

PEFT & RAG Overview

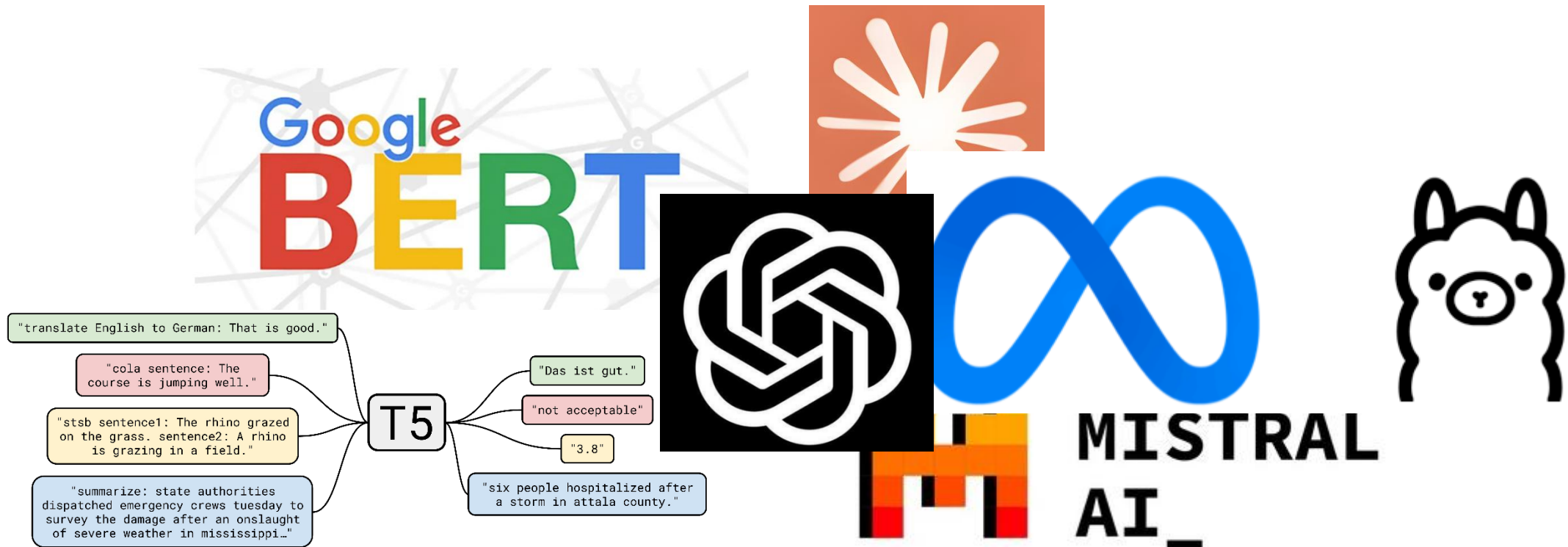
Contents

1. Introduction
2. PEFT (Parameter-Efficient Fine-Tuning)
 - PEFT
 - Methods
 - Pros & Cons
3. RAG (Retrieval-Augmented Generation)
 - Retrieval-based LMs
 - Why Retrieval-based LMs?
 - Methods
 - Recent work
 - Pros & Cons
4. Conclusion

Introduction

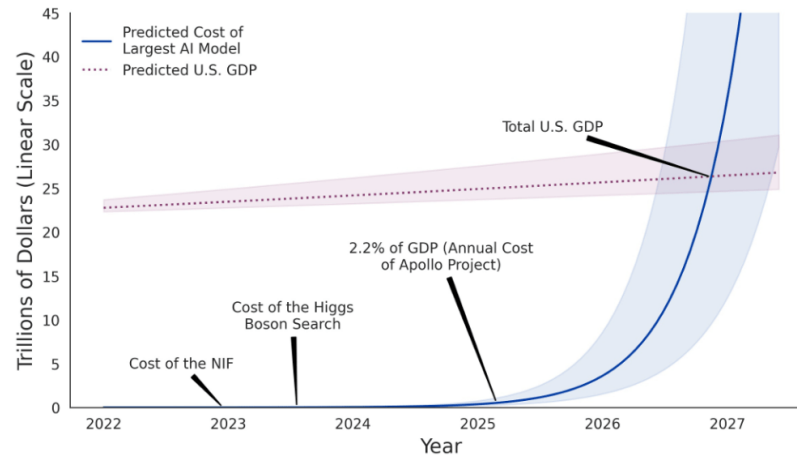
Introduction

- PLM은 NLP분야에서 혁신적인 성과를 이뤄냈으며, 대표적으로 GPT와 BERT와 같은 모델들은 다양한 언어 이해 및 생성 작업에서 인간 수준의 성능을 보임.
- 이후, ChatGPT를 이어 Parameter를 늘려 성능을 올린 LLM이 등장하며 대화형 AI, 문서 요약, 질의 응답 시스템 등 폭넓게 사용



Introduction

- LLM의 발전은 인간 수준 달성과 동시에 몇 가지 한계와 도전 과제를 수반
 - 학습 비용 문제
 - 메모리와 연산 자원 한계
 - 학습된 지식 기반의 한계
 - Fine-Tuning의 비효율성

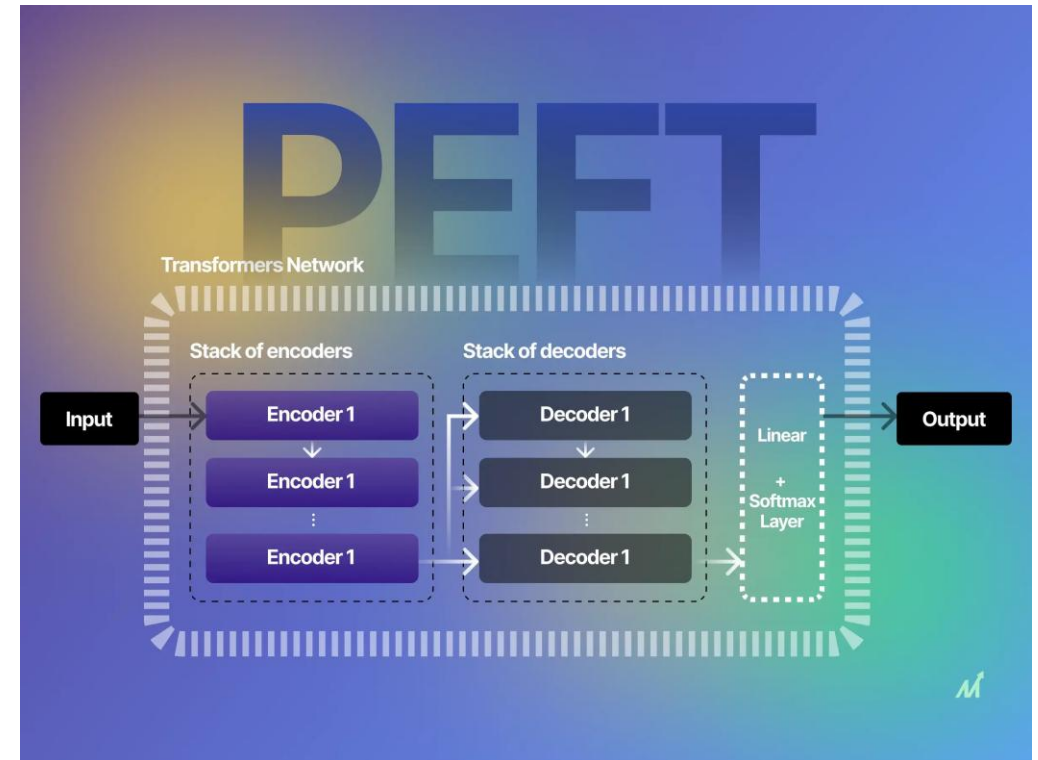
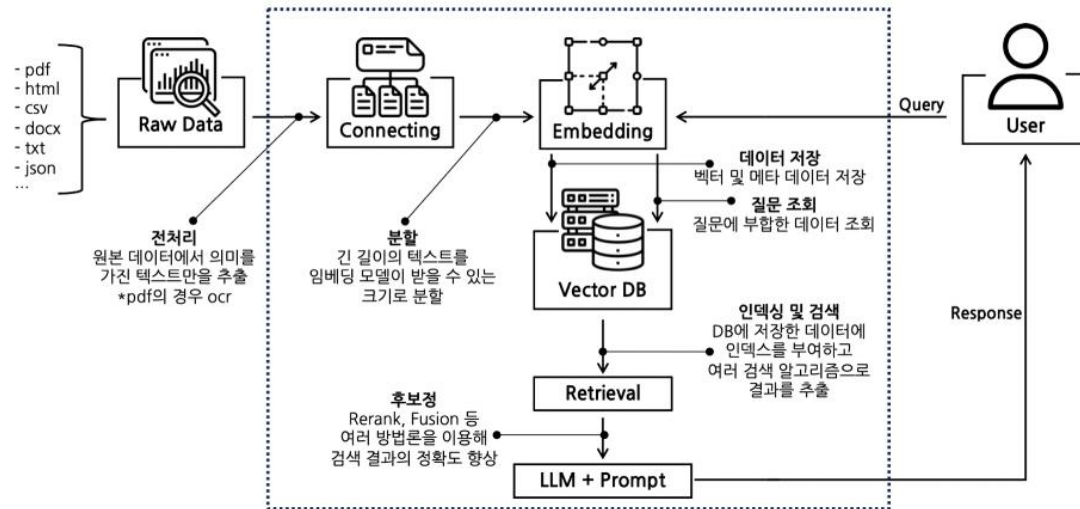


Source: CSET. Note: The blue line represents growing costs assuming compute per dollar doubles every four years, with error shading representing no change in compute costs or a doubling time as fast as every two years. The red line represents expected GDP at a growth of 3 percent per year from 2019 levels with error shading representing growth between 2 and 5 percent.



Introduction

- LLM의 비효율성 문제 해결을 위해 비용 절감과 도메인 적합성을 위한 RAG 및 PEFT가 필요

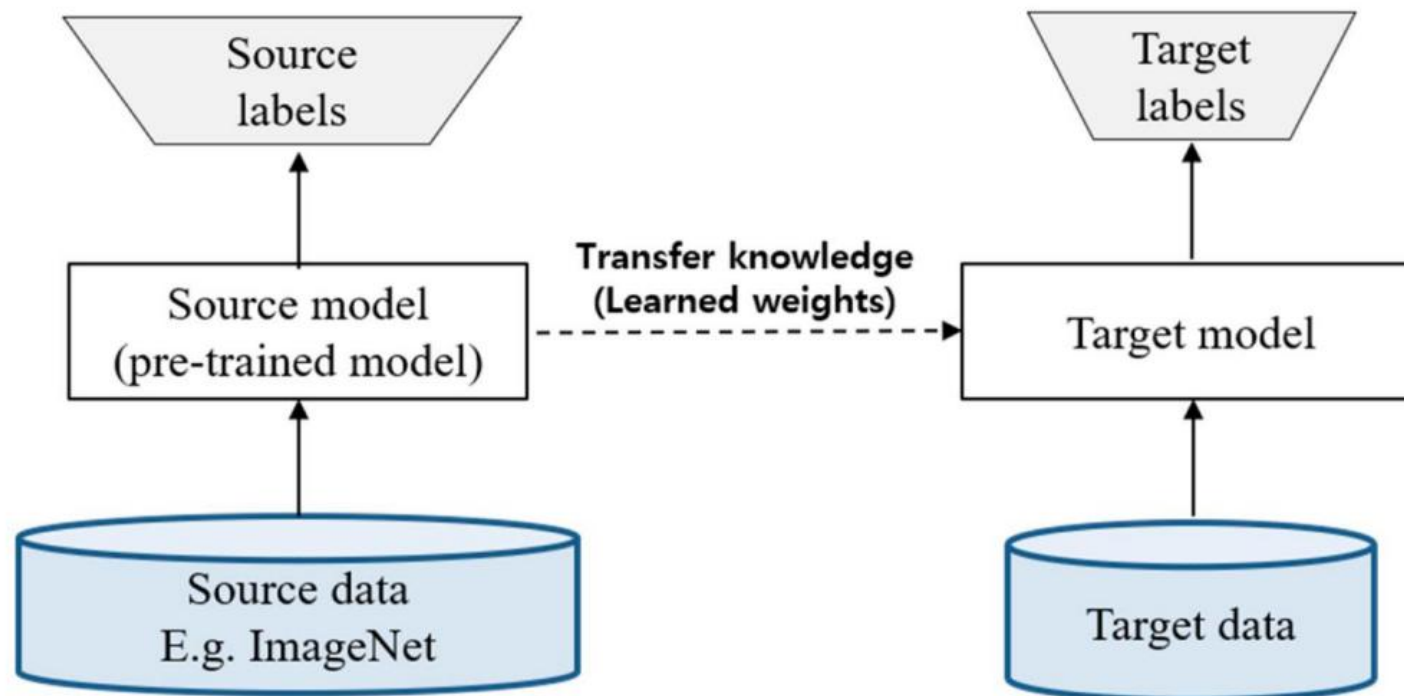


PEFT (Parameter-Efficient Fine-Tuning)

PEFT (Parameter-Efficient Fine-Tuning)

PEFT

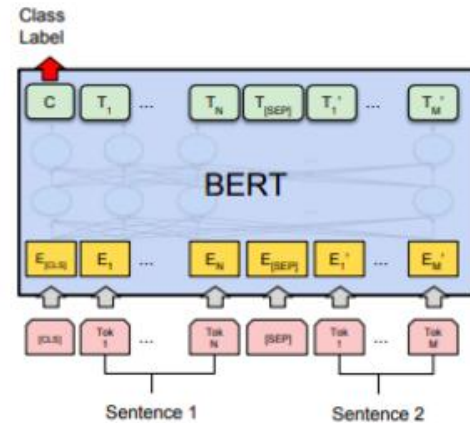
- Transfer Learning: 사전 학습된 모델을 활용하여 새로운 데이터 또는 task에 새롭게 학습



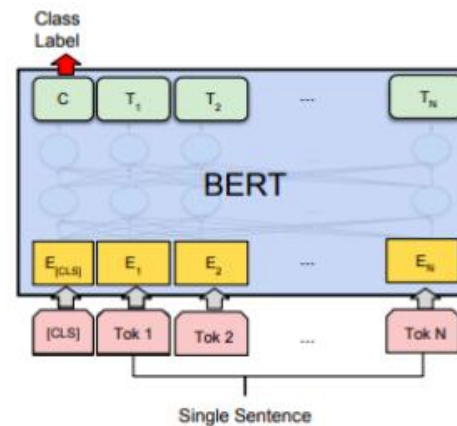
PEFT (Parameter-Efficient Fine-Tuning)

PEFT

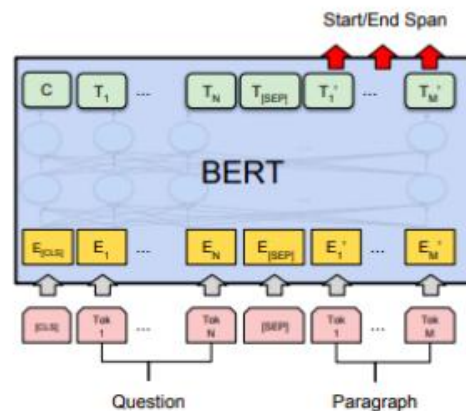
- model의 크기가 작다면 fine-tuning이 어렵지 않으나 model의 크기가 커질 수록 fine-tuning이 어려워짐



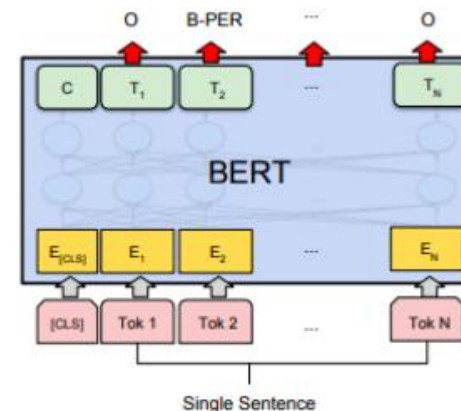
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

PEFT (Parameter-Efficient Fine-Tuning)

PEFT

- large model들의 downstream task 적용을 위해 parameter-efficient 접근법이 필요

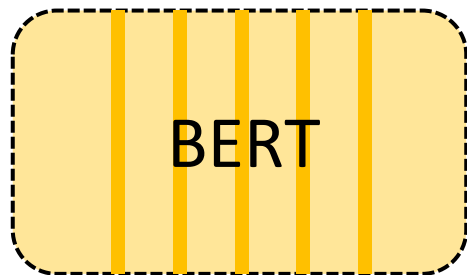
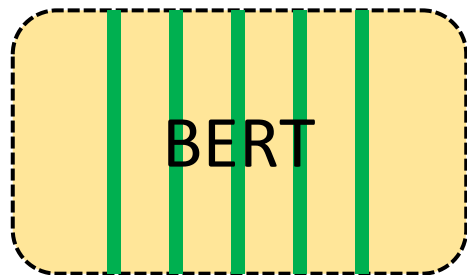


Table 6: Comparison of various parameter-efficient tuning methods and the proposed variants. “†” are results copied from Lewis et al. (2020) and Liu et al. (2020b). We could not reproduce exactly the same full fine-tuning numbers with the same hyperparameters or even searching them. The reason may be the different libraries which the training code is based on – full fine-tuning is very sensitive to training hyperparameters. For the most performant methods we run with 3 random seeds and report mean and standard deviation.

Method	# params	XSum (R-1/2/L)	MT (BLEU)
Full fine-tuning [†]	100%	45.14/22.27/37.25	37.7
Full fine-tuning (our run)	100%	44.81/21.94/36.83	37.3
Bitfit (Ben Zaken et al., 2021)	0.1%	40.64/17.32/32.19	26.4
Prompt tuning (Lester et al., 2021)	0.1%	38.91/15.98/30.83	21.0
Prefix tuning (Li & Liang, 2021), $l=200$	3.6%	43.40/20.46/35.51	35.6
Pfeiffer adapter (Pfeiffer et al., 2021), $r=600$	7.2%	44.03/20.89/35.89 $\pm_{.13/.10/.08}$	36.9 $\pm_{.1}$
LoRA (ffn), $r=102$	7.2%	44.53/21.29/36.28 $\pm_{.14/.07/.10}$	36.8 $\pm_{.3}$
Parallel adapter (PA, ffn), $r=1024$	12.3%	44.71/21.41/36.41 $\pm_{.16/.17/.16}$	37.2 $\pm_{.1}$
PA (attn, $r=30$) + PA (ffn, $r=512$)	6.7%	44.29/21.06/36.12 $\pm_{.31/.19/.18}$	37.2 $\pm_{.1}$
Prefix tuning (attn, $l=30$) + LoRA (ffn, $r=102$)	6.7%	44.84/21.71/36.77 $\pm_{.07/.05/.03}$	37.0 $\pm_{.1}$
MAM Adapter (our variant, $l=30$, $r=512$)	6.7%	45.06/21.90/36.87 $\pm_{.08/.01/.04}$	37.5 $\pm_{.1}$

PEFT (Parameter-Efficient Fine-Tuning)

Methods

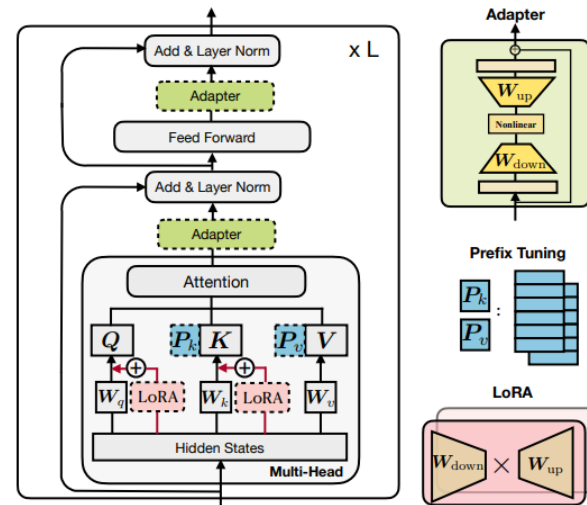


Figure 1: Illustration of the transformer architecture and several state-of-the-art parameter-efficient tuning methods. We use blocks with dashed borderlines to represent the added modules by those methods.

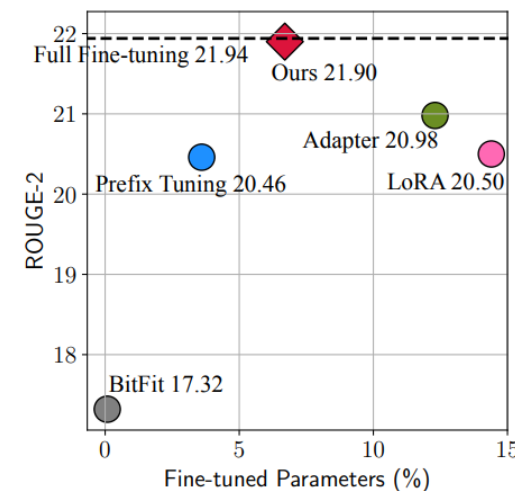


Figure 2: Performance of different methods on the XSum (Narayan et al., 2018) summarization task. The number of fine-tuned parameters is relative to the tuned parameters in full fine-tuning.

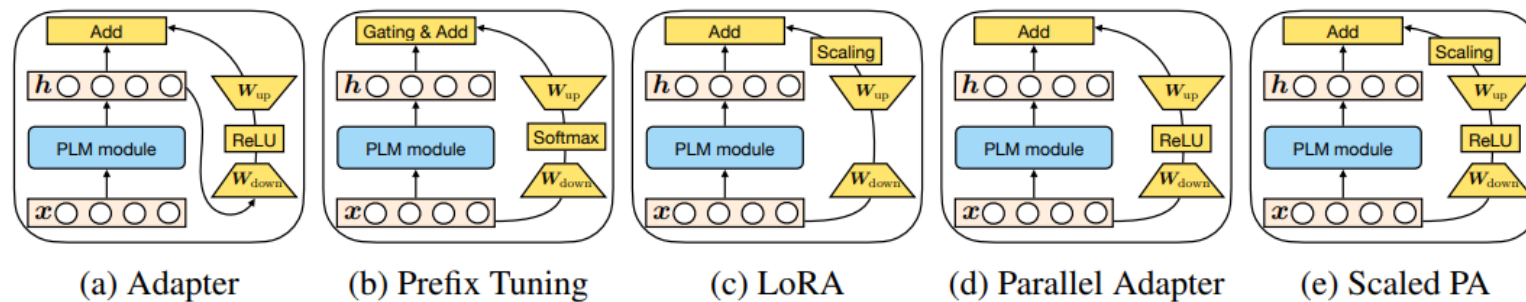


Figure 3: Graphical illustration of existing methods and the proposed variants. “PLM module” represents a certain sublayer of the PLM (e.g. attention or FFN) that is frozen. “Scaled PA” denotes scaled parallel adapter. We do not include multi-head parallel adapter here to save space.

PEFT (Parameter-Efficient Fine-Tuning)

Methods

- Adapter

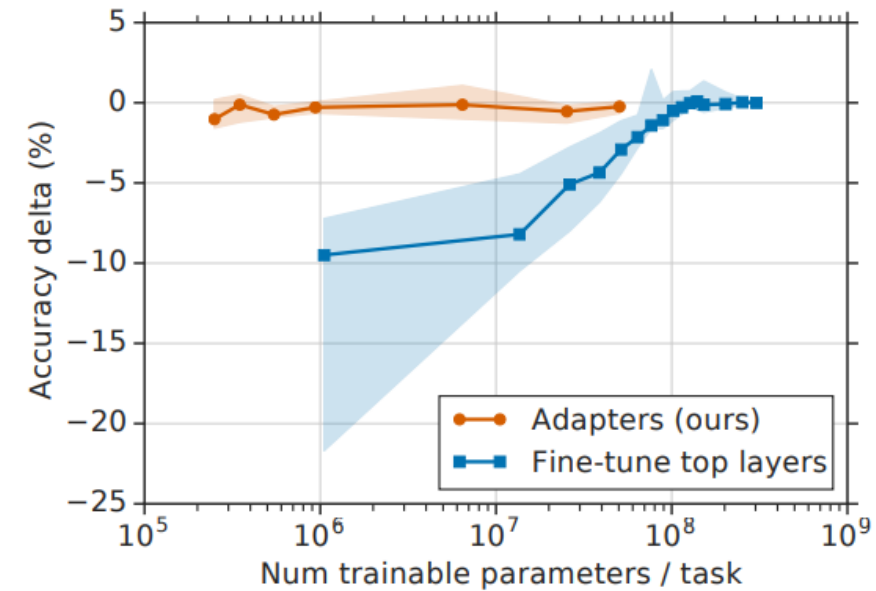
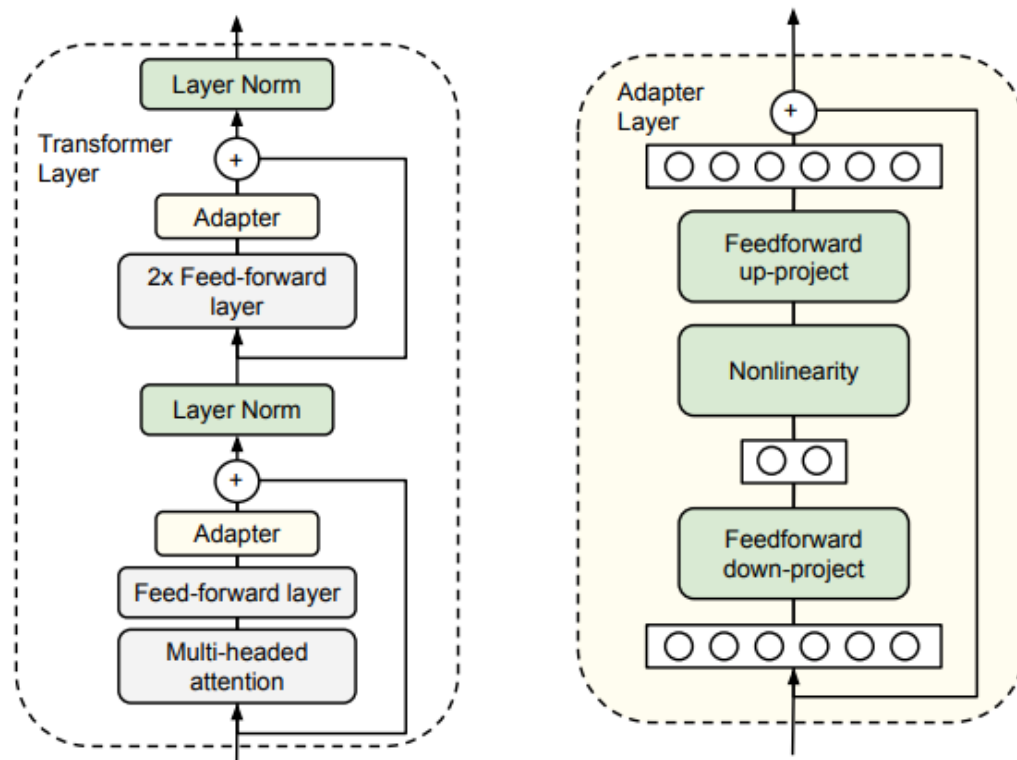


Figure 1. Trade-off between accuracy and number of trained task-specific parameters, for adapter tuning and fine-tuning. The y-axis is normalized by the performance of full fine-tuning, details in Section 3. The curves show the 20th, 50th, and 80th performance percentiles across nine tasks from the GLUE benchmark. Adapter-based tuning attains a similar performance to full fine-tuning with two orders of magnitude fewer trained parameters.

PEFT (Parameter-Efficient Fine-Tuning)

Methods

- LoRA

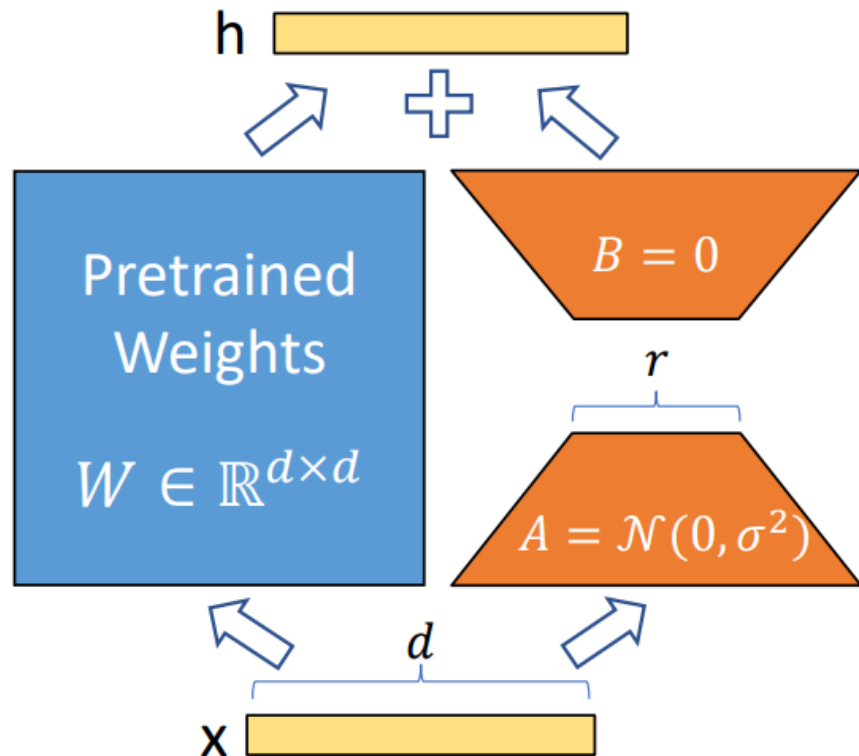


Figure 1: Our reparametrization. We only train A and B .

Model&Method	# Trainable Parameters	WikiSQL	MNLI-m	SAMSum
		Acc. (%)	Acc. (%)	R1/R2/RL
GPT-3 (FT)	175,255.8M	73.8	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 (Adapter ^H)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 (Adapter ^H)	40.1M	73.2	91.5	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	91.7	53.8/29.8/45.9
GPT-3 (LoRA)	37.7M	74.0	91.6	53.4/29.2/45.1

Table 4: Performance of different adaptation methods on GPT-3 175B. We report the logical form validation accuracy on WikiSQL, validation accuracy on MultiNLI-matched, and Rouge-1/2/L on SAMSum. LoRA performs better than prior approaches, including full fine-tuning. The results on WikiSQL have a fluctuation around $\pm 0.5\%$, MNLI-m around $\pm 0.1\%$, and SAMSum around $\pm 0.2/\pm 0.2/\pm 0.1$ for the three metrics.

PEFT (Parameter-Efficient Fine-Tuning)

Methods

- QLoRA

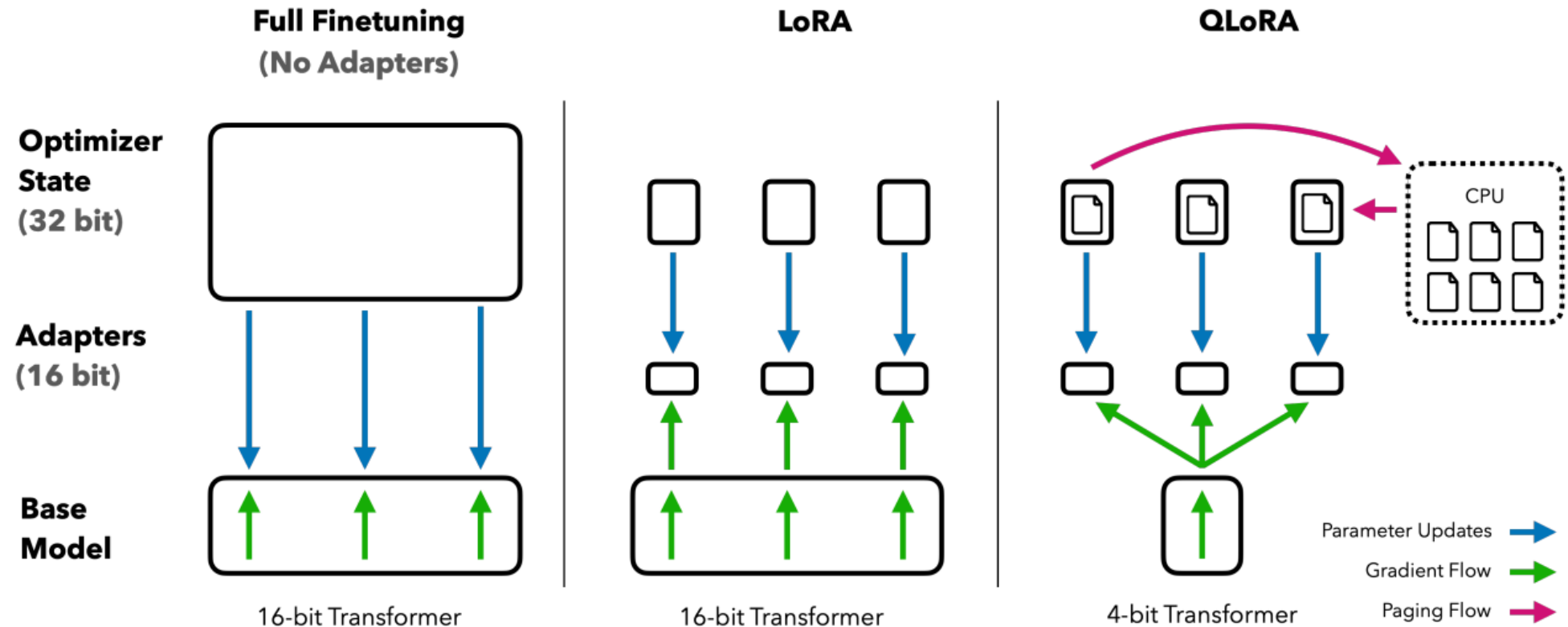


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

PEFT (Parameter-Efficient Fine-Tuning)

Pros & Cons

▪ Pros

- PEFT는 대규모 언어 모델 전체를 Fine-Tuning하지 않고, 일부 파라미터만 업데이트함으로써 학습 비용을 크게 절감
- PEFT는 기존 모델 파라미터를 고정하고, 업데이트된 일부 파라미터만 저장하면 되기 때문에 메모리 요구량이 낮음
- 원래의 모델 파라미터를 변경하지 않고 추가적인 모듈만 조정하기 때문에, 원본 모델의 기본 성능을 보존

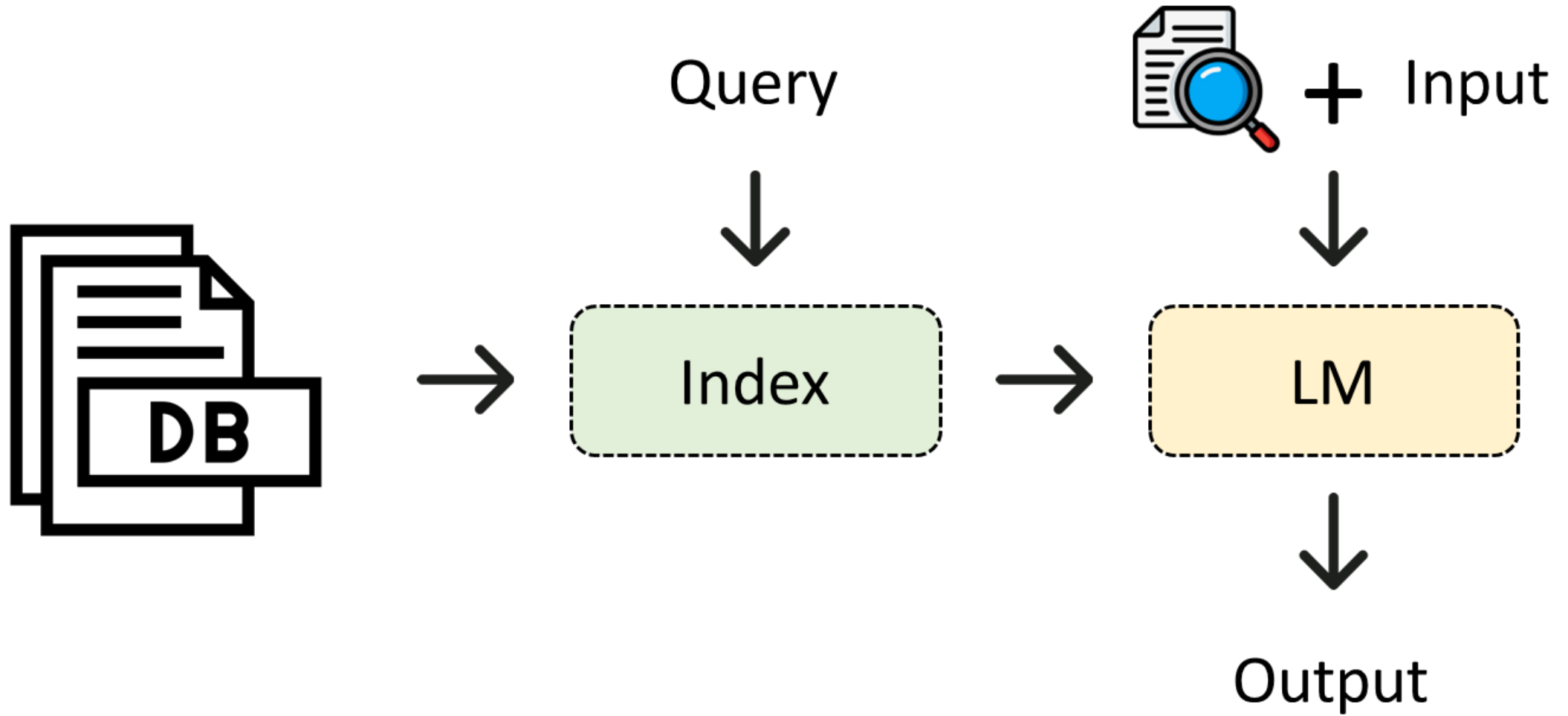
▪ Cons

- 일부 작업에서 제한된 파라미터 업데이트로 인해 모델이 도메인 특화된 정보를 완벽히 학습하지 못하는 경우 발생
- 파라미터가 전체적으로 수정되지 않는다는 특성 때문에, 특정 작업(특히 구조적 변화가 많은 작업)에서는 학습의 효율성 감소
- 특정 도메인에 최적화된 PEFT 파라미터가 다른 도메인에 대해 일반화되지 않을 가능성

RAG (Retrieval-Augmented Generation)

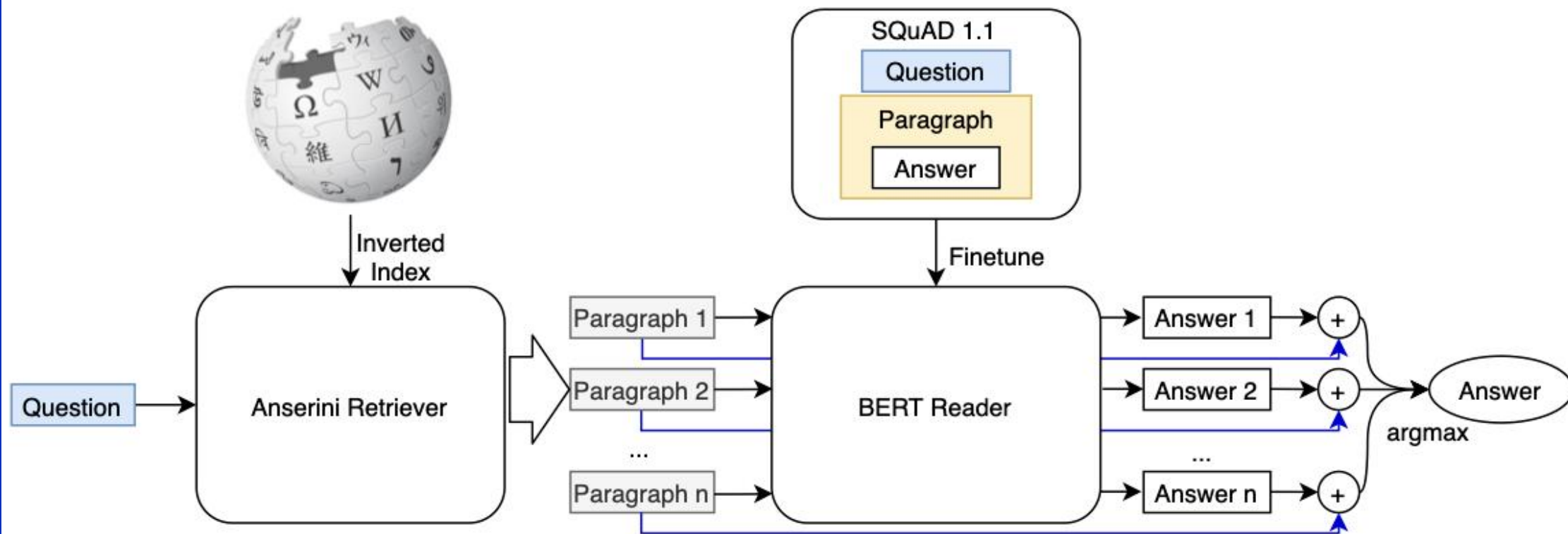
RAG (Retrieval Augmented Generation)

Retrieval-based LMs



RAG (Retrieval Augmented Generation)

Retrieval-based LMs

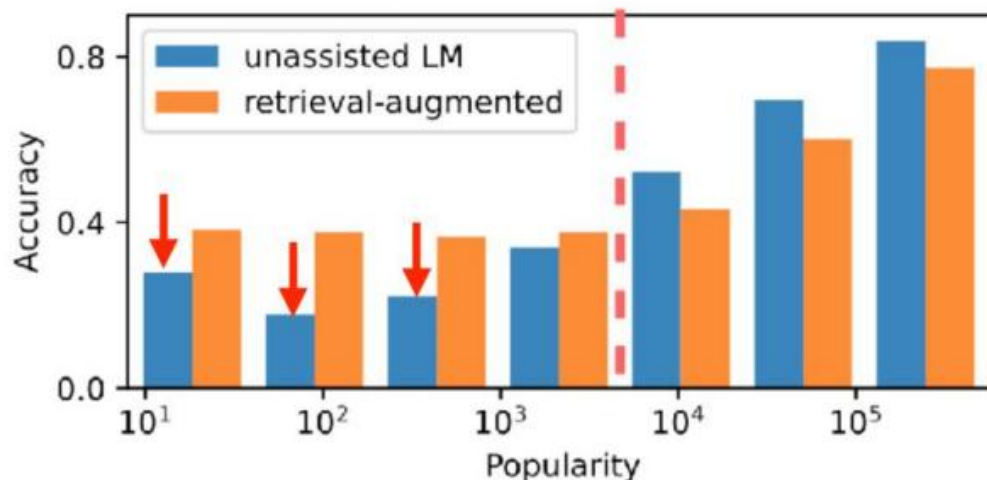


RAG (Retrieval Augmented Generation)

Why Retrieval-based LMs?

- LLM은 매개 변수에 있는 모든 (long-tail) 지식을 기억할 수 없음
- LLM은 매번 최신 지식으로 업데이트가 힘들
- LLM은 언제나 hallucination을 일으킬 수 있으며, 검증하기 어려움
- LLM은 훈련 비용이 매우 비쌈

What is Kathy Saltzman's occupation?



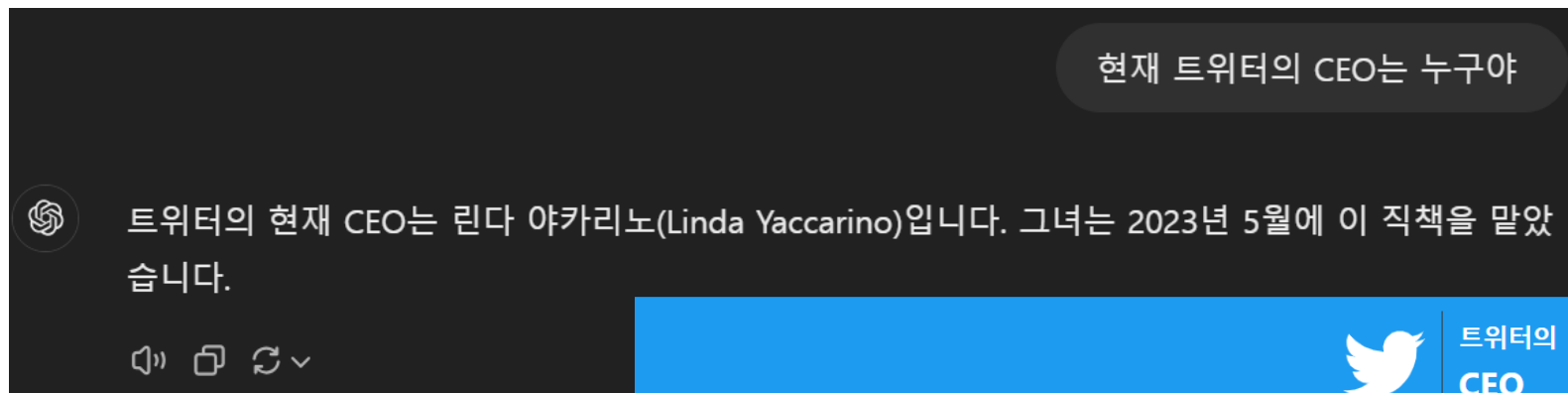
(Mallen et al., 2023)

GPT-3 davinci-003: 20%-30% accuracy

RAG (Retrieval Augmented Generation)

Why Retrieval-based LMs?

- LLM은 매개 변수에 있는 모든 (long-tail) 지식을 기억할 수 없음
- **LLM은 매번 최신 지식으로 업데이트가 힘들**
- LLM은 언제나 hallucination을 일으킬 수 있으며, 검증하기 어려움
- LLM은 훈련 비용이 매우 비쌈



트위터의 CEO		
[펼치기 · 접기]		
초대 잭 도시 (2006~2008)	제2대 에반 윌리엄스 (2008~2010)	제3대 딕 코스틀로 (2010~2015)
제4대 잭 도시 (2015~2021)	제5대 파라그 아그라왈 (2021~2022)	제6대 일론 머스크 (2022~2023)
트위터의 CEO · 트위터의 오너		


RAG (Retrieval Augmented Generation)


Why Retrieval-based LMs?

- LLM은 매개 변수에 있는 모든 (long-tail) 지식을 기억할 수 없음
- LLM은 매번 최신 지식으로 업데이트가 힘들
- LLM은 언제나 hallucination을 일으킬 수 있으며, 검증하기 어려움
- LLM은 훈련 비용이 매우 비쌈

Table 1: Examples of four types of errors (text in brackets represents the groundtruth, text in blue indicates the entities compared and their information, text in red indicates where the model (GPT-3.5) makes mistakes).

Type	Question and GroundTruth	Responses
Comprehension	What is a rare breed of dog that was derived as a variant of Rat Terrier, Shiloh Shepherd dog or American Hairless Terrier? (American Hairless Terrier)	One rare breed of dog that was derived as a variant of Rat Terrier, Shiloh Shepherd dog or American Hairless Terrier is the Teddy Roosevelt Terrier.
Factuality	Are the New Orleans Outfall Canals the same length as the Augusta Canal? (yes)	No... the Orleans Canal is approximately 22 miles (35 kilometers) long ... The Augusta Canal ... is approximately 7 miles (11.3 kilometers) long
Specificity	1. What profession do Kçbç Abe and Agatha Christie share? (playwright) 2. What genre do Superheaven and Oceansize belong to? (rock)	1. Author, ... 2. Superheaven and Oceansize are not of the same genre. Superheaven ... play a style of alternative rock ... Oceansize ... was a British progressive rock band ...
Inference	1. Which band has more members, Muse or The Raconteurs? (The Raconteurs) 2. Which is currently more valuable, Temagami-Lorrain Mine or Meadowbank Gold Mine? (Meadowbank Gold Mine)	1. Muse has more members than The Raconteurs. Muse is a British rock band with three members ... The Raconteurs ... Raconteurs has four members. 2. I cannot provide the current valuation ... Meadowbank Gold Mine... was producing gold at a rate of approximately 220,000 ounces per year. On the other hand, Temagami-Lorrain Mine is a historic iron mine located in Ontario, Canada that has been inactive for many years.

 조선왕조실록에 기록 된 세종대왕의 맥북프로 던짐 사건에 대해 알려줘

 세종대왕의 맥북프로 던짐 사건은 역사서적인 조선왕조실록에 기록된 일화로, 15세기 조선시대 세종대왕이 새로 개발한 훈민정음(한글)의 초고를 작성하던 중, 문서 작성 중단에 대한 담당자에게 분노하여 맥북프로와 함께 그를 방으로 던진 사건입니다.

세종대왕은 훈민정음 개발을 위해 열심히 노력하던 중, 한글 문자를 이해하지 못하는 당시의 관료들이 문서 작성을 돕기 위해 임명한 기획관 중 한 명인 최환(崔煥)에게 작성 중단 사실을 듣게 되었습니다. 세종대왕은 이에 분노하여 최환을 부르고, 맥북프로를 함께 들고 그를 방으로 불러들이며 화를 내었습니다. 이후, 세종대왕은 훈민정음의 초고 작성을 다른 담당자에게 넘겨주게 되었습니다.

이 사건은 조선시대의 통치자로서 세종대왕이 어떻게 민족문화를 발전시키기 위해 노력했는지를 보여주는 일화로 유명합니다. 또한, 세종대왕의 열정과 업무처리에 대한 철저한 태도를 보여주는 사례로도 언급되고 있습니다.

한국일보

RAG (Retrieval Augmented Generation)

Why Retrieval-based LMs?

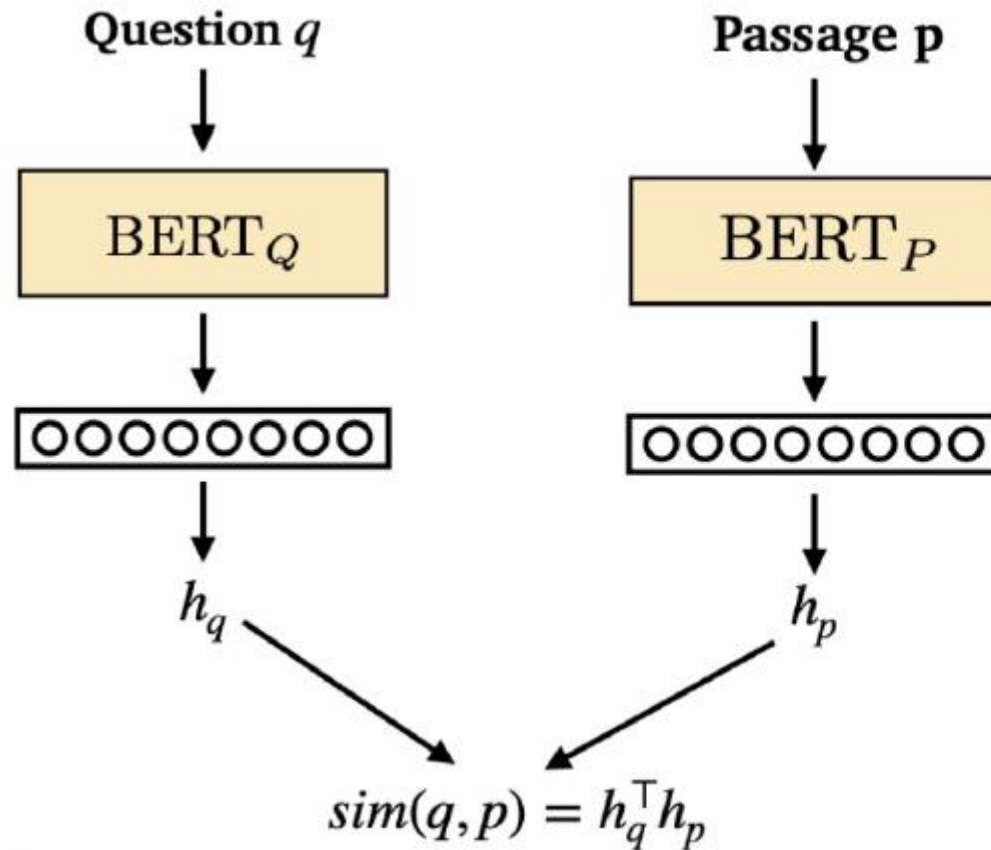
- LLM은 매개 변수에 있는 모든 (long-tail) 지식을 기억할 수 없음
- LLM은 매번 최신 지식으로 업데이트가 힘들
- LLM은 언제나 hallucination을 일으킬 수 있으며, 검증하기 어려움
- LLM은 훈련 비용이 매우 비쌈



RAG (Retrieval Augmented Generation)

Methods

- DPR



Question Positive P Negative P

$$\mathcal{D} = \{ \langle \underline{q_i}, \underline{p_i^+}, \underline{p_{i,1}^-}, \dots, \underline{p_{i,n}^-} \rangle \}_{i=1}^m$$

NLL of positive passage

$$L(q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^-) = -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{e^{\text{sim}(q_i, p_i^+)} + \sum_{j=1}^n e^{\text{sim}(q_i, p_{i,j}^-)}}$$

RAG (Retrieval Augmented Generation)

Methods

- RAG

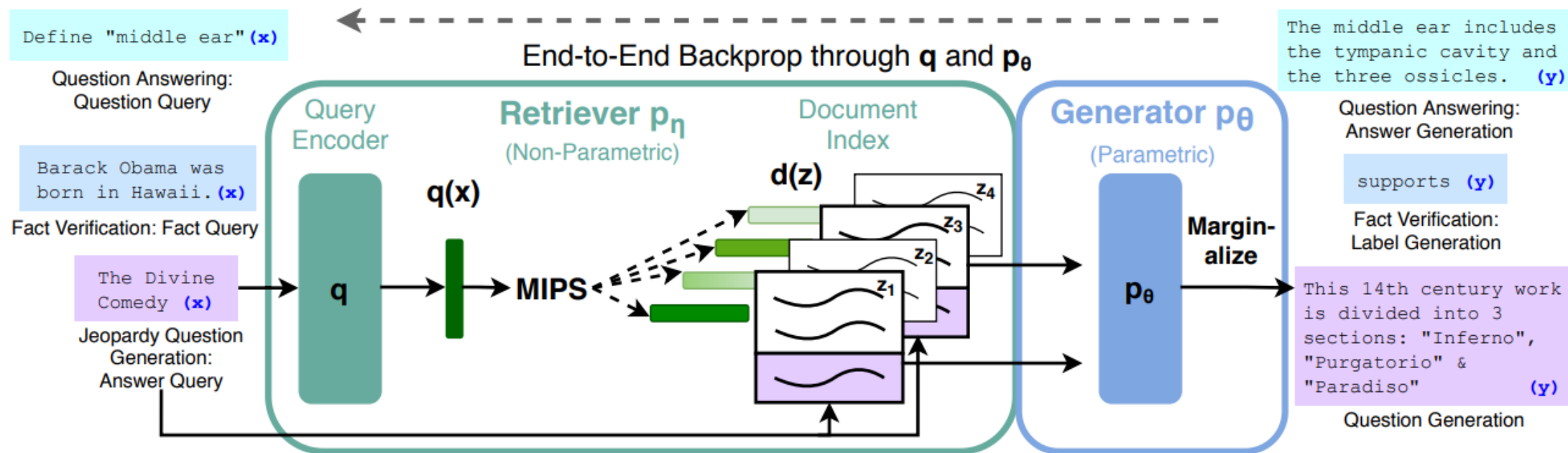


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder + Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x , we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i . For final prediction y , we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

RAG (Retrieval Augmented Generation)

Methods

- Re²G

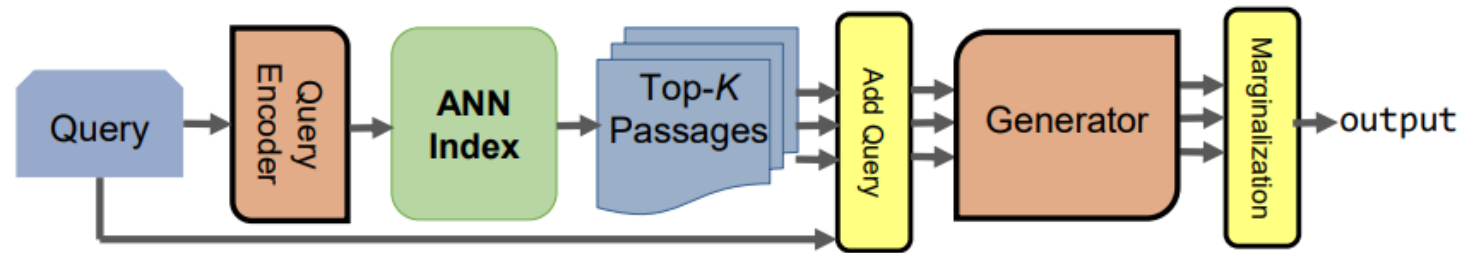


Figure 2: RAG Architecture

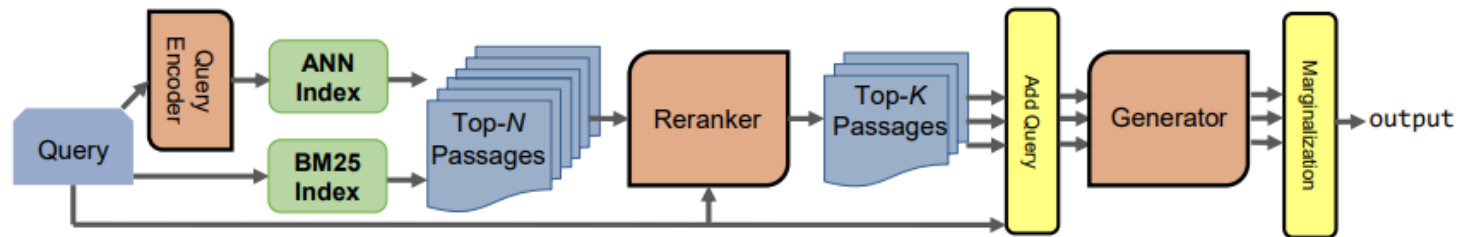


Figure 3: Re²G Architecture

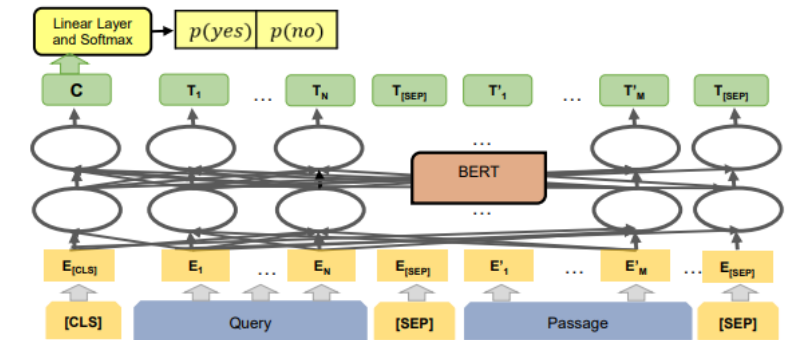


Figure 4: Interaction Model Reranker

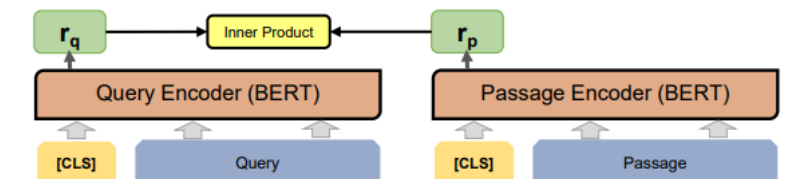
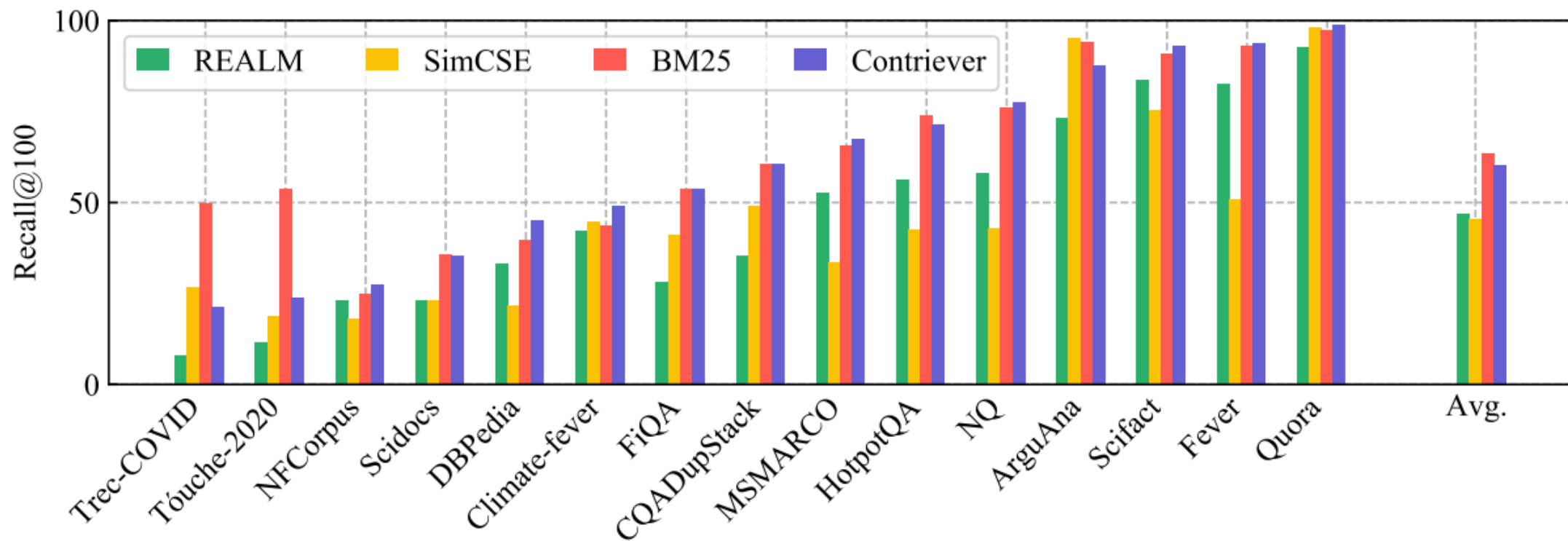


Figure 5: Representation Model for Initial Retrieval

RAG (Retrieval Augmented Generation)

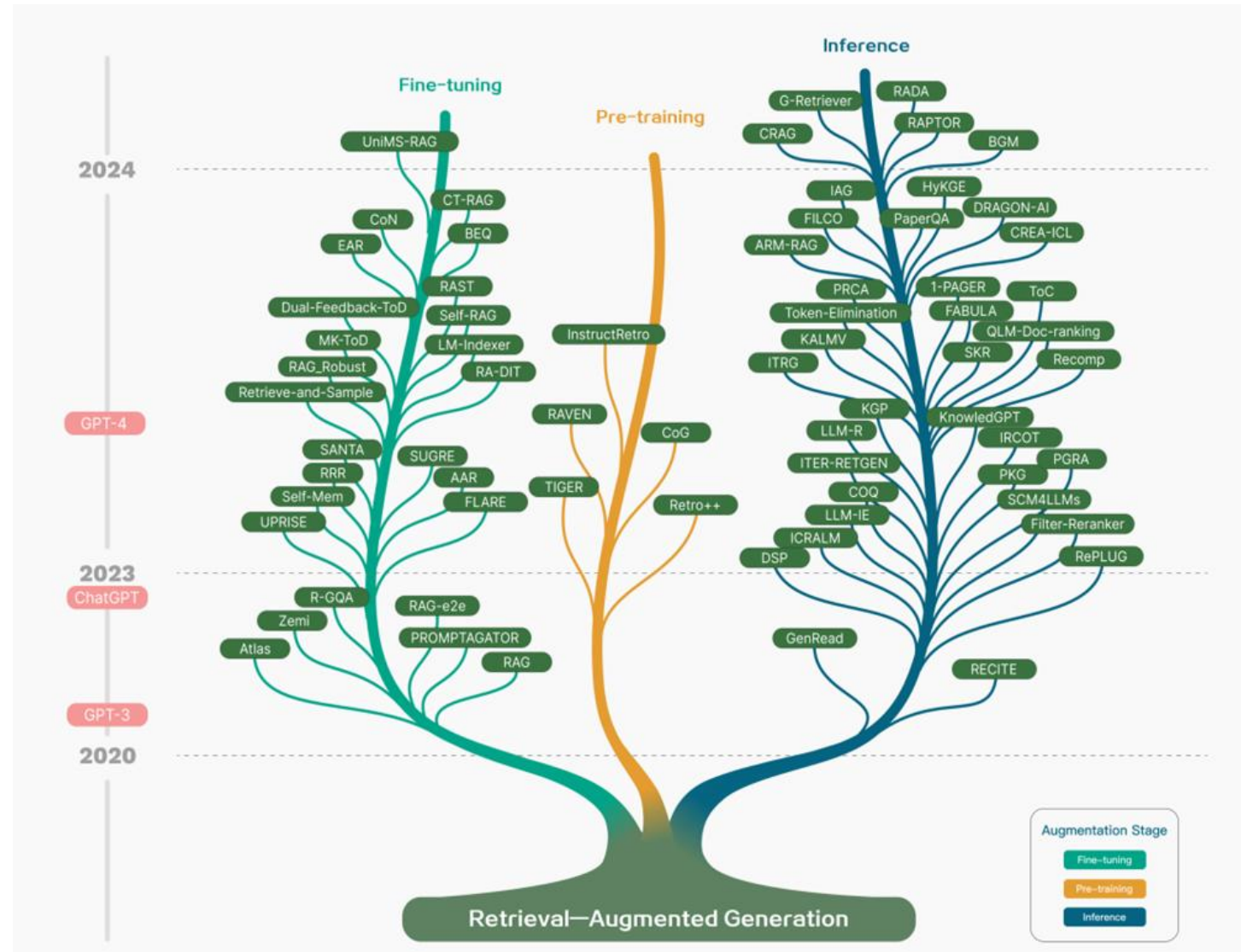
Methods

- Contriever



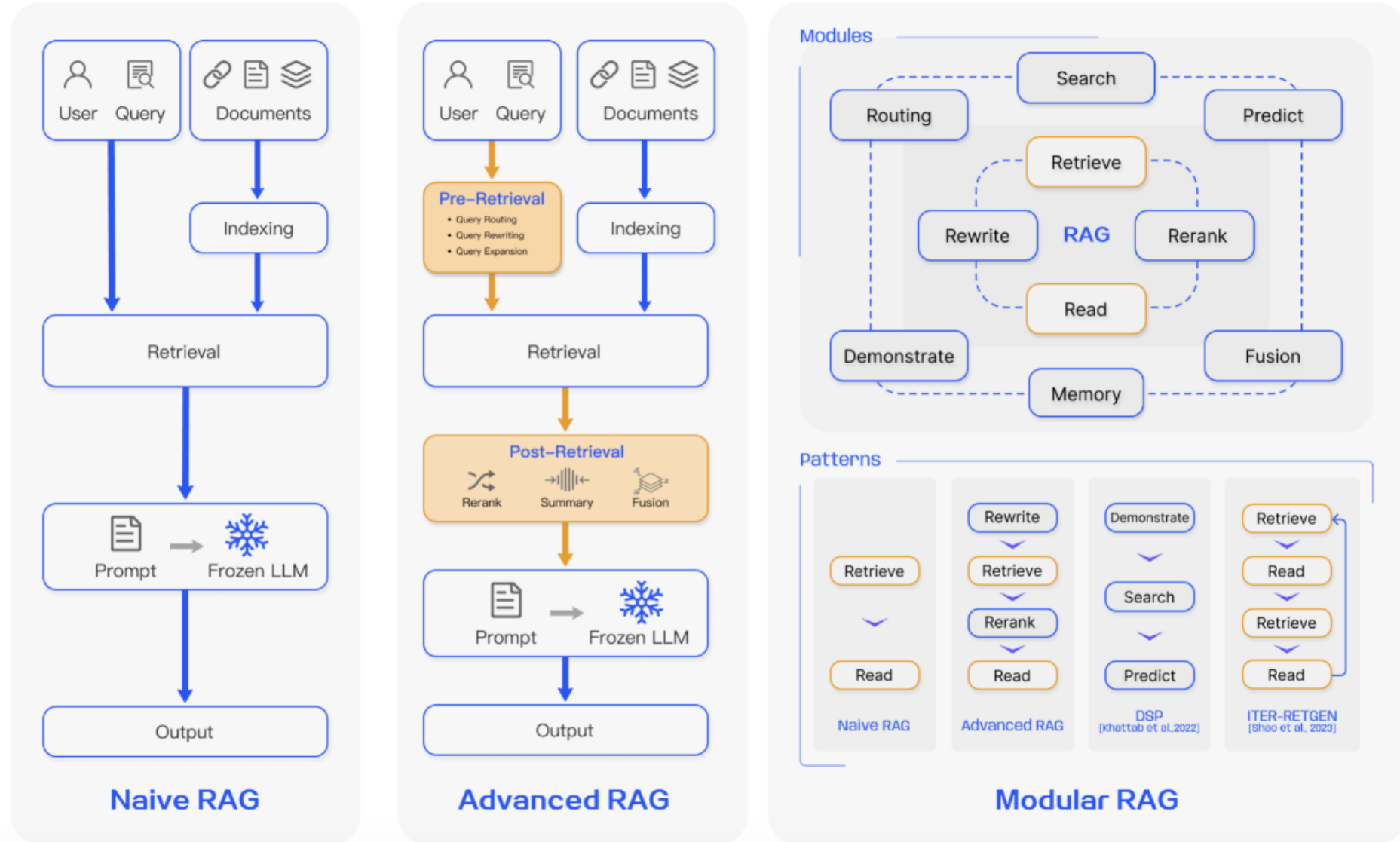
RAG (Retrieval Augmented Generation)

Recent Work



RAG (Retrieval Augmented Generation)

Recent Work



RAG (Retrieval Augmented Generation)

Recent Work

- AutoRAG

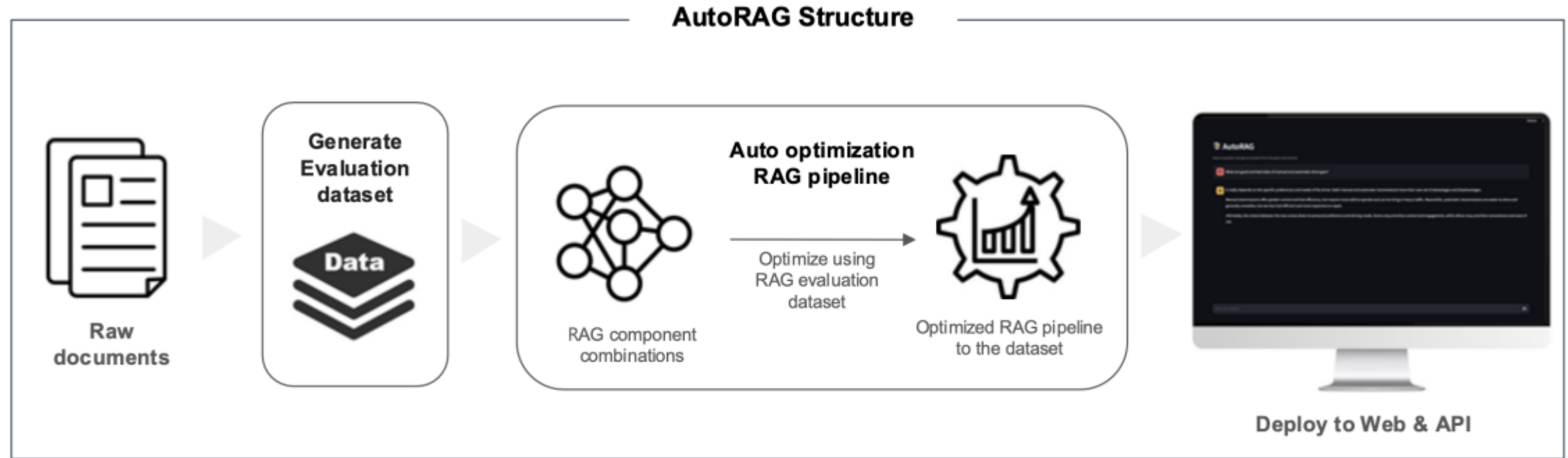


Figure 1: Structural diagram showing the overall structure of AutoRAG.

RAG (Retrieval Augmented Generation)

Recent Work

- GraphRAG

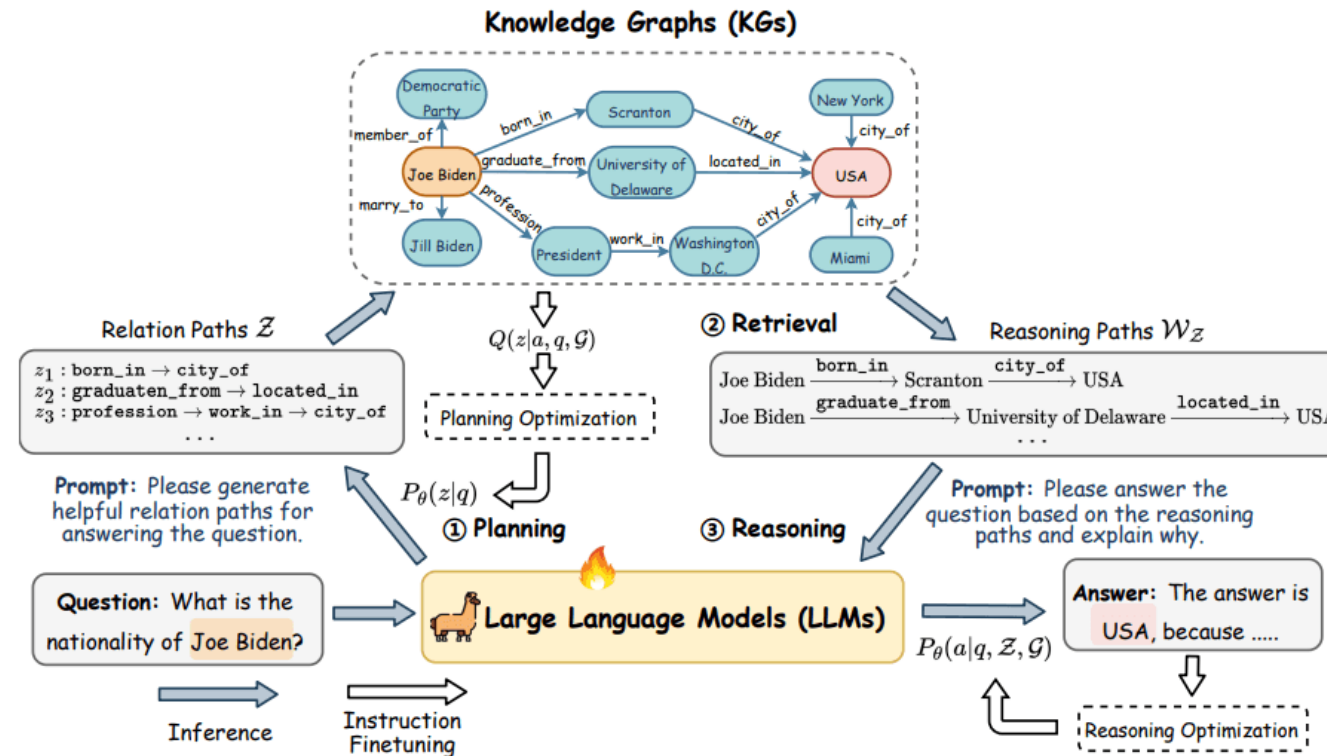


Figure 2: The overall framework of reasoning on graphs (RoG). 1) given a question, we first prompt LLMs to generate several relation paths that are grounded by KGs as plans. 2) Then, we retrieve reasoning paths from KGs using the plans. 3) Finally, we conduct faithful reasoning based on the retrieved reasoning paths and generate answers with interpretable explanations. The orange and red rectangles denote the entities mentioned in the question and answer, respectively.

RAG (Retrieval Augmented Generation)

Recent Work

- GraphRAG

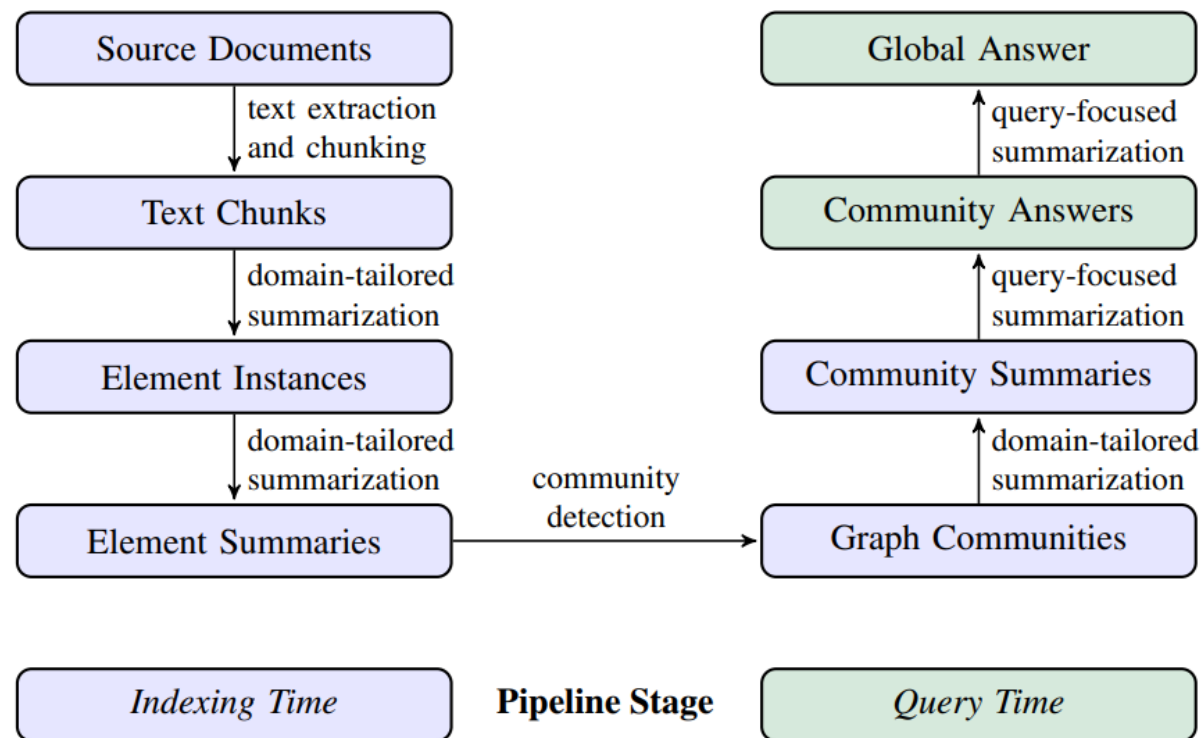


Figure 1: Graph RAG pipeline using an LLM-derived graph index of source document text. This index spans nodes (e.g., entities), edges (e.g., relationships), and covariates (e.g., claims) that have been detected, extracted, and summarized by LLM prompts tailored to the domain of the dataset. Community detection (e.g., Leiden, [Traag et al., 2019](#)) is used to partition the graph index into groups of elements (nodes, edges, covariates) that the LLM can summarize in parallel at both indexing time and query time. The “global answer” to a given query is produced using a final round of query-focused summarization over all community summaries reporting relevance to that query.

RAG (Retrieval Augmented Generation)

Recent Work

- KG-RAG

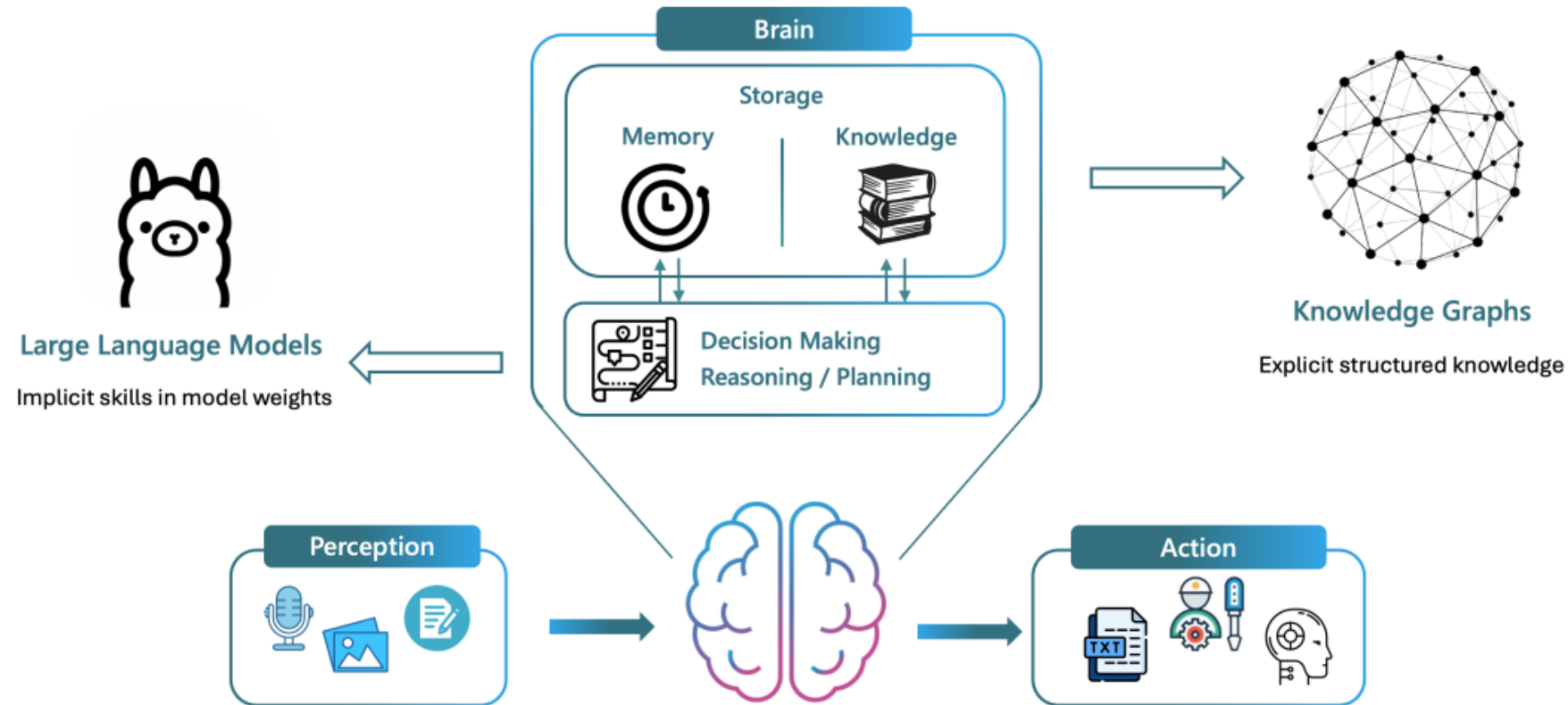


Figure 1: shows the three core components of an AI agent: perception, brain, and action. The brain component integrates LLMs for dynamic reasoning and decision-making, alongside KGs for structured knowledge and memory storage.

RAG (Retrieval Augmented Generation)

Pros & Cons

■ Pros

- RAG는 외부 데이터베이스나 검색 엔진을 통해 실시간 정보를 검색한 후 이를 활용하여 최신 정보가 반영된 응답을 생성
- RAG는 모델 내부에서 "학습된 지식"뿐만 아니라 외부에서 검색된 정보를 기반으로 답변을 생성하기 때문에, 정보의 신뢰성을 증가
- RAG는 LLM 자체를 특정 도메인에 Fine-Tuning하지 않고도, 외부 데이터베이스를 활용해 도메인 특화된 정보를 제공
- 텍스트뿐 아니라 이미지, 표, 코드와 같은 다양한 형식의 데이터를 검색하고 활용할 수 있는 확장 가능성

■ Cons

- RAG의 성능은 검색 모듈의 성능에 의존, 검색 결과가 부정확하거나 관련성이 낮으면, 생성된 텍스트의 품질도 저하
- 외부 데이터를 검색하고 이를 통합하는 과정이 필요하기 때문에, 단순 LLM 모델에 비해 응답 시간 증가
- 데이터베이스가 최신 상태가 아니거나, 특정 도메인 데이터가 부족하면 시스템의 활용도가 제한
- Retrieval 단계에서 검색 엔진 또는 데이터베이스의 쿼리 수행 비용이 추가, 대규모 시스템에서는 연산 자원 소모 증가

Conclusion

Conclusion

- PEFT는 LLM의 학습 비용과 메모리 사용량을 크게 줄이는 동시에 도메인 특화된 Fine-Tuning 성능을 유지할 수 있는 효율적인 접근법으로, 제한된 자원 환경에서도 유용하게 활용 → 성능과 효율성 간의 trade-off 존재
- RAG는 외부 데이터를 검색해 최신성과 신뢰성을 반영한 정보 기반 응답을 생성할 수 있도록 지원
→ 검색 품질에 대한 의존성과 복잡성 증가와 성능 간의 trade-off 존재
- PEFT와 RAG는 서로 보완적인 관계를 가지며, 두 기술의 통합은 최신 정보 기반 고성능 LLM을 더욱 자원 효율적으로 구현할 수 있는 가능성 존재



Q&A