Neural Processing Letters (2023) 55:1847–1867
https://doi.org/10.1007/s11063-022-10966-8

Transformer-Encoder-GRU (T-E-GRU) for Chinese Sentiment Analysis on Chinese Comment Text

Binlong Zhang[1] · **Wei Zhou**[1]

Accepted: 5 July 2022 / Published online: 12 July 2022

Abstract
Chinese sentiment analysis (CSA) has always been one of the challenges in natural language processing due to its complexity and uncertainty. Transformer has been successfully utilized in the understanding of semantics. However, it captures the sequence features in the text through position encoding, which is naturally insufficient compared with the recurrent model. To address this problem, we propose T-E-GRU. T-E-GRU combines the powerful global feature extraction of Transformer encoder and the natural sequence feature extraction of GRU for CSA. The experimental evaluations are conducted on three real Chinese datasets, the experimental results show that T-E-GRU has unique advantages over recurrent model, recurrent model with attention and BERT-based model.

Keywords Chinese comments · Chinese sentiment analysis · Recurrent model · Transformer-Encoder · T-E-GRU

1 Introduction

Sentiment analysis has become one of the most active fields in natural language processing (NLP) [14, 45]. With the development of sentiment analysis, the research is not only limited to the computer science, but also expanded to management, social science, etc. Its potential commercial value has also attracted the attention of all sectors of society. With the development of GPU, sentiment analysis methods based on machine learning and deep learning gradually occupy the dominant position and continue to achieve the state-of-the-art performance.

The earliest CSA is based on sentiment dictionary, which judges the sentiment tendency of the text through sentiment dictionary. But none of them can capture sequence feature. Recurrent neural network (RNN) proposed by Jeffrey L Elman [20] is one of the effective methods to alleviate the defect, but it was accompanying by vanishing/exploding gradient

✉ Wei Zhou
mczhouwei12@gmail.com

Binlong Zhang
binlong_zhang@163.com

[1] Northwest University, Xi'an, China

problem [3]. Subsequently, a series of recurrent models based on RNN were proposed to address the vanishing/exploding gradient problem, such as long short-term memory (LSTM) [17], gate recurrent neural network (GRU) [9], etc. However, the vanishing/exploding gradient problem on long sequence is still one of its most important limitations. To address the vanishing/exploding gradient, Bahdanau et al. [1] proposed the attention mechanism, which assigns weights to input sequence to make the model pay more attention to keywords. After that, the models with attention continue to achieve state-of-the-art performance. In particular, in 2017, Vaswani et al. [40] proposed scaled dot-product attention which has become one of the effective ways to get attention weights and transformer model. Transformer and its variants have achieved the state-of-the-art performance on basic problems in many fields [10, 40]. Transformer-Encoder realizes the extraction of sequence feature through position encoding (PE), which still has a gap with the natural sequence feature extractor such as RNN.

Although sentiment analysis has made remarkable achievements in recent years, most of them are based on English. Compared with English, Chinese natural language processing is more challenging. On the one hand, Chinese has more complicated vocabulary and semantics. On the other hand, Chinese text semantics are more dependent on context. Inspired by the powerful global feature extraction ability of transformer and the powerful sequence feature extraction ability of recurrent model, we proposed Transformer-Encoder-GRU (T-E-GRU) which combine the GRU with the structure of transformer, and use it for CSA. We compare it with recurrent models, recurrent models with attention, Bert-based models, etc. The results show that our model has unique advantages.

The main contributions of this paper are summarized as following:

1. We proposed a novel model called T-E-GRU based on GRU and transformer encoder, which is exactly suitable for CSA.
2. Focus on the complexity and confusion of punctuation in Chinese comments, we selectively retain some punctuation for clauses.
3. Extensive experiment show that T-E-GRU has unique advantages against CSA.

2 Related Work

2.1 Chinese Text To Sequence

In Chinese natural language processing, Chinese word segmentation (CWS) is a basic and important task, which segments text into independent word units. Compared with space (natural delimiter) in English, Chinese has no formal delimiters and no strict division method, and even sometimes the division method depends on the context. The main disadvantages of dictionary-based word segmentation algorithms such as forward maximum matching (FMM), backward maximum matching (BMM), and bi-direction matching (BiMM) are slow in matching and cumbersome implementation of supplementing unrecorded words [19]. Based on statistical machine learning algorithms, the main idea is that the more adjacent characters appear in the context at the same time, the more likely they are to form a word. The current main models are: N-gram model (N-gram) [5], hidden Markov model (HMM) [34], condition random field (CRF) [21], RNN variant model [7, 43]. Various open CWS libraries such as Jieba, FoolNLTK, LTP, etc. have performed well in CWS. Although some people have raised doubts about the necessity of CWS in deep learning [24], most of the state-of-the-art models are based on CWS.

The simplest way to represent words is one-hot encoding which brings curse of dimensionality, semantic gap and poor scalability. Fighting the curse of dimensionality, Bengio Y et al. proposed to represent vocabulary with low-dimensional vectors [2]. Word Embedding is a by-product of their model. In 2013, Tomas Mikolov proposed word2Vec [29], which is a faster and better method for training word embedding model. There are two algorithms: skip-grams and continuous bag-of-words (CBOW) [30], both of which based on N-gram model. The N-gram model assumes that a word is only related to N surrounding words. This assumption determines that the disadvantage is insufficient use of global information. Jeffrey Pennington et al. proposed global vectors for Word representation (GloVe) which considers both global and local information [32]. However, the word representation mentioned above is a static model, which cannot cope with the situation where a word has multiple meanings. In 2018, Peters M E et al. proposed embeddings from language model (ELMO) [33] by using bidirectional LSTM. Since transformer has been proved to have powerful feature extraction ability in many researches [40], generative pre-training (GPT) was proposed [35]. It replaces LSTM in ELMO with transformer. Although GPT uses a unidirectional language model, it achieves the best results in 9 of the 12 tasks in NLP. In 2019, Devlin et al. proposed bidirectional encoder representations from transformers (BERT) [10] which replace unidirectional with bidirectional language models, and also combined with the tricks of CBOW. BERT, as synthesis of word representation model in recent years, achieved state-of-the-art performances in multiple NLP basic tasks.

2.2 Recurrent Models

Recurrent model is one of the effective methods for processing sequence input. In 1990, neuroscientist Jeffrey L Elman proposed RNN which is based on the Jordan Network [20] back-propagation (BP) [36]. However, vanishing/exploding gradient problem on long sequence comes with RNN [3]. In 1997, Hochreiter et al. proposed LSTM, which can alleviate this problem by introducing delicate gate mechanism [17]. However, LSTM often has overfitting on some tasks due to the complexity of its structure. Cho et al. simplified the gate of LSTM and proposed GRU [9]. GRU has similar performance as LSTM but with fewer parameters. RNN, LSTM and GRU have become the most important recurrent models, but none of them can completely completely overcome the vanishing/exploding gradient. Up to now, methods to improve RNN from different perspectives are still proposed one after another [27].

In addition to improving the RNN architecture, some researchers address the vanishing/exploding gradient from other perspectives. Penalty/Lagrangian methods [16, 48] lift recursive equality constraints into the object function. Zhang et al. proposed using stochastic bi-level optimization to interpret RNNs and train them more stable [49]. Liu et al. proposed to apply gradient activation functions on gradients to address deep neural network training failures include the vanishing/exploding gradient [26].

2.3 Attention and Transformer

In 2014, Mnih V et al. applied attention mechanism on RNN to solve image classification tasks [31]. Then, attention was used as a way to solve the information overload problem in NLP tasks. Bahdanau et al. extended the basic seq2seq model with attention, and it yields good results on longer sentences [1]. Then Cheng proposed intra-attention [8], which broke the limitation of attention mechanism used in seq2seq. Many researches [1, 6, 8, 31] show

that attention is effective in long sequence information loss. Vaswani et al. completely abandoned RNN and CNN to build transformer which is entirely based on the fully-connected layer and attention mechanism [40]. Subsequently, transformer and attention mechanism gradually showed their power in deep learning. Dosovitskiy et al. has achieved state-of-the-art performance on multiple image recognition benchmarks with transformer [12]. Zhou et al. also showed the potential of transformer in video understanding [51]. Yu et al. proposed Modular Co-Attention Network which combine self-attention and guided-attention achieved state-of-the-art performance in visual question answering [44]. In 2020, Zhou H et al. proposed probsparse self-attention mechanism and informer, and this model achieved high prediction capacity in the long sequence time-series forecasting [50]. Transformer is divided into two parts: encoding and decoding. Transformer-encoder uses position encoding to make the model understand sequence feature. T-E-GRU replaces the position encoding with GRU. So T-E-GRU has the advantages of transformer-encoder and recurrent model, which will be more suitable for CSA.

2.4 Chinese Sentiment Analysis

Chinese sentiment analysis is very challenging due to the complexity of Chinese. A plain method is based on the n-gram model and syntax tree, using some classic machine learning classification models such as naive Bayesian (NB), support vector machine (SVM), etc. for CWS [53]. The performance of their models were unsatisfactory. Ding et al. [11] also proposed a method of matching sentiment words in specific field for CSA. J Liang et al. [25] combined polarity shifting and LSTM for CSA. Xiao Z [42] proposed bidirectional LSTM with word embedding for CSA. Their model already has high accuracy in CSA. As the attention mechanism and transformer were proposed, a series of models were for CSA. Bin et al. [4] proposed multi-attention convolutional neural network (CNN) for aspect-based sentiment analysis. In 2019, Shi, Xiaoming et al. [37] proposed attention-based bidirectional hierarchical LSTM networks. In 2021, Wang, Shitao et al. [41] proposed bigru-multi-head self-attention network for CSA. In addition, many state-of-the-art models based on transformer or BERT have been proposed for CSA [15, 38, 52].

3 T-E-GRU

At present, the state-of-the-art models for NLP are mainly based on transformer or recurrent model. In general, transformer is powerful in global feature capture and is very suitable for CSA. The natural sequence structure of recurrent model is well suited for capturing sequence feature. In particular, GRU has become one of the best recurrent models in recent years due to its cheapness and efficiency.

Inspired by the powerful global feature extraction of transformer and the natural sequence feature extraction of GRU, we proposed T-E-GRU for CSA. T-E-GRU combines transformer-encoder and GRU. Compared with recurrent model, the attention mechanism and residual connection of the transformer-encdoer in T-E-GRU can further reduce the information loss of long sequences. Different from transformer, T-E-GRU uses GRU to extract sequence feature instead of position encoding, which can better deal with the problem that text sentiment depends on word order. In addition, compared to recurrent model with attention, T-E-GRU combines the transformer structure achieved better results in CSA.

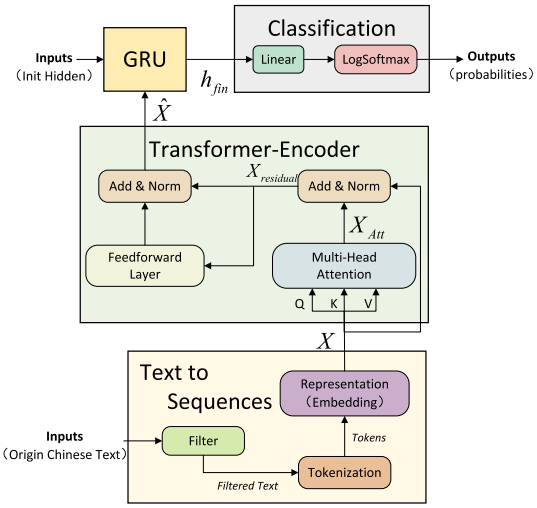


Fig. 1 Architecture of T-E-GRU, shown here based on transformer-encoder [40] and GRU [9]. The origin Chinese text is processed through the four parts of Text to Sequences, Transformer-Encoder, GRU, Classification in turn to obtain the sentiment probability. Text to Sequence converts input into vector sequence X by Filter, Tokenization and Representation. Then, transformer-encoder and GRU capture the global and sequence features of X, respectively. Finally, Classification takes h_{fin} from T-E-GRU as input and output sentiment probability

As shown in Fig. 1, our framework consists of four parts: text to sequence, transformer-encoder, GRU and classification. Text to sequence converts original Chinese text into vector sequence. Transformer-encoder reprocesses the original vector sequence so that each processed vector is determined by the entire input vector sequence. Then, GRU with the advantage of recurrent model is used to capture sequence feature. Due to the reprocessing in the transformer-encoder, processed vector has global information, which can better against long sequence information loss that is difficult to be overcome by recurrent model. Finally, the final state of the GRU is adopted as the input of the classification to output the predicted probability.

3.1 Text To Sequence

Text to sequence maps the original Chinese text to vector sequence $X = (x_1, x_2, \cdots, x_n)$, where $x_i \in \mathbb{R}^{d_R}$. n is the number of tokens and d_R is the dimension of word representation. Text to sequence mainly consists of three parts: filter, tokenization, representation. Filter is used to deal with irregular punctuation, characters, etc. in the original Chinese text. After filtering, the tokens generated by tokenization will be shorter than before and will be fewer meaningless tokens. Specifically, the filter selectively removes most of the punctuation that have no ability to clause. Then tokenization convert the filtered text into a series of tokens. Here, any suitable CWS library can be used, and we use Jieba to finish CWS. Next, we use vector to represent these tokens. Word embedding is one of the important representation methods. Representation takes these tokens as input and vector sequence $X = (x_1, x_2, \cdots, x_n)$ as output. After the above processes, text to sequence converts the original Chinese text into vector sequence X.

3.2 Transformer-Encoder

Transformer-encoder takes X from text to sequence as input and generates output $\hat{X} = (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n)$. As shown in Fig. 1, transform-encoder is mainly divided into multi-head self-attention and feedforward network.

Attention mechanism has become one of the most effective methods to capture global information [18, 39, 40]. Multi-head attention is based on scaled dot-product attention proposed by Vaswani et al. [40]. Multi-head attention is shown in the following equations:

$$\begin{aligned} MultiHead(Q, K, V) &= concat(Att_1, Att_2, \ldots, Att_n) \\ where\ \ Att_i &= softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \end{aligned} \tag{1}$$

where Q, K, V represent query, keys and values respectively which they are all input matrices, d_k represents the dimension of the keys, n equals to the number of heads, and we set $n = 2$ in our model. Especially when Q, K, V are the same, it is called self-attention. Here we use the output X from text to sequence as Q, K, V, and then output X_{Att}. Then we use residual connection and feedforward network in the following:

$$\begin{aligned} X_{Att} &= MultiHead(X, X, X) \\ X_{residual} &= norm(X_{Att} + X) \\ \hat{X} &= norm(X_{residual} + \mathrm{FFN}(X_{residual})) \end{aligned} \tag{2}$$

where *norm* is normalization layer, FFN consists of two linear transformations with a ReLU:

$$\mathrm{FFN}(X_{residual}) = Linear(\max(0, Linear(X_{residual}))) \tag{3}$$

where the dimension of inner-layer is 2048 and other details of transformer-encoder are mentioned in [40].

Finally, transformer-encoder transfers $X = (x_1, x_2, \cdots, x_n)$ to $\hat{X} = (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n)$, where $\hat{x}_i$ is determined by the entire input X.

3.3 GRU

GRU is one of the variants of RNN, which improved from the LSTM. It performs similarly to LSTM, but is computationally cheaper. As shown in Fig. 2. For any input state x_t and the hidden state h_{t-1} generated by the previous sequence:

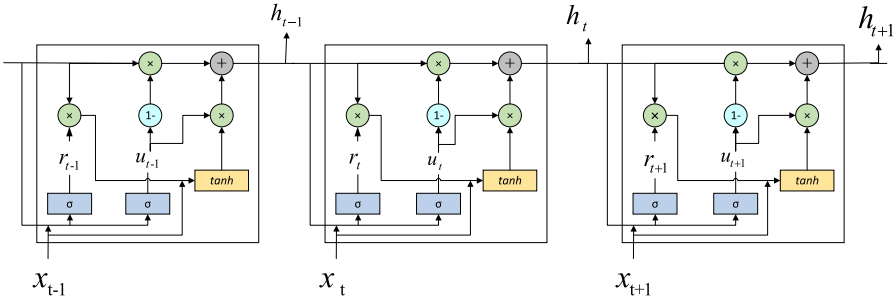


Fig. 2 An illustration of GRU proposed in [9].The symbol σ in the figure is the sigmoid function, and *tanh* represents the hyperbolic tangent function. For a GRU unit, it takes x_t, h_{t-1} as input to generate the output h_t. See Eq.(4) for more details

$$
\begin{aligned}
u_t &= sigmoid(x_t W_z + h_{t-1} U_z) \\
r_t &= sigmoid(x_t W_r + h_{t-1} U_r) \\
h_t &= (1 - u_t) h_{t-1} + u_t * tanh(x_t W_h + (h_{t-1} \cdot r_t) U_h)
\end{aligned} \tag{4}
$$

where r_t is the reset gate, u_t is the update gate, $W_{z,r,h}$ and $U_{z,r,h}$ are the weight matrices that need to be trained.

The reset gate determines how much previous sequence information h_{t-1} is combined with the current input x_t. The update gate balances the combined information $tanh(x_t W_h + (h_{t-1} \cdot r_t) U_h)$ and previous sequence information h_{t-1} to generate the output h_t. In this way, h_t selectively extracts all the information of the sequence $(x_1, x_2, \cdots, x_t)$.

3.4 T-E-GRU

T-E-GRU combines Transformer-Encoder and GRU to classify the sequence X generated from text to sequence. Firstly, we generate a sequence $\hat{X} = (\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n)$ with global information through the two heads Transformer-Encoder by Eq.(2). Then we classify the sequence $(\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n)$ according to the following equations:

$$
\begin{aligned}
h_n &= GRU(\hat{X}, h_0) \;\; where \;\; h_0 = 0 \\
outputs &= Logsoftmax(Linear(h_n))
\end{aligned} \tag{5}
$$

Here we take $\hat{X}$ as input of GRU to generate the final state h_n as the output of GRU. The details of GRU are shown in Sect. 3.3. The classification here is simply composed of a linear layer and logsoftmax for CSA. The complete structure is shown in Fig. 1.

In addition, other recurrent models (RNN, LSTM) can also replace the GRU to capture sequence information. The naive RNN has limited ability to capture sequence feature (without dealing with the vanishing/exploding gradient), LSTM is computationally expensive and prone to overfitting for CSA, so we choose GRU here. Further, we demonstrate the effectiveness of GRU in ablation study in next section, see Sect. 4.4.1 for more details.

4 Experimental

In this section, We compared our model with a series of models (Including various recurrent models, recurrent models with attention and BERT-based models) on three datasets for CSA, including DMCS_V2 (Sect. 4.1), YF_DianPing (Sect. 4.2) and Online_shopping_10 cats (Sect. 4.3) datasets. Then, we also conduct ablation study to further reveal the performance of T-E-GRU. All our experiments are run on RTX 3080 with 8704 cuda cores, 10GB memory.

4.1 DMCS_V2

Dataset. Douban movie short comments dataset V2 (DMSC_V2[1]) comes from the Kaggle competition. It collects user reviews, ratings, and likes of movies on Douban (a platform for Chinese film lovers). This dataset collected short user reviews of 28 popular movies, with a total of more than 2 million comments. We label 1, 2-star reviews as "negative reviews" and 4,5-star reviews as "positive reviews" (3-star reviews are ignored because of its vague

[1] https://www.kaggle.com/utmhikari/doubanmovieshortcomments.

Table 1 Details of text to sequences. The table shows the alignment length of the input sequence and the number of words in the vocabulary, where the numbers in parentheses indicate the degree of coverage of the training set.The upper and lower parts show the text-to-sequences details of BERT-based model and other models, respectively

	DMCS_V2	YF_DianPing	Online_shopping_10 cats
Align (BERT)	140 (98.1%)	500 (93.7%)	250 (97.8%)
Vocabs (BERT)	21,028 (93.86%)	21,028 (99.6%)	21,028 (99.6%)
Align (Others)	100 (99.8%)	400 (96.2%)	200 (99.0%)
Vocabs (Others)	200,000 (94.0%)	150,000 (86.0%)	200,000 (94.9%)

Table 2 Hyperparameters for models. In the Table 2(a), Recurrent includes RNN, LSTM, GRU. Bi-Recurrent includes Bi-RNN, Bi-LSTM, Bi-GRU. '-' means no such setting or default. In the Table 2(b), Recurrent includes RNN, LSTM, GRU, Bi-RNN, Bi-LSTM, Bi-GRU. NLL represents the negative log-likelihood loss function, and Cross-Entroy represents the cross-entropy loss function. SGD and AdaW represent stochastic gradient descent and Adam of "decoupled weight decay regularization", respectively. StepLR means decays the learning rate of each parameter group by gamma every step_size epochs, where CN and EN represent the settings for Chinese and English, respectively. WarmupLinear increases the learning rate through warmup_step steps to the preset value and then decreases linearly to 0

Models	Recurrent		Attention Heads	Transformer-encoder		
	Hiddens	Layers		Dropout	Num	
(a) Hyperparameters for model structure						
Recurrent	256	1	-	-	-	
Bi-Recurrent	128	1	-	-	-	
Recurrent&Attention	256	1	1	-	-	
Bi-Recurrent &Attention	128	1	1	-	-	
T-E-GRU	256	1	2	0.3	1	
BERT[2]	-	-	12	0.1	12	
ALBERT[3]	-	-	12	-	1	
(b) Hyperparameters for training the models						
Moedls	Loss	Optim	Lr	Batch	epoch	Schedule
T-E-GRU	NLL	SGD	1e-2	128	100	StepLR: (step_size=50(CN) step_size=30(EN) gamma=0.5)
Recurrent	NLL	SGD	2e-3	128	100	
Recurrent&Attention	NLL	SGD	2e-2	128	100	
BERT-based	Cross-Entropy	AdamW	5e-5	DMCS_V2:324 DianPing:8 Shopping:16 IMDB:8 Yelp_review:8		WarmupLinear: (warmup_steps=0, num_steps=Total)

sentimentality). In the experiments we randomly selected 700,000 comments data including 350,000 positive samples and 350,000 negative samples. We divide them into three parts, 480,000 for the training set, 120,000 for the validation set, and 100,000 for the test set.

Data preprocessing. Firstly, we need to deal with the punctuation in the text data. Since the use of punctuation in most short film reviews is not standardized, and some punctuation has less semantic information, we retain some punctuation with clause ability, such as comma, full stops, question marks, semicolons, etc. and deleted others. Our focus is not to discuss CWS model or the necessity of CWS in deep learning models, so we use open source library Jieba developed by Baidu engineer Sun Junyi. After word segmentation, we pad and truncate the tokens for batch processing. In addition, we use pre-trained word embedding to convert words into vectors, which was trained by Li S, Zhao Z, Hu R, et al. [23] by using skip-gram with negative sample method based on Zhihu Q&A [23]. It contains 260,000 used high-frequency words, which can cover 93.98% of the words in train set. The dimension of the word vector is 300. DMCS_V2 is a short movies comments dataset. After tokenization, 99.8% of samples' tokens are less than 100. So we align the input sequence to 100 by padding and truncating it at the front of it. For word representation, we use 200,000 high-frequency words in the pre-trained word embedding model, which can cover 93.86% of the word needed in there. However, there are some different for the BERT-based models. BERT as representation model for Chinese is often based on characters rather than words, so we continue this setting. This will result in a smaller vocabulary required and longer sequences produced. The same vocabulary is used for the tokenizers of BERT [2] and ALBERT[3] [22] from Hugging Face[4]. So for both of them we align the sequences to 140 using the same method as before, which covers 98.1% of train samples. The vocabulary contains 21,028 characters covering 93.9% of the characters in train samples. Details about text to sequences on DMCS_V2 are organized in the Table 1.

Train Details. In the experiments of RNN, LSTM and GRU, we use negative log-likelihood (NLL) loss function, set the batch size and the hidden size to 128 and 256 respectively, adopt Stochastic Gradient Descent (SGD) optimizer with learning rate of 0.002 for 100 epochs, and decay the learning rate by 0.5 every 50 epochs. We use the same settings for the bi-directional recurrent model, and the model combined with attention mechanism and recurrent models as mentioned in before, except that the hidden layer in a single direction is set to 128, and set the learning rate to 0.01. Finally, we also use the same setting on our T-E-GRU, and set the dropout of transformer-encoder to 0.3. The BERT-based model, as a powerful representation model, fine-tuning is one of the most commonly used methods to solve downstream tasks. We implement CAS by taking the $[CLS]$ representation as linear layer input as suggested in the original paper [10]. Pre-trained BERT[2] and ALBERT[2] models and fine-tuning settings we follow the recommendation of Hugging Face. Due to device (RTX3080 with 10GB memory) limitations, we had to scale down the batch size to 32 when experimenting with the BERT-based models. More details about the model hyperparameters are organized and presented in the Table 2.

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

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

[4] https://huggingface.co.

Results and Analysis.

The performance of the above models on the test set is shown in Table 3. We use accuracy and F1 as the criteria:

$$\begin{aligned} Acc &= \frac{TP + TN}{TP + TN + FP + FN} \\ F1 &= \frac{2 \times TP}{2 \times TP + FP + FN} \end{aligned} \tag{6}$$

where TP, TN, FP and FN represent true positive, true negative, false positive, and false negative respectively. Test Time denotes the time required for trained model to test a batch of comments .

The result of DMSC_v2 is show in Table 3. In the recurrent models, the best accuracy and F1 are 88.80% and 88.80%, respectively. Among them, RNN, GRU, Bi-GRU have similar prediction performance, but RNN is the fastest, and it only takes 8.7ms to test a batch of samples. After the introduction of attention, the accuracy and F1 of the recurrent model did not improve significantly, but it takes more time to test. The best recurrent model with attention is LSTM-Attention, its accuracy and F1 are 89.43% and 89.32%, which get about 0.6% improvement in accuracy and F1 with nearly 300% test time. In addition, it is worth noting that when the recurrent model combined with the attention layer, only the structure of the attention layer behind the recurrent model can bring improvement. Compared with LSTM-Attention, T-E-GRU takes only 15% more test time to get about 0.7% improvement in accuracy and F1. Obviously, the bert-based model has a significant improvement in accuracy and F1. Especially the original BERT, which has nearly 3.6% improvement in accuracy and F1 than T-E-GRU. However for these improvements, we have to pay for expensive testing time and memory. Due to device (RTX 3080 with 10GB memory) limitations we had to

Table 3 The performance of various models on the DMCS_V2. The test time represents the time required to test a batch (128). For batch_size less than 128, we add '*(128/batch size)' after a batch required time. In addition, the best results in the table have been bolded

Method	Accuracy	F1	Test Time(ms)
RNN [13]	88.80%	88.77%	**8.7**
LSTM [17]	88.41%	88.28%	21.0
GRU [9]	88.70%	88.80%	20.1
Bi-RNN	88.30%	88.30%	15.3
Bi-LSTM	87.95%	88.02%	41.5
Bi-GRU	88.00%	88.80%	16.1
RNN-Attention	88.50%	88.34%	10.5
LSTM-Attention	89.43%	89.32%	24.5
GRU-Attention	88.08%	88.27%	24.3
Attention-RNN	85.77%	86.09%	10.6
Attention-LSTM	85.67%	85.55%	24.7
Attention-GRU	84.68%	84.32%	24.6
BiRNN-Attention	88.64%	88.50%	19.5
BiLSTM-Attention	88.45%	88.48%	47.6
BiGRU-Attention	88.18%	87.60%	20.4
BERT[2] [10]	**93.64%**	**93.69%**	56.7*4
ALBERT[3] [22]	91.78%	91.79%	58.9*4
T-E-GRU	90.09%	90.07%	28.0

reduce the batch size to 32, which is only 1/4 of the other models. Nevertheless, it still takes almost 200% of the time of T-E-GRU to test a batch of samples.

In general, we can find on DMSC_V2:

1. The simple RNN is the fastest and achieves 88.80% accuracy and 88.77% F1.
2. LSTM-Attention, as the best recurrent model with attention, has about 0.6% improvement in prediction performance (accuracy, F1) than the recurrent model, but it requires more test time (180% more test time than RNN).
3. Compared with LSTM-Attention, T-E-GRU has nearly 0.7% improvement in prediction performance with 17% additional test time.
4. Compared to T-E-GRU, the original BERT takes more time and memory to get further improvement in prediction performance (3.55% accuracy and 3.62% F1). In the fact that we had to reduce the batch size to 1/4, its test time of one batch samples is still more than 200% of T-E-GRU.

4.2 YF_DianPing Dataset

Dataset. YF_DianPing [47] is customer reviews of restaurant, which has more characters and more complicated than DMSC_V2. It contains the user reviews crawled from a famous Chinese online review webset DianPing.com, including the 3,605,300 reviews of 510,071 users. After eliminating the NaN data, we randomly selected 80000 1-points reviews as positive samples, and 80,000 1-points reviews as negative samples. We divide them into two parts, 80% are used as the training set and 20% are used as the validation set. Then we obtained the test set by taking 30,000 4-points reviews as positive samples and 30,000 2-points reviews with as negative samples.

Data preprocessing. The processing of punctuation, CWS, the padding/ truncating and pre-trained word embedding model are the same as Sect. 4.1. After tokenization, the longest sample has 1386 tokens and 96.2% of the samples' tokens are less than 400. So we align the input sequence to 400 by using the same method as in Sect. 4.1. For word representation, we use 150,000 high-frequency words, which covering 86.0% of the words in the training set. Due to the complexity of the words used in the data set, using of 260,000 high-frequency words only covers 86.5%, so it is more appropriate to use 150,000 high-frequency. For BERT-based models, as mentioned in section 4.1, the tokens after character-based tokenization will be longer than word-based, so we align the input sequence to 500, and 93.7% of the sequences are smaller than the number. The vocabulary is the same as before with 21028 characters, covering 99.6% of the characters in the training samples. Details about text to sequences on YF_DianPing Dataset are organized in the table 1.

Train Details. On YF_DianPing Dataset, we did the same experiment and only modified the dataset. The various hyperparameters, loss functions, optimizers, etc. of each model are the same as Sect. 4.1. The only difference is that due to device (RTX3080 with 10GB memory) limitations, we had to reduce the batch size to 8 when experimenting on the BERT-based models. More details about the model hyperparameters are organized and presented in the Table 2.

Results and Analysis

The results of various models on the DianPing test set are shown in Table 4. The test set uses 2 points and 4 points review data, while the training set uses 1 point and 5 points review data, so there may be more ambiguous texts in the test set, resulting in poor performance. On this test set, Bi-GRU has the best performance in recurrent model, with accuracy and F1 reaching 89.55% and 89.94% respectively. Compared with the best-performing model

Table 4 The performance of various models on the YF_DianPing. The '-' in the table indicates that the result is too bad. The Test Time indicates the time required to test a batch (128) samples. For batch_size less than 128, we add '*(128/(batch_size))' after the required time for a batch. In addition, the best results in the table have been bolded

Method	Accuracy	F1	Test Time(ms)
RNN [13]	87.95%	88.38%	**36.8**
LSTM [17]	88.40%	88.91%	91.0
GRU [9]	88.83%	89.44%	90.9
Bi-RNN	89.23%	89.51%	67.3
Bi-LSTM	87.90%	87.50%	174.2
Bi-GRU	89.55%	89.94%	64.5
RNN-Attention	88.92%	89.46%	43.7
LSTM-Attention	90.45%	90.81%	93.4
GRU-Attention	88.55%	89.07%	93.3
Attention-RNN	87.29%	88.12%	38.7
Attention-LSTM	87.92%	88.54%	99.4
Attention-GRU	88.06%	88.64%	98.2
BiRNN-Attention	89.54%	89.86%	67.4
BiLSTM-Attention	87.70%	88.20%	189.2
BiGRU-Attention	88.88%	89.36%	102.3
BERT	–	–	–
ALBERT	**92.25%**	**92.33%**	128.7*16
T-E-GRU	90.76%	90.98%	97

RNN in Douban data, Bi-GRU has an improvement of about 1.5% on both metrics. LSTM-Attention achieves the best performance in attention-based recurrent models with 90.45% accuracy and 90.81% F1. Compared with LSTM-Attention, T-E-GRU trades extra 4% test time for a slight improvement in prediction performance (accuracy, F1). The loss of the original BERT cannot be stably decreased and converged. We speculate that it is due to the inappropriate batch size, which also reflects that the fine-tuning of the original BERT may face more complicated situations when the device (RTX 3080 with 10GB memory) is limited. Compared with T-E-GRU, ALBERT has about 1.5% improvement in prediction performance. However, when the batch size is only 1/16 of T-E-GRU, it still takes 25% more time to test a batch of samples.

In general, We can find:

1. In recurrent models and recurrent models with attention, LSTM-Attention achieves the best prediction performance. Compared with the fastest RNN, it takes nearly 150% more test time and obtains about 2.5% prediction performance improvement.
2. Compared with LSTM-Attention, T-E-GRU trades additional 4% test time for about 0.25% prediction performance improvement.
3. Due to device (RTX 3080 with 10GB memory) limitations, the BERT-based model has to reduce the batch size to 8 when the input sequence becomes longer. On this dataset, the original BERT training fails. ALBERT has a 1.5% prediction performance improvement over T-E-GRU at expensive test time cost. It takes only 25% more test time under the batch size is only 1/16 of T-E-GRU.

4.3 Online_shopping_10 Cats Dataset

Dataset. This dataset comes from various Chinese E-commerce platforms, with a total of more than 60,000 comment data, including 30,000 positive and negative comments. We ignored their category tags and only used sentimental tags. We divide them into training set, validation set, and test set according to the ratio of 6:1:3. This dataset is smaller than the previous two datasets, and we use it to test the performance of our model on small-scale datasets.

Data preprocessing. Compared with the previous two datasets, it is a small-scale moderate-length dataset. After tokenization, the longest sample has 1502 tokens and 99.0% of the samples' tokens are less than 200. So we align the input sequence to 200. For word representation, we use 200,000 high-frequency words, which can cover 94.9% of the word needed in the training set. Using all words in pre-trained word embedding model is only about 0.2% improvement. As with the previous two datasets, for the BERT-based model, we re-tokenize and align the sequence to 250. 97.8% of the training samples are less than 250. The BERT-based vocabulary is still 21,028 characters, covering 99.6% of the characters in the training samples. Details about Text to Sequence on Online_shopping_10 cats Dataset are organized in the Table 1.

Train Details. In the experiments of Online_shopping_10 cats dataset, the various hyperparameters, loss functions, optimizers, etc. of each model are the same as Sect. 4.1. However, since the size of this data set is small, many models use 100 epochs can not complete the training. Addressing the situation, we double the initial learning rate and set the training epochs to 300. The performances of each model are shown in Table 5, where 300 epochs of training are marked after the model name. In addition, we adjust the batch size of the BERT-based models to 16 to fit the device (RTX 3080 with 10GB memory). More details about the model hyperparameters are organized and presented in the Table 2.

Results and Analysis.

The results of various models on the Online_shopping_10 cats test set are shown in Table 5. On this test set, various recurrent models have similar performance. The accuracy and F1 of recurrent models are around 90% and 90%. BiRNN-Attention achieves the best performance in recurrent models with attention, with accuracy and F1 reaching 92.32% and 92.38% respectively. Compared with the recurrent models, BiRNN-Attention has an improvement of about 2%. In contrast, LSTM-Attenion, which performed well before, was defeated on this dataset. Compared with BiRNN-Attention, T-E-GRU has about 0.6% and 1% improvement in accuracy and F1 respectively, and it takes 46% more in test time. The original BERT achieves the best prediction performance, which is about 1.8% better than T-E-GRU. Similarly, it still takes more 12% test time than T-E-GRU under the premise that the batch size is reduced to 1/8.

In general, we can find:

1. In recurrent models and recurrent models with attention, LSTM-Attention is defeated by BiRNN-Attention with about 0.5% prediction performance improvement, but it is still the second prediction performance except T-E-GRU and BERT-based models.
2. Compared with BiRNN-Attention, T-E-GRU trades 40% test time cost for more than 1% prediction performance improvement.

Table 5 The performance of various models on the Online_Shopping. Among them, because the Online_Shopping is relatively small, it takes 300 rounds of training for many models to converge. Those who need to train for 300 epochs have added '-300' after the model name. The Test Time indicates the time required to test a batch (128) samples. For batch_size less than 128, we add '*(128/(batch_size))' after the required time for a batch. In addition, the best results in the table have been bolded

Method	Accuracy	F1	Test Time(ms)
RNN	89.68%	89.93%	**17.5**
LSTM	90.21%	90.06%	41.6
GRU	90.08%	90.36%	42.6
Bi-RNN	90.10%	90.07%	29.5
Bi-LSTM	90.10%	90.10%	81.5
Bi-GRU	90.34%	90.13%	32
RNN-Attention	90.07%	90.10%	21.1
LSTM-Attention	91.86%	91.76%	45.6
GRU-Attention-300	90.93%	90.96%	46.1
Attention-RNN	90.68%	90.72%	20.8
Attention-LSTM-300	90.91%	90.91%	46.5
Attention-GRU-300	89.56%	89.65%	46.8
BiRNN-Attention-300	92.32%	92.38%	34
BiLSTM-Attention-300	90.28%	90.16%	87.2
BiGRU-Attention-300	91.09%	91.05%	36.6
BERT	**94.77%**	**94.83**	55.4*8
ALBERT	94.28%	94.34%	57.1*8
T-E-GRU-300	92.92%	93.05%	49.6

3. The original BERT achieves the best prediction performance, which is about 1.8% better than T-E-GRU. The original BERT achieves a prediction performance improvement of about 1.8 over T-E-GRU. However, similar to the previous experiments, it still takes more 12% test time than T-E-GRU under the premise that the batch size is reduced to 1/8.

4.4 Ablation Study

We conducted ablation experiments to better understand the contributions and the hyper-parameter setting of T-E-GRU. We also conduct experiments on on the DMSC_V2, YF_DianPing and Online_shopping datasets. In addition, in Sect. 4.4.3 we also test the performance of T-E-GRU on the English datasets.

4.4.1 Replacing GRU with other Recurrent Layer

In this section, we tried replacing GRU with other recurrent models (RNN, LSTM). The experiment was run on RTX 2080Ti with 11GB memory. It's worth noting that the Test Time in the results here represents the time it takes to test a comment on the trained model. Details of other hyperparameters of the models can be found in the Table 2. The experimental results are shown in Table 6. On DMSC_V2, T-E-GRU has achieved the best performance compared to other models, but it has only a slight improvement in accuracy

Table 6 The performance of replacing GRU with other recurrent layers. Among them, because Online_shopping is relatively small, it takes 300 epochs of training for T-E-GRU to converge. The best result has been bolded. The T-E-GRU achieves the best prediction performance

Dataset	Method	Accuracy	F1	Test Time (ms)
DMSC_V2	T-E-RNN	89.25%	89.30%	**1.5390**
	T-E-LSTM	89.75%	89.88%	2.9321
	T-E-BiRNN	89.26%	89.30%	2.1434
	T-E-BiLSTM	89.81%	89.74%	4.7107
	T-E-BiGRU	89.77%	89.85%	2.2185
	T-E-GRU	**90.09%**	**90.07%**	2.6987
YF_DianPing	T-E-RNN	88.91%	89.42%	**3.4685**
	T-E-LSTM	90.25%	90.70%	8.0919
	T-E-BiRNN	88.94%	89.40%	5.9280
	T-E-BiLSTM	90.63%	90.92%	15.4538
	T-E-BiGRU	90.73%	90.89%	6.3677
	T-E-GRU	**90.76%**	**90.98%**	8.0009
Online_shopping	T-E-RNN	91.16%	91.01%	**2.1586**
	T-E-LSTM	92.43%	92.59%	4.4119
	T-E-BiRNN	91.40%	91.49%	3.3178
	T-E-BiLSTM	92.41%	92.47%	8.2653
	T-E-BiGRU	92.38%	92.60%	3.3085
	T-E-GRU-300	**92.92%**	**93.05%**	4.3890

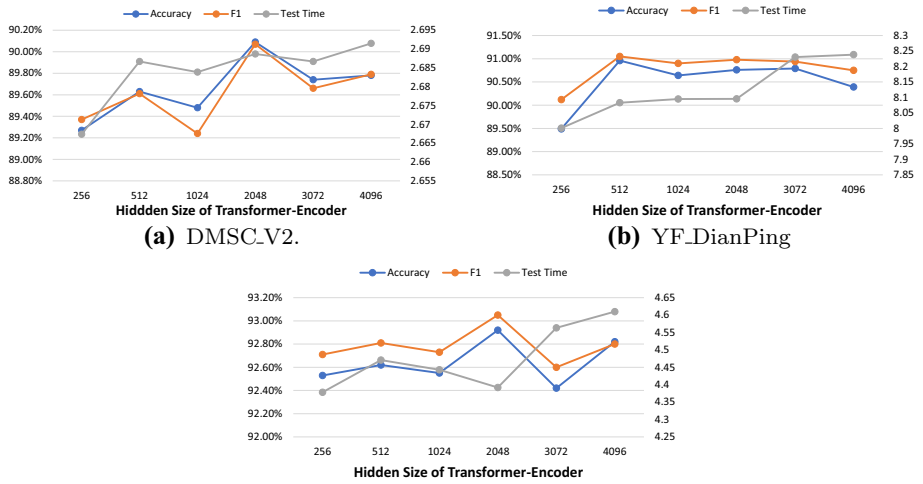
and F1. On YF_DianPing, T-E-GRU only has less than 1% improvement in accuracy, F1 than other models. T-E-GRU is still the best accuracy and F1, which are 90.76% and 90.98% respectively. On Online_shopping, T-E-GRU still obtain the best accuracy and F1. In general, GRU is more suitable for combining with transformer-encoder. The multiple of the time spent is usually positively correlated with the number of tokens.

4.4.2 Adjustment of Hyperparameters in T-E-GRU

Firstly, we tried to adjust the hidden size of the transformer-encoder. The results on three datasets mentioned in Sect. 4.1, 4.2, 4.3 are shown in Fig. 3. The accuracy, F1 are introduced in Sect. 4.1. The Test Time in the results here represents the time it takes to test a comment on the trained model. The experiment also run on RTX 2080Ti with 11GB memory. Details of other hyperparameters of the models can be found in the Table 2. From the results, we find that the different hidden sizes of transformer-encoder have little effect on Test Time, mostly less than 0.5(ms). For accuracy and F1, DMSC_V2 and Online_shopping have the best performance in 2048, while 2048 and 3072 on YF_DianPing have very similar performance.

Then we adjusted the number of heads in transformer-encoder. On the same datasets, the experiment results are shown in Fig. 4. In fact, the number of heads from 1 to 4 has less than 0.5(ms) effect on Test Time, and it is very slight. On DMSC_V2 and Online_shopping, the number of heads is 2 can achieve the best accuracy and F1. On YF_DianPing, the number of heads is 1 or 2 with similar predict performance.

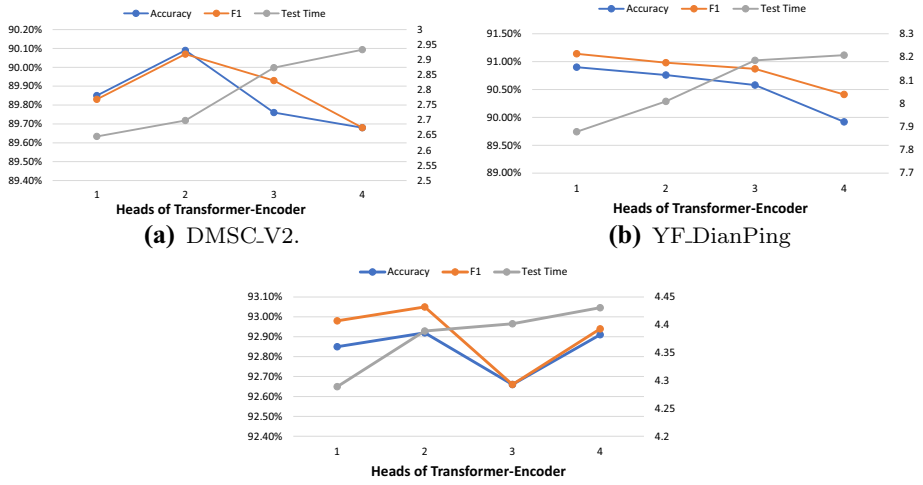
Finally, we adjusted the dropout of transformer-encoder and GRU on the same datasets. Since dropout does not affect the complexity of the model, we removed Test Time, and the



(a) DMSC_V2. **(b)** YF_DianPing

(c) Online_shopping_10 cats Dataset

Fig. 3 Evaluation results of T-E-GRU on different hidden size of transformer-encoder. We also conducted experiments on the previous three datasets. We note that 2048 achieves the best F1 and accuracy on DMSC_V2 and Online_shopping_10 cats. On YF_DianPing, the 512 achieves the best F1 and accuracy, but the 2048 is very close to it

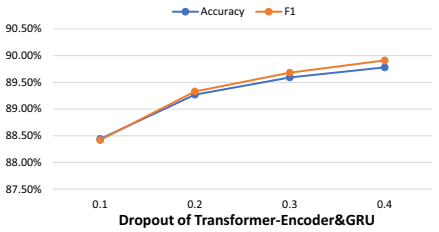


(a) DMSC_V2. **(b)** YF_DianPing

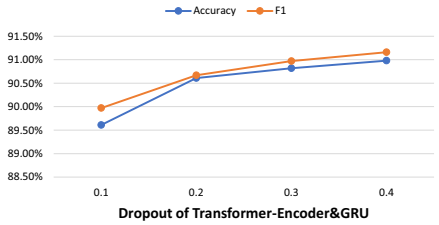
(c) Online_shopping_10 cats Dataset

Fig. 4 Evaluation results of T-E-GRU on different number of transformer-encoder for the previous three datasets. We note that 2 heads achieve the best F1 and accuracy on DMSC_V2 and Online_shopping_10 cats, but on YF_DianPing, single head is the best

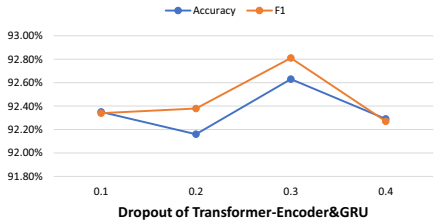
result is shown in Fig. 5. As the result shows, on DMSC_V2 and YF_DianPing, dropout of 0.4 is the best in accuracy and F1, while on Online_shopping, 0.3 is the best.



(a) DMSC_V2.



(b) YF_DianPing



(c) Online_shopping_10 cats Dataset

Fig. 5 Evaluation results of T-E-GRU on different dropout for the previous three datasets. We note that dropout of 0.4 is the best on DMSC_V2 and YF_DianPing, but on Online_shopping, 0.3 is the best

Table 7 The data division of the two English datasets and the details of text to sequence. Train, Val, Test represent the number of samples in the training set, validation set, and test set, respectively. The Align column shows the length of the sequence align, and the parentheses indicates a percentage less than this length. Vocabs column represents the number of words in the vocabulary, and parentheses represent their coverage on training examples

Dataset	Train	Val	Test	Model	Align	Vocabs
IMDB	35,000	7,500	7,500	BERT	510 (87.1%)	28,896 (≈ 100%)
				LSTM-Attention	500 (90.0%)	400,000 (87.7%)
				T-E-GRU	500 (90.0%)	400,000 (87.7%)
Yelp_review	448,000	112,000	38,000	BERT	510 (96.6%)	28,896 (≈ 100%)
				LSTM-Attention	500 (97.0%)	400,000 (88.1%)
				T-E-GRU	500 (97.0%)	400,000 (88.1%)

In general, we can find:

1. GRU is more suitable for combination with transformer-ecoder than LSTM and RNN.
2. For T-E-GRU, adjusting the hidden size of transformer-encoder, heads of transformer-encoder and dropout has little effect on Test Time, while the accuracy and F1 are different depending on the datasets.

4.4.3 The Performance of T-E-GRU in English

Although our model is designed for Chinese, we also conduct experiments on English datasets. In previous experiments, Bert and LSTM-Attention achieves the best performance on Bert-based model and recurrent model with attention, respectively. So we compared them with T-E-GRU on IMDB [28] and Yelp_review [46].

Table 8 The performances of T-E-GRU in English datasets. The table shows the experimental results of origin BERT, LSTM-Attention, and T-E-GRU on IMDB and Yelp_review. The Test Time indicates the time required to test a batch (128) samples. For batch_size less than 128, we add '*(128/(batch_size))' after the required time for a batch. The best results have been bolded

Dataset	Method	Accuracy	F1	Test Time(ms)
IMDB	BERT	87.47%	87.57%	62.1*16
	LSTM-Attention	87.97%	87.69%	**35.8**
	T-E-GRU	**90.23%**	**90.30%**	48.9
Yelp_review	BERT	–	–	–
	LSTM-Attention	94.86%	94.88%	**34.1**
	T-E-GRU	**95.43%**	**95.42%**	48.7

Specifically, we first filter some non-semantic symbols in original text, such as HTML tags '< br />', '\', ''s', etc. So it can alleviate the sequences after tokenization are too long. Then for T-E-GRU and LSTM-Attention, we use NLTK[5] and Glove-wiki-gigaword-300 [32] respectively for tokenization and word embedding. For BERT, we also use pre-trained model[6] and Tokenizer from Hugging Face. After tokenization, we do the same pre-padding and pre-truncating for all models. Since English text sequences are usually longer than Chinese, we align the input sequence to 500 for T-E-GRU and LSTM-Attention, which can cover IMDB 90.0% and Yelp_review 97.0%. For BERT, we align it to 510 which can cover IMDB 87.1% and Yelp_review 96.6%. These details are organized in Table 7. For BERT fine-tuning hyperparameters, we follow the recommendation[6]. More details are organized in Table 2 except for changing the step_size of the schedule in T-E-GRU to 30. In fact, this change will not affect the experimental results.

Due to device (RTX 3080 with 10GB memory) limitations, we again reduce the BERT batch size to 8. The experimental results are shown in Table 8, where the Test Time represents the time required for the trained model to test a batch of (128) samples. On both English datasets, the loss of BERT cannot drop steadily. We suspect that this phenomenon is caused by the batch size (8) being too small, but further experiments cannot be carried out due to device limitations. Although BERT on IMDB is not the best, it still has the ability to classify, while the training of BERT on Yelp_review fails completely. Compared with the second best LSTM-Attention, T-E-GRU achieves about 3% and 0.6% prediction performance improvement on IMDB and Yelp_review. At the same time, T-E-GRU also spends about 40% more test time on average, which is insignificant compared to BERT.

In general, we can find:

1. Limited by device (RTX 3080 with 10GB memory), BERT does not show the prediction performance advantage.
2. T-E-GRU achieves the best prediction performance on both English datasets (3% and 0.6% improvement on IMDB and Yelp_review, respectively). Compared with LSTM-Attention, it takes nearly 40% more testing time on average.

[5] https://www.nltk.org/.

[6] https://huggingface.co/bert-base-cased.

5 Conclusion

For Chinese sentiment analysis, we propose T-E-GRU to address the insufficiency of Transformer to capture sequence feature by position encoding, and the problems of recurrent model on long sequences (information loss, the vanishing/exploding gradient, etc.). Integrating both the advantages of Transformer and recurrent model, T-E-GRU firstly generates reprocessed sequence with global information by the transformer-encoder, then uses the GRU to extract the feature of the reprocessed sequence, and finally classifies the final state output by the GRU. We compared our proposed models against various recurrent models, recurrent models with attention and BERT-based models on three real Chinese datasets. We select these datasets from different domains, and filtered out some punctuation that does not have the ability of clause. Apart from fine-tuning the BERT-based models, T-E-GRU achieved the best prediction performances. Although BERT performed about 2% improvement of prediction than T-E-GRU, it has too much parameters to be trained and may face training failure with the limited device (RTX3080 with 10GB memory). It should be noted that the test time of T-E-GRU is 8 times faster than the fine-tuning BERT-based model. In Chinese text sentiment analysis, although T-E-GRU has unique advantages compared to common models (recurrent models, recurrent models with attention, BERT-based models), its prediction performance is slightly inferior to fine-tuning BERT-based model. In the future, we will improve the prediction performance of T-E-GRU on the basis of not increasing the delay as much as possible, or even faster.

Acknowledgements This work is supported by the Key Research and Development Program of Shaanxi Province (No. 2020KW-068), the National Natural Science Foundation of China under Grant (No. 62106199, No. 62002290, No. 62001385), China Postdoctoral Science Foundation (No. 2021MD703883) and General Project of Education Department of Shaanxi Provincial Government under Grant (No. 21JK0926).

References

1. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473
2. Bengio Y, Ducharme R, Vincent P, Janvin C (2003) A neural probabilistic language model. J Mach Learn Res 3:1137–1155
3. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Networks 5(2):157–166
4. Bin L, Quan L, Jin X, Qian Z, Peng Z (2017) Aspect-based sentiment analysis based on multi-attention CNN. J Comput Res Dev 54(8):1724
5. Cavnar WB, Trenkle JM, et al. (1994) N-gram-based text categorization. In: Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, 161175. Citeseer
6. Chen P, Sun Z, Bing L, Yang W (2017) Recurrent attention network on memory for aspect sentiment analysis. In: Proceedings of The 2017 Conference on Empirical Methods in Natural Language Processing, pp 452–461
7. Chen X, Qiu X, Zhu C, Liu P, Huang XJ (2015) Long short-term memory neural networks for Chinese word segmentation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp 1197–1206
8. Cheng J, Dong L, Lapata M (2016) Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733
9. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078
10. Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805

11. Ding X, Liu B, Yu PS (2008) A holistic lexicon-based approach to opinion mining. In: Proceedings of The 2008 International Conference on Web Search and Data Mining, pp. 231–240
12. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al. (2020) An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929
13. Elman JL (1990) Finding structure in time. Cogn Sci 14(2):179–211
14. Feldman R (2013) Techniques and applications for sentiment analysis. Commun ACM 56(4):82–89
15. Gao J (2021) Chinese sentiment classification model based on pre-trained BERT. In: 2021 2nd International Conference on Computers, Information Processing and Advanced Education, pp 1296–1300
16. Gu F, Askari A, El Ghaoui L (2020) Fenchel lifted networks: A lagrange relaxation of neural network training. In: International Conference on Artificial Intelligence and Statistics, pp 3362–3371. PMLR
17. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
18. Hu D (2019) An introductory survey on attention mechanisms in NLP problems. In: Proceedings of SAI Intelligent Systems Conference, pp 432–448. Springer
19. Huang C, Zhao H (2007) Chinese word segmentation: A decade review. J Chin Inf Process 21(3):8–20
20. Jordan M (1986) Serial order: a parallel distributed processing approach. Technical report, June 1985-March 1986. Tech. rep., California Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science
21. Lafferty J, McCallum A, Pereira FC (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data
22. Lan Z, Chen M, Goodman S, Gimpel K, Sharma P, Soricut R (2019) Albert: A lite BERT for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942
23. Li S, Zhao Z, Hu R, Li W, Liu T, Du X (2018) Analogical reasoning on Chinese morphological and semantic relations. arXiv preprint arXiv:1805.06504
24. Li X, Meng Y, Sun X, Han Q, Yuan A, Li J (2019) Is word segmentation necessary for deep learning of Chinese representations? arXiv preprint arXiv:1905.05526
25. Liang J, Chai Y, Yuan H, Gao M, Zan H (2015) Polarity shifting and LSTM based recursive networks for sentiment analysis. J Chine Inf Process 29(5):152–159
26. Liu M, Chen L, Du X, Jin L, Shang M (2021) Activated gradients for deep neural networks. IEEE Transactions on Neural Networks and Learning Systems
27. Liu S, Mocanu DC, Pei Y, Pechenizkiy M (2021) Selfish sparse RNN training. arXiv preprint arXiv:2101.09048
28. Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C (2011) Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp 142–150. Association for Computational Linguistics, Portland, Oregon, USA. http://www.aclweb.org/anthology/P11-1015
29. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. arXiv preprint arXiv:1310.4546
30. Mikolov T, Yih Wt, Zweig G (2013) Linguistic regularities in continuous space word representations. In: Proceedings of The 2013 Conference of The North American Chapter of The Association for Computational Linguistics: Human Language Technologies, pp 746–751
31. Mnih V, Heess N, Graves A, Kavukcuoglu K (2014) Recurrent models of visual attention. arXiv preprint arXiv:1406.6247
32. Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: Proceedings of The 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 1532–1543
33. Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. arXiv preprint arXiv:1802.05365
34. Rabiner L, Juang B (1986) An introduction to hidden Markov models. IEEE ASSP Mag 3(1):4–16
35. Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding by generative pre-training
36. Rumelhart DE, Hinton GE, Williams RJ (1985) Learning internal representations by error propagation. Tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science
37. Shi X, Lu R (2019) Attention-based bidirectional hierarchical LSTM networks for text semantic classification. In: 2019 10th International Conference on Information Technology in Medicine and Education (ITME), pp 618–622. IEEE
38. Tang F, Nongpong K (2021) Chinese sentiment analysis based on lightweight character-level BERT. In: 2021 13th International Conference on Knowledge and Smart Technology (KST), pp 27–32. IEEE
39. Vashishth S, Upadhyay S, Tomar GS, Faruqui M (2019) Attention interpretability across NLP tasks. arXiv preprint arXiv:1909.11218

40. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. arXiv preprint arXiv:1706.03762
41. Wang S, Li J, Hu D (2021) BiGRU-multi-head self-attention network for Chinese sentiment classification. In: Journal of Physics: Conference Series, 1827: 012169. IOP Publishing
42. Xiao Z, Liang P (2016) Chinese sentiment analysis using bidirectional LSTM with word embedding. In: International Conference on Cloud Computing and Security, pp 601–610. Springer
43. Yao Y, Huang Z (2016) Bi-directional LSTM recurrent neural network for Chinese word segmentation. In: International Conference on Neural Information Processing, pp 345–353. Springer
44. Yu Z, Yu J, Cui Y, Tao D, Tian Q (2019) Deep modular co-attention networks for visual question answering. In: Proceedings of The IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 6281–6290
45. Zhang L, Wang S, Liu B (2018) Deep learning for sentiment analysis: A survey. Wiley Interdiscip Reviews Data Mining Knowl Discov 8(4):e1253
46. Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. Advances in Neural Information Processing Systems 28:649–657
47. Zhang Y, Zhang M, Liu Y, Ma S, Feng S (2013) Localized matrix factorization for recommendation based on matrix block diagonal forms. In: Proceedings of The 22nd International Conference on World Wide Web, pp 1511–1520
48. Zhang Z, Brand M (2017) Convergent block coordinate descent for training tikhonov regularized deep neural networks. Advances in Neural Information Processing Systems 30:1721–1730
49. Zhang Z, Yue Y, Wu G, Li Y, Zhang H (2021) SBO-RNN: reformulating recurrent neural networks via stochastic bilevel optimization. Adv Neural Inf Process Syst 34:25839–25851
50. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W (2020) Informer: beyond efficient transformer for long sequence time-series forecasting. arXiv preprint arXiv:2012.07436
51. Zhou L, Zhou Y, Corso JJ, Socher R, Xiong C (2018) End-to-end dense video captioning with masked transformer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 8739–8748
52. Zhu Z, Zhou Y, Xu S (2019) Transformer based Chinese sentiment classification. In: Proceedings of the 2019 2nd International Conference on Computational Intelligence and Intelligent Systems, pp 51–56
53. Zou H, Tang X, Xie B, Liu B (2015) Sentiment classification using machine learning techniques with syntax features. In: 2015 International Conference on Computational Science and Computational Intelligence (CSCI), pp 175–179. IEEE