Deep Learning Final Project Report

Title: Beyond Attention: Breaking the Limits of Transformer Context Length with Recurrent Memory

Authors:

NCCU Computer Science Senior Students

- 鄭睿宏 (110703007)
- 潘煜智 (110703013)

Abstract:

This report explores the Recurrent Memory Transformer (RMT) as a solution to overcome the context length limitations of traditional transformers. By integrating token-based memory augmentation, RMT extends context length from 128k tokens to over 2 million tokens, enabling efficient handling of long-term dependencies. The proposed approach maintains computational efficiency, enhances scalability, and supports seamless integration with pre-trained models like BERT and GPT-2.

Introduction:

Transformers have become a cornerstone of modern natural language processing tasks. However, their quadratic computational complexity poses significant challenges when processing long sequences. Traditional solutions, such as memory-augmented neural networks and modified attention mechanisms, often require complex architectural changes or remain constrained by hardware limitations. This project investigates RMT, which introduces recurrent memory mechanisms to address these bottlenecks without sacrificing performance or scalability.

Background:

Transformers like BERT and GPT have revolutionized NLP tasks due to their ability to model complex dependencies in data. However, the self-attention mechanism becomes computationally expensive for long sequences. Memory-augmented neural networks and alternatives like Transformer-XL have partially addressed this issue but face limitations in scalability and integration.

Problem Statement:

The limitations of traditional transformers in handling long sequences effectively has prompted research into alternative architectures. Key questions include:

1. Can RMT enable efficient handling of extended context lengths?
2. How does RMT applied to pre-trained models without significant modifications?
3. How does RMT compare to Retrieval Augmented Generation (RAG) frameworks for long-context tasks?

Significance:

The increasing demand for models that handle long-term dependencies, such as in legal document analysis, bioinformatics, and large-scale language understanding, highlights the necessity of scalable solutions. Addressing these demands efficiently could transform industries reliant on long-context data processing.

The RMT solution delivers significant advancements:

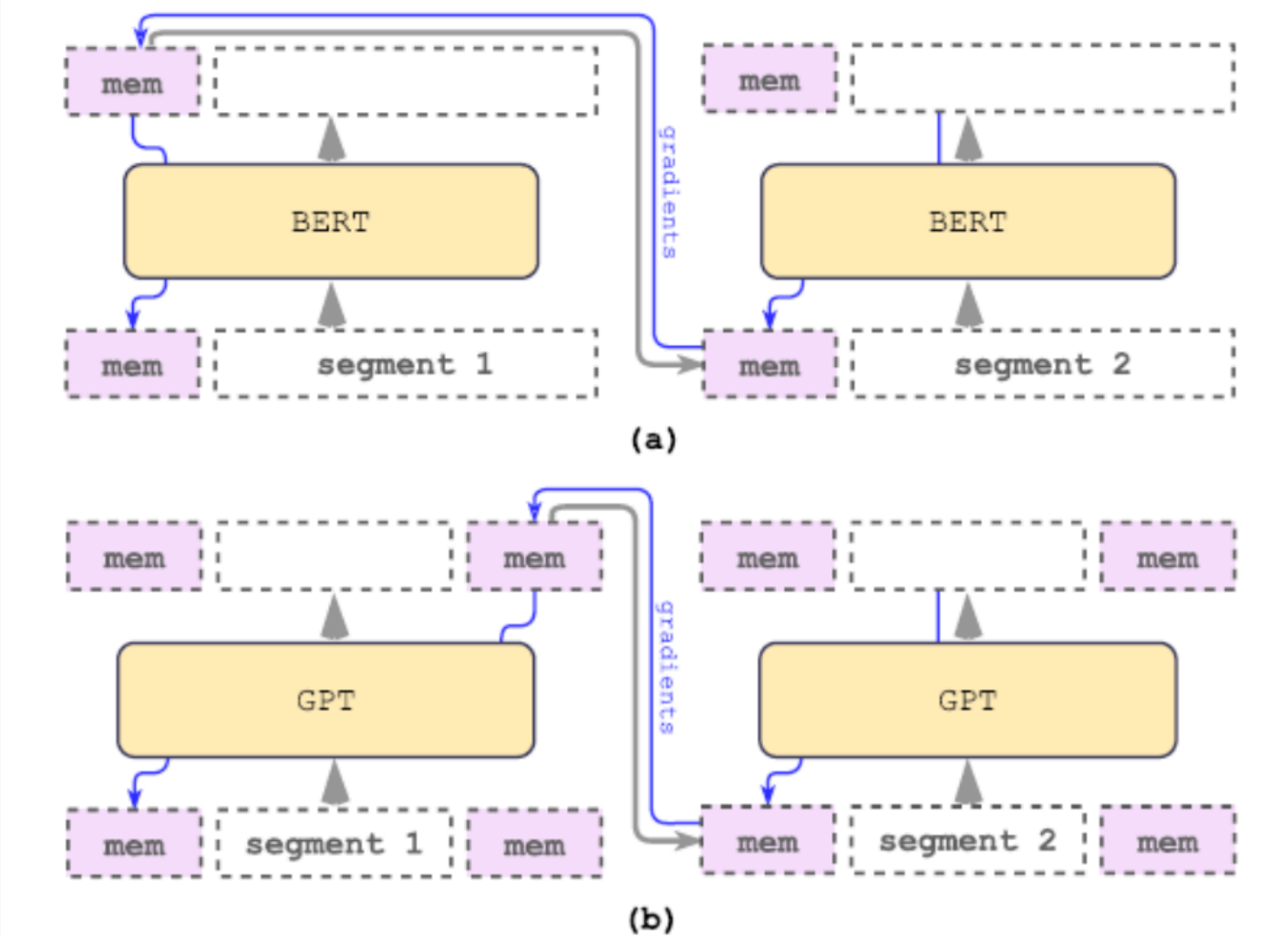
- Enpands sequences handling from 128k tokens to 2 million tokens.
- Supports long-term dependency tasks in natural language understanding and generation.
- Improves scalability for memory-intensive applications

Methodology:

To address the challenges of scaling transformers for long-context tasks, the paper outline a comprehensive methodology that integrates novel architectural innovations with practical training techniques, ensuring both effectiveness and ease of adoption.

Recurrent Memory Transformer (RMT):

The Recurrent Memory Transformer introduces a mechanism for handling extended sequences efficiently by leveraging memory tokens and segmented processing, making it a robust solution for overcoming the input length limitations of traditional transformers.



- **Memory Tokens**:
  - Prepend memory tokens to input/output sequences.
  - Memory tokens act as a bridge between segmented sequences, enabling the model to retain and reuse contextual information across large input spans, thereby addressing long-term dependency challenges.
- **Segmented Processing** :
  - Long sequences are segmented, and memory is passed recurrently between segments.
  - By dividing long sequences into manageable segments and recurrently passing memory between them, RMT ensures computational feasibility while preserving information continuity.
- **Plug-and-Play Integration** :
  - RMT is implemented as a wrapper for pre-trained transformers, allowing seamless adoption.
  - Designed as a lightweight wrapper, RMT seamlessly integrates with pre-trained transformers, facilitating its adoption without requiring extensive architectural modifications.

Curriculum Learning:

- Begin with short sequences during fine-tuning.

- Gradually increase input lengths to stabilize training.

Discussion:

RMT's linear computational scaling and memory token augmentation present significant advancements over existing methods. Unlike prior approaches, RMT avoids architectural complexities and hardware limitations, making it an attractive solution for tasks requiring extended contexts. Furthermore, its compatibility with closed-source language models broadens its applicability.

Limitations:

- Memory token storage may introduce challenges for extremely large datasets.
- Requires extensive fine-tuning to achieve optimal performance.

Conclusion:

The Recurrent Memory Transformer (RMT) addresses the key limitations of traditional transformers by significantly extending their context length capabilities while maintaining efficiency and scalability. By leveraging memory tokens and recurrent mechanisms, RMT simplifies integration into pre-trained models, offering a practical solution for real-world applications.

Future Work:

- Expanding RMT's application to multi-modal data.
- Exploring hardware-specific optimizations to further enhance efficiency.

Experiments: Exploring the value of long context in the Era of RAG

(Don't Do RAG: When Cache-Augmented Generation is All You Need for Knowledge Tasks)

Introduction:

In the rapidly evolving field of natural language processing, Retrieval-Augmented Generation (RAG) has emerged as a widely adopted paradigm for enhancing language model performance by integrating external knowledge sources.

This experiment explores the potential of long context generation inferencing for question answering holds its own value in the era of RAG. Moreover, optimizing the long context interference with Cache-

Augmented Generation (CAG) by preloading a constrained and manageable set of knowledge resources directly into the model's context, CAG bypasses the need for real-time retrieval, reducing system complexity and mitigating retrieval-related errors.

Through evaluations on datasets such as SQuAD, HotpotQA, and TriviaQA, and using advanced metrics like BERTScore, this study investigates how CAG compares to traditional RAG pipelines. The goal is to assess whether CAG can achieve similar or superior performance while providing significant efficiency gains, particularly in scenarios with a limited knowledge base.

Experiment Design:

- **Datasets:** SQuAD, HotpotQA, and TriviaQA.
- **Baseline Models:** Llama 3.1 (8b-instruct) for language modeling.
- **Retrieval Techniques:** BM25 (sparse) and OpenAI embeddings (dense).
- **Evaluation Metrics:** BERTScore for performance and computational efficiency metrics for acceleration.

Language Model :

- Llama 3.1 8b-instruct

Retrirval Model :

- BM25 (Sparse Retrieval)
- OpenAI (Dense Retrieval)

RAG Framework :

- Llama-index

Experiment Dataset: 21k 30k 50k
(Fixed Random seed)

Source	Size	# Docs	# Tokens	# QA Pairs
HotPotQA	Small	16	21k	1,392
	Medium	32	43k	1,056
	Large	64	85k	1,344
SQuAD	Small	3	21k	500
	Medium	4	32k	500
	Large	7	50k	500

(DON'T DO RAG)

Interference Language Model

Choosing the the datasets document length is because of the long context ability of the Llama 3.1 8b.

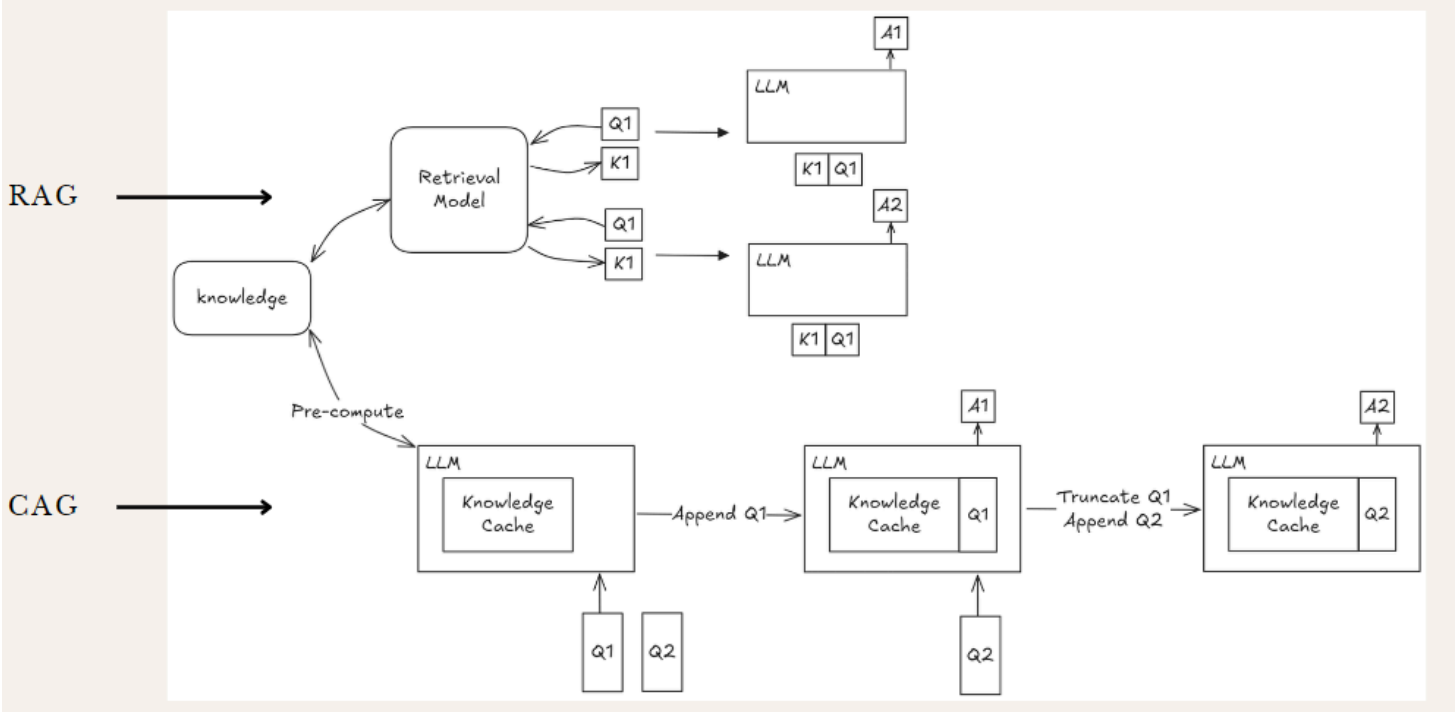
Model	av.	2k	4k	8k	16k	32k	64k	96k	125k
o1-preview-2024-09-12	0.763	0.582	0.747	0.772	0.787	0.799	0.831	0.824	0.763
o1-mini-2024-09-12	0.731	0.566	0.728	0.754	0.772	0.777	0.769	0.778	0.704
gpt-4o-2024-05-13	0.709	0.467	0.671	0.721	0.752	0.759	0.769	0.769	0.767
claude-3-5-sonnet-20240620	0.695	0.506	0.684	0.723	0.718	0.748	0.741	0.732	0.706
claude-3-opus-20240229	0.686	0.463	0.652	0.702	0.716	0.725	0.755	0.732	0.741
claude-3-haiku-20240307	0.649	0.466	0.666	0.678	0.705	0.69	0.668	0.663	0.656
qwen2-72b-instruct	0.637	0.469	0.628	0.669	0.672	0.682	0.683	0.648	0.645
gpt-4o-mini-2024-07-18	0.61	0.424	0.587	0.624	0.649	0.662	0.648	0.646	0.643
gpt-4-turbo-2024-04-09	0.588	0.465	0.6	0.634	0.641	0.623	0.623	0.562	0.56
gemini-1.5-pro	0.584	0.368	0.51	0.55	0.58	0.595	0.634	0.636	0.622
claude-3-sonnet-20240229	0.569	0.432	0.587	0.662	0.668	0.631	0.525	0.559	0.485
gpt-4-0125-preview	0.568	0.466	0.614	0.64	0.664	0.622	0.585	0.505	0.452
llama-3.1-405b-instruct	0.55	0.445	0.591	0.615	0.623	0.594	0.587	0.516	0.426
gemini-1.5-flash	0.505	0.349	0.478	0.517	0.538	0.534	0.522	0.52	0.521
llama-3-70b-instruct	0.48	0.365	0.53	0.546	0.555	0.562	0.573	0.583	0.593
mixtral-8x7b-instruct	0.469	0.414	0.518	0.506	0.488	0.417	-	-	-
llama-3.1-70b-instruct	0.45	0.403	0.526	0.527	0.478	0.469	0.444	0.401	0.353
dbrx-instruct	0.447	0.438	0.539	0.528	0.477	0.255	-	-	-
gpt-3.5-turbo	0.44	0.362	0.463	0.486	0.447	-	-	-	-
llama-3.1-8b-instruct	0.411	0.368	0.547	0.536	0.523	0.485	0.383	0.296	0.15

Table S3: LLM answer correctness up to 125k tokens. Same data as Fig. 1.

LONG CONTEXT RAG PERFORMANCE OF LARGE LANGUAGE MODELS

ref: https://arxiv.org/abs/2411.03538v1

Structure:



Results:

1. **Performance:**
    - CAG demonstrated superior handling of long contexts compared to standard transformers.
    - Efficient integration with existing pre-trained models enabled faster training and inference.

METRIC: BERT SCORE

Table 2: Experimental Results

Size	System	Top-k	HotPotQA BERT-Score	SQuAD BERT-Score
Small	Sparse RAG	1	0.0673	0.7469
		3	0.0673	0.7999
		5	0.7549	0.8022
		10	0.7461	0.8191
	Dense RAG	1	0.7079	0.6445
		3	0.7509	0.7304
		5	0.7414	0.7583
		10	0.7516	0.8035
	CAG (Ours)		**0.7759**	**0.8265**
Medium	Sparse RAG	1	0.6652	0.7036
		3	0.7619	0.7471
		5	0.7616	0.7467
		10	0.7238	0.7420
	Dense RAG	1	0.7135	0.6188
		3	0.7464	0.6869
		5	0.7278	0.7047
		10	0.7451	0.7350
	CAG (Ours)		**0.7696**	**0.7512**
Large	Sparse RAG	1	0.6567	0.7135
		3	0.7424	0.7510
		5	0.7495	0.7543
		10	0.7358	0.7548
	Dense RAG	1	0.6969	0.6057
		3	0.7426	0.6908
		5	0.7300	0.7169
		10	0.7398	0.7499
	CAG (Ours)		**0.7527**	**0.7640**

(DON'T DO RAG)

2. **Acceleration:**
    - Cache-Augmented Generation (CAG) further reduced computational overhead, leveraging kvcache for knowledge storage and retrieval.

Table 3: Comparison of Generation Time

Dataset	Size	System	Generation Time (s)
HotpotQA	Small	CAG	0.85292
		w/o CAG	9.24734
	Medium	CAG	1.66132
		w/o CAG	28.81642
	Large	CAG	2.32667
		w/o CAG	94.34917
SQuAD	Small	CAG	1.06509
		w/o CAG	10.29533
	Medium	CAG	1.73114
		w/o CAG	13.35784
	Large	CAG	2.40577
		w/o CAG	31.08368

(DON'T DO RAG)

3. **Comparison to RAG:**
    - CAG outperformed RAG in terms of computational efficiency and ease of implementation.

Source Code:

Github: https://github.com/hhhuang/CAG

Thanks:

We thank our advisor and team members for their support and guidance throughout this project. Special thanks to the NCCU Computer Science department for providing computational resources.

References:

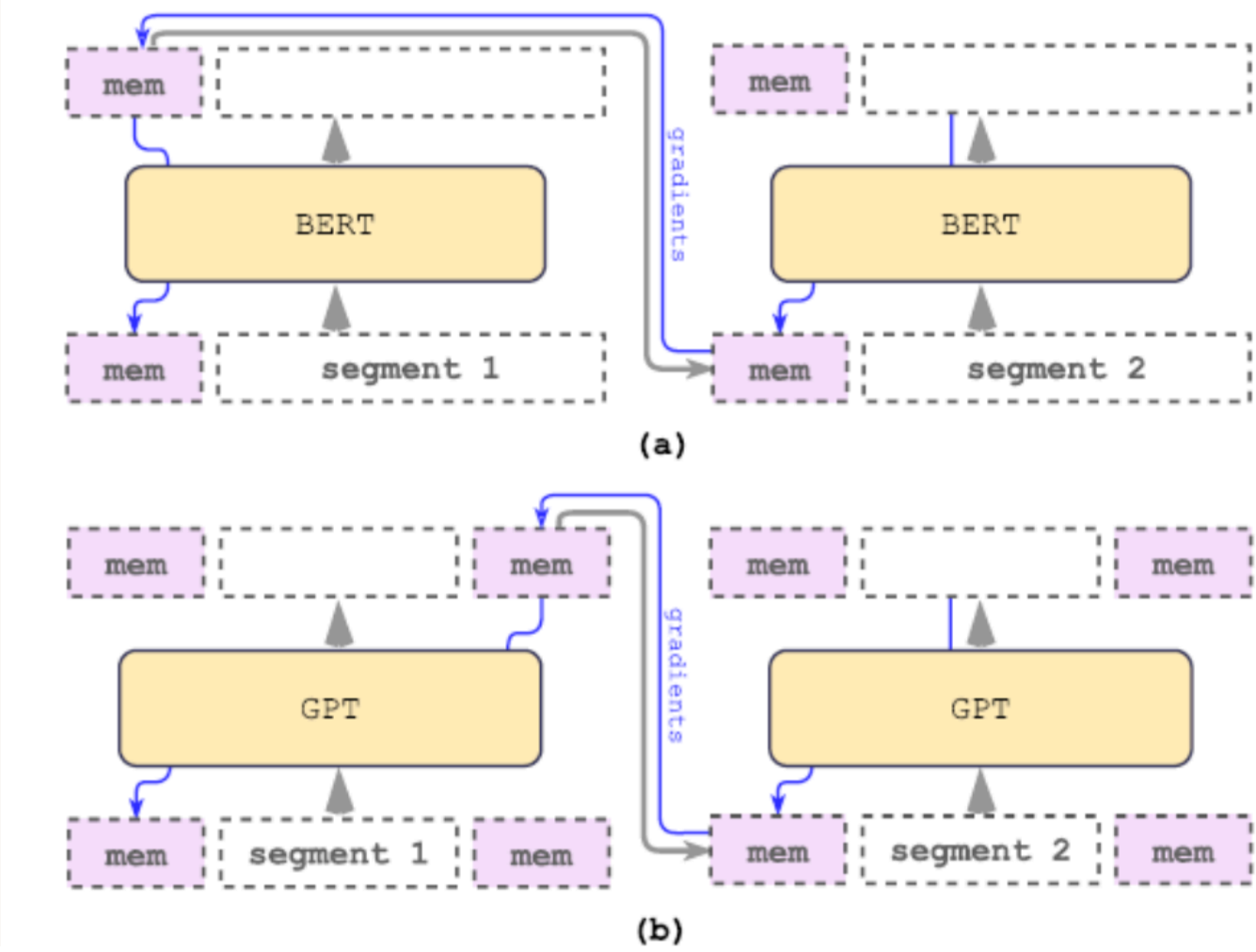
1. Bulatov, A., Kuratov, Y., Kapushev, Y., & Burtsev, M. (2024). *Beyond Attention: Breaking the Limits of Transformer Context Length with Recurrent Memory.* Proceedings of the AAAI Conference on

Artificial Intelligence, 38(16), 17700-17708.
2. Brian J Chan, Chao-Ting Chen, Jui-Hung Cheng, & Hen-Hsen Huang. (2024). *Don't Do RAG: When Cache-Augmented Generation is All You Need for Knowledge Tasks.* arXiv preprint, arXiv:2412.15605v1 [cs.CL] .

Appendix:

Figures:

1. **RMT Architecture Diagram**



2. **Performance Graphs**
    - Graph comparing model performance across datasets.
    - Chart showcasing memory usage efficiency.

Tables:

- Table summarizing datasets and model configurations.
- Results table for BERTScore and efficiency metrics.

Note: Placeholder for images has been added. Please insert appropriate diagrams and graphs to further enhance the visual appeal and readability of the report.