READTWICE: Reading Very Large Documents with Memories

Yury Zemlyanskiy*
University of Southern California
yury.zemlyanskiy@usc.edu

Joshua Ainslie
Google Research
jainslie@google.com

Michiel de Jong
University of Southern California
msdejong@usc.edu

Philip Pham
Google Research
phillypham@google.com

Ilya Eckstein
Google Research
ilyaeck@google.com

Fei Sha†
Google Research
fsha@google.com

Abstract

Knowledge-intensive tasks such as question answering often require assimilating information from different sections of large inputs such as books or article collections. We propose READTWICE[1], a simple and effective technique that combines several strengths of prior approaches to model long-range dependencies with Transformers. The main idea is to read text in small segments, in parallel, summarizing each segment into a memory table to be used in a second read of the text. We show that the method outperforms models of comparable size on several question answering (QA) datasets and sets a new state of the art on the challenging NarrativeQA task, with questions about entire books.

1 Introduction

Transformer-based models such as BERT are very effective in capturing long-range dependencies in text passages through the attention mechanism (Vaswani et al., 2017; Devlin et al., 2019). However, the amount of compute in attention depends quadratically on the number of tokens in an input text passage. As such, the standard BERT implementation limits input size to a fixed number (often 512) of tokens.

In reality, dependencies over significantly longer ranges are common and modeling them is crucial. For instance, in a sentence like *Inside the Sammath Naur, the Ring-bearer struggled to throw the Ring into the volcano*, the narrative interweaves several prior storylines from a book. Comprehending this sentence therefore requires looking up previous mentions of *Ring-bearer* and *Sammath Naur*, located many tokens away.

Several methods have been proposed to address this challenge; see (Tay et al., 2020) for a survey and §3 for a detailed discussion. One popular strategy is to reduce the number of tokens attended to. Longer inputs can in fact be processed in this way – but only up to a limit of around 5,000 tokens, as used in (Ainslie et al., 2020; Zaheer et al., 2020; Beltagy et al., 2020) – far below the context sizes required to model long documents such as books.

Another strategy such as HIBERT (Zhang et al., 2019) splits inputs into smaller segments which are processed individually, then assembled into a hierarchical representation. As a downside, intersegment context is unavailable during encoding.

We propose READTWICE, a simple approach that combines the strengths of both strategies. As its name suggests, the main idea is to process the input twice: a long text input (such as a document, or even a book) is treated as a collection of shorter text segments which are read independently and in parallel. Then, the encoder reads each segement again, now *augmented* with compressed information from other segments.

The crucial component in READTWICE, as illustrated in Figure 1, is a memory module that holds compressed information from all segments. That compressed information is used only *once*: in the second pass. Thus, READTWICE is much more computationally efficient than models like ETC that rely on memory for all segments, in every layer. While READTWICE requires two passes, it differs from hierarchical models such as HIBERT that do not condition segment encoding on other segments. §3 contrasts these approaches in more detail.

We validate the efficacy of READTWICE on extractive question answering (QA) tasks, showing strong performance on HotpotQA (Yang et al., 2018), TriviaQA (Joshi et al., 2017) and Narra-

*Work is done while at Google

† On leave from University of Southern California (feisha@usc.edu)

[1]Source code and pre-trained checkpoints for READTWICE can be found at https://goo.gle/research-readtwice.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5189–5195
June 6–11, 2021. ©2021 Association for Computational Linguistics

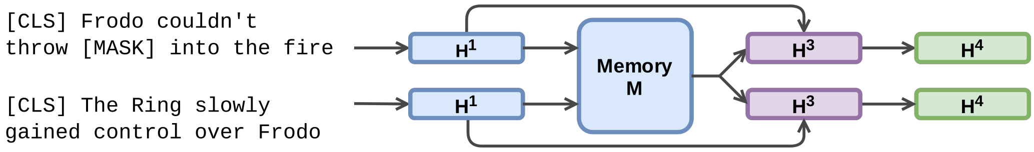

Figure 1: READTWICE model architecture. The input is processed twice, with a memory table for inter-segment information sharing.

tiveQA (Kociský et al., 2018). In particular, READTWICE significantly improves the state-of-the-art on QA based on *entire books* in NarrativeQA, with absolutes gains of 4.5 ROUGE-L points and 3 BLEU-1 points (relative improvements of 23% and 17%, respectively).

2 Method

We first describe the READTWICE model, followed by its pre-training procedure.

2.1 READTWICE

The model reads a large text document split into N segments $x_1, \ldots, x_N$; each x_i is limited to 512 tokens, as in a typical BERT model.

The model architecture is depicted in Figure 1. In the first read, each segment is encoded independently with standard BERT. Then, memories are extracted from each segment—a process we describe in detail later—and gathered into a global memory pool. For the second read, a `MemoryAttention` layer (with a residual connection and a `LayerNorm` on top) is first used to merge the information from the former intra-segmental contextual token embeddings and the global memory. The merged result is then read by another small BERT model with only *two* Transformer layers to produce the final output. The rationale is that the first read already generates rich contextualized embeddings, and the second read only needs to incorporate information from the memory. More formally:

$$
\begin{aligned}
H_i^0 &= \texttt{TokenEmbed}(x_i), H_i^1 = \texttt{BERT}_1(x_i), \forall\, i \\
M_i &= \texttt{ExtractMemories}(H_i^1), \forall i \\
M &= \texttt{Gather}([M_1, \ldots, M_N]) \\
H_i^2 &= \texttt{MemoryAttention}(H_i^1, M), \forall i \\
H_i^3 &= \texttt{LayerNorm}(H_i^1 + H_i^2), \forall\, i \\
H_i^4 &= \texttt{BERT}_2(H_i^3), \forall\, i
\end{aligned}
$$

Next, we describe the newly introduced layers.

`ExtractMemories` and `Gather` Our aim is to compress the information in each segment and disseminate it to other segments to be used in the second read. We consider three types of memories:

- READTWICE (`CLS`). One obvious choice is to use the CLS token representation associated with segment x_i as a summary of the segment.
- READTWICE (`STS`). To obtain more fine-grained memories, we extract a memory vector for each consecutive span of 32 tokens. Contextual embeddings of each span's first and the last tokens are concatenated and linearly projected to a single point in the token vector space as the span representation. The projection matrix is learned end to end.
- READTWICE (`E`). In another variant of span-based memory, we memorize representations of entity mention spans. To obtain these spans, we first annotate each segment with an external Named Entity Recognition system. Then, each entity mention span is encoded in the same way as in READTWICE (`STS`). This design is motivated by the intuition that long-range dependencies primarily occur between entities.

Empirically, we find that READTWICE (`E`) leads to best performance (see the ablation in Section 4.4) and it is the memory type used in our headline results.

We collect all memories from all segments into a flat memory table. The table size is given by the number of segments (`CLS`), the number of 32-token spans (`STS`), or the number of entity mentions (`E`).

`MemoryAttention` In this layer, we let contextual token embeddings from individual segments interact with other segments' memories via dot-product attention over the memory table.

Let h_{ij} be the contextual embedding of token j in segment i after the first read. And let m be a memory table entry whose source segment is given by m_s. We then define its attention weight as:

$$\alpha_m = \frac{e^{h_{ij}^T M_m + r_{i,m_s}}}{\sum_m e^{h_{ij}^T M_m + r_{i,m_s}} + e^{h_{ij}^T M_0}} \tag{1}$$

where M_0 is a learnable no-op memory not associated with any specific text. r_{i,m_s} is a learned position score which captures the relative distance between segment i and the memory M_m, akin to Shaw et al. (2018):

$$r_{i,m_s} = \omega(\mathtt{dist}(i, m_s)) \tag{2}$$

where ω is a set of weights indexed by the distance

$$\mathtt{dist}(i, m_s) = \begin{cases} -B & i - m_s < -B \\ B & i - m_s > B \\ i - m_s & \mathtt{otherwise} \end{cases} \tag{3}$$

where the cutoff threshold B clips the effect of distance to $[-B, B]$. We set B to 10 in this work.

Finally, the `MemoryAttention` layer output for a given token is given by

$$h_{ij}^2 = \sum_{m=1} \alpha_m M_m \tag{4}$$

2.2 Pre-training

We pretrain READTWICE similarly to (Devlin et al., 2019), using the Wikipedia and BooksCorpus datasets. When entity mentions are used in the memory table, the texts are processed with the Entity Linking (EL) and Named Entity Recognition (NER) tools from the Google Cloud NLP API[2]. Moreover, we use existing hyperlinks in Wikipedia as additional entity annotations. The first and the second BERT readers are trained end-to-end.

Our pre-training objective is the standard Masked Language Model (MLM) task, with the MLM prediction loss computed based on the output of the second reader.

In order to encourage the model to rely on the memory, we increase the difficulty of the MLM task. Following the entity masking procedure in (Guu et al., 2020; Sun et al., 2019), we mask entity mention tokens more aggressively at a 25% rate and jointly mask all tokens within a mention. By contrast, for non-entity tokens, we mask contiguous sequences of random length at a 15% rate.

[2] https://cloud.google.com/natural-language/docs/basics#entity_analysis

3 Related Work

One way to extend the limit on input size is by reducing the number of tokens attended to. ETC (Ainslie et al., 2020) and LONGFORMER (Beltagy et al., 2020) allow standard attention only between tokens within a fixed distance. To allow information flow over longer distances, they use auxiliary global "memory" tokens which attend to all regular tokens and vice versa. BIGBIRD (Zaheer et al., 2020) additionally has each token attend to a random subset of other tokens. While reducing asymptotic complexity from quadratic to linear (in input size), these global tokens are added at each attention layer, incurring a high computational cost.

Another approach is to split the input into multiple segments and then aggregate information across segments. This is achieved through hierarchical modeling (Chang et al., 2019; Zhang et al., 2019). While reducing the attention size to the number of segments, each individual segment has no information about its siblings during token-level encoding. Alternatively, recurrent models (Dai et al., 2019; Rae et al., 2019) read a large input from left to right, dynamically compressing faraway contexts, thus allowing unidirectional information aggregation (left to right). One disadvantage is that the input needs to be processed sequentially, which becomes time-consuming for producing contextualized representations of a large input.

Our method brings these lines of work together. Processing segments independently and in parallel, then memorizing their compressed representations and sharing memory across segments enables contextual embeddings to be updated based on faraway information. Enabling memory sharing only once—during the second read—allows it be done cheaply.

Note that the memory module here is internally generated from the input, as opposed to external memory models which are orthogonal to our approach (Peters et al., 2019; Févry et al., 2020).

4 Experiments

4.1 Pre-training setup

All READTWICE models are initialized with the public ROBERTA (base) checkpoint[3] adapted to Tensorflow by Rothe et al. (2020). Further, models are pre-trained for 1M steps on 64 TPU cores using the LAMB optimizer (You et al., 2020).

[3] https://dl.fbaipublicfiles.com/fairseq/models/roberta.base.tar.gz

Each batch contains 512 segments, with at most 128 segments per document. The segments are consecutive spans of 512 tokens. Therefore, the model can process documents up to 65k ($\approx 128 \times 512$) tokens. Each batch contains the maximum number of documents such that the total number of segments is at most 512. Approximately half of Wikipedia articles fit in one segment (thus not needing memory), with a fat tail of longer documents.

In terms of compute and memory overhead, READTWICE is about 30% slower than the ROBERTA-base model and uses 15M (or 12%) more parameters: 14M owing to the second read BERT_2 and 1M due to `ExtractMemories` and `MemoryAttention` layers.

4.2 Evaluation setup

We evaluate READTWICE on the downstream extractive question-answering task using several datasets: HotpotQA (HQA) (Yang et al., 2018), TriviaQA (TQA) (Joshi et al., 2017) and NarrativeQA (NQA) (Kociský et al., 2018).

In HQA, questions are based on relatively short text passages (2 evidence paragraphs), with eight additional distractor passages. In TQA, evidence text is medium-sized. NQA asks questions about entire books, requiring a successful QA system to model very long-range dependencies. The NQA dataset has an average of 62,000 words per document with a maximum of 400,000. Only 40% of NQA's answers are span-based – we use a ROUGE-L oracle as training labels for the other questions.

READTWICE is fine-tuned on each task. QA-specific heads are used to generate span-based predictions, consisting of fully-connected layers that take contextual embeddings from the second reader as inputs. These layers output a score for whether the corresponding tokens are the beginning or ending of an answer span. For a similar setup, see multi-segment based QA tasks (Clark and Gardner, 2018; Cheng et al., 2020).

During fine-tuning, batches contain 128 segments for all tasks (also with up to 128 segments per document). Every segment contains 512 tokens, but as neighboring segments have 128 token overlaps, the model can process documents of up to 49K tokens ($\approx 128 \times (512 - 128)$). For TQA and HQA, documents have approximately 10 segments. For NQA, we split the documents into sub-documents with 49k tokens and apply memory only within these sub-documents.

Model	HQA F1 (ans)	TQA F1(dev)	TQA F1(test)
LF	74.3	75.2	-
ETC	75.1	-	-
BIGBIRD	75.7	79.5	-
ROBERTA (us)	72.0	75.9	-
READTWICE-E	**75.9**	**80.7**	**80.9**

Table 1: Results on HotpotQA development set (**answer only** F1 score) and on TriviaQA development and test splits for the Wikipedia full setting. Additional test results are available on the public leaderboard[4]

We perform hyperparameter search only over learning rate $\lambda \in \{5e-6,\ 1e-5,\ 3e-5\}$ and train for 6 epochs with 10% warm up proportion. Moreover, we use early stopping based on the performance on the development set.

4.3 Main Results

Results for HQA and TQA are reported in Table 1. We compare to prior art (using reported results where available or from our own implementations otherwise, denoted as "us"): Longformer (LF) (Beltagy et al., 2020), ETC (Ainslie et al., 2020), BigBird (Zaheer et al., 2020), and ROBERTA (Liu et al., 2019). By default, we compare against the "base" configuration of those models where the number of parameters is comparable to BERT-Base, as is the case for READTWICE.

Table 1 shows that for small to medium sized text passages, the proposed READTWICE outperforms all models of comparable size.

Table 2 contrasts READTWICE to other methods on extremely large contexts: BiDAF (Kociský et al., 2018), R^3 (Wang et al., 2018), BM25 + BERT Reader / Ranker (Mou et al., 2020) and our own implementation of ROBERTA and ETC[5]. READTWICE significantly outperforms all previous work and establishes new state-of-the-art results, demonstrating the effectiveness of performing a second read conditioned on global memory for processing extremely long texts.

4.4 Ablation Analysis & Discussion

To isolate individual components' contributions, Table 3 contrasts several variants of READTWICE.

[4]See https://competitions.codalab.org/competitions/17208#results, tab "Wikipedia".

[5]For ETC we use the public (base configuration) checkpoint https://storage.googleapis.com/gresearch/etcmodel/checkpoints/etc_base_2x_pretrain.zip

Model	ROUGE-L	BLEU-1	BLEU-4	METEOR
BiDAF (Kociský et al., 2018)	6.3 / 6.2	5.8 / 5.7	0.2 / 0.3	3.8 / 3.7
R^3 (Wang et al., 2018)	11.4 / 11.9	16.4 / 15.7	0.5 / 0.5	3.5 / 3.5
BM25+BERT (Mou et al., 2020)	14.8 / 15.5	14.6 / 14.5	1.8 / 1.4	5.1 / 5.0
RoBERTa (us)	17.4 / 18.0	18.2 / 18.0	2.4 / 2.6	5.4 / 5.4
ETC (us)	18.3 / 18.8	16.1 / 17.2	2.4 / 2.7	5.4 / 5.4
ReadTwice (E)	**22.7 / 23.3**	**21.1 / 21.1**	**3.6 / 4.0**	**6.7 / 7.0**

Table 2: Results on the NarrativeQA's development / test splits.

These ablations lead to two key insights.

Inter-segment memory matters We introduce a variant ReadTwice-E(SS) (where SS stands for "Single Segment") to isolate the gains from the memory layer. ReadTwice-E(SS) prevents segments from attending to memories of other segments, thus disabling long-range dependency modeling. We observe that ReadTwice-E improves over ReadTwice-E(SS) on all tasks, modestly but non-negligibly for TQA, and significantly for HQA and especially NQA.

This matches our knowledge of those datasets: TQA questions are based on a relatively short context and can typically be answered using a single passage in the context document. HQA questions have a similarly sized context, but are explicitly constructed to require information from multiple paragraphs to answer, and ReadTwice shows accordingly larger gains. Finally, NQA has much larger contexts, and its questions generally require information from different parts of the document, increasing the importance of long-range dependency modeling and accordingly, the performance boost from ReadTwice.

Entities matter Entity mentions appears to be the most effective memory type in most experiments, leading to noticeably improved performance on both HQA and NQA. The difference is most pronounced in NQA whose particularly long and challenging contexts make it a perfect testbed.

Source of non-memory gains The non-memory gains over a baseline RoBERTa model originate from the two extra layers and the entity-based MLM objective. In order to disentangle the sources of gains we train the ReadTwice-E(SS) model using a 10-layer Transformer for $\texttt{BERT}_1$ (denoted as $\texttt{E}(\texttt{SS}, \texttt{10L})$ in Table 3), with the same number of layers as RoBERTa. While the gains from 2 extra layers are significant ($\texttt{E}(\texttt{SS})$ vs $\texttt{E}(\texttt{SS}, \texttt{10L})$), most of the gains appear to result from the custom pre-training procedure ($\texttt{E}(\texttt{SS}, \texttt{10L})$ vs RoBERTa).

Model	HQA	NQA-R	NQA-B	TQA
E	75.89	22.71	21.07	80.7
E(SS)	75.08	21.93	18.39	80.3
E(SS, 10L)	74.70	21.39	18.37	80.4
RoBERTa	72.00	17.40	18.2	75.9
CLS	75.32	20.89	17.80	80.6
STS	75.39	21.08	18.38	80.4

Table 3: Ablation studies on variants of ReadTwice on the dev sets. We report F1 (answer only) score for HQA, ROUGE-L and BLEU-1 for NQA (denoted -R and -B respectively) and F1 for TQA.

5 Conclusion & Future Work

ReadTwice performs well on several QA tasks, particularly NarrativeQA where long-range dependencies among entities appear to be very important. The proposed method is conceptually simple, easy to implement and is capable of reading entire books. For future work, we plan to explore new memory types, hierarchies and aggregation functions. We also aim to apply the model to other tasks, particularly long text summarization, likely to benefit from a memory-forming mechanism.

Acknowlegements

We thank Santiago Ontanon, Manzil Zaheer, Sudeep Gandhe, Anirudh Ravula, Bhargav Kanagal, Jules Gagnon-Marchand and Sumit Sanghai for insightful discussions, Mou et al. (2020) for a sample evaluation code for NarrativeQA and reviewers for their feedback. This work is partially supported by NSF Awards IIS-1513966/ 1632803/1833137, CCF-1139148, DARPA Awards#: FA8750-18-2-0117, FA8750-19-1-0504, DARPA-D3M - Award UCB-00009528, Google Research Awards, gifts from Facebook and Netflix, and ARO# W911NF-12-1-0241 and W911NF-15-1-0484.

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online. Association for Computational Linguistics.

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

Ming-Wei Chang, Kristina Toutanova, Kenton Lee, and Jacob Devlin. 2019. Language model pre-training for hierarchical document representations. *CoRR*, abs/1901.09128.

Hao Cheng, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2020. Probabilistic assumptions matter: Improved models for distantly-supervised document-level question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5657–5667. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 845–855. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Thibault Févry, Livio Baldini Soares, Nicholas Arthur FitzGerald, Eunsol Choi, and Tom Kwiatkowski. 2020. Entities as experts: Sparse memory access with entity supervision. In *EMNLP 2020 - Conference on Empirical Methods in Natural Language Processing*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Trans. Assoc. Comput. Linguistics*, 6:317–328.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Xiangyang Mou, Mo Yu, Bingsheng Yao, Chenghao Yang, Xiaoxiao Guo, Saloni Potdar, and Hui Su. 2020. Frustratingly hard evidence retrieval for QA over books. *CoRR*, abs/2007.09878.

Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 43–54. Association for Computational Linguistics.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Trans. Assoc. Comput. Linguistics*, 8:264–280.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 464–468. Association for Computational Linguistics.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. ERNIE: enhanced representation through knowledge integration. *CoRR*, abs/1904.09223.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. 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, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerry Tesauro, Bowen Zhou, and Jing Jiang. 2018. R^3: Reinforced ranker-reader for open-domain 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 5981–5988. AAAI Press.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Manzil Zaheer, Guru Guruganesh, 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. *CoRR*, abs/2007.14062.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: document level pre-training of hierarchical bidirectional transformers for document summarization. *CoRR*, abs/1905.06566.