Recurrent Memory Transformer for Incremental Summarisation of extremely long text

Kaifan Li Shuntaro Yada Eiji Aramaki
Nara Institute of Science and Technology
{li.kaifan.ln9,s-yada,aramaki}@is.naist.jp

Abstract

Pupular transformer-based language models are often constrained with a limited number of input length, resulting in subpar performance on the task of long-text summarisation. Prior work has attempted to alter the model's internal architecture to accommodate longer inputs. Even if a model supports longer input texts, limited RAMs in university laboratories and edge devices prohibit us from unleashing that input length. We, thus, explore the task of long-text summarisation based on Recurrent Memory Transformer (RMT) which provides an external memory for the transformer-based models without modifying the internal structure, and further proposed RMT-Summ. To demonstrate the validity of RMT-Summ, we introduce an incremental summarisation task, and built a dedicated dataset from PubMed medical articles containing structured abstracts. Our experimental results show that an RMT-Summ powered BART model performed better than the baseline original BART by 1.24 points in ROUGE-1.

1 Introduction

Large language Models (LLMs) play a pivotal role in NLP. Self-attention, as the foundation of LLMs, suffers from limitation of the input length due to the quadratic computational complexity. Pupular transformer-based language models are often constrained with a limited number of input tokens, such as around 1K to 4K. Since this constraint harms long-text processing tasks (e.g. summarisation and complex dialogue), previous work has attempted modifications to the self-attention mechanism to accommodate longer inputs. Despite the model's support for longer text inputs, the increased length places a burden on RAM, making the utilization of the model less feasible for typical university laboratories and edge devices.

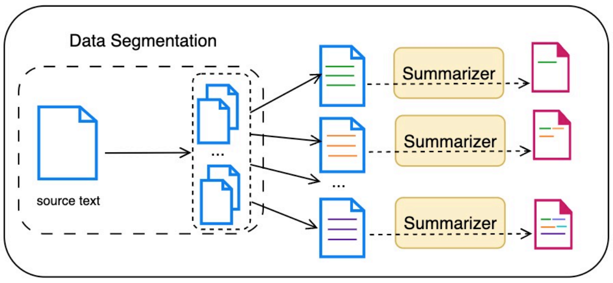

Figure 1 Overview of Incremental summarisation task: The source text is firstly segmented based on its discourse structure. Subsequently, each segment is fed into the summarizer to generate a summary that encapsulates all previously mentioned content.

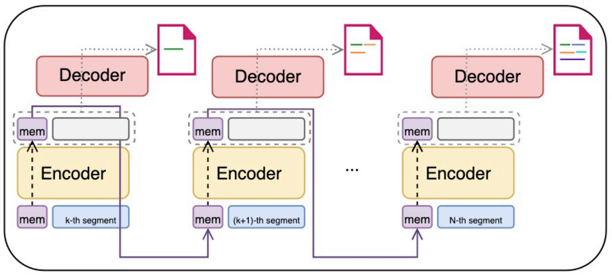

Figure 2 Overview of RMT-Summ structure for Incremental Summarisation task.

To address the former constraint, Recurrent Memory Transformer (RMT) [1, 2] was proposed, which is simple yet effective external memory mechanism for the models suffering from limited input length, by directly incorporating randomly initialized special tokens into the prefix of the input embeddings. Despite the excellent performance of RMT in Natural Language Understanding tasks such as enwik8 and WikiText-103, its effectiveness in Natural Language Generation remains to be validated. Hence, we apply the idea of RMT into an encoder-decoder architecture, aligning better with the sequence-to-sequence task. Furthermore, we refine the training objective in this context, referring to this variant as RMT-Summ. Specificlly, for an extremely long text, we initially segment it and then process each segment in turn. During this process, the memory tokens of RMT-Summ are inserted into the embeddings of the input to preserve the memory of previous segment through the depth propagation, and facilitate memory transmission between segments. RMT and RMT-Summ does not alter the internal structure of the model; therefore, in a plug-and-play manner, RMT-Summ can effectively integrates with various transformer-based models and other additional technologies.

To validate the effectiveness of RMT-Summ, we introduce a novel **incremental summarisation task** shown in Figure 1: a long text is inputted per segment; the model must output summarisations of the current segment along with all the past segments so far. Notably, this summarisation task deviates from just summation or cumulation of segmented summaries because it should occasionally modify (edit) the past summary, where the model must memorise the past segments.

Based on this definition, we created a sub dataset from the PubMed that contains structured abstracts for experiments. The experimental results showed a substantial enhancement in the performance of the RMT-Summ powered BART compared to the baseline original BART.

2 Related Work

The most straightforward approach to address the long-text summarisation is by directly truncating the input text, often achieved by directly extracting the initial segment up to the maximum input length of the PLMs. However, this method is significantly impacted by lead bias [3].

To minimize the information loss, it's essential to extend the length constraints of the PLMs while controlling computational resource consumption. Previous research has primarily attempted breakthroughs from two perspectives: Effective Attention (Sec. 2.1) and the Divide & Conquer strategy (Sec. 2.2).

2.1 Effective Attention

A popular method is to replace full-attention mechanism with fixed-patterns to achieve more effecient model inference. Longformer [4], as a seminal work, introduced a hybrid attention mechanism that utilizes a form of local attention known as slidng window attention. This alteration allowed Longformer to extend its maximum sequence length up to 16,384 tokens. BigBird [5] introduced random attention on top of Longformer, allowing each token to randomly access several tokens from other positions.

The fusion of BigBird and Pegasus [6] at that time resulted in a significant improvement and achieved the state-of-the-art (SOTA) performance across long-text summarisation datasets. LongT5 [7], upon the local attention, introduced Transient Global Attention, which avoids token selection or adding extra tokens for global attention. Instead, it dynamically constructs global tokens before self-attention at each step.

However, since these models reconstructed self-attention, retraining on large-scale corpora is required, which introduced additional computational overhead. Our RMT-Summ, as a plug-and-play approach, efficiently leverages pre-trained models and requires only limited computational resources for fine-tuning.

2.2 Divide & Conquer

Traditional text summarisation tasks often involve generating a single summary for an entire text. However, with the increase in text length, such as in scientific papers datasets [8], long texts are often structured by sections or chapters.

Multiple prior works attempted to decompose the task of long text summarisation into a multi-step process. DANCER [9] is a representative work, which initially divides long text into multiple non-overlapping segments based on semantic structures. Subsequently, each segment is individually modeled and processed. Finally, the generated summaries for each segment are concatenated together. SUMM^N [10] adopts a similar approach by initially segmenting documents into multiple sections. It generates coarse-grained summaries for each segment and subsequently integrates these coarse-grained summaries to produce finer-grained summaries.

Unlike prior research, we introduce an extra memory prefix that pass salient information throught a segment-level recurrence. This memory prefix efficiently transfers crucial information between different sections, enhancing the generation of more precise summaries.

3 Methods

3.1 Recurrent Memory Transformer

Since our proposed method RMT-Summ (Sec. 3.2) is highly based on the seminal work, Rucurrent Memory

Transformer (RMT) [1, 2], we introduce its foundation first. Compare to a prior long-text model called Transformer-XL [11], RMT utilizes a set of global tokens as memory units instead of directly caching the hidden states of the entire segment. RMT adopted a decoder-only architecture in their first work [1]. Due to the unidirectional nature of the decoder, within RMT, the memory unit is divided into read and write modules. The read cell in the prefix part serves as the continuation of memory from the previous segment, while the write cell acts as the storage for the memory of the current segment. Specifically, for time step τ and segment hidden representations H_τ^0, the segment-level recurrence implementation in RMT is realized by the following equations:

$$\tilde{H}_\tau^0 = [H_\tau^{mem} \circ H_\tau^0 \circ H_\tau^{mem}]$$

$$\bar{H}_\tau^N = Transformer(\tilde{H}_\tau^0)$$

$$[H_\tau^{read} \circ H_\tau^0 \circ H_\tau^{write}] := \bar{H}_\tau^N$$

Here N represents the number of transformer layers, while $\circ$ represente the concatenate operation.

In the model's embedding layer, the prefix and postfix memory cells are concatenated with the input text's embeddings. Subsequently, they undergo a vertical passage through the transformer. During this process, the write tokens from the postfix progressively retain the memory of the current segment as the network passes through to achieve the depth-level propagation. Subsequently, to achieve segment-level recurrence, the write tokens of the current segment are passed on to the next segment:

$$H_{\tau+1}^{mem} := H_\tau^{write}, \tilde{H}_{\tau+1}^0 = [H_{\tau+1}^{mem} \circ H_{\tau+1}^0 \circ H_{\tau+1}^{mem}]$$

3.2 RMT-Summ

Text summarisation, as a sequence-to-sequence task, is more suitable for application to a sequence-to-sequence model rather than encoder-only or decoder-only structures [12]. In an encoder-decoder architecture, the sentence representation is encoded in the encoder and the target sentence is generated in the decoder. Due to the bidirectional nature of the encoder, a single memory cell can simultaneously serve both read and write functions. Based on the aforementioned concept, we only insert memory prefix into the encoder following the latest work of RMT [2], to retain the memory of the current segment and circulate between segments, while we utilize an addtion decoder for summary generation. Specifically, in the context of an encoder-decoder architecture, the update of memory and the propagation of our RMT-Summ occur as follows:

$$\tilde{H}_\tau^0 = [H_\tau^{mem} \circ H_\tau^0]$$

$$\bar{H}_\tau^N = Encoder(\tilde{H}_\tau^0)$$

$$[\bar{H}_\tau^{mem} \circ H_\tau^0] := \bar{H}_\tau^N$$

$$\tilde{T}_\tau = Decoder(\bar{H}_\tau^N, T_\tau)$$

$$H_{\tau+1}^{mem} := \bar{H}_\tau^{mem}, \tilde{H}_{\tau+1}^0 = [H_{\tau+1}^{mem} \circ H_{\tau+1}^0]$$

Notably, in contrast to the traditional summarization task, our optimization objective is tailored for the incremental summarization task, as elaborated in Section 4.

4 Incremental Summarisation

To align RMT-Summ with summarisation task, in this work, we propose the concept of incremental summarisation. Specifically, given an input document D comprising multiple segments $S = \{S_1, S_2, ..., S_N\}$, the corresponding summary y can also be segmented into multiple parts $y = \{y_1, y_2, ..., y_N\}$. For each input segment S_τ, our training target should be $T_\tau = edit(concat(y_1, y_2, ..., y_\tau))$. It is important to emphasize here that the function ***edit*** implies that in practical scenarios, our summaries may not simply be a straightforward concatenation of segment summaries but a reintegration process as shown in Figure 1.

5 Experiment Setup

5.1 Data

PubMed [13] is a large scale long scientific dataset collected from PubMed.com, which encompassing a wealth of biomedical and life science literature and journals. Moreover, due to the presence of structured abstracts within certain papers in PubMed, it is well-suited for validating our incremental summarisation task. We applied specific preprocessing to this dataset and filtered out a subset, which we named **PubMed-IncreSumm**.

Specifically, following the methodology outlined in this work [14], we extracted the main sections from the body text and segmented it into four parts based on the section headings. Additionally, for some literature, the abstract sections could be explicitly categorized into "Introduction", "Methods", "Results", and "Conclusion" through

Section	Keyword
Introduction	introduction, case, objective, purpose, . . .
Method	methods, materials, methodology, . . .
Results	result, experiments, observations, . . .
Conclusion	conclusion, discussion, summary, concluding, . . .

Table 1 Keywords used to match the corresponding sections. If a section heading is not found in this keyword list, we disregard it. Additionally, we performed the same grouping operation for the abstract sections.

keyword matching, as specified in Table 1. We filtered out such articles accordingly.

From PubMed's train, test, and valid sets, we acquired 24,843, 1,399, and 1,431 data entries, respectively as PubMed-IncreSumm. We employed the NLTK [15] tokenizer to conduct text length statistics for all datasets. The average length of the source text was 2,736.26, while the final abstracts had an average length of 298.68.

5.2 Model & Implementation settings

We opted to employ BART [16] (i.e. 'facebook/bart-base' available on Hugging Face) as the backbone model because of its excellent performance in short text summarisation tasks. The sample counts for the training set, validation set, and test set were 24,843, 1,399, and 1,431, respectively. We set the learning rate to 5.0×10^{-5}, beam search width to 4, and the maximum text length for each section's main text to 512, with a maximum generation length of 300. We utilized the F1-score of ROUGE-1, ROUGE-2, and ROUGE-L as evaluation metrics, invoking the evaluation package provided by HuggingFace. Here, we computed the results for the generated output of each segment, rather than solely focusing on the output of the final segment. We utilized two Quadro RTX 8000 GPUs, each with 48GiB memory, alongside one Quadro P6000 GPU with 24GiB memory.

6 Results and Discussion

Our experiments primarily aimed to validate the impact of different sizes of memory prefix in contrast to baseline results. As shown in Table 2, under the settings of incremental summarisation, the BART model performs strongly even without any additional mechanism in the PubMed-Incremental dataset. This is to some extent reliant on BART's powerful language modeling capabilities. On the other hand, scientific papers often have substantial overlaps between various sections, which we believe might contribute to the model's sustained strong performance. Furthermore, we observed that an increase in the number of memory tokens exhibits a certain positive correlation with the ROUGE scores. The performance reaches its peak at around 200 tokens, with the ROUGE-1 score reaching 50.59, surpassing the baseline by 1.24 points. Subsequently, after further increasing the token count to 256, the rouge scores remained almost unchanged. However, the number of memory tokens around 200 appears to be excessive for a practical situation, suggesting room for improvement of our approach.

#Tokens for memory	Rouge1	Rouge2	RougeL
Baseline BART	49.35	19.38	48.05
20	48.94	18.76	47.66
32	49.44	19.16	48.14
64	49.14	18.79	47.85
100	49.83	19.66	48.52
128	50.04	19.75	48.72
150	50.15	19.91	48.85
200	**50.59**	**20.24**	**49.31**
256	50.25	20.03	48.92

Table 2 ROUGE scores on PubMed-IncreSumm. The number of tokens represents the length of the memory prefix for RMT–Summ. We denote the optimal results with bold font and the second-best results with an underline.

7 Conclusions

For long document summarisation, we proposed a variant of plug-and-play RMT termed RMT-Summ, which incorporates memory prefixes into the encoder-decoder architecture. To verify the effectiveness of the RMT-Summ, we introduced a new optimisation objective, incremental summarisation, and created a dedicated dataset from PubMed articles, i.e. PubMed-IncreSumm datasets. The experimental results show that when the token length of the memory prefix is around 200, the metric of ROUGE scores outperforms the baseline. Since carefully designing a task and training dataset could enable models to generate memory-encoded vectors that are useful for long-text processing, we will continue exploring our approach by evaluating it on diverse models and tasks.

Acknowledgement

This work was supported by JST CREST (Grant Number: JPMJCR22N1), Japan.

References

[1] Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent Memory Transformer. **Advances in Neural Information Processing Systems**, Vol. 35, pp. 11079–11091, 2022.

[2] Aydar Bulatov, Yuri Kuratov, and Mikhail S Burtsev. Scaling transformer to 1m tokens and beyond with RMT. **arXiv preprint arXiv:2304.11062**, 2023.

[3] Zican Dong, Tianyi Tang, Lunyi Li, and Wayne Xin Zhao. A survey on long text modeling with transformers. **arXiv preprint arXiv:2302.14502**, 2023.

[4] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. **arXiv preprint arXiv:2004.05150**, 2020.

[5] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. **Advances in neural information processing systems**, Vol. 33, pp. 17283–17297, 2020.

[6] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In **International Conference on Machine Learning**, pp. 11328–11339. PMLR, 2020.

[7] Mandy Guo, Joshua Ainslie, David C Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. Longt5: Efficient text-to-text transformer for long sequences. In **Findings of the Association for Computational Linguistics: NAACL 2022**, pp. 724–736, 2022.

[8] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)**, 2018.

[9] Alexios Gidiotis and Grigorios Tsoumakas. A divide-and-conquer approach to the summarization of long documents. **IEEE/ACM Transactions on Audio, Speech, and Language Processing**, Vol. 28, pp. 3029–3040, 2020.

[10] Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed Awadallah, Dragomir Radev, and Rui Zhang. Summn: A multi-stage summarization framework for long input dialogues and documents: A multi-stage summarization framework for long input dialogues and documents. In **Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 1592–1604, 2022.

[11] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**, pp. 2978–2988, 2019.

[12] Huan Yee Koh, Jiaxin Ju, Ming Liu, and Shirui Pan. An empirical survey on long document summarization: Datasets, models, and metrics. **ACM computing surveys**, Vol. 55, No. 8, pp. 1–35, 2022.

[13] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In **Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)**, pp. 615–621, 2018.

[14] Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In **Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**, pp. 1419–1436, 2021.

[15] Steven Bird, Ewan Klein, and Edward Loper. **Natural language processing with Python: analyzing text with the natural language toolkit**. " O'Reilly Media, Inc.", 2009.

[16] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 7871–7880, 2020.