腾讯AI平台部
Tencent AI
Platform Dept.

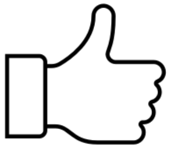
Block-Attention for Efficient Prefilling

Dongyang Ma, Yan Wang, Tian Lan

Tencent AI Platform Dept

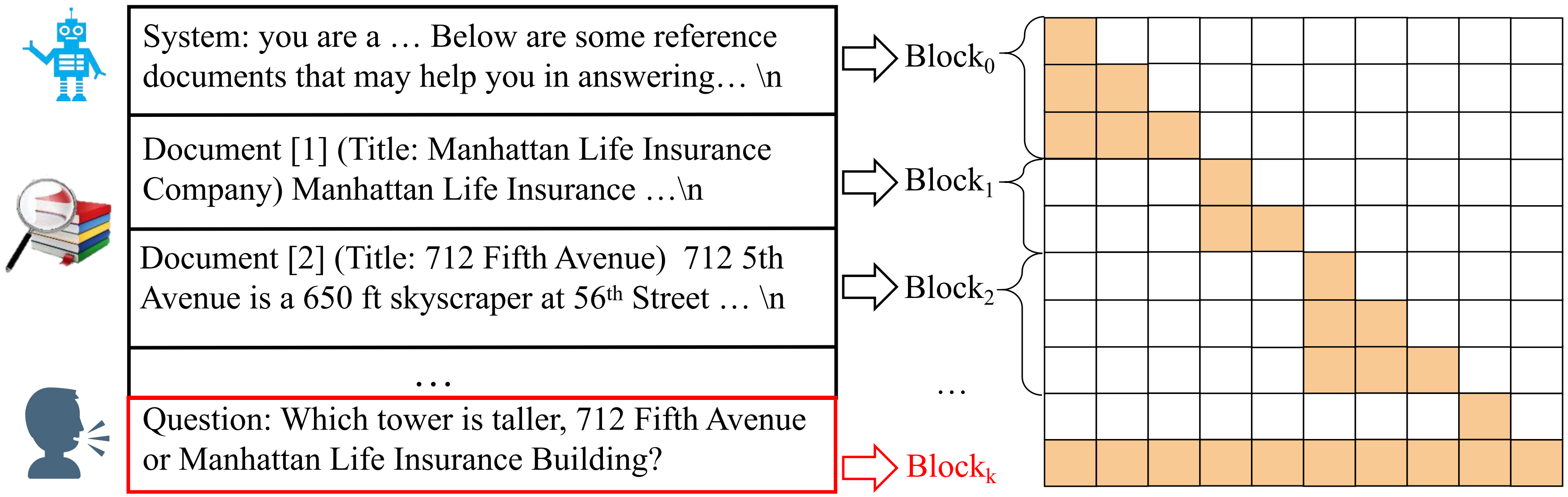
The Dilemma of RAG

RAG



Efficiency?

You need Block-Attention!



Core Idea:

- Divide input into **Independent Blocks**
- **Parallel Encoding**. Only the final block attends to full context
- Block **KV Cache Reuse**!

Full Attention $\Rightarrow$ Block Attention

Challenges?

Performance drop from **66.1% to 42.5%**, because of

- Incorrect positional encoding
- Never seen block-attention before

Position Re-encoding: position $i \Rightarrow$ target position i_Δ

Encoding: $$f(x_i, i) = R(i\theta) \cdot p_i$$

Reset: $$f(x_i, 0) = R(-i\theta) \cdot f(x_i, i)$$

Re-encoding: $$f(x_{i_\Delta}, i_\Delta) = R(i_\Delta\theta) \cdot f(x_i, 0)$$

First set the positional encoding **to 0**, and then rotate it **to the target position i_Δ!**

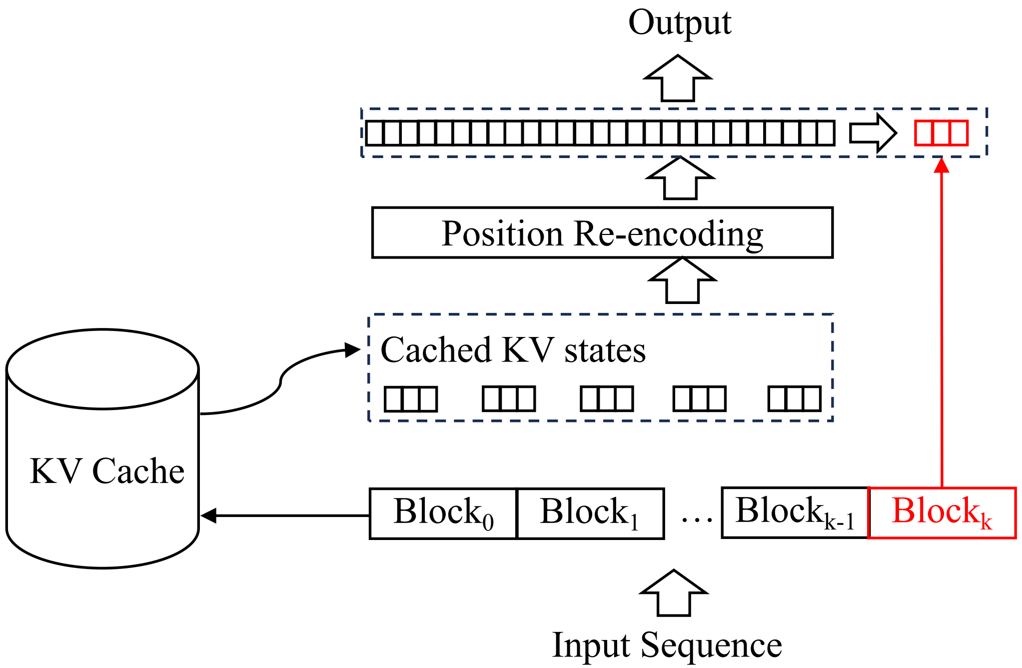
Full Attention ⇨ Block Attention

Block Fine-Tune:

- Training with Block and full-attention data simultaneously

Inference with Block Attention:

Retrieval Cache ⇨ Position Re-encoding ⇨ Computation



Experiments: RAG

Train Set: Tulu3-SFT,TriviaQA and 2Wiki
Test Set:
- **RAG:** 2Wiki, HQA, NQ, TQA
- **General:** IFEval, HumanEval, MMLU - **ICL**: GSM8K, MATH, BBH, DROP

Models	2wiki	HQA	NQ	TQA
Tulu3-SFT	62.0	68.4	58.6	75.7
Tulu3-RAG	**73.2**	**74.8**	**61.5**	**75.8**
Tulu3-RAG-Superposition	30.1	32.3	35.9	58.9
Tulu3-RAG-promptCache	32.4	31.6	44.4	61.8
Tulu3-block-ft	**72.2**	72.3	**60.4**	**75.1**
Tulu3-block-ft-full	**73.6**	**75.2**	**62.2**	**76.2**
Tulu3-block-ft-w/o-pos	68.9	69.9	59.2	74.4
Tulu3-block-w/o-ft	42.9	42.1	48.3	66.5

RAG : Block-Attention (Tulu3-block-ft) is **comparable** to full-attention (Tulu3-SFT and Tulu3-RAG)

Experiments: General and ICL

Task Type	General			ICL			
dataset	IFEval	HumanEval	MMLU	GMS8K	MATH	BBH	DROP
setup	0-shot	0-shot	0-shot	4-shot	4-shot	3-shot	3-shot
Tulu3-SFT	68.5	58.5	**63.7**	75.5	**29.2**	**68.5**	9.4
Tulu3-RAG	68.3	**65.2**	63.6	75.6	28.6	**68.5**	10.4
Tulu3-block-ft	**70.0**	59.1	63.0	**75.7**	28.8	65.3	**14.4**

General and ICL : Block-Attention performs comparably or **slightly better** than full-attention models (Tulu3-SFT)

Seamlessly switches between block and full attention, **without any performance loss!**

Efficiency

Prompt Length	50	512	1K	2K	4K	8K	16K	32K
TTFT-vanilla	26	50	87	167	330	691	1515	3638
TTFT-block	26	26(48%)	26(71%)	26(84%)	27(91%)	29(95%)	34(97%)	45(98.7%)
FLOPs-TFT-vanilla	7.5e+11	7.6e+12	1.5e+13	3.0e+13	6.1e+13	1.2e+14	2.45e+14	4.9e+14
FLOPs-TFT-block	7.5e+11	7.5e+11	7.5e+11	7.5e+11	7.5e+11	7.5e+11	7.5e+11	7.5e+11
Reduction	-	90.1%	95.0%	97.5%	98.7%	99.3%	99.6%	99.8%

Within a 32K-token input and 50-token question (last block), Block-Attention:

- Reduces TTFT by **98.7%** compared to full-attention models
- Corresponding FLOPs-TFT reduced by **99.8%**
- **Only Need** 800 further fine-tuning steps

Conclusion and Highlights

Generalization:

- Block-Attention is a general efficient prefilling method, not just specific to RAG. (Don't miss its **disruptive impact on real-time game agents**. Please pay attention to our **Appendix A**.)

Performance:

- Through some real-world practice, we are convinced that block fine-tuning can easily ensure that there is no performance loss even when there are **hundreds of blocks**.

Flexibility :

- Easily adapt to any scenarios through the **seamless switching** between block and full attention

腾讯AI平台部
Tencent AI
Platform Dept.

THANKS

Job or Intern Opportunity? Contact us!
(yanwang.branden@gmail.com)