Exploring the Combination of Different Backbone Neural Networks for Text Classification

Zesheng Zhan, Yizhou Zeng,Hanyang Shao, Huisan Xu
23020221154142, 23020221154141, 23020221154107, 23020221154130
School of Informatics Xiamen University (National Demonstrative Software School)
4221 Xiang'an South Road, Xiamen University Xiang'an Campus
Xiamen, Fujian 361000

Abstract

Current thinking in dealing with text classification tasks is often limited to fine-tuning on downstream tasks using autocoding or autoregressive pre-training models. Current research has focused on how to design the training task so that it trains larger and better models, but most of the work is still based on pre-trained masked language models like Bert.In this paper, we try to combine different backbone networks to discover how to combine the advantages of different backbone networks more efficiently to achieve better processing of text information, and try to make improvements and breakthroughs in the structure of the pre-trained model itself. In this paper, we will test CNN, RNN, LSTM, and Bert models individually and with different fusion strategies on IMDB dataset, and we will try to determine the opportunities for structural improvements.

Introduction

Text classification refers to the automatic classification and labeling of text (or other entities) by computer according to a certain classification system or criteria. It has numerous application scenarios and is a fundamental component of many NLP programs. Examples include sentiment analysis[1], relationship extraction[2], and spam detection[3].

Therefore many researchers have proposed different models to improve the accuracy of text classification. Bag-of-words model uses a set of unordered words to represent a paragraph or a document[4]. However, it cannot encode discourse order and syntactic features. Recurrent neural networks are used in the field of natural language processing due to their time-series processing in a similar way to text reading. The model structure is able to learn historical and positional information efficiently, which helps to solve the long-distance dependency problem. Examples include RNNs using 1D max-pool operations[5] or attention-based operations[6].

Long-term short-term memory networks use forgetting gate structure to filter the transmission of textual information therefore can capture connections between contextual feature words over longer time scales and mitigate the gradient disappearance problem in RNNs by increasing the parameter matrix. However, the training of RNN/LSTM is time-consuming due to the large memory bandwidth required for the linear layer to perform the computation, and the LSTM does not completely solve the gradient disappearance problem.

Kim designed an unbiased model of convolutional neural networks to successfully use CNNs for text classification tasks, called TextCNN[7]. It can better determine the distinguished phrases in the maximum pooling layer by one layer of convolution and learn hyperparameters other than the word vector by keeping the word vector static. However the inductive bias of CNN locality does not apply on long texts.

Recently, Jacob Devlin et al. proposed the BERT (Bidirectional Encoder representation from Transformers) pre-trained language model[8], which contains a deep two-channel transformer network to better learn semantic knowledge.

Bert changed the technical development of the NLP field. Since then researchers have tended to limit themselves to how to design the training task to train a better pre-trained mask model, ignoring the impact on the task caused by the characteristics of the different backbone networks themselves.

In this paper, we try to combine different backbone networks and explore how to more effectively combine the advantages of different backbone networks to achieve better representation of text information. In this paper, we will test the CNN, RNN, LSTM and Bert models separately on IMDB datasets and test the results using different fusion strategies. In the end, Bert directly combined with lstm has the best performance, reaching 92.99

Related work

Neural network models based on deep learning have achieved good results on text classification tasks.These models typically use a projection layer to map text words to a high-dimensional vector. The vectors are then combined with different neural networks to output a fixed-length representation. Depending on the structure of the backbone network, we classify the following four types: RNN, CNN, Transformer, and other neural networks.

Recurrent Neural Network (RNN)[9]. The most important aspect of solving large-scale text classification problems using deep learning is to solve the problem of text representation, and then automatically acquire feature representation capabilities using network structures such as CNN/RNNs,

no longer relying on complex manual feature engineering thus solving the problem end-to-end. Recurrent neural network (RNN) is a kind of neural network for processing sequential data. Compared with ordinary neural networks, it can handle sequential changes in data, and it can effectively learn historical and location information, which helps to solve long-distance dependency problems. In RNN backpropagation, the weights are adjusted by the gradient calculated by successive multiplication of the derivatives. If the derivatives are very small, the continuous multiplication may have the problem of gradient disappearance. Long Short Time Memory Network (LSTM)[10] is a special kind of RNN, which is mainly designed to solve the gradient vanishing and gradient explosion problems during the training of long sequences. It consists of a memory unit and three gate structures, which are forgetting gate, updating gate and output gate. It filters invalid information through the forgetting gate structure to learn better to longer distance historical information.

Convolutional Neural Network (CNNs)[11]. CNNs were originally used for image classification, where convolutional filters can extract image features. Unlike RNNS, CNNS can apply different kernel-defined convolutions to multiple blocks of a sequence at the same time. As a result, CNNS is used for many NLP tasks, including text classification. For text classification, text needs to be represented as a vector similar to an image representation, and text features can be implemented by multiple channels in the convolution process to filter information from multiple perspectives.

CNNS and RNNS provide good results in tasks related to text classification, but the drawback is the presence of hidden vectors in the middle of these models, which are not intuitive enough and poorly interpretable, especially for some classification errors, which cannot be explained due to the unreadability of the hidden data. Therefore, attention models are proposed.

In essence, the attention mechanism in deep learning is similar to the human selective visual attention mechanism. Its core goal is to select the information that is more critical to the current task goal from a multitude of information. In fact, in problems involving language or vision, some parts of the input are more important for decision making than others, so the use of attention mechanisms often improves the accuracy of decisions.

Transformer[12].Transformer completely abandons RNNs and CNNs, and consists only of a self-attentive mechanism and a forward neural network. transformer can perform parallel computation without considering sequential information, and is suitable for large-scale datasets, which makes it popular for NLP tasks. Bert is one type of network that uses transformer as its backbone pre-trained language models. It is a generative language model that uses Transformer's encoder. By transforming the input vector and designing training tasks for Masked LM and Next Sentence Prediction, Bert effectively learns global semantic representations and significantly improves the model's performance on various NLP tasks, including text classification, named entity recognition, question-and-answer tasks, etc. Subsequent work typically uses an unsupervised approach to automatically mine semantic knowledge, enabling the machine to improve its understanding of semantics by constructing better pre-trained targets.

Other neural networks.Since graph neural networks can learn the syntactic structure of sentences, some researchers have tried to apply GNNs to text classification.DGCNN[13] is a graph network that converts text to graphs and learns different levels of semantics compared to CNN/RNN models.Yao et al[14] investigated the text graph convolutional network (TextGCN), which creates a heterogeneous graph for the entire dataset constructs a heterogeneous word-text graph and captures global word co-occurrence information.

Model

We mainly used TextCNN, RNN, LSTM, Bert as components and tried different fusion strategies. The details of the different components are described in the following sections.

TextCNN

Yoon Kim proposed the textCNN model for text classification with some deformations of the input layer of CNN. textCNN structure is shown in Figure 1, compared with the traditional image CNN network, textCNN has only one layer of convolution and one layer of max-pooling in the network structure, and finally the output is connected to softmax for the classification task. Compared with CNN networks for images, the biggest difference of textCNN is the difference in the input data. The image is two-dimensional data, the image convolution kernel is sliding from left to right, top to bottom for feature extraction. Natural language is one-dimensional data, and although word-embedding generates a two-dimensional vector, it does not make sense to convolve the word vector by sliding it from left to right. For example, if "today" corresponds to the vector [0, 0, 0, 0, 0, 1], sliding from left to right by window size of 1*2 gives four vectors [0,0], [0,0], [0,0], [0, 1], which all correspond to the word "today", this sliding does not help. The success of TextCNN is not the success of the structure itself, but the introduction of trained word vectors to achieve performance beyond benchmark on multiple datasets, further demonstrating that constructing a better embedding is a key capability to improve nlp tasks.

RNN

Humans don't start their thinking from a blank brain every moment of the day. When humans read texts, they tend to infer the true meaning of the word at hand based on the understanding they already have of the word they saw earlier. We don't just throw it all away and think with a blank brain. Our minds possess persistence. RNN simulates this process. Figure 2 shows the process of RNN understanding text

In the following, we note that x is the input, h is the hidden unit, o is the output, L is the loss function, and y is the label of the training set. The t with the upper right corner of these elements represents the state at the moment t. V, W, and U are the weights, and the weights of the same type of

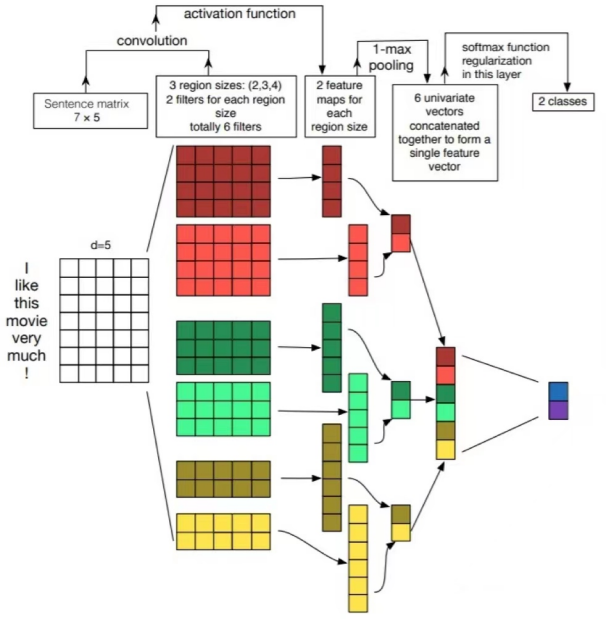

Figure 1: TextCNN structure

weight connections have the same weights. RNN in forward propagation, for moment t:

$$h^{(t)} = \phi(Ux^{(t)} + Wh^{(t-1)} + b)$$

where $\phi()$ is the activation function, and in general the tanh function is chosen, and b is the bias. The output at moment t is.

$$o^{(t)} = Vh^{(t)} + c$$

The predicted output of the final model is.

$$\hat{y}^{(t)} = \sigma(o^{(t)})$$

where σ is the activation function, and usually RNN is used for classification, so the softmax function is generally used here. For the back propagation of RNN, there are three parameters for the optimization search, which are U, V, and W. The optimization search process for two parameters, W and U, needs to trace back the previous historical data, and the parameter V only needs to focus on the present.

$$\frac{\partial L^{(t)}}{\partial V} = \frac{\partial L^{(t)}}{\partial o^{(t)}} \cdot \frac{\partial o^{(t)}}{\partial V}$$

$$L = \sum_{t=1}^{n} L^{(t)}$$

$$\frac{\partial L}{\partial V} = \sum_{t=1}^{n} \frac{\partial L^{(t)}}{\partial o^{(t)}} \cdot \frac{\partial o^{(t)}}{\partial V}$$

$$\frac{\partial L^{(t)}}{\partial W} = \sum_{k=0}^{t} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \left(\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \right) \frac{\partial h^{(k)}}{\partial W}$$

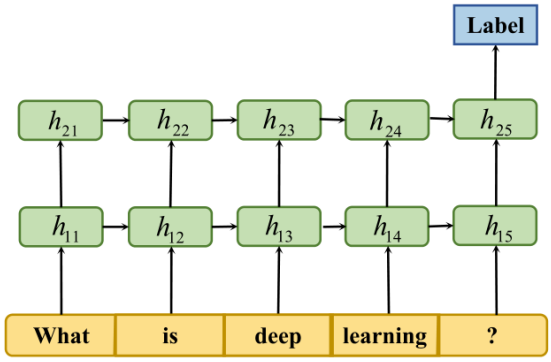

Figure 2: The process of RNN understanding text

$$\frac{\partial L^{(t)}}{\partial U} = \sum_{k=0}^{t} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \left(\prod_{j=k+1}^{t} \frac{\partial h^{(j)}}{\partial h^{(j-1)}} \right) \frac{\partial h^{(k)}}{\partial U}$$

If we put in the activation function and take out the part of the intermediate cumulative multiplication.

$$\prod_{j=k+1}^{t} \frac{\partial h^{j}}{\partial h^{j-1}} = \prod_{j=k+1}^{t} \tanh' \cdot W_s$$

For the input text sequence, the vector representation of a word in the input text at each time step of the RNN, calculate the hidden state at the current time step, and then use it for the output of the current time step and pass it to the next time step and use it as the RNN unit input together with the word vector of the next word, and then calculate the hidden state of the RNN at the next time step, and so on... until every word in the input text is processed, which takes n time steps since the length of the input text is n.

LSTM

RNN has only short-term memory due to gradient disappearance. LSTM network combines short-term memory with long-term memory through gate control and solves the problem of gradient disappearance to some extent. Figure 3 shows the cell structure of LSTM, which mainly contains three gates (forget gate, input gate, output gate) and one memory cell. The horizontal line at the top of the box, called cell state, is responsible for controlling the information passed to the next moment.

We record f_t, c_t, and o_t as the forgetting gate, input gate, and output gate, respectively, represented by the sigmoid layer. The two tanh layers in the above figure then correspond to the input and output of the cell, respectively. LSTM generates a value between 0 and 1 from the output of the previous moment and the current input to decide how much of the information learned in the previous moment is retained C_{t-1}.

$$\sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

Similarly, LSTM uses the input gate to decide which values to update with and the tanh layer to generate new candidate

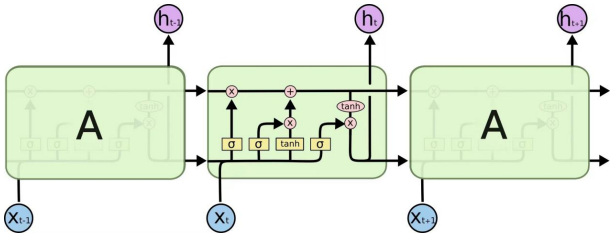

Figure 3: The cell structure of LSTM

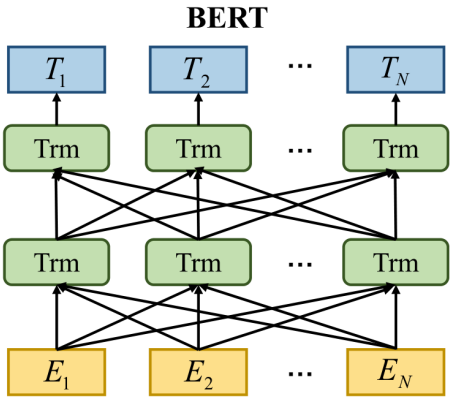

Figure 4: The structure of the Bert model

values

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right)$$

Combining the forgetting gate and the input gate generates the final candidate value

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM uses a sigmoid layer to get an initial output and uses tanh to scale the candidate values to between [-1,1]. Finally these two values are multiplied together to get the final output.

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM process the text sequence the same as a RNN as explained above in detail.

Bert

The structure of the Bert model is shown in Figure 4, which is a bidirectional Transformer block connection. The essence of Bert is to learn the context-based word vector representation and make good use of the bi-directional information of the sentence. The structure of each Transformer block of Bert is illustrated in Figure 5, which is actually the encoder structure of the Transformer

After the text is first represented as a word vector matrix, positional encoding information is added to each vector. The Transformer completely discards the RNN/CNN structure. It only consists of a multi-headed self-attentive mechanism

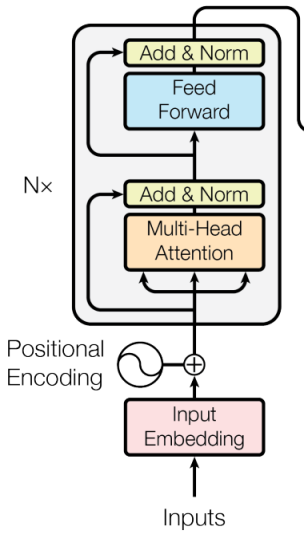

Figure 5: The structure of each Transformer block in Bert

and a feedforward neural network with a residual structure. The self-attentive mechanism is computed as follows[12]:

$$Attention(Q, K, V) = softmax\left(\frac{QK}{\sqrt{d_k}}\right)V$$

Q, K, and V are obtained by matrix multiplication of the encoded word vectors with three weight matrices W_Q, W_K, and W_V, respectively Multihead self-attention is to use multiple sets of weight matrices W_Q, W_K, W_V simultaneously to get multiple sets of Query, Keys, Values matrices For the text classification task, Bert only needs to take the final hidden state C of the first CLS tag, add a weight matrix W and then do Softmax to get the predicted label probability P.

Experiment

Experimental data introduction

IMDB is a dataset for binary sentiment classification with textual content of clearly polarized movie reviews. The dataset contains much more data than the previous benchmark dataset. It provides a set of 25,000 movie reviews for training and 25,000 for testing.

Training configuration information

In this experiment, since more than 80% of the sentences in the dataset are less than 320, we set the maximum length of sentences output to the neural network to 320 and the learning rate to 1e -5. Due to limited training resources, the epoch of learning is set to 4, train_batch_size to 16 and test_batch_size to 32, The weight_delay is 0.01.

Analysis of results

We first tested the performance of TextCNN, RNN, LSTM, and Bert separately. The experimental results are shown in Table 1. the performance of Bert significantly outperforms

id	Model Name	Accuracy rate
1	TextCNN	85%
2	RNN	87.72%
3	LSTM	89.63%
4	Bert	91.47%

Table 1: The performance of TextCNN, RNN, LSTM, and Bert.

id	Model Name	Accuracy rate
1	Bert+TextCNN	91.96%
2	Bert+RNN	92.86%
3	Bert+LSTM	92.99%
4	Bert+BiLSTM	92.44%
5	Bert+FNN	92.80%

Table 2: The performance of the combination of Bert and TextCNN, RNN, LSTM, BiLSTM and FNN .

several other networks indicating that a high level of text understanding can be accomplished with a dynamic vector representation of words.

Then we tested the combination of Bert and TextCNN, RNN, LSTM, BiLSTM and FNN respectively.The experimental results are shown in Table 2.

Experiments show that directly feeding Bert's semantic representation of the text to TextCNN brings little performance improvement afterwards. This indicates that directly feeding trained semantic information to CNNs cannot exploit the inductive bias of localization in CNNs. For humans, there is often a center of focus, and information within a certain range of this center is given stronger attention relative to other locations. Our attempt to combine Bert with TextCNN is also aimed at simulating this distance-centered attention mechanism in humans. However, for the trained word vectors processing text, even the inclusion of CNNs with local information interaction has little effect, suggesting that we should modify the multi-headed attention mechanism in the encoder structure when training the pre-trained mask language model. In the encoder, after each layer of Transformer block to get the attention result, the position with the largest attention weight is selected as the center, and the relative distance information between other positions and this position is added to the network, and then the Transformer block is passed again. The effect of adding FNN is not as good as adding RNN/BiLSTM/LSTM, because FNN itself is part of the transformer block in bert, and adding FNN is equivalent to deepening the number of model layers and enhancing the nonlinear fitting ability, and does not bring other information. The combination of Bert and LSTM reached the best in this experiment, with 92.99% accuracy, which has surpassed Roberta's 92.96 accuracy. This shows that combining the structure of selectively filtering historical information and absorbing current input leads to better semantic understanding for text sequences. This structural improvement is no less than designing more complex pre-training tasks to make the pre-trained model more capable of word vector representation.The combination of Bert and BiLSTM is not as effective as the combination of Bert and LSTM because Bert's representation of semantics in pretraining has already combined contextual bi-directional information, and BiLSTM then goes to do reverse information transfer may conflict with the hidden information learned by Bert, resulting in worse performance than LSTM instead.

Conclusion

This article verifies the performance improvement brought by the advantages of different network structures for text classification by comparing the performance of the backbone networks commonly used in deep learning so far on text classification and combining them with each other. It is demonstrated that network structures such as RNN/LSTM with recurrent nature of processing information by time step can be used with Transformer structured models directly spliced, while CNN direct splicing has less improvement effect. The combination of Bert and LSTM has the best effect, reaching 92.96% and surpassing Roberta, demonstrating that combining Transformer and LSTM backbone networks brings no less improvement than improving the semantic representation ability from the pre-training task.If one wants to combine the advantages of CNN and Transformer, one needs to combine on the attention structure of Transformer by connecting the convolutional feature map with the self-attentive feature map to enhance the convolutional operation to capture remote interactions. Alternatively, the backbone network of the model adopts a dual-stream network paradigm, where the Transformer branch and the CNN branch achieve higher-order information fusion through a well-designed interaction layer. A knowledge distillation scheme is also considered, using the CNN as a teacher model to guide the Transformer to converge faster and learn features with local attention capabilities of the CNN.

Future work

We found that the fusion of Transformer structures with CNN structures is more difficult and has more potential than RNN/LSTM, because Transformer structures tend to focus more on global information leading to the neglect of local feature details. In the future, we will try to modify the transformer structure using dual-stream networks to obtain information representations with both global attention and local feature details by fusing the information in the convolutional feature map and the self-attentive feature map, thus improving the performance on NLP tasks.

References

[1] Richard Socher et al. "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. URL: https://aclanthology.org/D13-1170.

[2] Daojian Zeng et al. “Relation Classification via Convolutional Deep Neural Network”. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, Aug. 2014, pp. 2335–2344. URL: https://aclanthology.org/C14-1220.

[3] Alex Hai Wang. “Don’t follow me: Spam detection in Twitter”. In: *2010 International Conference on Security and Cryptography (SECRYPT)* (2010), pp. 1–10.

[4] Sida I Wang and Christopher D Manning. “Baselines and bigrams: Simple, good sentiment and topic classification”. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2012, pp. 90–94.

[5] Siwei Lai et al. “Recurrent convolutional neural networks for text classification”. In: *Twenty-ninth AAAI conference on artificial intelligence*. 2015.

[6] Peng Zhou et al. “Attention-based bidirectional long short-term memory networks for relation classification”. In: *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)*. 2016, pp. 207–212.

[7] Yoon Kim. “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: https://aclanthology.org/D14-1181.

[8] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[9] Samira Pouyanfar et al. “A survey on deep learning: Algorithms, techniques, and applications”. In: *ACM Computing Surveys (CSUR)* 51.5 (2018), pp. 1–36.

[10] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[11] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. “Understanding of a convolutional neural network”. In: *2017 international conference on engineering and technology (ICET)*. Ieee. 2017, pp. 1–6.

[12] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[13] Hao Peng et al. “Large-scale hierarchical text classification with recursively regularized deep graph-cnn”. In: *Proceedings of the 2018 world wide web conference*. 2018, pp. 1063–1072.

[14] Liang Yao, Chengsheng Mao, and Yuan Luo. “Graph convolutional networks for text classification”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 7370–7377.