wu *et al.* [15] propose to model the same piece of text from the user's and item's perspective, respectively, and then combine the document representations from these two views together for prediction. Based on that, Yuan *et al.* [16] introduce the memory network [24] for users and items to alleviate the cold-start problem by modeling the inherent correlation between users or items. Specifically, they store representations of representative users or items in memory slots and then use them to infer representations for cold-start users or items. Different from all these methods, Amplayo [13] attempt to represent users and items with their proposed ''chunk-wise'' weight matrices instead of bias vectors and inject them into four locations (i.e. embedding, encoding, attention, classifier) in a model.

Note that there are some works for rating prediction [25], [26] which also take user, item, and text as input and predict the rating scores. However, they assume that the target review text is unknown when testing, which is different from our task. Therefore, we are not considering them for comparison.

C. PRE-TRAINED LANGUAGE MODELS

In recent years, a large number of Pre-trained Language Models (PLMs) based on Transformers [7] have emerged. These models are firstly pre-trained on a large-scale text corpus which is unlabeled and therefore easy to acquire. In this pre-training step, the models can learn universal language representations [27]. After pre-training, a model can then be easily applied to a specific task and fine-tuned together with randomly initialized task-specific parameters using a small learning rate. Since the model has been pre-trained on lots of data before, it can avoid overfitting the small amount of labeled data for specific tasks. Instead, the model converges fast and usually outperforms traditional methods which are only trained on the task-specific labeled data.

Since the Transformer [7] consists of two parts: an encoder and a decoder, there are correspondingly two types of pre-trained language models.

The first type are the autoregressive models. These models predict a word based on the words which precede or succeed it, which is similar to the traditional statistical language models. Therefore, methods of this type are usually used for generative tasks. Representative works are the GPT series [28]–[30]. GPT [28] proposes the stages of generative pre-training and discriminative fine-tuning and is the first work to make use of transformer structure for text modeling. GPT-2 [29] formulates the supervised tasks as unsupervised problems and demonstrates their model's ability to perform a wide range of tasks in a zero-shot setting. GPT-3 [30] removes the stage of fine-tuning. It takes the idea similar to MAML [31], uses two nested structures called ''inner-loop'' and ''outer-loop'' in pre-training, and learns a good initialization point for the model. Starting from this initialization point, when faced with any specific task, the model quickly fits in with the task and converges within only a few samples.

The second type, the autoencoder models, encode token correlations in a bi-directional manner. The encoding way of ''bi-direction'' makes it unavailable for generative tasks but greatly suitable for discriminative tasks such as text classification, sequence labeling, etc. Representative works mainly consist of BERT [8] and its variants [9], [10], [32], [33]. BERT introduces two tasks in the pre-training stage, i.e. Masked LM (MLM) and Next Sentence Prediction (NSP). RoBERTa [9] improves BERT by performing dynamic masking of tokens. ALBERT [10] uses techniques of factorized embedding parameterization and cross-layer parameter sharing to reduce the parameters of BERT. SpanBERT [32] masks contiguous words instead of single tokens and introduces the span boundary objective to extend the BERT's sense of text spans. Similarly, ERNIE [33] proposed by Baidu integrates knowledge into BERT with entity-level and phrase-level masking strategies.

Since we mainly focus on methods for the task of sentiment classification, we are not going to mention the decoder-based methods in later sections. Therefore, we refer to the encoder-based methods (BERT, RoBERTa, etc.) as pre-trained language models or PLMs in this paper for convenience.

III. METHODOLOGY

In this section, we first define the problem of personalized sentiment classification. Then we introduce our proposed U-PLMs which is divided into three modules. Finally, we illustrate the two-stage training procedure of our framework.

Fig. 1 shows an overview of our framework. As shown in the figure, user identity is used in two modules in our work.

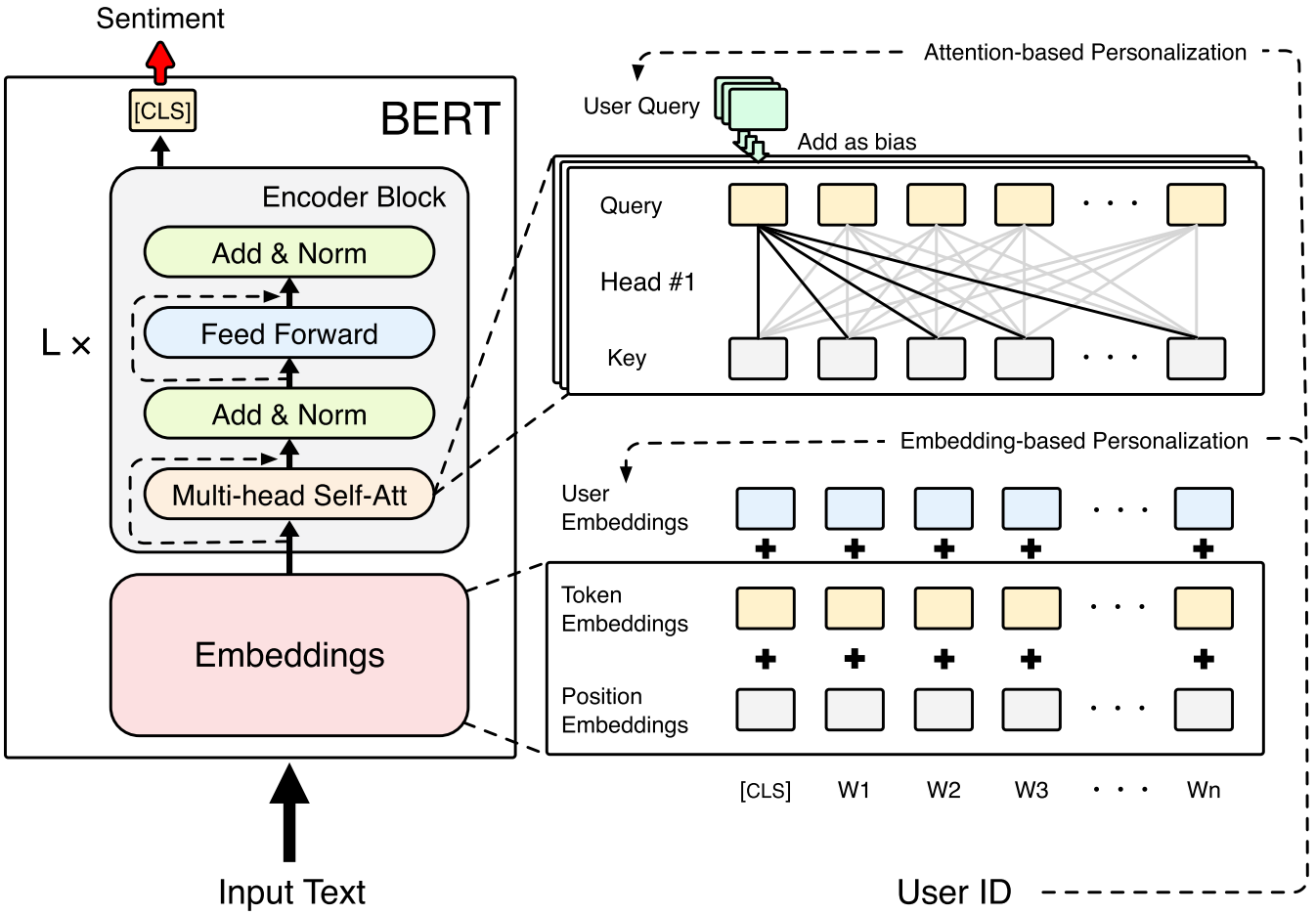

FIGURE 1. An overview of U-PLMs with BERT as the backbone model.

The embedding-based personalization is introduced in the embedding module (section III-C), and the attention-based personalization is introduced in the self-attention module in encoder blocks (section III-D). We use BERT as the backbone model for illustration in this section.

A. PROBLEM DEFINITION

Given a piece of text y written by a user u for an item v, the goal of personalized sentiment classification is to predict the sentiment category r (e.g. 1-5 stars) of the text, which represents the user's opinion on the item.

Note that we're using text y and user u, but not item v, in our framework. This is because we found that user information is much more important and effective for personalization than item information in our experiments, which is consistent with the observation of [11].

B. MODEL INPUT

At the very first step, a piece of natural language text is tokenized into a sequence of tokens/subwords by a subword algorithm (e.g. WordPiece [34] for BERT). The sequence is represented as:

$$\{[CLS], W_1, W_2, W_3, \ldots, W_n\} \tag{1}$$

Note that a special token [CLS] is padded at the beginning of the sequence. This token has no actual meaning itself but is designed to provide sentence-level information in BERT. Its output is often used for sentence-level tasks, including text classification. In our work, we apply BERT to the task of document-level sentiment classification by regarding a document as a long sentence.

C. EMBEDDING MODULE

In this module, BERT accepts the sequence in (1) as input and convert it to a sequence of hidden states:

$$\boldsymbol{E} = \{\boldsymbol{e}_{[CLS]}, \boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3, \ldots, \boldsymbol{e}_n\} \tag{2}$$

Each hidden state is computed via taking summation of token embeddings, position embeddings, and segment embeddings:

$$\boldsymbol{e}_{[CLS]} = \boldsymbol{e}^{token}_{[CLS]} + \boldsymbol{e}^{pos}_0 + \boldsymbol{e}^{seg}_0$$
$$\boldsymbol{e}_1 = \boldsymbol{e}^{token}_{w_1} + \boldsymbol{e}^{pos}_1 + \boldsymbol{e}^{seg}_0$$

$$\vdots$$
$$\boldsymbol{e}_n = \boldsymbol{e}^{token}_{w_n} + \boldsymbol{e}^{pos}_n + \boldsymbol{e}^{seg}_0 \tag{3}$$

Since segment embeddings are used to distinguish tokens in different sentences for sentence pair tasks such as question answering. However, they are not important in single sentence tasks including ours. Based on this reason, we are not mentioning them in Fig. 1 for lack of space.

1) USER INJECTION

To inject user information into the embedding module, we take advantage of the design of PLMs that the sum of multiple embeddings is used as the representation for a token. Specifically, we introduce a new parameter matrix of user embeddings and add the user embedding vector for each text to all tokens as a document-level global bias. That is, equation in (3) is updated as follow:

$$\begin{aligned}
\boldsymbol{e}_{[CLS]} &= \boldsymbol{e}^{token}_{[CLS]} + \boldsymbol{e}^{pos}_0 + \boldsymbol{e}^{seg}_0 + \boldsymbol{e}^{user}_u \\
\boldsymbol{e}_1 &= \boldsymbol{e}^{token}_{w_1} + \boldsymbol{e}^{pos}_1 + \boldsymbol{e}^{seg}_0 + \boldsymbol{e}^{user}_u \\
&\vdots \\
\boldsymbol{e}_n &= \boldsymbol{e}^{token}_{w_n} + \boldsymbol{e}^{pos}_n + \boldsymbol{e}^{seg}_0 + \boldsymbol{e}^{user}_u
\end{aligned} \tag{4}$$

where $\boldsymbol{e}^{user}_u$ represents the embedding vector for user u who wrote the review text.

Note that although both the user embedding and the segment embedding are identically added to all tokens, their importance are different. In our task, the segment embeddings are all the same across all tokens in a document and also across all documents over the whole dataset. So they provide no useful information in our task. However, user embeddings differ from document to document, since the documents are written by different users (there are also ones written by the same user though). This difference helps the PLMs encode token correlations in a personalized way, and output more accurate representations.

D. ENCODER MODULE

In this module, BERT uses the structure of transformer encoder, which is composed of L identical layers of encoder blocks, as shown in Fig. 1. Each encoder block has two parts: a multi-head self-attention mechanism and a fully connected feed-forward network. Each part is surrounded by a residual connection [35] and followed by a layer normalization [36] component.

Each block accepts a sequence as input, update representations with these two parts, and outputs the sequence with the same shape, which is passed into the next block as input. The input of the first block is the output of the embedding module. That is:

$$\begin{aligned}
\boldsymbol{H}_l &= \text{Encoder}^{(l)}(\boldsymbol{H}_{l-1}), \quad l \in [1, L] \\
\boldsymbol{H}_0 &= \boldsymbol{E}
\end{aligned} \tag{5}$$

where $\boldsymbol{H}_l \in \mathbb{R}^{N\times d}$, N is sequence length and d is the dimension of the hidden states.

1) MULTI-HEAD SELF-ATTENTION

The multi-head self-attention module is applied to pass context information between tokens and update token representations iteratively.

Formally, considering the self-attention module in layer l, this process can be explained in the following steps:

(1) Apply three MLP layers to all tokens in $\boldsymbol{H}_{l-1}$ and reshape them to get query $\boldsymbol{Q}_{l,a}$, key $\boldsymbol{K}_{l,a}$ and value $\boldsymbol{V}_{l,a}$ for each attention head a. All of them are in the shape of $\mathbb{R}^{N\times A\times d_a}$, where N is sequence length, A is the number of heads and d_a is the dimension per head.

(2) Calculate similarity between all token pairs by calculating the dot product of $\boldsymbol{Q}$ and $\boldsymbol{K}$:

$$\boldsymbol{S}_{l,a} = \text{softmax}(\frac{\boldsymbol{Q}_{l,a}\boldsymbol{K}^T_{l,a}}{\sqrt{d_a}}) \tag{6}$$

where $\boldsymbol{S}_{l,a} \in \mathbb{R}^{N\times N}$ represents the similarity matrix which is normalized along the dimension of key. Each element $\boldsymbol{S}(i, j)$ represents the importance of the jth token (among all tokens) to the ith token.

(3) Using the similarity matrix as attention weights, calculate the weighted sum of the whole value sequence for each query token separately:

$$\boldsymbol{O}_{l,a} = \boldsymbol{S}_{l,a}\boldsymbol{V}_{l,a} \tag{7}$$

and outputs $\boldsymbol{O}_{l,a} \in \mathbb{R}^{N\times A\times d_a}$.

(4) Finally, for each token x, aggregate the updated representation of all heads together:

$$\boldsymbol{O}^{(x)}_l = W^T_l \,\text{Concat}(\{\boldsymbol{O}^{(x)}_{l,1}, \ldots, \boldsymbol{O}^{(x)}_{l,A}\}) + b_l \tag{8}$$

where $W_l \in \mathbb{R}^{(A\times d_a)\times d}$, $\boldsymbol{O}^{(x)}_l \in \mathbb{R}^d$. Output of the whole sequence is $\boldsymbol{O}_l \in \mathbb{R}^{N\times d}$.

2) FEED FORWARD NETWORK

The part of feed forward network is designed to further increase the model capacity. It is composed of two linear transformations with a activation function between them:

$$\text{FFN}_l(\boldsymbol{x}) = W^T_{l,2} f(W^T_{l,1}\boldsymbol{x} + b_{l,1}) + b_{l,2} \tag{9}$$

where $\boldsymbol{x} \in \mathbb{R}^d$, $W_{l,1} \in \mathbb{R}^{d\times 4d}$ and $W_{l,2} \in \mathbb{R}^{4d\times d}$. f is the activation function, i.e. GeLU [37] in BERT. The function in (9) is applied to $\boldsymbol{O}^{(x)}_l$ for all tokens identically.

3) USER INJECTION

We now propose to inject user information into the encoder module, or its ''multi-head self-attention'' part to be more specific. This design is inspired by the traditional methods for personalized sentiment classification. These methods are mostly based on hierarchical RNN/CNNs with word-level and sentence-level attention functions. To inject additional context information, They incorporate user embeddings and item embeddings as global biases in these attention functions.

Interestingly, we found that there is also a similar pattern in BERT: since the [CLS] token is designed to represent the whole document, the attention with [CLS] as query and all

TABLE 1. Dataset statistics.

Dataset	#classes	#train	#dev	#test	#users	#items	#docs/user (average)	#docs/item (average)	#words/doc (average)	#words/doc (median)
IMDB	10	67,426	8,381	9,112	1,310	1,635	64.82	51.94	394.6	346
Yelp13	5	62,522	7,773	8,671	1,631	1,633	48.42	48.36	189.3	162
Yelp14	5	183,019	22,745	25,399	4,818	4,194	47.97	55.11	196.9	166

tokens as key aims at gathering information of important tokens in the document.

Based on this observation, we propose to add a user vector as bias to the query vector of [CLS]. Then the similarity calculation in self-attention for head a in layer l, i.e. the equation as shown in (6), is now formulated as:

$$\boldsymbol{S}_{l,a} = \text{softmax}(\frac{(\boldsymbol{Q}_{l,a} + \boldsymbol{U}_{l,a})\boldsymbol{K}_{l,a}^T}{\sqrt{d_a}}) \tag{10}$$

where both $\boldsymbol{Q}_{l,a}$ and $\boldsymbol{U}_{l,a}$ are in the shape of $\mathbb{R}^{N\times A\times d_a}$. For $\boldsymbol{U}_{l,a} \in \mathbb{R}^{N\times A\times d_a}$, we only initialize the part which is added to the query of [CLS] (i.e. $\boldsymbol{U}_{l,a}[0, :, :]$) with random values, and fill all others with 0.

E. OUTPUT MODULE

After the stacked encoder blocks, BERT outputs a sequence of hidden states $\boldsymbol{H}_L$ which is then used for specific tasks. The task of sentiment classification is defined as a sentence-level classification problem, which uses the output of the [CLS] token for prediction.

Specifically, it passes the output of [CLS] through a MLP layer and get $\boldsymbol{h} \in \mathbb{R}^d$ to represent the whole text, and then uses a classifier with softmax for classification:

$$\boldsymbol{p} = softmax(W^T\boldsymbol{h} + b) \tag{11}$$

where $\boldsymbol{p} \in \mathbb{R}^{|C|}$ is the vector of class probabilities, in which $|C|$ is number of classes. $W \in \mathbb{R}^{|C|\times d}$ and $b \in \mathbb{R}^{|C|}$.

F. TWO-STAGE TRAINING PROCEDURE

After we load the pre-trained BERT, we train it in two stages successively:

1) In the first stage, we pre-train the model with the Masked LM (MLM) task in the same way as [8], using our train set. This is intended for our framework to learn doma in-specific knowledge in the dataset.
2) In this stage, we introduce user-specific parameters by following one of the two personalization strategies and fine-tune the whole model for sentiment classification with cross-entropy loss.

Note that in our training procedure, user identity is only used in the second stage because we expect our framework to firstly learn general knowledge of the words in the domain in the first stage, and then "fine-tune" their representations with user-specific parameters for the task together.

IV. EXPERIMENTS

We conduct various experiments to evaluate our proposed U-PLMs in this section.

A. EXPERIMENTS SETTINGS

Following prior works, we conduct experiments on three sentiment classification datasets[1] which consist of user and item IDs apart from review texts. These datasets are collected from IMDB and Yelp websites and split into train set, dev set, and test set with a ratio of 8:1:1. Statistical details of the datasets are given in Table 1. We use *Accuracy* on dev set to select the best model, and use *Accuracy* and *RMSE* on test set for evaluation.

For experiments of vanilla BERT and U-PLMs, We pad or clip the text to be with a max length of 500. Following [38], we load the BERT model from BERT-base which contains $L = 12$ encoder blocks. We run the in-domain pre-training with a learning rate of 5e-5 for 100,000 steps. As for fine-tuning, we set the learning rate to 2e-5 and the batch size to 30 to fully leverage the GPU memory. We empirically set the max epoch number to 3. We optimize our model with AdamW [39] and use *slanted triangular learning rates* [40] with a warmup ratio of 0.1 for both in-domain pre-training and fine-tuning.

As for the user-specific parameters in U-PLMs, the user embedding in embedding-based personalization is initialized with a normal distribution $N(0, 0.005^2)$ for better performance. The ones in self-attention modules are initialized with $N(0, 0.02^2)$ and we add a layer normalization [36] component in each module so that the user bias falls into a similar range with the query representation of [CLS].

To show the robustness of our framework, we also use RoBERTa (loaded from roberta-base) as our backbone model, using the same hyper-parameters with BERT. We run our models 10 times and report the average results. All experiments are conducted on an RTX 3090 GPU.

B. BASELINES

We compare our methods with the following baseline methods:

- NSC [14] uses hierarchical LSTM and attention mechanism to encode the review text.
- BERT [8] corresponds to the vanilla BERT model. It is trained with the same setting as our method except that no user information is used.
- RoBERTa [9] denotes the vanilla RoBERTa model.
- NSC + UPA [14] is the enhanced NSC model that incorporates user and item identities into the attention mechanism.

[1] http://ir.hit.edu.cn/~dytang/paper/acl2015/dataset.7z

TABLE 2. Results of our method and baselines on test sets. Acc. (higher is better) and RMSE (lower is better) are used as evaluation metrics. The best performances are in bold and the second best are underlined.

Model Name	Backbone	IMDB		Yelp13		Yelp14	
		Acc.	RMSE	Acc.	RMSE	Acc.	RMSE
Models without user or item identity							
NSC	LSTM	0.443	1.465	0.627	0.701	0.637	0.686
BERT	BERT	0.532	1.125	0.691	0.601	0.688	0.604
RoBERTa	RoBERTa	0.544	1.086	0.702	0.580	0.703	0.577
Models with user and item identities							
NSC + UPA	LSTM	0.533	1.281	0.650	0.692	0.667	0.654
HUAPA	BiLSTM	0.550	1.185	0.683	0.628	0.686	0.626
CHIM	BiLSTM	0.564	1.161	0.678	0.641	0.692	0.622
RRP-UPM	BiLSTM + CNN	0.562	1.174	0.690	0.629	0.691	0.621
IUPC	BERT	0.538	1.151	0.705	0.589	0.712	0.592
Models with user identity (Ours)							
U-PLMs (Emb)	BERT	0.587	1.021	0.716	0.576	0.720	0.569
U-PLMs (Att)	BERT	0.587	1.025	0.714	0.577	0.715	0.575
U-PLMs (Emb)	RoBERTa	0.590	**0.977**	**0.733**	**0.547**	**0.734**	**0.544**
U-PLMs (Att)	RoBERTa	**0.594**	0.980	0.728	0.555	0.729	0.550

- HUAPA [15] uses two separate networks with the same structure to model the text from the view of user and item respectively and combine them for final prediction.
- CHIM [13] studies how and where to incorporate user and item information in a sentiment classification model.
- RRP-UPM [16] is based on HUAPA and considers the inherent correlation between users or items for better representation.
- IUPC [17] is the first method to apply BERT into personalized sentiment classification. It uses historical reviews to represent user and item and combines these two representations with the target review for better prediction.

Since the embedding-based and attention-based personalization are independent of each other, we employ them separately in our model when comparing with baselines to study their effects respectively. The combination of them is discussed in section IV-D.

C. MODEL COMPARISONS

We implement and train BERT, RoBERTa, and our framework, respectively, while using the results of other baselines reported in their papers.

The results are listed in Table. 2. The methods are divided into three different groups according to additional information they use apart from the text: (1) models considering no user or item identity; (2) models considering both user and item identities; and (3) models considering user identities only.

From these results, we can make the following observations:

Firstly, vanilla BERT and RoBERTa perform much better than NSC. This proves the effectiveness of PLMs for document-level sentiment classification.

Secondly, traditional models are improved after using user and item identities. For example, NSC + UPA gains great improvements over NSC. RRP-UPM achieves competitive results with BERT on accuracy. These results show that additional context information truly brings help to the task.

Thirdly, IUPC outperforms other baselines on most metrics. This shows that it's feasible and worthwhile to combine PLMs and context information for better performance.

Finally, our methods outperform all baselines including IUPC on all metrics. This proves that our ways of injecting user identity help both text modeling and final prediction.

It's worth mentioning that we've also tried to incorporate item identity as additional information. Unfortunately, we didn't find an obvious improvement in performance, which is consistent with the observation of [11] that user information is much more effective than item information.

D. ABLATION STUDY

We conduct an ablation study to evaluate the effect of each component. Specifically, we use vanilla BERT and RoBERTa as base models and then add different combinations of three components, i.e. the in-domain pre-training stage, embedding-based personalization, and attention-based personalization, to construct different alternatives of our framework. Results are shown in Table. 3.

From the results, we can observe the positive effects of the in-domain pre-training and both of our user-related components. Firstly, by training the model on our task-specific training data with MLM before fine-tuning, the model is proved to fit in with the dataset better. Based on this, a remarkable improvement is further brought by the injection of user information. Both the embedding-based and attention-based personalization improve the performance, and among them, the embedding-based one is slightly better. Finally, since the two strategies are independent of each other, we've also tried to combine them together in a single model. However, no further improvement is achieved.

E. ANALYSIS FOR EMBEDDING-BASED PERSONALIZATION

As a part of the summation in the embedding module (as shown in (4)), the value range of the user embeddings matters

TABLE 3. Ablation study for three components. PT means the training stage of in-domain pre-training. Emb-U means embedding-based personalization. Att-U means attention-based personalization.

Backbone	PT	Emb-U	Att-U	IMDB		Yelp13		Yelp14	
				Acc.	RMSE	Acc.	RMSE	Acc.	RMSE
BERT				0.524	1.177	0.678	0.613	0.679	0.615
	✓			0.532	1.125	0.691	0.601	0.688	0.604
	✓	✓		0.587	**1.021**	0.716	0.576	**0.720**	**0.569**
	✓		✓	0.587	1.025	0.714	0.577	0.715	0.575
	✓	✓	✓	**0.589**	1.024	**0.719**	**0.568**	0.717	0.571
RoBERTa				0.533	1.113	0.696	0.590	0.697	0.587
	✓			0.544	1.086	0.702	0.580	0.703	0.577
	✓	✓		0.590	**0.977**	**0.733**	**0.547**	**0.734**	**0.544**
	✓		✓	0.594	0.980	0.728	0.555	0.729	0.550
	✓	✓	✓	**0.598**	0.979	0.725	0.555	0.730	0.550

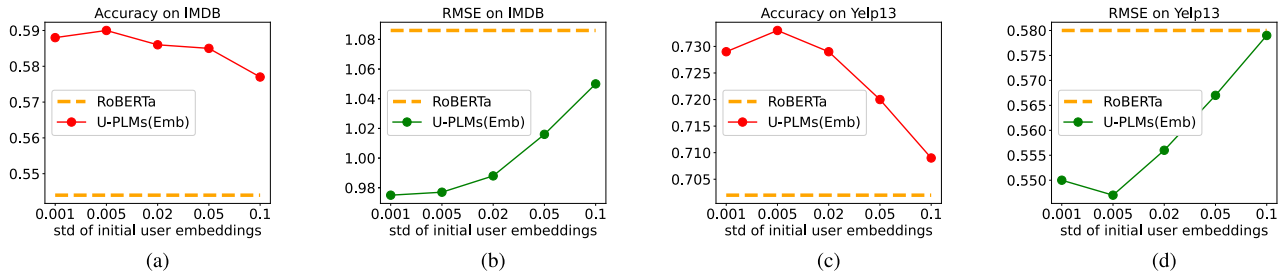

FIGURE 2. Impact of the std of initialized user embeddings on IMDB and Yelp13 datasets with RoBERTa as backbone.

a lot. If it's too large compared to other embedding values, it might be dominant in the token representation and therefore eliminate the difference between tokens in a text. On the other hand, if it's too small, the injected user information may fail to affect the token representation. Therefore, it's necessary to choose an appropriate initialization range for the user embeddings.

1) IMPACT OF USER EMBEDDING INITIALIZATION

The original embeddings (word, pos, segment) in BERT and RoBERTa are initialized with a normal distribution $N(0, 0.02^2)$. Accordingly, we initialize the user embeddings with $N(0, std^2)$, where the standard deviation *std* is chosen from [0.001, 0.005, 0.02, 0.05, 0.1].

The results on IMDB and Yelp13 with RoBERTa as backbone are shown in Fig. 2. Although our model outperforms the vanilla PLM consistently under different initializations, it is observed that a *std* which is the same as or slightly smaller than the original embeddings (i.e. 0.02) is appropriate. When *std* is larger than 0.02, the performance of our method significantly drops. We can also find a trend of drop when *std* turns from 0.005 into 0.001.

F. ANALYSIS FOR ATTENTION-BASED PERSONALIZATION

In the proposed U-PLMs (Att), we inject user vectors into the self-attention modules of all encoder blocks, serving as biases to queries of [CLS] tokens. In this section, we make further analysis and explore the different choices of (1) encoder layers and (2) tokens to be biased.

1) IMPACT OF PESONALIZATION LAYERS

Instead of injecting user vectors into all encoder layers, we only choose some of them for injection in this experiment. Specifically, we divide all 12 layers into four parts, each consisting of three consecutive layers. Then we inject user vectors into only one part to affect this part directly and later parts indirectly. For example, if user vectors are injected into layers 7-9, the text is firstly modeled the same as in vanilla PLMs in layers 1-6. Then the queries of [CLS] in self-attention modules of layers 7-9 are biased with the user. Finally, user information is passed to latter layers implicitly in the biased [CLS] representations.

Results are shown in Table. 4. Overall, comparisons in three datasets show that injecting user vectors into all encoder layers is better than all the alternatives. This is because the introduced user parameters differ from layer to layer. Injection into all layers enables the PLMs to encode text more flexibly in each layer.

TABLE 4. Impact of encoder layers to get biased. U-Layers means layers into which user vector are injected.

Backbone	U-Layers	IMDB		Yelp13		Yelp14	
		Acc.	RMSE	Acc.	RMSE	Acc.	RMSE
BERT	1-3	0.573	1.034	0.710	0.579	0.714	0.577
	4-6	0.575	1.043	0.704	0.585	0.712	0.577
	7-9	0.576	1.042	0.707	0.582	**0.716**	**0.573**
	10-12	0.584	1.028	0.712	0.579	0.715	**0.573**
	All	**0.587**	**1.025**	**0.714**	**0.577**	0.715	0.575
RoBERTa	1-3	0.584	0.995	0.724	0.557	0.728	0.552
	4-6	0.585	0.992	0.726	0.557	0.729	0.550
	7-9	0.592	0.985	0.725	0.557	0.727	0.551
	10-12	0.593	0.987	0.726	**0.554**	**0.731**	**0.549**
	All	**0.594**	**0.980**	**0.728**	0.555	0.729	0.550

TABLE 5. Impact of tokens to get biased. U-Tokens means tokens whose queries are biased by user vectors.

Backbone	U-Tokens	IMDB		Yelp13		Yelp14	
		Acc.	RMSE	Acc.	RMSE	Acc.	RMSE
BERT	None	0.532	1.125	0.691	0.601	0.688	0.604
	[CLS]	**0.587**	**1.025**	**0.714**	**0.577**	**0.715**	**0.575**
	All	0.573	1.055	0.702	0.592	0.703	0.588
RoBERTa	None	0.544	1.086	0.702	0.580	0.703	0.577
	[CLS]	**0.594**	**0.980**	**0.728**	**0.555**	**0.729**	**0.550**
	All	0.586	1.014	0.712	0.577	0.719	0.565

However, some of the alternatives such as ''10-12'' can achieve similar results with ''All''. This means that injection into three layers is enough for the exploitation of user information. There is also a trend that the model tends to perform better when user vectors are injected into latter layers, especially for the IMDB dataset which is more complicated. These observations can help us with the improvement of attention-based personalization in future work.

2) IMPACT OF PESONALIZATION TOKENS

In our proposed strategy, we add user vectors to the query of only the [CLS] token. The intuition behind this design is that the [CLS] token is used to represent the whole document in BERT or RoBERTa, and adding bias to its query in self-attention has a similar pattern with the personalized attention mechanism in traditional methods.

As an alternative, we try to explore this strategy further by adding user vectors to all tokens, instead of the [CLS] token only. Results, as shown in Table. 5, show that the performance of our model drops when we add biases to not only [CLS] but also all other tokens. The reason might be the different roles of [CLS] and normal tokens as queries in self-attention. The [CLS] token has no actual meaning itself and adding bias to its query is to help select important tokens within the document for the user. However, relations between normal tokens are not quite different between users. Therefore, injection into these tokens is not necessary. However, this alternative still outperforms the vanilla model, which proves the importance of user information.

G. ARE ALL USERS FULLY TRAINED?

To study whether users with different numbers of training samples are all well-trained, we compare our framework including U-PLMs (Emb) and U-PLMs (Att) with vanilla PLMs for different groups of users.

We first calculate the metrics for each user based on his/her corresponding test samples. Then we divide all users into several groups according to their numbers of training samples. The metric for a group is obtained by averaging over metrics of all users in this group, and we conduct the comparison between three models for each group separately.

To save space, we only exhibit the results using RoBERTa as the backbone model in this experiment, as shown in Table 6. We can see that both of our methods obtain improvements consistently for users of all groups. This means that our framework, both embedding-based and attention-based, works well for all users with many or a few training samples.

H. COMPARISON BETWEEN EMB-BASED AND ATT-BASED PERSONALIZATION

In this section, we look back to the experiment results again and make further observations on the difference between our two methods, i.e. U-PLMs (Emb) and U-PLMs (Att).

It can be observed from Table 2 that U-PLMs (Emb) performs slightly better than U-PLMs (Att) for both backbone models, except for the IMDB dataset.

We think the reason might be the complexity of user-specific parameters, the difficulty of the task, the amount of introduced user information, and their relationships. Specifically, the only additional parameter introduced in U-PLMs (Emb) is the user embedding matrix in the embedding module. On the other hand, U-PLMs (Att) introduces a new embedding matrix for each encoder block, which is much more complicated. Since we only use user ID as additional context information for user privacy, the embedding-based strategy might be enough to model this information.

However, on the IMDB dataset which consists of more sentiment classes and much longer sentences than the Yelp datasets, U-PLMs (Att) achieves similar or even better performance than U-PLMs (Emb). This might be because the task is more difficult, and to model the user information with the complicated version of U-PLMs is more appropriate.

Similarly, results in Table 6 also support our judgement. On all three datasets and both metrics, U-PLMs (Att) performs worse than U-PLMs (Emb) for users with under 20 training samples. However, with the training samples per user increasing, the gap gradually decreases. U-PLMs (Att) can even perform better in some of the results.

Therefore, we can safely draw a conclusion that our proposed embedding-based personalization is suitable for simple scenarios while the attention-based one is better for complicated cases. Furthermore, the attention-based personalization is more flexible, with many alternatives such as ones in section IV-F1 and IV-F2 can be explored in future work.

I. RESEARCH IMPLICATIONS

We focus on the study of the combination of pre-trained language models and user identity information for document-level sentiment classification, which is shown to be effective and worth further discovering in our experiments.

Firstly, both personalized data and PLMs are available in practice, especially for corporations. In addition, our experiment results show that both of these are quite effective for the task. However, few studies have been done to combined their advantages and further improve the performance.

In our research, we deeply study the ways of combining PLMs and user data. Unlike existing works, we introduce user data into not only the prediction module but also the procedure of text modeling. Experiment results show that these two information can indeed be exploited jointly. We believe that

TABLE 6. User-group-wise results using RoBERTa as the backbone.

#Samples/User	Model	IMDB		Yelp13		Yelp14	
		Acc.	RMSE	Acc.	RMSE	Acc.	RMSE
0-20	RoBERTa	0.502	0.924	0.678	0.463	0.696	0.439
	U-PLMs (Emb)	**0.582**	**0.773**	**0.725**	**0.401**	**0.728**	**0.402**
	U-PLMs (Att)	0.568	0.784	0.705	0.430	0.717	0.413
20-40	RoBERTa	0.499	0.994	0.696	0.477	0.694	0.479
	U-PLMs (Emb)	0.564	**0.859**	**0.731**	**0.433**	**0.730**	**0.430**
	U-PLMs (Att)	**0.567**	0.862	0.723	0.452	0.719	0.444
40-60	RoBERTa	0.543	0.934	0.709	0.503	0.698	0.533
	U-PLMs (Emb)	0.598	0.807	0.747	0.464	**0.728**	**0.486**
	U-PLMs (Att)	**0.606**	**0.792**	**0.748**	**0.457**	0.724	0.498
60+	RoBERTa	0.544	0.907	0.700	0.558	0.709	0.537
	U-PLMs (Emb)	**0.588**	0.804	**0.733**	0.519	**0.739**	0.507
	U-PLMs (Att)	0.587	**0.794**	0.732	**0.515**	0.737	**0.505**

there are many better ways which deserve discovering in this field.

V. CONCLUSION

In this paper, we propose two attempts to inject user identity into PLMs to build U-PLMs, i.e. user-enhanced pre-trained language models, for personalized sentiment analysis. Experimental results show that both of our two methods outperform vanilla pre-trained models and state-of-the-art models for personalized sentiment classification greatly. These observations further indicates that pre-trained language models and personalized data can be exploited jointly for better performance in the task of document-level sentiment classification. Furthermore, we found that the embedding-based personalization is enough to model the user id which contains not much information, while the attention-based strategy is suitable for more complicated situations. In the future work, we will try to make improvements based on our two strategies and explore other better ways of injecting user identity and other context information of the text into PLMs.

VI. ETHIC CONSIDERATIONS

In this paper, we use a unique ID, as the only information to represent a user. The ID is in the form of a meaningless string (e.g. "U0001"), and contains no real information (gender, race, etc.) about users.

To ensure acceptable privacy practice, all the datasets we use in this paper are publicly available, and we use them in a purely observational and non-intrusive manner.

REFERENCES

[1] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. A. Chenaghlu, and J. Gao, "Deep learning–based text classification," *ACM Comput. Surv.*, vol. 54, no. 3, pp. 1–40, Apr. 2021.

[2] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP*, Doha, Qatar, 2014, pp. 1746–1751.

[3] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Lisbon, Portugal, 2015, pp. 1422–1432.

[4] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, San Diego, CA, USA, 2016, pp. 1480–1489.

[5] J. Bhatta, D. Shrestha, S. Nepal, S. Pandey, and S. Koirala, "Efficient estimation of nepali word representations in vector space," *J. Innov. Eng. Educ.*, vol. 3, no. 1, pp. 71–77, Mar. 2020.

[6] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. EMNLP*, Doha, Qatar, 2014, pp. 1532–1543.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, Long Beach, CA, USA, 2017, pp. 5998–6008.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, Minneapolis, MN, USA, 2019, pp. 4171–4186.

[9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[10] P.-H. Chi, P.-H. Chung, T.-H. Wu, C.-C. Hsieh, Y.-H. Chen, S.-W. Li, and H.-Y. Lee, "Audio albert: A lite bert for self-supervised learning of audio representation," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Addis Ababa, Ethiopia, Jan. 2021, pp. 344–350.

[11] D. Tang, B. Qin, and T. Liu, "Learning semantic representations of users and products for document level sentiment classification," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, Beijing, China, 2015, pp. 1014–1023.

[12] R. K. Amplayo, J. Kim, S. Sung, and S.-W. Hwang, "Cold-start aware user and product attention for sentiment classification," in *Proc. ACL*, Melbourne, VIC, Australia, 2018, pp. 2535–2544.

[13] R. K. Amplayo, "Rethinking attribute representation and injection for sentiment classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, Hong Kong, 2019, pp. 5602–5613.

[14] H. Chen, M. Sun, C. Tu, Y. Lin, and Z. Liu, "Neural sentiment classification with user and product attention," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Austin,TX, USA, 2016, pp. 1650–1659.

[15] Z. Wu, X. Dai, C. Yin, S. Huang, and J. Chen, "Improving review representations with user attention and product attention for sentiment classification," in *Proc. AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, vol. 2018, pp. 5989–5996.

[16] Z. Yuan, F. Wu, J. Liu, C. Wu, Y. Huang, and X. Xie, "Neural review rating prediction with user and product memory," in *Proc. CIKM*, Beijing, China, 2019, pp. 2341–2344.

[17] C. Lyu, J. Foster, and Y. Graham, "Improving document-level sentiment analysis with user and product context," in *Proc. COLING*, Barcelona, Spain, 2020, pp. 6724–6729.

[18] J. Xu, D. Chen, X. Qiu, and X. Huang, "Cached long short-term memory neural networks for document-level sentiment classification," in *Proc. EMNLP*, Austin, TX, USA, 2016, pp. 1660–1669.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[20] D. Tang, F. Wei, B. Qin, N. Yang, T. Liu, and M. Zhou, "Sentiment embeddings with applications to sentiment analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 2, pp. 496–509, Feb. 2016.

[21] P. Fu, Z. Lin, F. Yuan, W. Wang, and D. Meng, "Learning sentiment-specific word embedding via global sentiment representation," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 4808–4815.

[22] L.-C. Yu, J. Wang, K. R. Lai, and X. Zhang, "Refining word embeddings using intensity scores for sentiment analysis," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 3, pp. 671–681, Mar. 2018.
[23] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis : A survey," 2018, *arXiv:1801.07883*.
[24] J. Weston, S. Chopra, and A. Bordes, "Memory networks," 2014, *arXiv:1410.3916*.
[25] W. Zhang, Q. Yuan, J. Han, and J. Wang, "Collaborative multi-level embedding learning from reviews for rating prediction," in *Proc. IJCAI*, New York, NY, USA, 2016, pp. 2986–2992.
[26] W. Zhang and J. Wang, "Integrating topic and latent factors for scalable personalized review-based rating prediction," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 11, pp. 3013–3027, Nov. 2016.
[27] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Sci. China Technol. Sci.*, vol. 63, no. 10, pp. 1872–1897, 2020.
[28] A. Radford and K. Narasimhan. (2018). *Improving Language Understanding by Generative Pre-Training*. [Online]. Available: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/langua% ge-unsupervised/language_understanding_paper.pdf
[29] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. (2019). *Language Models are Unsupervised Multitask Learners*. [Online]. Available: https://d4mucfpksywv.cloudfront.net/better-language-models/language_mod% els_are_unsupervised_multitask_learners.pdf
[30] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, and S. Agarwal, "Language models are few-shot learners," 2020, *arXiv:2005.14165*.
[31] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," 2017, *arXiv:1703.03400*.
[32] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "SpanBERT: Improving pre-training by representing and predicting spans," 2019, *arXiv:1907.10529*.
[33] Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu, "ERNIE: Enhanced representation through knowledge integration," 2019, *arXiv:1904.09223*.
[34] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, and J. Klingner, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*.
[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Las Vegas, NV, USA, 2016, pp. 770–778.
[36] J. Lei Ba, J. Ryan Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
[37] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.
[38] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune BERT for text classification?" 2019, *arXiv:1905.05583*.
[39] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.
[40] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. ACL*, Melbourne, VIC, Australia, 2018, pp. 328–339.

XINLEI CAO received the B.S. degree in software engineering from East China Normal University, China, in 2019, where he is currently pursuing the master's degree with the Department of Computer Science and Technology. His research interests include natural language processing, pre-trained language models, and information retrieval.

JINYANG YU is a Ph.D. in Medicine. He is engaged in post marketing drug monitoring and evaluation. He has presided over or been responsible for a number of post marketing drug research projects. Currently, he is mainly responsible for the medical devices monitoring and information construction.

YAN ZHUANG received the Ph.D. degree from Tsinghua University, China, in 2019. He is a Senior Engineer with the Medical Big Data Research Center, Chinese People's Liberation Army (PLA) General Hospital, engaged in hospital informatization construction and research for more than ten years, specializing in medical big data and artificial intelligence related technologies, and doing research in database and data warehouse, human–machine computing, knowledge graph, and graph neural networks. He has published more than 90 papers in the field of medical big data and won the Best Paper Award at the 2017 ACM International Conference on Information and Knowledge Management (CIKM2017).