Deep dive to efficient LLM: KV Compression



Pusan National University
Dept. of Electronic Engineering
Computer Vision & Signal Processing Lab.



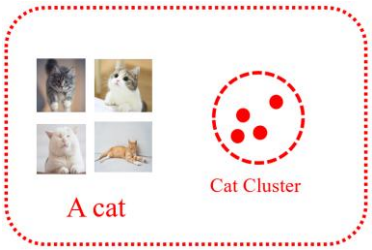
Presented By
백창우(M.S. Student, 2025.3~)

Outline

- Background
- Papers
  - SnapKV: LLM Knows What You are Looking for Before Generatio (NeurIPS 2024)
  - KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction (NeurIPS 2025 oral)
- Conclusion

Background

- Background
  - 하나의 단어(토큰)은 하나의 임베딩으로 표현
    - 유사한 단어일수록 거리가 가깝고 연관이 없는 단어일수록 멀게 벡터로 표현됨(Word2vec)



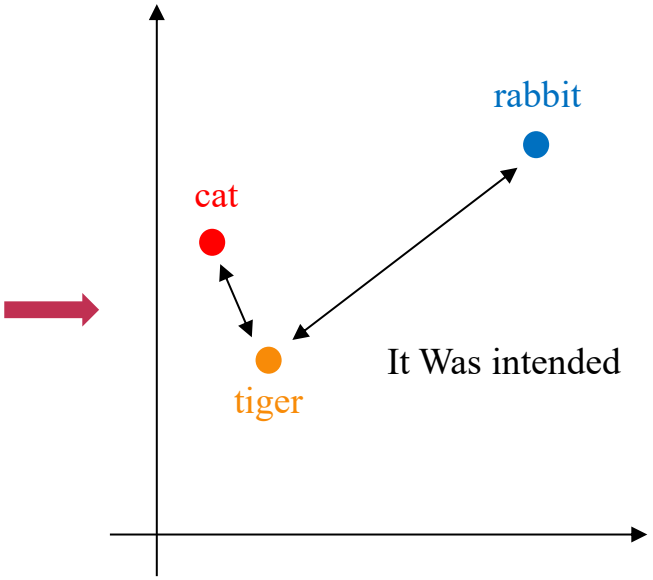
In vision

Lookup Table

Cat : [0.2, 0.5, …, -0.3]

tiger : [0.1, 0.4, …, -0.3]

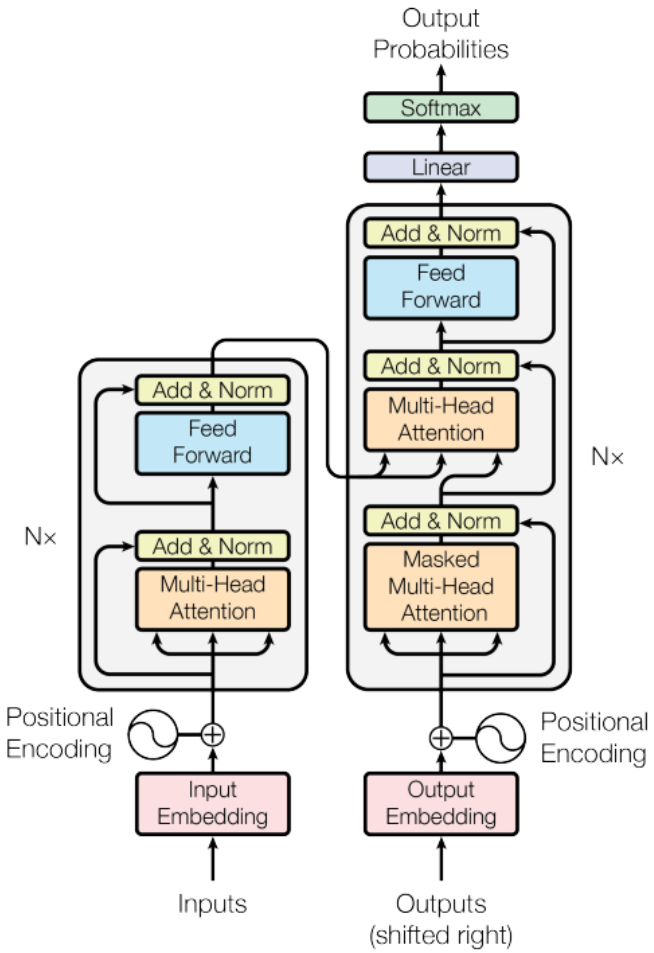
rabbit : [0.2, 0.6, …, -0.3]

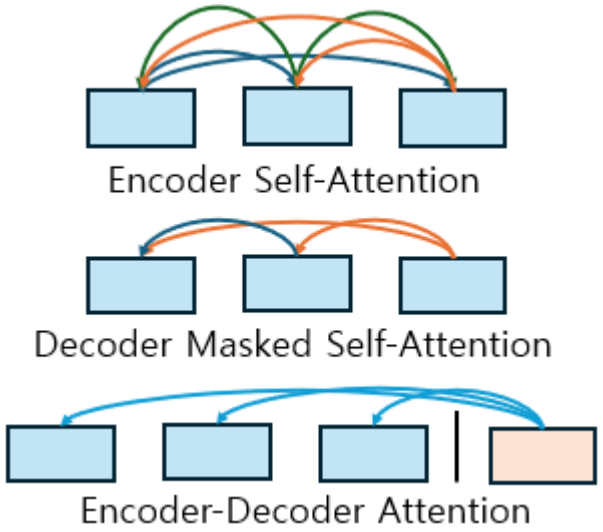


In Language

Background

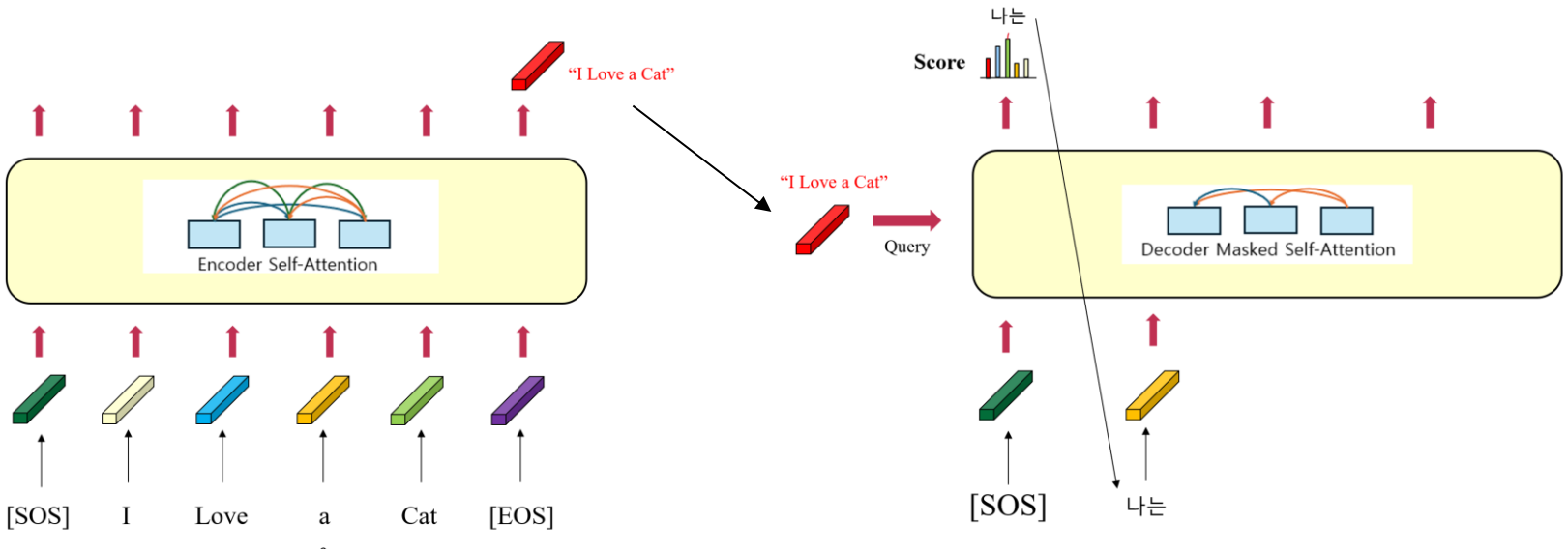
- Background
  - Transformer





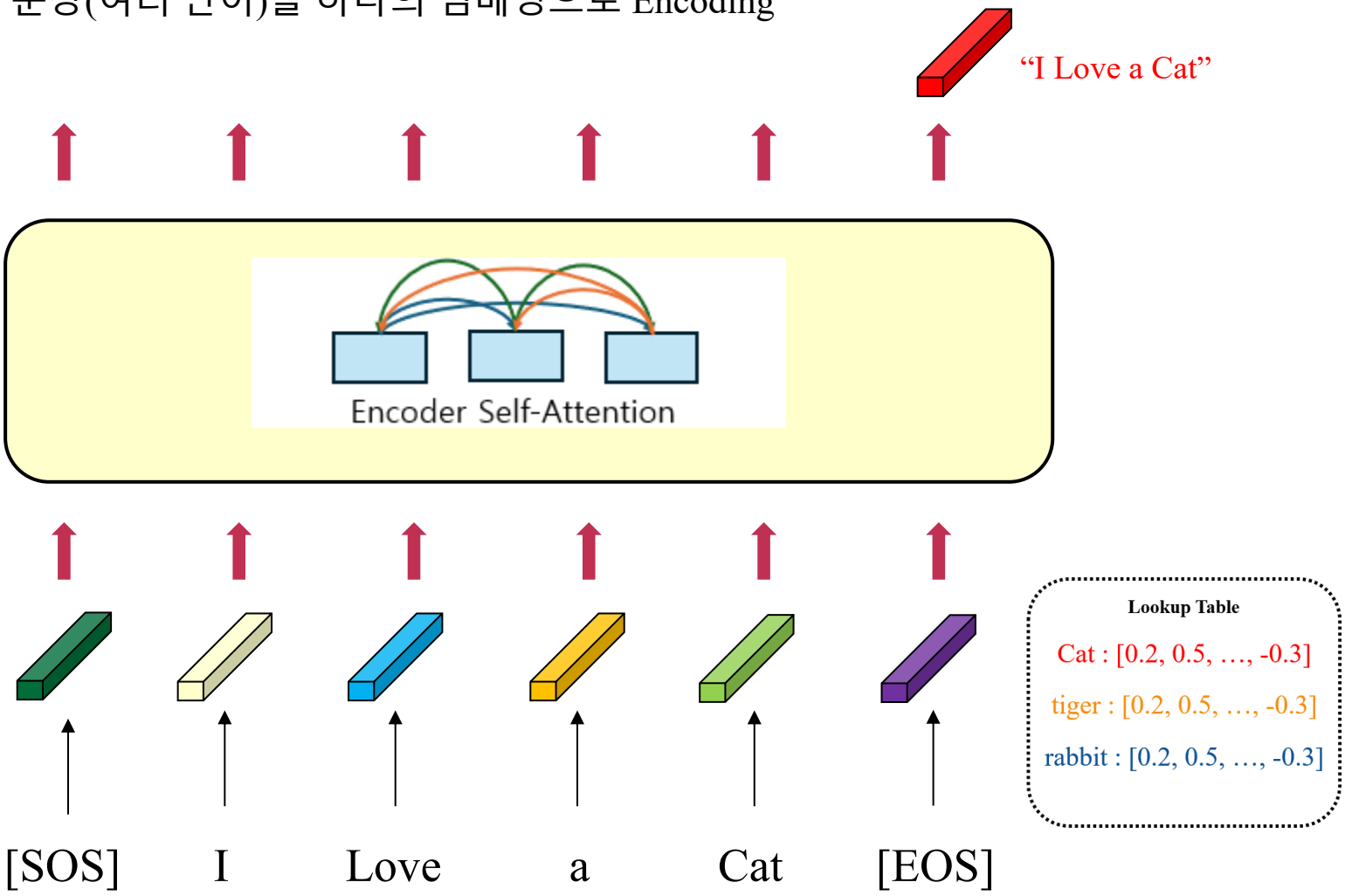
Background

- Background
  - Transformer



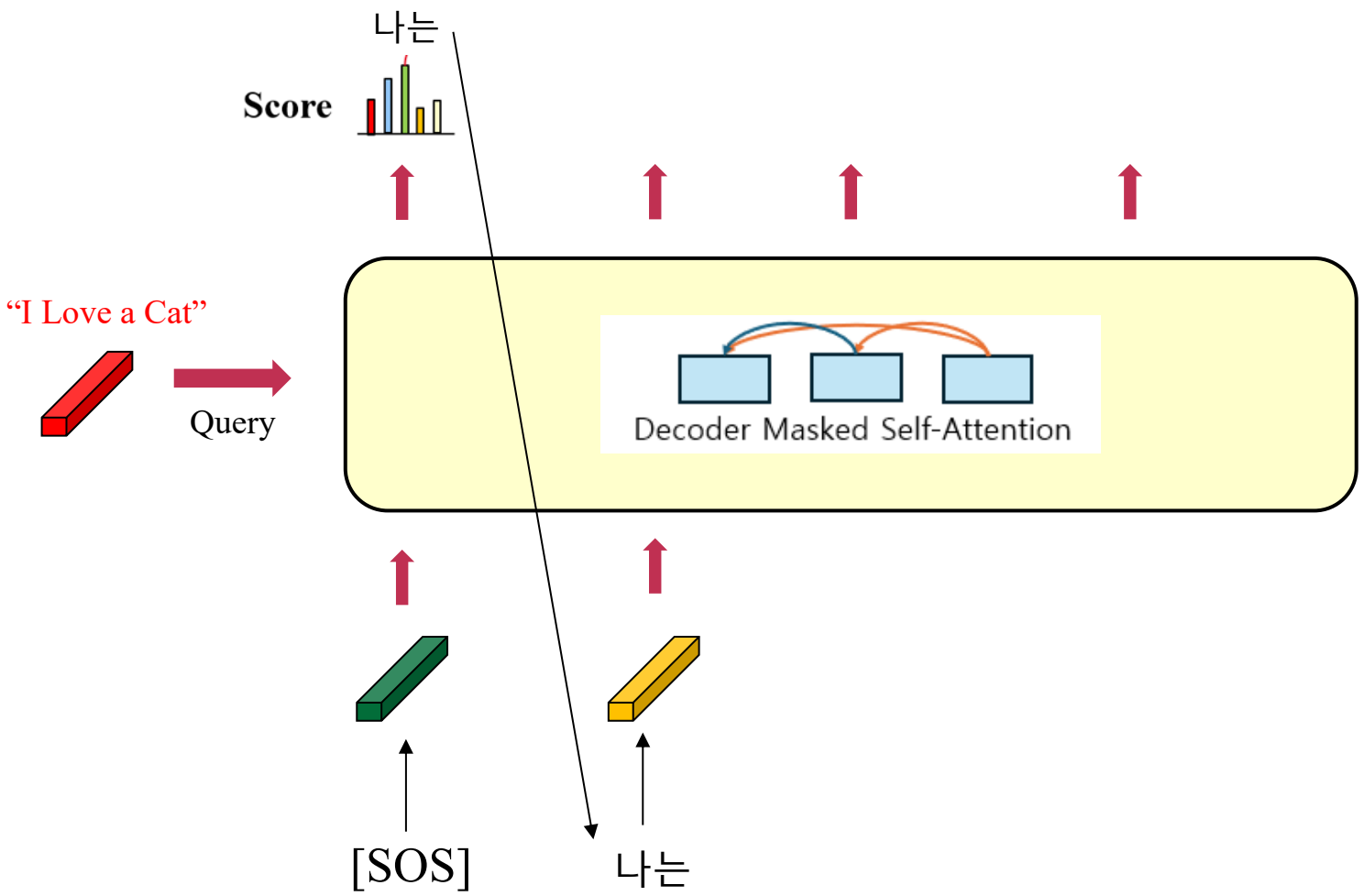
Background

- Background
  - Encoder
    - 입력 문장(여러 단어)를 하나의 임베딩으로 Encoding



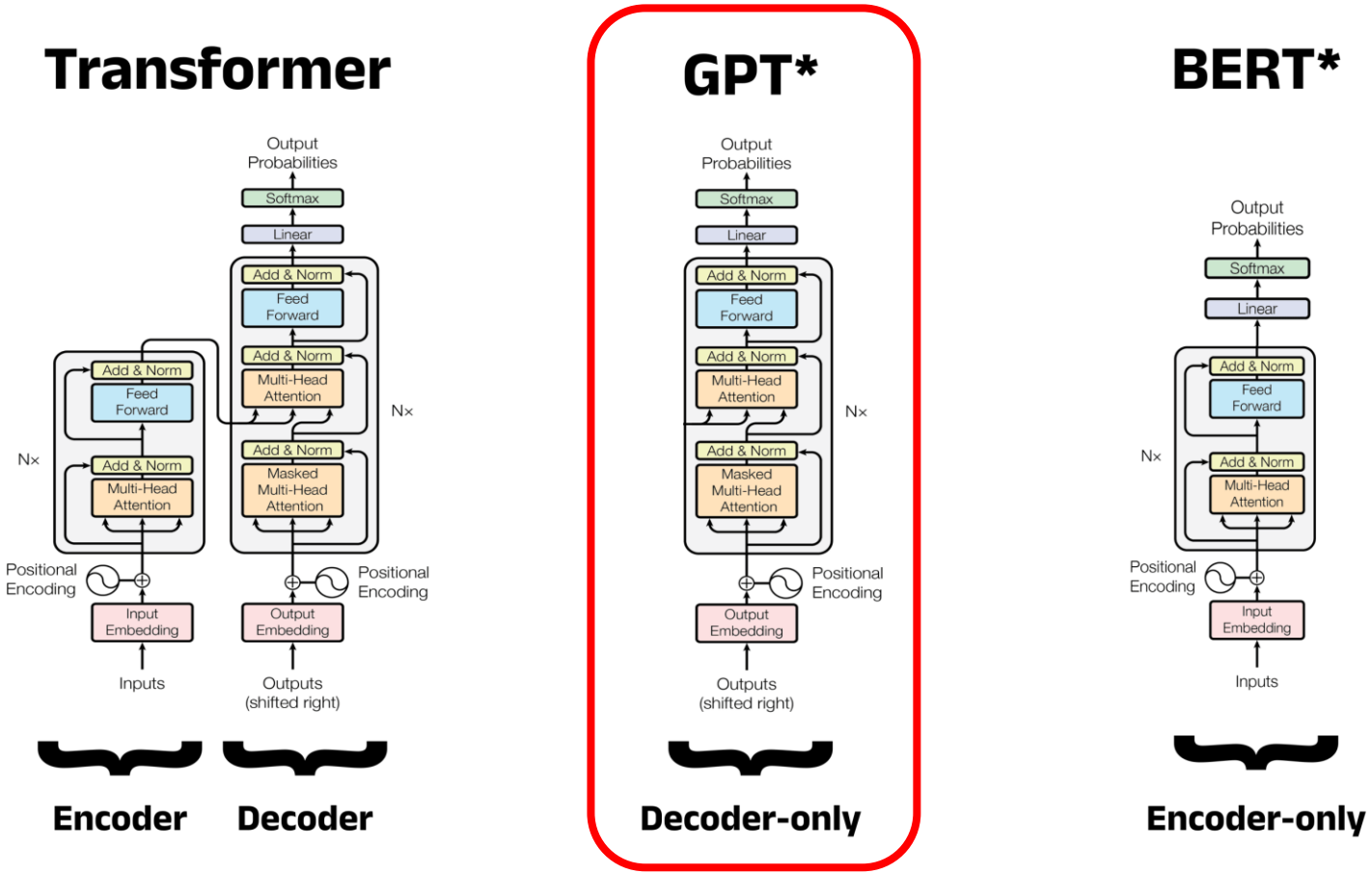
Background

- Background
  - Decoder
    - 인코딩된 임베딩으로부터 출력 단어(토큰)을 하나씩 디코딩(Auto-regressive decoding)



Background

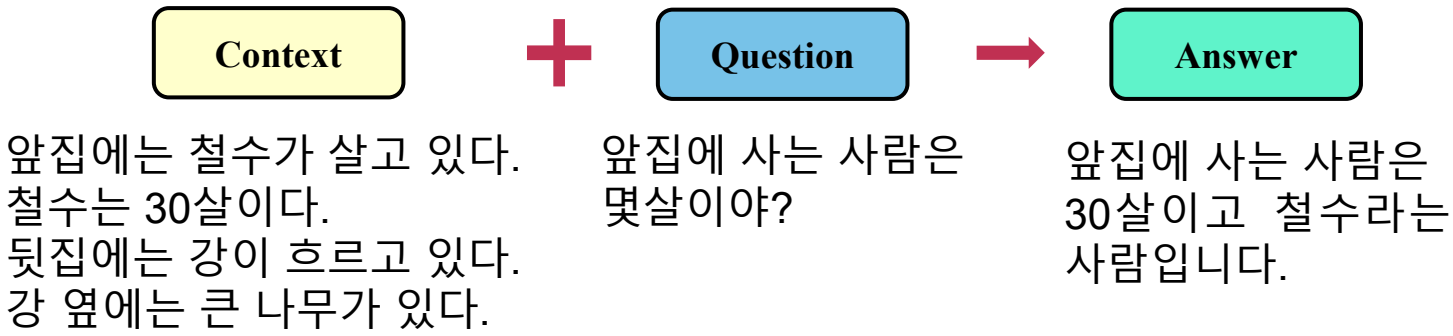
- Background
  - LLM



*Illustrative example, exact model architecture may vary slightly

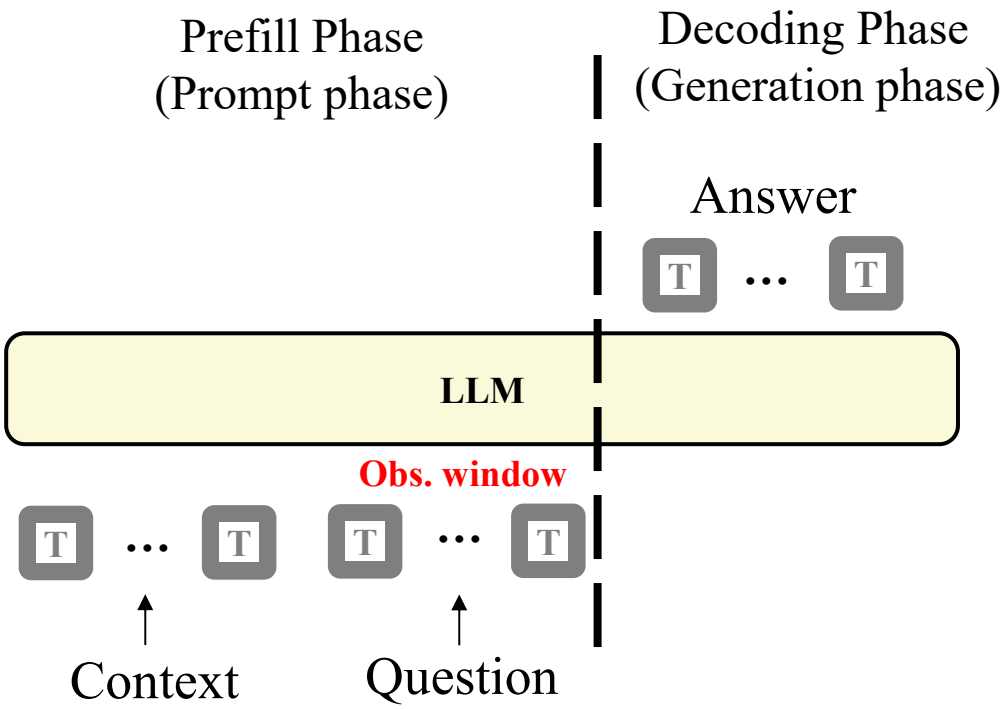
Background

- Background
  - LLM Task
    - Context: 사전 맥락
    - Question(Query): 질문 혹은 지시
    - Answer: 답변



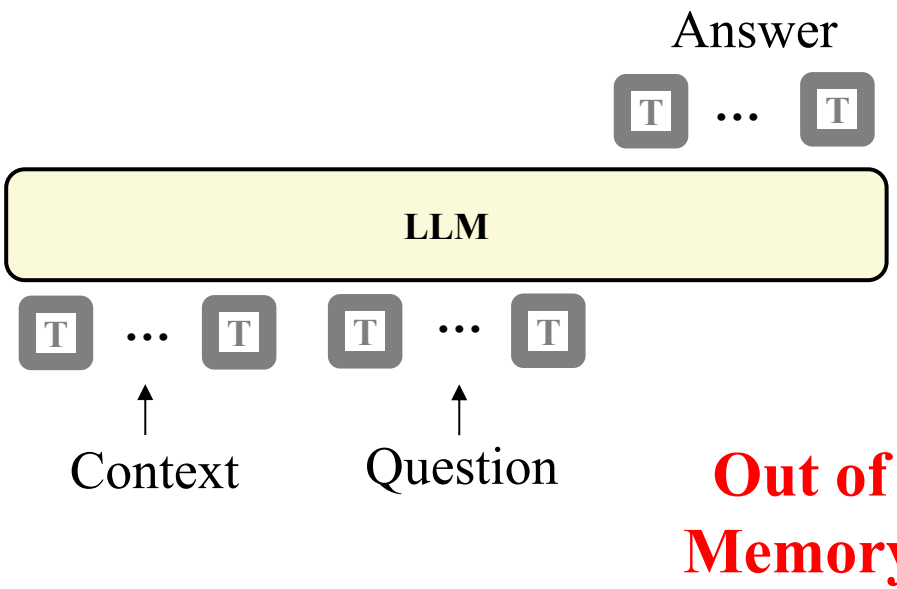
Background

- Background
  - LLM Inference



Background

- LLM Problem
  - Transformer의 입력 길이가 길어질 수록 연산 비용 및 메모리 사용량 증가
    - Attention 연산 복잡도는 $O(n^2)$에 비례
      - KV-Cache 활용 시 $O(n)$까지 감소 가능
        - ✓KV-Cache: 매번 다시 어텐션을 계산하지 않고 KV값을 저장
    - KV-Cache가 누적됨에 따라 메모리 사용량 증가

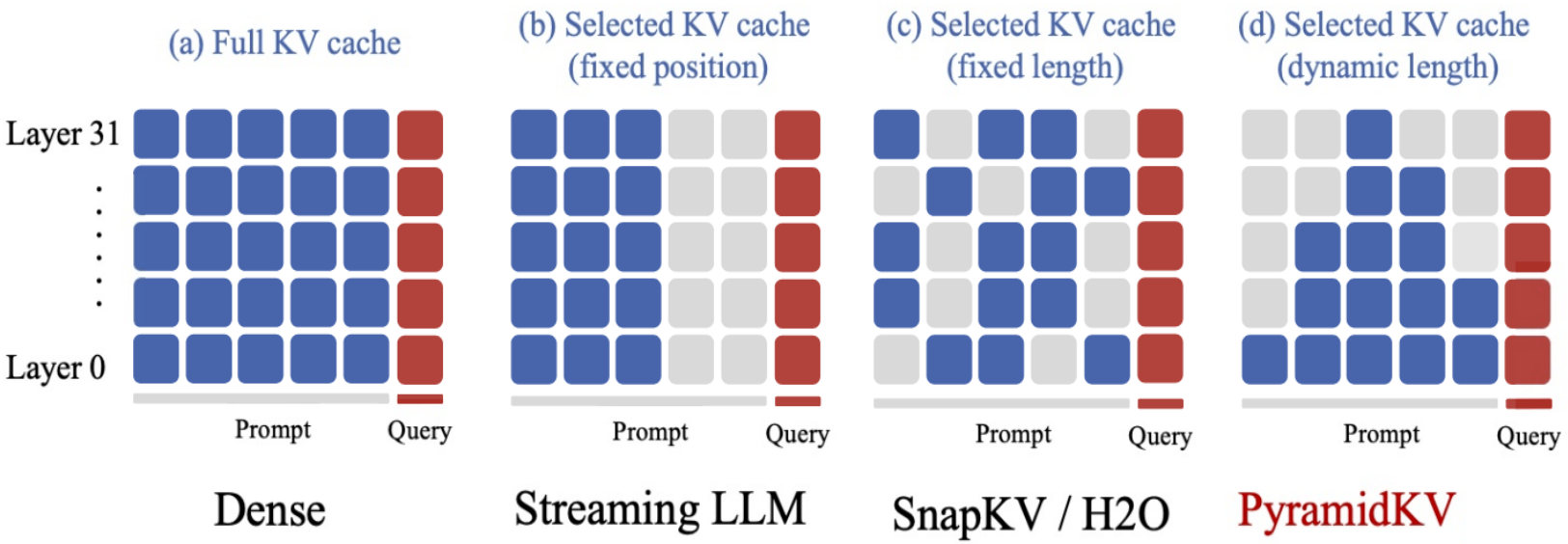


Auto regressive Decoding

1 + 4 + 9 + … +10000

Background

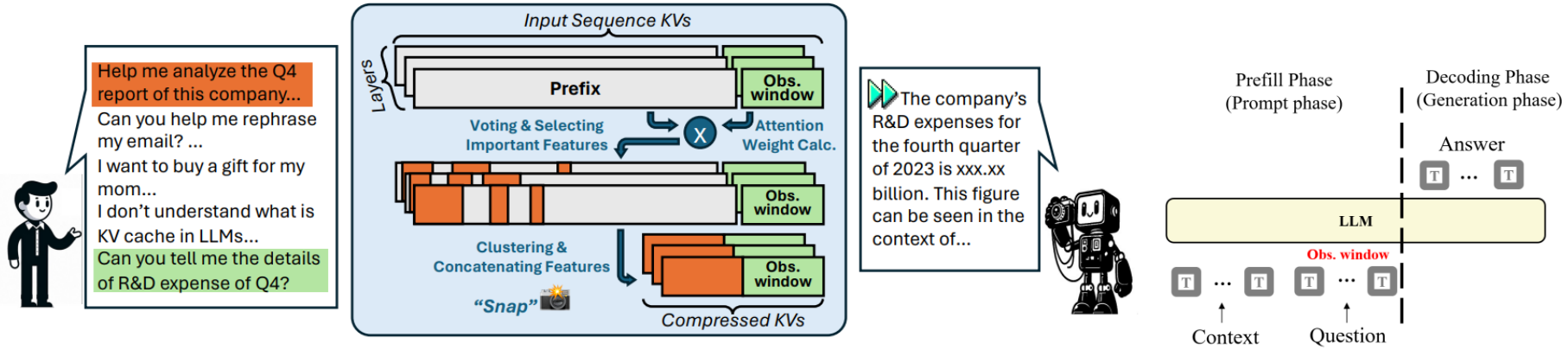
- LLM Problem
  - KV-Cache Compression
    - KV-Cache를 제거, 병합 등의 방식으로 효율적으로 압축
      - 근본적인 계산 복잡도 $O(n)$ 해결 $(n\downarrow)$
      - Cache size 감축으로 인한 저장 메모리 감소



Paper I

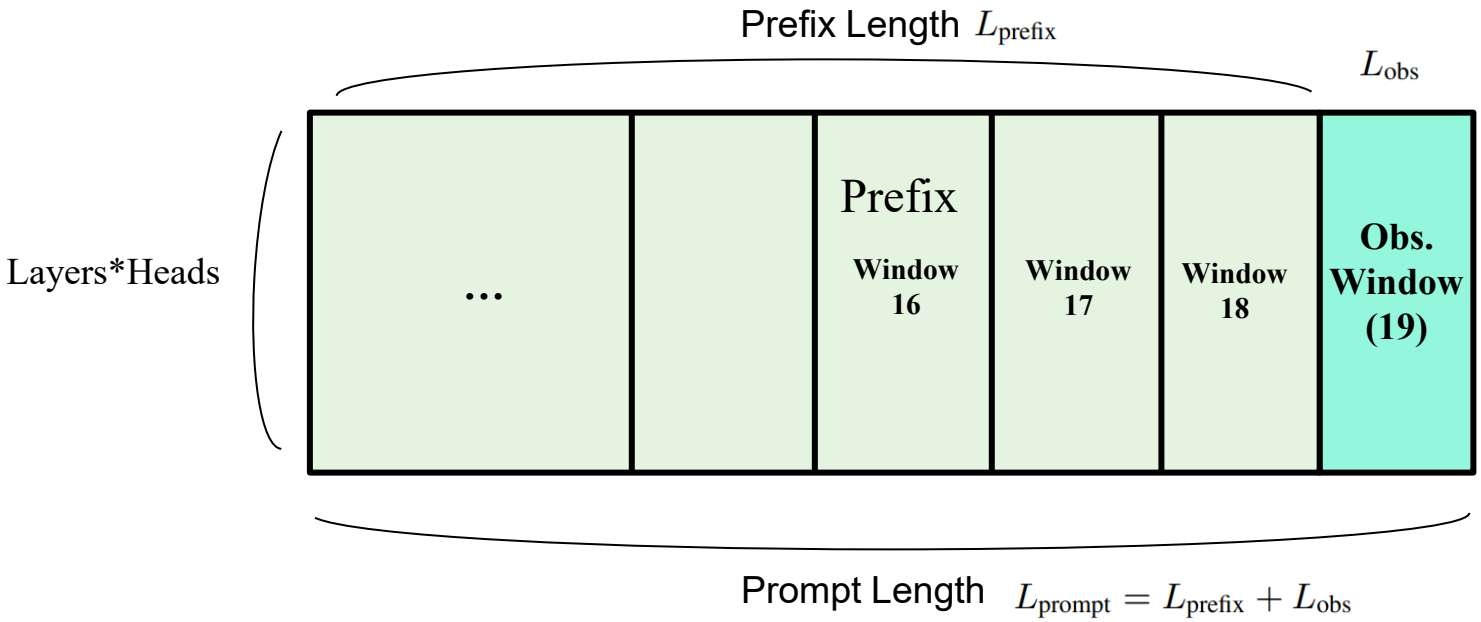
- SnapKV: LLM Knows What You are Looking for Before Generatio (NeurIPS 2024)
  - LLM에서 추가 학습 없이 KV 캐시를 효율적으로 압축
    - 핵심 논리
      - (1) 이론적으로, 어텐션 관점에서 생성 단계의 응답 토큰 쿼리에 대해 높은 어텐션을 받는 KV를 유지하는 것이 풀 KV 캐시를 사용하는 경우를 잘 근사함
      - (2) 하지만 생성 단계의 응답 토큰 쿼리를 직접 활용 하는것은 연산 측면에서 비효율적
      - (3) 생성 쿼리의 어텐션 패턴은 프리필 단계의 Obs. Window 쿼리의 어텐션 패턴과 유사함을 발견

      => 생성 단계 전 Obs. Window를 기준으로 어텐션 높은 KV 유지



Paper I

- SnapKV: LLM Knows What You are Looking for Before Generatio (NeurIPS 2024)
  - Definition
    - **Prompt**: 전체 입력
    - **Obsservation Window:** 여러 윈도우 중 마지막 윈도우
    - **Prefix:** obs. Window를 제외한 앞의 구간



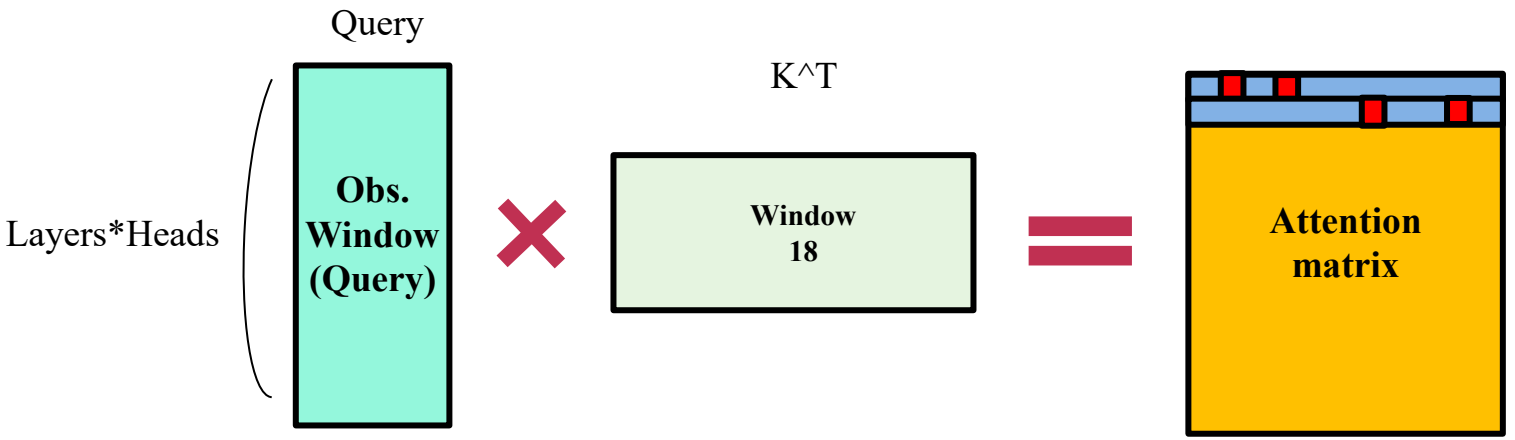
Paper I

- SnapKV: LLM Knows What You are Looking for Before Generatio (NeurIPS 2024)
  - Definition
    - Voting
      - 한 윈도우 내에서 Obs window(19)의 각 query별 어텐션 값들의 합이 가장 높은 k개의 Cache
      - 레이어별, 헤드별 계산

      => 각 윈도우 내에서 가장 obs. Window에 어텐션이 강하게 반응하는 Cache를 고름

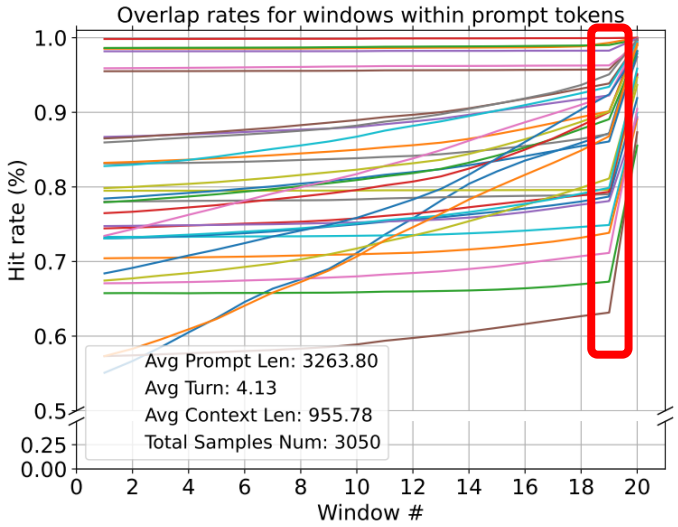
$$\mathbf{C} = \sum_{i=0}^{L_{\text{obs}}} \mathbf{W}_{\text{obs}}[:, i, :]$$

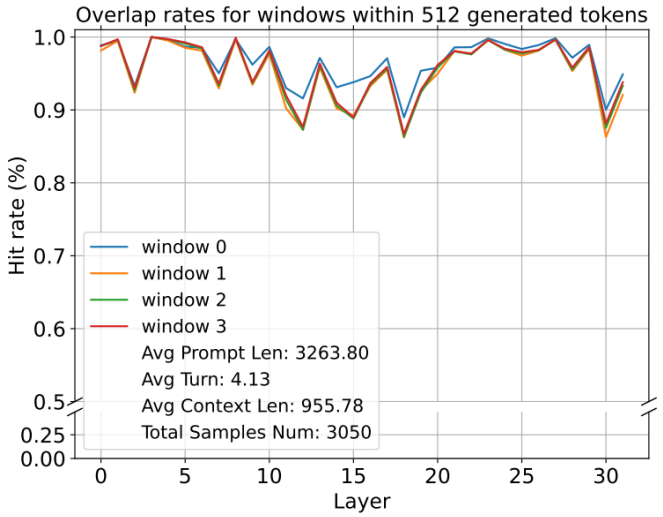
$$I = \text{Top}_k(\mathbf{C}, k)$$



Paper I

- SnapKV: LLM Knows What You are Looking for Before Generatio (NeurIPS 2024)
  - Definition
    - Hit rate $\mathcal{H}(\mathbf{M}_{\text{threshold_cur}}, \mathbf{M}_{\text{vote_obs}})$
      - 실제 응답 쿼리의 어텐션 값 상위(Upper threshold) Cache와 Voting으로 뽑힌 Cache간 중첩 비율을 의미
        - ✓헤드별로 Hit rate를 구한 후 평균
        - ✓높을수록 obs. Window의 어텐션 패턴이 응답 쿼리의 어텐션 패턴과 유사
      - (a) obs. Window가 마지막(19)일때 가장 hit rate가 높음
      - (b) 생성 토큰을 4개의 윈도우로 나눴을 때 obs. Window와의 Hit rate가 전반적으로 높은 값을 보임





Paper I

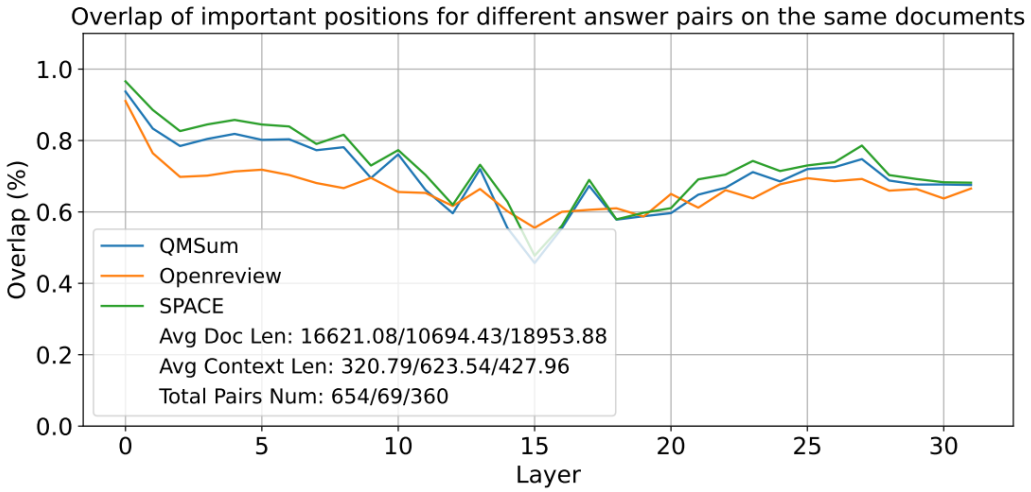
- SnapKV: LLM Knows What You are Looking for Before Generatio (NeurIPS 2024)
  - Attention pattern robostness
    - (1) Does the nature of instructions in the prompt affect the hit rate?
      - 마지막 윈도우의 문장에 따른 어텐션 패턴 변동성
    - (2) Does the context and instruction positioning affect the hit rate?
      - 지시문 위치에 따른 어텐션 패턴 변동성

Paper I

- SnapKV: LLM Knows What You are Looking for Before Generatio (NeurIPS 2024)
  - Attention pattern robostness
    - (1) Does the nature of instructions in the prompt affect the hit rate?

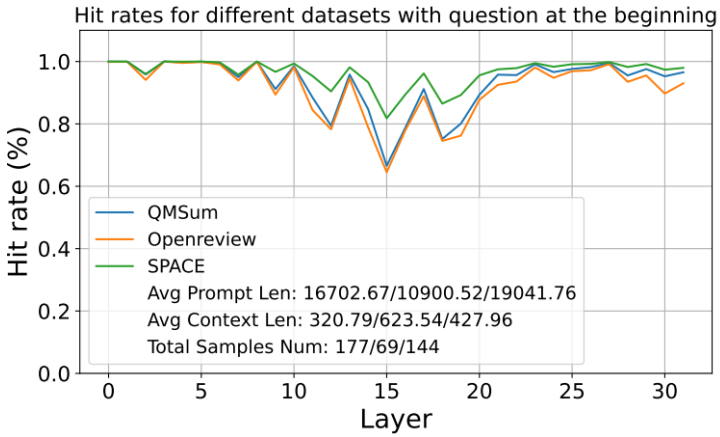
$$H(M_{\text{vote}_A}, M_{\text{vote}_B})$$

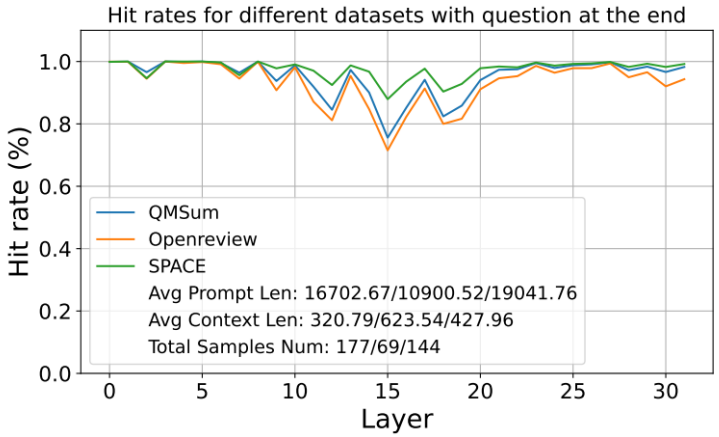
      - 마지막 윈도우의 문장(지시문)에 따른 어텐션 패턴 변동성
        - ✓제공된 컨텍스트가 동일하더라도 지시문이 중요한 특징 선택에 영향을 미치는지를 분석
        - ✓Prepix + A지시문, Prepix + B지시문 간의 Hit rate가 높지 않음
          - 같은 Prefix여도 마지막 지시문에 따라 뽑히는 context가 다름



Paper I

- SnapKV: LLM Knows What You are Looking for Before Generatio (NeurIPS 2024)
  - Attention pattern robostness
    - (2) Does the context and instruction positioning affect the hit rate? $\mathcal{H}(\mathbf{M}_{\text{threshold_cur}}, \mathbf{M}_{\text{vote_obs}})$
      - 지시문 위치에 따른 어텐션 패턴 변동성
        - ✓원래 지시문의 위치를 뒤에서 앞으로 바꿔도 Hit rate 경향이 달라지지 않음
        - ✓지시문 위치와 관계없이 obs. Window(19) 생성 토큰 간의 어텐션 패턴 유사성이 일관되게 유지됨





Paper I

- SnapKV: LLM Knows What You are Looking for Before Generatio (NeurIPS 2024)
  - Method
    - (1) Voting을 통해 각 cache별 score를 계산
    - (2) Voting score를 1D pooling
      - 언어에서 중요한 정보는 보통 단어 하나가 아니라 최소 구 단위이므로 주변 값을 함께 고려
      - Ex) max pooling(size 3) [0.2, 0.3, 0.5, 0.1, 0.7] => [0.3, 0.5, 0.5, 0.7, 0.7]
    - (3) 점수 재산정(pooling) 후 각 윈도우 내 top-k에 해당되는 cache들을 보존
      - Obs. Window cahce들은 전원 보존

Paper I

- SnapKV: LLM Knows What You are Looking for Before Generatio (NeurIPS 2024)
  - Experiments

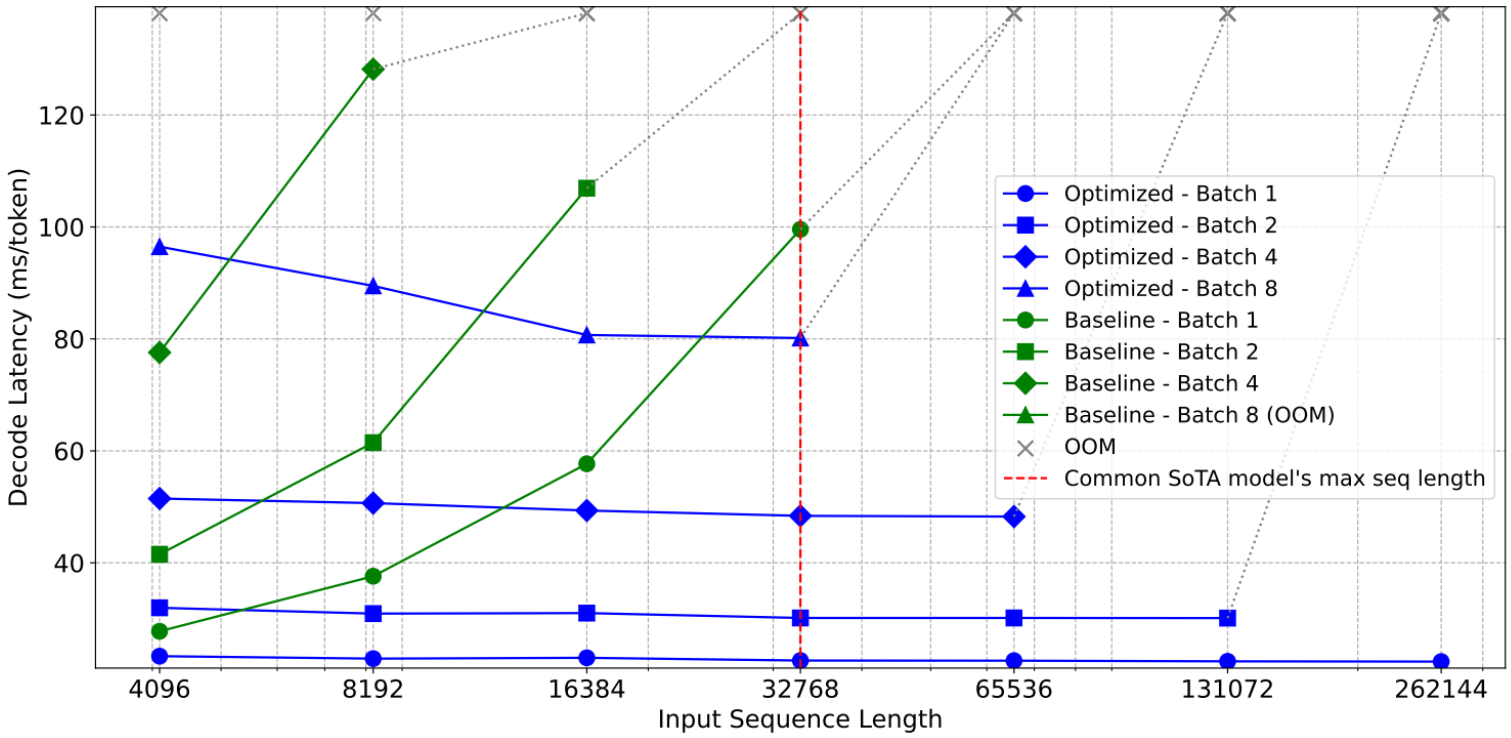
Table 1: Performance comparison of SnapKV and H2O across various LLMs on LongBench.

		Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code	
	LLMs *	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PRe	Lcc	RB-P
LWMChat	All KV	**18.18**	**25.56**	40.94	24.57	19.39	10.49	**27.97**	24.9	**24.81**	71.0	60.9	39.73	3.17	3.5	44.4	43.82
	SnapKV: 1024	18.02	23.73	40.25	24.61	**19.84**	10.77	19.79	24.44	23.53	70.0	**61.42**	39.64	1.67	3.0	43.34	44.0
	SnapKV: 2048	17.92	25.03	**41.38**	24.49	19.38	**11.34**	21.6	24.22	24.36	70.0	61.11	39.91	2.17	4.0	44.46	**44.92**
	SnapKV: 4096	17.92	25.47	40.76	**24.92**	19.53	11.27	25.34	**25.42**	24.58	70.5	61.08	39.62	**3.17**	**4.0**	**44.49**	44.08
	H2O: 4096	13.17	24.82	20.01	16.86	9.74	7.2	25.77	23.26	23.83	**71.0**	61.06	**40.33**	0.0	0.0	41.52	40.97
LongChat	All KV	**20.88**	**29.36**	**43.2**	33.05	24.58	**14.66**	**30.89**	22.76	**26.61**	**66.5**	**83.99**	**40.83**	0.0	30.5	54.89	**59.05**
	SnapKV: 1024	19.32	26.6	37.93	34.15	23.34	12.71	23.45	21.81	24.93	65.0	80.88	38.19	0.0	31.0	53.63	57.62
	SnapKV: 2048	19.28	28.81	40.26	**35.31**	23.75	13.44	26.3	22.29	25.73	66.0	79.93	39.59	0.0	**31.0**	**56.05**	58.61
	SnapKV: 4096	20.68	29.34	42.21	33.95	**24.88**	14.15	28.55	**23.11**	26.45	66.0	**81.25**	**40.52**	0.0	29.5	54.79	58.81
	H2O: 4096	19.31	28.3	37.75	30.51	23.06	11.76	27.55	21.37	26.49	66.0	75.8	39.92	0.0	25.5	53.56	55.53
Mistral	All KV	**26.82**	33.06	49.28	**42.77**	27.33	19.27	**32.85**	24.25	27.06	71.0	86.23	42.98	2.75	86.98	55.51	**52.88**
	SnapKV: 1024	25.54	29.51	49.25	40.94	25.7	**19.42**	25.89	23.82	26.11	69.5	**86.48**	42.06	2.98	**88.56**	55.65	51.87
	SnapKV: 2048	25.89	32.47	48.6	41.71	27.31	18.69	28.81	**24.5**	26.6	70.0	86.27	42.47	3.09	87.43	**55.93**	52.01
	SnapKV: 4096	26.41	**33.36**	**49.81**	42.32	**27.93**	18.76	30.74	24.19	**27.08**	**71.0**	86.25	**43.01**	2.73	86.18	55.62	52.65
	H2O: 4096	22.61	29.06	47.22	36.54	20.6	16.25	30.0	23.8	26.75	70.5	86.16	42.97	**3.46**	86.38	53.72	51.1
Mixtral	All KV	26.81	**37.06**	51.55	47.77	32.46	**26.59**	**34.25**	**26.05**	27.91	76.0	90.57	46.98	5.5	100.0	**69.07**	69.65
	SnapKV: 1024	26.01	34.65	51.58	**48.23**	32.67	25.92	27.77	25.0	27.25	74.5	90.42	46.48	5.5	99.5	69.02	68.98
	SnapKV: 2048	**27.12**	36.9	51.91	47.46	33.23	26.27	30.19	25.84	27.8	**76.0**	90.24	46.31	5.5	100.0	68.72	**70.01**
	SnapKV: 4096	26.46	37.03	**52.62**	47.71	**33.35**	26.45	32.64	25.87	**27.94**	75.5	**90.71**	**47.14**	5.5	**100.0**	68.81	69.56
	H2O: 4096	20.45	32.09	48.02	34.76	25.69	16.5	29.76	23.53	26.84	74.5	90.24	47.1	**7.06**	99.42	64.91	63.52

* Credit to Jin et al. [20] for the template used in the table.

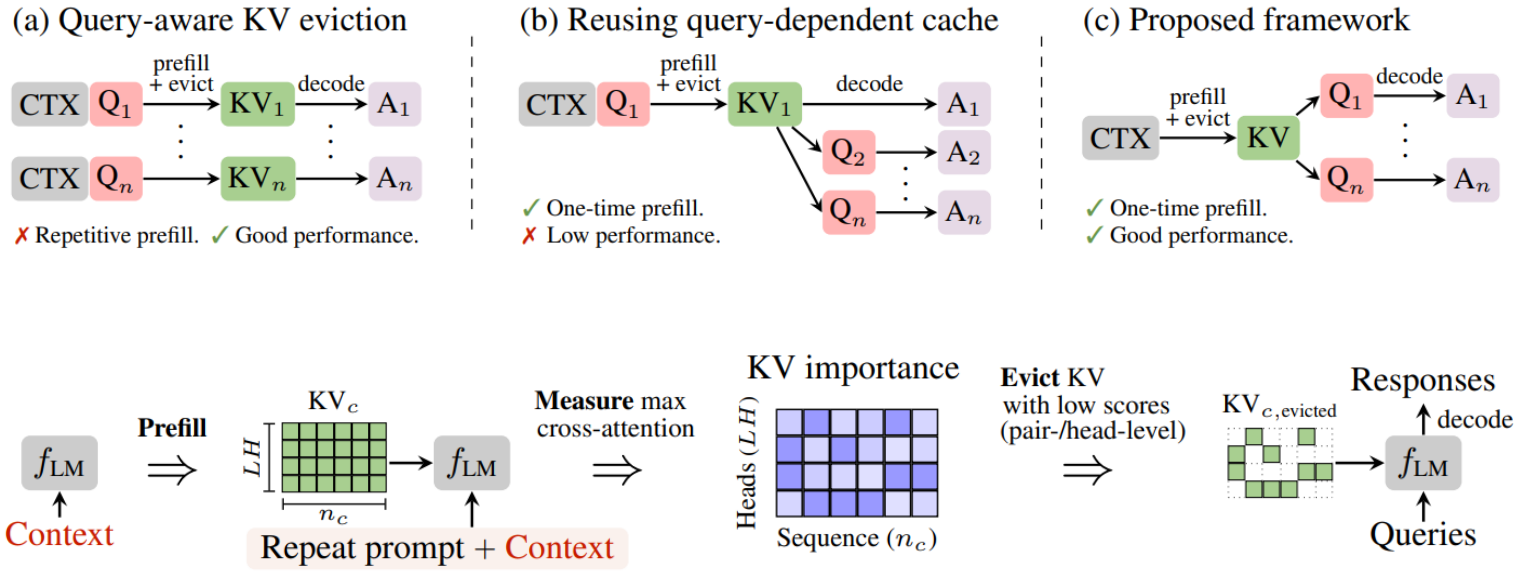
Paper I

- SnapKV: LLM Knows What You are Looking for Before Generatio (NeurIPS 2024)
  - Efficieny anlasiis
    - 배치 크기와 입력 토큰 수가 증가하더라도, KV 캐시를 압축함으로써 OOM 없이 안정적인 추론이 가능
      - 디코딩 지연(latency) 또한 크게 감소



Paper II

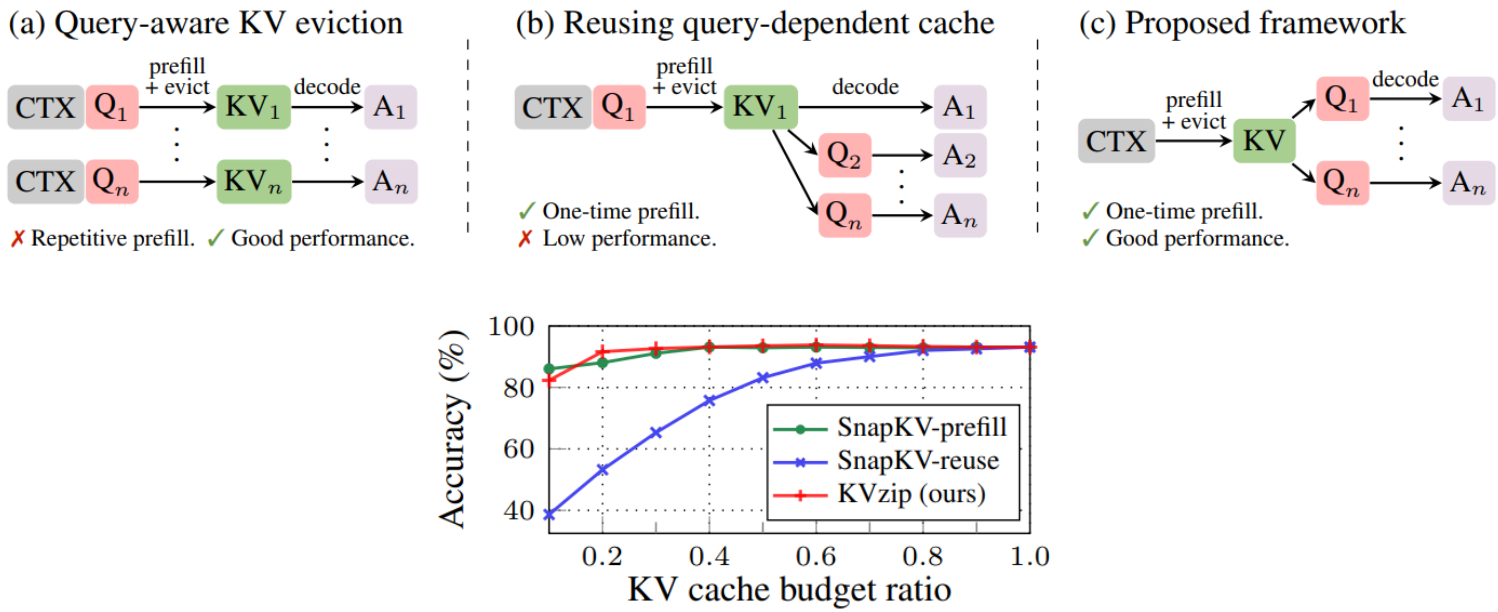
- KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction
  - 기존 query-aware 접근법의 다중 쿼리 환경에서의 비효율성 문제 제기
    - Query와 무관하게 재사용 가능한 압축 KV cache를 생성하는 query-agnostic 방식을 제안
    - KV Cache를 재구성하는 과정에서 중요한 캐시들을 보존



Paper II

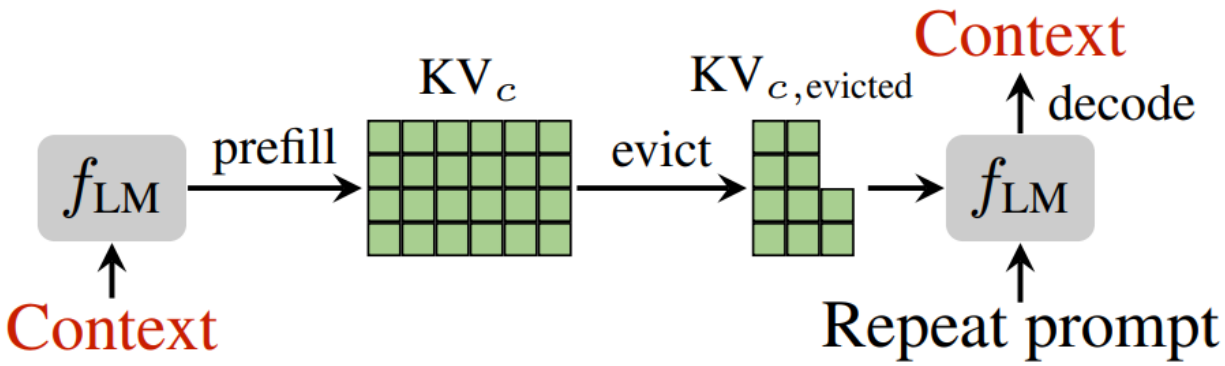
- KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction
  - Query-agnostic motivation
    - (a) 기존 쿼리에 맞게 캐시를 압축하면 쿼리가 들어올때마다 새로 압축해야함
    - (b) 특정 쿼리(q_1)에 최적화된 캐시를 다른 쿼리(q_2)에 재사용할 경우 성능 저하가 발생
    - (c) 제안 방식은 쿼리와 독립적으로 context를 압축하여, 서로 다른 쿼리 간 재사용 시에도 높은 성능을 유지함

$$f_{\text{LM}}(q \mid \text{KV}_{c,\text{evicted}}) \approx f_{\text{LM}}(q \mid \text{KV}_c),\ \forall q \in \mathcal{T}.$$



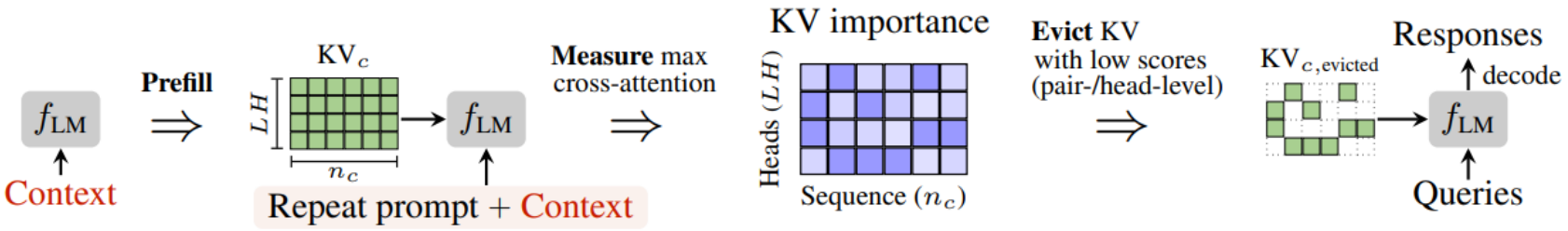
Paper II

- KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction
  - Intuition
    - 압축된 캐시에 대한 가설
      - 만약 원래의 Cache 정보를 잘 보전하였다면 “Repeat the previous context:”라는 프롬포트만으로도 원래의 Context를 완벽히 재구성 할 것임
        - ✓압축된 KV 캐시의 완전성은 repeat prompt를 통해 원본 컨텍스트 재구성 가능 여부로 검증



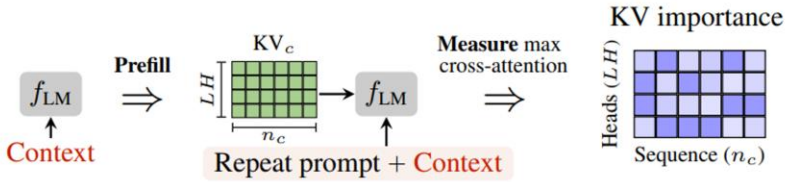
Paper II

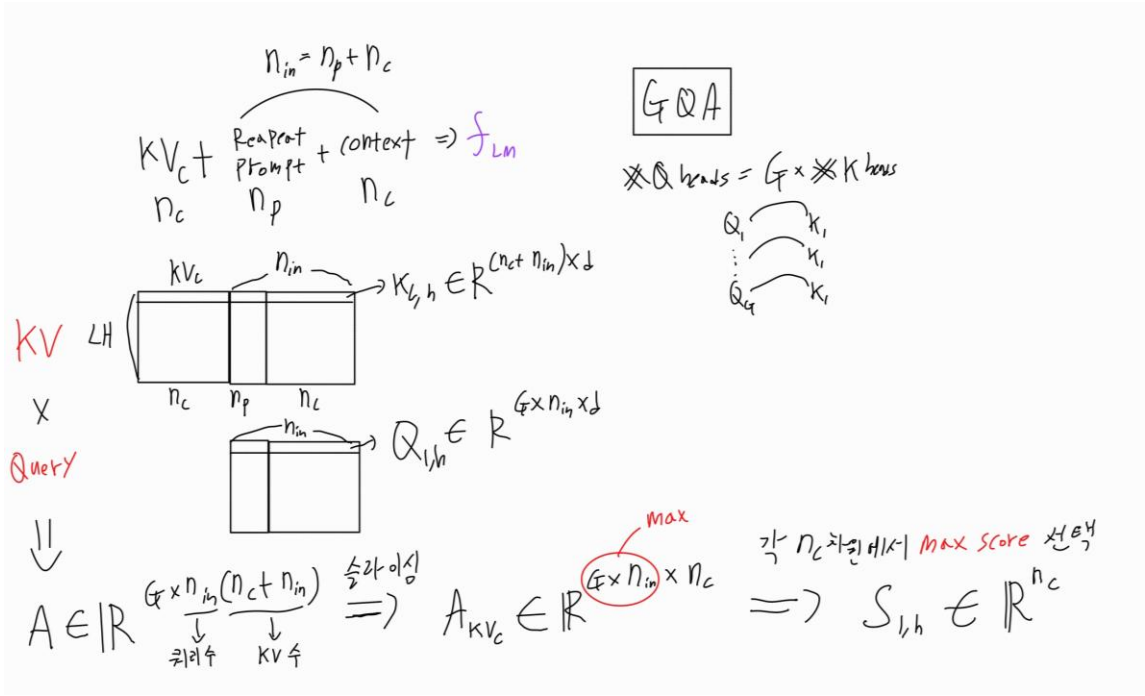
- KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction
  - Main idea
    - (1) Context를 정확히 재구성할 수 있도록 하는 KV 캐시를 보존
      - Context 를 정확히 재생성 할수록 압축된 캐시는 원래의 캐시의 정보를 가장 효과적으로 압축 한 것
    - (2) 재구성 단계에서 응답 토큰의 쿼리에 대해 높은 어텐션 가중치를 갖는 토큰들을 보존
      - SnapKV의 전제 (1)와 동일
    - (3) 재구성 단계에서 높은 어텐션을 받는 토큰들이 다른 다운스트림 태스크에서도 유사하게 높은 어텐션을 받는다는 점을 실험적으로 관찰하여, 제안 방법의 타당성을 입증
      - SnapKV의 전제 (3)의 논리와 유사



Paper II

- KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction
  - KV importance Scoring
    - Context를 Supervision으로 Repeat prompt와 함께 입력
    - n_{in}이 쿼리일때 해당 Cache가 받는 최대 어텐션 값을 스코어로 사용





Paper II

- KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction
  - Observation
    - (a) 초기 KV cache 생성 시의 self-attention 분포에 비해, 재구성 단계에서 KV cache가 받는 어텐션 스코어는 더 희소한 분포를 보임
    - (b) 재구성 과정에서 높은 스코어를 받은 토큰들은 다양한 다운스트림 태스크에서도 일관되게 높은 어텐션을 받는 경향을 보보임
      - (c) 서로 다른 QA 태스크 간에는 이러한 일관성이 감소하며 상반된 어텐션 경향이 관찰됨

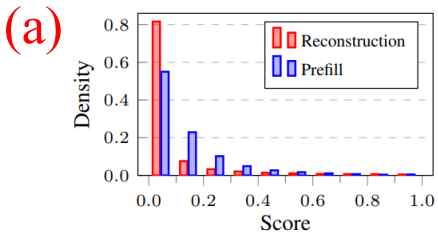
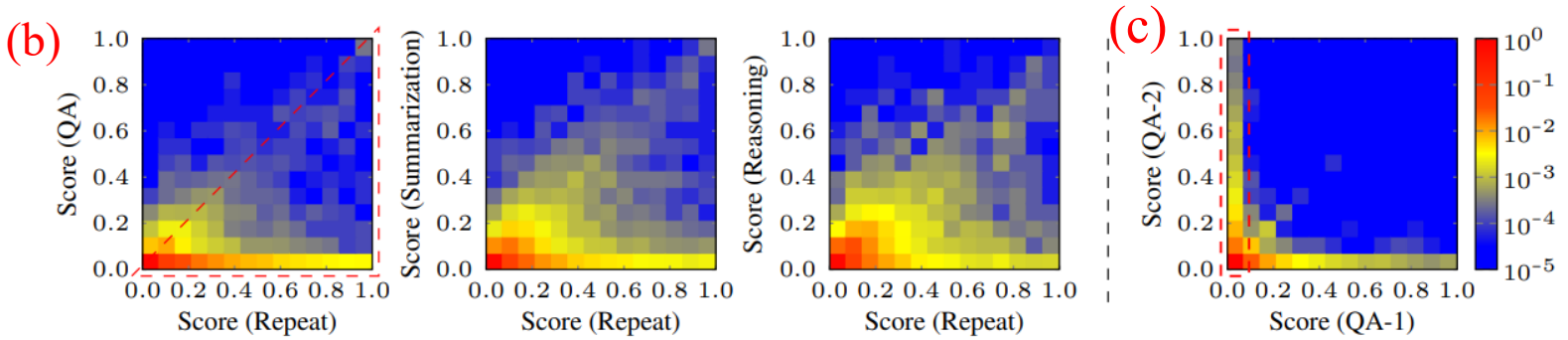
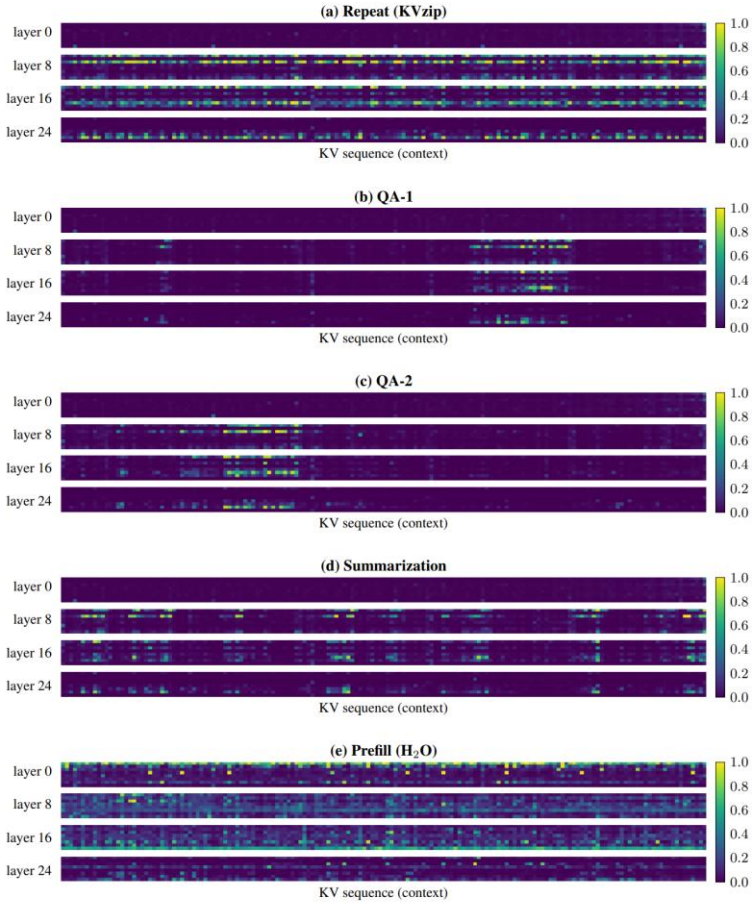


Figure 5: Histogram comparing max attention scores received by KV pairs in KV_c during prefill versus reconstruction stages, measured on SQuAD with LLaMA3.1-8B.



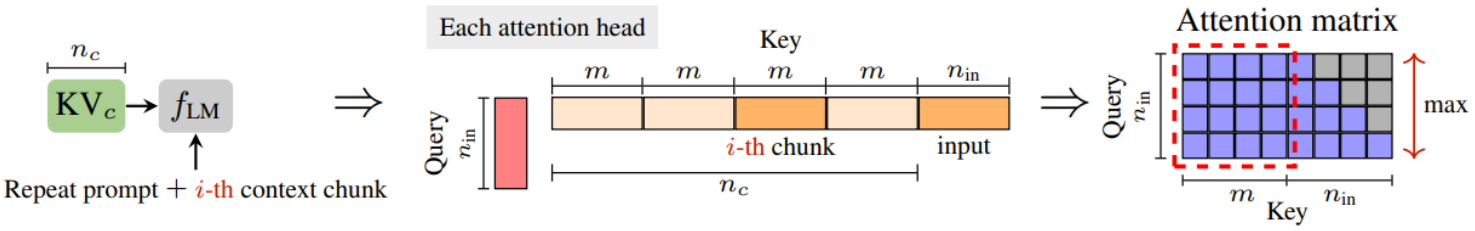
Paper II

- KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction
  - Observation
    - (a) Repeat에서 스코어 패턴은 다운스트림 태스크 패턴 (b)~(d)를 모두 포함
    - (e) Prefill 어텐션은 분산되어 포함하지 못함
      - 프리필 단계에서의 중요도 추정 기법인 H2O보다 효과적임을 입증



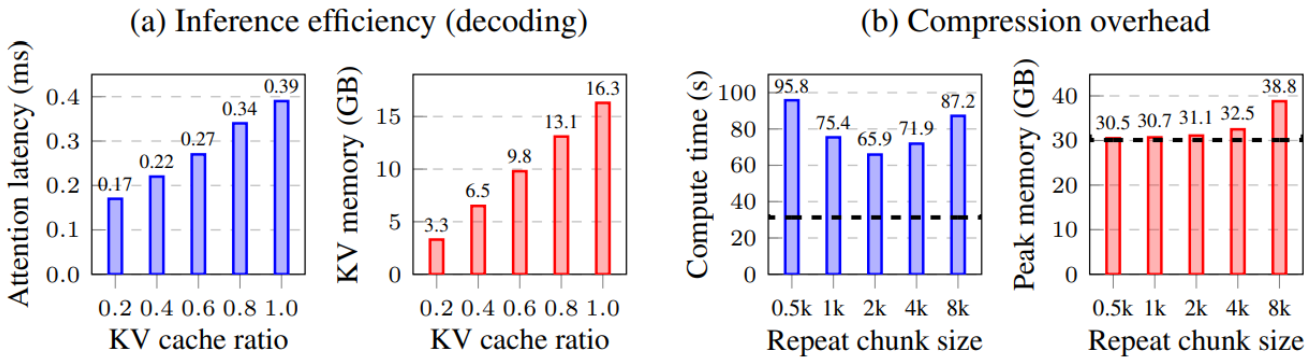
Paper II

- KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction
  - Technical Challenge and Solution
    - 앞선 방식대로 KV Cache scoreing을 하면 연산 비용이 많이 듬 $O((2n_c + n_{prompt})^2)$
      - Chunk 단위 Scoring으로 연산을 획기적으로 줄임 $O(mn_C)$ m: $chunk\ size(= 2k)$
      - 기존 supervision context를 한번에 넣는것이아닌 context를 chunk 단위로 분할하여 각 chunk에 대응하는 KV cache에 대해서만 어텐션을 수행
        - ✓이를 Chunk 수 만큼 반복



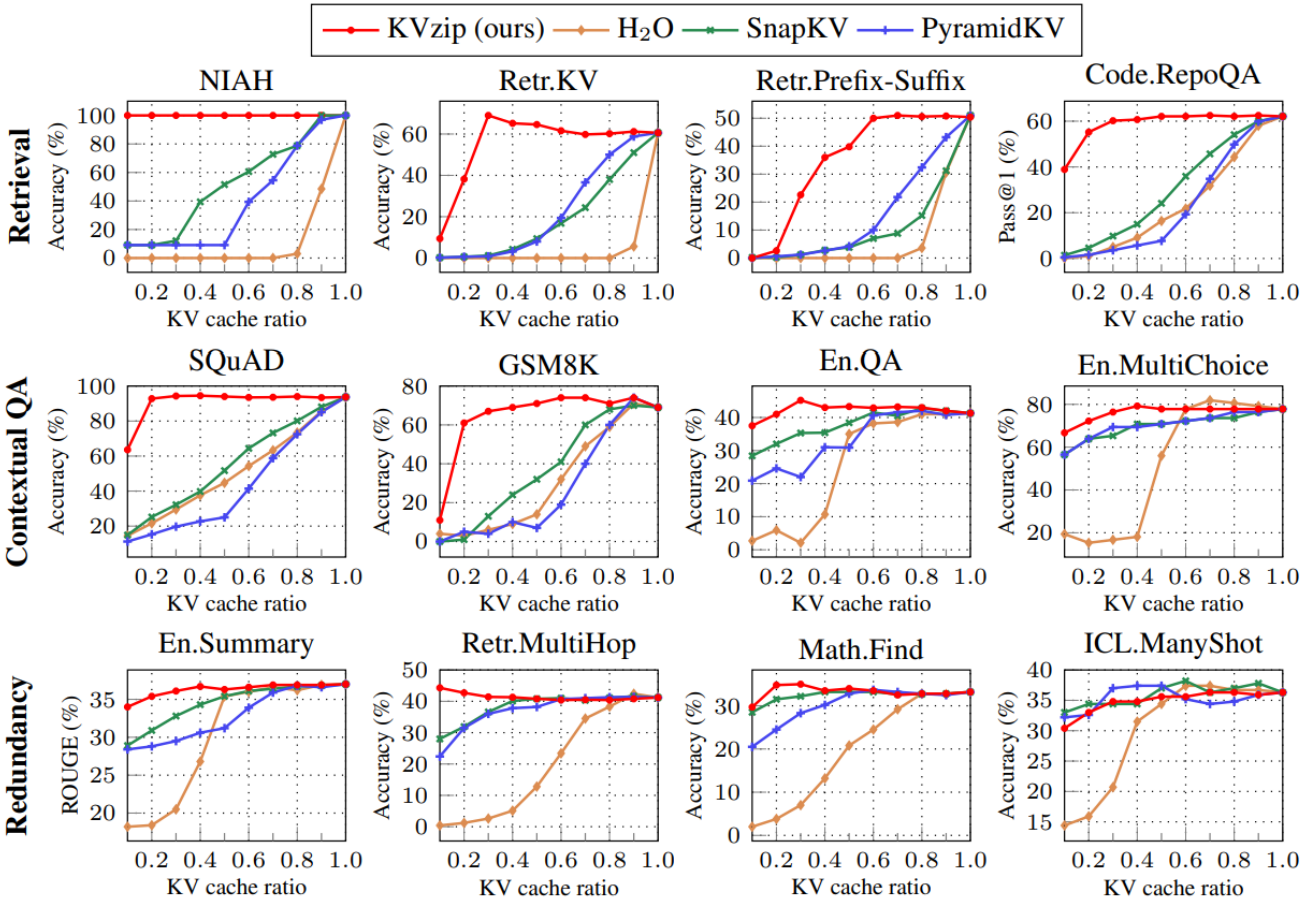
Paper II

- KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction
  - Technical Challenge and Solution
    - (a) 연산 시간 및 메모리 감축
    - (b) 총 압축 비용은 prefill 대비 약 2배이며, 메모리 증가는 2% 미만
      - Chunk size에 따라 overhead가 상이함



Paper II

- KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction
  - Experiments



Paper II

- KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction
  - Experiments

