BigBird: Transformers for Longer Sequences - A Reanalysis Study

Amit Gopal Hattimare
CICS, UMass Amherst
ahattimare@umass.edu

Abstract

Transformer based models like BERT have become the state-of-the-art performers for many NLP applications. Their robust nature allows similar architectures to be used for a variety of NLP problems. One issue identified with such models is the quadratic memory and computation requirement on the input sequence length which prevents their application on larger sequences using existing hardware. The BigBird [20] model solves this by proposing an architecture having linear complexity. This paper performs a reanalysis of some parts of that original paper. We show the importance of global tokens in sparsification, the robustness of BigBird model for NLP tasks, the ability to use 8x sequence length on existing hardware and the importance of longer context in QA problems. This study opens up the possibility of usage of BERT like models in resource constrained environments like mobile devices.

1 Introduction

Transformer based models are seen in many state-of-the-art solutions to NLP problems. Zaheer et al. [20] suggest improvements in this model to reduce memory and computation. The *Behaviour* of the paper is to understand memory consumption, latency, and performance of transformer based models on common NLP tasks. It analyses accuracy and F1 score on Question-Answering (QA) and some document classification datasets, and ROUGE score on some long-document summarization datasets. The *System* consists of the BigBird attention mechanism architecture proposed for Transformers. It consists of 3 main parts - a set of global attention tokens, neighbouring tokens attending each token, and some random attention tokens. It also includes the criteria for selecting these tokens which reduce quadratic memory dependency to linear on the input sequence length. The *Task* is to perform 3 common NLP tasks of QA, Document Summarization and Document Classification on some popular datasets. The *Environment* consists of the multiple human language datasets which include real-world examples. They contain the complexities of human language, the way humans ask questions, the hidden answer components in a long document, the criteria for deciding the class of a document, and the way a human would understand and summarize a long document.

In this paper we perform a partial reanalysis study. Sections 3, 4 and 5 go through one research question (RQ) each that we ask followed by corresponding hypotheses that we make to answer those RQs. Then in 6 we explain our experimentation technique, models and hyperparameters, and the data sets used. We present our results in 7 and discuss their implications on our hypotheses and associated RQs. We denote section 8 to mention the challenges we faced in this reanalysis study to help future researchers in making their work more accessible. Finally, we conclude the work in 9 where we also discuss the larger implications of our work and how it can benefit in different areas. Section 10 mentions hypotheses that we considered but could not test, and other ideas that we leave for future work.

Preprint. Under review.

2 Related Work

Transformer model was introduced in Vaswani et al. [15] to remove the limitations of Recurrent Neural Networks, LSTM [7] and GRU [5] based models which were sequential in nature even after multiple improvements in efficiency through factorization [10] and conditional computation [14] methods. The concept of attention was known beforehand and was used in conjunction with RNNs but Transformers completely dropped recurrence and solely relied on the attention mechanism. Self-attention reduced the length of the longest path from one token to another from $O(n)$ to $O(1)$ which allowed tokens to depend on far away tokens and also allowed them to look-forward which was not possible in sequential models. BERT [6] introduced bi-directional learning and made transfer learning easier. Tasks like QA benefited by incorporating text from both the directions. Training of BERT consists of separate *pre-training* and *fine-tuning* steps. In pre-training, unlabelled data is used to train on Masked Language Modelling (MLM) and Next Sentence Prediction (NSP) tasks whereas in fine-tuning task-specific labelled data is used to fit the model for a specific task.

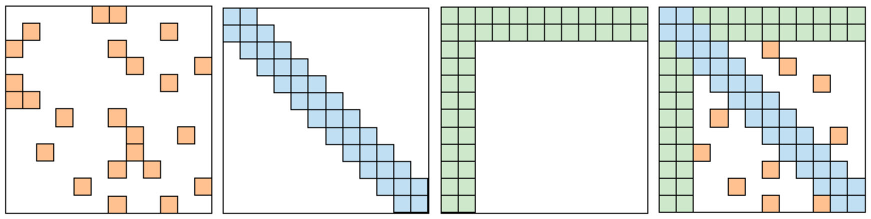
BERT based pre-trained models achieved great success in language understanding tasks but language generation was not explored much. Chen et al. [3] utilized BERT for language generation problems by introducing Conditional Masked Language Modeling (C-MLM). It consisted of a fine-tuned *teacher* BERT that improved a conventional *student* Seq2Seq model. Machine translation and text summarization tasks benefited the most from this approach. Usual auto-regressive tasks were only aware of the local context while generating the next token, but by having a global BERT teacher, the model could incorporate its knowledge to predict tokens that have better global coherence. The BERT acted as a regularizer over the output of the Seq2Seq model. The authors named this as Knowledge Distillation process.

By now, Transformers were actively used in all NLP tasks but their expressive powers were not analyzed. Yun et al. [19] did the analysis and found that Transformers are indeed universal approximators of all sequence-to-sequence tasks. Even after knowing this capability, they could not be used for some applications that involved dealing with longer text sequences. This was because of the quadratic memory and computation dependency on the input sequence length. After many attempts to reduce the complexity, first to $O(n^{3/2})$ by Child et al. [4] who proposed sparse models and then to $O(n \log n)$ by Kitaev et al. [9] who used locality sensitive hashing to compute nearest neighbours, Beltagy et al. [1] introduced Longformer in a contemporaneous work. It was a variation in Transformers which reduced the complexity to $O(n)$ and unlocked applications involving much longer sequences. The attention mechanism of Longformer was a drop-in replacement for the standard self-attention and combined local window attention with task motivated global attention. It continued its training from the checkpoint released by RoBERTa [11]. It introduced three attention patterns that replaced the full-attention of standard Transformer models. (1) Sliding Window - Given a fixed window of size w, each token attended to $w/2$ tokens on each side. These gave importance to local context. w could be fine-tuned for each layer to balance efficiency and model representation. (2) Dilated Sliding Window - this was equal to sliding window but had d sized gaps in the window due which token could attend to farther tokens without increasing the computation. For different heads, different d values could be set. (3) Global Attention - in BERT models we saw that the entire sequence representation was aggregated into special `[CLS]` token. This gave the importance of global attention tokens. Longformer specified certain tokens as global according to the task. For e.g., in QA, all the question tokens will be global.

3 RQ1: How to reduce attention count in transformers?

We want to investigate if quadratic attentions are a necessity in transformers and how do we keep lesser number of attentions while maintaining similar empirical performance on common NLP problems. Can we exploit some known properties of the data set or language structure to find some probable ways? Could the problem be re-formulated in a way which allows us to apply existing techniques and improves our understanding about attention connections?

Due to quadratic complexity, transformers can roughly handle only 512 tokens on common hardware which reduces its applicability on many NLP problems like QA, document classification and document summarization involving longer sequences. A lot of work has been done in this direction and complexity has been reduced in [4], [9], [18] and [13]. But these techniques use heuristics to improve performance, introduce complex architectures, and perform well only for limited datasets. They are not as generic as the original transformer.



(a) Random attention (b) Window attention (c) Global attention (d) Bigbird attention

Figure 1: Different attention mechanisms in transformers. (a) shows random attention with $r = 2$, (b) shows local window (lattice) attention with $w = 3$, (c) shows internal global attention with $g = 2$, and (d) shows a combination of all attention types which we call as Bigbird attention model.

We note that the self-attention of transformer between query and key vectors can be viewed as an adjacency graph. Let $\mathcal{D}$ be graph having n vertices and A its adjacency matrix representation. Then, $A \in [0, 1]^{n \times n}$ where $A(i, j) = 1$ iff query i attends to key j and zero otherwise. For the original transformer, A is a matrix of all ones. With this reformulation, reducing the number of attentions is now equal to a *graph sparsification problem*. We know that quadratic attentions perform well. We can decompose this interaction into multiple simpler types of interactions, each with lesser complexity, and analyze their individual performances and their combination to see if we can localize good performance to any specific interaction type or a set of interaction types. Below we present one hypothesis which tests the combination.

3.1 H: BigBird attention can approximate full-attention

The BigBird attention model uses a combination of three attention patterns between the query and key vectors - random sparse attention, local window attention and global attention. We hypothesise that such a connection can learn the important relationships between query and key vectors and lead to good performance. Unlike original transformers, this only has complexity linear to the number of token. This is a *causal hypothesis* because we propose that using BigBird attention leads to performance improvement. Due to its design, it certainly reduces memory and computation complexity. Figure 1 shows the 3 components that make the BigBird model. We explain them briefly in subsequent paragraphs.

A simple random graph construction is given by the Erdős-Rényi model, where each edge is chosen independently with a fixed probability. Such a graph has $\tilde{\Theta}(n)$ edges and the average length between any two nodes is $\log(n)$. Such a random graph approximates the complete graph spectrally and its second eigenvalue is quite far from the first eigenvalue [2]. This suggests that information can flow fast between any pair of nodes. We propose sparse attention where each query (node) attends over r random keys (nodes) i.e., $A(i, .) = 1$ for r random nodes.

Most contexts within NLP have data that shows a great deal of *locality of reference*. That is, a lot can be inferred about a token by looking at its neighbouring tokens. In graph theory, this can be related with clustering coefficient of a node which is high if there are more cliques or near cliques in the graph. Simple Erdős-Rényi models do not have high clustering coefficient but regular lattices, as mentioned in [16], have so. We use a clustering window length w where each token attends to $w/2$ tokens on either size.

The idea of global tokens was introduced in Beltagy et al. [1] which itself took ideas from BERT that uses a single global `[CLS]` token that aggregates the representation of the whole input sequence. Global tokens are also needed to prove that sparse transformers, just like original transformers, are universal approximators of seq2seq models in the style of Yun et al. [19] and are Turing complete [20]. We choose g global tokens that attend to every other token.

4 RQ2: What is the expressive power of sparse transformers?

We are interested in knowing how a sparse transformer compares with original transformer in terms of expressiveness. Yun et al. [19] showed that full transformers are universal approximators of seq2seq models but that doesn't apply to the sparse variation. If the expressive power is low then it is of interest to know which class of problems would degrade in performance and under what conditions. Zaheer et al. [20] show theoretically that sparse transformers are indeed universal approximators of seq2seq models and are also Turing complete. Without doing the theoretical work, we can put the below hypothesis to check the expressive power of sparse transformers.

4.1 H: A variety of NLP tasks show comparable performance

We hypothesise that the BigBird architecture works well for a variety of NLP tasks and is not a task specific optimization. The resultant transformers are robust just like the original ones. This is a *causal hypothesis* because we are proposing that the sparsification techniques of BigBird are able to learn important language features in a generic way and result in good performance for all common NLP tasks.

5 RQ3: Is longer context better for NLP tasks?

If we have to choose between denser and longer attention models, which one would we choose and what would be the basis for selection? The answer could be dependent on the type of problem - summarization, QA, classification, etc. For summarization there are two reasons to believe that longer documents are better. First, documents can have their salient features evenly distributed across the text so a good summary would need to include every important part of it. Second, longer documents have a richer discourse structure so looking at more content can help in grabbing the central ideas better. Similar arguments can be made for QA where longer documents can have an initial answer in the first part of the document which is revised later on. Or trivially, the answer is present only in later parts of the document. We try to understand the answer to this question in the context of one type of NLP task - QA.

5.1 H: Longer context is beneficial for QA problems

We put a rather limited hypothesis that longer context is better than shorter for QA datasets when the density of attentions is kept constant. This is a *causal hypothesis* because we are asserting that longer input context leads to better model performance.

6 Methodology

6.1 Dataset

For analysing RQ1 we perform long document classification, more specifically sentiment analysis, using IMDB movie review dataset given by [12]. It consists of 50,000 reviews from IMDB, containing no more than 30 reviews per movie. It has an even number of positive and negative reviews and contains only highly polar reviews. Neutral reviews are not included in this dataset.

For analysing RQ2 and RQ3 we perform QA task using TriviaQA dataset [8]. It is a reading comprehension dataset containing over 650K question-answer-evidence triplets. It has 95K question-answer pairs that are answered by trivia enthusiasts and their supporting evidence documents have been collected independently with 6 evidences per question on an average.

6.2 Experimentation

To evaluate the hypothesis that BigBird attention is a good approximation to full-attention, we use the transformer based architecture used by Zaheer et al. [20] and released openly and use it to perform long document classification on IMDB movie reviews. We vary the variable `attention_type` which takes two values *original_full* and *block_sparse*. The former uses original quadratic attentions of transformer whereas the latter uses the hybrid model explained in 3.1. Both models are initialized with the publicly available RoBERTa parameters [11] and then fine-tuned for our movie review task for a fixed number of iterations. We do not tune other hyperparameters of the model and use them as in [20]. The model is check-pointed after every 1000 batches of the training data. We note down the training time along with loss value at every checkpoint. Table 1 shows the training loss for BigBird

Checkpoint	**Training Loss**
ckpt-0	-
ckpt-1000	0.0236
ckpt-2000	0.0047
ckpt-3000	0.0040
ckpt-4000	0.0040
ckpt-5000	0.1290
ckpt-6000	0.0027
ckpt-7000	0.0025

(a) BigBird attention model.

Checkpoint	**Training Loss**
ckpt-0	-
ckpt-1000	0.0148
ckpt-2000	0.0113
ckpt-3000	0.0176
ckpt-4000	0.0079
ckpt-5000	0.0048
ckpt-6000	0.0027
ckpt-7000	0.0329

(b) Full-attention model.

Table 1: Training loss for the two fine-tuned models after every checkpoint.

	BigBird attention		Full attention	
Checkpoint	**Loss**	**F1**	**Loss**	**F1**
ckpt-0	0.7901	0.4999	0.7034	0.5000
ckpt-1000	0.1703	0.9408	0.1667	0.9392
ckpt-2000	0.1534	0.9467	0.1716	0.9364
ckpt-3000	0.1524	0.9466	0.1447	0.9481
ckpt-4000	0.1719	0.9443	0.1511	0.9489
ckpt-5000	0.1363	0.9499	0.1699	0.9429
ckpt-6000	0.1605	0.9498	0.1732	0.9477
ckpt-7000	0.1733	0.9495	0.1324	0.9506

Table 2: Loss and F1 scores at various checkpoints of the two types of models for test data.

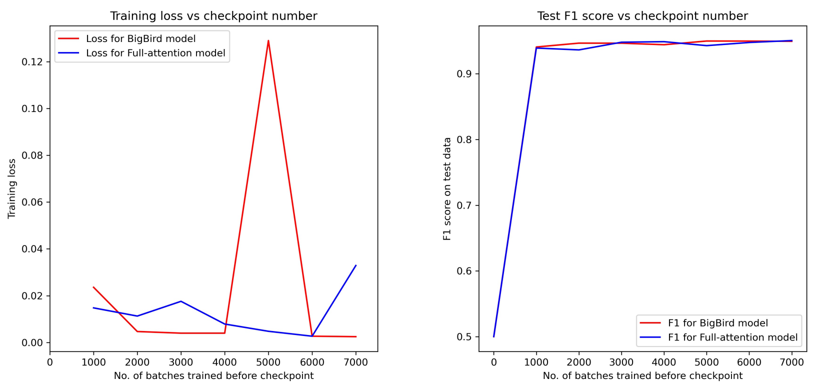
and full-attention based models. During inference, we find loss and F1 score on test data for every model checkpoint. Test results are shown in table 2.

We test the hypothesis that BigBird attention model gives comparable performance on many NLP tasks by evaluating a QA problem using TriviaQA dataset as we have already checked its performance on a classification problem. For baseline, we select F1 score obtained by the base RoBERTa model on the entire test set as reported in [20]. For checking BigBird performance, we use the Huggingface open source implementation of BigBird [17] and use a pre-trained model for this particular dataset. For validation we use a small subset (2064 examples) of the TriviaQA dataset having non-empty context because the original dataset would take 6+ hours to evaluate. We fix the input sequence length to 4096. We use a RoBERTa like model for QA which means that it does not generate answers in an auto-regressive way but rather gives us the start position and length so that we can extract answer from the input context (the paragraph from which to find the answer). The predicted answer and the expected answer usually do not match exactly so we create a `matching` function which normalizes and expands the strings before comparison to produce a better comparison result. Dataset characteristics and evaluation results can be see in table 3.

Dataset	**Training examples**	**Test examples**	**RoBERTa F1**	**Sampled Test examples (len>0)**	**BigBird accuracy**
TriviaQA	61,888	7993	74.3	2604	66.23

Table 3: Number of training and test examples in Trivia-QA dataset, F1 score obtained on all test examples from the baseline RoBERTa model, and accuracy obtained on sampled test data using BigBird model.

To evaluate our third hypothesis that longer context is better for QA problems, we use the same TriviaQA dataset and Huggingface pre-trained model of BigBird. We use a subset of the test data and find model accuracy for various input embedding lengths. We expect the accuracy to go up as the input sequence length increases. We also create a new *short* dataset from this subset where the number of tokens in each context are less than 4096. We evaluate this data also on various input



(a) Training data loss (b) Test data F1 score

Figure 2: Training data loss and Testing data F1 score at different checkpoint intervals for BigBird and full-attention models. Models were checkpointed after every 1000 batches.

embedding lengths. If the input context is smaller, we expect model to perform better as less input will be truncated to fit into the model. For e.g. when the embedding length is 4096, no input from the short dataset will be truncated whereas all longer inputs from the original subset will be truncated.

7 Results

In this section we discuss and analyse the results of all experiments, check the validity of our proposed hypotheses, and try to answer the related research questions.

RQ1: For classification problem, during training time we notice that BigBird model converges much faster than the original model. Graph $2(a)$ shows the comparison of training loss values for both models vs the number of batches seen. Notice that the bigbird model achieves training loss of 0.0047 in checkpoint ckpt-2000 whereas the full-attention model reaches that range in ckpt-5000 where it gets loss 0.0048. Apart from being faster to train, the inference produced by BigBird model is also comparable to the original model. We don't observe any performance degradation due to less attentions. Graph $2(b)$ shows the F1 score on test data for both models for each 1000-batch checkpoint. We retain our hypotheses that BigBird attention can approximate full-attention in a transformer model.

Answer: Using sparse attention architecture as proposed by BigBird is faster to train, uses less memory and computation, and gives comparable inference performance for NLP problems.

RQ2: The accuracy of BigBird model for QA dataset does not match the base model as seen in table 3. We expected the model to perform at par with the base model. There are two major reasons for this observation. First, the base model evaluated the entire test dataset whereas BigBird only looked at a subset of it due to compute limitations. Second, the evaluation methods especially the comparison of predicted and expected answers is different for both tests. We used a rather simple method to compare results by using the `matching` function as explained before whereas the base model used sophisticated techniques such as splitting long contexts, etc. Also, note that we used accuracy metric for evaluation whereas the base model used F1 score. Since the model obtained a decent score despite the testing limitations, we retain our hypothesis that BigBird attention works for a variety of NLP tasks.

Answer: We cannot dismiss the claim that BigBird sparse attention is not a task specific optimization and performs well for all NLP problems. We also mention that better tests need to be done to compare its performance with existing baselines for various NLP problems.

RQ3: Experimental results can be seen in table 4. As expected, for a given dataset, the model accuracy goes up when the input embedding length is increased. This is more clearly visible for

	Embedding length		
Test data	2048	3000	4096
All (#2068)	61.68	64.24	66.23
Short (#521)	82.92	82.15	83.88

Table 4: Exact match accuracy of BigBird model against short and complete test data for various input sequence (embedding) lengths.

the *all* dataset. For *short*, we see a small dip from 2048 to 3000 length which could be because of less number of examples tested. Also, for a given embedding length, the accuracy is higher if the input context is shorter which naturally requires less pruning before passing into the model. These observations give more weight to our hypothesis that longer context improves NLP performance if attention count is kept constant. Specific to QA, this makes sense because the answers to questions in longer documents could be in later parts of the document. Also, due to richer discourse structure in longer documents, the answer mentioned in the initial sections might be revised by a statement later in the document.

Answer: Utilizing longer input context gives better performance for NLP problems.

8 Challenges in Reanalysis

Reanalysis of the original paper [20] was made possible because of publicly released code and associated data by the authors. It would not have been easy or even possible without that. But while we performed the reanalysis, we faced challenges that impacted the quality and extent of experimentation we could perform. We mention some of those here.

- Training and evaluation take a long time. 1000 batches of long movie reviews take 10 min to evaluate. The original paper trains for 16,000 batches while we only perform partial training.
- There are existing issues (due to library upgrades) in running evaluation task on the PubMed dataset for summarization. Only training is happening without issues. For ArXiv data, we could not even perform training. This prevented us from analysis BigBird on a summarization task.
- Much of the codebase is still a black box for us and it's not clear how hyperparameter tuning was done by the authors. The current code has the correct model architecture with tuned hyperparamters already hard-coded into it. The experimentation which requires us to tune the hyperparameters (window size w, number of random attentions per node r) cannot be done without further understanding the codebase.
- Since the paper proposes a different transformer attention scheme, it is not using transformer as a black-box. Making changes in the attention scheme to try out only random connections, only lattice connections, a combination of both, and with ITC/ETC global connections requires a much deeper understanding of both the transformer code and how that is positioned in the codebase. There is no lever available to turn these features on/off and test them for various values.
- The original paper performs a lot of experiments but provides scripts for replicating only a handful of those, and that too in limited ways. We put many more hypotheses than what we see above in the paper but could not test them due to limited code functionality and ultimately had to omit them.
- The code doesn't compile locally and requires google cloud infrastructure to run. There is no feature to perform local debugging and easily inspecting different variables at runtime. This hampers quick understanding and changes in the code. It limits our ability to put and test various hypotheses.
- The paper suggests to run on 32 core v3 type TPU (multiple TPUs for lower running times) but we could only use one 8 core v2 type TPU because of budget constraints.

9 Conclusion

We performed a partial reanalysis of the BigBird architecture paper that brings down the memory and computation complexity of transformers from quadratic to linear in the input sequence length.

We both trained the model from an existing checkpoint and used pre-trained model to evaluate our hypotheses in order to answer some research questions asked in this paper. With our results, we understand the importance of global tokens in sparsification. We can say confidently that sparsification is possible without compromising on test data accuracy. The proposed architecture can be used as a plug-in replacement in all transformer based models that are employed for a variety of NLP problems. By using the same hardware, up to 8x the existing sequence length can be supported which will improve test accuracy. Without modifying context length, smaller hardware can be used to get similar performance. These findings are helpful for deploying BERT based models locally in resource constrained environments like mobile devices, robots, virtual assistants, etc. A lot of BERT-based models initialize from a pre-trained model and fine-tune on a specific task. Having a smaller model will make training and inference faster and more efficient in such 'transfer learning' scenarios.

10 Future Work

While we were able to verify some of the claims made in the original paper, we did not cover a lot of them due to issues mentioned in section 8 and time constraints. In this section, we state some hypotheses that were explored in the original paper but omitted by us. In a future work, these can be verified. We knowingly omit studies that were done by the authors but are not related to NLP area for e.g. applications of this work on genomics data.

- While evaluating RQ1 we put up a hypothesis to use BigBird attention which consists of 3 types of attention patterns. We did not evaluate the benefits of each of them in isolation and in pairs. If we do that, we can identify the contribution of each in the final performance. We mention those hypotheses here. (1) Random sparse connections between nodes learn important language features. (2) Connections resulting in high clustering coefficient learn important language features. (3) Having both random sparse connections and high clustering coefficient is better than having only one of them.
- While evaluating RQ2 we verified the performance of BigBird on a QA dataset. We should also verify its efficiency on short and long document summarization problems. To have further confidence in the generalization of BigBird, we can evaluate each problem (QA, summarization, classification) on multiple datasets.
- While evaluating RQ3 we put up a very limiting hypothesis by keeping the attention count consistent. We can improve it further by (1) Comparing two models where the first one uses longer context and sparse attentions, and the second one uses shorter context and dense attentions. (2) Doing this experiment for various NLP problems with multiple benchmark datasets.

11 Acknowledgements

This work has been done as part of 691DD course project during Spring 2021. We would like to thank Dr. David Jensen for providing regular actionable feedback. We also thank the original authors for making their code and data publicly available, and the huggingface team for open-sourcing their BigBird re-implementation.

References

[1] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

[2] Florent Benaych-Georges, Charles Bordenave, and Antti Knowles. Largest eigenvalues of sparse inhomogeneous erdos-rényi graphs, 2017.

[3] Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. Distilling knowledge learned in bert for text generation, 2020.

[4] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[8] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, art. arXiv:1705.03551, 2017.

[9] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020.

[10] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for lstm networks, 2018.

[11] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[12] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 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, 2011.

[13] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding, 2020.

[14] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017.

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

[16] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. `https://doi.org/10.1038/30918`, 1998.

[17] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

[18] Zihao Ye, Qipeng Guo, Quan Gan, Xipeng Qiu, and Zheng Zhang. Bp-transformer: Modelling long-range context via binary partitioning, 2019.

[19] Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions?, 2020.

[20] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences, 2021.