A Study on Improving ALBERT with Additive Attention for Text Classification

Zepeng Zhang, Hua Chen(✉), Jiagui Xiong, Jiayu Hu, and Wenlong Ni

Jiangxi Normal University, Nanchang 330022, Jiangxi, China
{gottenzzp,hua.chen,jiaguixiong,jyhu,wni}@jxnu.edu.cn

Abstract. The Transformer has made significant advances in various fields, but high computational costs and lengthy training times pose challenges for models based on this architecture. To address this issue, we propose an improved ALBERT-based model, which replaces ALBERT's self-attention mechanism with an additive attention mechanism. This modification can reduce computational complexity and enhance the model's flexibility. We compare our proposed model with other Transformer-based models, demonstrating that it achieves a lower parameter count and significantly reduces computational complexity. Through extensive evaluations on diverse datasets, we establish the superior efficiency of our proposed model over alternative ones. With its reduced parameter count, our proposed model emerges as a promising approach to enhance the efficiency and practicality of Transformer-based models. Notably, it enables practical training under resource and time limitations, highlighting its adaptability and versatility in real-world scenarios.

Keywords: Additive attention · ALBERT · Fastformer · NLP · Text classification

1 Introduction

Since the emergence of the Transformer architecture [1], it has garnered significant achievements across multiple domains. Models such as BERT [2], GPT [3–5], built upon this architecture, have become the standard benchmarks for numerous tasks in Natural Language Processing (NLP) and exhibited commendable performance. Before the advent of the Transformer, traditional structures such as Recurrent Neural Networks (RNNs) [6] and Convolutional Neural Networks (CNNs) [7] were widely used in NLP but suffered from issues like information loss and long-term dependencies. In contrast, the Transformer adopts an attention mechanism to concurrently process elements within a sequence, avoiding these issues and enabling better representation learning through pre-training. Besides NLP, researchers are increasingly exploring its applications in other fields, such as Computer Vision. For instance, Vision Transformer (ViT) [8] processes input image blocks as sequential data and has achieved impressive performance in diverse image classification tasks. Similarly, the DETR model [9] leverages the Transformer architecture for object detection tasks, outperforming traditional approaches

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
H. Lu et al. (Eds.): ACPR 2023, LNCS 14407, pp. 192–202, 2023.
https://doi.org/10.1007/978-3-031-47637-2_15

in accuracy. Additionally, the DALL-E model [10] excels at generating images from textual descriptions, while the Transformer-based Temporal Modeling (TTM) method effectively handles temporal and spatial information in video sequences.

Although models based on the Transformer architecture have achieved impressive results, the dot-product attention in their architecture requires matrix multiplication operations between two matrices $Q^{N \times d}$ and $\left(K^T\right)^{d \times N}$ (where N is the sequence length and d is the feature dimension) for similarity calculations, resulting in a complexity of $O\left(N^2 \cdot d\right)$. This leads to two significant issues: 1. High resource consumption during training; 2. Long training time. Without adequate GPU or TPU resources, it becomes challenging to train an effective model quickly or within limited computing resources, constraining low-resource researchers or companies from pursuing related tasks. To address these problems, researchers have proposed some solutions. For instance, DistBERT [11] employs knowledge distillation to learn from a teacher model, thereby reducing training time. Longformer [12], conversely, enhances the attention mechanism by utilizing local attention, specifically a sliding window mechanism, to compute attention between each word. This approach effectively reduces the time complexity associated with the traditional Transformer's dot-product attention, subsequently shortening the model's training time. While these methods have somewhat expedited training, the resources and time required remain prohibitive for many people.

In this paper, we propose an improved ALBERT [13] based model that replaces ALBERT's self-attention mechanism with an additive attention mechanism. This change achieves a linear reduction in time complexity and significantly reduces the model's parameter count by transitioning from dot product attention to additive attention. We conduct experiments on two text classification datasets to evaluate our model's performance. The results demonstrate that the proposed model outperforms several other models in terms of computational complexity and inference speed. The main contributions of this paper are: 1) proposing an improved ALBERT-based model that incorporates additive attention, reducing computational complexity and enhancing model flexibility; 2) conducting experiments on two text classification datasets, showcasing our model's strong competitiveness.

The rest of this paper is organized as follows: Sect. 2 reviews some related works. Section 3 provides a detailed introduction to our proposed approach. Experiments highlighting the merits of our approach are presented in Sect. 4. Section 5 concludes the paper and discusses future work.

2 Related Work

2.1 Efficient Transformer-Based Models

The Transformer architecture is known for its high time complexity and vast number of parameters, demanding considerable computational resources during training. To address this challenge, several approaches have been proposed to reduce the training time and resource requirements of Transformers. For instance, the Reformer model [14] groups attention scores with similar values into the same bucket, thereby reducing time complexity. Likewise, the Sparse-Transformer [15] divides the input text into fixed-length blocks and limits attention computations to these blocks, effectively diminishing the number of attention operations and further cutting down time complexity. Recently, the BigBird model [16] was introduced to address the computational challenges associated with processing long sequences. By employing a hierarchical block sparse attention mechanism that merges global attention with block-level random attention, BigBird significantly curtails the computational resources needed for processing long sequences.

2.2 ALBERT Model

ALBERT is a variation of BERT that introduces three significant improvements over the original BERT model:

1. ALBERT shares weight parameters across all layers of the model, leading to a reduction in the model's parameter count.
2. ALBERT effectively addresses the interdependency between the word embedding vector's dimension E and the hidden layer size H through factorization. This substantially reduces the model's parameter count, especially when $E \ll H$.
3. The Next Sentence Prediction (NSP) task in BERT has been found ineffective, leading to its removal from the model. Instead, the Sentence-Order Prediction (SOP) loss is used for training, resulting in improved performance in natural language processing tasks.

Benefiting from these improvements, ALBERT retains the same hyperparameter settings as BERT but has substantially fewer parameters—reducing from 108M in BERT to just 12M in ALBERT. Moreover, ALBERT's training speed is 1.7 times faster than that of BERT.

2.3 Self-attention and Additive-Attention Mechanism

The self-attention mechanism is a fundamental computational module of the Transformer architecture, allowing the model to capture contextual dependencies and calculate attention weights for contextual representations. Partitioning the self-attention mechanism into multiple heads enables the model to capture diverse types of contextual relationships. The main equations are shown below:

$$head_i = Attention(Q_i, K_i, V_i) = softmax\left(\frac{QW_i^Q\left(KW_i^K\right)^T}{\sqrt{d}}\right)VW_i^V \tag{1}$$

$$MultiHead(Q, K, V) = Concat(head_i, \ldots, head_s)W^O \tag{2}$$

where $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d \times d}$ is the learnable parameter matrix and W^O is a fully connected linear matrix. Q, K, V $\in \mathbb{R}^{N \times d}$ is the Query, Key and Value matrix for multiplying the input sequence X with W^Q, W^K, W^V. N is the sequence length and d is the hidden layer dimension.

The Fastformer [17] architecture incorporates the additive attention mechanism as its main computational module, building upon the self-attention mechanism of the Transformer. It first aggregates the query sequences into a global query vector and models their interaction with attention keywords using elementwise products. Next, it employs additive attention to aggregate the keywords into a global keyword vector and model their interaction with attention values. Linear transformations are then applied to learn the context-aware attention values added to the attention queries to generate the final output. This approach reduces the time complexity of attention computation from in the Transformer to linear complexity, significantly accelerating the training speed of the model.

3 Method

The overall framework of the proposed method is illustrated in Fig. 1. The key modification involves replacing the additive attention mechanism with the self-attention mechanism within the ALBERT model. This substitution results in improved computational speed for the whole framework.

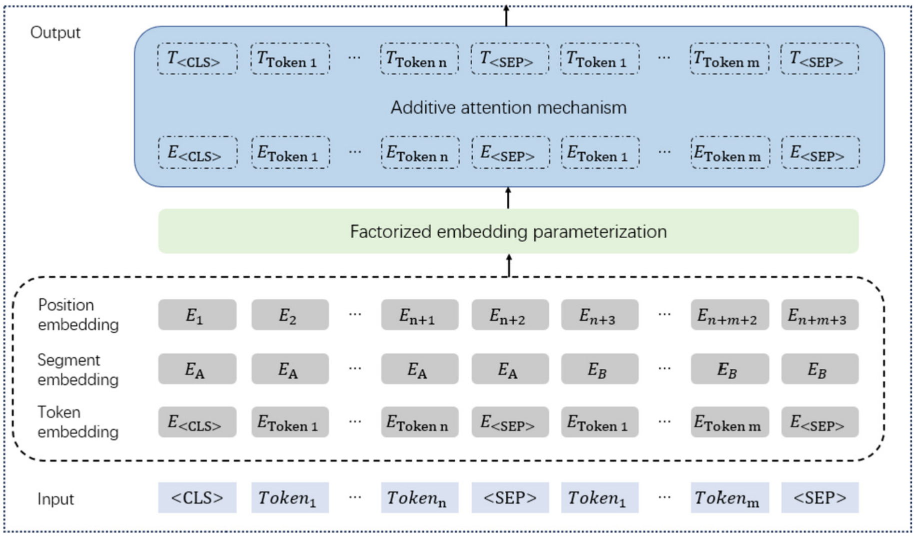
Output
$T_{<CLS>}$ $T_{Token 1}$ ··· $T_{Token n}$ $T_{<SEP>}$ $T_{Token 1}$ ··· $T_{Token m}$ $T_{<SEP>}$
Additive attention mechanism
$E_{<CLS>}$ $E_{Token 1}$ ··· $E_{Token n}$ $E_{<SEP>}$ $E_{Token 1}$ ··· $E_{Token m}$ $E_{<SEP>}$
Factorized embedding parameterization
Position embedding
E_1 E_2 ··· E_{n+1} E_{n+2} E_{n+3} ··· E_{n+m+2} E_{n+m+3}
Segment embedding
E_A E_A ··· E_A E_A E_B ··· E_B E_B
Token embedding
$E_{<CLS>}$ $E_{Token 1}$ ··· $E_{Token n}$ $E_{<SEP>}$ $E_{Token 1}$ ··· $E_{Token m}$ $E_{<SEP>}$
Input
<CLS> $Token_1$ ··· $Token_n$ <SEP> $Token_1$ ··· $Token_m$ <SEP>

Fig. 1. Framework of the proposed method.

3.1 Overview and Procedure

First, we define the input sequence as $T \in \mathbb{R}^{N}$, where N represents the number of tokens. Then, we perform word embedding on the sequence to obtain $E \in \mathbb{R}^{N \times ed}$, where N is the length of the input sequence and ed is the word embedding dimension. We also add positional embedding and sentence embedding vectors. Next, the sequence is projected into $E \in \mathbb{R}^{N \times d}$ using factorization in ALBERT, where d represents the hidden layer dimension. Specifically, E is represented as $E = [e_1, e_2, \ldots, e_N]$. Finally, the sequence E is processed using additive attention to compute the interactions between contexts and obtain feature vectors with contextual information. One of the computation graphs for additive attention is shown in Fig. 2.

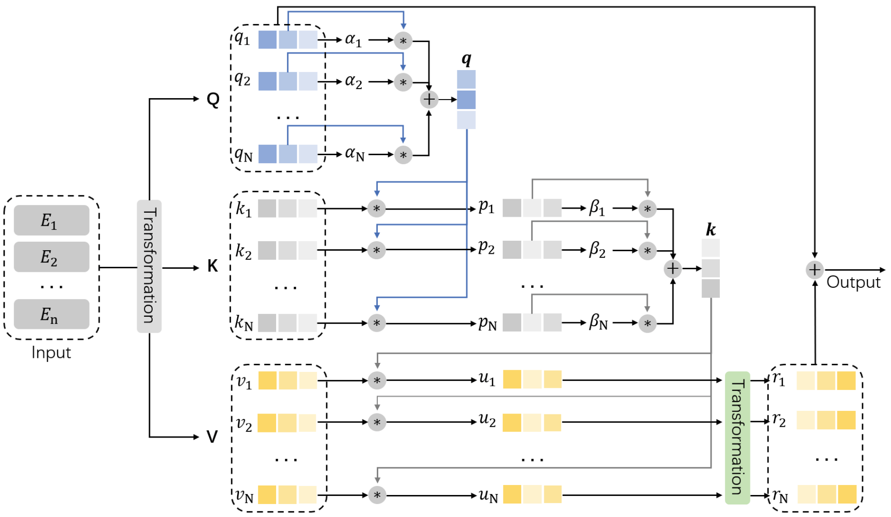

Fig. 2. Additive attention.

The input vector E is multiplied by $W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$ to obtain three matrices Q, K, V. By calculating the global query and key vectors and engaging in additive attention with the value, the final output is obtained. The computation is defined as follows:

$$\mathrm{q} = \sum\nolimits_{i=1}^{N} \alpha_i \mathrm{q}_i = \sum\nolimits_{i=1}^{N} \frac{\exp\left(W_q^T \mathrm{q}_i/\sqrt{d}\right)}{\sum_{j=1}^{N} \exp\left(W_q^T \mathrm{q}_j/\sqrt{d}\right)} \mathrm{q}_i \tag{3}$$

where $\mathrm{q} \in \mathbb{R}^d$ represents the global query vector obtained through additive attention to the q matrix, α_i denotes the attention weight for the i_{th} q vector, and $\mathrm{w_q} \in \mathbb{R}^d$ is the learnable parameter vector.

$$\mathrm{k} = \sum\nolimits_{i=1}^{N} \beta_i \mathrm{p}_i = \sum\nolimits_{i=1}^{N} \frac{\exp\left(W_k^T \mathrm{p}_i/\sqrt{d}\right)}{\sum_{j=1}^{N} \exp\left(W_k^T \mathrm{p}_j/\sqrt{d}\right)} \mathrm{p}_i \tag{4}$$

Since the interaction between the global q vector and each k_i needs to be modeled, $\mathrm{p_i} = \mathrm{q} * \mathrm{k_i}$ is computed through element-wise multiplication. The k matrix, encapsulating the global context, is obtained using the additive attention mechanism. The global $\mathrm{k} \in \mathbb{R}^d$ vector is defined as in the equation above, and $\mathrm{w_k} \in \mathbb{R}^d$ represents a learnable parameter vector. Similarly, the global key and each value vector are multiplied between elements to compute the vector of key-value interactions $\mathrm{u_i} = \mathrm{k} * \mathrm{v_i}$. The resulting vectors are passed through a linear transformation layer to learn the hidden representations.

3.2 Parameter and Computational Complexity Analysis

In this section, we analyze the computational complexity of our model. In Transformer-based models, the self-attention mechanism is standard, and the complexity of calculating attention weights in self-attention is $O\left(N^2 \cdot d\right)$. Conversely, the time and space requirements for learning global query vectors and key vectors in additive attention networks are $O(N \cdot d)$, and the time and space costs of element-wise multiplication are also $O(N \cdot d)$.

In addition, the total number of parameters for additive attention networks and element-wise multiplication is $2hd$, where h is the number of attention heads. At last, we can obtain the parameter size of each layer in our model as $3d^2 + 2hd$ (the sum of the parameters of the two weight matrices W^Q, W^K is $2d^2$; the parameter size of the output transformation matrix is d^2; and adding the $2hd$ mentioned above, the parameter size of each layer is $3d^2 + 2hd$). Moreover, because we use weight sharing for all layers of the model, the total parameter size of our model can also be approximated as $3d^2 + 2hd$, which is fewer parameters compared to the Transformer model that has at least $4d^2$ parameters per layer (encompassing the three weight matrices W^Q, W^K, W^V, the output transformation matrix; excluding bias terms and layer normalization). In conclusion, these analytical results underscore the theoretical efficiency of the method proposed in this paper.

4 Experiments

4.1 Experimental Setup

Our training was conducted on an NVIDIA RTX3090 graphics card, boasting 24 GB of video memory. The system operated on a Windows 10 64-bit platform, with Python version 3.7 and PyTorch version 1.10.1.

We initialized the embedding matrix of our models using Glove [18] word embeddings. This approach allowed us to harness pre-trained word representations, facilitating the capture of intricate semantic relationships within our dataset. To ensure consistent and reliable results across experiments, we set the random seed to 2023. Moreover, each experiment was conducted three times to account for potential variability, with the average performance being reported.

Throughout the training process, we maintained a consistent batch size of 64 for all models. This size indicates the number of training examples processed during each neural network training iteration. For optimization, we chose the AdamW optimizer, renowned for its efficacy in refining deep neural networks. The specific parameters of our hyperparameter settings are detailed in Table 1.

Baselines. We selected four models as our baselines:

Table 1. Hyperparameter setting.

Method	IMDB	Sentiment140
Encoder layer	12	12
Decoder layer	–	–
Window size (Longformer, Bigbird)	8	16
Block length (Bigbird)	4	8
Attention head	12	12
Maximum text length	512	512
Hidden dimension	768	768
Loss	Crossentropy	Crossentropy
Batch size	64	64
Optimizer	AdamW	AdamW
Learning rate	1e−4	1e−4
Epochs	3	3
Dropout	0.1	0.1

Longformer. The Longformer model, a pre-trained language model built on the Transformer architecture, excels in handling lengthy texts. It distinguishes itself from conventional Transformer models by incorporating a novel attention mechanism known as sliding window attention. This mechanism enables the Longformer to sustain high computational efficiency while processing long texts. Furthermore, the Longformer model introduces a new position encoding method called global attention mask. This enables the model to have a more comprehensive understanding of the context in long texts.

Bigbird. The BigBird model is also a pre-trained language model based on the Transformer architecture, which is designed to handle even longer texts than the Longformer model. It introduces a novel sparse attention mechanism that enables the model to focus on a small subset of input tokens during each step while effectively capturing global dependencies across the entire input sequence. Moreover, the BigBird model incorporates a novel position encoding technique known as random attention spans, enhancing its proficiency in handling long texts.

DistilBERT. DistilBERT is an optimized and streamlined version developed from the BERT model. It achieves a more petite model size and faster inference speed through distillation. This approach involves training a compact model to mimic the behavior and knowledge of a larger model. By distilling the essential information from the larger model, DistilBERT retains comparable language understanding capabilities while significantly reducing computational requirements.

ALBERT. ALBERT is based on a simplified version of the BERT model, which addresses the issue of model size and computational efficiency by utilizing parameter sharing and weight matrix factorization techniques. ALBERT maintains the effectiveness of BERT while significantly reducing its size and improving processing speed. This makes it a practical choice for various natural language processing tasks, providing a balance between performance and efficiency.

4.2 Datasets and Evaluation Methodology

To evaluate the performance of our model, we carried out experiments on two text classification datasets: IMDB [19] and Sentiment140 [20]. The IMDB dataset focuses on movie rating prediction, whereas the Sentiment140 dataset is made up of 16,000 English tweets, each categorized as positive, negative, or neutral. This dataset is extensively used for training and evaluating sentiment analysis models. The detailed information of these datasets is showcased in Table 2.

Table 2. Datasets for Text Classification.

Dataset	Train	Val	Test	Avg. len	Class
IMDB	25k	12.5k	12.5k	98.3	2
Sent.140	128.8k	16.1k	16.1k	505.4	2

The reasons for choosing these two datasets are as follows. The IMDB dataset contains short text samples, while the Sentiment140 dataset comprises long text samples. This allows us to evaluate the model's performance on both short and long texts separately.

For text classification, accuracy and the F1 score stand out as pivotal metrics of evaluation. These metrics can be calculated using formulas 5–8, where TP (True Positive) represents correctly predicting positive samples, TN (True Negative) represents correctly predicting negative samples, FP (False Positive) represents incorrectly predicting negative samples as positive, and FN (False Negative) represents incorrectly predicting positive samples as negative. Precision and recall, calculated as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

The F1-score is computed using accuracy and recall rates, as illustrated in the subsequent formula:

$$F1 = 2\frac{Precision \times Recall}{Precision + Recall} \tag{8}$$

4.3 Comparison of Parameter Numbers and Inference Speed

Table 3 provides a comparison of parameter counts and inference speeds across different models. From the table, it's clear that our model stands out in terms of complexity, parameter counts, and inference speed. Remarkably, our proposed model boasts an impressive 109.7% enhancement in inference speed when pitted against the ALBERT model, eclipsing other models with an acceleration surpassing 222.1%. Furthermore, when juxtaposed with alternative models, our model's complexity remains linear, translating to faster training times and augmented efficiency.

Table 3. The comparison of parameter numbers and inference speed for different models.

Method	Complexity	Parameter numbers	Inference speed
Longformer	$O(N \cdot k \cdot d)$	110M	5.405
Bigbird	$O(N \cdot k \cdot d)$	137M	3.823
DistilBERT	$O\left(N^2 \cdot d\right)$	66M	1.475
ALBERT	$O\left(N^2 \cdot d\right)$	12M	3.520
Our proposed approach	$O(N \cdot d)$	9.5M	1.678

4.4 Experimental Results

Table 4. Experimental results of various models.

Method	IMDB		Sentiment140	
	Accuracy	F1	Accuracy	F1
Longformer	0.850	0.850	0.781	0.781
Bigbird	0.843	0.842	0.783	0.783
DistilBERT	0.847	0.846	0.792	0.792
ALBERT	0.881	0.880	0.839	0.838
Our proposed approach	0.852	0.851	0.825	0.824

As illustrated in Table 4, our proposed approach showcases exemplary performance on both short and long text datasets. Moreover, as indicated in Table 3, our method registers only a slight performance decrement, approximately 1.7%, when compared to ALBERT. Yet, it marks a notable surge of 109.7% in inference speed. These findings underscore the significant benefits of our proposed method, particularly with its incorporation of additive attention, especially in terms of inference speed.

5 Conclusions and Future Work

In this paper, we present an improved ALBERT-based model that replaces ALBERT's self-attention mechanism with additive attention. This additive attention transforms the query matrix into a global query vector. This vector then collaborates with each key vector to generate a global key vector enriched with contextual information. Using the additive attention mechanism, this global key vector interacts with all value vectors, leading to the derivation of the final global attention value. The output from this last attention layer is achieved by melding it with the query matrix using residual concatenation. Our experiments, conducted on two distinct text classification datasets, indicate that our proposed model demands considerably fewer training resources than most Transformer-based counterparts, translating into expedited training durations.

In the future, we plan to pre-train the proposed model to ensure more nuanced context modeling. Additionally, we also envisage integrating a decoder component to fortify tasks related to extended sequence text generation.

References

1. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
3. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
4. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog **1**(8), 9 (2019)
5. Brown, T.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901 (2020)
6. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)
7. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv: 1408.5882 (2014)
8. Dosovitskiy, A., et al.: An image is worth 16 $\times$ 16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
9. Carion, Nicolas, Massa, Francisco, Synnaeve, Gabriel, Usunier, Nicolas, Kirillov, Alexander, Zagoruyko, Sergey: End-to-end object detection with transformers. In: Vedaldi, Andrea, Bischof, Horst, Brox, Thomas, Frahm, Jan-Michael. (eds.) ECCV 2020. LNCS, vol. 12346, pp. 213–229. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_13
10. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125 (2022)
11. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019)
12. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: the long-document transformer. arXiv preprint arXiv:2004.05150 (2020)
13. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: ALBERT: a lite BERT for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019)

14. Kitaev, N., Kaiser, Ł., Levskaya, A.: Reformer: the efficient transformer. arXiv preprint arXiv:2001.04451 (2020)
15. Zhao, G., Lin, J., Zhang, Z., Ren, X., Sun, X.: Sparse transformer: concentrated attention through explicit selection (2019)
16. Zaheer, M., et al.: Big bird: transformers for longer sequences. Adv. Neural. Inf. Process. Syst. **33**, 17283–17297 (2020)
17. Wu, C., Wu, F., Qi, T., Huang, Y., Xie, X.: Fastformer: additive attention can be all you need. arXiv preprint arXiv:2108.09084 (2021)
18. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543, October 2014
19. Maas, A., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: 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 (2011)
20. Go, A., Bhayani, R., Huang, L.: Twitter sentiment classification using distant supervision. CS224N project report, Stanford, 1(12) (2009)