ClusterFormer: Neural Clustering Attention for Efficient and Effective Transformer

Ningning Wang[1*], **Guobing Gan**[1*], **Peng Zhang**[1†], **Shuai Zhang**[1], **Junqiu Wei**[2], **Qun Liu**[3], **Xin Jiang**[3]
[1]College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]The Hong Kong Polytechnic University, China
[3]Huawei Noah's Ark Lab, China
{w_ning1215,ganguobing,pzhang,szhang96}@tju.edu.cn
junwei@polyu.edu.hk,{qun.liu,Jiang.Xin}@huawei.com

Abstract

Recently, a lot of research has been carried out to improve the efficiency of Transformer. Among them, the sparse pattern-based method is an important branch of efficient Transformers. However, some existing sparse methods usually use fixed patterns to select words, without considering similarities between words. Other sparse methods use clustering patterns to select words, but the clustering process is separate from the training process of the target task, which causes a decrease in effectiveness. To address these limitations, we design a neural clustering method, which can be seamlessly integrated into the Self-Attention Mechanism in Transformer. The clustering task and the target task are jointly trained and optimized to benefit each other, leading to significant effectiveness improvement. In addition, our method groups the words with strong dependencies into the same cluster and performs the attention mechanism for each cluster independently, which improves the efficiency. We verified our method on machine translation, text classification, natural language inference, and text matching tasks. Experimental results show that our method outperforms two typical sparse attention methods, Reformer and Routing Transformer while having a comparable or even better time and memory efficiency.

1 Introduction

Transformer (Vaswani et al., 2017) has been widely used and achieved state-of-the-art results in a variety of NLP tasks such as neural machine translation (Bahdanau et al., 2015), text classification, etc. Its good effectiveness benefits from its core component Self-Attention Mechanism which can capture global dependencies well. However, the large calculation and memory cost limit the further application of Transformer on long sequence tasks due to the complexity of $O(N^2d)$ of Self-Attention. As a result, many research works have been carried out to improve the efficiency of Transformer (Tay et al., 2020b). These efficient Transformers can be roughly divided into two categories: approximation-based (Tay et al., 2020a; Katharopoulos et al., 2020) and sparse pattern-based methods (Qiu et al., 2020; Ho et al., 2019; Beltagy et al., 2020; Liu et al., 2018).

Regarding approximation-based methods, some works are based on low-rank approximation (Tay et al., 2020a; Wang et al., 2020) while others are based on kernels (Katharopoulos et al., 2020; Choromanski et al., 2020). Specifically, Linformer (Wang et al., 2020) adopts a low-rank approximation idea and projects the length dimension of keys and values to a lower dimension ($N \to k$). This reduces the complexity to $O(Nkd)$. However, the projection matrix in this method requires that all input sequences must be filled to the same length N, which makes it cannot handle the variable-length sequence well. Linear Transformer (Katharopoulos et al., 2020) uses kernels and the associative property of matrix multiplication to linearize the softmax attention, which reduces the complexity to $O(Nd^2)$. However, the approximation error to the softmax matrix in Self-Attention can be large in some cases (Xiong et al., 2021).

Sparse pattern-based methods introduce *sparse patterns* into the attention mechanism and limit the number of key vectors that the query vector should pay attention to. Prior work (Child et al., 2019; Qiu et al., 2020; Ho et al., 2019; Beltagy et al., 2020; Liu et al., 2018) proposed to use *fixed* sparse patterns to improve the efficiency of Self-Attention. For example, Sparse Transformer (Child et al., 2019) restricts the query to focus only on keys that are nearby or at fixed intervals. Such fixed sparse patterns do not consider the similarity between the query and different keys, and directly filter keys according to their location, which re-

[*]The first two authors contributed equally
[†]Corresponding Author

sults in a degradation of the model effectiveness. More recently, the *clustering* sparse patterns are proposed. Reformer (Kitaev et al., 2020) and Routing Transformer (Roy et al., 2021) use Locality Sensitive Hashing (LSH) and K-Means algorithms, respectively, to divide the words in the sequence into different clusters, and then the attention operation is restricted within each cluster independently. In this way, they reduce the complexity to $O(N \log Nd)$ and $O(N\sqrt{N}d)$, respectively. However, in Reformer and Routing Transformer, both LSH and K-Means only play the role of cluster partitioning, but run separately from the attention network training. In addition, these two methods also have the problem of inconsistency in the similarity measure between the clustering and attention operation. LSH and K-Means respectively use the hash value obtained by random projections and the negative Euclidean distance as the similarity measure between the input vectors while the attention operations use the inner product. Therefore, such a sparse pattern idea often results in the reduced effectiveness.

To address the reduced effectiveness issue of many efficient Transformers, especially sparse pattern-based methods, we propose Neural Clustering Method to learn the sparse pattern of the Attention. It can be seamlessly integrated into neural networks for joint training and optimization. In our method, the cluster center (centroid) is updated by a weighted sum of all word hidden states. At the same time, the members of clusters are divided according to the subordinate matrix of the centroids and word hidden states. The optimization of the clustering loss can guide the representation of word hidden states while learning cluster centroids. The integration of the neural clustering method and attention training enables our Neural Clustering Attention to perform better than previous clustering-based sparse attention mechanisms. Our Neural Clustering Method is a general clustering method, in the sense that in addition to being integrated into the network of the specific task, it is can handle clustering tasks alone.

Our overall model is called ClusterFormer and it is obtained by replacing the Self-Attention Mechanism in Transformer with Neural Clustering Attention Mechanism. In order to validate the benefits of ClusterFormer, we have carried out comparison experiments of the efficiency and effectiveness respectively. For efficiency, we provide a detailed analysis about the time and memory on dataset 20NEWS of text classification. Results show that our model has a comparable or even better efficiency compared with two typical sparse attention models, Reformer and Routing Transformer. Especially, when the sequence length exceeds 2000, our model, Routing Transfomer and Reformer reduce the memory of Transformer by 53.8%, 60.8% and 31.8% while reducing the training time by 51.4%, 41.8% and 14.4%, respectively on GPU. For effectiveness, we test it on machine translation, text classification, natural language inference, and text matching tasks. Experimental results show that on all tasks, our model consistently outperforms Reformer and Routing Transformer. In particular, our method improves the accuracy by 15.6% and 7.2% on SciTail datasets of natural language inference task compared with Reformer and Routing Transformer, respectively.

The major contributions of our work are as follows:

- We propose a general end-to-end fuzzy clustering method based on neural network, named Neural Clustering Method, which can dynamically learn weights of each word and then update centroids by weighting all the input words along with the training of specific tasks.
- We design the Neural Clustering Attention Mechanism based on our proposed clustering method to refactor Self-Attention Mechanism. The experimental results show that our method has comparable efficiency and better effectiveness than typical sparse attention models.

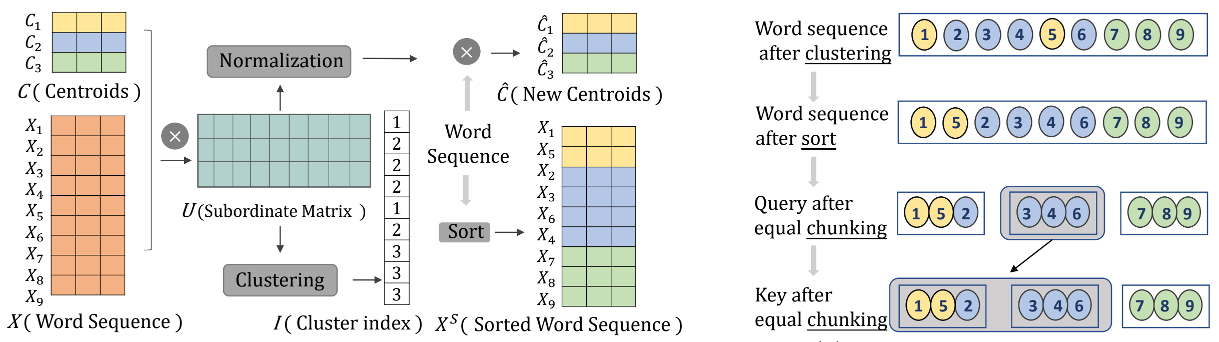
2 Related Work

2.1 Self-Attention Mechanism

The Self-Attention is the core component of Transformer (Vaswani et al., 2017). It extracts sequence features by processing the interaction of three sequence matrices Q, K, and V. Referring to the standard Transformer, its function can be written as follows:

$$\begin{gathered} Attention(Q,K,V) = Softmax(\frac{QK^T}{\sqrt{d}})V \\ Q,K,V = XW^Q, XW^K, XW^V \end{gathered} \tag{1}$$

where $X \in \mathbb{R}^{N \times d_{model}}$, Q, $K, V \in \mathbb{R}^{N \times d}$, $W^Q, W^V \in \mathbb{R}^{d_{model} \times d}$, N is the length of the se-



(a) Neural Clustering Method (b) Sorting and Chunking Operation

Figure 1: (a) is the Neural Clustering Method. Its main idea is to update centroids by weighting all the word hidden states and divide clusters according to the subordinate matrix. (b) is the sorting and chunking operations. It aims to make each cluster perform the attention mechanism in parallel.

quence, d_{model} is the dimensionality of the model, and d is the dimensionality of the attention head. In Self-Attention Mechanism, the interaction of Q and K gives the $N \times N$ attention (weight) matrix, and it leads to the complexity of $O(N^2 d)$, which has been one of the crucial limitation of Transformer.

2.2 Sparse variants of Transformer

Transformer has been developed into many variants to reduce the complexity of the attention mechanism. In these works, one of the main research directions is to use a sparse attention to substitute the quadratic-cost attention.

Some early works (Qiu et al., 2020; Ho et al., 2019; Beltagy et al., 2020) have been proposed to reduce the time complexity by restricting every query to focus only on keys that are nearby or at fixed intervals. This method fixes the sparsity pattern without considering the similarity between queries and keys, limiting its ability to assemble critical information from large contexts. Different from these works, our method attempts to automatically aggregate critical keys for each query based on dependency relationships.

Moreover, the clustering-pattern methods were used in Self-Attention to implement a sparse attention. For example, Reformer (Kitaev et al., 2020) and Routing Transformer (Roy et al., 2021) introduce Locality-Sensitive Hashing and K-Means algorithms, respectively, to reduce complexity to $O(N \log N d)$ and $O(N\sqrt{N}d)$. However, in this kind of method, the clustering process and training process are separate, which is a limitation in improving effectiveness. Based on previous researches, we proposed a novel Neural Clustering Method, which can be seamlessly integrated into the network of specific tasks for joint training and optimization to improve effectiveness.

3 Model

In this section, we first introduce our Neural Clustering Method. Then, we introduce our Neural Clustering Attention Mechanism which combines our clustering method and the Self-Attention Mechanism.

3.1 Neural Clustering Method

As shown in Figure 1 (a), our clustering method takes word hidden states $X \in \mathbb{R}^{N \times d}$ and centroid hidden states $C \in \mathbb{R}^{k \times d}$ as inputs. C is initialized randomly in the first layer. Then, we can get the subordinate (similarity) matrix U between word vectors and centroid vectors. It can be defined as:

$$U_{ij} = \frac{exp(\phi(C_i, X_j W^C))}{\sum_{j=1}^{N} exp(\phi(C_i, X_j W^C))} \quad (2)$$
$$1 \leq i \leq k\,, \quad 1 \leq j \leq N$$

where k is the number of clusters and N is the length of sequence. $W^C \in \mathbb{R}^{d_{model} \times d_{model}}$ is a parameter matrix. $\phi(\cdot)$ is a similarity measure function and it is the inner product operation in this scenario. X_j is the *j*-th row of matrix X and C_i is the *i*-th row of matrix C. The subordinate value $U_{ij} \in [0, 1]$ is the normalized similarity value between the *i*-th centroid vector and emphj-th word vector, and it represents the degree of the word X_j belonging to the centroid C_i.

Then, we get the updated centroids by weighting all the word hidden states. The corresponding formula is as follows:

$$\hat{C}_i = \sum_{j=1}^{N} U_{ij} X_j W^C \quad 1 \leq i \leq k \quad (3)$$

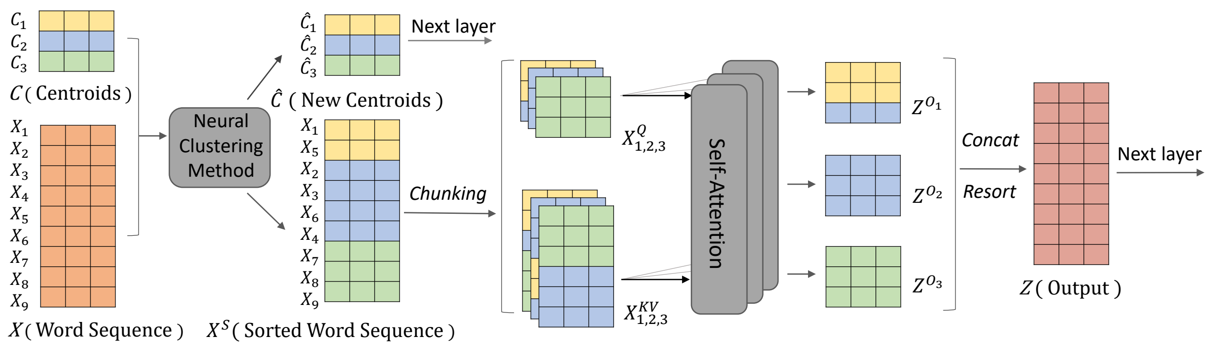


Figure 2: Neural Clustering Attention Mechanism. First, we group vectors with strong dependency into the same cluster and get a new word sequence X^S. Second, we carry out an equal chunking operation to get some block sequences, and then perform the attention within each cluster separately.

where i and j represent the index value of the centroid and word, respectively. Then we group the word vectors according to the subordinate matrix U, as follows:

$$I_j = Argmax(U_{:j}) \quad 1 \le j \le N \tag{4}$$

where $U_{:j}$ is the j-th column of the matrix U and function $Argmax(\cdot)$ assigns word hidden states to the corresponding cluster according to the maximum subordinate value. Therefore, $I \in \mathbb{R}^N$ represents the cluster index of all the word hidden states. Then, we sort the word vectors according to the cluster indexes I, as follows:

$$X^S,\ I^{'} = Sort(X, I) \tag{5}$$

where the function $Sort(\cdot)$ is used to arrange word hidden states belonging to the same cluster to adjacent positions in ascending order of cluster index. $X^S \in \mathbb{R}^{N \times d_{model}}$ is the sorted word vectors. $I^{'} \in \mathbb{R}^N$ is used to record the original positions of shuffled word hidden states in the sequence and will be used in Eq. 10. Through the above process, we get the grouped and sorted word hidden states X^S, as shown in Figure 1 (a).

Clustering Loss: Clustering Loss (L_1) is the mean of the negative similarity scores of word hidden states and their belonging centroids, and it will give guidance to learn the optimal clustering scheme. It is defined as follows:

$$L_1 = -\frac{1}{N}\sum_{j=1}^{N} \phi\left(X_j, \widehat{C}_{I_j}\right) \tag{6}$$

where X_j, $\widehat{C}_{I_j}$ represent the j-th word hidden state in the sequence and the updated centroid. The function $\phi(\cdot)$ is a similarity measure function and needs to be consistent with Eq. 2.

From the above analysis, our Neural Clustering Method is based on the soft clustering. There is a subordinate value between each pair of word vectors and centroid vectors, which can quantitatively describe the fuzzy relationship, so that the clustering can be carried out objectively and accurately. In addition, Neural Clustering Method is based on the neural network, which is easy to integrate into the network corresponding to the target task. The reconstruction of centroid vectors depends on all the word vectors and is based on the continuous optimization for the clustering objective function (as shown in Eq. 6) and the task-specific objective function to get better effectiveness.

In addition, we carried out a clustering comparison experiments between our method and traditional clustering methods and observed improvements of our method in effectiveness. See Appendix A for more details.

3.2 Neural Clustering Attention Mechanism

As described in Section 3.1, our Neural Clustering Method groups word vectors with strong dependency into the same cluster and outputs the sorted word vectors X^S. Then, we use different matrices to project X^S into matrix Q^S, K^S, and V^S, as follows:

$$Q^S, K^S, V^S = X^S W^Q, X^S W^K, X^S W^V \tag{7}$$

where W^Q, W^K and $W^V \in \mathbb{R}^{d_{model} \times d}$ are weight matrices. Q^S, K^S and V^S are matrices Query, Value and Key, respectively.

The number of members in each cluster may not be uniform, which makes it difficult for all clusters to perform the attention mechanism in parallel. For parallel computing, after arranging word hidden states in the same cluster to be in adjacent positions, we chunk them into equal blocks in order, as shown in Figure 1 (b) (essentially similar to the masking of Reformer). The process can be written as follows:

$$\begin{aligned} Q^{O_i} &= Q^S\Big((i-1)\left\lceil\frac{N}{k}\right\rceil : i\left\lceil\frac{N}{k}\right\rceil\Big] \\ K^{O_i} &= K^S\Big((i-2)\left\lceil\frac{N}{k}\right\rceil : i\left\lceil\frac{N}{k}\right\rceil\Big] \quad 1 \le i \le k \end{aligned} \tag{8}$$

where $Q^{O_i} \in \mathbb{R}^{w\times d}$ and $K^{O_i} \in \mathbb{R}^{2w\times d}$ are the i-th Query block and Key block respectively. w ($w = \frac{N}{k}$) is the number of members in each block. Matrix V^{O_i} has operations similar to K^{O_i}. After chunking, Query contains one sequence block while Key and Value consist of two contiguous blocks, which corresponds to L_2 mentioned in Eq. 11. Each token in Query focuses on two blocks of tokens so that the query can cover the words in the same cluster as much as possible. Of course, it does not have to be 2, and can be adjusted.

Then, we perform the attention operation within the sequence block in parallel and concatenate the output of each block.

$$\begin{aligned} Z^{O_i} &= \mathit{Attention}(Q^{O_i}, K^{O_i}, V^{O_i}) \\ Z^O &= \mathit{Concat}(Z^{O_1}, ..., Z^{O_k}) \qquad 1 \le i \le k \end{aligned} \tag{9}$$

where $Z^{O_i} \in \mathbb{R}^{w\times d}$ and $Z^O \in \mathbb{R}^{N\times d}$. Z^{O_i} is the output of the i-th sequence block after the attention operation.

Finally, we recover the shuffled sequence (output) to obtain the final result, as follows:

$$Z = \mathit{Resort}(Z^O, I^{'}) \tag{10}$$

where the function $\mathit{Resort}(\cdot)$ aims to recover shuffled sequence according to the original position record vector $I^{'}$ obtained from the Eq. 5. $Z \in \mathbb{R}^{N\times d}$ is the output of Neural Clustering Attention.

For the autoregressive modeling, we provide a Masked Neural Clustering Attention Mechanism to prevent the leftward information flow. More details can be found in Appendix B.

Centroid Sorting Loss: Centroid Sorting Loss (L_2) is the mean of the negative similarity scores of the adjacent centroid pairs. In Eq. 8, each token in Query block is expected to focus on two continuous blocks of tokens. L_2 makes word hidden states belonging to adjacent clusters are also close to each other. It is defined as follows:

$$L_2 = -\frac{1}{k}\left(\left(\sum_{i=2}^{k}\phi\left(\widehat{C}_i, \widehat{C}_{i-1}\right)\right) + \phi\left(\widehat{C}_1, \widehat{C}_k\right)\right) \tag{11}$$

where k is the number of centroids, $\widehat{C}_i$ is the i-th updated centroid, and the meaning of $\phi(\cdot)$ is consistent with Eq. 6.

In our method, Clustering Loss, Centroid Sorting Loss, and the loss of target tasks of the model are assigned different weights for joint optimization. More details can be found in Appendix C.

3.3 Analysis of Complexity

The complexity of Neural Clustering Attention Mechanism comes from two parts: (i) Neural Clustering Method. In this part, we need to calculate the subordinate matrix between centroid hidden states $C \in \mathbb{R}^{k\times d}$ and the word hidden states $X \in \mathbb{R}^{N\times d}$, referring to the Eq. 2, which leads to the complexity of $O(Nkd)$. (ii) Attention Mechanism. For this part, we compute attention within the Query block ($\in \mathbb{R}^{k\times w\times d}$) and Key block ($\in \mathbb{R}^{k\times 2w\times d}$), referring to the Eq. 9, which leads to the complexity of $O(kw^2d)$ where $w = \frac{N}{k}$. In summary, the overall complexity is $O(Nkd + kw^2d)$. When k is set to $\sqrt{N}$, the complexity is approximately $O(N\sqrt{N}d)$.

4 Experiments

In order to verify the effectiveness and efficiency of our method, we carried out the following tasks. We choose Transformer and its clustering-pattern variants (Reformer, Routing transformer) as baseline models. The implementations of the attention layer of Reformer and Routing transformer refer to the open source codes [1][2]. For a fair comparison, our proposed method and baseline models have the same architecture, except for the attention layer.

4.1 Machine Translation

We validate our model on IWSLT14 German-English and WMT14 English-German benchmarks, which have been widely used for machine translation tasks. For IWSLT14 De-En, it contains about 160K training sentence pairs and is pre-processed by using prepare-iwslt14en2de.sh [3]. For WMT14 En-De, it contains about 4.5 million training sentence pairs and it is pre-processed by using prepare-wmt14en2de.sh [4]. We use the BLEU score as the effectiveness evaluation metric. Some hyperparameters are set: the number of encoder and decoder

[1]https://github.com/lucidrains/reformer-pytorch

[2]https://github.com/lucidrains/routing-transformer

[3]https://github.com/pytorch/fairseq/blob/master/examples/translation/prepare-iwslt14.sh

[4]https://github.com/pytorch/fairseq/blob/master/examples/translation/prepare-wmt14en2de.sh

Model	IWSLT14 De-En	WMT14 En-De
Transformer† (Vaswani et al., 2017)	34.4	27.3 / 26.4
Reformer† (Kitaev et al., 2020)	34.0	26.3 / 25.4
Routing Transformer† (Roy et al., 2021)	32.5	24.3 / 23.6
ClusterFormer	**34.9**	**27.4 / 26.5**

Table 1: Test BLEU on IWSLT14 (De-En) and WMT14(En-De). For IWSLT14, we reported Tokenized BLEU results. For WMT14, we reported Tokenized BLEU and SacreBLEU results, arranged on the left and right. "†" indicates that the results of the model are our implementations.

Model	CR	MR	SUBJ	MPQA	20NEWS	Average
DiSAN (Shen et al., 2018)	84.8	–	94.2	90.1	–	–
MPSAN (Dai et al., 2020)	85.4	–	94.6	**90.4**	–	–
Transformer† (Vaswani et al., 2017)	86.2	81.8	95.4	89.9	83.6	87.38
Reformer† (Kitaev et al., 2020)	83.0	79.7	94.7	88.6	81.7	85.54
Routing Transformer† (Roy et al., 2021)	80.1	78.8	94.3	81.2	81.3	83.14
ClusterFormer	**88.1**	**82.7**	**96.2**	**90.4**	**83.8**	**88.24**

Table 2: Experimental results (Accuracy) on text classification tasks. "–" means that results are not reported.

layers L = 6, the number of centroids k = 3. The dimension of word embedding and model d_{model} = 512. Specifically, for IWSLT14, the number of heads is set to 4 and d_{ff} = 1024. For WMT14, the number of heads is set to 8 and d_{ff} = 2048.

As shown in Table 1, our method boosts effectiveness on both datasets. Specifically, the Tokenized BLEU score is improved by at least 1.5% compared with other models on IWSLT14 datasets. Compared with the latest models Reformer and Routing Transformer, ClusterFormer respectively has 2.6% and 7.4% improvement. Our method shows the same trend on WMT14 datasets. Especially, compared with Reformer and Routing Transformer, the Tokenized BLEU score of ClusterFormer respectively has 4.2% and 12.8% improvement and the sacreBLEU score respectively has 4.3% and 12.3% improvement.

4.2 Text Classification

We validate our model on five text classification tasks. CR (Hu and Liu, 2004): Customer reviews composed of positive or negative product reviews; MR (Pang and Lee, 2004): Movie reviews divided into positive and negative categories; SUBJ: Subjectivity dataset where the target is to classify a text as being subjective or objective; MPQA (Wiebe et al., 2005): Opinion polarity detection subtask. 20NEWS: A international standard dataset for text classification, text mining, and information retrieval research. The dataset collects about 20,000 newsgroup documents, divided into a collection of newsgroups on 20 different topics. Accuracy is used as the evaluation metric for these datasets. In addition, for all datasets, word embeddings are initialized by GloVe (Pennington et al., 2014) with 300-dimension. Some hyperparameters are set: The number of encoder layers L = 2, the dimension of model d = 300, the number of heads h = 4, and the number of centroids k is adjusted near the square root of the max length.

As shown in Table 2, ClusterFormer outperforms all baseline models and improves the test accuracy by at least 3.16%, 1.70% for CR and SUBJ datasets, respectively. In addition, on the MPQA dataset, ClusterFormer achieves a comparable result with MPSAN. We also carry out the text classification task on the long text dataset 20NEWS. The accuracy for the 20NEWS dataset increases at least 0.24% compared with other models. In addition, compared with the latest models Reformer and Routing Transformer, our model respectively has 6.1%, 3.8%, 1.6%, 2.0%, 2.6% and 10.0%, 4.9%, 2.0%, 11.3%, 3.1% improvement for CR, MR, SUBJ, MPQA and 20NEWS datasets.

4.3 Natural Language Inference (NLI) and Text Matching

In this section, we conduct Natural Language Inference tasks on SNIL, SciTail datasets, and Text Matching tasks on Quora, WikiQA datasets. SNLI (Bowman et al., 2015) is a benchmark dataset

Model	SNLI	SciTail	Quora	WikiQA	
				map	mrr
DELTA (Han et al., 2019)	80.7	–	–	–	–
Bigram-CNN (Yu et al., 2014)	–	–	–	0.619	0.628
Transformer[†] (Vaswani et al., 2017)	83.7	76.6	**85.4**	0.601	0.613
Reformer[†] (Kitaev et al., 2020)	78.6	67.3	74.3	0.587	0.603
Routing Transformer[†] (Roy et al., 2021)	76.3	72.6	81.5	0.560	0.574
ClusterFormer	**83.9**	**77.8**	**85.4**	**0.630**	**0.648**

Table 3: Experimental results on Neural Language Inference (NLI) and Text Matching tasks.

for natural language inference. There are 570k human-annotated sentence pairs with four labels. SciTail (Khot et al., 2018) is an entailment classification dataset constructed from science questions and answers. Quora Question Pairs is a dataset for paraphrase identification with two classes indicating whether one question is a paraphrase of the other. The evaluation metric for these three data sets is Accuracy. WikiQA (Yang et al., 2015) is a retrieval-based question answering dataset based on Wikipedia, which is composed of 20.4k/2.7k/6.2k (train/dev/test) samples. The mean average precision (MAP) and mean reciprocal rank (MRR) are used as the evaluation metrics. For SNIL and Quora datasets, word embeddings are initialized by GloVe (Pennington et al., 2014) with 300 dimensions. For the rest, we use random word embedding vectors with 300 dimensions. Some hyperparameters are set: $L = 1$, the number of heads $h = 6$ and the number of centroids $k = 3$.

As shown in Table 3, our model achieves the best results for most datasets. Specifically, the accuracy of our model is at least 1.6% higher than baseline models on the SciTail dataset. On WikiQA, our model improves the result by at least 1.8% and 3.2% in MAP and MRR evaluation metrics, respectively. Our model and Transformer have considerable effectiveness on SNLI and Quora datasets. In addition, compared with the latest models Reformer and Routing Transformer, our model has 6.7%, 15.6%, 14.9%, and 10.0%, 7.2%, 4.8% improvement for SNLI, SciTail, Quora datasets. For the WikiQA dataset, the score increases 7.3% and 7.5% by our model in MAP and MRR compared to Reformer. The score increases 12.5% and 13.0% compared to Routing Transformer.

4.4 The choice of clustering numbers k

In this section, we test the effect of different clustering numbers (k) on the effectiveness and efficiency.

Model	Acc(%)	Memory(MiB)	Time (s)
Transformer	84.8	21268 (x)	263.5 (y)
Our (k=5)	84.9	10980 (1.48x)	197.9 (1.25y)
Our (k=10)	84.3	9312 (1.56x)	169.4 (1.36y)
Our (k=20)	**85.0**	8542 (1.60x)	153.6 (1.42y)
Our (k=30)	84.5	8282 (1.61x)	**147.6 (1.44y)**
Our (k=60)	84.3	8072 (1.62x)	150.4 (1.43y)
Our (k=100)	83.6	7920 (1.63x)	151.7 (1.42y)
Our (k=300)	83.6	**7830 (1.63x)**	170.0 (1.35y)
Our (k=500)	83.2	8444 (1.60x)	194.4 (1.26y)

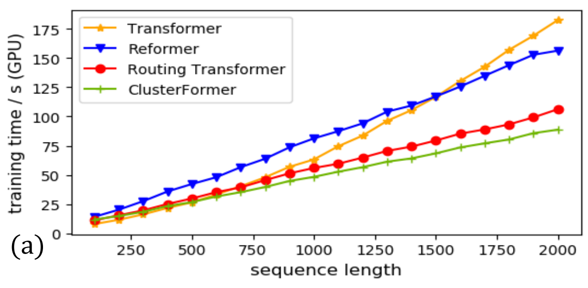
Table 4: Experimental results of different clustering numbers k for ClusterFormer.

We test our model on the 20NEWS dataset of text classification tasks with a NVIDIA V100 (16GB) GPU. Some hyperparameters are set: the number of encoder layers L is 2, the dimension of the model d is 300, the batch size is 64, and the max sequence length N is 1500.

From Table 4, we can draw the following conclusions: (i) Accuracy of our model: In general, within a certain range during the growth of k, the performance of our model is relatively stable. When the value of k goes beyond a certain threshold, the performance of our model degrades; (ii) Memory cost of our model: As the number of centroids k increases, the memory cost of the model decreases first and then increases; (iii) Training time of our model: As the number of centroids k increases, the training time of the model also decreases first and then increases. Therefore, according to this law, our method can simultaneously gain both the effectiveness and efficiency of the model by determining an appropriate k value through finite experiments.

4.5 Ablation study for Clustering Losses

In this section, we provide an ablation experiment about the two kinds of clustering losses. We verify the effectiveness of the two loss modules by assigning different weight. Some hyperparameters

(a) 

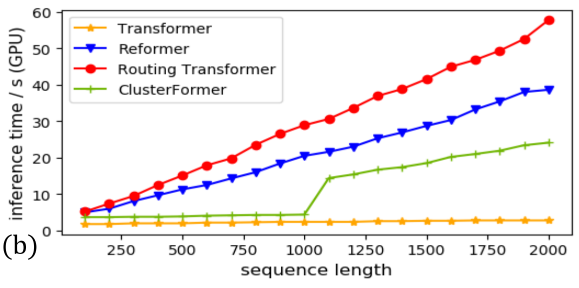
(b) 

Figure 3: (a), (b) is the training and inference time versus the sequence length on GPU.

Datasets / Weight(L1/L2)	N/N	N/Y	Y/N	Y/Y
SciTail(Acc)	77.19	77.75	75.96	**78.32**
WikiQA(map)	0.612	0.620	0.619	**0.631**
WikiQA(mrr)	0.636	0.640	0.633	**0.648**

Table 5: Experimental results of Clustering Loss ablation. Specifically, 'N' represents the weight is zero and 'Y' represents the weight is non-zero.

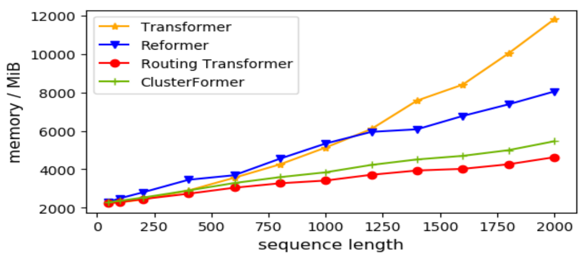


Figure 4: Memory cost versus the sequence length.

are set: the number of encoder layers L is 1, the dimension of model d is 300, the batch size is 128 and the max sequence length N is 500.

From Table 5, the experimental result shows that both L_1 and L_2 contribute to the performance. For example, on dataset SciTail, the accuracy with the best result is improved by 1.46% (acc) compared with the result without the two losses. On dataset WikiQA, the accuracy with the best result is improved by 3.10% (map), 1.89% (mrr) compared with the result without the two losses.

4.6 Time and Memory Analysis

In this section, we provide a comparison experiment on dataset 20NEWS about the time and memory cost for different models. About the dataset, its average sequence length is approximately 280 and the maximum sequence length exceeds 10,000. To compare time and memory cost, we set the range of sequence length N as (0, 2000] and batch size to 20. We test the memory and time cost on a NVIDIA V100 GPU. We take the time of 1000 steps forward propagation of the model as the inference time, and the time of 1000 steps forward and back propagation as the training time.

As shown in Figure 4, as the sentence length increases, both Routing Transformer and our model can significantly reduce memory cost compared to Transformer. When N exceeds 2000, our model, Routing Transfomer and Reformer reduce the memory by 53.8%, 60.8%, and 31.8%, respectively.

As shown in Figure 3, the training time of Transformer increases significantly with increasing sequence length, while our model and Routing Transformer have a relatively small increase on GPU devices. When N is 2000, our model, Routing Transfomer and Reformer reduce the training time by 51.4%, 41.8%, and 14.4%. However, the inference speed of these improvements is inferior compared with Transformer, which may be caused by the decrease of the model parallelism. The above analysis fully demonstrates the efficiency and effectiveness of our proposed Neural Clustering Mechanism.

5 Conclusion

In this paper, we propose a Neural Clustering Attention Mechanism to address the reduced effectiveness issue in sparse attention methods. This issue is mainly caused by the introduction of a sparse pattern that is separated from the target task or does not consider the similarity between words. In our method, we design a neural clustering algorithm to better capture critical pairs of dependencies. We integrate this clustering algorithm and the neural network to jointly train and optimize with specific tasks together to further contribute to the effectiveness and efficiency. The experimental results show that our model can achieve better effectiveness and a comparable or even better efficiency, compared with the latest typical sparse attention models, Reformer and Routing Transformer.

Acknowledgements

This work is supported in part by the state key development program of China (grant No.2017YFE0111900), Natural Science Foundation of China (grant No.61772363), PolyU internal fund (grant No.1-BD47) under the research project (P0039657) of the Hong Kong Polytechnic University, and the Beijing Academy of Artificial Intelligence(BAAI).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Iz Beltagy, Matthew E. Peters, Arman Cohan, et al. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, David Belanger, Lucy Colwell, et al. 2020. Masked language modeling for proteins via linearly scalable long-context transformers. *CoRR*, abs/2006.03555.

Biyun Dai, Jinlong Li, and Ruoyi Xu. 2020. Multiple positional self-attention network for text classification. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7610–7617. AAAI Press.

Kun Han, Junwen Chen, Hui Zhang, Haiyang Xu, Yiping Peng, Yun Wang, Ning Ding, Hui Deng, Yonghu Gao, Tingwei Guo, Yi Zhang, Yahao He, Baochang Ma, Yulong Zhou, Kangli Zhang, Chao Liu, Ying Lyu, Chenxi Wang, Cheng Gong, Yunbo Wang, Wei Zou, Hui Song, and Xiangang Li. 2019. DELTA: A deep learning based language technology platform. *CoRR*, abs/1908.01853.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. 2019. Axial attention in multidimensional transformers. *CoRR*, abs/1912.12180.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5189–5197. AAAI Press.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 271–278, Barcelona, Spain.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. 2020. Blockwise self-attention for long document understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2555–2565, Online. Association for Computational Linguistics.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5446–5455. AAAI Press.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2020a. Synthesizer: Rethinking self-attention in transformer models. *CoRR*, abs/2005.00743.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020b. Efficient transformers: A survey. *CoRR*, abs/2009.06732.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768.

Janyce Wiebe, Theresa Wilson, et al. 2005. Annotating expressions of opinions and emotions in language. *Lang. Resour. Evaluation*, 39(2-3):165–210.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention. *CoRR*, abs/2102.03902.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *CoRR*, abs/1412.1632.

A Comparison experiment for clustering methods

In this section, we carry out the comparison experiment between the Neural Clustering Method and other clustering methods to verify the effectiveness of our clustering method.

Firstly, according to the division mode, we introduce the following two kinds of clustering methods.

Hard Clustering: Each element to be recognized is strictly divided into a certain cluster . It defines an *either/or relationship* $R \in \{0, 1\}$ between the element and clusters.

Soft Clustering (Fuzzy Clustering): Each element to be recognized is subordinate to all clusters (with different subordinate values). It defines a *fuzzy relationship* $U \in [0, 1]$ between the element and clusters.

Regarding the selection of the comparative clustering method, we chose the classic hard clustering algorithm K-means (used for Routing Transformer) and the soft clustering algorithm SOM, a competitive neural network. In addition, since Locality Sensitive Hashing (used for Reformer) cannot construct the loss function (no iteration condition), it cannot be used for the following clustering task on the MNIST dataset. In the experiment, we set the number of centroids k to 10, 50, 100, 200, and 300 respectively.

Data set	SOM	K-Means	Proposed method
MNIST(k=10)	59.8	59.2	**60.7**(+0.9, +1.5)
MNIST(k=50)	80.9	**82.5**	**82.5**(+1.6, +0)
MNIST(k=100)	87.8	87.7	**90.1**(+2.3, +2.4)
MNIST(k=200)	91.2	90.7	**91.8**(+0.6, +1.1)
MNIST(k=300)	93.1	92.0	**93.6**(+0.5, +1.6)

Table 6: Comparison of clustering accuracy of different clustering algorithms on MNIST task.

As shown in Table 6, our method consistently has the best effectiveness in experiments with different centroids. In particular, when the number of centroids exceeds 300, the accuracy of our method can reach 93.6, which improved the accuracy by 0.54% and 1.74% compared with SOM and K-Means, respectively. From the above analysis, we have confirmed that Neural Clustering Method is a general clustering method. It can also achieve better effectiveness compared with K-Means and SOM when handling clustering tasks alone.

B Masked Neural Clustering Attention Mechanism

For autoregressive modeling, we provide a Masked Neural Clustering Attention Mechanism to prevent the leftward information flow. We first obtain the original position indexes of the Q^{O_i}, K^{O_i} matrix. The formula is as follows:

$$\begin{aligned} I^{Q^{O_i}} &= I^{'}\left((i-1)\left\lceil\frac{N}{k}\right\rceil : i\left\lceil\frac{N}{k}\right\rceil\right] \\ I^{K^{O_i}} &= I^{'}\left((i-2)\left\lceil\frac{N}{k}\right\rceil : i\left\lceil\frac{N}{k}\right\rceil\right] \quad 1 \leq i \leq k \end{aligned} \tag{12}$$

where $I^{Q^{O_i}} \in \mathbb{R}^{w}$ and $I^{K^{O_i}} \in \mathbb{R}^{2w}$. $I^{Q^{O_i}}$ and $I^{K^{O_i}}$ are the original position indexes of the *i*-th sequence block Q^{O_i} and K^{O_i} respectively.

Then, we extend $I^{Q^{O_i}}$ in the second dimension to get $M^{Q^{O_i}} \in \mathbb{R}^{w \times 2w}$, and extend $I^{K^{O_i}}$ in the first dimension to get $M^{K^{O_i}} \in \mathbb{R}^{w \times 2w}$. Therefore, we can obtain a mask matrix M^{O_i} for the *i*-th sequence block by comparing these position indexes, as follows:

$$\begin{aligned} & M_{uv}^{O_i} = M_{uv}^{K^{O_i}} \geq M_{uv}^{K^{O_i}} \\ & 1 \leq i \leq k\,, \quad 1 \leq u \leq w\,, \quad 1 \leq v \leq 2w \end{aligned} \tag{13}$$

where $M^{O_i} \in \mathbb{R}^{w \times 2w}$ is the mask matrix of the *i*-th block and the M^{O_i} is composed of either 0 or 1. $M_{uv}^{O_i}$ is the value of the *u*-th row and *v*-th column of the matrix M^{O_i}. Then, for each word in Query block, it will mask the words whose index value in Key block is greater than it according to mask matrix M^{O_i}, as follows:

$$\begin{aligned} & S_{uv}^{O_i} = \sum_{m=1}^{d} Q_{um}^{O_i} K_{vm}^{O_i} M_{uv}^{O_i} \\ & 1 \leq i \leq k\,, \quad 1 \leq u \leq w\,, \quad 1 \leq v \leq 2w \end{aligned} \tag{14}$$

where $S^{O_i} \in \mathbb{R}^{w \times 2w}$ is the similarity matrix of the *i*-th sequence block. The subsequent operations are the same as Neural Clustering Attention Mechanism.

C Optimization of the multi-task learning for ClusterFormer

As shown in Figure 1, our model, ClusterFormer, consists of two joint training tasks, clustering tasks, and specific tasks of the model. Therefore, it contains the loss from the two tasks. The loss functions related to the clustering task are Clustering Loss (L_1) and Centroid Sorting Loss (L_2). Their equa-

Evaluation Metrics	Model/Length	2500	5000	7500	10000	15000	25000	35000	45000	55000
Memory(MiB)	Transformer	3271	7663	15519	–	–	–	–	–	–
	Reformer	2699	3807	4795	6067	9515	15867	–	–	–
	Routing Transformer	2129	2467	2841	3189	3779	5477	7451	9117	11397
	ClusterFormer	2257	2689	3165	3623	4435	6839	9647	12553	15695
Training Time(s)	Transformer	46.5	127.6	260.9	–	–	–	–	–	–
	Reformer	30.7	51.2	72.3	89.0	135.0	252.1	–	–	–
	Routing Transformer	24.8	38.5	58.5	69.8	114.9	207.2	343.4	490.1	684.6
	ClusterFormer	30.2	44.8	60.4	77.2	115.4	197.1	277.2	375.6	461.3

Table 7: Memory efficiency (measuring unit: MiB) and Training-time efficiency(measuring unit: second) of different methods under the ultra-long sequence condition. '-' represent memory cost exceeds the upper limit of the GPU device and the corresponding training time can not be provided.

tions are as follows:

$$L_1 = -\frac{1}{N}\sum_{j=1}^{N}\phi(X_j, \widehat{C}_{I_j})$$
$$L_2 = -\frac{1}{k}((\sum_{i=2}^{k}\phi(\widehat{C}_i, \widehat{C}_{i-1})) + \phi(\widehat{C}_1, \widehat{C}_k)) \tag{15}$$

where X and C are respectively word vectors and centroid vectors. N is the length of the input and k is the number of clusters. The loss function of a specific task (e.g., text classification) is formulated as follows:

$$L_3 = -\sum_{i=1}^{n} y_i log(\widehat{y_i}) + (1-y_i)log(1-\widehat{y_i}) \tag{16}$$

where $\widehat{y_i}$ is predictive value and y_i is the corresponding target value. Then, the overall loss function can be written as:

$$L_{total}(X, C, W, b) = \mu L_1 + \nu L_2 + \lambda L_3 \tag{17}$$

where C, W, and b are respectively centroid parameters, weight parameters, and bias parameters in ClusterFormer. μ, ν, and $\lambda \geq 0$ are non-negative coefficients, which are used to adjust the proportion of the importance of corresponding tasks.

Therefore, we can obtain the optimal parameters in the neural network by minimizing the loss function L_{total} through gradient descent, as follows:

$$c_i = c_i - \eta\frac{\partial L_{total}(X, C, W, b)}{\partial c_i} \quad 1 \leq i \leq k \times d_{model}$$
$$w_j = w_j - \eta\frac{\partial L_{total}(X, C, W, b)}{\partial w_j} \quad 1 \leq j \leq m$$
$$b_r = b_r - \eta\frac{\partial L_{total}(X, C, W, b)}{\partial b_r} \quad 1 \leq r \leq n \tag{18}$$

where c_i, w_i and b_i represent the element value of the corresponding vectors. d_{model} is the dimensionality of the model. m and n represent the number of weight and bias parameters, respectively. η is the learning rate. From the above, it can be seen that the update of centroid, weight, and bias parameters is the result of multi-task joint learning in ClusterFormer.

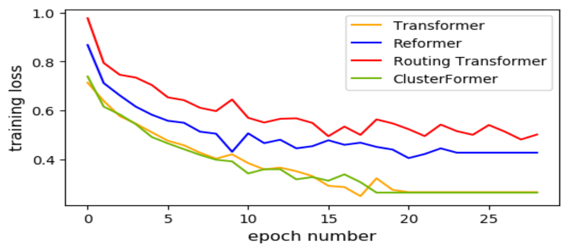


Figure 5: Comparison of training loss on SNLI dataset.

D Convergence Analysis

In this section, we provide a comparison experiment on the SNLI dataset about the convergence speed for standard Transformer and efficient Transformers during training, as shown in Figure 5. In our experiment, the epochs of different models to achieve convergence is: Transformer has 21 epochs, Reformer has 24 epochs Routing Transformer has 29 epochs and ClusterFormer has 19 epochs. Compared with Transformer, our model has a comparative convergence rate and is more stable. In addition, compared with the latest model Reformer and Routing Transformer, our model not only has a faster and more stable convergence speed, but also has better effectiveness.

E Comparison experiment of different methods under the ultra-long sequence condition

In this section, we have supplemented the experiment of time and memory cost in different methods on extremely long sequence tasks. We tested them on text classification of the 20NEWS dataset and trained them on a NVIDIA V100 GPU. We set the

Model	MNLI	QQP	QNLI	SST2	CoLA	STSB	MRPC	RTE	Average
Bert-Small-uncased	77.6/77.0	68.1/87.0	86.4	89.7	27.8	78.8/77.0	83.4/76.2	61.8	74.23
Pretrained ClusterFormer	75.21/75.7	83.1/86.9	82.5	88.3	34.2	82.7/82.4	85.2/77.7	60.3	**76.18**

Table 8: GLUE Dev results, the "Average" column represents the average of all datasets scores. F1 and accuracy scores are reported for QQP and MRPC. Spearman correlations are reported for STSB. Accuracy scores are reported for the other tasks.

number of centroids k to the square root of the max length and set the batch size to 2 (constrained by resources).

As shown in Table 7, we can see that our method has a better efficiency advantage on long sequences compared with Transformer. And as the sequence length increases, the advantage in memory and training time are even more significant.

F Pretraining experiment with the Neural Clustering Attention Mechanism

In this section, we pretrain a model with the Neural Clustering Attention Mechanism with two unsupervised tasks, masked language modeling (MLM) and next sentence prediction (NSP). The parameter settings of our pretraining model are similar to BERT-small-uncased. Some hyperparameters are set: the number of layers L is 4, the hidden size is 512, and the number of heads h is 8.

For downstream tasks, we use the General Language Understanding Evaluation (GLEU) benchmark which is a collection of diverse natural language understanding tasks. We use a batch size of 32 and fine-tune On a scale of 3 to 10 epochs over the data for all GLEU tasks. For each task, we selected the best fine-tuning learning rate (5e-5, 4e-5, 3e-5, and 2e-5) on the Dev set.

As shown in Table 8, experimental results demonstrate that our method can have a good performance improvement through Pretraining. Especially, compared with the Bert-Small-uncased, pretraining ClusterFormer respectively has 23.0%, 4.9% / 7.0%, and 2.2% / 2.0% improvement on CoLA, STSB, and MRPC datasets. The experimental results show that our model has the potential to do more NLP tasks including pretraining and non-pretraining tasks.