HiPool: Modeling Long Documents Using Graph Neural Networks

Irene R. Li[1], Aosong Feng[2], Dragomir Radev[2], Rex Ying[2]
[1]University of Tokyo, [2]Yale University
ireneli@ds.itc.u-tokyo.ac.jp, {aosong.feng, dragomir.radev, rex.ying}@yale.edu

Abstract

Encoding long sequences in Natural Language Processing (NLP) is a challenging problem. Though recent pretraining language models achieve satisfying performances in many NLP tasks, they are still restricted by a pre-defined maximum length, making them challenging to be extended to longer sequences. So some recent works utilize hierarchies to model long sequences. However, most of them apply sequential models for upper hierarchies, suffering from long dependency issues. In this paper, we alleviate these issues through a graph-based method. We first chunk the sequence with a fixed length to model the sentence-level information. We then leverage graphs to model intra- and cross-sentence correlations with a new attention mechanism. Additionally, due to limited standard benchmarks for long document classification (LDC), we propose a new challenging benchmark, totaling six datasets with up to 53k samples and 4034 average tokens' length. Evaluation shows our model surpasses competitive baselines by 2.6% in F1 score, and 4.8% on the longest sequence dataset. Our method is shown to outperform hierarchical sequential models with better performance and scalability, especially for longer sequences.

1 Introduction

Transformer-based models like BERT (Vaswani et al., 2017a) and RoBERTa (Zhuang et al., 2021) have achieved satisfying results in many Natural Language Processing (NLP) tasks thanks to large-scale pretraining (Vaswani et al., 2017b). However, they usually have a fixed length limit, due to the quadratic complexity of the dense self-attention mechanism, making it challenging to encode long sequences.

One way to solve this problem is to adapt Transformers to accommodate longer inputs and optimize the attention from BERT (Feng et al., 2022; Jaszczur et al., 2021). BigBird (Zaheer et al., 2020) applies sparse attention that combines random, global, and sliding window attention in a long sequence, reducing the quadratic dependency of full attention to linear. Similarly, Longformer (Beltagy et al., 2020) applies an efficient self-attention with dilated windows that scale linearly to the window length. Both models can take up to 4096 input tokens. Though it is possible to train even larger models for longer sequences, they are restricted by a pre-defined maximum length with poor scalability. More importantly, they fail to capture high-level structures, such as relations among sentences or paragraphs, which are essential to improving NLP system performance (Zhang et al., 2018; Zhu et al., 2019).

Another way is to apply a hierarchical structure to process adjustable input lengths with chunking representations for scalability on long sequences. Hi-Transformer (Wu et al., 2021) encodes both sentence-level and document-level representations using Transformers. ToBERT (Pappagari et al., 2019) applies a similar approach that stacks a sentence-level Transformer over a pretrained BERT model. While most of the existing work models upper-level hierarchy using *sequential structures*, such as multiple layers of LSTMs (Hochreiter and Schmidhuber, 1997) or Transformers, this may still bring the long dependency issue when the sequence gets longer. To alleviate this, we investigate graph modeling as a novel hierarchy for upper levels. Besides, we also consider inter-hierarchy relationships using a new attention mechanism.

Our key insight is to replace the sequence-based model with a hierarchical attentional graph for long documents. We first apply a basic pretrained language model, BERT or RoBERTa, to encode local representation on document chunks with a fixed length. The number of chunks could be extended for longer sequences for better scalability. Different from other works, we apply a graph neural network (GNN) (Zhou et al., 2018) to model the upper-level hierarchy to aggregate local sentence in-

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 161–171

formation. This is to alleviate the long dependency issue of the sequential model. Moreover, within such a graph structure, we propose a new heterogeneous attention mechanism to consider intra- and cross- sentence-level correlations.

Our contributions are two-fold: 1) We propose HiPool with multi-level hierarchies for long sequence tasks with a novel inter-hierarchy graph attention structure. Such heterogeneous graph attention is shown to outperform hierarchical sequential models with better performance and scalability, especially for longer sequences; 2) We benchmark the LDC (long document classification) task with better scaled and length-extended datasets. Evaluation shows that HiPool surpasses competitive baselines by 2.6% in F1 score, and 4.8% on the longest sequence dataset. Code is available at https://github.com/IreneZihuiLi/HiPool.

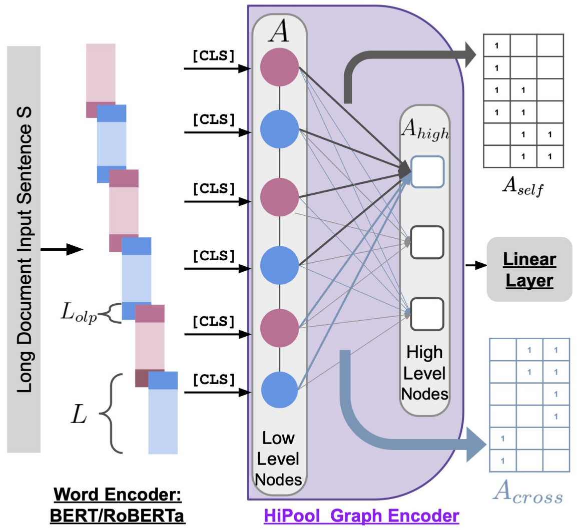


Figure 1: HiPool model illustration. It consists of a sequence encoder, HiPool graph encoder and a linear layer.

2 Model

We introduce the HiPool (**Hi**erarchical **Pool**ing) model for long document classification, illustrated in Fig. 1. It consists of an overlapping sequence encoder, a HiPool graph encoder, and a linear layer.

Overlapping Sequence Encoder. Given the input document S, we first chunk the document into a number of shorter pieces with a fixed length L, and we set the overlapping window size to be L_{olp}. Overlapping encoding makes it possible for a chunk to carry information from its adjacent chunks but not isolated, differentiating our model from other hierarchical ones. Then each chunk is encoded with a pretrained Transformer model, i.e., BERT or RoBERTa; we choose the `CLS` token representation as the input to our HiPool layer: $X = \text{BERT}(S)$.

HiPool Graph Encoder. We apply a graph neural network to encode incoming word-level information. Such a model has shown its potential in some NLP tasks (Li et al., 2022, 2021). We construct a graph, defined by $G(V, E)$, where V is a set of nodes, and E is a set of node connections. There are two node types: n *low-level nodes* and m *high-level nodes*, and typically $m < n$. In our experiment, we set $m = n/p$, and $p \geq 0$. The feedforward operation goes from low- to high-level nodes. In layer l, low-level nodes are inputs from the previous layer $l-1$, while high-level nodes at layer l are computed based on low-level ones. Moreover, these high-level nodes will be the input to the next layer $l+1$, becoming the low-level nodes in that layer. We consider X the low-level nodes in the first HiPool layer, as shown in the figure.

In each HiPool layer, given node representation H^l and adjacency matrix A^l at layer l, the task is to obtain H^{l+1}:

$$H^{l+1} = \text{HiPool}(H^l, A^l). \quad (1)$$

Inspired by DiffPool (Ying et al., 2018), we conduct a clustering method to aggregate information. We assign node clusters with a fixed pattern based on their position. For example, adjacent low-level neighbors should map to the same high-level clustering node. So we first define a clustering adjacency matrix $A_{self} \in \mathbb{R}^{n \times m}$ that maps n nodes to m nodes, indicating the relations from low- to high- level nodes, marked as black arrows in the figure. Note that our approach allows overlapping, in which some nodes may belong to two clusters. We set the clustering sliding window to be $2p$, with a stride to be p. In the figure, we show the case of $p = 2$. We denote interactions between low-level nodes by the adjacency matrix A^l,[1] and we model it using a chain graph, according to the natural order of the document.[2]

Then, the relations between high-level nodes A^l_{high} and their node representations H^l_{high} are computed:

$$\begin{aligned} A^l_{high} &= A^T_{self} A^l A_{self}, \\ H^l_{high} &= A_{self} H^l. \end{aligned} \quad (2)$$

[1]We eliminated the subscript of A_{low} for simplicity, and this also makes Eq. 1 more generalized as other GNNs.

[2]We tested with a complete graph and BigBird attention structures but found little differences.

Besides, for each high-level node, to strengthen the connections across different clusters, we propose an attention mechanism to obtain cross-sentence information. We propose a new edge type that connects external cluster low-level nodes to each high-level node, and the adjacency matrix is simply $A_{cross} = 1 - A_{self}$, marked by green in the figure. We update H^l_{high} as the following:

$$\begin{aligned} W_{score} &= H^l_{self} W_{atten} (H^l)^T, \\ W_{score} &= W_{score} A^T_{cross}, \\ H^l_{high} &\leftarrow W_{score} H^l + H^l_{high}, \end{aligned} \tag{3}$$

where W_{atten} is trainable, and W_{score} is a scoring matrix. We then apply a GNN to obtain H^{l+1}. For example, a graph convolution network (GCN) (Kipf and Welling, 2016):

$$H^{l+1} = \text{GCN}(H^l_{high}, A^l_{high}). \tag{4}$$

We run our experiments with two layers, and apply a sum aggregator to achieve document embeddings. More HiPool layers are also possible.

Linear Layer. Finally, a linear layer is connected and cross-entropy loss is applied during training.

3 Experiments

3.1 LDC Benchmark

The LDC benchmark contains six datasets. We first choose four widely-used public datasets. **Hyperpartisan** (HYP) (Kiesel et al., 2019) and **20NewsGroups** (20NG) (Lang, 1995) are both news text datasets with different scales. **IMDB** (Maas et al., 2011) is a movie review dataset for sentiment classification. **ILDC** (Malik et al., 2021) is a large corpus of legal cases annotated with binary court decisions ("accepted"and "rejected").

Limitation and new datasets. However, 20NewsGroups and IMDB cannot test the limit of models in encoding long documents since the average length of sentence is still relatively small; whereas Hyperpartisan only contains 645 examples and is thus prone to overfitting and not representative. ILDC is large and contains long texts, but it is mainly in the legal domain. Therefore, to enrich evaluation scenario, we select and propose two new benchmarks with longer documents based on an existing large-scale corpus, Amazon product reviews (He and McAuley, 2016), to conduct long document classification. **Amazon-512** (A-512) contains all reviews that are longer than 512 words from the *Electronics* category; **Amazon-2048** (A-2048) contains 10,000 randomly sampled reviews that are longer than 2048 words from the *Books* category. We randomly split 8/1/1 as train/dev/test sets for both datasets. The proposed datasets enable us to draw statistically significant conclusions on model performance as sequence lengths increase, as demonstrated in in Table 1.

	HYP	20NG	IMDB	A-512	A-2048	ILDC
Mean	741.44	587.56	301.14	879.62	2,915.03	4039.85
Max	5,368	144,592	3,152	17,988	14,120	501,091
Min	21	37	10	512	2,048	53
Med.	547	360	225	725	2,505	2,663
95pt.	2,030	1,229	771	1,696	5,216	11,416
Total	645	18,846	50,000	53,471	10,000	34,816
Class	2	20	2	5	5	2

Table 1: Dataset statistics on LDC benchmark. `Med.` is the median value. `95pt.` indicates 95th percentile. `Class` indicates the number of classes.

3.2 Evaluation

Hyperparameters. We list details in Appendix C.

Baselines. We select four pretrained models: BERT (Devlin et al., 2019), RoBERTa (Zhuang et al., 2021), BigBird (Zaheer et al., 2020) and Longformer (Beltagy et al., 2020). We also compare with a hierarchical Transformer model ToBERT (Pappagari et al., 2019). Hi-Transformer (Wu et al., 2021) failed to be reproduced as there is no code available. We evaluate two variations of our HiPool method by changing the sequence encoder model: HiPool-BERT and HiPool-RoBERTa. We report the Micro-F1 score in Tab. 2.

Main Results. Among the pretrained models, Longformer and BigBird perform better than BERT and RoBERTa. ToBERT can only surpass BERT as it is a hierarchical model that applies BERT as its text encoder. On average, HiPool-BERT improves significantly on BERT by 5.9% and on ToBERT by 3%. Compared to ToBERT, the superior performance of HiPool can be explained by the fact that sentence-level representations in ToBERT fails to capture cross-sentence information. HiPool surpasses baselines on A-512, A-2048 and ILDC that contain longer sequences. Notably, the best model, HiPool-RoBERTa, outperforms BigBird by 4.8% on ILDC. While our model applies a basic pretrained text encoder (the maximum length is 512), it can still surpass larger pretrained language models (i.e., the maximum length is 4096). Although HiPool is worse on HYP and IMDB, we note that HYP only has 65 examples in testing and is prone to overfitting. We further show that even in IMDB, HiPool still out-performs the best model for long

	HYP	20NG	IMDB	A-512	A-2048	ILDC	Avg.
BERT	0.857	0.853	0.913	0.592	0.503	0.556	0.712
RoBERTa	0.874	0.857	0.953	0.650	0.579	0.560	0.745
BigBird	0.922	0.823	0.952	0.674	0.636	0.637	<u>0.774</u>
Longformer	**0.938**	0.863	**0.957**	0.673	0.612	0.562	0.768
ToBERT	0.862	0.901	0.924	0.587	0.560	0.611	0.741
HiPool-BERT	0.865±0.030	**0.908±0.005**	0.931±0.001	0.660±0.009	0.612±0.011	0.651±0.010	0.771
HiPool-RoBERTa	0.886±0.018	0.904±0.001	0.948±0.001	**0.690±0.007**	**0.648±0.017**	**0.685±0.018**	**0.794**

Table 2: Main evaluation results on LDC benchmark. We underscore the best average of baselines, and bold the best overall models.

Hierarchy	F1	Hierarchy	F1
Sequential		*Graph*	
Simple	0.618	Aggr-mean	0.621
CNN	0.608	Aggr-std	0.620
Trans.	0.560	Aggr-pna	0.633
		HiPool	**0.648**

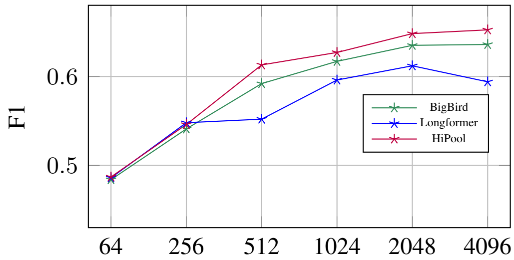
Table 3: Comparison of multiple hierarchies.

sequence in Appendix A.

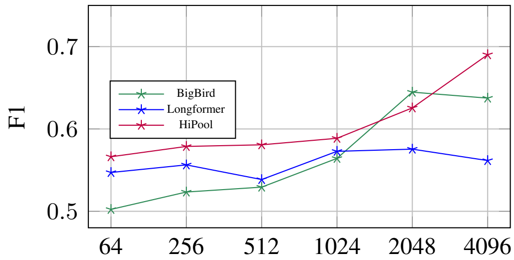
Hierarchy variations. To further compare sequential and graph hierarchy, we keep the word encoder and replace the HiPool graph encoder with the following sequential modules: `Simple` linear summation over low-level nodes; `CNN` applies a 1-dimension convolution; `Trans` is to apply a Transformer on top of low-level nodes. Besides, we also look at multiple graph settings: `Aggr-mean` is to use a mean aggregator to obtain the final document representation; `Aggr-std` is to use a feature-wise standard deviation aggregator; finally, `Aggr-pcp` applies Principal Neighbourhood Aggregation (PNA) (Corso et al., 2020). We report results on Amazon-2048 in Tab. 3, as it has the longest sequence on average. An observation is that applying aggregators are better than simpler structures, while keeping a graph is still a better choice. HiPool also considers attention in message passing, so it is doing even better. We also test other variations in Appendix B.

3.3 Ablation Study

Effect of input length. To better understand the effect of input length, in Fig. 2, we present an ablation study on the Amazon-2048 and ILDC, and compare three models: BigBird, Longformer, and HiPool. In general, the models benefit from longer input sequences in both datasets. Interestingly, when sequence is larger than 2048, Longformer and Bigbird could not improve and they are limited in maximum lengths. In contrast, as the input sequence gets longer, HiPool steadily improves, showing its ability to encode long documents in a hierarchical structure.



(a) Amazon-2048



(b) ILDC

Figure 2: Ablation study on the input text length. (X-axis shows input length.)

	A-512	A-2048	ILDC	Avg.
HiPool-RoBERTa	**0.690**	**0.648**	**0.685**	**0.674**
w/o RoBERTa	0.660	0.612	0.651	0.641
w/o HiPool	0.601	0.578	0.620	0.600
w/o Overlapping	0.587	0.560	0.611	0.586

Table 4: The effect of sequence encoding layer, HiPool layer and overlapping modules.

Model component. Next, we look at how each component of HiPool affects performance. As shown in Tab. 4, we first take the best model setting, HiPool-RoBERTa, and compare it with the following settings: 1) `w/o RoBERTa` is to replace RoBERTa with BERT, then the model becomes HiPool-BERT; 2) `w/o HiPool` is to remove the proposed HiPool module and replace with a simple CNN (Kim, 2014); 3) `w/o Overlapping` is to remove the overlapping word encoding. We could see that removing the HiPool Layer leads to a significant drop, indicating the importance of the proposed method. Moreover, the HiPool framework can work with many pretrained language models, as we can see that applying RoBERTa improves BERT. A complete result table can be found in Appendix.

4 Conclusion

In this paper, we proposed a hierarchical framework for long document classification. The evaluation shows our model surpasses competitive baselines.

5 Limitations and Potential Risks

Limitations The model we proposed is specifically for classification, while it is possible to be extended to other NLP tasks by changing the high-level task-specific layer. Besides, in the evaluation, we focused on English corpora. We plan to test on other languages in the future.

Potential Risks We make our code publicly available so that everyone can access our code. As the model is a classification model, it does not generate risky content. Users should also notice that the classification predictions may not be perfectly correct.

6 Acknowledgements

This paper is dedicated to the memory of Professor Dragomir Radev, who passed away while this paper was being peer-reviewed.

References

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

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. 2020. Principal neighbourhood aggregation for graph nets.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Aosong Feng, Irene Li, Yuang Jiang, and Rex Ying. 2022. Diffuser: Efficient transformers with multi-hop attention diffusion for long sequences. *arXiv preprint arXiv:2210.11794*.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. 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 1024–1034.

Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 507–517. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. 2021. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34:9895–9907.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 829–839, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 331–339. Morgan Kaufmann.

Irene Li, Linfeng Song, Kun Xu, and Dong Yu. 2022. Variational graph autoencoding as cheap supervision for AMR coreference resolution. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2790–2800, Dublin, Ireland. Association for Computational Linguistics.

Irene Li, Vanessa Yan, Tianxiao Li, Rihao Qu, and Dragomir Radev. 2021. Unsupervised cross-domain prerequisite chain learning using variational graph autoencoders. *arXiv preprint arXiv:2105.03505*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Vijit Malik, Rishabh Sanjay, Shubham Kumar Nigam, Kripabandhu Ghosh, Shouvik Kumar Guha, Arnab Bhattacharya, and Ashutosh Modi. 2021. ILDC for CJPE: Indian legal documents corpus for court judgment prediction and explanation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4046–4062, Online. Association for Computational Linguistics.

Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844. IEEE.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Don Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *ICLR 2021*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017a. 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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017b. Attention is all you need.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Hi-transformer: Hierarchical interactive transformer for efficient and effective long document modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, pages 848–853. Association for Computational Linguistics.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4805–4815.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Tianyang Zhang, Minlie Huang, and Li Zhao. 2018. Learning structured representation for text classification via reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434.

Ming Zhu, Aman Ahuja, Wei Wei, and Chandan K Reddy. 2019. A hierarchical attention retrieval model for healthcare question answering. In *The World Wide Web Conference*, pages 2472–2482.

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A robustly optimized BERT pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China.

A IMDB-long Dataset

HiPool Performs The Best for Long Sequences in IMDB. As a supplementary analysis, we look at the IMDB dataset, in which HiPool performs worse than BigBird and Longformer. We filter out the sequences that are longer than 512 tokens to construct the **IMDB-long** dataset, resulting in 3250 and 3490 samples for training and testing. We show the detailed statistics of the IMDB-long dataset in Tab. 5. We show the evaluation in Fig. 3. We can observe that HiPool can do better for long sequences.

	Train	**Test**
Mean	761.35	764.65
Max	2,977	3,152
Min	512	512
Med	689	693
50th pctl.	689	693
95th pctl.	1,236	1,232
Total	3,250	3,490

Table 5: IMDB-long dataset statistics.

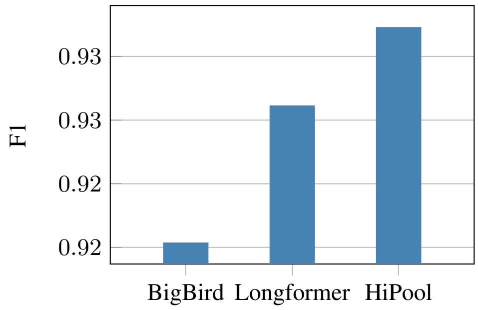


Figure 3: Performance on IMDB-long. HiPool outperforms BigBird and Longformer when the sequence length is larger than 512.

B Graph Variations

We study other possible GNN types for hierarchy modeling. In Eq. 1, we replace the HiPool graph encoder with a GCN or GAT encoder. We apply two layers of the graph networks before the linear layer to compare fairly, and show results in Fig. 6. We notice that using GCN and GAT results in lower performance than that of HiPool. A possible reason is that they only focus on modeling the low-level nodes, ignoring a cross-sentence attention mechanism to strengthen high-level communication on long sequences like HiPool.

	HYP	**20NG**	**IMDB**	**A-512**	**A-2048**	**ILDC**	**Avg.**
BERT-GCN	0.859	0.904	0.927	0.645	0.591	0.623	0.758
BERT-GAT	0.846	0.907	0.929	0.653	0.602	0.626	0.760
BERT-HiPool	0.865	0.908	0.931	0.660	0.612	0.651	**0.771**
RoBERTa-GCN	0.874	0.903	0.944	0.670	0.631	0.656	0.780
RoBERTa-GAT	0.849	0.899	0.945	0.678	0.640	0.673	0.781
RoBERTa-HiPool	0.886	0.904	0.948	0.690	0.648	0.690	**0.794**

Table 6: Comparison of other GNN types: we report F1 scores for individual dataset and the average. HiPool method is better than GCN and GAT.

C Hyperparameters, Experimental Settings

We run our experiments on 4 NVIDIA RTX A6000 GPUs, with the memory to be 48GB. We list hyperparameters for baselines and HiPool model in Tab. 7. For all datasets, we apply Adam optimizer (Kingma and Ba, 2014) for all experiments. For HiPool, we set the chunk length $L = 300$, and the overlapping length L_{olp} is $L/2 = 150$. We apply two layers of HiPool, reducing the number of nodes for each layer by $p = 2$. Among the baseline models, ToBERT (Pappagari et al., 2019) is adjustable for the maximum length, because it takes the maximum value in a batch during training. We evaluated F1 scores using scikit-learn: https://scikit-learn.org/stable/.

	HYP	**20NG**	**IMDB**	**A-512**	**A-1024**	**ILDC**	**Time***
BERT, RoBERTa							20
max_len	512	512	512	512	512	512	
#epoch	10	10	10	10	10	10	
learning rate	5e-6	5e-6	5e-6	5e-6	5e-6	5e-6	
BigBird, Longformer							40
max_len	1024	1024	1024	2048	4096	4096	
#epoch	10	10	10	10	10	10	
learning rate	5e-6	5e-6	5e-6	5e-6	5e-6	5e-6	
ToBERT							25
#epoch	8	10	10	12	12	12	
learning rate	1e-5	1e-5	1e-5	1e-5	1e-5	1e-5	
HiPool							50×5
#max_node	10	8	8	10	15	15	
#epoch	8	10	10	12	12	12	
learning rate: BERT	1e-5	1e-5	1e-5	1e-5	1e-5	5e-6	
learning rate: RoBERTa	5e-6	5e-6	5e-6	5e-6	5e-6	5e-6	

Table 7: Hyperparameters for baseline models and HiPool. `Time*` indicates how many hours on overall trial, training and testing using a single GPU. Note that we report average and standard deviation for HiPool, so we ran the evaluation at least 5 times there.

D Frequently Asked Questions

- Q: *Why do we call it a heterogeneous graph?*

  A: We use the term "heterogeneous"to distinguish the nodes from the graph. We wish to emphasize that the nodes are not the same, and they come from multiple levels and represent different information.

- Q: *Are there other possible variations for modeling the hierarchy?*

  A: Yes, our HiPool model is a framework that applies a graph structure for high-level hierarchy, so it is possible to apply other GNN models. One can use Relational Graph Convolutional Networks (R-GCNs) (Schlichtkrull et al., 2018) to model the different relations for A_{self} and A_{cross}. Besides, some inductive methods like GraphSAGE (Hamilton et al., 2017) can also be applied to obtain node embeddings in the graph. We leave this topic as future work.

- Q: *How does the aggregator work in Tab. 3.?*

  A: We replace the sum aggregator of our original HiPool with those mentioned aggregators. The applied PyTorch implementation: https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html#aggregation-operators.

- Q: *Why did not evaluate on the LRA (Long Range Arena) (Tay et al., 2021) benchmark?*

  A: LRA is more suitable for testing the efficiency of Transformer-based models and it consists of multiple types of long sequences. As we mentioned in the Introduction, our proposed model belongs to another category for long sequence encoding, not the efficiency transformer category that focuses on optimizing KQV attention.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Appendix Section E

☑ A2. Did you discuss any potential risks of your work?
Appendix Section E

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3

☑ B1. Did you cite the creators of artifacts you used?
Section 3

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Section 3

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 3

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Section 3

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 3

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 3, Appendix A

C ☑ Did you run computational experiments?

Section 3

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix C, Section 3

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix C, Section 3

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Appendix B, C, Section 3

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix C,D, Section 3

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.