

AWQ: Activation-aware Weight Quantization (2023)

Presenters: Zihao Wang, Dylan Harden
Date: 11/16/2024

Why On-Device LLMs?

- Reduce cloud computing costs
- Enhance user privacy by processing locally
- Meet increasing demand for deploying LLMs on resource-limited devices

However:

- Astronomical model size requires significant storage and memory
 - (GPT-3 has 175B parameters, which is 350GB in FP16)
- Limited hardware resources on edge devices
- High computational complexity impacting real-time inference

Introduction to Quantization

What is Quantization?

- Quantization reduces the model size and computational complexity.
- Key for deploying deep learning models on resource-constrained devices.

Challenges of Quantization

- Precision degradation due to weight and activation rounding.
- Ignoring the interaction between weights and activations leads to higher inference errors.

Motivation for AWQ

Existing Limitations

- Current quantization methods independently quantify weights and activations.
- Lacks attention to activation distribution changes during inference.

Why Activation-aware Quantization?

- Activations have a direct impact on model output.
- Understanding activation behavior can minimize quantization errors.

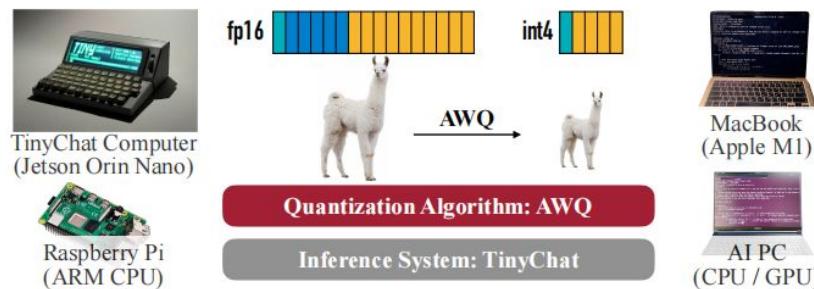
AWQ: Key Idea

Activation-aware Weight Quantization (AWQ)

- A method to integrate activation statistics into weight quantization.
- Optimizes quantization based on activation value distribution.

Main Concept

- Jointly model weights and activations to reduce overall error.
- Makes quantized models behave more like full-precision versions.



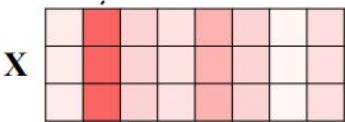
AWQ: Key Idea

\mathbf{W}_{FP16}
+1.2 -0.2 -2.4 -3.4
-2.5 -3.5 +1.9 +1.4
-0.9 +1.6 -2.5 -1.9
-3.5 +1.5 +0.5 -0.1
+1.8 -1.6 -3.2 -3.4
+2.4 -3.5 -2.8 -3.9
+0.1 -3.8 +2.4 +3.4
+0.9 +3.3 -1.9 -2.3

\xrightarrow{Q}

$\mathbf{Q}(\mathbf{W})_{\text{INT3}}$
+1 +0 -2 -3
-3 -4 +2 +1
-1 +2 -3 -2
-4 +2 +1 +0
+2 -2 -3 -3
+2 -4 -3 -4
+0 -4 +2 +3
+1 +3 -2 -2

determine the salient weights by activation



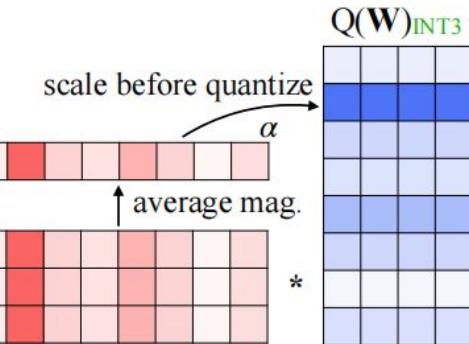
(a) RTN quantization (**PPL 43.2**)

(b) Keep 1% salient weights in FP16 (**PPL 13.0**)

bad hardware efficiency
→ $\mathbf{Q}(\mathbf{W})_{\text{MixPrec}}$

$\mathbf{Q}(\mathbf{W})_{\text{MixPrec}}$
+1 +0 -2 -3
-2.5 -3.5 +1.9 +1.4
-1 +2 -3 -2
-4 +2 +1 +0
+2 -2 -3 -3
+2 -4 -3 -4
+0 -4 +2 +3
+1 +3 -2 -2

FP16 channel



(c) Scale the weights before quantization (**PPL 13.0**)

Figure 2. We observe that we can find 1% of the salient weights in LLMs based on the *activation distribution* (middle). Keeping the salient weights in FP16 can significantly improve the quantized performance (PPL from 43.2 (left) to 13.0 (middle)), but the mixed-precision format is not hardware-efficient. We follow the activation-awareness principle and propose AWQ (right). AWQ performs per-channel scaling to protect the salient weights and reduce quantization error. We measure the perplexity of OPT-6.7B under INT3-g128 quantization.

Why model the quantization based on Activations?

PPL ↓	FP16	RTN (w3-g128)	FP16% (based on act.)			FP16% (based on W)			FP16% (random)		
			0.1%	1%	3%	0.1%	1%	3%	0.1%	1%	3%
OPT-1.3B	14.62	119.00	25.03	16.91	16.68	108.71	98.55	98.08	119.76	109.38	61.49
OPT-6.7B	10.86	23.54	11.58	11.39	11.36	23.41	22.37	22.45	23.54	24.23	24.22
OPT-13B	10.13	46.04	10.51	10.43	10.42	46.07	48.96	54.49	44.87	42.00	39.71

AWQ Methodology

Activation Statistics Collection

- Run the unquantized model to gather activation statistics (mean, variance).

Quantization Strategy

- Use activation information to determine optimal quantization intervals.
- Quantize weights while considering the impact on activation distributions.

Optimization Goal

- Minimize differences between activations in quantized vs. original models.

Advantages of AWQ

Reduced Quantization Error

- By accounting for activation behavior, AWQ reduces the propagation of quantization errors.

Higher Accuracy

- Improved performance, especially in deep networks where activations are more complex.

Broader Applicability

- Effective for both convolutional neural networks (CNNs) and transformer-based models.

Results

Quantization Method Performance Wishlist

We want a method that:

Minimizes Generation Quality Degradation

- Lower Perplexity, Higher Benchmark Scores etc.

Speeds up inference

- Highest Tokens per Second

Results - generation

PPL↓		Llama-2			LLaMA			
		7B	13B	70B	7B	13B	30B	65B
FP16	-	5.47	4.88	3.32	5.68	5.09	4.10	3.53
INT3	RTN	6.66	5.52	3.98	7.01	5.88	4.88	4.24
	GPTQ	6.43	5.48	3.88	8.81	5.66	4.88	4.17
	GPTQ-R	6.42	5.41	3.86	6.53	5.64	4.74	4.21
	AWQ	6.24	5.32	3.74	6.35	5.52	4.61	3.95
g128	RTN	5.73	4.98	3.46	5.96	5.25	4.23	3.67
	GPTQ	5.69	4.98	3.42	6.22	5.23	4.24	3.66
	GPTQ-R	5.63	4.99	3.43	5.83	5.20	4.22	3.66
	AWQ	5.60	4.97	3.41	5.78	5.19	4.21	3.62

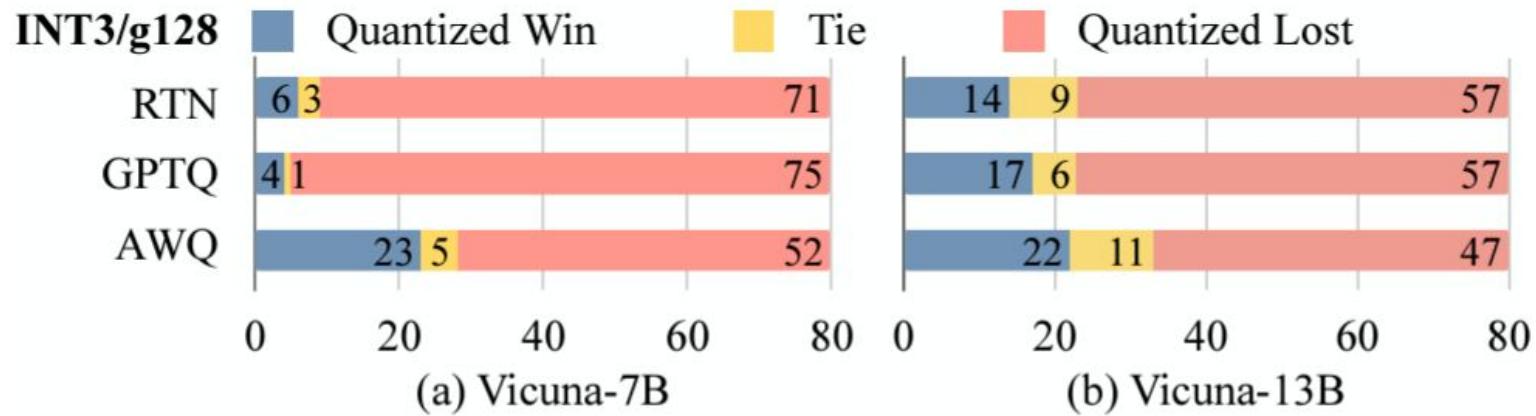
AWQ Outperforms Other Methods Across Model Sizes and Bit Configurations

Results - generation

COCO (CIDEr \uparrow)	0-shot	4-shot	8-shot	16-shot	32-shot	$\Delta(32\text{-}shot)$
 W4-RTN: A model airplane flying in the sky. W4-AWQ: Two toy airplanes sit on a grass field.	 W4-RTN: A man is holding a baby elephant in his arms. W4-AWQ: A man and his daughter pose with an elephant.	 W4-RTN: A man is holding a baby elephant in his arms. W4-AWQ: A man and his daughter pose with an elephant.	 W4-RTN: A man is holding a baby elephant in his arms. W4-AWQ: A man and his daughter pose with an elephant.	 W4-RTN: A man and a dog walking past some bushes. W4-AWQ: Two dogs are walking on the street.	 W4-RTN: A man and a dog walking past some bushes. W4-AWQ: Two dogs are walking on the street.	-10.95 -7.23
g128 UR1Q AWQ	49.04 56.33	50.11 64.73	50.55 68.79	50.54 72.86	54.11 74.47	

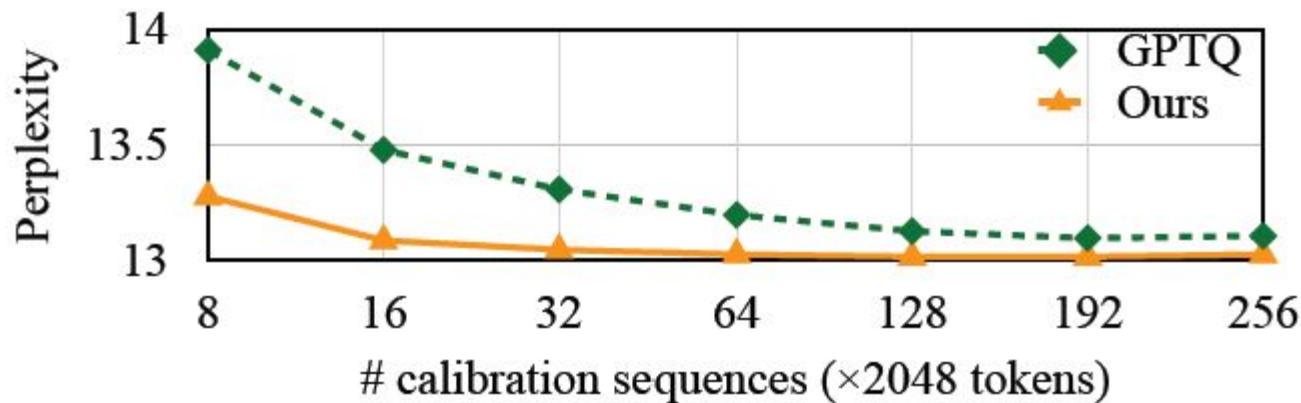
AWQ Outperforms Other Methods on Multi-Modal Benchmarks

Results - generation



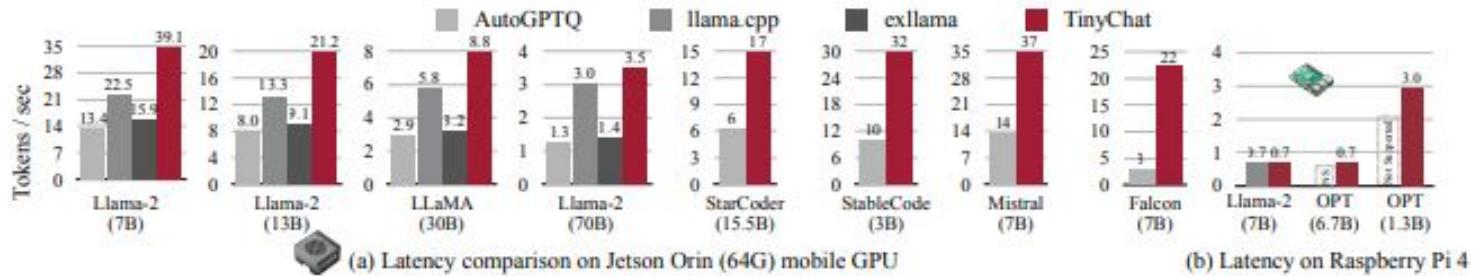
AWQ is more preferred than RTN/GPTQ when compared with FP16 version

Results - generation



AWQ Requires Fewer Calibration Sequences than GPTQ

Results - inference speed



TinyChat outperforms other inference engines 3x

Results - inference speed

Model (Throughput↑)	Precision	A100	4090	Orin
VILA-7B	FP16	81.6	58.5	11.5
VILA-7B-AWQ	W4A16	155.3	168.1	35.6
VILA-13B	FP16	48.5	OOM	6.1
VILA-13B-AWQ	W4A16	102.1	99.0	17.5

AWQ Significantly improves inference speed over FP16 baseline

Strengths

Efficient

- 3x inference speedup with minimal performance degradation

Generality

- Shown to be effective across many domains and in the multi-modal setting

Robust

- Less sensitive to calibration set size and distribution

Weaknesses

Requires Calibration Dataset

- A zero-shot algorithm would be preferred

Requires Customized Inference Engine to realize benefits

- TinyChat

Weaknesses

From llama.cpp repo:

The image shows two separate screenshots of the Hugging Face Model Hub interface. Both screenshots feature a header with the Hugging Face logo, a search bar containing 'Search models, datasets, users...', and navigation links for 'Models', 'Datasets', and 'Tasks'. The top screenshot displays a search result for 'gguf', showing 48,752 models. The bottom screenshot displays a search result for 'awq', showing 4,122 models. Both results include a 'Filter Tasks by name' dropdown menu.

From llama.cpp repo:

Multimodal models

Hugging Face Search models, datasets, users... Models Datasets

Tasks Libraries Datasets Languages Licenses Other Models 48,752 gguf

Filter Tasks by name

Hugging Face Search models, datasets, users... Datasets

Tasks Libraries Datasets Languages Licenses Other Models 4,122 awq

Filter Tasks by name

llama.cpp/GGUF has been more widely adopted



Flash Attention

Group 10

Presenter: Chris Zhang, Yuxuan Zhang

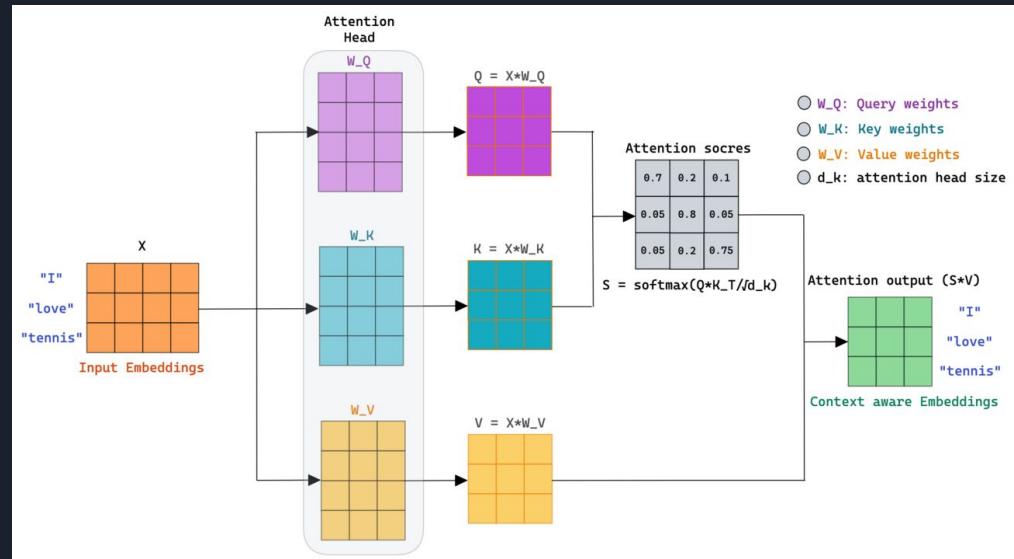


Background

- Transformer & Attention
- Why is attention costly
- Related work
- Their observation - memory IO is a bound
 - Why memory IO is a problem

Attention - Heart of transformer

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



What is cost

Algorithm 0 Standard Attention Implementation

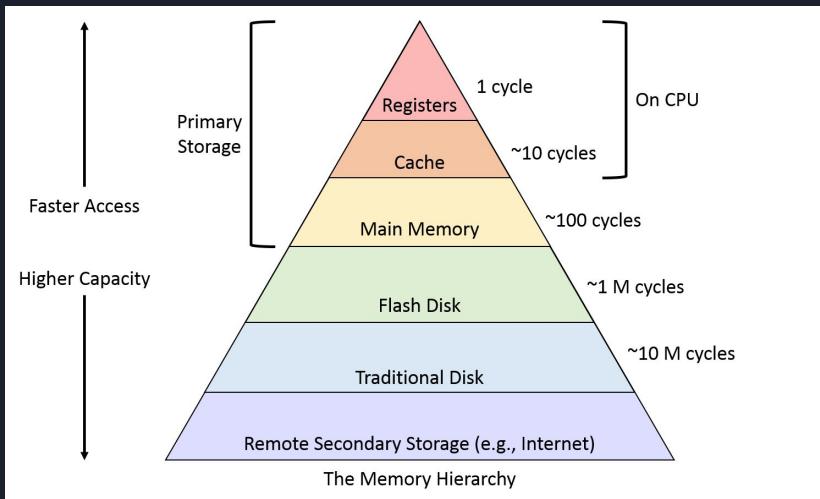
Require: Matrices $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$ in HBM.

- 1: Load \mathbf{Q}, \mathbf{K} by blocks from HBM, compute $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top$, write \mathbf{S} to HBM.
 - 2: Read \mathbf{S} from HBM, compute $\mathbf{P} = \text{softmax}(\mathbf{S})$, write \mathbf{P} to HBM.
 - 3: Load \mathbf{P} and \mathbf{V} by blocks from HBM, compute $\mathbf{O} = \mathbf{PV}$, write \mathbf{O} to HBM.
 - 4: Return \mathbf{O} .
-

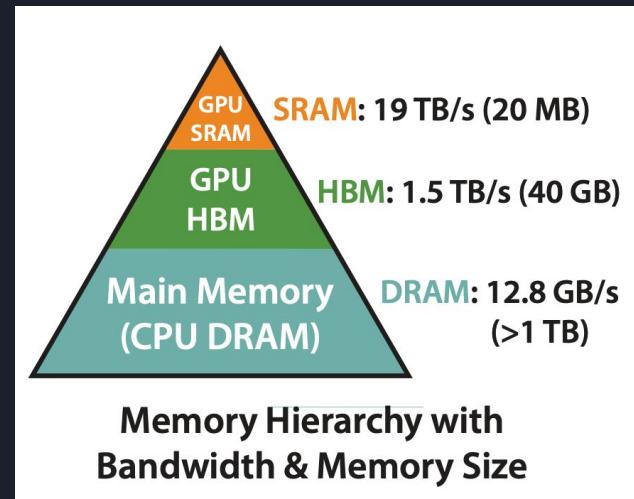
- All intermediate result are write back to HBM (VRAM)
- N (context window) usually very long
 - GPT-2: $N = 1,024$
 - Llama 3.2 - 1B: $N = 4,096$
- Huge HBM read/write

Memory Hierarchy

CPU:

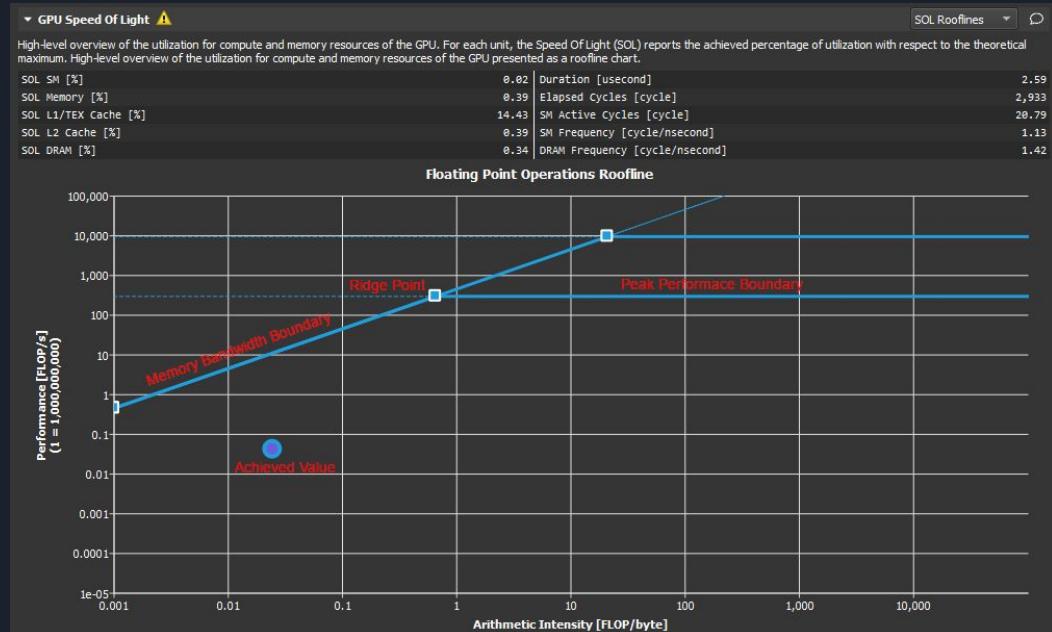


GPU (A100):

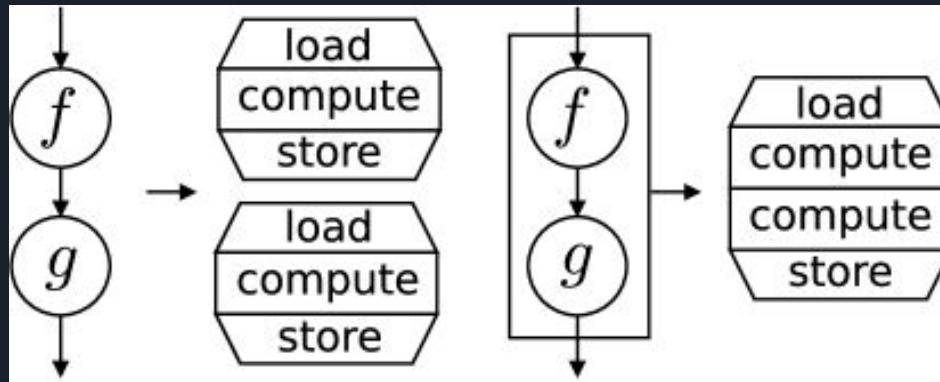


Related works

- Arithmetic intensity analysis per kernel



Related works



Kernel fusion example



Observations - bounds

- Compute-bound: Kernel runtime is highly dependent on **computing speed**
 - Matrix Multiplication with large inner dimension
 - Convolution with large number of channels
- Memory-bound: Kernel runtime is highly dependent on **memory IO speed**
 - Elementwise operations
 - *Activation, dropout, etc*
 - Reduction
 - *Sum, Softmax, Batch norm, Layer norm, etc*



Flash Attention - Basic Idea

Minimize memory usage by increasing computation, which is more efficient and cost-effective



Flash Attention - Challenge

1. Compute softmax reduction without access to the full input
 - a. Global Dependency
 - b. Normalization
 - c. Dimensionality
2. Backward propagation without the large matrix from forward path
 - a. Back propagation relies on intermediate outputs in the matrix
 - b. Recomputing can be costly

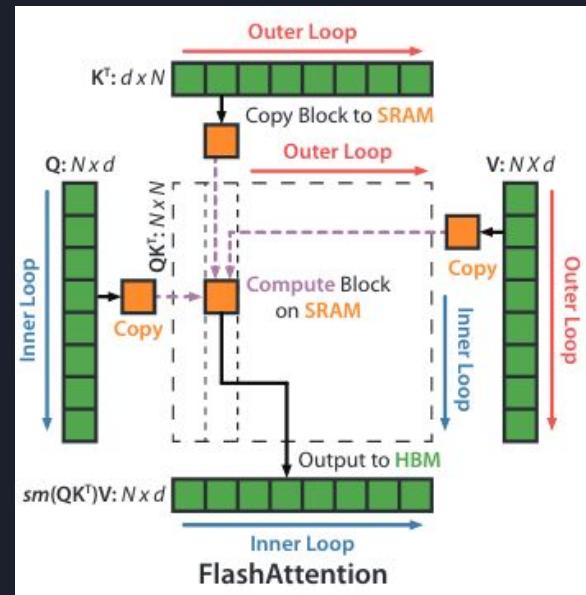
Flash Attention - Tiling

Decompose large softmax into smaller ones by scaling

1. Load inputs by blocks from HBM to SRAM (GPU)
 - a. Divide the original input Q, K, V matrices into blocks (block size is tunable)
2. On chip, compute attention output of that block

$$A_{block} = \text{softmax} \left(\frac{Q_{block} K_{block}^T}{\sqrt{d_k}} \right) V_{block}$$

3. Update output in HBM by scaling
 - a. Each block's contributions are scaled by the sum of exponentials accumulated across all blocks.



Flash Attention - Recomputation

Store softmax normalization factors from forward path, quickly recompute attention in the backward path from inputs in SRAM

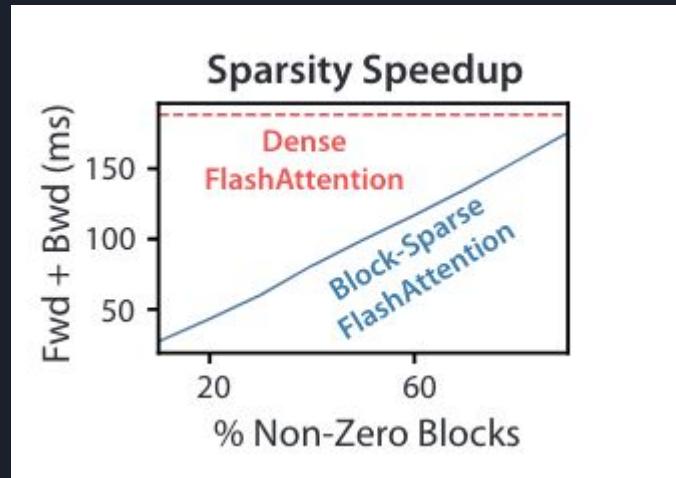
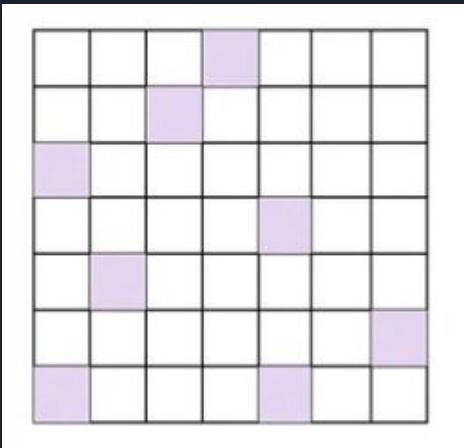
1. Calculate S using the Q and K stored in SRAM
2. Recompute softmax output A by dividing the stored normalization factor
3. Compute gradients

$$A = \text{softmax}(S) = \frac{e^S}{\sum e^S}, \text{ where } S = \frac{QK^T}{\sqrt{d_k}}$$

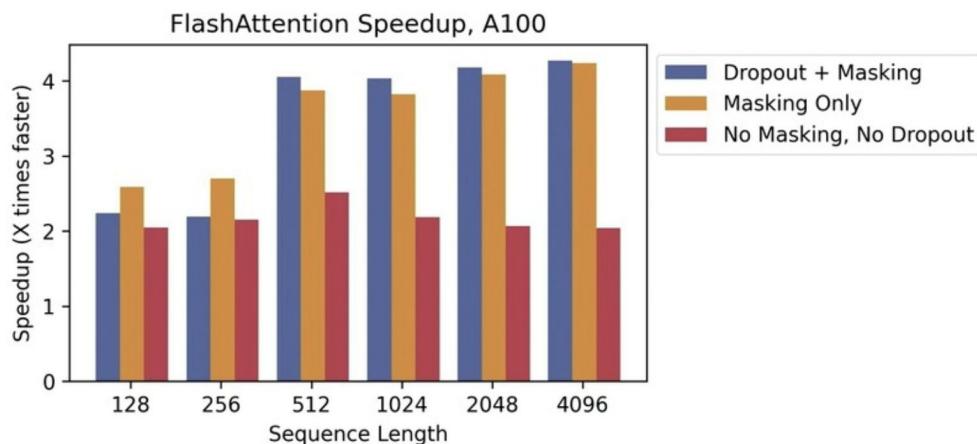
Attention	Standard	FLASHATTENTION
GFLOPs	66.6	75.2
HBM R/W (GB)	40.3	4.4
Runtime (ms)	41.7	7.3

Flash Attention - Sparsity

With a predefined block sparsity mask matrix applied, only compute the nonzero blocks of the attention matrix



Evaluation

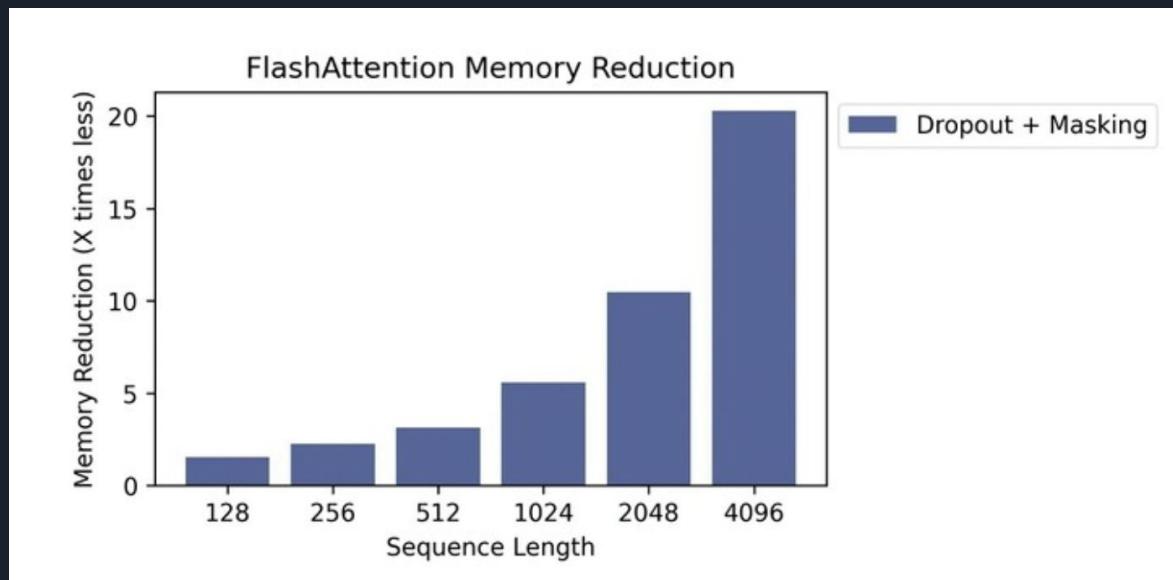


BERT Implementation	Training time (minutes)
Nvidia MLPerf 1.1 [58]	20.0 ± 1.5
FLASHATTENTION (ours)	17.4 ± 1.4

Evaluation

Model implementations	OpenWebText (ppl)	Training time (speedup)
GPT-2 small - Huggingface [87]	18.2	9.5 days (1.0×)
GPT-2 small - Megatron-LM [77]	18.2	4.7 days (2.0×)
GPT-2 small - FLASHATTENTION	18.2	2.7 days (3.5×)
GPT-2 medium - Huggingface [87]	14.2	21.0 days (1.0×)
GPT-2 medium - Megatron-LM [77]	14.3	11.5 days (1.8×)
GPT-2 medium - FLASHATTENTION	14.3	6.9 days (3.0×)

Evaluation



Evaluation

Model implementations	Context length	OpenWebText (ppl)	Training time (speedup)
GPT-2 small - Megatron-LM	1k	18.2	4.7 days (1.0×)
GPT-2 small - FLASHATTENTION	1k	18.2	2.7 days (1.7×)
GPT-2 small - FLASHATTENTION	2k	17.6	3.0 days (1.6×)
GPT-2 small - FLASHATTENTION	4k	17.5	3.6 days (1.3×)



Takeaways

- FlashAttention: fast and memory-efficient algorithm for exact attention
- Key algorithms: tiling, recomputation
- Provide faster model training and better models with longer sequences



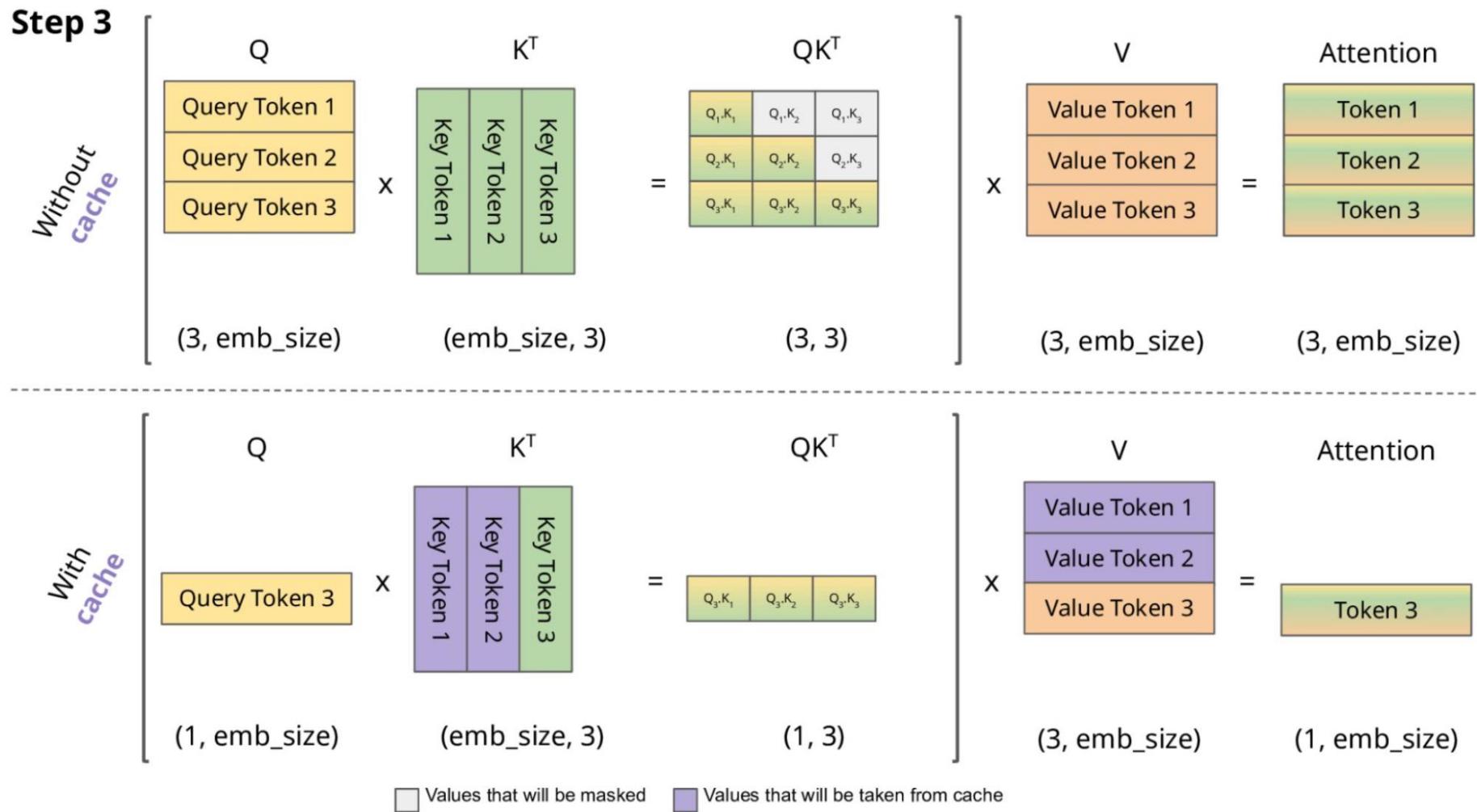
Thanks you!

Group 10

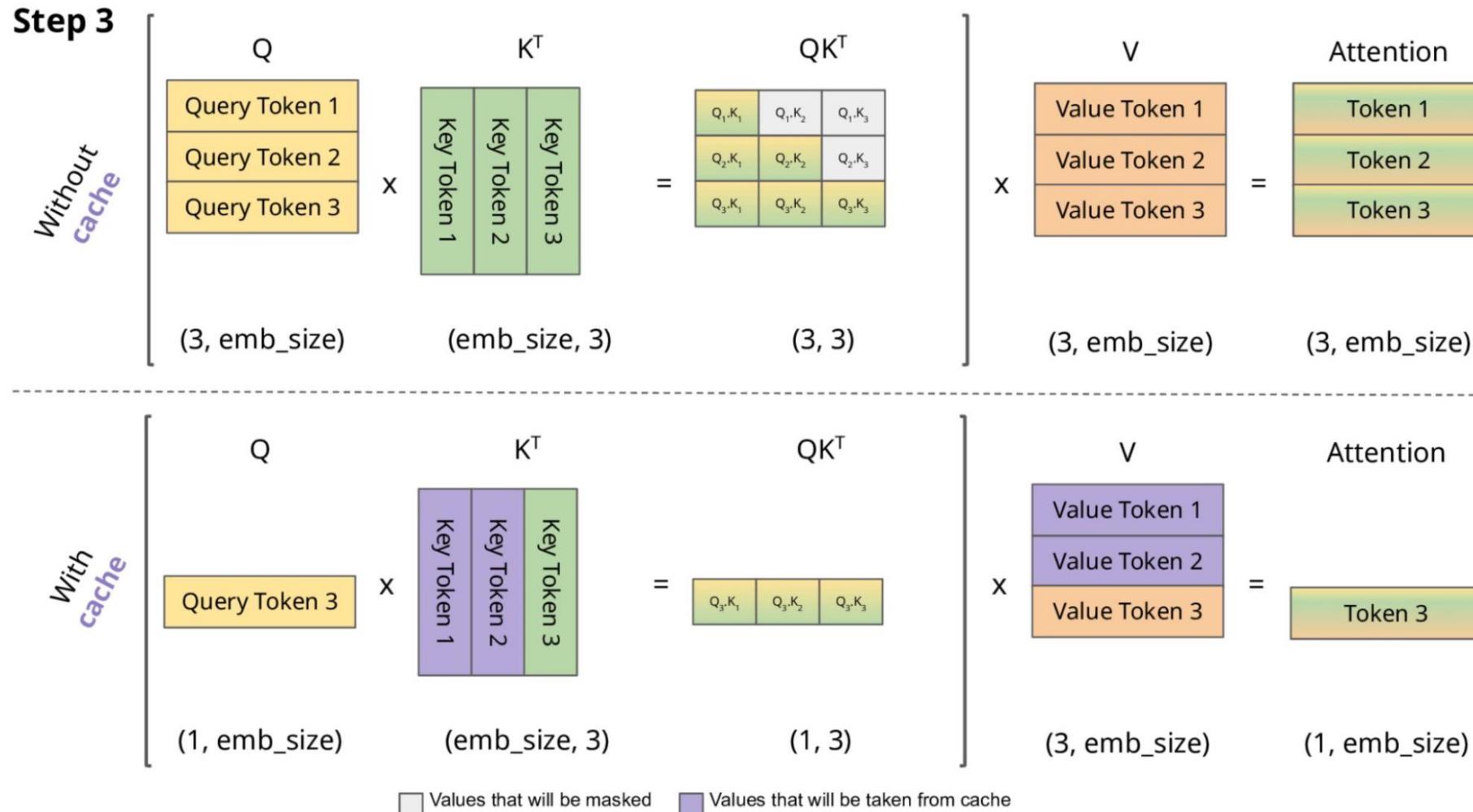
Presenter: Chris Zhang, Yuxuan Zhang

Efficient Memory Management for Large Language Model Serving with PagedAttention

Motivation & Challenge



Motivation & Challenge

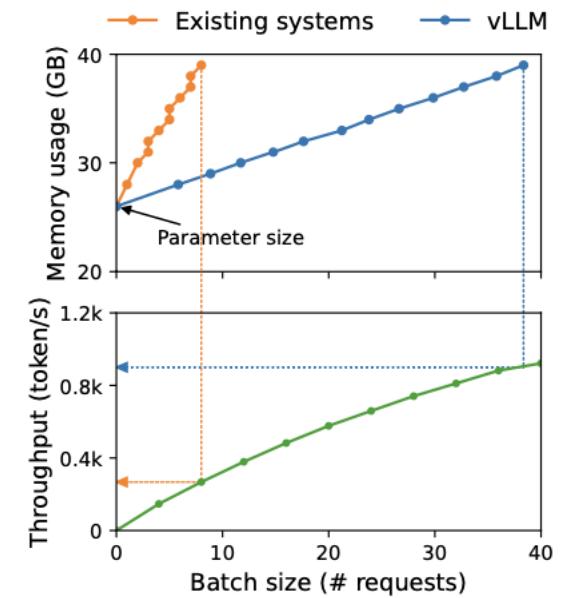
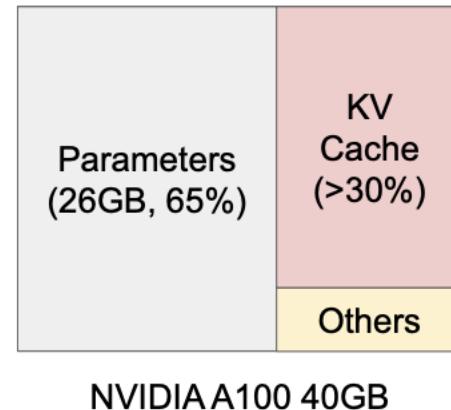


Most deep learning frameworks require tensors to be stored in contiguous memory.

However, unlike the tensors in the traditional deep learning workloads, the KV cache has unique characteristics:
it dynamically grows and shrinks over time as the model generates new tokens, and its lifetime and length are not known a priori.

Background

Obviously, there is a lot of free memory, and a GPU can handle multiple requests at the same time. When multiple requests are processed concurrently, there will be a memory scheduling problem.



The memory distribution for a 13B-parameter LLM on an NVIDIA A100 GPU with 40GB RAM

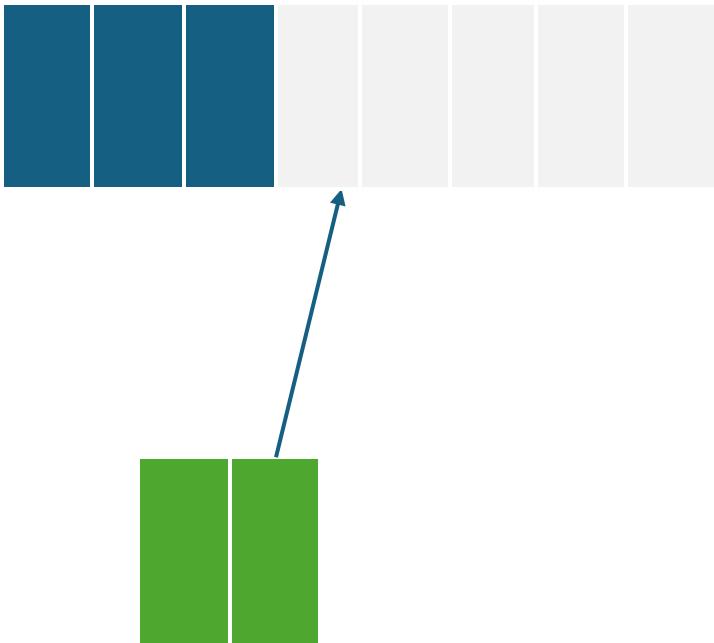
Potential Solutions

GPU Memory

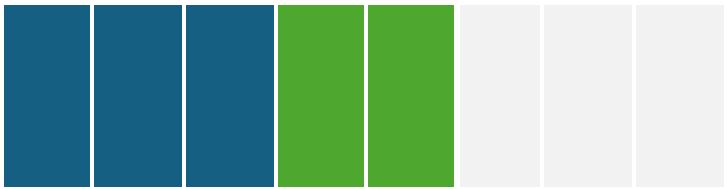


Potential Solutions

Request 2 is coming

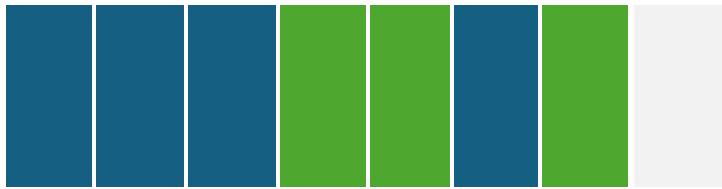


Potential Solutions



Potential Solutions

Continuous Generation



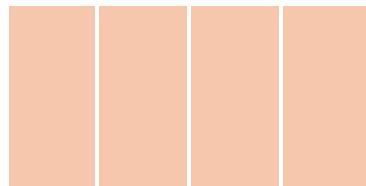
Potential Solutions

Request 1 completed, release memory

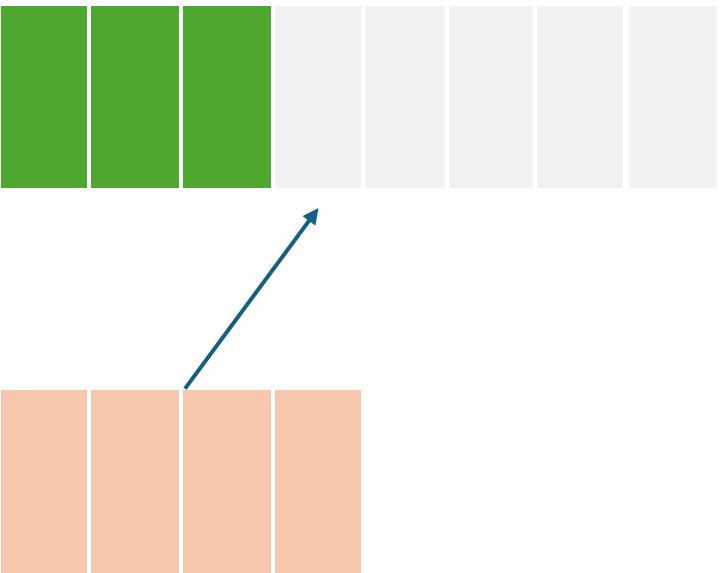


Potential Solutions

Request 3 is coming 😭😭😭

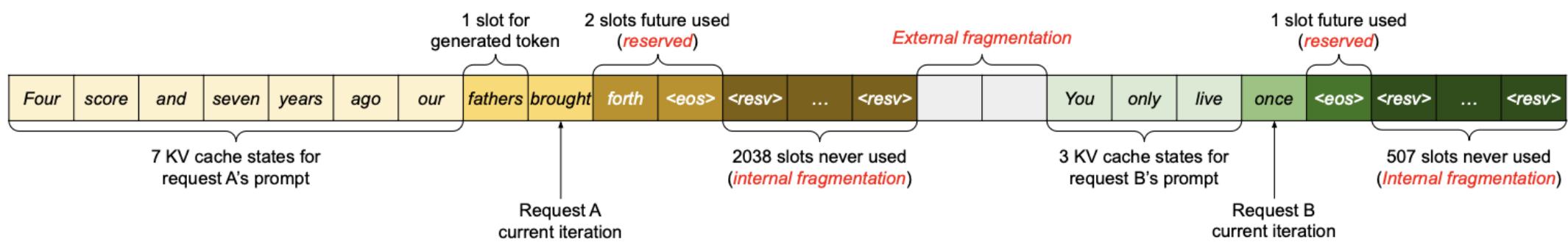


Potential Solutions



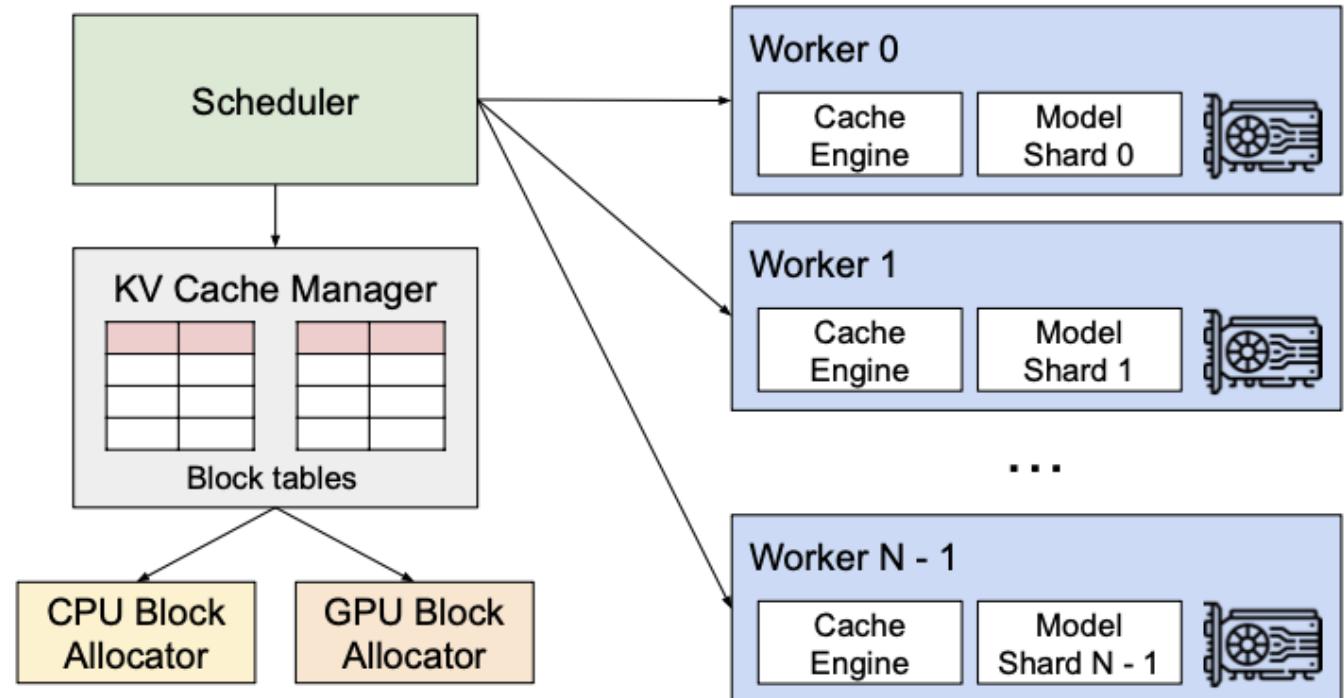
Memory fragmentation problem in KV Cache

prompt1	I	like	eating	...	<eos>	<resv>	<resv>	<resv>			
prompt2	Today	she	when	to	...	<eos>	<resv>	<resv>			
prompt3	Last	night	...	<eos>	<resv>	<resv>	<resv>	<resv>			



vLLM Architecture

vLLM adopts a centralized scheduler to coordinate the execution of distributed GPU workers. The KV cache manager effectively manages the KV cache in a paged fashion, enabled by PagedAttention. Specifically, the KV cache manager manages the physical KV cache memory on the GPU workers through the instructions sent by the centralized scheduler.



Paged Attention

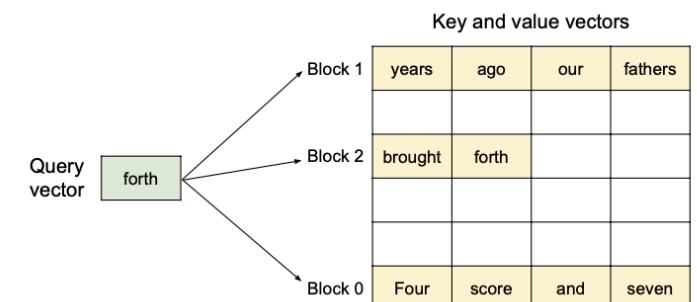
Paged Attention allows storing continuous keys and values in non-contiguous memory space

$$A_{ij} = \frac{\exp(q_i^\top K_j / \sqrt{d})}{\sum_{t=1}^{\lceil i/B \rceil} \exp(q_i^\top K_t \mathbf{1} / \sqrt{d})}, \quad o_i = \sum_{j=1}^{\lceil i/B \rceil} V_j A_{ij}^\top,$$

A_{ij} is the row vector of attention score on j -th KV block.

Key block K_j

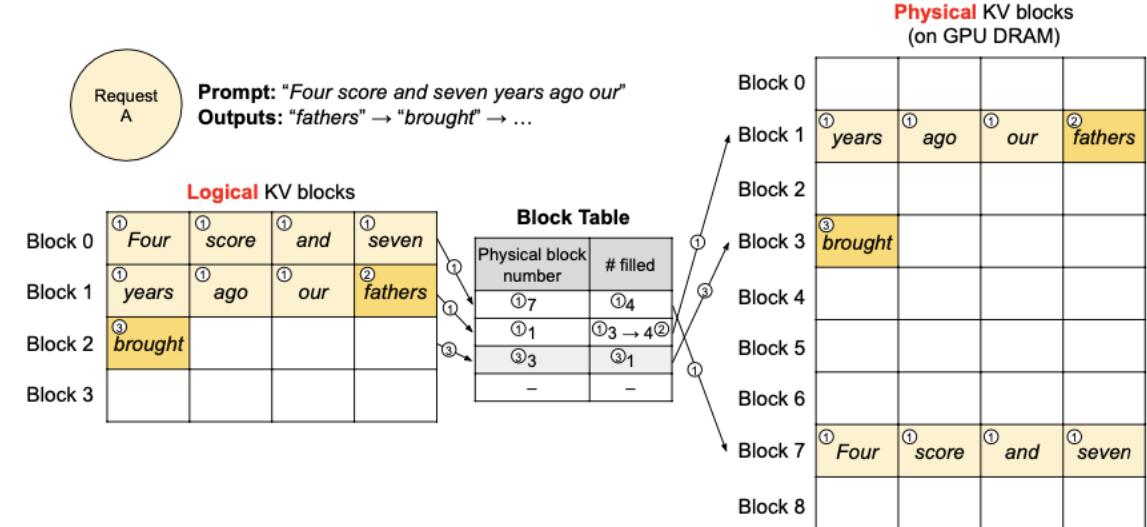
Value block V_j



KV Cache Manager

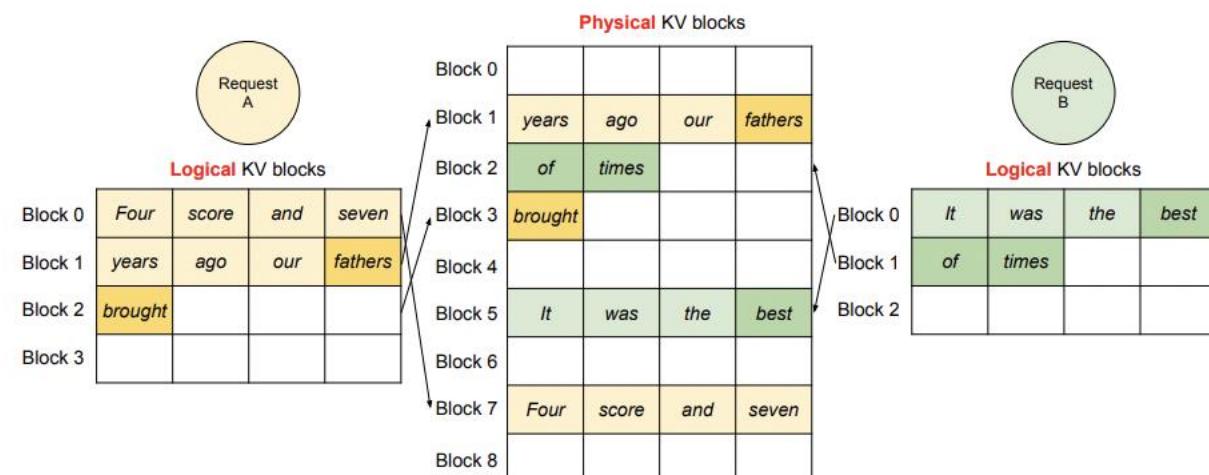
The key idea behind vLLM's memory manager is analogous to the virtual memory in operating systems

A request's KV cache is represented as a series of logical KV blocks, filled from left to right as new tokens and their KV cache are generated. The last KV block's unfilled positions are reserved for future generations.



Decoding with PagedAttention

- vLLM first selects a set of candidate sequences for batching and allocates the physical blocks for the newly required logical blocks.
 - It concatenates all the input tokens of the current iteration (all tokens for prompt phase and the latest tokens for generation phase) as one sequence and feeds it into the LLM.
 - Once a request finishes its generation, its KV blocks can be freed to store the KV cache of other requests.



- **Parallel Sampling:** LLM generates multiple sampled outputs for a single input prompt.
- One request includes multiple samples sharing the same input prompt, allowing the KV cache of the prompt to be shared as well.
- vLLM implements a copy-on-write mechanism at the block granularity for the physical blocks that need modification by multiple sequences.
- Reference Counts: Sequences or processes referencing a shared resource.

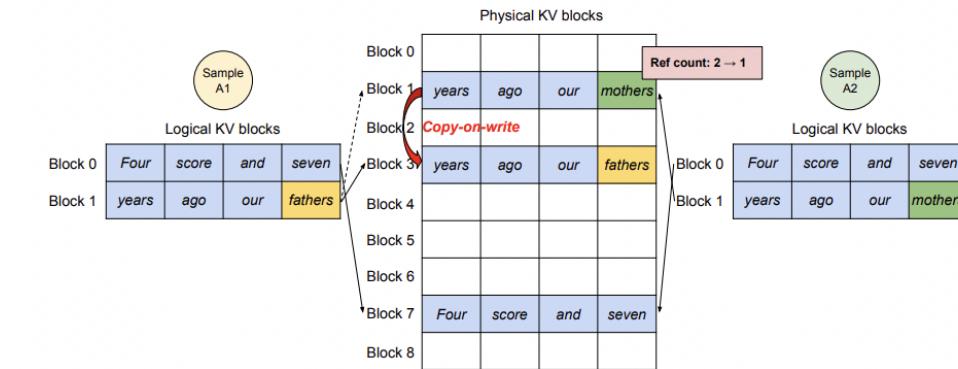


Figure 8. Parallel sampling example.

- **Beam Search:** In tasks like machine translation, the users expect the top- k most appropriate translations output by the LLM.
- Beam width parameter k - determines the number of top candidates retained at every step.
- Beam search facilities sharing not only the initial prompt blocks but also other blocks across different candidates, and the sharing patterns dynamically change as the decoding process advances.

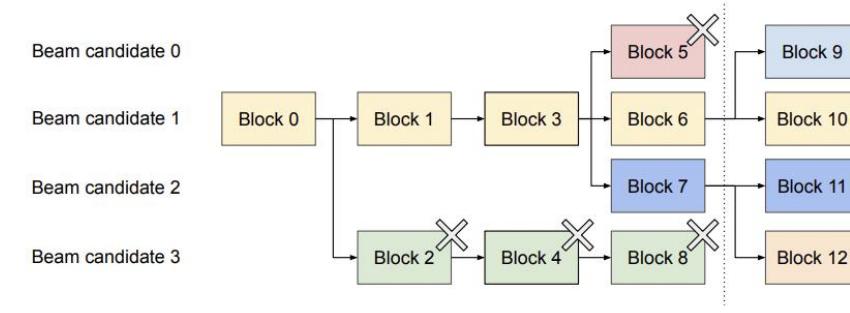
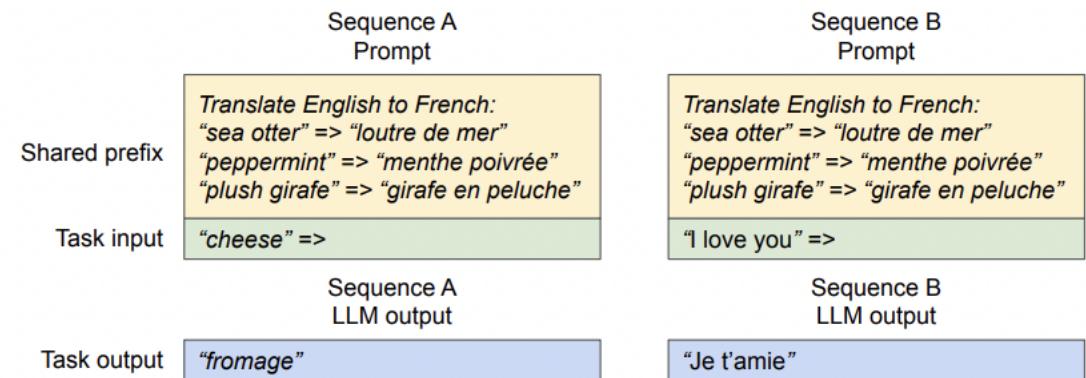


Figure 9. Beam search example.

- **Shared Prefix:** *System prompt* : LLM user provides a (long) description of the task including instructions and example inputs and outputs.
- For this type of application, many user prompts share a prefix, thus the LLM service provider can store the KV cache of the prefix in advance to reduce the redundant computation spent on the prefix.
- **Mixed Decoding:** vLLM conceals the complex memory sharing between different sequences via a common mapping layer that translates logical blocks to physical blocks.



Scheduling and Preemption

- When request traffic surpasses the system's capacity: Which blocks should it evict? How to recover evicted blocks if needed again?
 - Swap/Recompute the whole request since all previous tokens are required in every step.
 - Recomputation is surprisingly fast as the KV caches for all tokens can be computed in parallel (Tokens generated at decoding can be concatenated with the original user prompt as a new prompt).
 - Swapping : When vLLM exhausts free physical blocks for new tokens, it selects a set of sequences to evict and transfer their KV cache to the CPU. Smaller the block size higher is the overhead as this requires more frequent transfers.

Distributed Execution

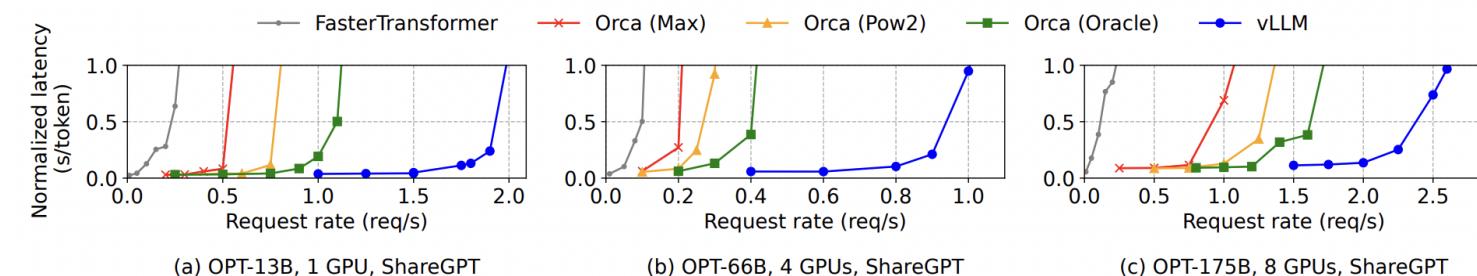
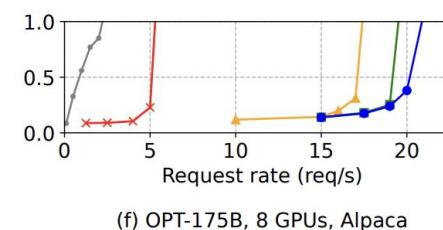
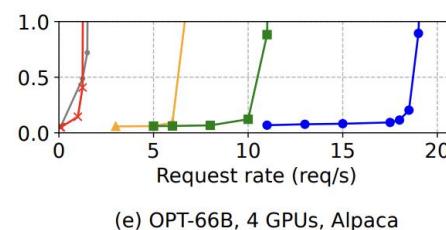
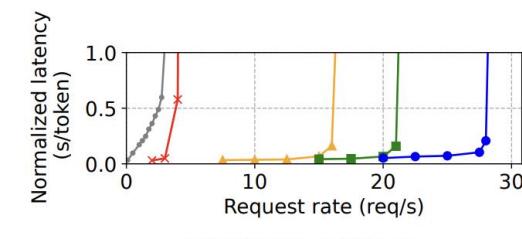
- LLMs often exceed the memory capacity of a single GPU.
Distributed execution is required for scalability across multiple GPUs.
- vLLM Supports:
 - **Tensor model parallelism** (e.g., Megatron-LM style).
 - Single **KV Cache Manager** within a centralized scheduler. (Shared across distributed GPU workers, Manages logical-to-physical KV block mapping)
- Scheduler broadcasts input token IDs and block table mappings.
GPU Workers execute attention layers, accessing KV cache via block tables. Synchronization happens only for intermediate results (e.g., using all-reduce).
- GPUs store KV cache only for their assigned attention heads.

Implementation

- Key Components: Scheduler, Block Manager, Custom Kernels.
 - **Fused Reshape and Write:** storage of KV blocks by combining reshaping and writing into a single efficient operation.
 - **Block Read and Attention:** Efficient memory access (read KV cache according to the block table and perform attention operations on the fly) during computation.
 - **Fused Copy:** Reduces overhead for parallel sampling and beam search. batches the copy operations for different blocks into a single kernel launch.
- vLLM implements various decoding algorithms using three key methods: fork, append, and free.

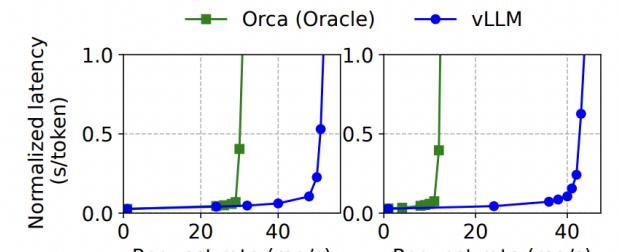
Evaluation

- **Configurations:** A2 instances with NVIDIA A100 GPUs on Google Cloud Platform.
- **Metric:** Serving throughput
- **Input/Output Length Distribution**
 - Alpaca dataset (instruction-following)
 - ShareGPT dataset (conversational tasks)
- **Baselines :** NVIDIA FasterTransformer (FT)
- Orca
 - Oracle: No over-reserve and know exact output lengths.
 - Pow2: Over-reserve the space for outputs by at most 2x.
 - Max: Over-reserve to the maximum possible output length.
- **Basic Sampling:** Each request generates a single sequence.
 - vLLM sustains **1.7×–8× higher request rates** than Orca baselines, Handles **4.3× more requests** compared to Orca (Max) in memory-bound scenarios.

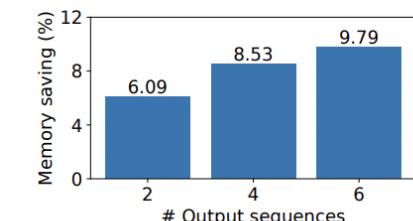


Sampling Methods Evaluated:

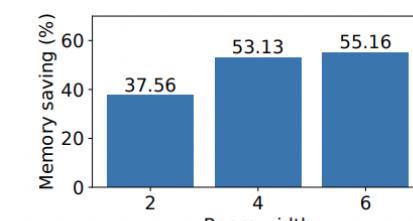
- **Shared Prefix:** vLLM achieves $1.67\times$ higher throughput than Orca (Oracle) when the one-shot prefix is shared.
- **Parallel Sampling:**
 1. All parallel sequences in a request share the KV cache for the prompt.
 2. Improvement over Orca increases with more sequences to sample.
- **Beam Search:**
 1. Allows for even greater memory sharing as beam width increases.
 2. vLLM performance improves from **1.3 \times (basic sampling)** to **2.3 \times (beam width of 6)** on the Alpaca dataset.



(a) 1-shot prefix prompt (b) 5-shot prefix prompt



(a) Parallel sampling



(b) Beam search

Memory Savings:

- **Parallel Sampling:**
 - **6.1% - 9.8%** (Alpaca Dataset)
 - **16.2% - 30.5%** (ShareGPT Dataset)
- **Beam Search:**
 - **37.6% - 55.2%** (Alpaca Dataset)
 - **44.3% - 66.3%** (ShareGPT Dataset)

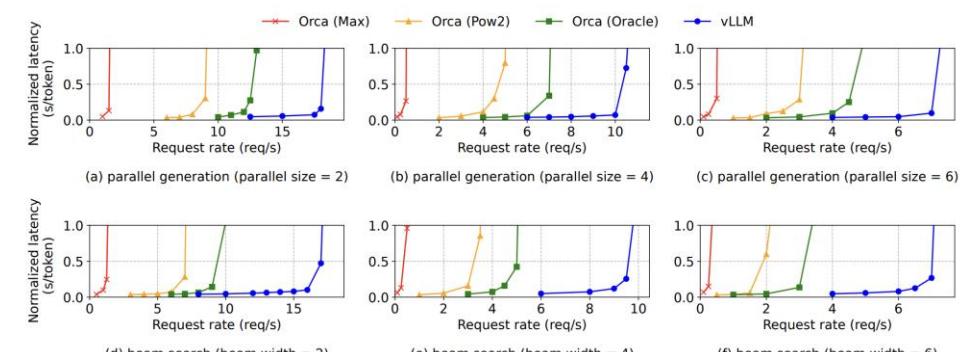
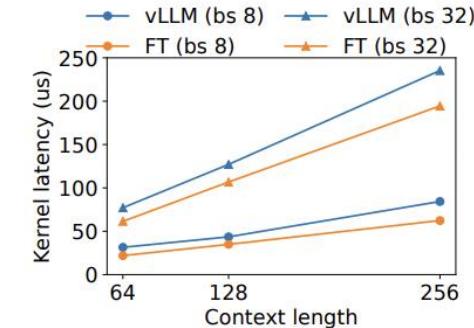


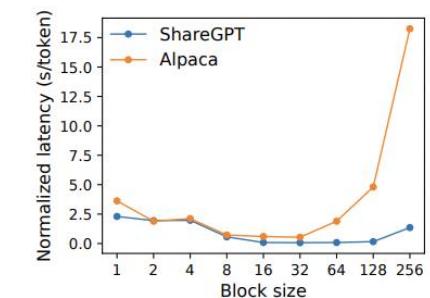
Figure 14. Parallel generation and beam search with OPT-13B on the Alpaca dataset.

Ablation Experiments

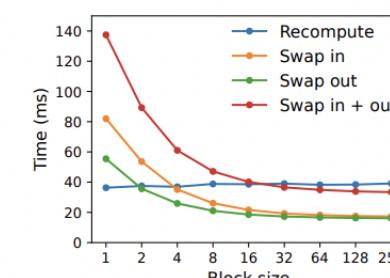
- **Kernel Microbenchmark** : Our GPU kernels involve extra overheads of accessing the block table, executing extra branches, and handling variable sequence lengths (20–26% higher attention kernel latency)
- **Impact of Block size** : Smaller block size – Do not utilize GPU’s parallelism, Larger block size – Increases Internal Frag.
- **Comparing Recomputation and Swapping** : small block sizes often result in numerous small data transfers between CPU and GPU.



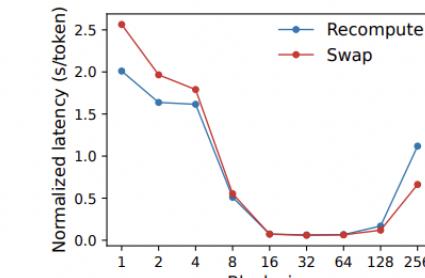
(a) Latency of attention kernels.



(b) End-to-end latency with different block sizes.



(a) Microbenchmark



(b) End-to-end performance

Discussion

- The virtual memory and paging technique used in vLLM effectively manages dynamic memory allocation in LLM serving, but it may not suit compute-bound or static tensor workloads like DNN training, where the tensor shapes are typically static.
- vLLM speeds things up by combining memory access and attention tasks into a single operation on the GPU, avoiding extra steps and making the process more efficient.



Can Generalist Foundation Models Outcompete Special-Purpose Tuning? Case Study in Medicine

Praneeth Somisetty, Vineet Gandham



Introduction

- Objective: Assess if generalist models can achieve specialist-level performance without domain-specific fine-tuning.
- Problem: Specialist models like Med-PaLM rely on costly fine-tuning with domain-specific data.
- Novel Idea: Medprompt, a novel prompting strategy that enhances GPT-4's performance in medical tasks.
- Generalist models can achieve and surpass benchmarks of specialist models with reduced costs

Prompting Strategies

Dynamic Few-Shot Selection:

- Uses k-Nearest Neighbors to select training examples relevant to the test case.

Self-Generated Chain-of-Thought (CoT):

- GPT-4 generates reasoning steps autonomously for complex questions.
- Enhances logical flow and problem-solving capabilities.

Choice Shuffling and Ensembling:

- Shuffles answer choices to eliminate positional bias.
- Combines multiple outputs for more robust predictions through majority voting.



Experimental Design:

Datasets:

- Nine medical datasets from the MultiMedQA benchmark suite
- Includes other domains like engineering and philosophy to test generalization.

Evaluation Methodology:

- "Eyes-off" testing: 20% of data held out during prompt engineering to control overfitting.
- Focus on maintaining general applicability across unseen data.

Performance Metrics:

- Accuracy measured across datasets.
- Comparison with state-of-the-art specialist models like Med-PaLM 2.

Methodology/Approach: Training

Q. A 40-year-old man presents with 5 days of productive cough and fever. Pseudomonas aeruginosa is isolated from a pulmonary abscess. Which of the following is the most appropriate antibiotic therapy?

- A) Ampicillin
- B) Ceftriaxone
- C) Ciprofloxacin
- D) Vancomycin

False CoT: Pseudomonas aeruginosa is a common pathogen, and broad-spectrum antibiotics like Ampicillin and Vancomycin are ruled out. Ceftriaxone is dismissed due to resistance concerns, so **Ciprofloxacin** is picked for general bacterial coverage, overlooking its specific effectiveness against Pseudomonas.

Discarded

Correct CoT: Pseudomonas aeruginosa is resistant to many antibiotics, and options like Ampicillin, Vancomycin, and Ceftriaxone are ineffective against it. **Ciprofloxacin**, however, has strong activity against this pathogen, making it the correct choice.

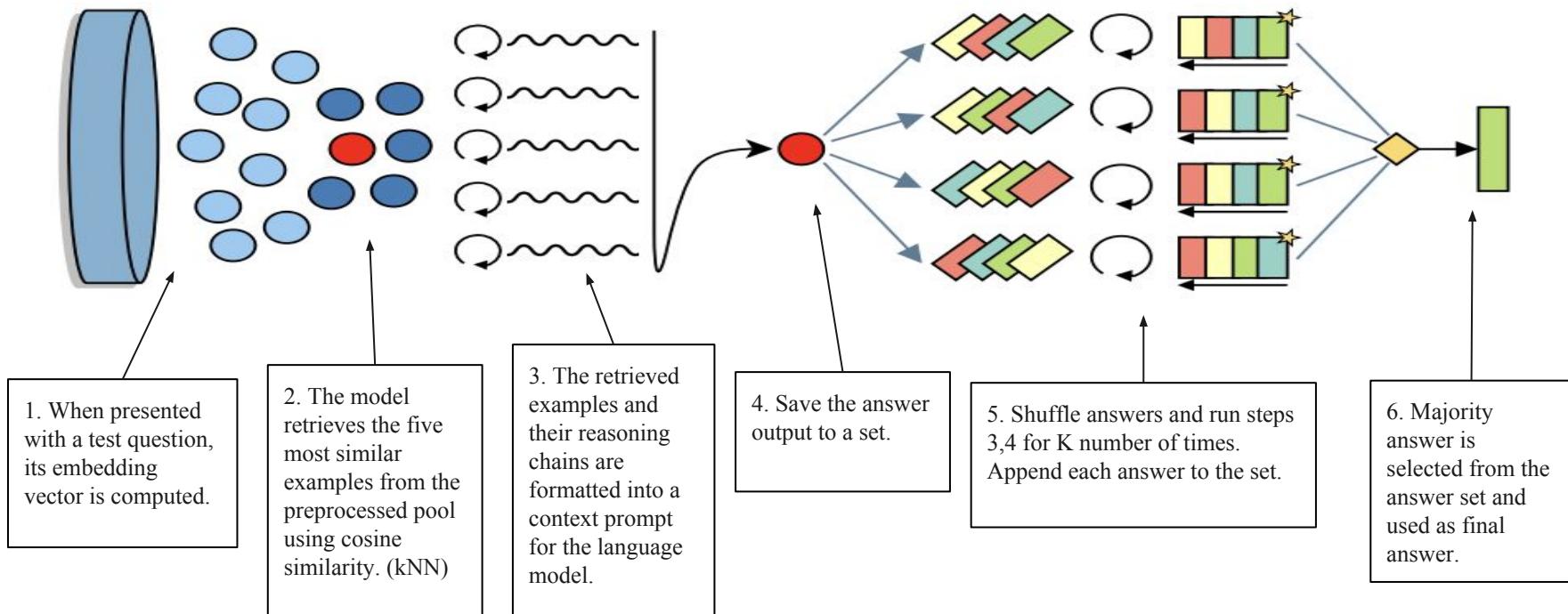
Vectorized (using OpenAI ada)
[9, 4, 6, 10...]

Question is Vectorized. While Ans, and CoT are saved and linked.

The choices are shuffled and multiple CoTs are generated. Only examples with correct answers are retained.

Ultimately, we have Vectorized questions (Vq), its corresponding correct answer, and correct CoT.

Methodology/approach: Inference



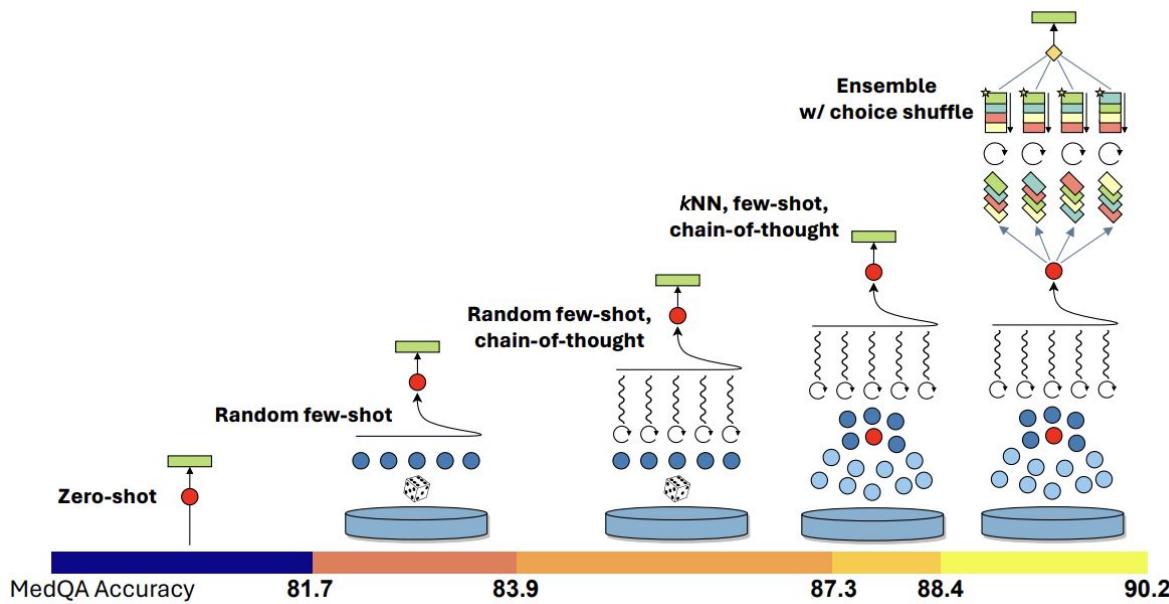
Outcome/Results

- GPT-4 with Medprompt achieves state-of-the-art performance across all tested datasets.
- MedQA (USMLE): 90.2% accuracy, surpassing Med-PaLM 2 by a wide margin.
- Error reduction of 27% on the MedQA dataset.

Generalization:

- Demonstrates applicability to domains beyond medicine (e.g., philosophy, engineering).
- Comparable improvement of ~7% in non-medical datasets.

Ablation studies



Zero-Shot Setting: The model answers test questions without any prior context, achieving a baseline accuracy of 81.7%.

Random Few-Shot: Providing randomly selected examples for context improves accuracy to 83.9%.

Random Few-Shot with CoT: Incorporating step-by-step reasoning (CoT) with random examples boosts accuracy to 87.3%.

kNN-Based Few-Shot with CoT: Using semantically similar examples via kNN selection, combined with CoT, raises accuracy to 88.4%.

Ensemble with Choice Shuffling: Combining kNN selection, CoT reasoning, and answer choice shuffling achieves the highest accuracy at 90.2%.



Ablation studies

	MedQA US (4-option)
Expert-crafted CoT prompt from [30]	83.8
GPT-4's self-generated CoT prompt	86.9 (+3.1)

Limitations/Risks

- **Memorization Concerns:** Efforts were made to address potential data leakage and memorization, but they cannot be fully ruled out.
- **Limited Real-World Transfer:** Strong benchmark performance may not translate to real-world healthcare applications due to task complexity.
- **Applicability Beyond MCQs:** Medprompt was not explicitly tested on non-multiple-choice tasks, limiting its demonstrated scope.
- **Hallucinations and Calibration:** Reducing hallucinations is challenging, though better calibration and confidence measures show promise.
- **Bias and Equity:** Ensuring fair and equitable model performance across populations(based on gender, race,etc) is crucial.

Reflection: Vineet

- Simple but effective technique! Takeaway is to utilize Medprompt for your task, but don't stop there.
- Methods like the PINE approach by Ziqi Wang and Prof. Zhang could be considered to enhance prediction diversity and performance robustness.
- Exploring multimodality and its effect on such prompting techniques could be beneficial. For example, retrieving relevant images for a question, etc.
- We could also embed chains of thought to be called back during inference. Makes sense to study embeddings that inherently include reasoning as an embedding model. Use fine-tuned bidirectional encoder such as BERT or even PaLM.
- A possible way to reduce hallucinations could be to keep negative CoT examples in the prompt so that the model hopefully understands what is wrong or right.

Reflection: Praneeth

- Generalist models, with proper prompting, can challenge traditional approaches relying on specialized models
- Medprompt's success shows the potential for prompt-based model adaptation

Future work:

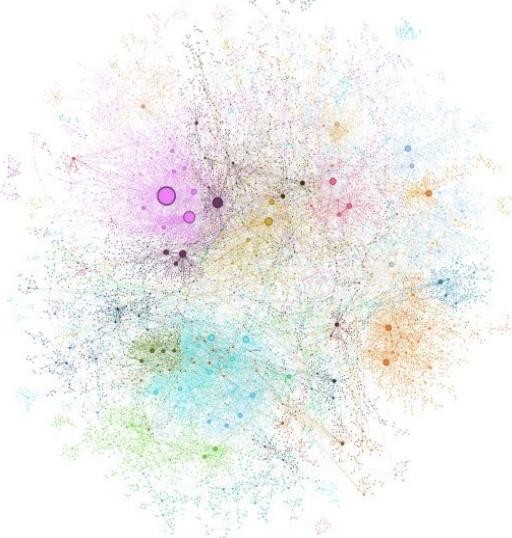
- Explore Medprompt for open-ended tasks and interactive applications.
- Refine verification methods for reasoning chains to reduce hallucination risks.

From Local to Global: A Graph RAG Approach to Query-Focused Summarization

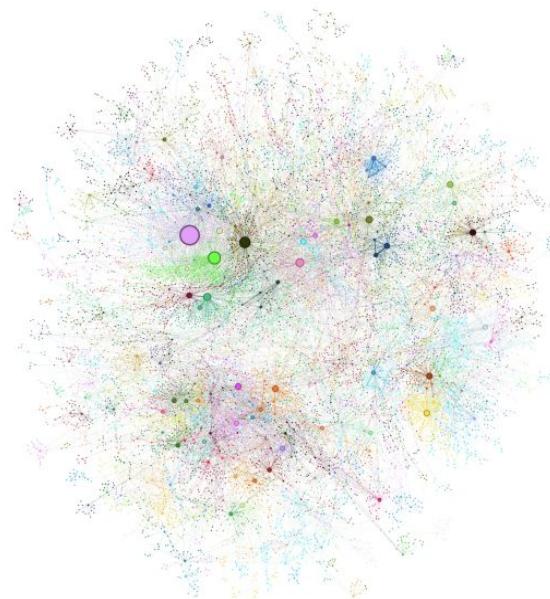


Presented By -
Manisha Panda
Neha Joshi

Introduction to GraphRAG



Root Communities at Level 0



Sub Communities at Level 1

- Retrieval Augmented Generation (RAG) is a technique that connects external data sources to enhance the output of large language models (LLMs).
- GraphRAG enhances RAG by incorporating knowledge graphs, which are data structures that store and link related or unrelated data based on their relationships.

Motivation

- A baseline RAG uses a vector database and a LLM.
- A vector database store and retrieves the contextual information from the given documents.
- LLM uses the context to generate the answer based on the questions asked.

Issues with Baseline RAG:

- Information retrieval focuses on similarity, not exact needs.
- Datasets may lack explicit answers to specific queries

Graph RAG leverages the relationship between the entities and concepts and thus enables a precise retrieval.

The main advantage is that, Graph RAG combines graph-based reasoning with retrieval to address multi-hop and implicit queries

Methodology

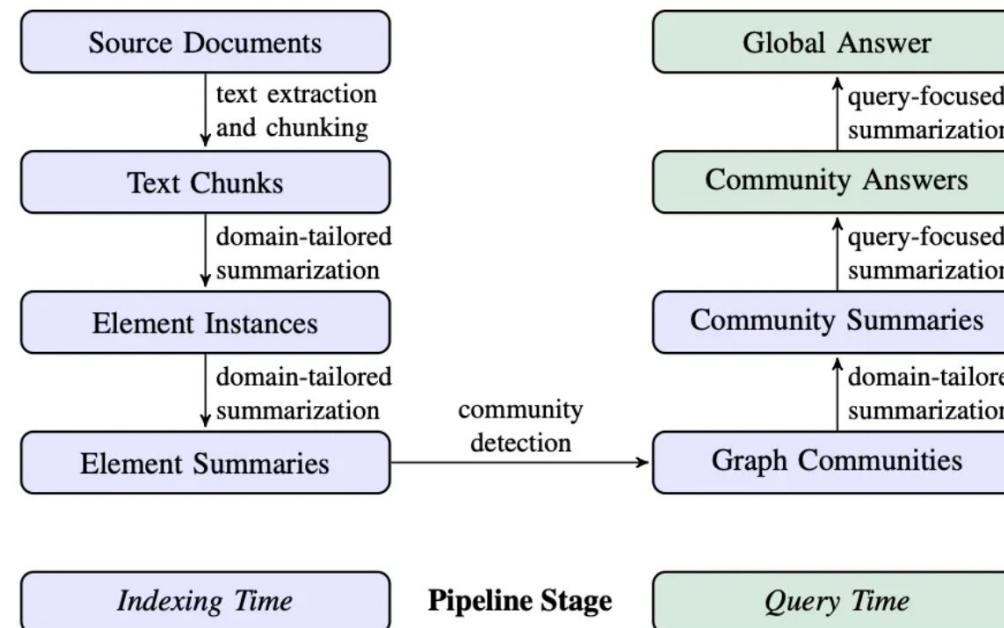
Graph RAG has two fundamental processes: Indexing and Querying

Indexing includes 4 major processes

- Text Unit Segmentation
- Entity, Relationship, and Claims Extraction
- Hierarchical Clustering
- Community Summary Generation

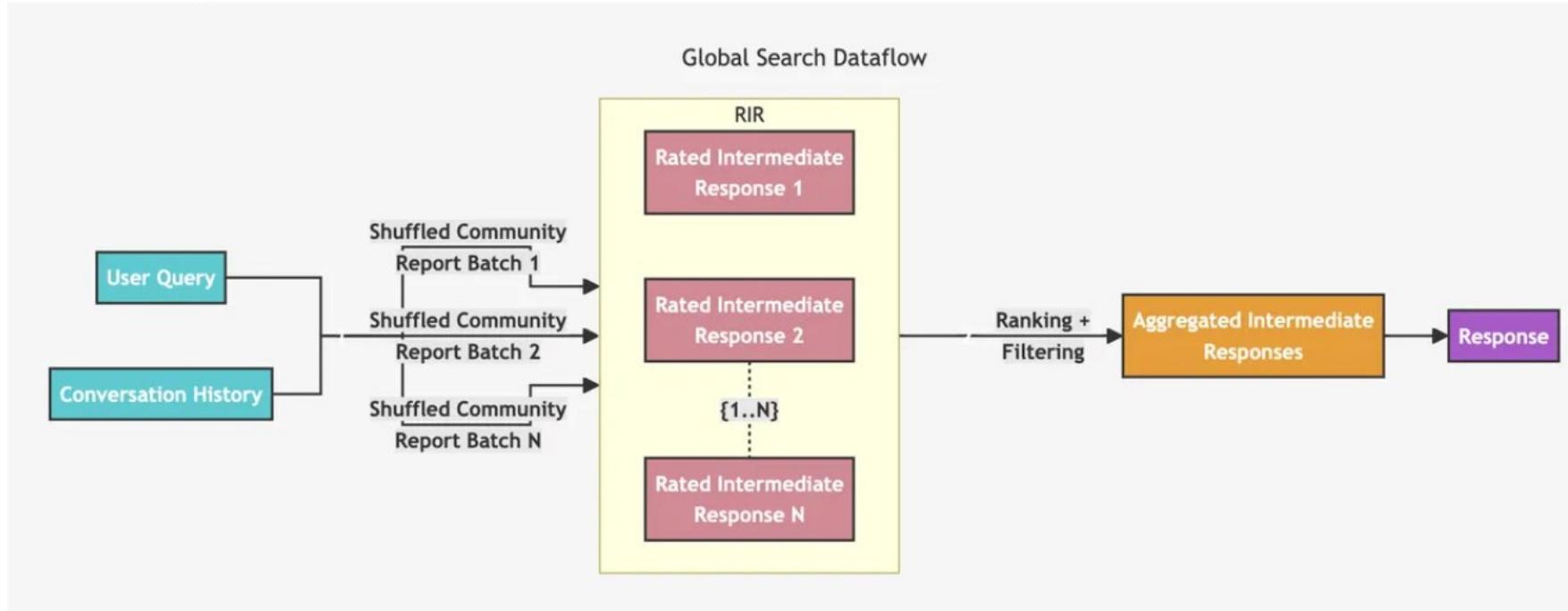
Graph RAG has 2 major querying techniques

- Global Search
- Local Search



The Graph RAG Pipeline

Methodology

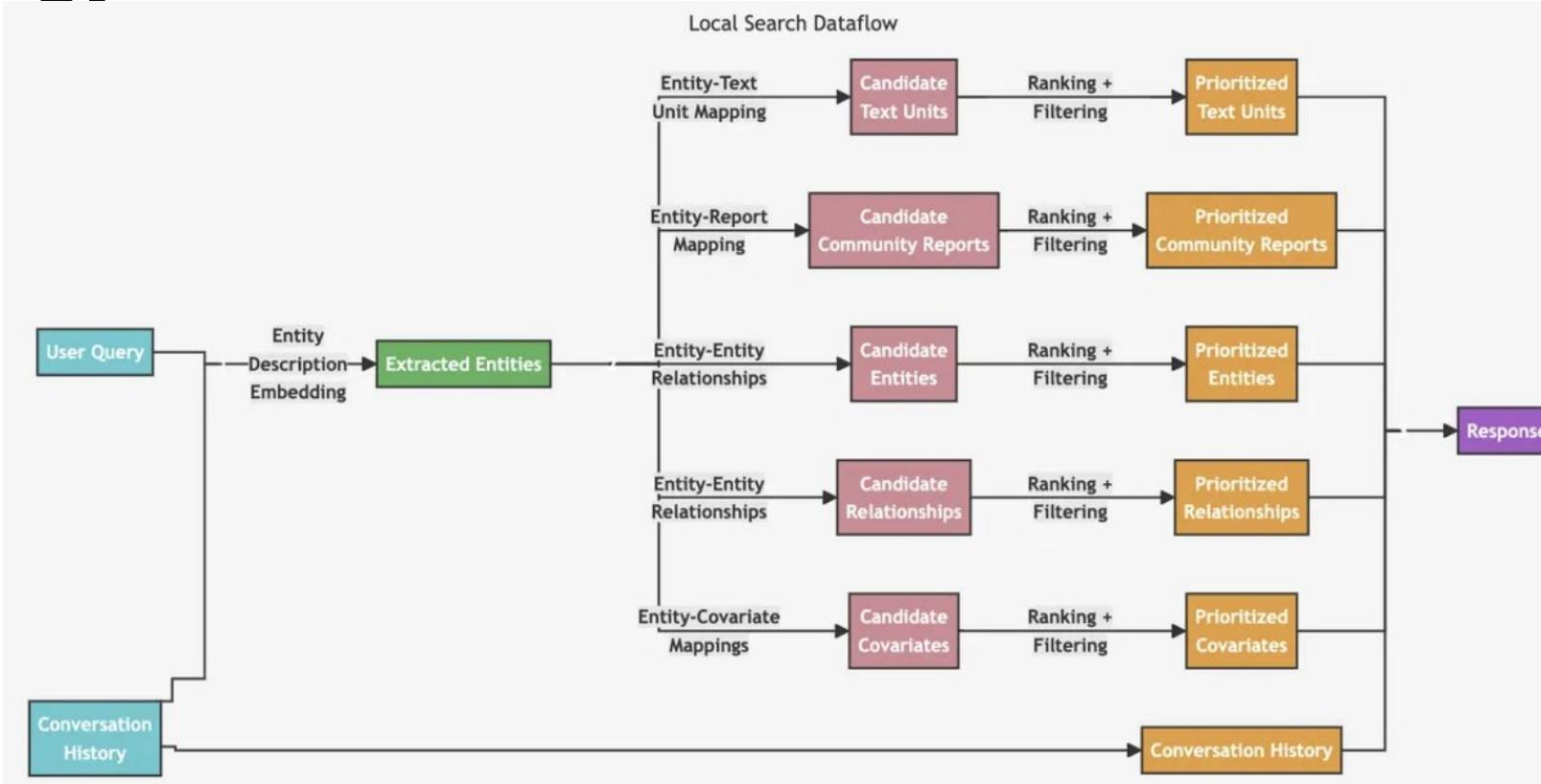


Global Search answers holistic questions related to the whole data corpus by using the community summaries

The workflow:

- User Query and Conversation History
- Community Report Batches
- Rated Intermediate Responses (RIR)
- Ranking and Filtering
- Final Response

Methodology



Local Search answers the specific entities by searching their neighbours and their entities.
The workflow is:

- User Query
- Similar Entity Search
- Entity-Text Unit Mapping
- Entity-Relationship Extraction

- Entity-Covariate Mapping
- Entity-Community Report Mapping
- Utilization of Conversation History
- Response Generation

Evaluation: Testing 2 datasets against 125 queries

Dataset and Configuration

- Podcast: behind the Tech, Scott, 2024
- 1,669 x 600 token text chunks
- 100 token overlap
- ~1 M tokens in total



- News Articles: benchmark dataset - news from Sep 2013 – Dec 2023
- 3,197 x 600 token text chunks
- 100 token overlap
- ~1.7 M tokens in total



8k context window proven universally best

LLM Assisted Query Generation



Dataset	Global sensemaking questions generated
Podcast transcript	User: A tech journalist looking for insights and trends in the tech industry Task: Understanding how tech leaders view the role of policy and regulation Q. Which episodes deal primarily with tech policy and government regulation?
News articles	User: Educator incorporating current affairs into curricula Task: Teaching about health and wellness Q. What current topics in health can be integrated into health education curricula? Q. How do news articles address the concepts of preventive medicine and wellness?

Generated 125 global questions per dataset

Evaluation Metrics

Comprehensiveness	Diversity	Empowerment	Directness
How much detail does the answer provide to cover all aspects and details of the question?	How varied and rich is the answer in providing different perspectives and insights on the question?	How well does the answer help the reader understand and make informed judgements about the topic?	How specifically and clearly does the answer address the question?

- Provide Question, target metric and pair of answers to be compared to the LLM
- Prompt the LLM to judge the answers based on the given metric
- Judgement needed: Win, Lose or Tie

6 Scenarios Tested for Answering Global Questions

C0

Uses root-level community summaries (fewest in number) to answer user queries.

C1

Uses high-level community summaries to answer queries. These are sub-communities of C0, if present, otherwise C0 communities projected down.

C2

Uses intermediate-level community summaries to answer queries. These are sub-communities of C1, if present, otherwise C1 communities projected down.

C3

Uses low-level community summaries (greatest in number) to answer queries. These are sub-communities of C2, if present, otherwise C2 communities projected down.

TS

Source texts (rather than community summaries) are shuffled and chunked for the map-reduce summarization stages.

SS

An implementation of naïve RAG in which text chunks are retrieved and added to the available context window until the specified token limit is reached.

Results

Podcast transcripts							
	SS	TS	C0	C1	C2	C3	
SS	50	17	28	25	22	21	SS
TS	83	50	50	48	43	44	TS
C0	72	50	50	53	50	49	C0
C1	75	52	47	50	52	50	C1
C2	78	57	50	48	50	52	C2
C3	79	56	51	50	48	50	C3

Comprehensiveness	Diversity	Empowerment	Directness
News articles			
Comprehensiveness	Diversity	Empowerment	Directness

	SS	TS	C0	C1	C2	C3	
SS	50	20	28	25	21	21	SS
TS	80	50	44	41	38	36	TS
C0	72	56	50	52	54	52	C0
C1	75	59	48	50	58	55	C1
C2	79	62	46	42	50	59	C2
C3	79	64	48	45	41	50	C3

Note: Row wins over columns are counted. Eg for comprehensiveness of podcast data, C0 wins over SS 72 times

- All GraphRAG scenarios outperform naïve RAG in Comprehensiveness and Diversity.
- Naïve RAG wins in directness for all scenarios.
- Empowerment shows mixed results in different scenarios.

Analysis

Advantages

- Global summarization and answering
- Graphs retain relations between data entities
- Answering in natural language
- Multiple levels of abstraction possible to answer the same question

Disadvantages

- Huge memory required as large graphs get generated
- High compute needed
- Directness of answers is compromised
- Testing done on limited data
- LLM based questions and evaluation

Conclusion

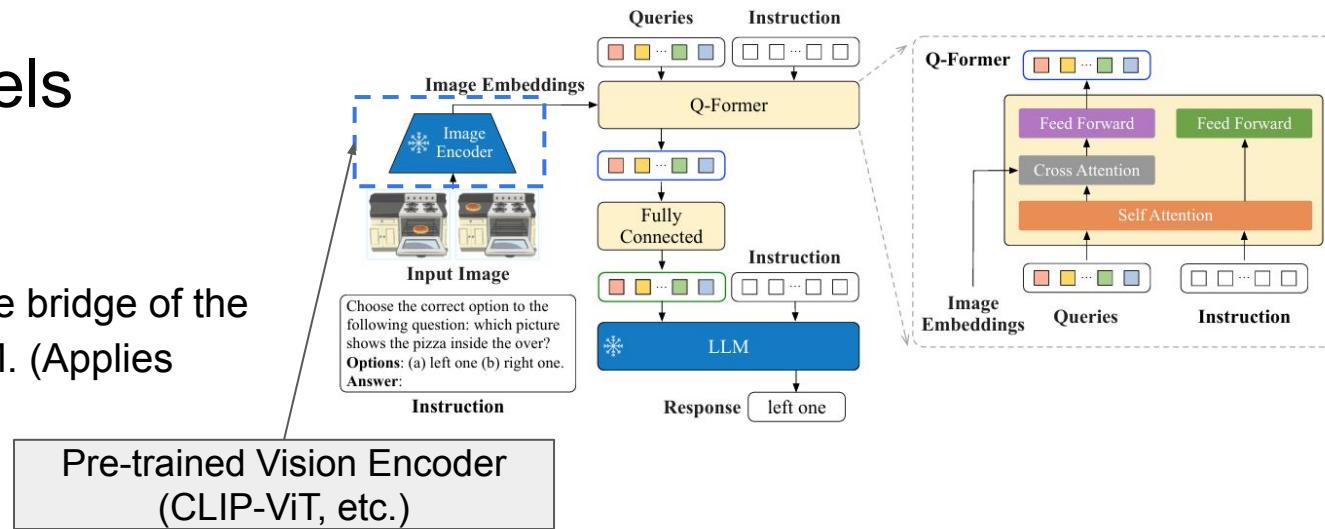
- Substantial improvements achieved for global question answering compared to naïve RAG.
- Multiple levels of abstractions achieved in answering. This might help in further downstream tasks as well.
- Outperforms naïve RAG in terms of Comprehensiveness and Diversity.
- Needs improvement in terms of directness in answers and empowerment.
- Scope for efficiency in terms of compute and memory.

Improved Baselines with Visual Instruction Tuning

Multimodal Models

InstructBLIP

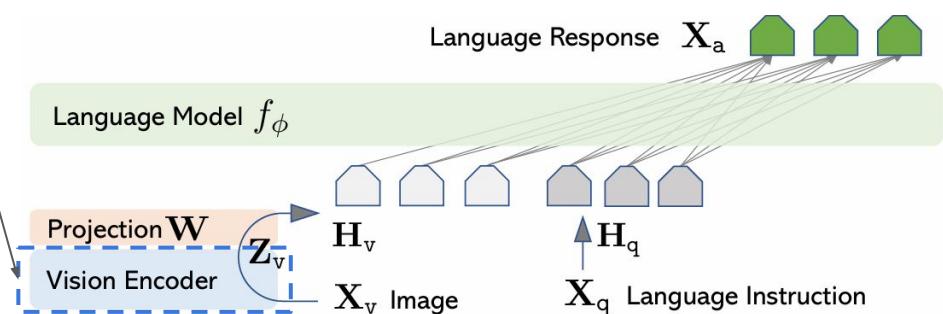
The Q-Former serves as the bridge of the vision-encoder and the LLM. (Applies cross-attention)



LLaVA

A projection matrix is applied to the image embeddings

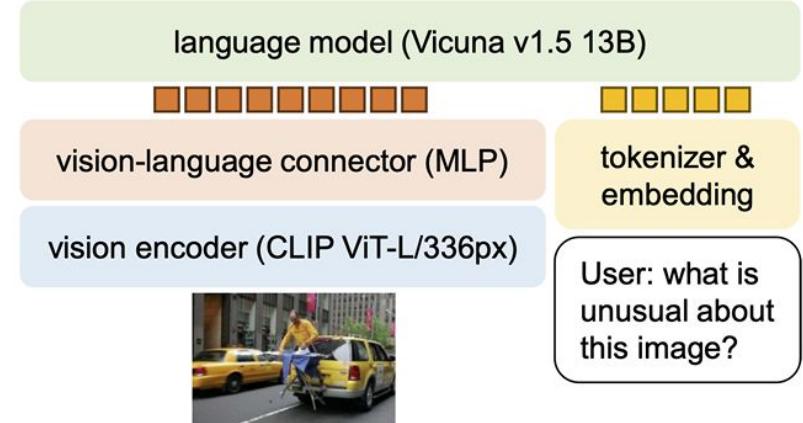
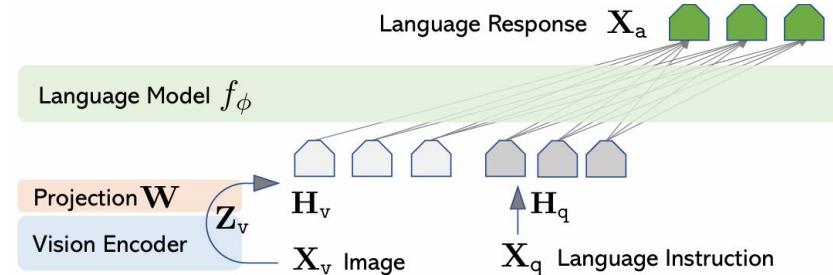
the result is then concatenated with the instruction embeddings.



LLaVA-1.5

Only some simple modifications to LLaVA

1. Replace the projection layer with 2 layers MLP
2. Apply response formatting prompts
3. Add academic-task-oriented data
4. Able to scale to higher resolutions



Training Process & Datasets

Two stages of training:

- Pretrain: freeze both the visual encoder and LLM
- Instruction Tuning: freeze only visual encoder

Academic task oriented datasets:

- OKVQA (Short-form answers)
- A-OKVQA (Multiple-choice questions)
- OCR (Short-form answers)
- GQA (Short-form answers)
- ...



Q: What does the man who sits have trouble doing?

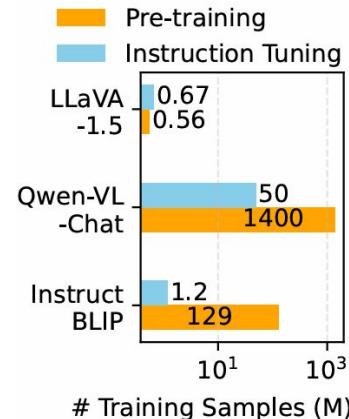
MC Answers:

- (a) Riding (b) Breathing (c) Walking (d) Magic

Direct Answers:

Walking, Walking, ...

A-OKVQA examples



LLaVA-1.5 has high training sample efficiency

Vehicles and Transportation



Q: What sort of vehicle uses this item?

A: firetruck

Brands, Companies and Products



Q: When was the soft drink company shown first created?

A: 1898

Objects, Material and Clothing



Q: What is the material used to make the vessels in this picture?

A: copper

OKVQA examples

Response Formatting Prompts

The InstructBLIP cannot handle the length of the output very well

Reasons include:

1. Ambiguous prompts, e.g. Q: {Question} A: {Answer}.
2. Only finetuning the Qformer, not finetuning the LLM

LLaVA-1.5 addresses the issue by finetuning LLM with response formatting prompts

Short-form answers: **Answer the question using a single word or phrase**

Multiple-choice questions: **Answer with the option's letter from the given choices directly**

Natural conversations: no special formatting prompt

Visual input example, Multitask Balancing Problem:

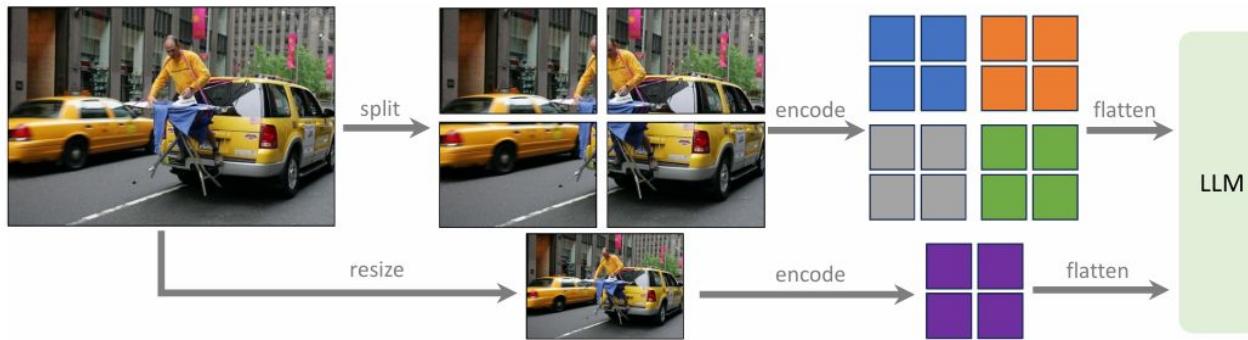


User	Is this unusual? Please explain in detail.
InstructBLIP	yes

Visual input example, Different Format Prompts:

Normal prompt	What is the color of the shirt that the man is wearing?
Response	The man is wearing a yellow shirt.
Ambiguous prompt	Q: What is the color of the shirt that the man is wearing? A:
Response	The man is wearing a yellow shirt.
Formatting prompt	What is the color of the shirt that the man is wearing? Answer the question using a single word or phrase.
Response	Yellow.

Scaling to Higher Resolutions



CLIP-ViT-L-336px is the highest resolution available for CLIP.

To support higher resolution, they split the image into grids and encode them independently.

Finally, the feature is combined with a global context to mitigate potential artifacts from the grid-based processing.

Benchmarks & Results

Method	LLM	Image Size	Sample Size		VQAv2 [19]	GQA [21]	VisWiz [20]	SciQA-IMG [38]	TextVQA [48]
			Pretrain	Finetune					
BLIP-2 [32]	Vicuna-13B	224 ²	129M	-	65.0	41	19.6	61	42.5
InstructBLIP [14]	Vicuna-7B	224 ²	129M	1.2M	-	49.2	34.5	60.5	50.1
InstructBLIP [14]	Vicuna-13B	224 ²	129M	1.2M	-	49.5	33.4	63.1	50.7
Shikra [8]	Vicuna-13B	224 ²	600K	5.5M	77.4*	-	-	-	-
IDEFICS-9B [22]	LLaMA-7B	224 ²	353M	1M	50.9	38.4	35.5	-	25.9
IDEFICS-80B [22]	LLaMA-65B	224 ²	353M	1M	60.0	45.2	36.0	-	30.9
Qwen-VL [3]	Qwen-7B	448 ²	1.4B [†]	50M [†]	78.8*	59.3*	35.2	67.1	63.8*
Qwen-VL-Chat [3]	Qwen-7B	448 ²	1.4B*	50M [†]	78.2*	57.5*	38.9	68.2	61.5*
LLaVA-1.5	Vicuna-7B	336 ²	558K	665K	78.5*	62.0*	50.0	66.8	58.2
LLaVA-1.5	Vicuna-13B	336 ²	558K	665K	80.0*	63.3*	53.6	71.6	61.3
LLaVA-1.5-HD	Vicuna-13B	448 ²	558K	665K	81.8*	64.7*	57.5	71.0	62.5
Specialist SOTA: PaLI-X-55B [11]					86.1*	72.1*	70.9*	-	71.4*

Method	POPE [34]			MME [17]	MMBench [37]		SEED-Bench [27]			LLaVA-Wild [36]	MM-Vet [55]
	rand	pop	adv		en	cn	all	img	vid		
BLIP2-14B [32]	89.6	85.5	80.9	1293.8	-	-	46.4	49.7	36.7	38.1	22.4
InstructBLIP-8B [14]	-	-	-	-	36	23.7	53.4	58.8	38.1	60.9	26.2
InstructBLIP-14B [14]	87.7	77	72	1212.8	-	-	-	-	-	58.2	25.6
Shikra-13B [8]	-	-	-	-	58.8	-	-	-	-	-	-
IDEFICS-9B [22]	-	-	-	-	48.2	25.2	-	44.5	-	-	-
IDEFICS-80B [22]	-	-	-	-	54.5	38.1	-	53.2	-	-	-
Qwen-VL [3]	-	-	-	-	38.2	7.4	56.3	62.3	39.1	-	-
Qwen-VL-Chat [3]	-	-	-	1487.5	60.6	56.7	58.2	65.4	37.8	-	-
LLaVA-7B [36]	76.3	72.2	70.1	809.6	38.7	36.4	33.5	37.0	23.8	62.8	25.5
LLaVA-1.5-7B	87.3	86.1	84.2	1510.7	64.3	58.3	58.6	66.1	37.3	65.4	31.1
LLaVA-1.5-13B	87.1	86.2	84.5	1531.3	67.7	63.6	61.6	68.2	42.7	72.5	36.1
LLaVA-1.5-13B-HD	87.5	86.4	85.0	1500.1	68.8	61.9	62.6	70.1	41.3	72.0	39.4

Emerging Properties

Visual input example, Tricky Question:



User If there are factual errors in the questions, point it out; if not, proceed to answering the question.
What's happening in the desert?

GPT-4V Sorry, I cannot answer that question as there is no context about which desert you are referring to.

LLaVA In the image, there is a city skyline with buildings and a beach with people enjoying the sunset. The scene is set in a desert, which is unusual for a beach setting.

LLaVA-1.5 There are no deserts in the image. The image features a beach with palm trees, a city skyline, and a large body of water.



Query: What are the things I should be cautious about when I visit here?

Reference (English): When visiting the pier over the lake, there are a few things you should be cautious about. First, ensure that you have appropriate footwear with good traction, as the pier may be wet or slippery, especially if it has been raining or if there is algae on the surface. Second, be mindful of the weather conditions, as the image shows a cloudy sky, which could indicate potential rain or storms. It is essential to stay safe and avoid being caught in a sudden storm or lightning. Lastly, be aware of your surroundings and any potential hazards, such as the presence of boats or other watercraft in the area. Always follow any posted signs or guidelines to ensure a safe and enjoyable experience.

Spanish: ¿Cuáles son las cosas de las que debería tener cuidado cuando visite aquí?

Cuando visite el lugar con la pier en el lago, deberá tener cuidado de las siguientes cosas:

1. Mantenerse alejado del borde de la pier para evitar caer al agua o sufrir una lesión.
2. No subir a la pier si está prohibido o si no está diseñada para ello.
3. No arrojar objetos o desperdiciar en el agua, ya que esto puede dañar el ecosistema del lago.
4. No interferir con la fauna local, como los pájaros que se pueden encontrar en la zona.

Si decide nadar en el lago, asegúrese de conocer las condiciones del agua, como la profundidad y la presencia de algas, y de tener las habilidades y equipos adecuados para hacerlo de manera segura.

Tenir en cuenta estas precauciones ayudará a garantizar una experiencia segura y responsable en el lugar.

Translation: When visiting the place with the pier at the lake, you should be careful about the following things:

1. Stay away from the edge of the pier to avoid falling into the water or getting injured.
2. Do not climb onto the pier if it is prohibited or not designed for it.
3. Do not throw objects or litter into the water, as this can harm the lake's ecosystem.
4. Do not interfere with the local wildlife, such as birds that may be found in the area.

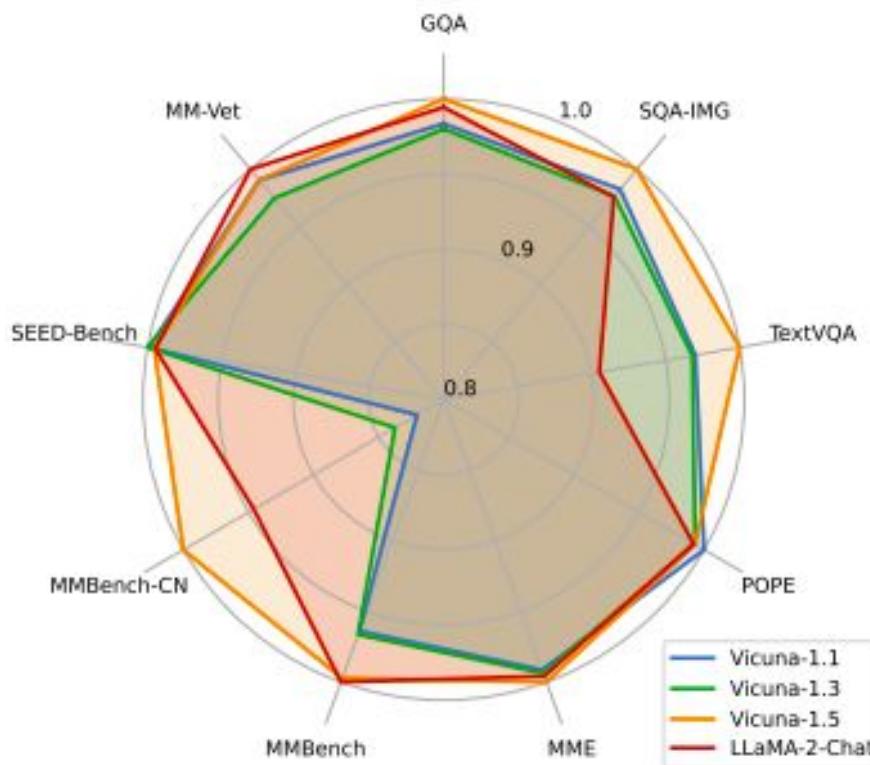
If you decide to swim in the lake, make sure to be aware of water conditions such as depth and the presence of algae, and have the appropriate skills and equipment to do so safely. Taking these precautions will help ensure a safe and responsible experience at the location.

Japanese: ここを訪れる際に注意すべきことは何ですか？

この場所は、寂しい湖に面した木製の桟橋です。訪れる際に注意すべきことは、桟橋の安全性と周囲の環境に対する配慮です。桟橋の構造が安定していることを確認し、桟橋が浸水していないことを確認することが重要です。また、湖の水深や周囲の地形に注意して、安全に桟橋を渡ることができるようになりますことも大切です。さらに、湖の水質や周囲の生態系に配慮して、環境に負荷をかけないようにすることも重要です。

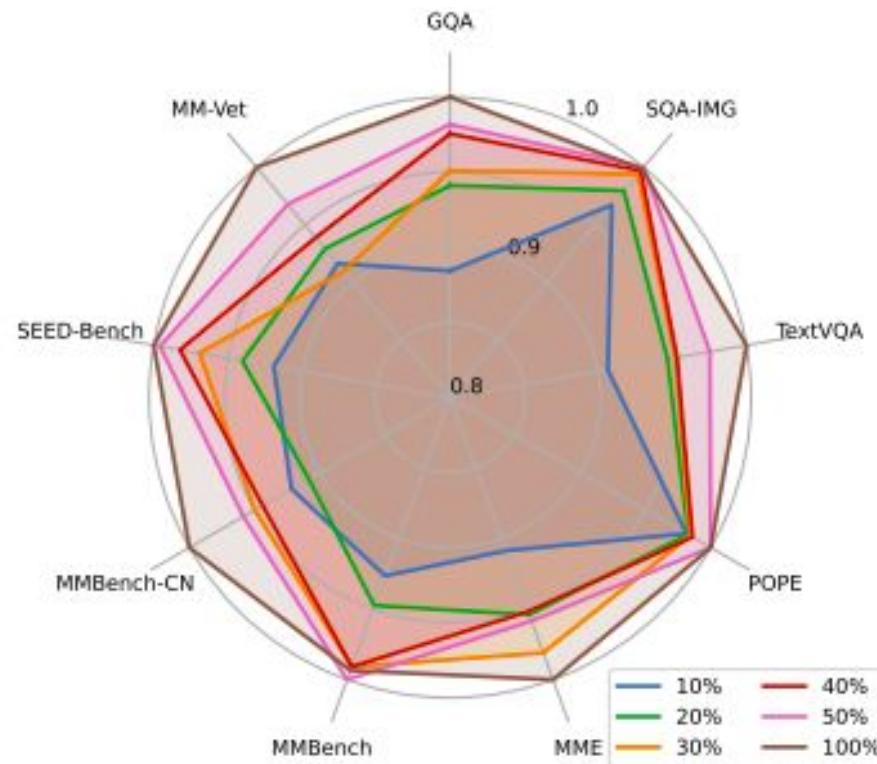
Translation: This place is a wooden pier facing a serene lake. When visiting, it's important to pay attention to the safety of the pier and respect the surrounding environment. Confirming the stability of the pier's structure and ensuring that it is not submerged in water are crucial. Additionally, being mindful of the lake's depth and the surrounding terrain is important to safely cross the pier. Furthermore, it's essential to consider the water quality of the lake and the local ecosystem to minimize any impact on the environment.

Ablation on LMM Choices



Data Efficiency

Includes Compositional
Capabilities



Conclusion & Limitations

Korean: 여기를 방문할 때 주의해야 할 사항은 무엇인가요?

이 장소는 물체 위의 부두로, 맑고 푸른 물체에 있습니다. 방문자들은 이 부두에서 물체를 감상하고, 물체 주변의 자연 경치를 즐길 수 있습니다. 그러나 이 부두는 물체 위에 있기 때문에, 물체의 수준이 높아질 수 있으므로, 방문자들은 물체의 수준을 확인하고 적절한 예방 조치를 취해야 합니다. 또한, 부두에서 물체로 뛰어들지 않도록 주의해야 합니다. 이 부두는 매우 아름다운 자연 경치를 제공하므로, 방문자들은 이 장소를 즐기면서 안전한 방법으로 물체를 감상할 수 있도록 주의해야 합니다.

Translation: This place is located on a pier above clear and blue water. Visitors can admire the object from this pier and enjoy the natural scenery around the object. However, since the pier is above the object, the object's level can be high, so visitors should check the object's level and take appropriate precautions. Also, be careful not to jump from the pier into the object. This pier offers incredibly beautiful natural scenery, so visitors should enjoy this place while being mindful of safe ways to appreciate the object.