Hybrid Embedding-to-Generation Architecture

2025. 08. 21.

윤정호

Natural Language Processing & Artificial Intelligence

KOREA UNIVERSITY 고려대학교

Intro

***현 RAG의 문제점**

- Long Context 문제
  - LLM 학습으로 굳어진 Context length에 고정

- Inference Cost
  - RAG는 Transformer 구조에서 검색된 Doc 수에 따라 2차적으로 증가

- Document 저장
  - Retrieve 후 가져올 Doc를 저장 및 불러와야 함

- Embedding vector의 1회성
  - Retrieve 후 Embedding Vector는 사용될 일이 없음

선행 연구

Generative Representational Instruction Tuning

Niklas Muennighoff [c] Hongjin Su [h] Liang Wang [m] Nan Yang [m]
Furu Wei [m] Tao Yu [h] Amanpreet Singh [c] Douwe Kiela [c]
[c] Contextual AI [h] The University of Hong Kong [m] Microsoft Corporation
niklas@contextual.ai

Generation + Embedding Model

ICLR 2025 Poster

Context Embeddings for Efficient Answer Generation in RAG

David Rau*
University of Amsterdam
Amsterdam, Netherlands
d.m.rau@uva.nl

Shuai Wang*†
The University of Queensland
Brisbane, Australia
shuai.wang2@uq.edu.au

Hervé Déjean
Naver Labs Europe
Grenoble, France
herve.dejean@naverlabs.com

Stéphane Clinchant
Naver Labs Europe
Grenoble, France
stephane.clinchant@naverlabs.com

WSDM 2025

PISCO: Pretty Simple Compression for Retrieval-Augmented Generation

Maxime LOUIS
Naver Labs Europe
maxime.louis@naverlabs.com

Hervé Dejean
Naver Labs Europe

Stéphane Clinchant
Naver Labs Europe

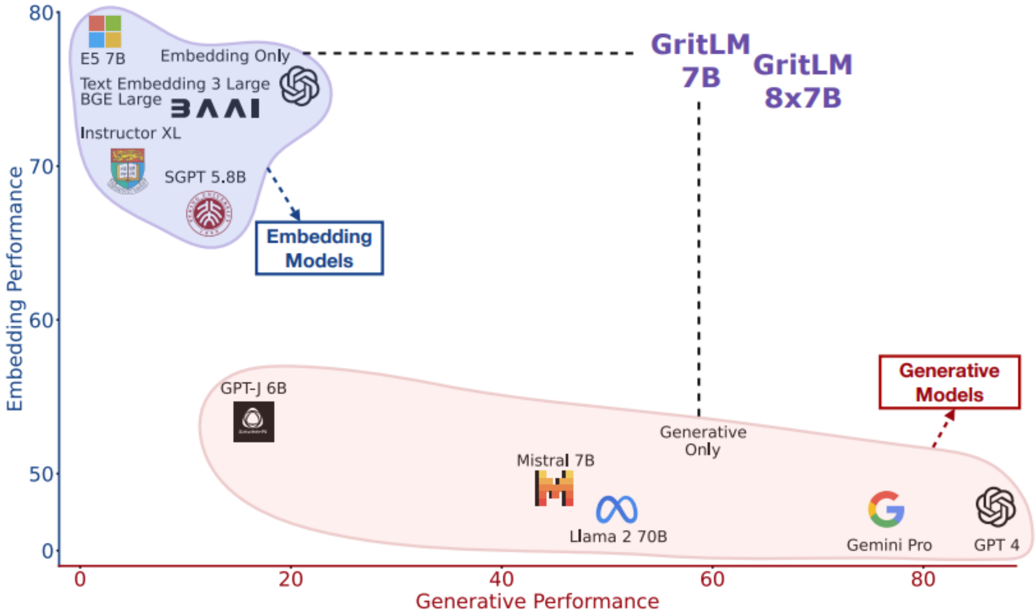
ACL 2025 findings

Context Embedding를 활용한 Context Length 감소

1. Generative Representational Instruction Tuning

***Abstract & Introduction**

- 기존 Model들은 Embedding, Generation 따로 진행
- Generation 모델에 Representation task를 진행하려면 fine-tuning이 필요하고, Generative 능력 상실



⇒ Generative Instruction tuning과 Representational instruction tuning을 동시에 진행하여 두 task 모두 성능을 높임

⇒ Query, Document 모두 Retrieve를 위해 Embedding 생성할 때 모델을 지나가기에 cache를 저장할 수 있고, 이를 통해 추론 속도를 높일 수 있음

METHODOLOGY

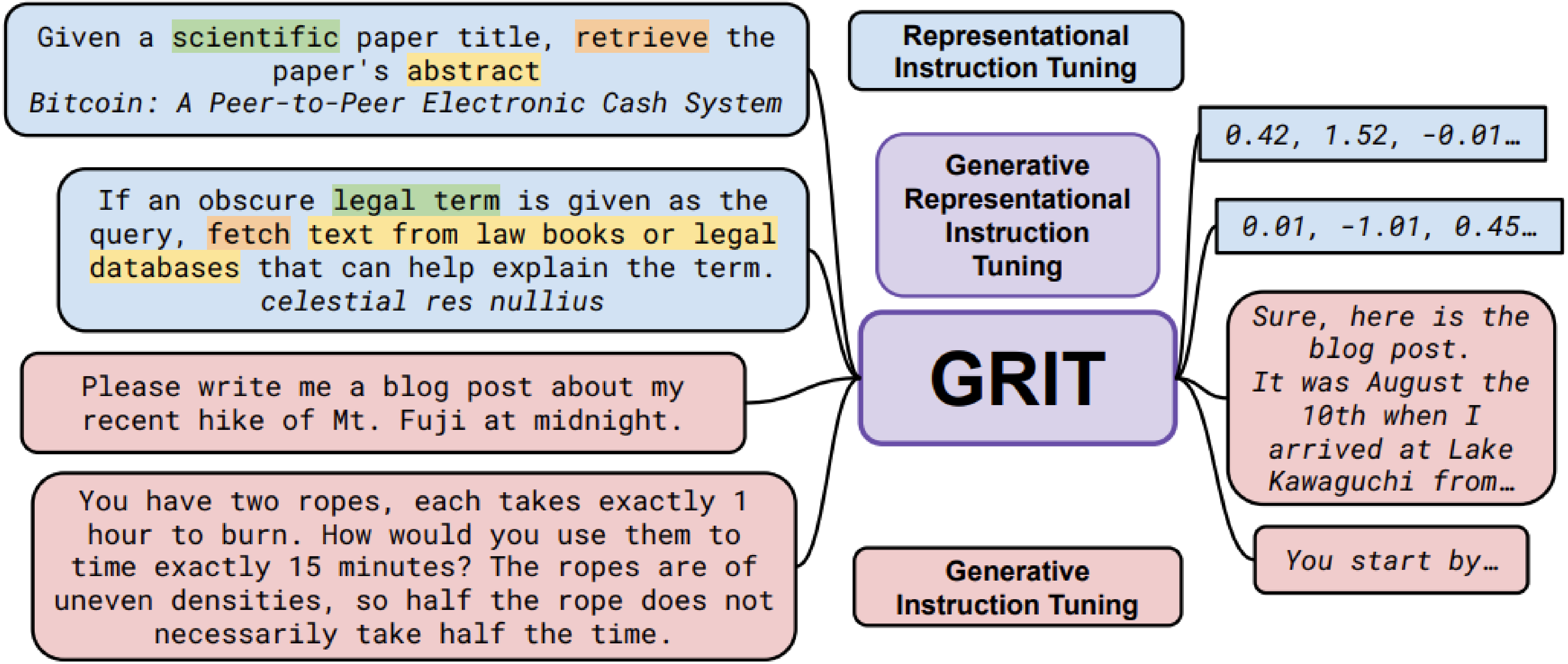
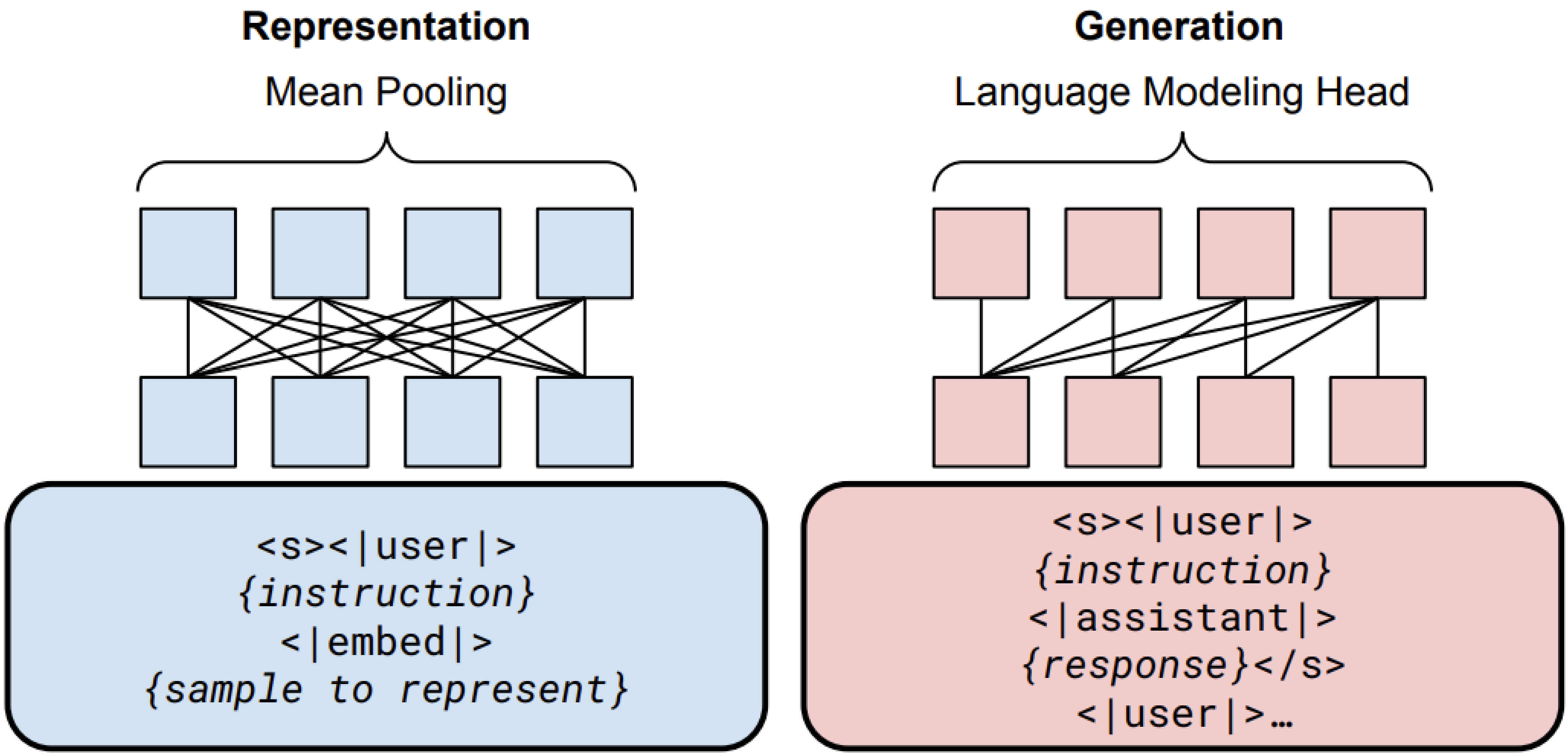


Figure 2: **GRIT.** The same model handles both text representation and generation tasks based on the given instruction. For representation tasks, instructions ideally contain the target domain, intent, and unit [5]. The representation is a tensor of numbers, while the generative output is text.

METHODOLOGY



METHODOLOGY

***GRIT Training**

$$\mathcal{L}_{\text{Rep}} = -\frac{1}{M}\sum_{i=1}^{M} \log \frac{\exp(\tau \cdot \sigma(f_\theta(q^{(i)}), f_\theta(d^{(i)})))}{\sum_{j=1}^{M} \exp(\tau \cdot \sigma(f_\theta(q^{(i)}), f_\theta(d^{(j)})))}$$

- E5 Dataset

- 기존 embedding 모델 Loss와 동일

$$\mathcal{L}_{\text{Gen}} = -\frac{1}{N}\sum_{i=1}^{N} \log P(f_{\theta,\eta}(x^{(i)}) | f_{\theta,\eta}(x^{(<i)}))$$

- Tulu 2 Dataset

- 기존 Generative 모델 Loss와 동일

$$\mathcal{L}_{\text{GRIT}} = \lambda_{\text{Rep}}\mathcal{L}_{\text{Rep}} + \lambda_{\text{Gen}}\mathcal{L}_{\text{Gen}}$$

EXPERIMENTS

*Embedding Performance

Task (→) Metric (→) Dataset # (→)	CLF Acc. 12	Clust. V-Meas. 11	PairCLF AP 3	Rerank MAP 4	Retrieval nDCG 15	STS Spear. 10	Summ. Spear. 1	Avg. 56
				Proprietary models♥				
OpenAI v3	75.5	49.0	85.7	59.2	55.4	81.7	29.9	64.6
				Other Open Models♥				
Llama 2 70B	60.4	29.0	47.1	38.5	9.0	49.1	26.1	35.6
Mistral 7B	63.5	34.6	53.5	43.2	13.2	57.4	19.7	40.5
Mistral 7B Instruct	67.1	34.6	59.6	44.8	16.3	63.4	25.9	43.7
GPT-J 6B	66.2	39.0	60.6	48.9	19.8	60.9	26.3	45.2
SGPT BE 5.8B	68.1	40.3	82.0	56.6	50.3	78.1	31.5	58.9
Instructor XL 1.5B	73.1	44.7	86.6	57.3	49.3	83.1	**32.3**	61.8
BGE Large 0.34B	76.0	46.1	87.1	60.0	54.3	83.1	31.6	64.2
E5 Mistral 7B	78.5	50.3	**88.3**	60.2	56.9	**84.6**	31.4	66.6
				GRITLM				
Gen.-only 7B	65.4	32.7	54.2	43.0	13.7	60.2	21.1	41.2
Emb.-only 7B	78.8	**51.1**	87.1	**60.7**	**57.5**	83.8	30.2	**66.8**
GRITLM 7B	**79.5**	50.6	87.2	60.5	57.4	83.4	30.4	**66.8**
GRITLM 8X7B	78.5	50.1	85.0	59.8	55.1	83.3	29.8	65.7

EXPERIMENTS

*Generative Performance

Dataset (→) Setup (→) Metric (→)	MMLU 0 FS EM	GSM8K 8 FS, CoT EM	BBH 3 FS, CoT EM	TyDi QA 1 FS, GP F1	HumanEval 0 FS pass@1	Alpaca 0 FS, 1.0 % Win	Avg.
			Proprietary models♥				
GPT-4-0613	81.4	95.0	89.1	65.2	86.6†	91.2	84.8
			Other Open Models♥				
GPT-J 6B	27.7	2.5	30.2	9.4	9.8	0.0	13.3
SGPT BE 5.8B	24.4	1.0	0.0	22.8	0.0	0.0	8.0
Zephyr 7B β	58.6	28.0	44.9	23.7	28.5	85.8	44.9
Llama 2 7B	41.8	12.0	39.3	51.2	12.8◆	0.0	26.2
Llama 2 13B	52.0	25.0	48.9	56.5	18.3◆	0.0	33.5
Llama 2 70B	64.5	55.5	66.0	**62.6**	29.9◆	0.0	46.4
Llama 2 Chat 13B	53.2	9.0	40.3	32.1	19.6†	91.4	40.9
Llama 2 Chat 70B	60.9	59.0	49.0	44.4	34.3†	94.5	57.0
Tülu 2 7B	50.4	34.0	48.5	46.4	24.5†	73.9	46.3
Tülu 2 13B	55.4	46.0	49.5	53.2	31.4	78.9	52.4
Tülu 2 70B	67.3	**73.0**	68.4	53.6	41.6	86.6	65.1
Mistral 7B	60.1	44.5	55.6	55.8	30.5	0.0	41.1
Mistral 7B Instruct	53.0	36.0	38.5	27.8	34.0	75.3	44.1
Mixtral 8x7B Instruct	**68.4**	65.0	55.9	24.3	**53.5**	**94.8**	60.3
			GRITLM				
Emb.-only 7B	23.5	1.0	0.0	21.0	0.0	0.0	7.6
Gen.-only 7B	57.5	52.0	55.4	56.6	34.5	75.4	55.2
GRITLM 7B	57.6	57.5	54.8	55.4	32.8	74.8	55.5
GRITLM 8X7B	66.7	61.5	**70.2**	58.2	53.4	84.0	**65.7**

EXPERIMENTS

*Analysis

MTEB DS (↓)	No Rerank	Rerank top 10
ArguAna	63.24	**64.39**
ClimateFEVER	30.91	**31.85**
CQADupstack	49.42	**50.05**
DBPedia	46.60	**47.82**
FiQA2018	59.95	**60.39**
FEVER	82.74	**82.85**
HotpotQA	79.40	**80.46**
NFCorpus	40.89	**41.23**
NQ	70.30	**71.49**
MSMARCO	41.96	**42.47**
QuoraRetrieval	**89.47**	88.67
SCIDOCS	24.41	**24.54**
SciFact	79.17	**79.28**
TRECCOVID	74.80	**75.24**
Touche2020	27.93	**28.41**
Average	57.4	**57.9**

Reranking with GRITLM

Train DS (→)	E5S		MEDI2	
MTEB DS (↓)	0 FS	1 FS	0 FS	1 FS
Banking77	**88.5**	88.3	**88.1**	87.9
Emotion	**52.8**	51.0	**52.5**	51.9
IMDB	**95.0**	93.9	**94.3**	92.2
BiorxivS2S	**39.8**	39.4	**37.6**	37.4
SprintDup.	93.0	**94.9**	95.2	**95.7**
TwitterSem	**81.1**	77.9	**76.8**	73.9
TwitterURL	**87.4**	87.1	85.9	**86.1**
ArguAna	**63.2**	51.7	**53.5**	53.2
SCIDOCS	**24.4**	19.7	**25.5**	25.5
AskUbuntu	**67.3**	64.7	**66.6**	66.0
STS12	77.3	**78.0**	**76.6**	73.5
SummEval	**30.4**	29.5	29.1	**31.5**

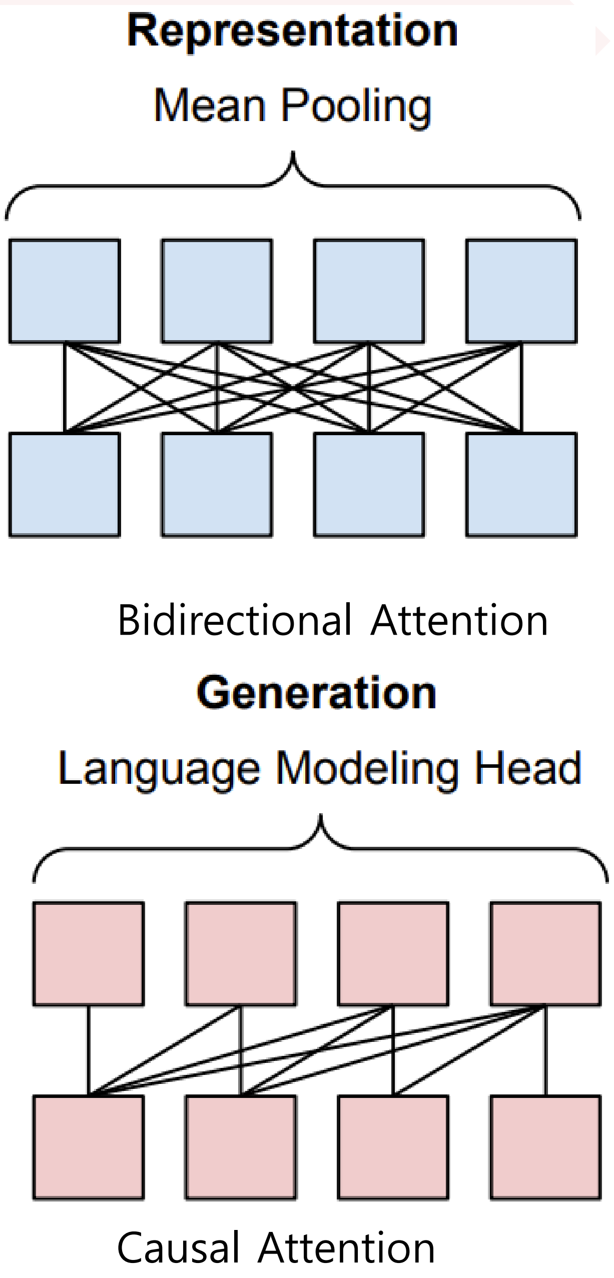
Few-shot embedding does not work

EXPERIMENTS

*Ablations

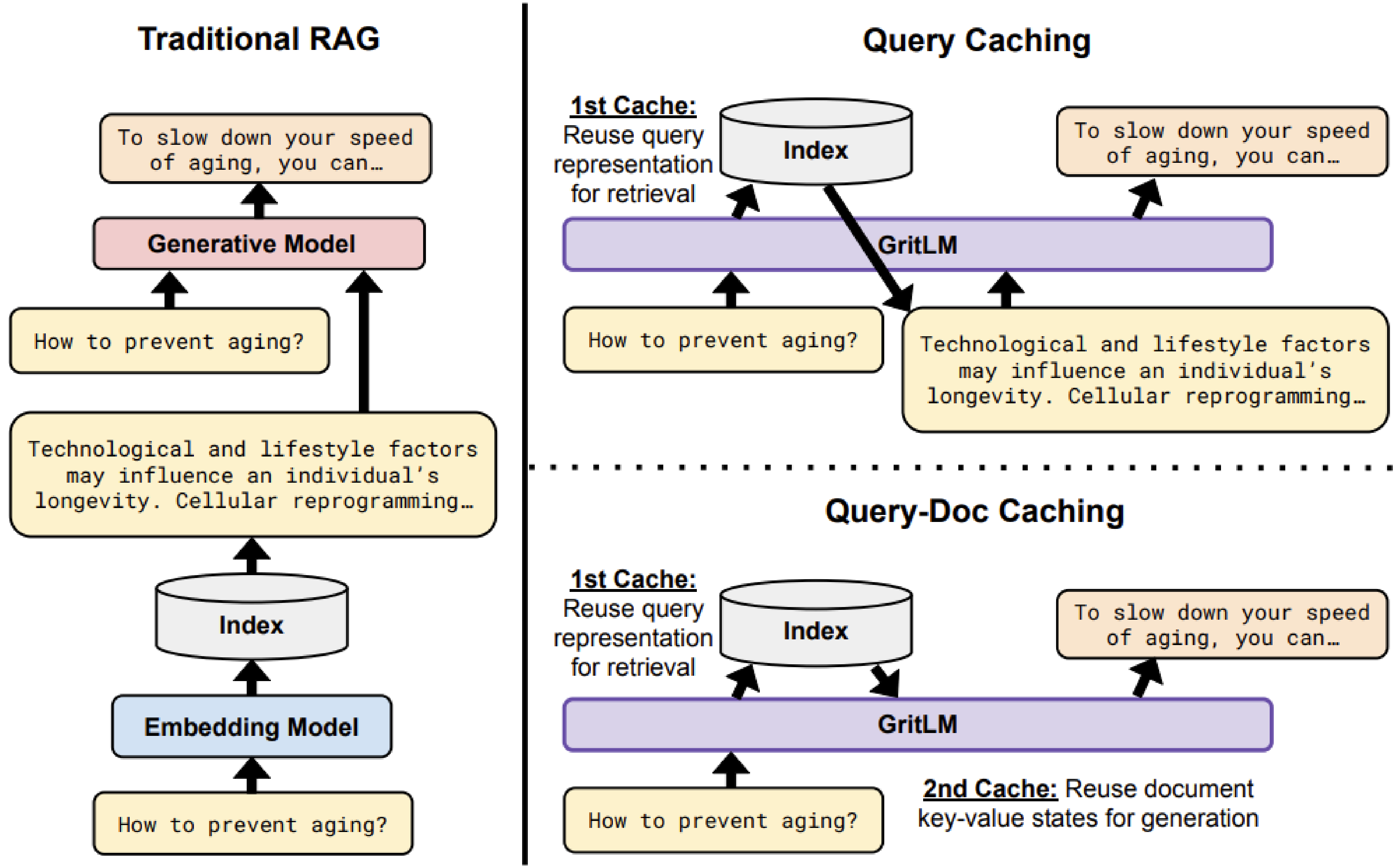
Attention Emb Instruction	Attention Emb Sample	Attention Gen Instruction	Attention Gen Sample	Pooling	Emb	Gen
			Embedding Only			
Causal				Wmean	60.0	-
Causal	Bidirectional			Mean	61.0	-
Bidirectional				Mean	61.8	-
			Generative Only			
		Causal			-	**55.2**
		Bidirectional	Causal		-	50.7
			Unified			
Causal		Causal		Last token	61.2	53.0
Causal		Causal		Wmean	62.8	52.8
Bidirectional		**Causal**		**Mean**	**64.0**	52.9

Attention and Pooling ablations



1. Generative Representational Instruction Tuning

*Caching



1. Generative Representational Instruction Tuning

*Caching Results

	Match (0-shot, ↑)	CPU Latency (s, ↓) Sample A	CPU Latency (s, ↓) Sample B	GPU Latency (s, ↓) Sample A	GPU Latency (s, ↓) Sample B	Storage (↓)
No RAG	21.00	4.3 ± 0.36	13.69 ± 1.0	0.24 ± 0.04	0.38 ± 0.04	**0GB**
Query then document prompt						
RAG	<u>30.50</u>	11.64 ± 0.74	14.88 ± 0.87	0.39 ± 0.02	0.40 ± 0.02	<u>43GB</u>
Query Caching	25.46	18.30 ± 0.76	6.87 ± 0.89	0.44 ± 0.03	<u>**0.27 ± 0.02**</u>	<u>43GB</u>
Query-Doc Caching	21.63	**5.12 ± 0.23**	<u>6.62 ± 0.97</u>	<u>0.27 ± 0.03</u>	0.29 ± 0.01	30TB
Document then query prompt						
RAG	30.47	14.18 ± 1.01	15.33 ± 0.87	0.39 ± 0.01	0.4 ± 0.01	<u>43GB</u>
Doc Caching	**33.38**	5.25 ± 0.34	23.23 ± 1.05	<u>0.27 ± 0.03</u>	0.45 ± 0.02	30TB
Doc-Query Caching	18.39	<u>5.23 ± 0.37</u>	**6.41 ± 0.96**	**0.26 ± 0.03**	<u>**0.27 ± 0.02**</u>	30TB

Caching storage and Latency time

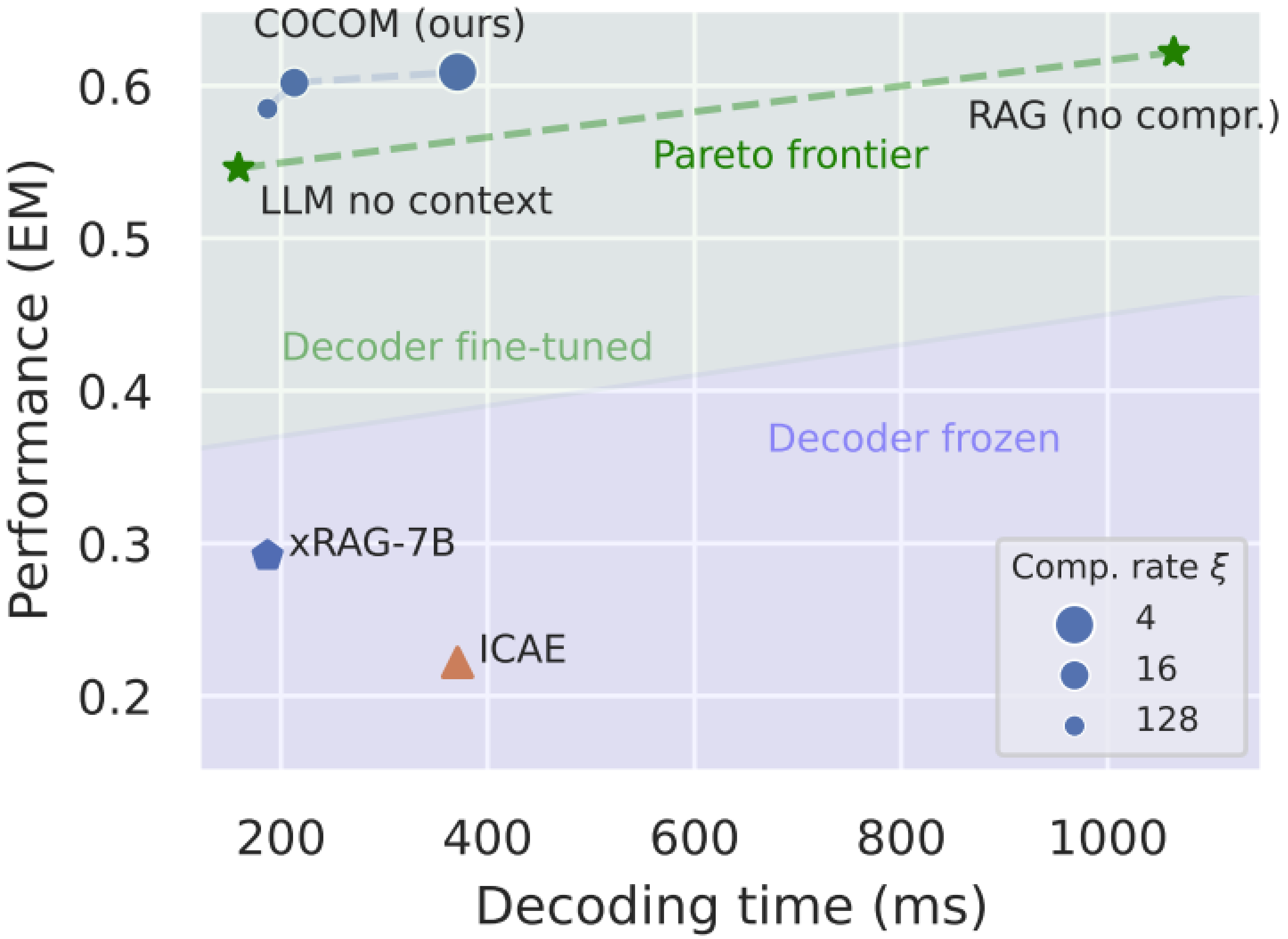
Conclusion

- Generation과 Embedding, Reranking을 한번에 한 모델로 진행할 수 있으며 간단함
- Caching을 통해 Inference 속도를 향상할 수 있음

***한계**

- Embedding model의 사이즈가 상용 모델에 비해 커지며, 차원도 늘어남
- Caching의 Storage size가 급격히 증가
- Embedding 할 때 Caching 값이 Attention 차이로 인해 그대로 활용하기 어려움

2. Context Embeddings for Efficient Answer Generation in RAG

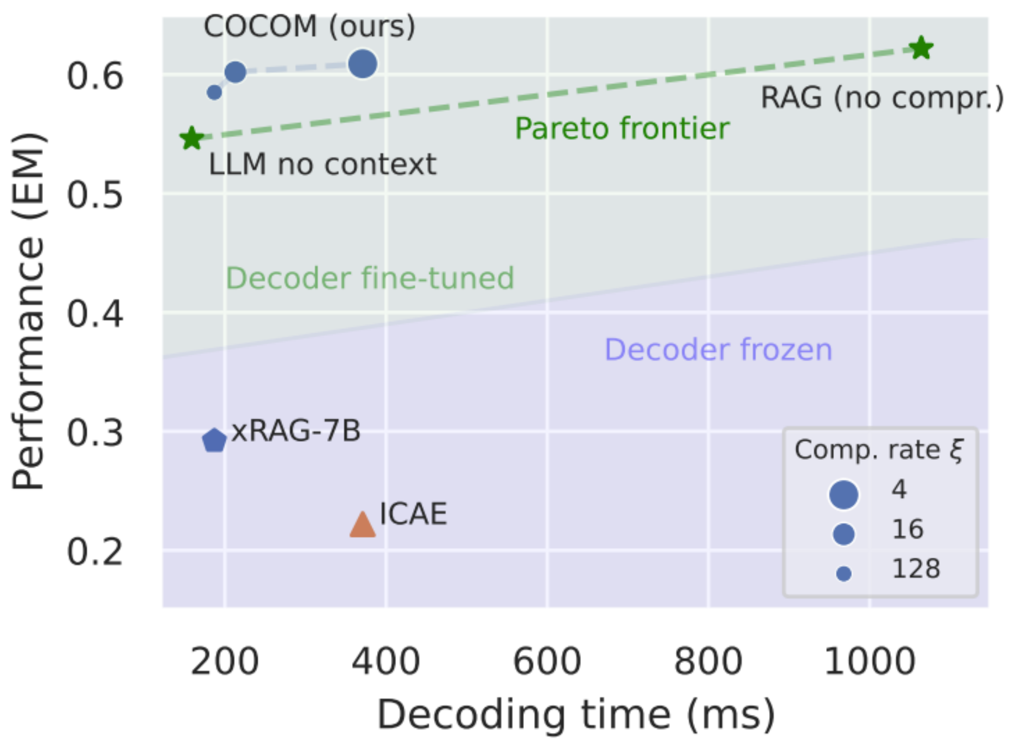


2. Context Embeddings for Efficient Answer Generation in RAG

***Abstract & Introduction**

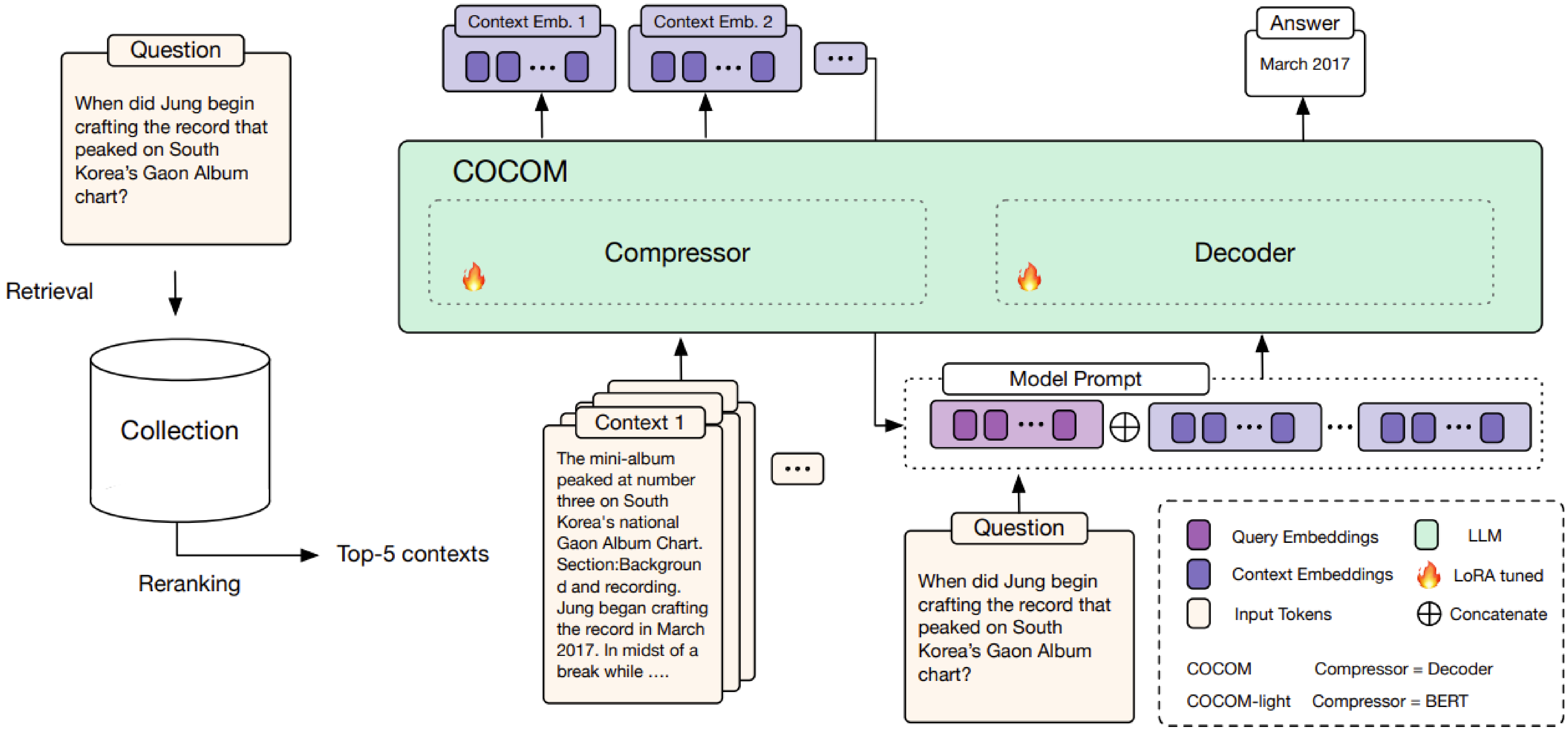
- 기존 RAG의 문제
  - 외부 정보를 입력으로 주면서 LLM의 제한된 지식을 극복할 수 있도록 하지만 Context가 길어져 Decoding 시간이 길어 짐

- 기존 Compressor의 문제
  - 대형 Compressor model에 의존
  - Decoder의 학습 부재로 인해 낮은 성능
  - 고정된 Compression rate
  - 단일 Document



=> Long Context를 압축하여 Context Embedding으로 줄여 Decoding Time을 단축

2. Context Embeddings for Efficient Answer Generation in RAG

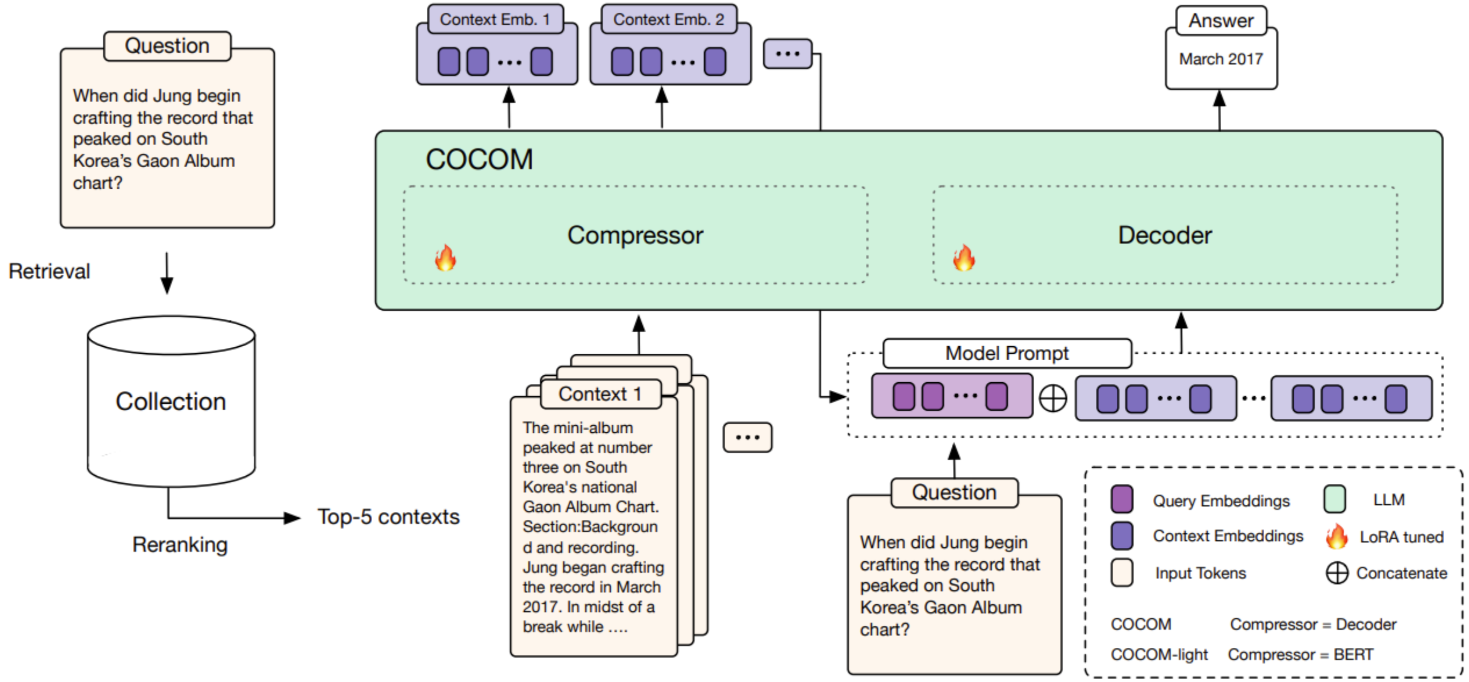


METHODOLOGY

*COCOM Inference

$$\phi_{comp} : \{t_1, t_2, \ldots, t_n\} \rightarrow \{e_1, e_2, \ldots, e_k\} \in \mathbb{R}^d$$

$$\theta_{LLM} : \{\mathcal{E}, x\} \rightarrow r \qquad \phi_{comp} = \theta_{LLM}$$



METHODOLOGY

***COCOM Pre-training 1**

$$\mathcal{E} = \phi_{comp}(x_1, x_2, \ldots, x_T)$$

$$\mathcal{L}(\theta_{LLM}, \phi_{comp}) = -\sum_{x_t \in \mathcal{X}} \log P_{\theta_{LLM}}(x_t \mid \mathcal{E}, x_1, \ldots, x_{t-1})$$

- Compressor을 통해 압축된 Embedding이 Decoder를 통해 원상복구 될 수 있도록 학습

***COCOM Pre-training 2**

$$\mathcal{X} = \{x_1, x_2, \ldots, x_T\} \qquad \mathcal{X}_A = \{x_1, x_2, x_j\} \qquad \mathcal{X}_B = \{x_{j+1}, \ldots, x_T\}$$

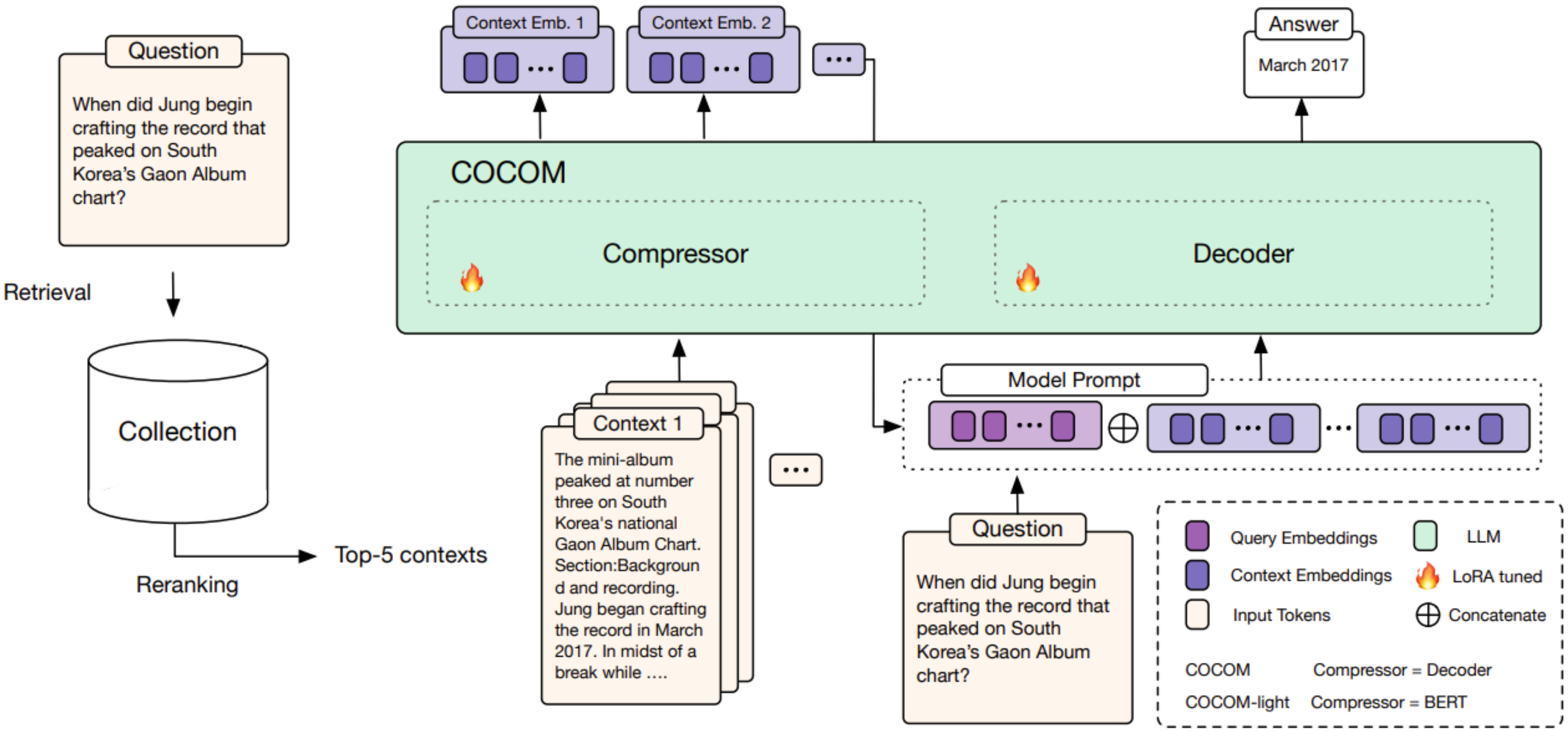
$$\mathcal{L}(\theta_{LLM}, \phi_{comp}) = -\sum_{x_t \in \mathcal{X}_B} \log P_{\theta_{LLM}}(x_t \mid \phi_{comp}(\mathcal{X}_A), x_1, \ldots, x_{t-1})$$

- Compressor을 통해 부분만 압축된 문장을 복구할 수 있도록 학습

METHODOLOGY

*COCOM Fine-tuning

$$\mathcal{L}(\theta_{LLM}, \phi_{comp}) = -\sum_{r_t \in R} \log P_{\theta_{LLM}}(r_t \mid I_{\mathcal{E},q}, r_1, r_2, \ldots, r_{t-1})$$



RESULTS

*Main Results

Decoder		Method	Compression rate (ξ)	NQ	TriviaQA	HotpotQA	ASQA	PopQA	Average
				Dataset					
Zero-shot		AutoCompressor [4]★	× 4	0.000	0.000	0.000	0.000	0.000	0.000
		ICAE [8]	× 4	0.210	0.592	0.184	0.222	0.290	0.300
		xRAG [3]★							
		Mistral-7B-v0.2	× 128	0.184	0.622	0.185	0.182	0.199	0.274
		Mixtral-8x7b	× 128	0.265	0.744	0.239	0.292	0.318	0.372
Fine-tuned	*Mistral-7B-v0.2*	RAG$^{\triangle}$ (no compression)	-	0.597	0.883	0.500	0.622*	0.514	0.623
		LLM$^{\triangledown}$ (without context)	-	0.359	0.708	0.264	0.546	0.199	0.416
		COCOM-light *(ours)*	× 4	0.539	0.849	0.409	0.601*	0.458	0.531
			× 16	0.492	0.823	0.367	0.565	0.385	0.526
			× 128	0.444	0.794	0.321	0.550	0.314	0.485
		COCOM (ours)	× 4	0.554	0.859	0.430	0.609	0.474	0.585
			× 16	0.539	0.852*	0.426*	0.602*	0.465	0.577
			× 128	0.511	0.835	0.378	0.585*	0.391	0.540

COCOM-light = Compressor를 BERT 모델로 활용하고 차원을 맞추기 위해 Projection layer 추가

RESULTS

*Computational Efficiency

Model *Mistral-7b-v0.2*	ξ	Decoding Time (ms)	GPU Mem. (GB)	GFLOPs
RAG (no compr.)	-	1064	18.1	25031
LLM (no context.)	-	159	14.1	607
COCOM (-light)	4	371 (× 2.87)	15.1 (× 1.20)	7016 (× 3.57)
	16	213 (× 5.00)	14.4 (× 1.29)	2465 (× 10.16)
	128	187 (× 5.69)	14.2 (× 1.27)	1138 (× 22.00)

시간, 메모리, 연산량 모두에서 극적인 감소를 보여줌

Compressor	ξ	Time (h)	Index size (TB)
COCOM	4	89	6.06
	16	77	1.51
	128	73	0.19
COCOM-light	4	1	6.06
	16	1	1.51
	128	1	0.19

24m Context 압축량

RESULTS

*Ablation

Model	ξ	NQ		ASQA	
		k=1	k=5	k=1	k=5
COCOM	4	0.499	0.554	0.558	0.609
	16	0.491	0.539	0.541	0.602
	128	0.482	0.511	0.544	0.585

Doc의 숫자를 늘이면 성능 증가

Ablation	Datasets	
	NQ	ASQA
COCOM-light (baseline)	0.444	0.550
w/o pre-training	0.423	0.524
pre-training on FineWeb	0.427	0.545
w/o tuning decoder	0.353	0.438
COCOM (baseline)	0.519	0.585
w/o pre-training	0.490	0.565
pre-training on FineWeb	0.503	0.581
w/o tuning decoder	0.421	0.521

Ablation study

Conclusion

- 여러 압축률을 선택할 수 있어 상황에 따라 품질과, 생성 시간의 균형을 유지할 수 있음
- Context를 여러 개 넣을 수 있어 생성 품질 향상

***한계**

- Retrieval model을 따로 써야 한다는 것은 변함 없음
- Context가 5개로 제한됨
- Document를 청크 크기로 나눈 후 Compressor model을 지날 때 7B 모델 전체를 지나가야 하므로 리소스 소모가 큼

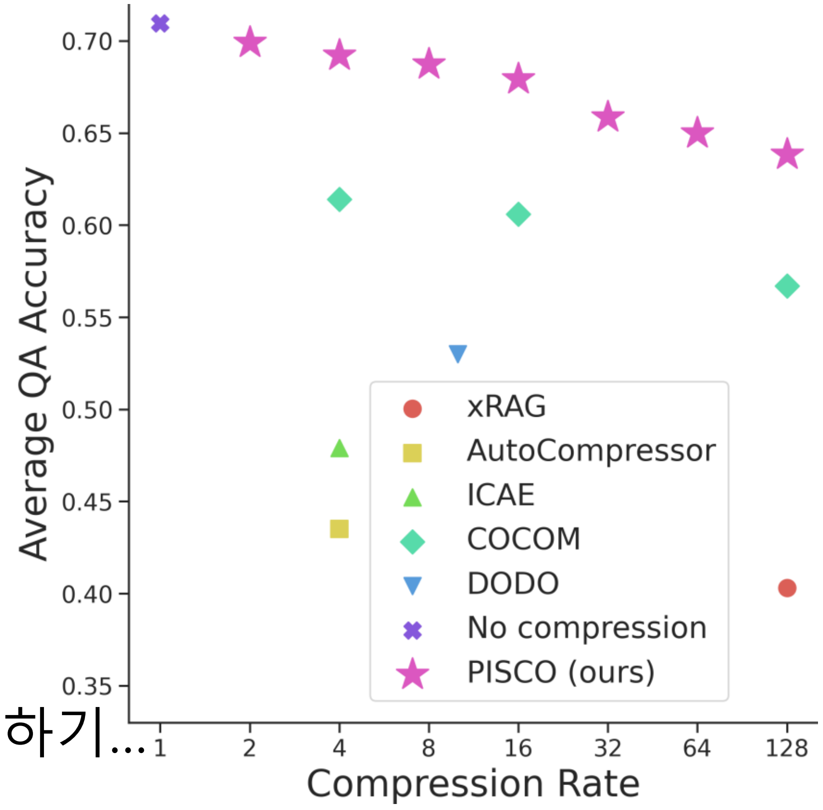
3. PISCO: Pretty Simple Compression for RAG

***Abstract & Introduction**

- 기존 RAG의 문제
  - Long Context
  - Inference Cost

- 기존 압축 방법의 문제
  - COCOM은 압축률이 높지만 정답률이 많이 떨어짐
  - PreTraining이 필요함

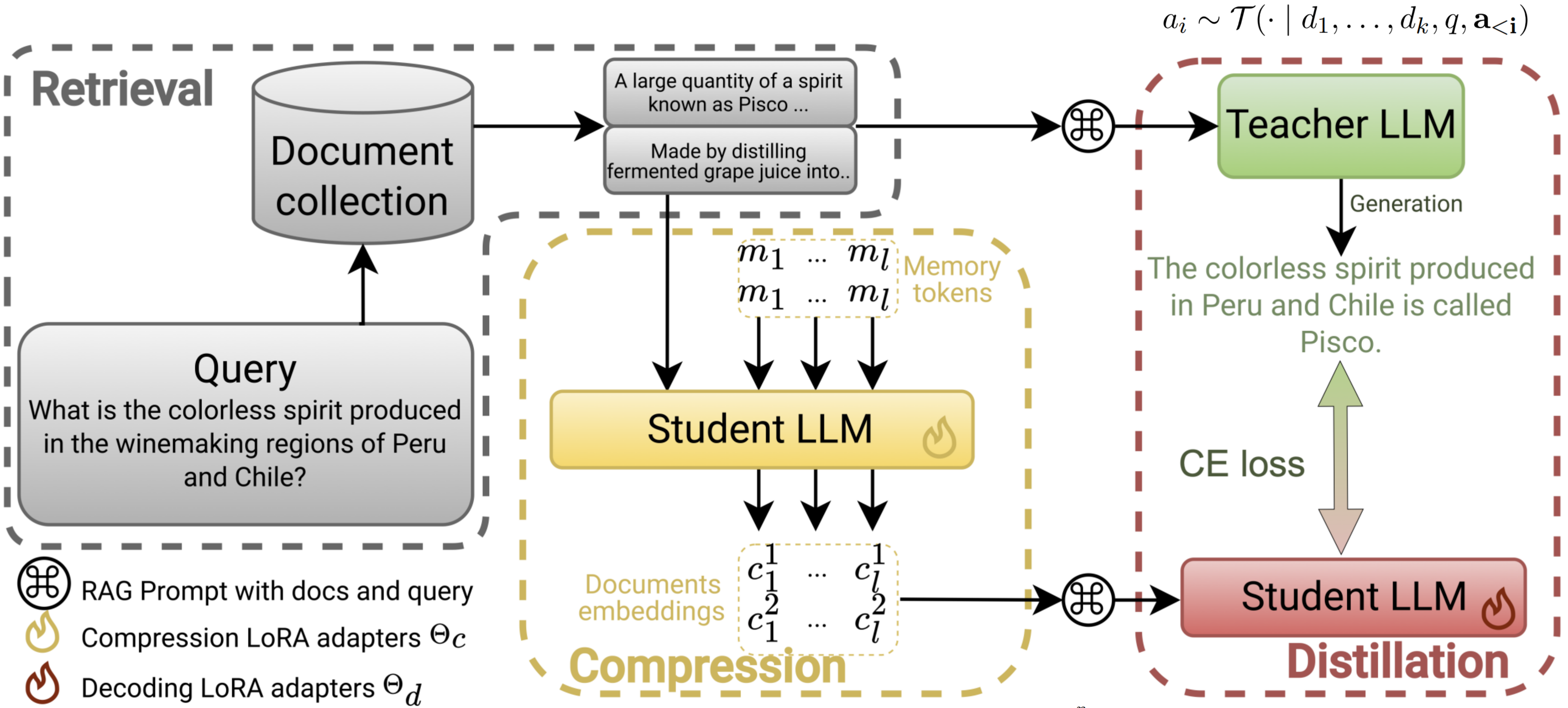
  PreTraining = 원문 복원하기, 빈칸 채우기, 뒷 문장 이어서 생성하기…



⇒ PreTraining 없이, 정확도 손실을 최소화하며 압축률을 유지한 SKD 방식을 활용

SKD = Sequence-level Knowledge Distillation

3. PISCO: Pretty Simple Compression for RAG



$$\mathbf{c_i} = (c_i^s)_{s=1,\ldots,l} = \mathcal{F}_{\theta_c}(d_i, m_1, \ldots, m_l)$$

$$\mathcal{L}(\theta_c, \theta_d) = -\sum_{i=1}^{r} \log \mathcal{F}_{\theta_d}(a_i \mid q, \mathbf{c_1}, \ldots, \mathbf{c_k}, \mathbf{a}_{<\mathbf{i}})$$

Training

***SKD**

- Compressor와 Decoder를 동시에 학습함

$$a_i \sim \mathcal{T}(\cdot \mid d_1, \ldots, d_k, q, \mathbf{a}_{<\mathbf{i}})$$

- Top-5로 뽑힌 128 토큰으로 나뉘어진 Doc를 Query와 함께 Teacher model에 집어넣어 Silver Label 생성

⇒ **여기선 Gold Label인지가 중요한 것이 아니라, 원래 모델이 출력하는 것을 그대로 출력할 수 있는지가 목표**

- Compressor는 문서와, Memory 토큰을 통해 Compress된 Vector C 생성

$$\mathbf{c_i} = (c_i^s)_{s=1,\ldots,l} = \mathcal{F}_{\theta_c}(d_i, m_1, \ldots, m_l)$$

- Decoder는 Query와 C를 통해 Silver Label을 생성하도록 학습

$$\mathcal{L}(\theta_c, \theta_d) = -\sum_{i=1}^{r} \log \mathcal{F}_{\theta_d}(a_i \mid q, \mathbf{c_1}, \ldots, \mathbf{c_k}, \mathbf{a}_{<\mathbf{i}})$$

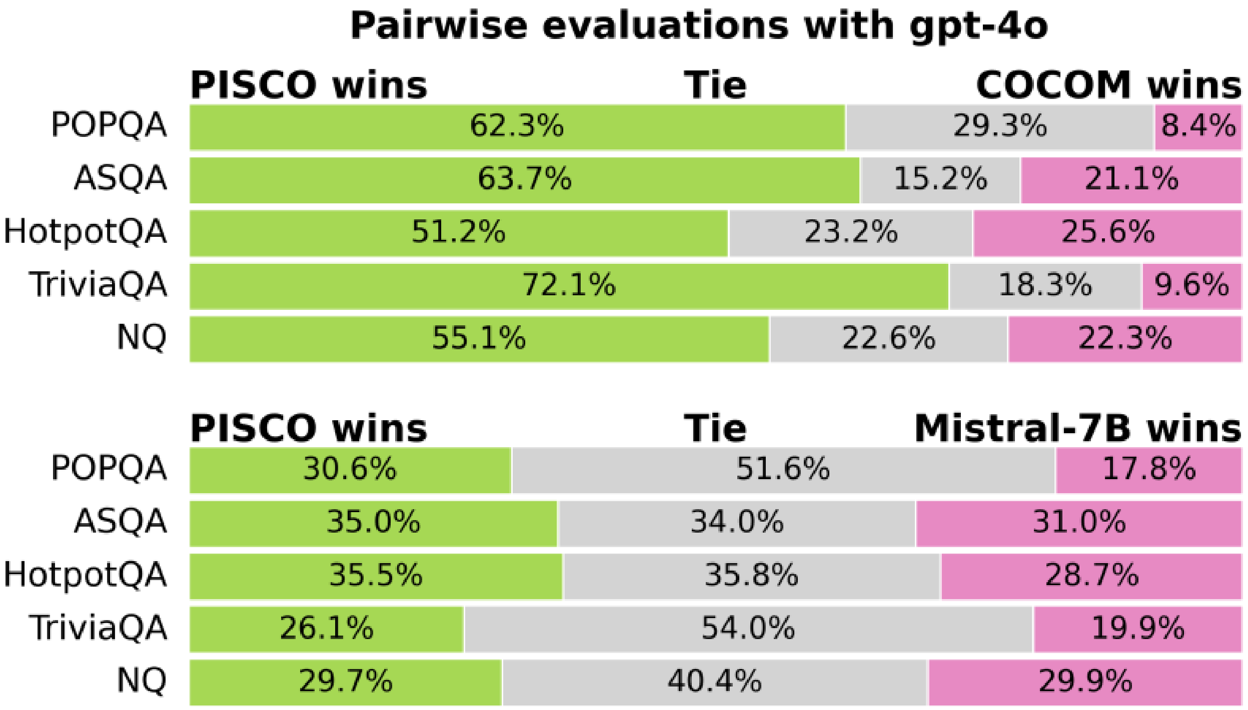
Experiments & Result

*Main Results

	Compression rate	ASQA	HotpotQA	NQ	TriviaQA	POPQA	Average
Decoders with no compression							
Mistral-7B	-	0.74	0.51	0.69	0.92	0.70	0.71
Llama-3.1-8B	-	0.71	0.50	0.65	0.90	0.68	0.69
Solar-10.7B	-	0.75	0.55	0.71	0.93	0.71	0.73
Compression Models							
xRAG-mistral-7B† (Cheng et al., 2024)	x128	0.34	0.27	0.32	0.77	0.33	0.40
AutoCompressor†‡ (Chevalier et al., 2023)	x4	0.57	0.31	0.35	0.70	0.24	0.43
ICAE‡ (Ge et al., 2023)	x4	0.47	0.29	0.42	0.78	0.43	0.48
DODO‡ (Qin et al., 2024)	x10	0.52	0.38	0.48	0.82	0.47	0.53
COCOM (Rau et al., 2024b)	x16	0.63	0.46	0.58	0.89	0.48	0.61
PISCO - Mistral$^{\triangle}$ (Ours)	x16	0.72	0.48	0.65	0.90	0.66	0.68
PISCO - Mistral (x128) (Ours)	x128	0.68	0.46	0.61	0.89	0.55	0.64
PISCO - Llama (Ours)	x16	0.72	0.50	0.64	0.91	0.66	0.69
PISCO - Solar (Ours)	x16	**0.78**	**0.57**	**0.70**	**0.94**	**0.71**	**0.74**

Experiments & Result

*Main Results



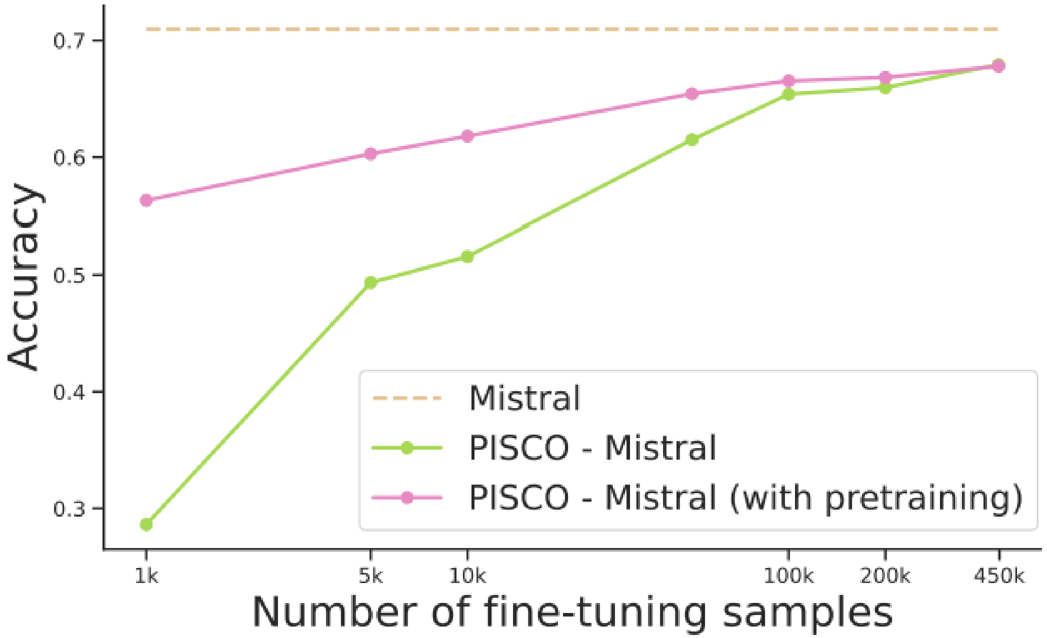
GPT 4o를 통한 Answer Quality Evaluation

Model	GFlops	Time(s)	Max batch size
Mistral-7B	11231	0.26	256
PISCO-Mistral	2803	0.06	1024
PISCO-Mistral (x128)	2312	0.05	>1024

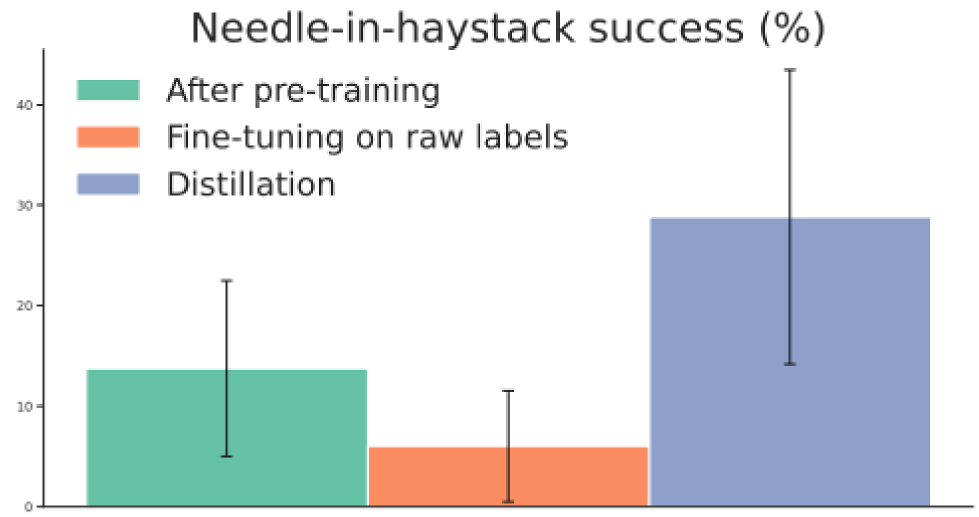
Computational efficiency

Experiments & Result

***Analysis of Training Data and Tasks**



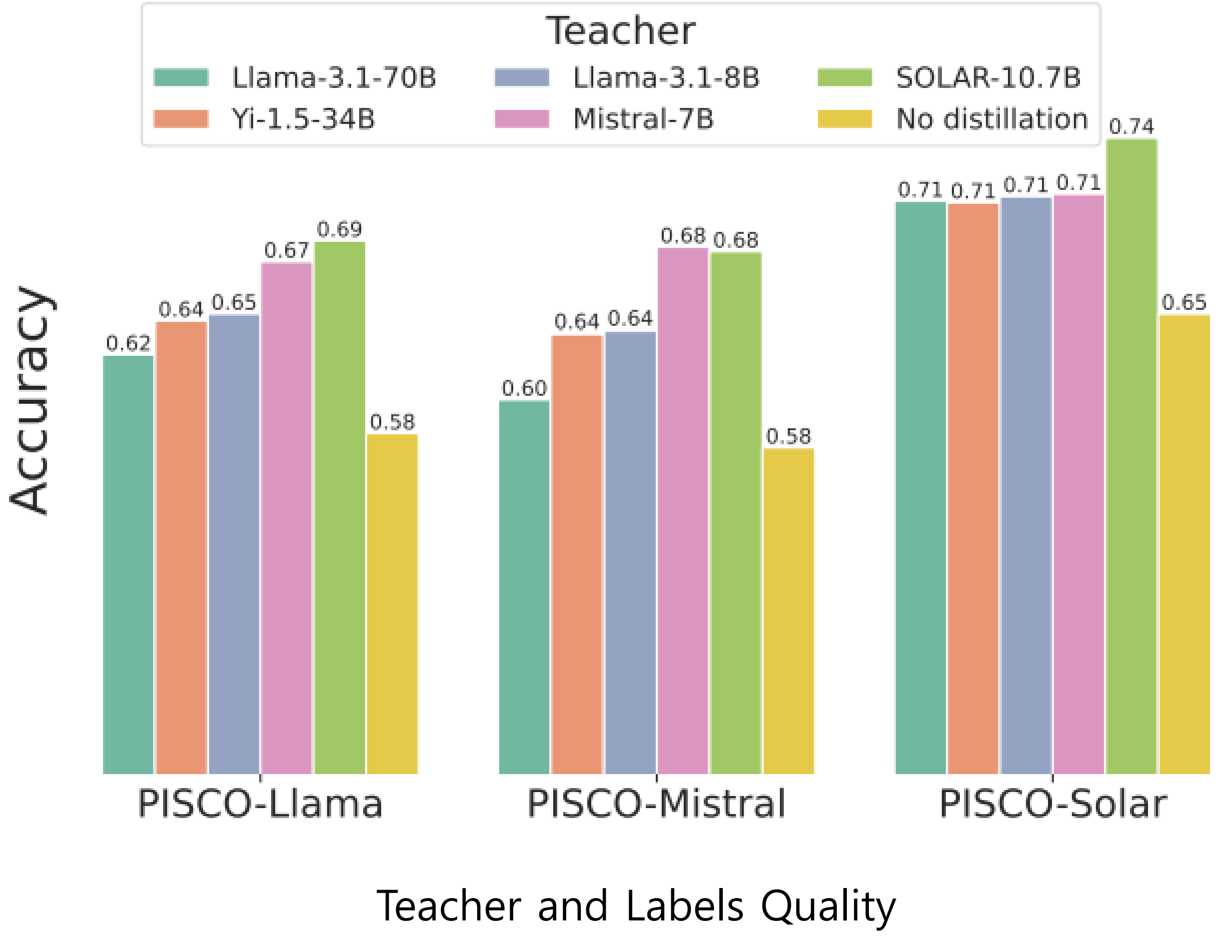
PreTraining을 진행해도 450k samples 이후로는 동등



Needle in haystack task에서 단순 fine-tuning이 실패

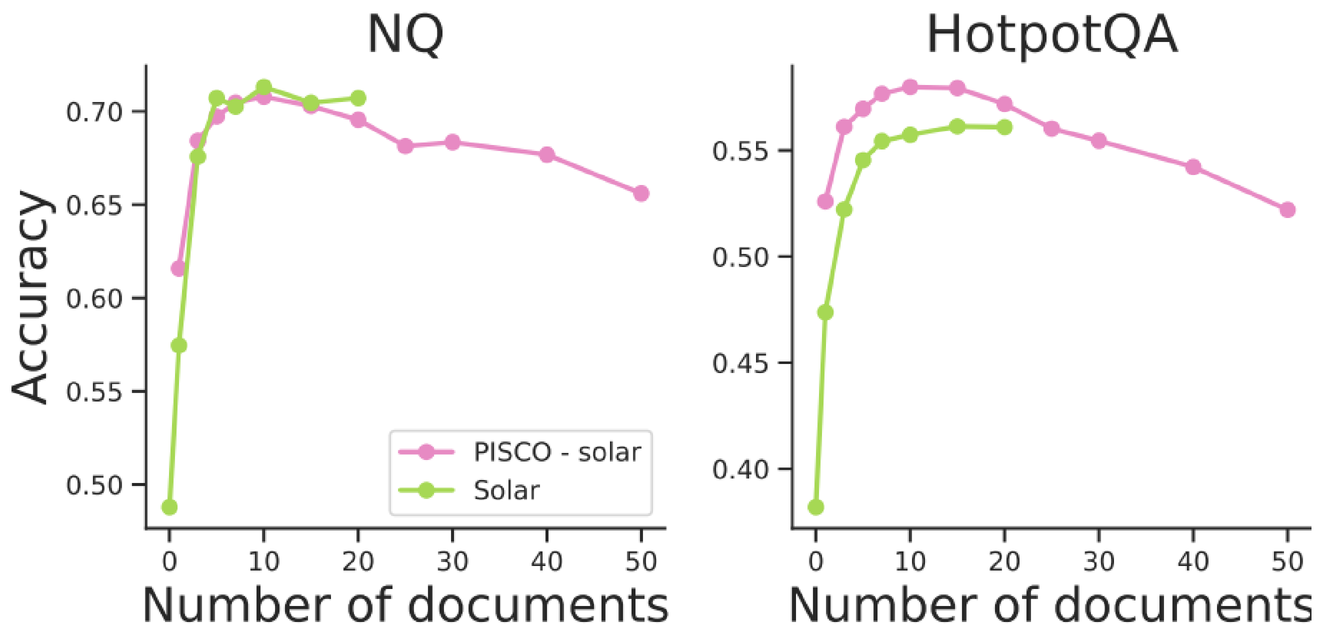
Experiments & Result

*Analysis of Training Data and Tasks



Experiments & Result

***Generalization Evaluation**



Increasing the num of documents

Experiments & Result

*Generalization Evaluation

Dataset Metric	Bio-QA Recall	Covid F1	ParaphraseRC Accuracy	RobustQA Lifestyle F1	Writing F1	Science F1	Recreation F1	Tech F1	Multilingual QA FR recall-3gram	KO	RU
Llama	0.26	**0.17**	0.48	0.25	0.23	0.25	0.25	0.23	0.56	0.28	0.47
PISCO - Llama	0.24	0.12	0.38	0.28	0.27	**0.26**	**0.26**	**0.26**	0.54	0.24	0.44
Mistral	0.27	0.13	0.49	0.28	0.27	**0.26**	0.25	0.25	0.57	0.26	0.40
PISCO - Mistral	0.26	0.11	0.38	0.28	0.27	**0.26**	0.25	**0.26**	0.52	0.17	0.35
Solar	0.28	0.14	**0.50**	0.28	0.27	**0.26**	**0.26**	0.25	0.59	0.20	0.52
PISCO - Solar	**0.29**	0.10	0.47	**0.29**	**0.28**	0.25	0.25	**0.26**	0.60	0.16	0.48

Out of domain & Multilinguality

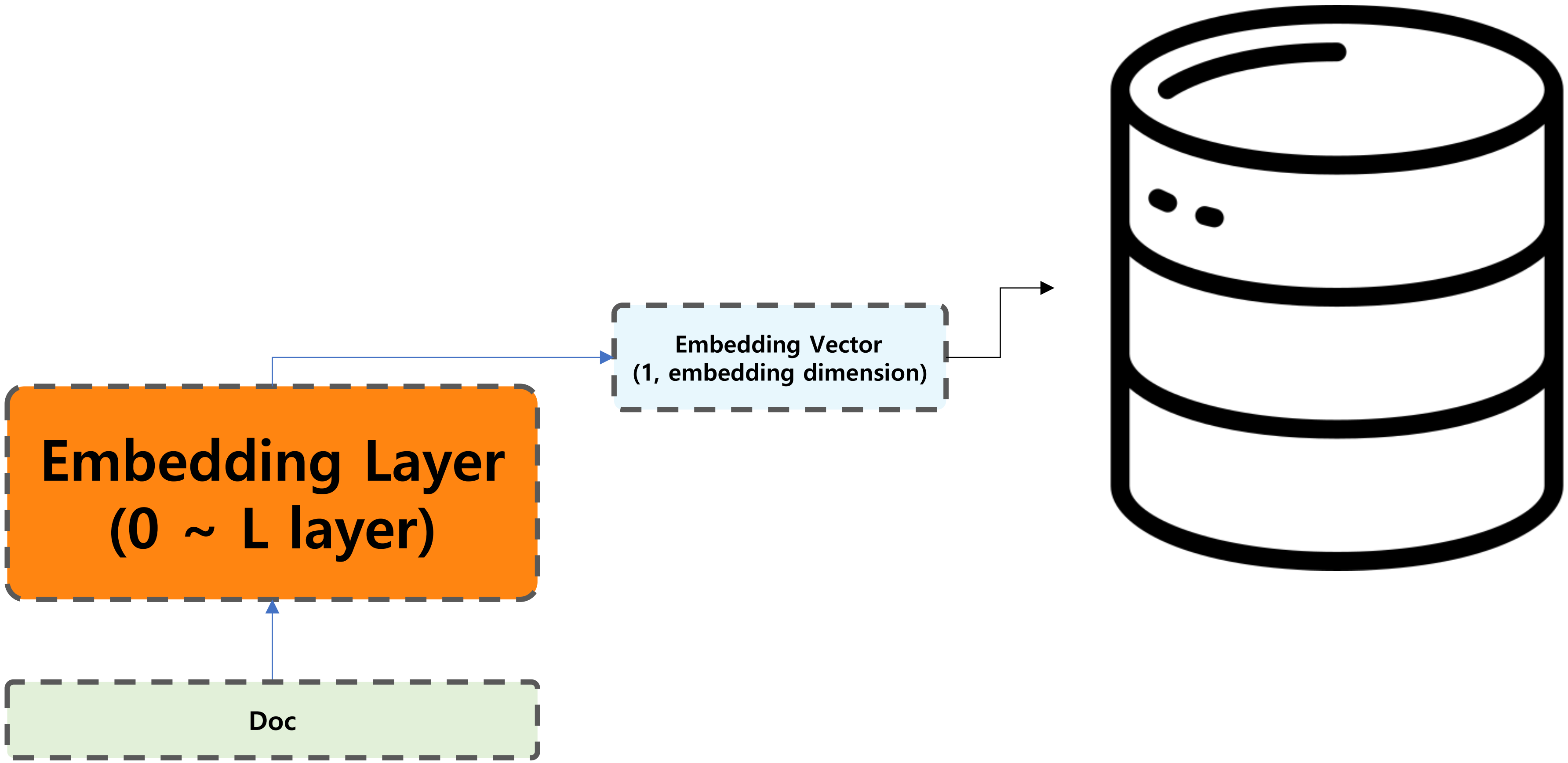
Conclusion

- 단순한 Gold Label을 학습하는 것 보다, 기존 모델의 출력인 Silver label을 그대로 따라가는 것이 중요함
- 다양한 실험을 통해 일반화를 진행하였고, 다양한 도메인과 언어에서도 강인한 모습을 보여주었음
- 성능 저하를 최소화하며 추론 비용과 시간을 줄였음

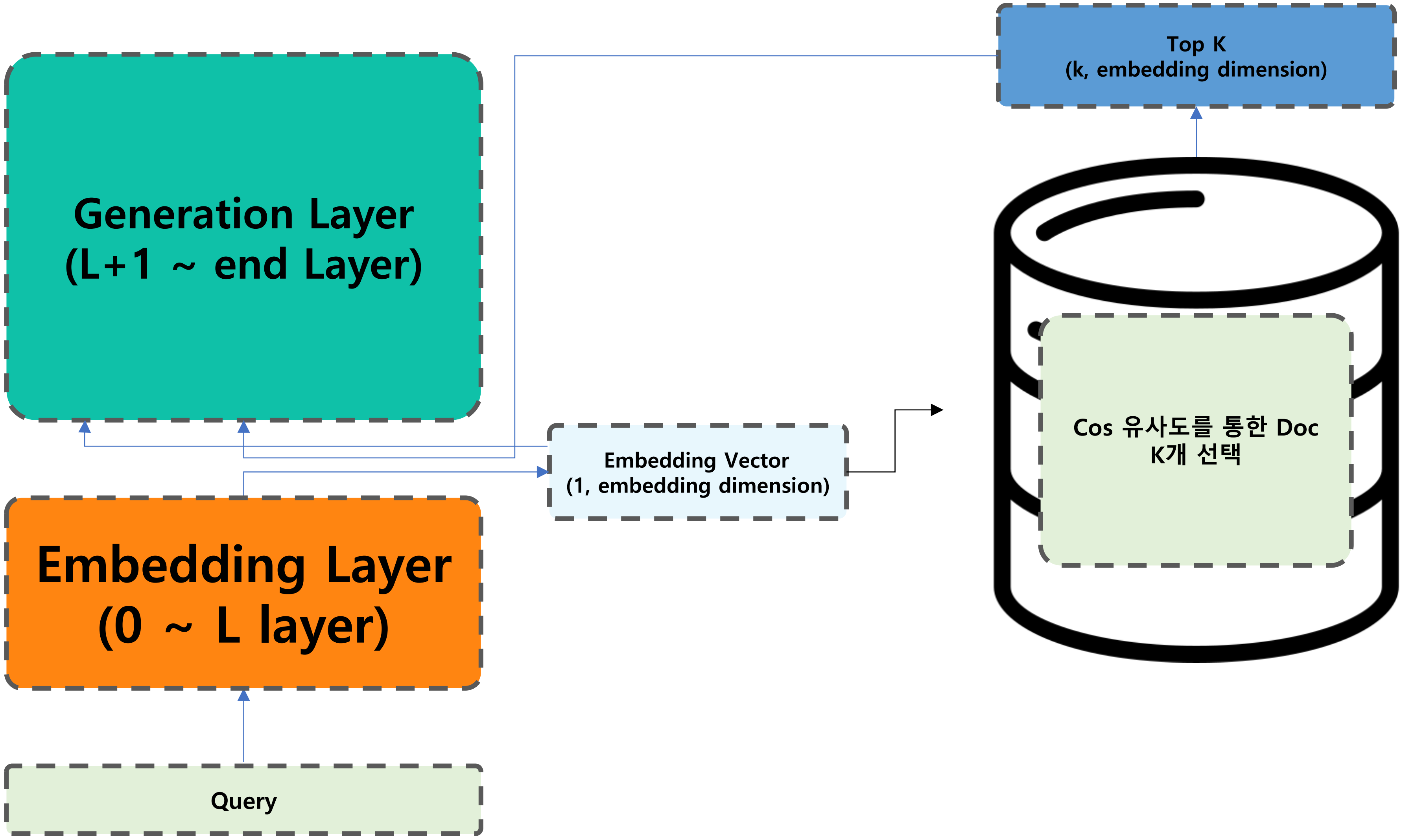
***한계**

- Retrieval model을 따로 써야 한다는 것은 변함 없으며, 모델을 3개나 써야 함
- Document를 청크 크기로 나눈 후 Compressor model을 지날 때 7 ~ 10B 모델 전체를 지나가야 하므로 리소스 소모가 큼

4. HEGA – Hybrid Embedding-to-Generation Architecture

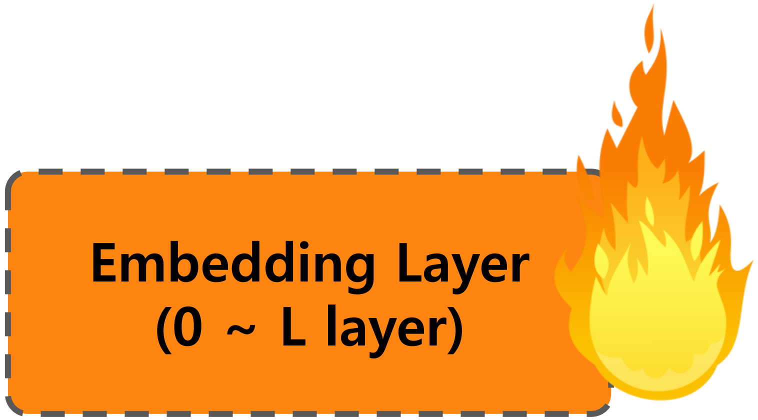


4. HEGA – Hybrid Embedding-to-Generation Architecture

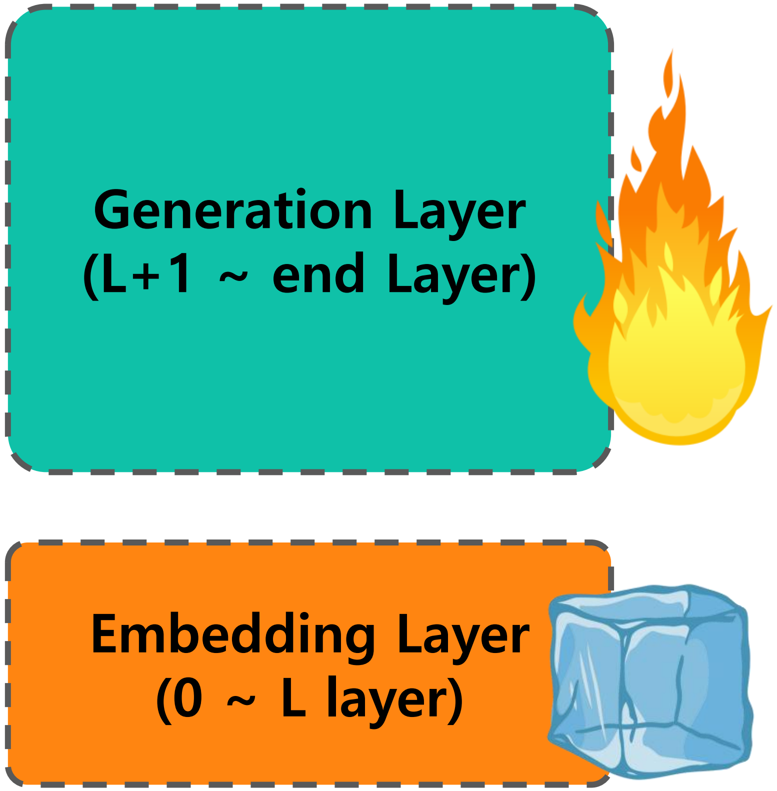


4. HEGA – Hybrid Embedding-to-Generation Architecture

*Training



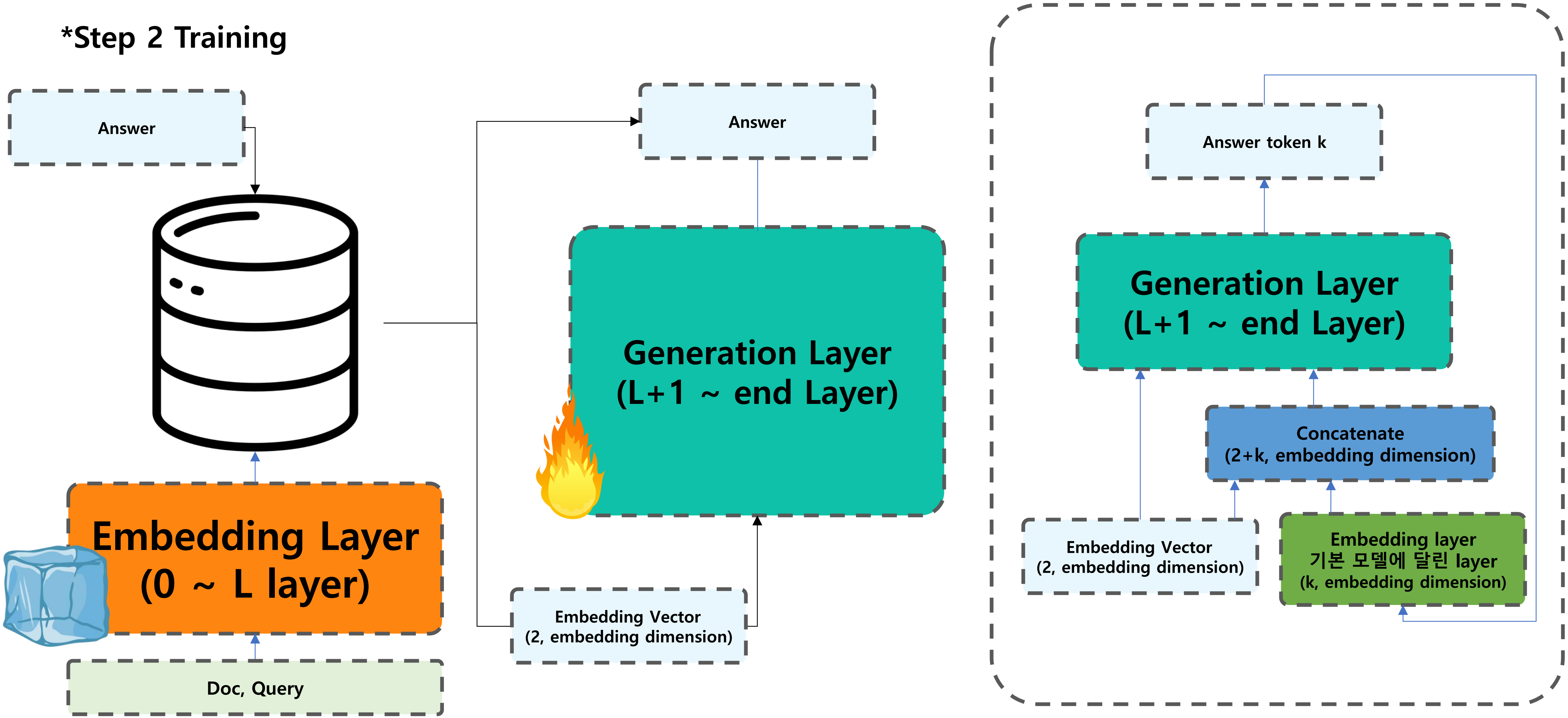
Training Step 1 - Embedding



Training Step 2 – Generation

4. HEGA – Hybrid Embedding-to-Generation Architecture

*Step 2 Training

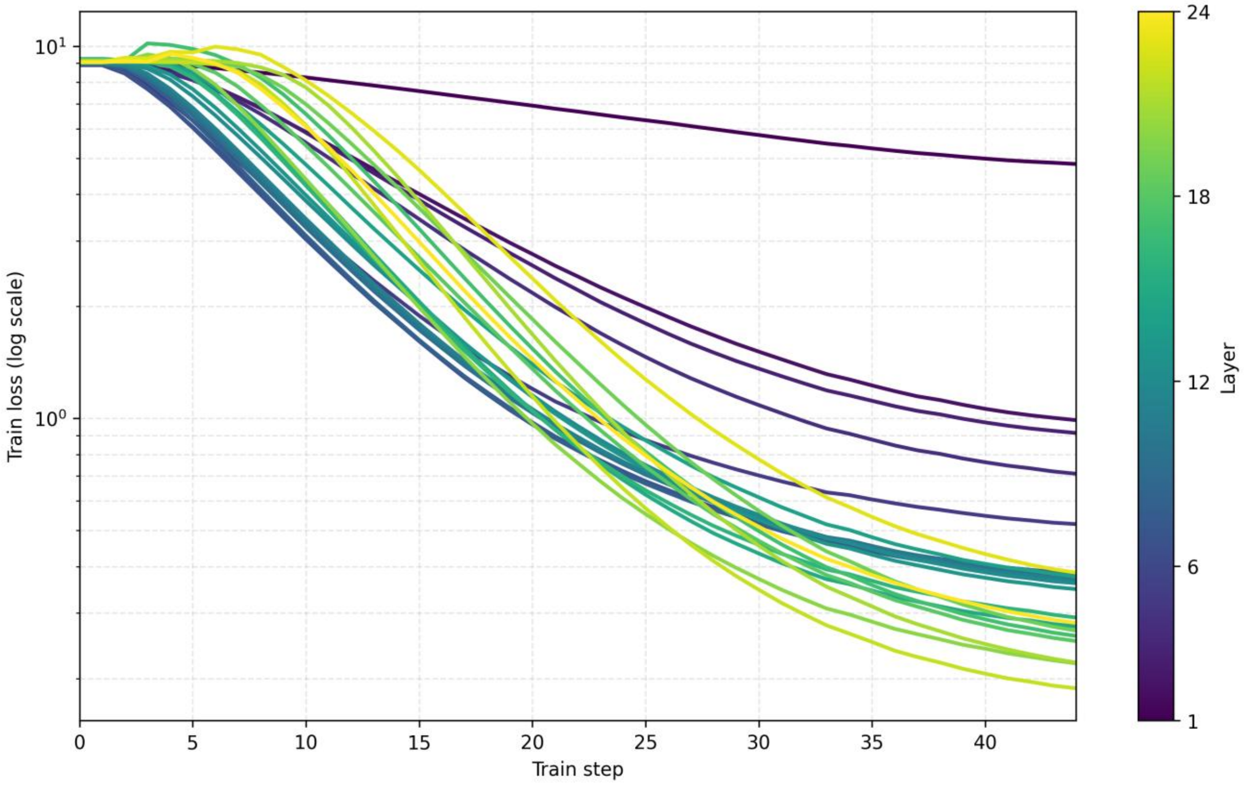


4. HEGA – Hybrid Embedding-to-Generation Architecture

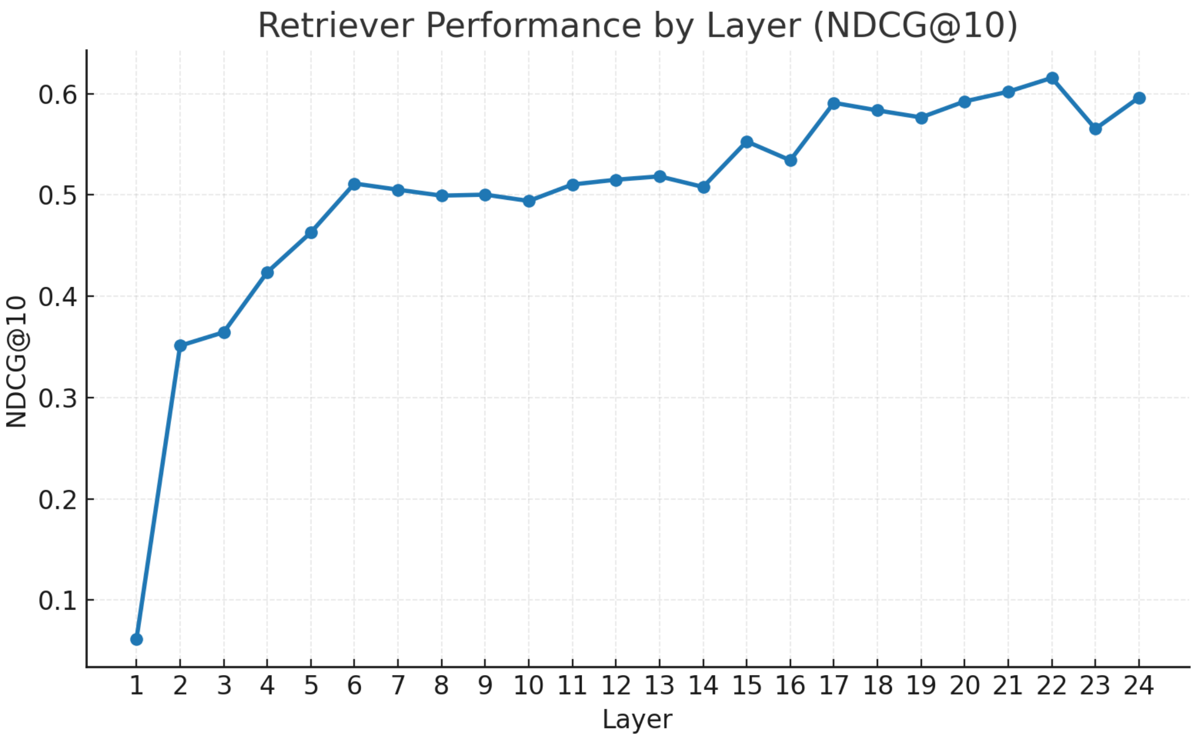
***Step 1 Training Experiments**

- Qwen2.5-0.5B : 24 Layer
- 1 ~ 24Layer까지 Pruning을 진행하여 Last Token을 embedding으로 사용하고, 동일한 데이터 셋(nlpai-lab/ko-triplet-v1.0)에서 학습하여 초반, 중반 부 Layer에서 모델을 다 쓰지 않아도 일정한 성능을 유지할 수 있는지 확인
- 1 epoch / 5e-5 Learning rate / 4096 Batch size / 1024 max length
- A 6000 4장에서 5일 걸림

4. HEGA – Hybrid Embedding-to-Generation Architecture



Embedding Training Loss



Embedding Evaluation

Thank you