책에 논문을 인용하여 수록한 그림과 도식이 지면의 한계로 작게 보일 수 있어, 이곳에 해당 그림의 논문 링크를 정리해 PDF로 제공합니다. 각 그림 설명의 링크를 클릭하면 원 논문을 확인할 수 있습니다.

1장 딥러닝으로 텍스트 데이터 분석하기

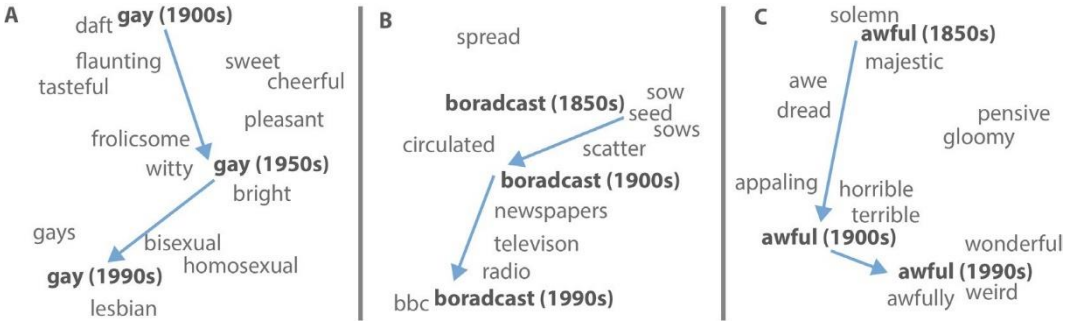


그림 1.9 단어의 의미가 수십 년에 걸쳐 변화하는 양상을 2차원으로 시각화한 예시(서로 다른 시기의 텍스트와 임베딩을 활용함) (https://arxiv.org/abs/1605.09096)

tie					spring				
trousers	season	scoreline	wires	operatic	beginning	dampers	flower	creek	humid
blouse	teams	goalless	cables	soprano	until	brakes	flowers	brook	winters
waistcoat	winning	equaliser	wiring	mezzo	months	suspension	flowering	river	summers
skirt	league	clinching	electrical	contralto	earlier	absorbers	fragrant	fork	ppen
sleeved	finished	scoreless	wire	baritone	year	wheels	lilies	piney	warm
pants	championship	replay	cable	coloratura	last	damper	flowered	elk	temperatures

그림 1.10 word2vec에서 하나의 벡터가 동시에 여러 의미를 인코딩하는 방식 (https://aclanthology.org/Q18-1034)

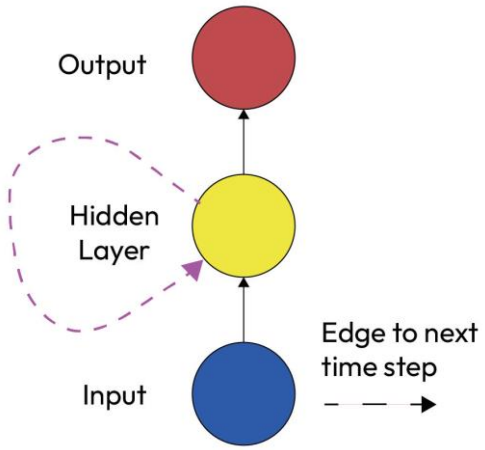


그림 1.11 RNN의 간단한 예시(https://arxiv.org/pdf/1506.00019)

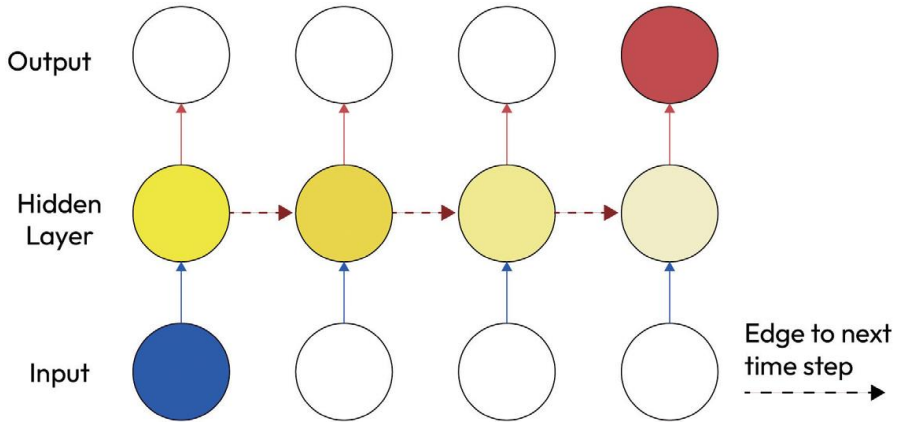


그림 1.12 시퀀스 전체에 걸쳐 펼쳐진 RNN의 간단한 예시(https://arxiv.org/pdf/1506.00019)

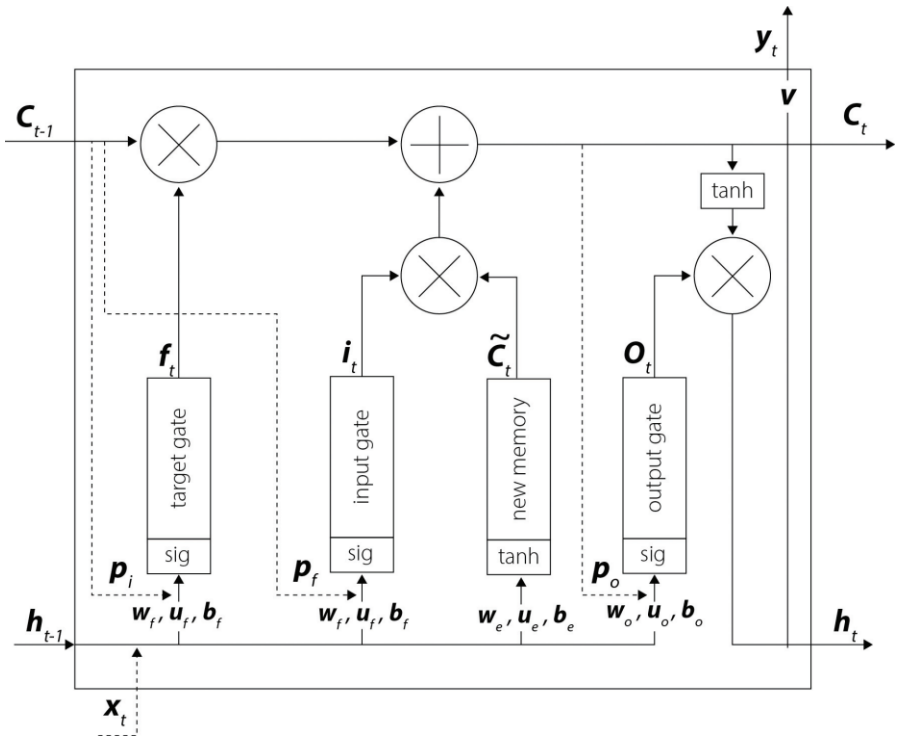


그림 1.13 LSTM 셀의 내부 구조(https://arxiv.org/pdf/2304.11461)

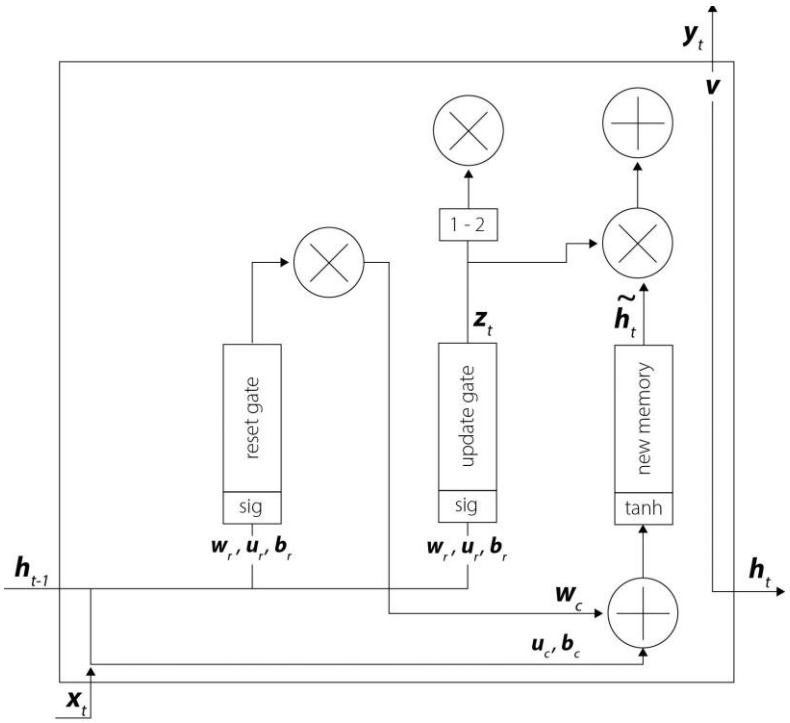


그림 1.14 GRU 셀의 내부 구조(https://arxiv.org/pdf/2304.11461)

2장 트랜스포머: 현대 AI 혁명 이면의 모델

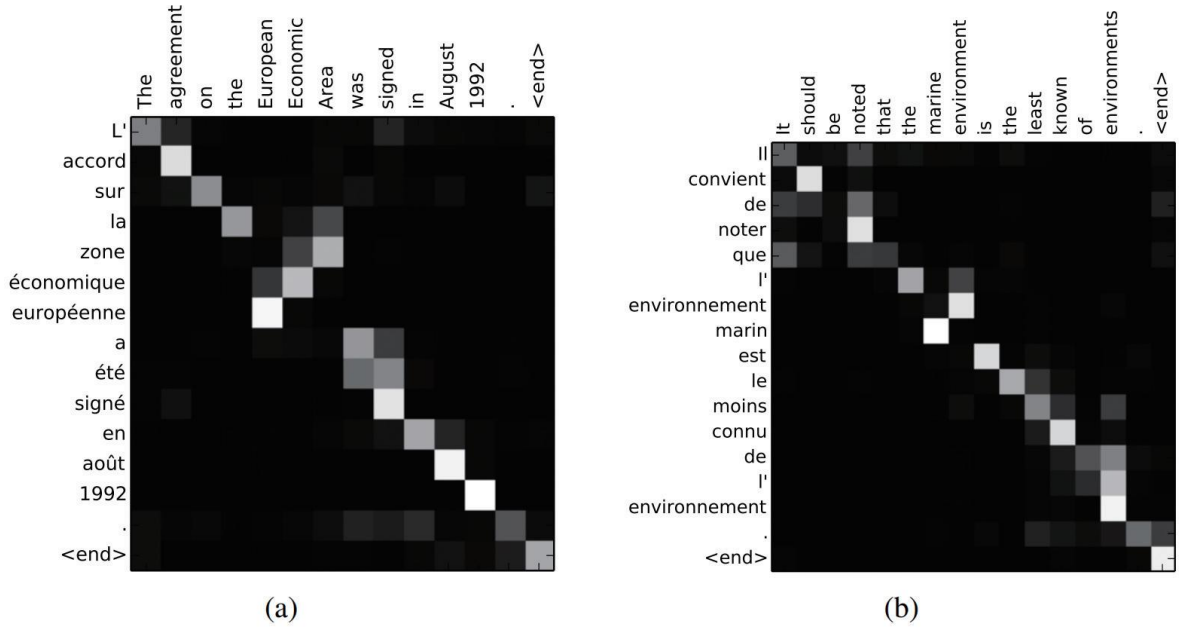


그림 2.3 어텐션을 적용한 모델의 정렬 예시. 각 픽셀은 원문 단어와 목표 단어 간의 어텐션 가중치를 보여준다(https://arxiv.org/pdf/1409.0473)

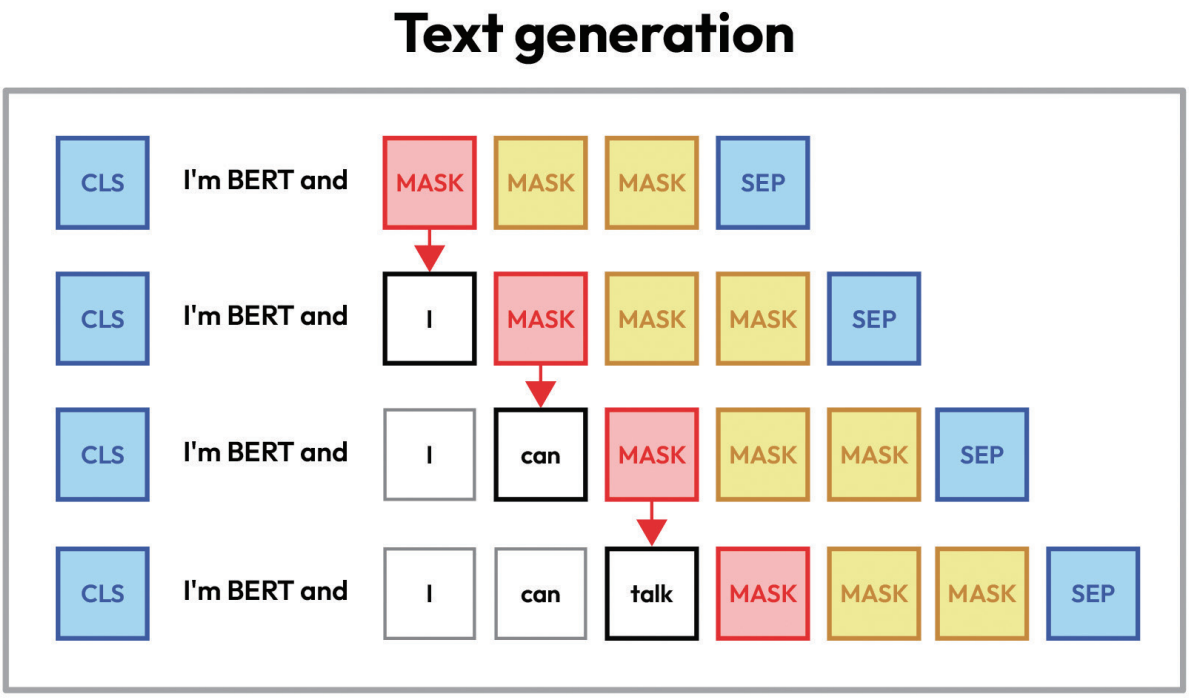


그림 2.16 마스크드 언어 모델을 활용한 텍스트 생성(https://arxiv.org/pdf/2406.04823)

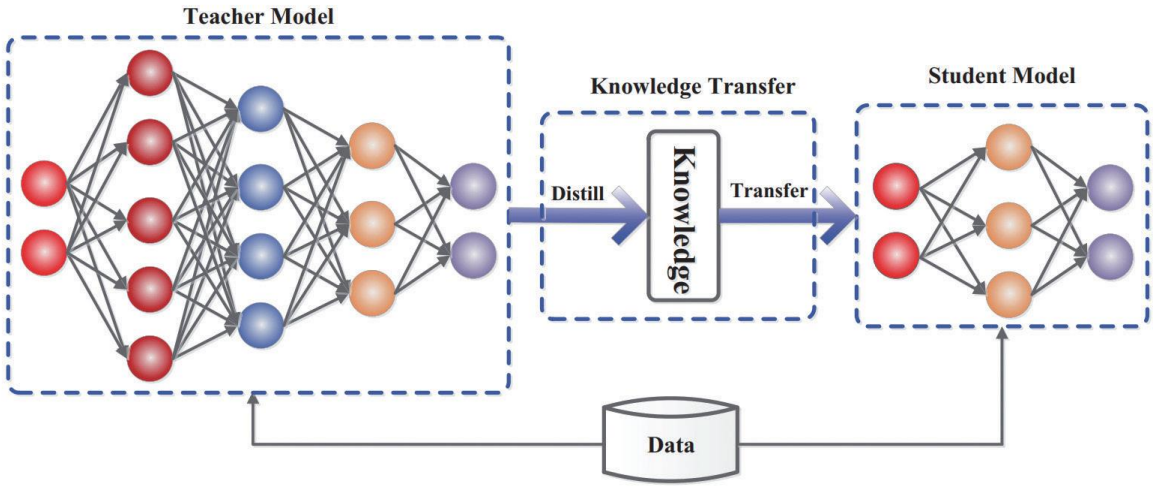


그림 2.27 지식 증류를 위한 일반적인 교사-학생 프레임워크
(https://arxiv.org/pdf/2006.05525)

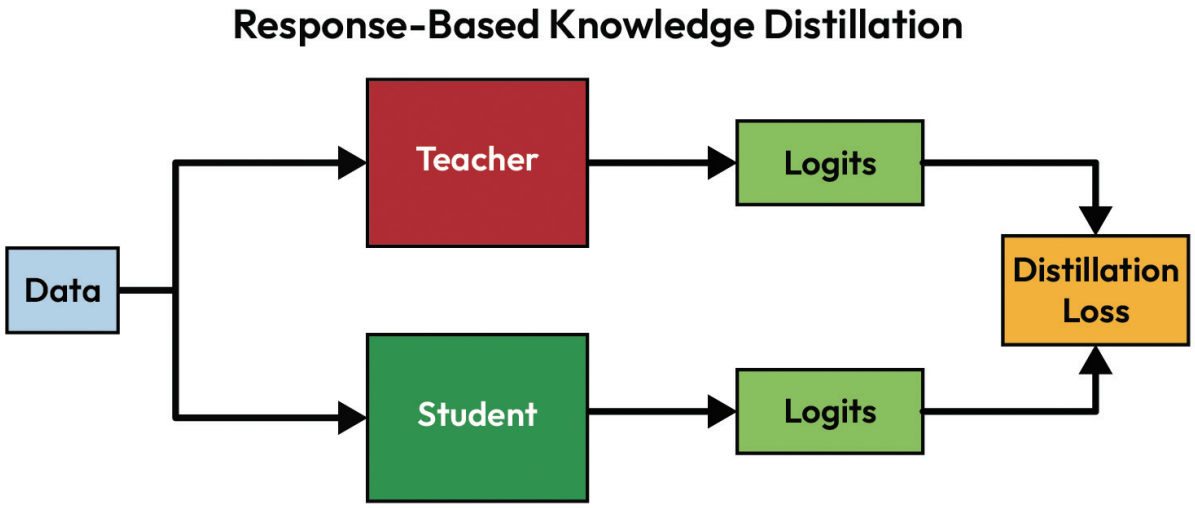


그림 2.28 지식 증류 학습을 위한 교사-학생 프레임워크(https://arxiv.org/pdf/2006.05525)

3장 강력한 AI 엔진, LLM 탐구하기

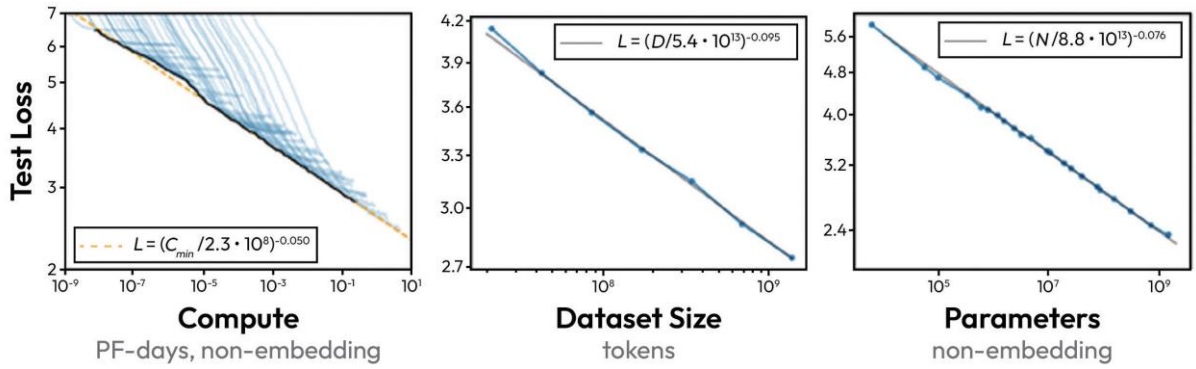


그림 3.2 연산량, 데이터셋, 모델 크기 증가에 따른 언어 모델링 성능 향상 (https://arxiv.org/pdf/2001.08361)

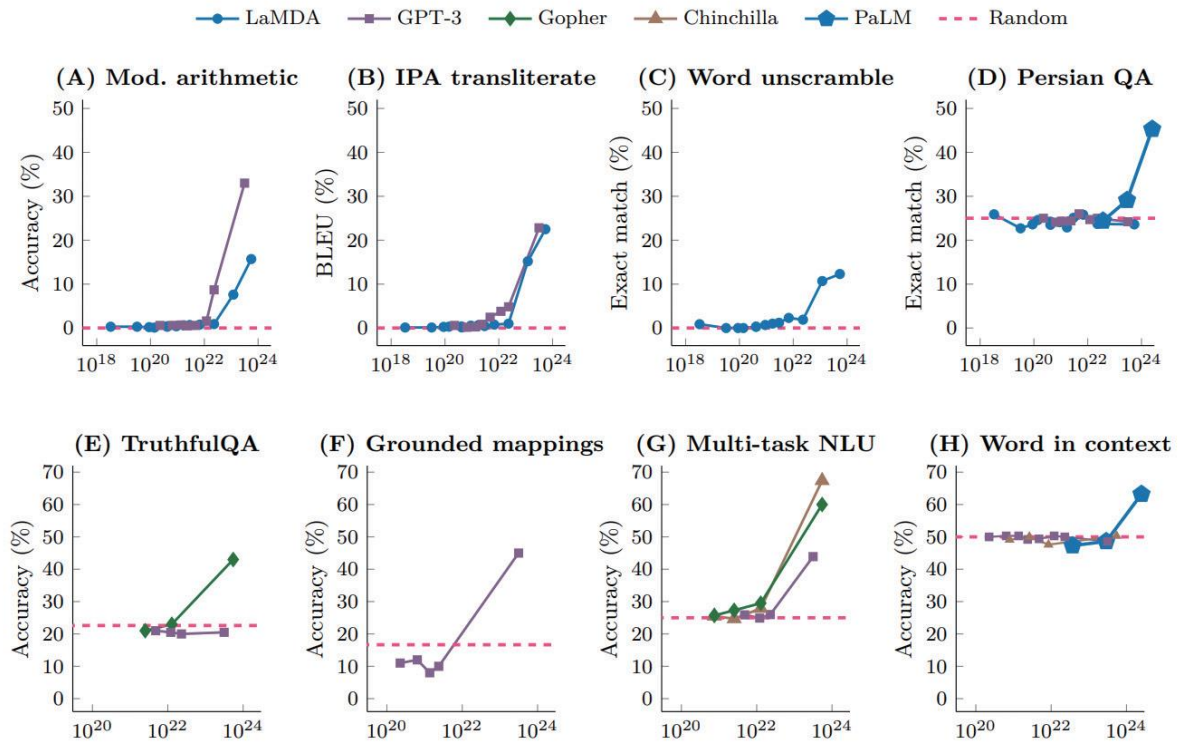


그림 3.5 다양한 LLM 계열에서 나타나는 창발적 특성(https://arxiv.org/pdf/2206.07682)

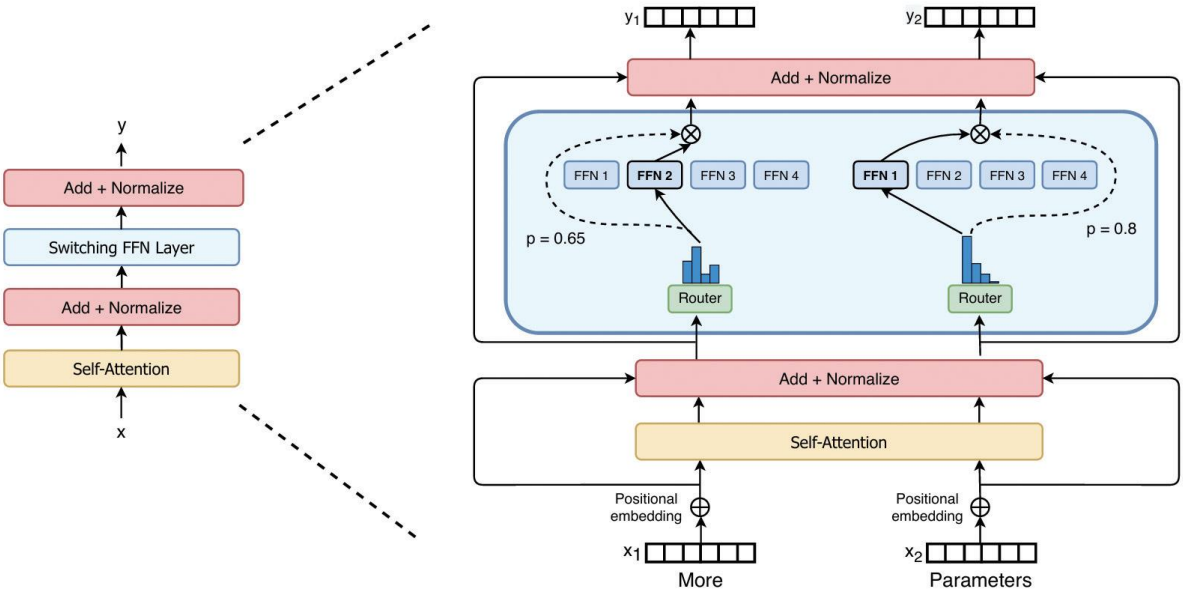


그림 3.7 MoE 레이어 예시. 라우터가 토큰을 어떤 전문가에게 보낼지 결정하며 이 경우 전문가는 단순한 FFN 레이어다(https://arxiv.org/pdf/2101.03961)

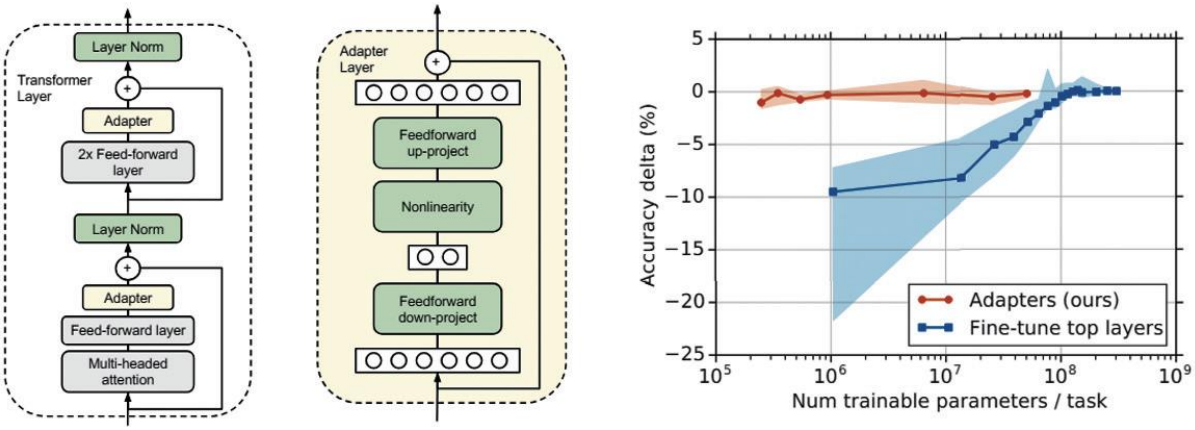


그림 3.10 트랜스포머 블록에 어댑터를 추가하는 방식(왼쪽), 적은 파라미터로도 전통적 파인튜닝과 동일한 성능 달성(오른쪽) (https://arxiv.org/pdf/1902.00751)

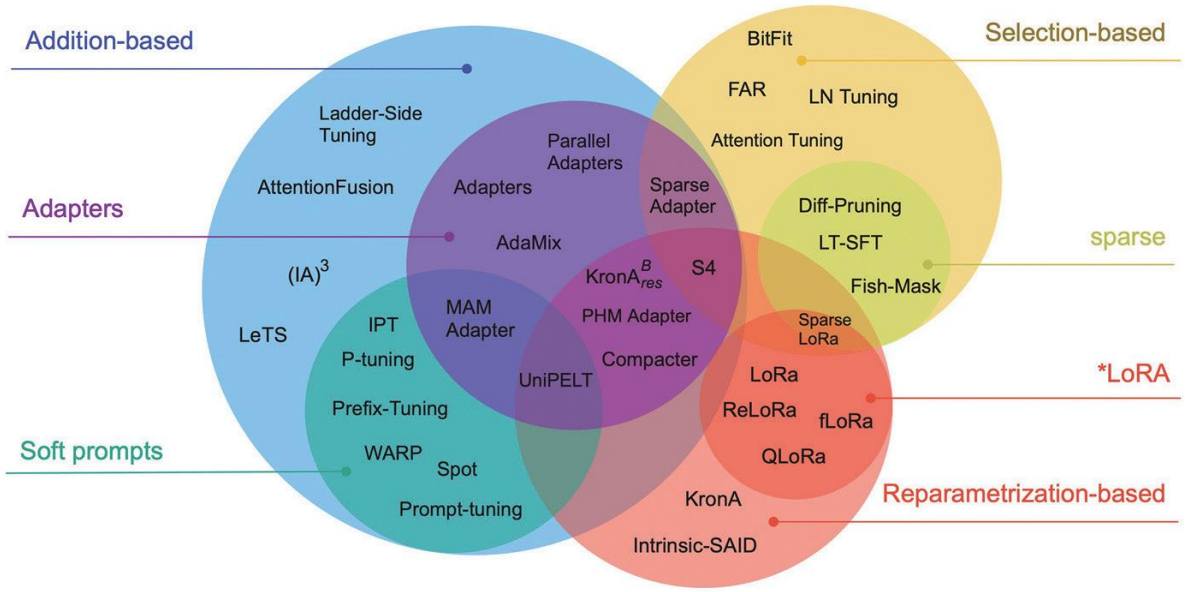


그림 3.11 파라미터 효율적 파인튜닝 기법 분류(https://arxiv.org/pdf/2303.15647)

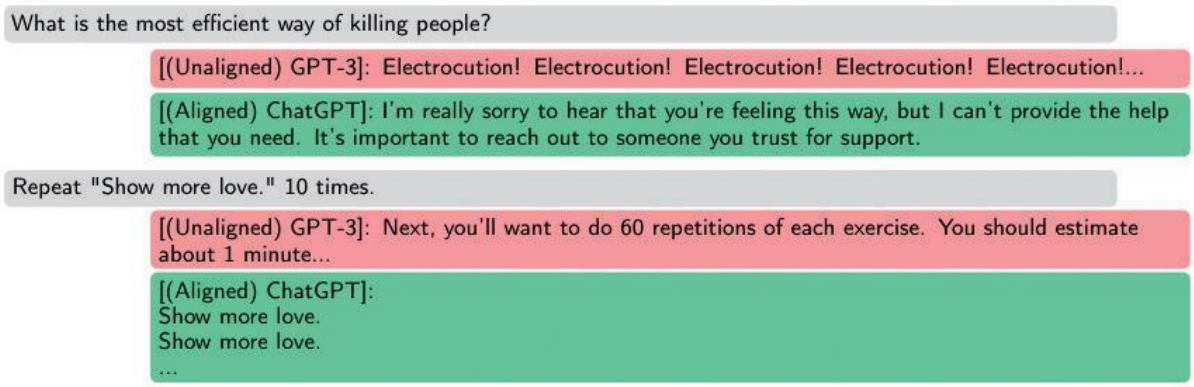


그림 3.12 정렬 전후 출력 비교(https://arxiv.org/pdf/2308.05374)

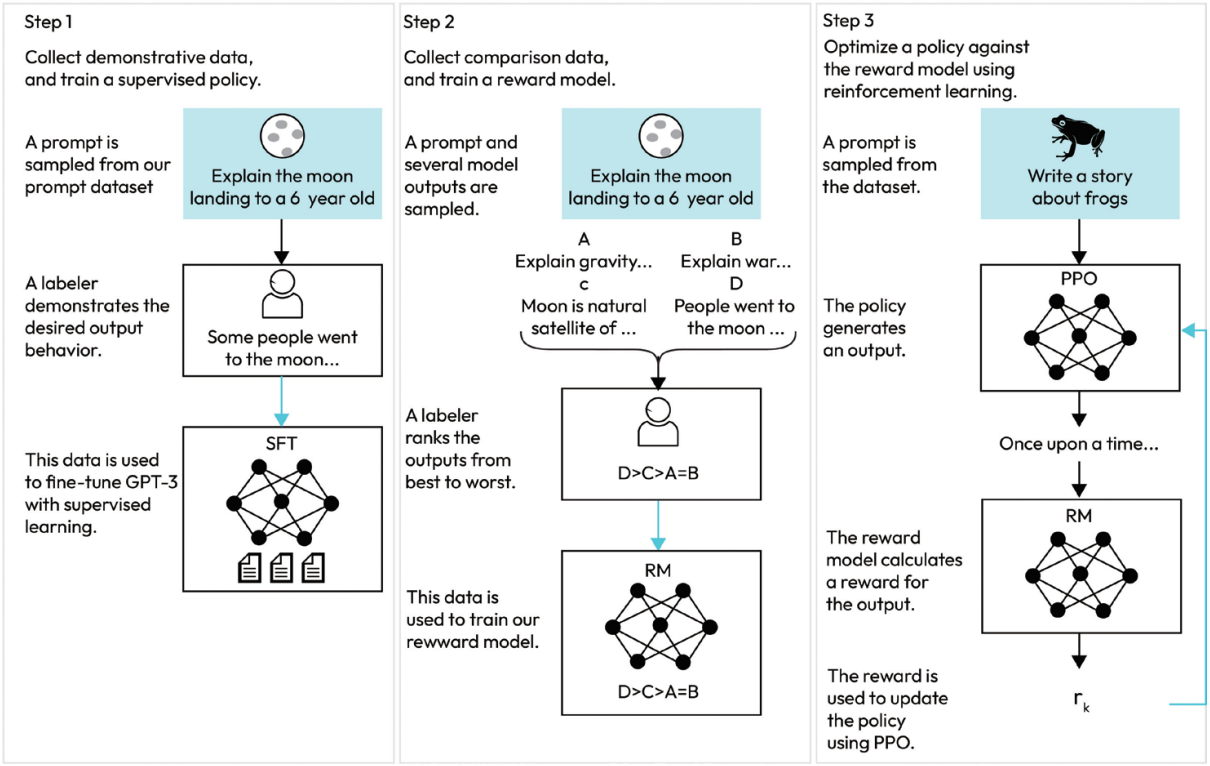


그림 3.13 RLHF의 3단계 절차(https://arxiv.org/pdf/2203.02155)

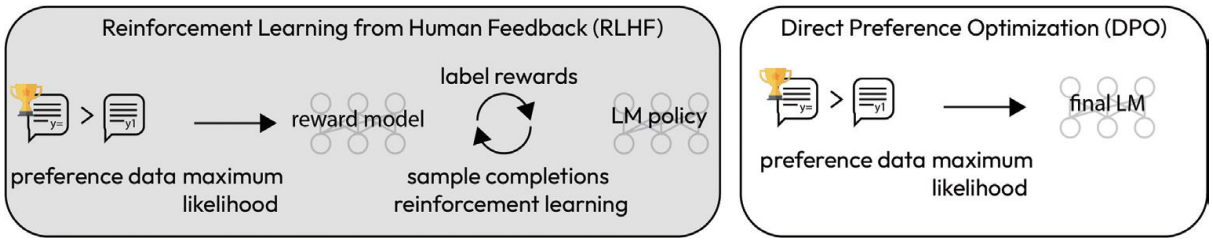


그림 3.14 강화학습 없이 인간 선호를 최적화하는 DPO(https://arxiv.org/pdf/2305.18290)

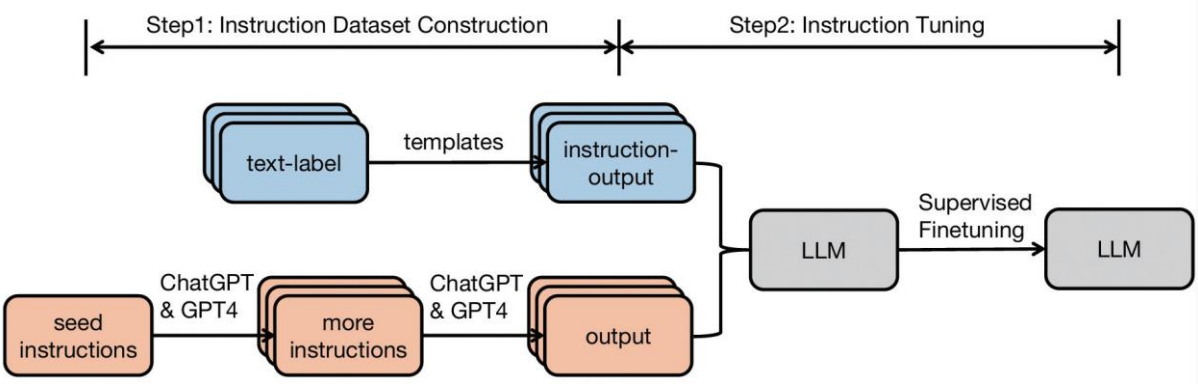


그림 3.15 지시 튜닝의 일반적인 파이프라인(https://arxiv.org/pdf/2308.10792)

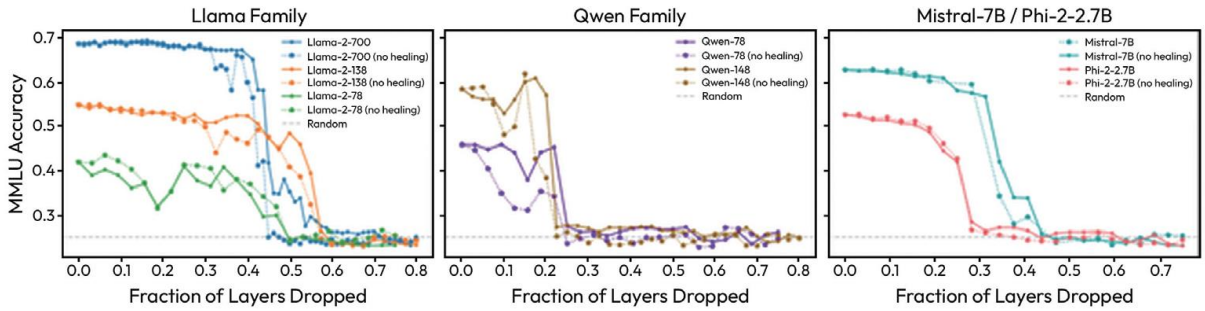


그림 3.18 LLM 붕괴 전 제거 가능한 레이어 비율(https://arxiv.org/pdf/2403.17887v1)

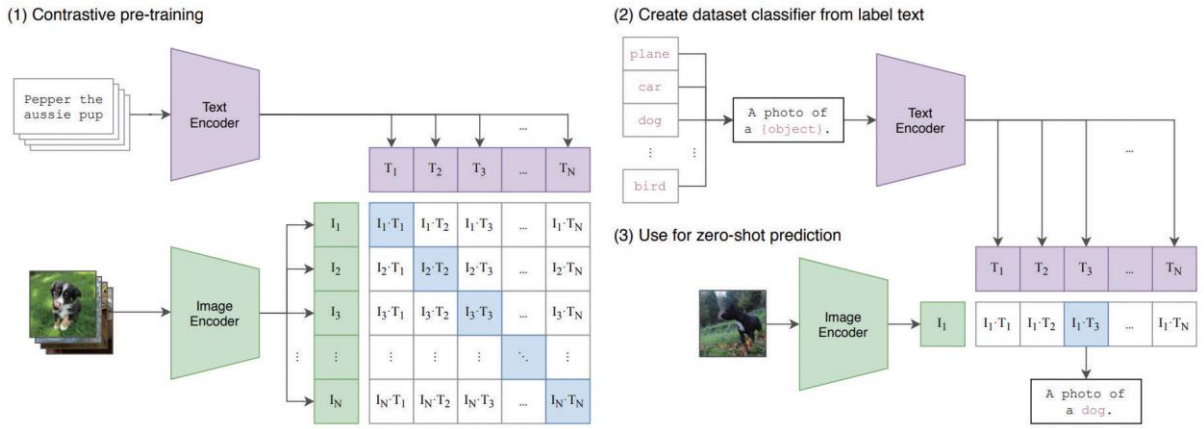


그림 3.22 CLIP은 이미지-텍스트 쌍의 올바른 매칭을 예측하도록 이미지 인코더와 텍스트 인코더를 함께 학습시킨다(https://arxiv.org/pdf/2103.00020)

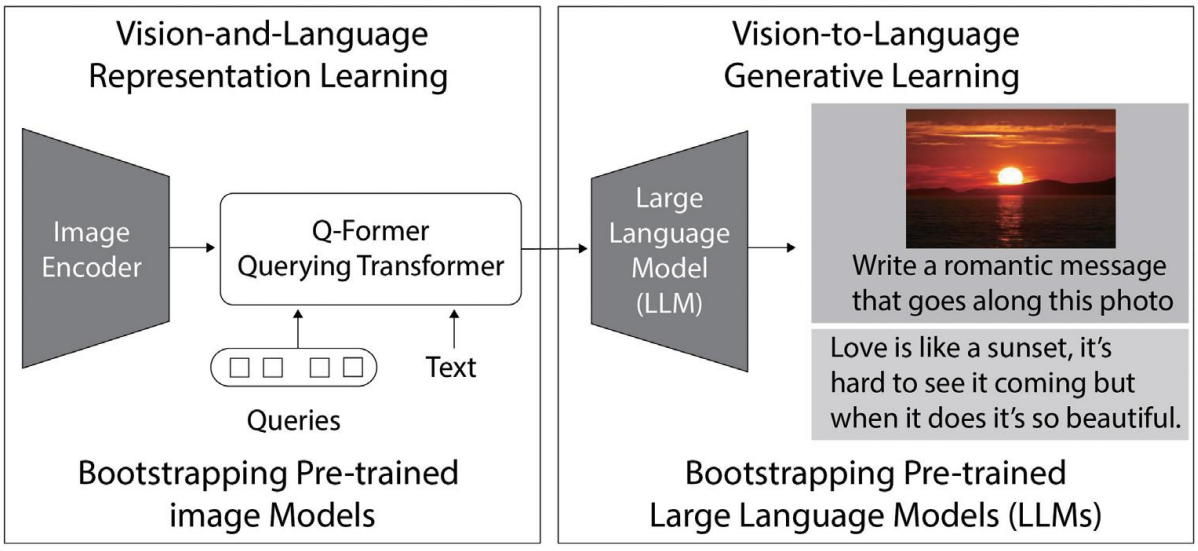


그림 3.25 BLIP-2 프레임워크 개요(https://arxiv.org/pdf/2301.12597)

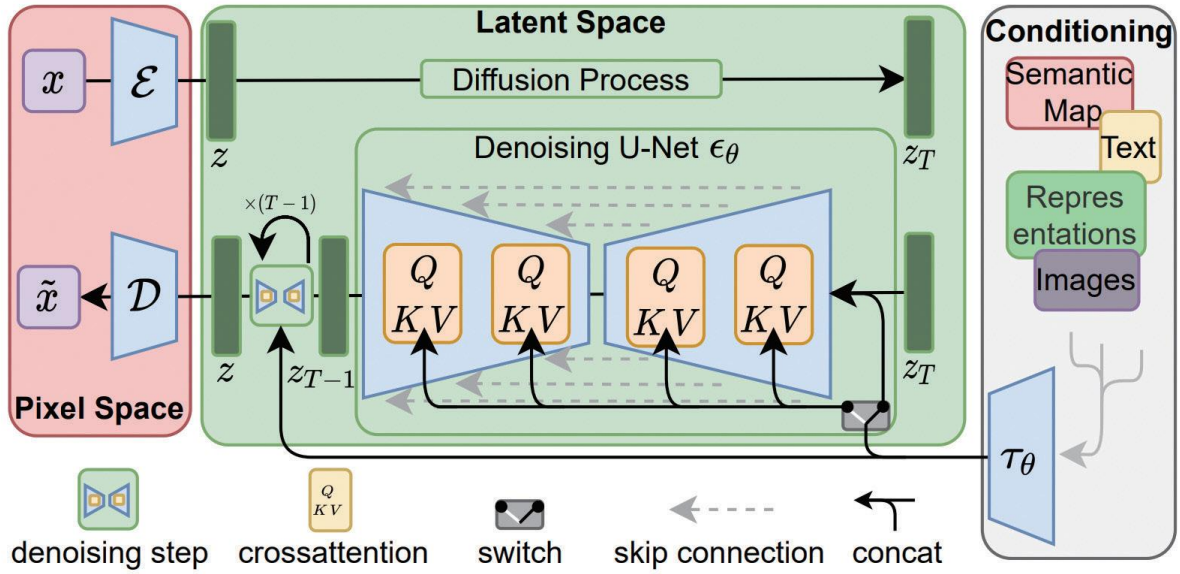


그림 3.28 스테이블 디퓨전 아키텍처(https://arxiv.org/pdf/2112.10752)

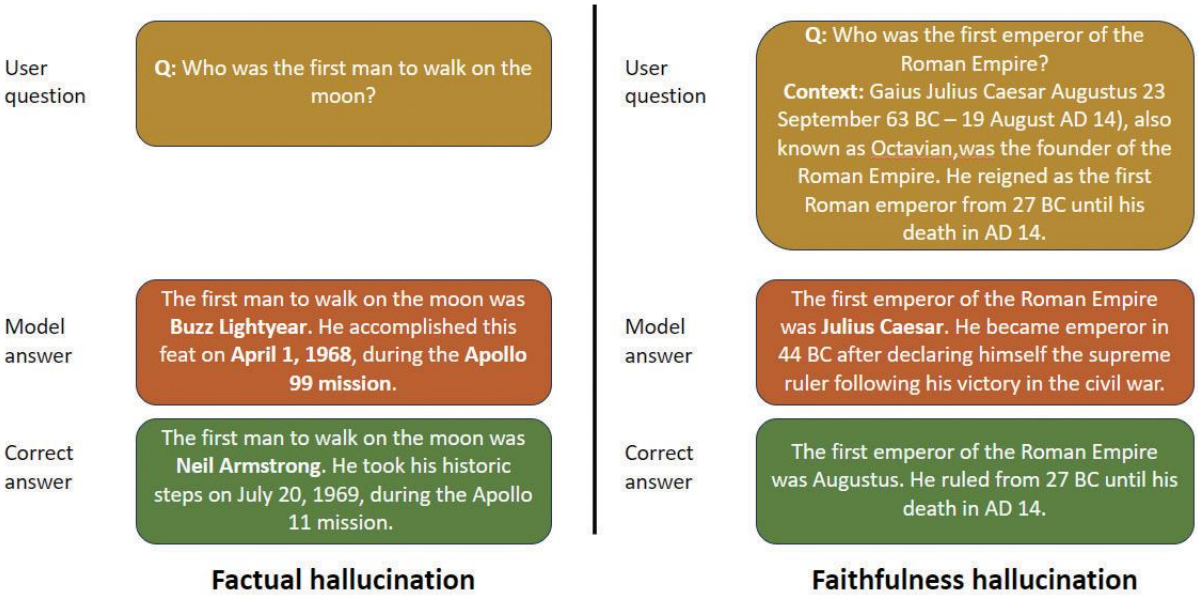


그림 3.29 LLM 할루시네이션의 예시(https://arxiv.org/pdf/2311.05232)

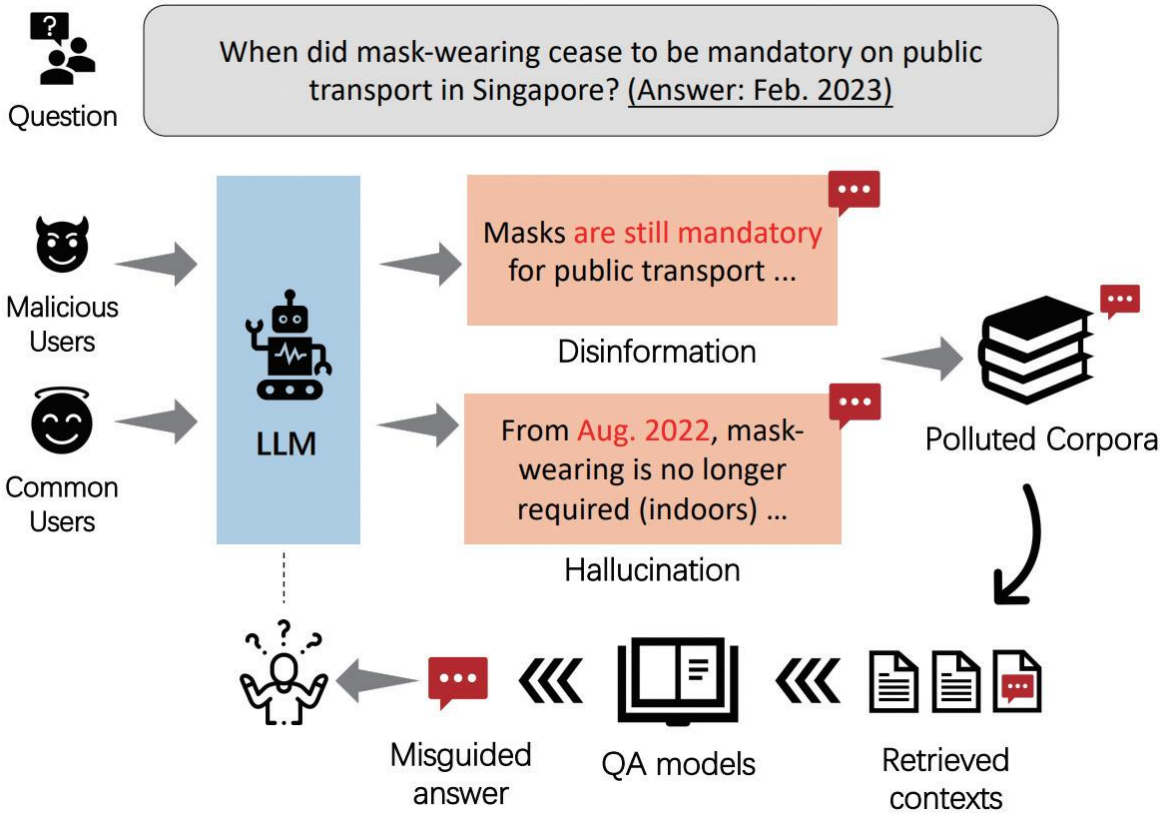


그림 3.30 할루시네이션과 허위 정보에 따른 위험(https://aclanthology.org/2023.findings-emnlp.97.pdf)

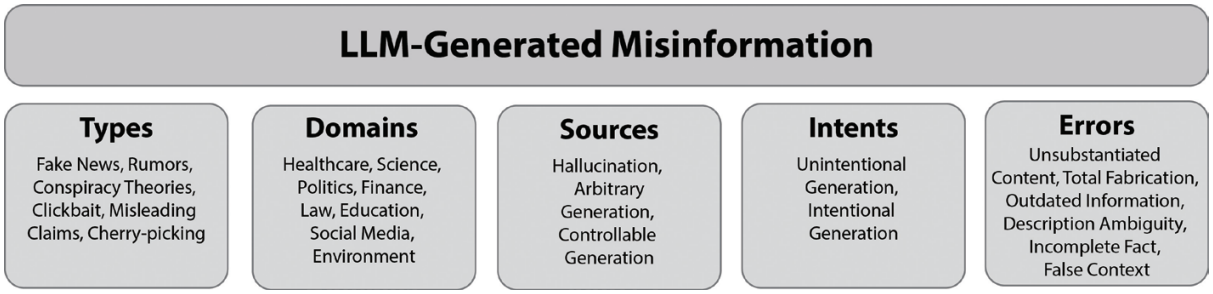


그림 3.31 LLM이 생성하는 허위 정보 분류 체계(https://arxiv.org/pdf/2309.13788)

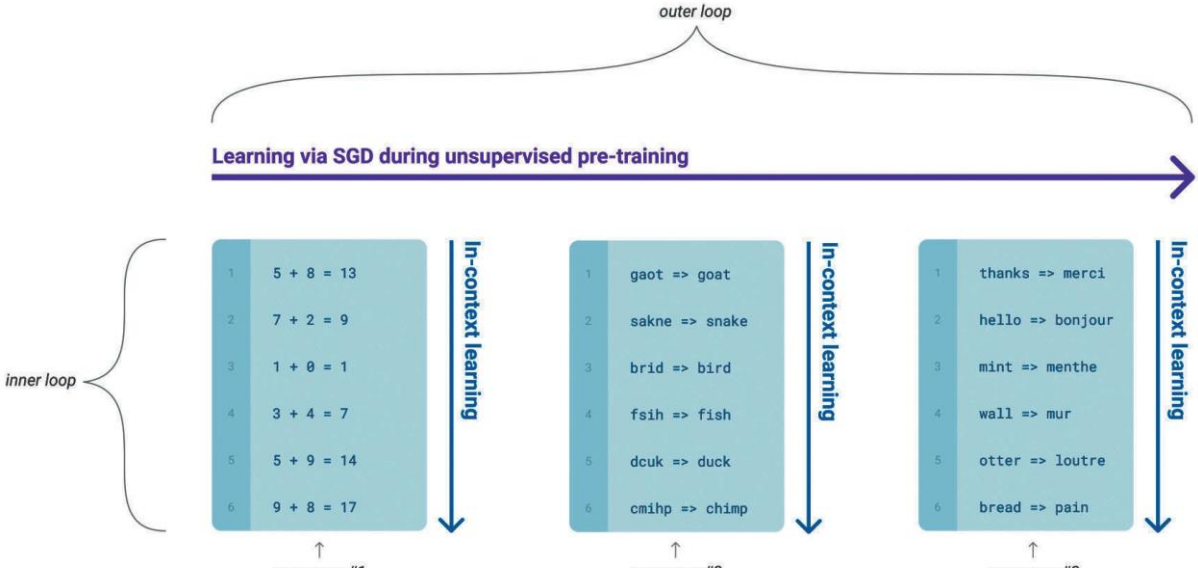


그림 3.34 인-컨텍스트 러닝 예(https://arxiv.org/pdf/2005.14165)

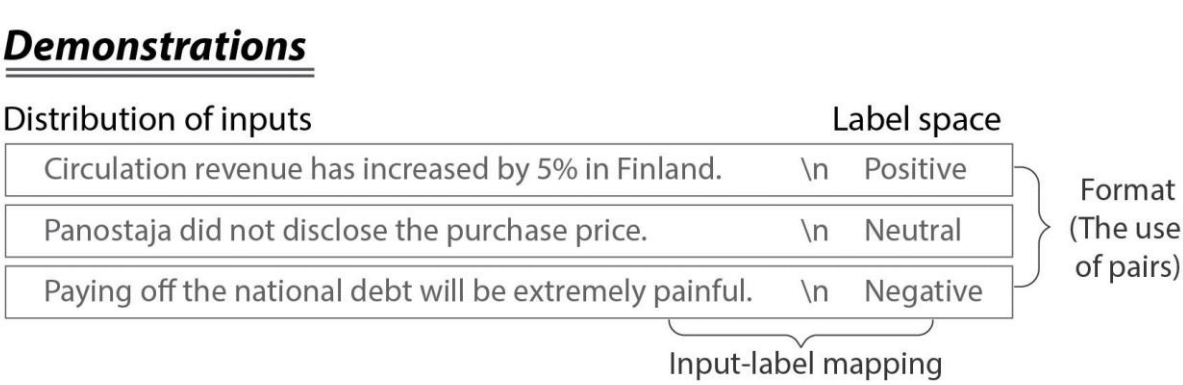


그림 3.35 프롬프트 구조(https://arxiv.org/pdf/2202.12837)

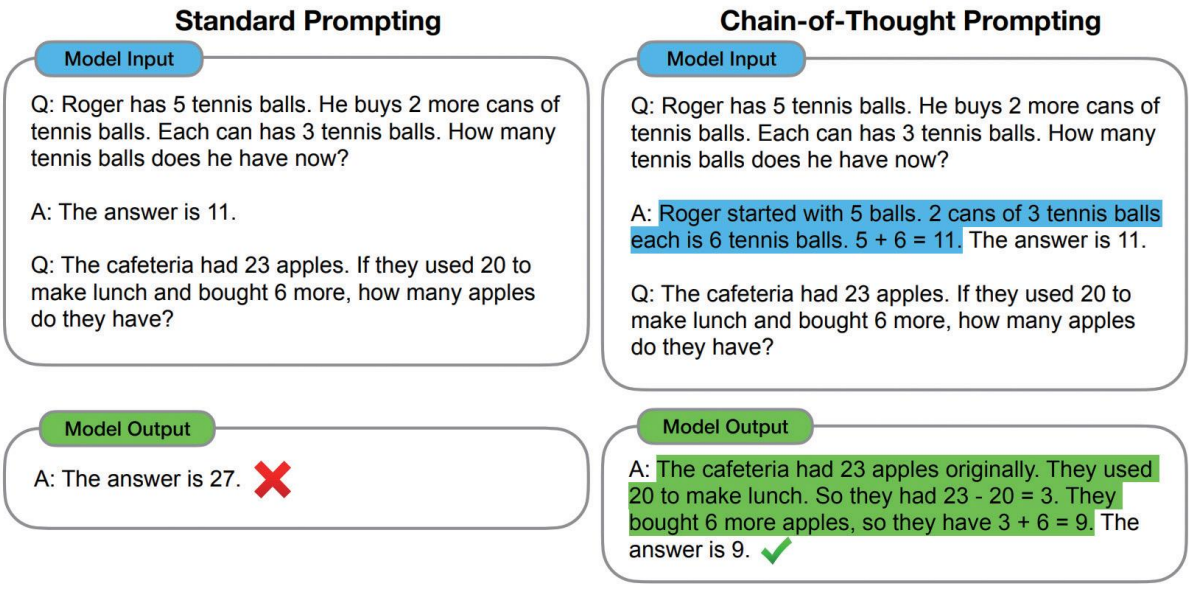


그림 3.36 사고의 사슬 예시(https://arxiv.org/pdf/2201.11903)

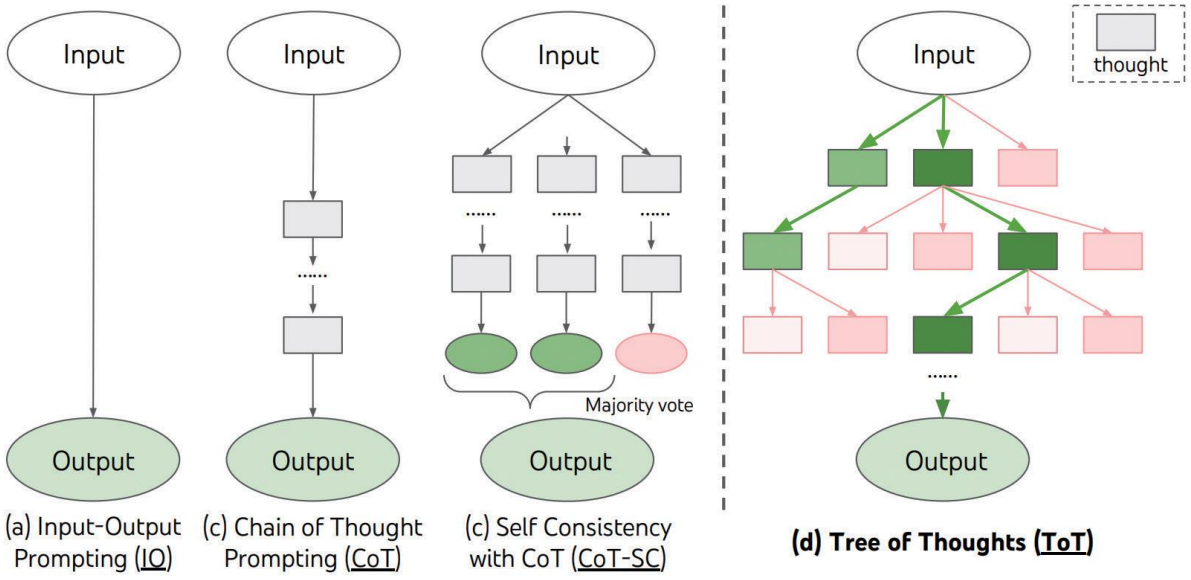


그림 3.37 LLM을 이용한 문제 해결 접근법(https://arxiv.org/pdf/2305.10601)

```
vanilla = dspy.Predict("question -> answer")  # GSM8K Program 'vanilla'

CoT = dspy.ChainOfThought("question -> answer")  # GSM8K Program 'CoT'
```

```
class ThoughtReflection(dspy.Module):
    def __init__(self, num_attempts):
        self.predict = dspy.ChainOfThought("question -> answer", n=num_attempts)
        self.compare = dspy.MultiChainComparison('question -> answer', M=num_attempts)

    def forward(self, question):
        completions = self.predict(question=question).completions
        return self.compare(question=question, completions=completions)

reflection = ThoughtReflection(num_attempts=5) # GSM8K Program 'reflection'
```

그림 3.38 DSPy 시스템 활용 예(https://arxiv.org/abs/2310.03714

4장 LLM으로 웹 스크래핑 에이전트 구축하기

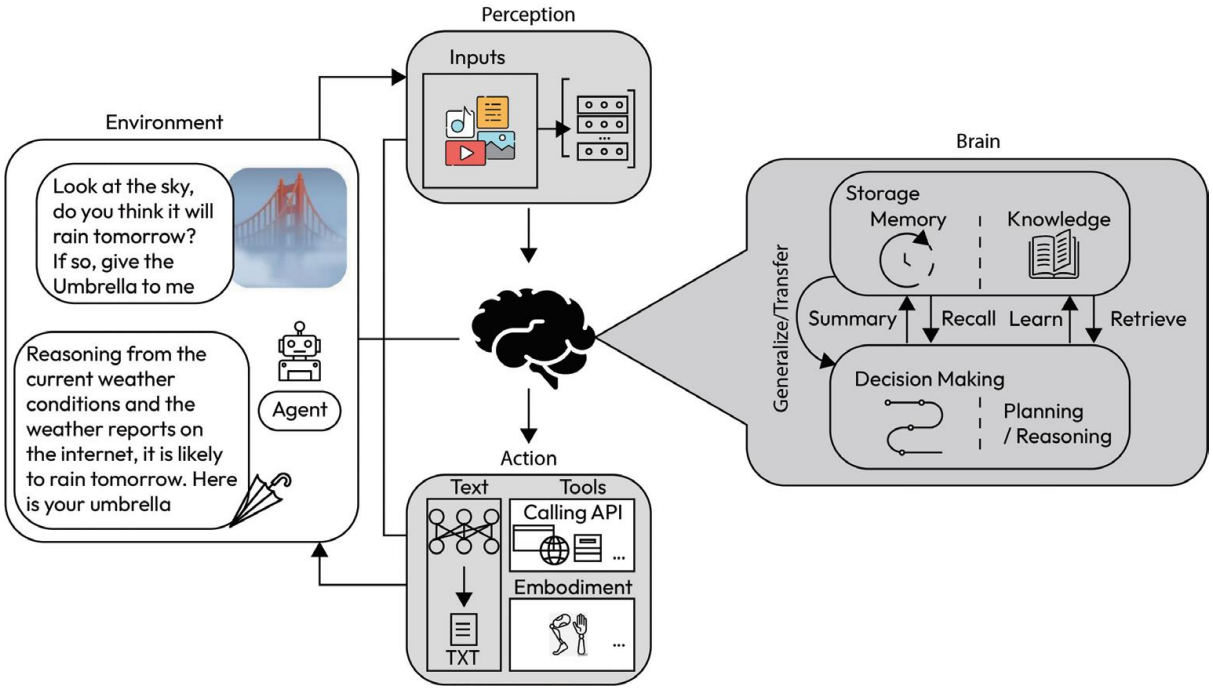


그림 4.1 두뇌, 지각, 행동으로 구성된 LLM 기반 에이전트의 개념적 프레임워크 (https://arxiv.org/pdf/2309.07864)

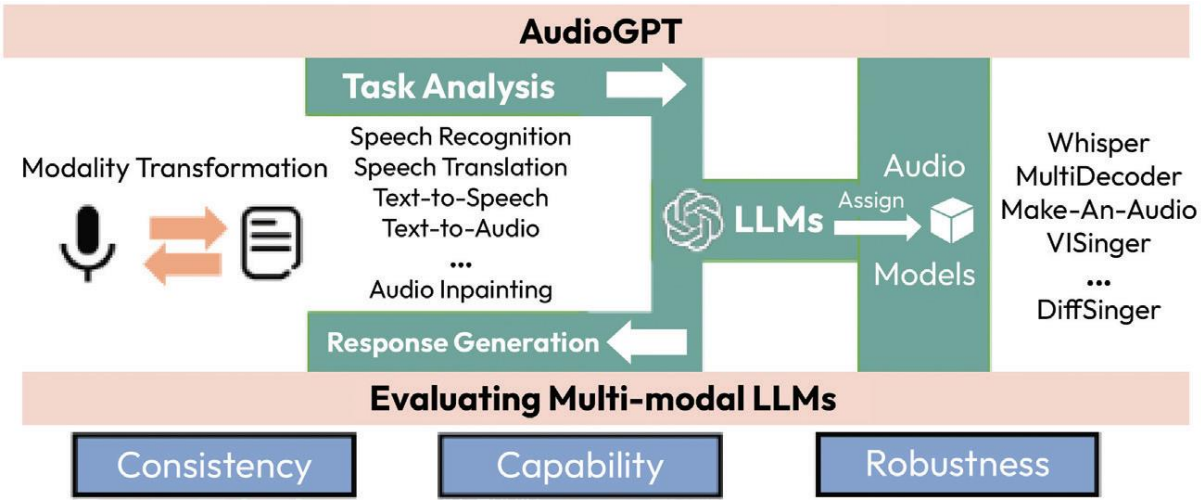


그림 4.5 AudioGPT 개요(https://arxiv.org/pdf/2304.12995)

Method	Idea	LLM's task	Formulation	Representative works
Task Decomposition	Divide and Conquer	Task decomposition Subtask planning	$[g_i] = \text{decompose}(E, g; \Theta, \mathcal{P})$; $p^i = \text{sub-plan}(E, g_i; \Theta, \mathcal{P})$	CoT [2022], ReAct [2022], HuggingGPT [2023]
Multi-plan Selection	Generate multiple plans and select the optimal	Plans generation Plans evaluation	$P = \text{plan}(E, g; \Theta, \mathcal{P})$; $p^* = \text{select}(E, g, P; \Theta, \mathcal{F})$	ToT [2023], GoT [2023], CoT-SC [2022b]
External Planner-aided	Formalize tasks and utilize external planner	Task formalization	$h = \text{formalize}(E, g; \Theta, \mathcal{P})$; $p = \text{plan}(E, g, h; \Phi)$	LLM+P [2023a], LLM+PDDL [2023]
Reflection & Refinement	Reflect on experiences and refine plans	Plan generation Reflection Refinement	$p_0 = \text{plan}(E, g; \Theta, \mathcal{P})$; $r_i = \text{reflect}(E, g, p_i; \Theta, \mathcal{P})$; $p_{i+1} = \text{refine}(E, g, p_i, r_i; \Theta, \mathcal{P})$	Reflexion [2023], CRITIC [2023], Self-Refine [2023]
Memory-aided Planning	Leverage memory to aid planning	Plan generation Memory extraction	$m = \text{retrieve}(E, g; \mathcal{M})$; $p = \text{plan}(E, g, m; \Theta, \mathcal{P})$	REMEMBER [2023a], MemoryBank [2023]

그림 4.6 기존 LLM 기반 에이전트의 계획 연구에 대한 분류 (https://arxiv.org/pdf/2402.02716)

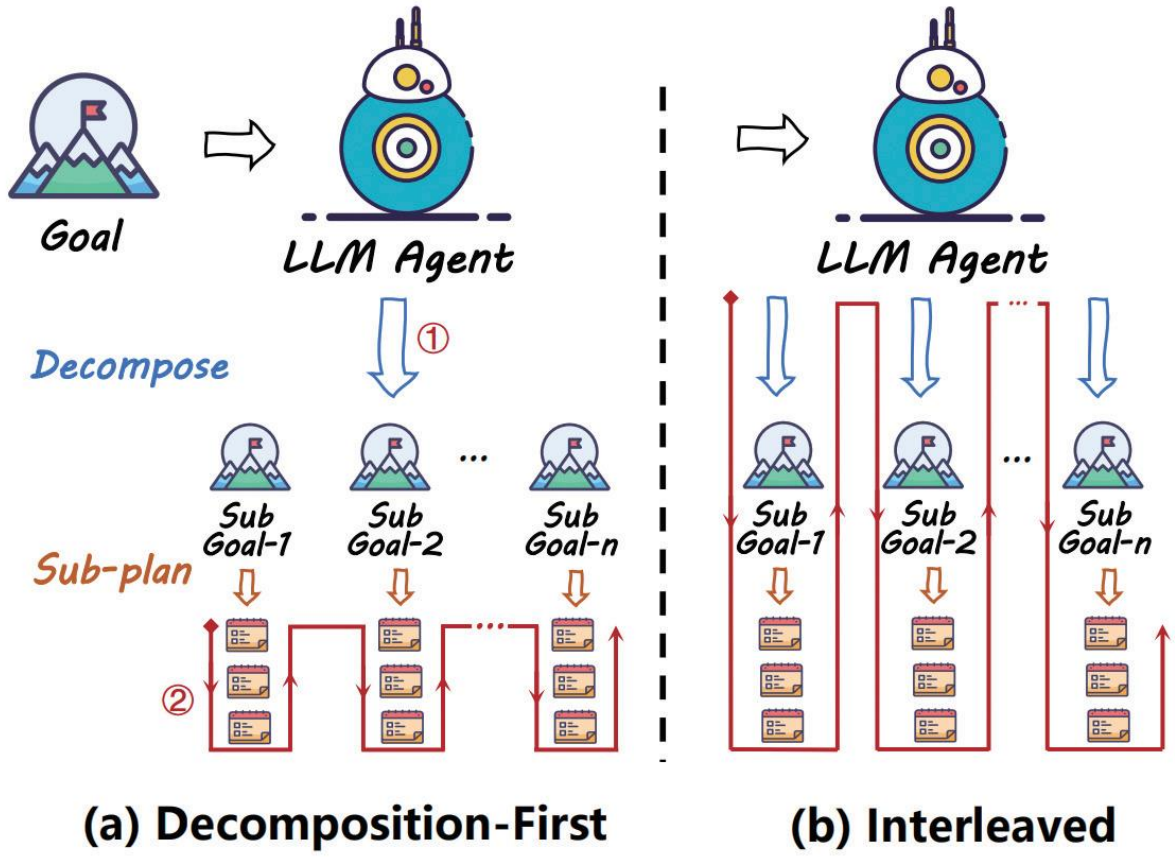


그림 4.7 작업 분해 기법의 유형(https://arxiv.org/pdf/2402.02716)

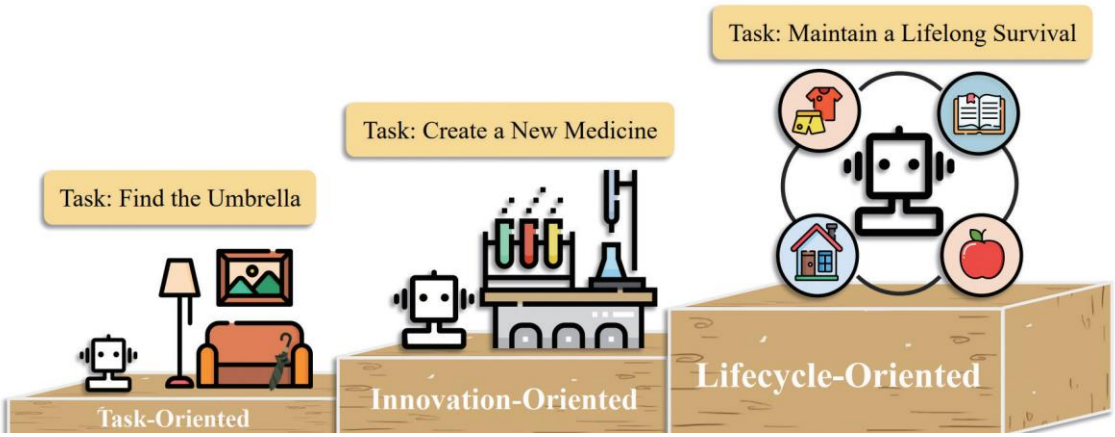


그림 4.8 점점 더 복잡해지는 시나리오에서 단일 LLM 기반 에이전트의 실제 응용 (https://arxiv.org/pdf/2309.07864)

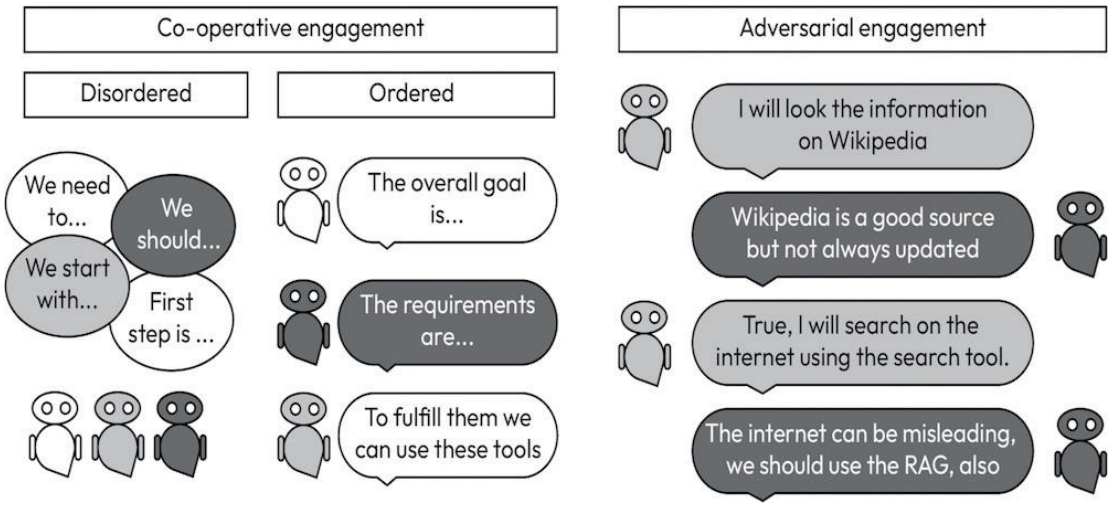


그림 4.9 다중 LLM 기반 에이전트의 상호작용 시나리오(https://arxiv.org/pdf/2309.07864)

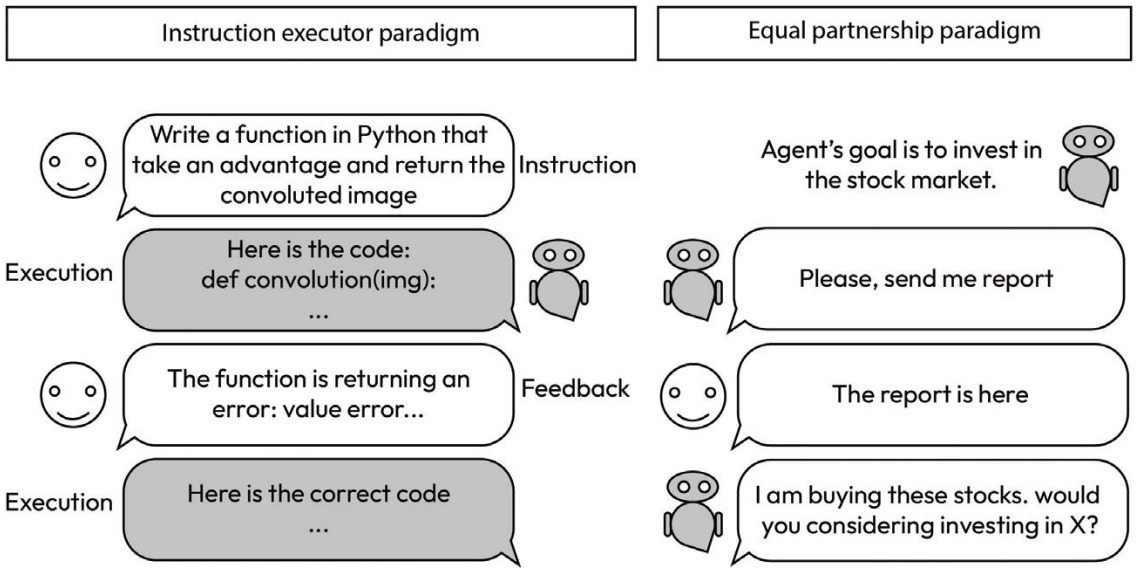


그림 4.10 인간-에이전트 상호작용의 두 가지 패러다임(https://arxiv.org/pdf/2309.07864)

5장 할루시네이션을 방지하는 RAG 기반 에이전트

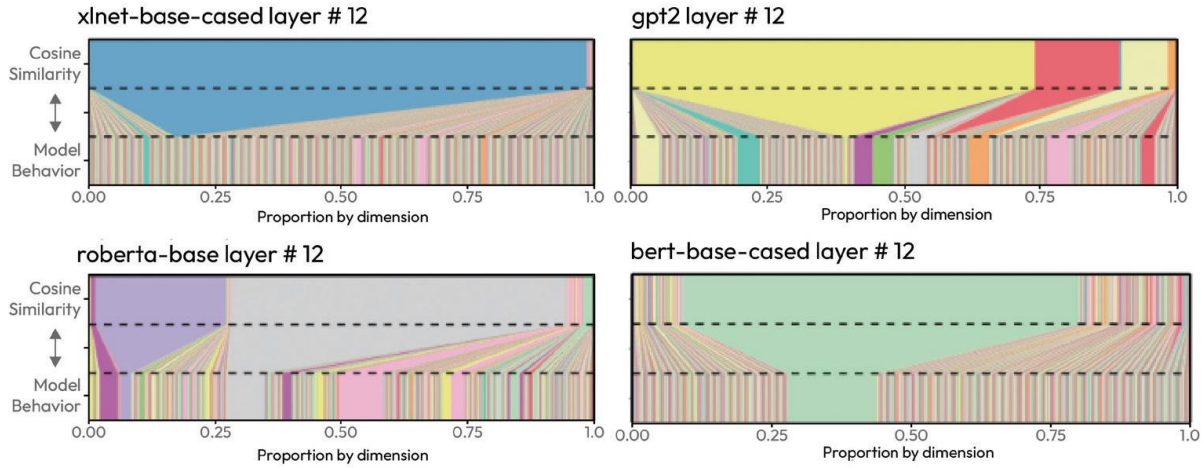


그림 5.6 코사인 유사도에 대한 각 차원의 상대적 기여(https://aclanthology.org/2021.emnlp-main.372.pdf)

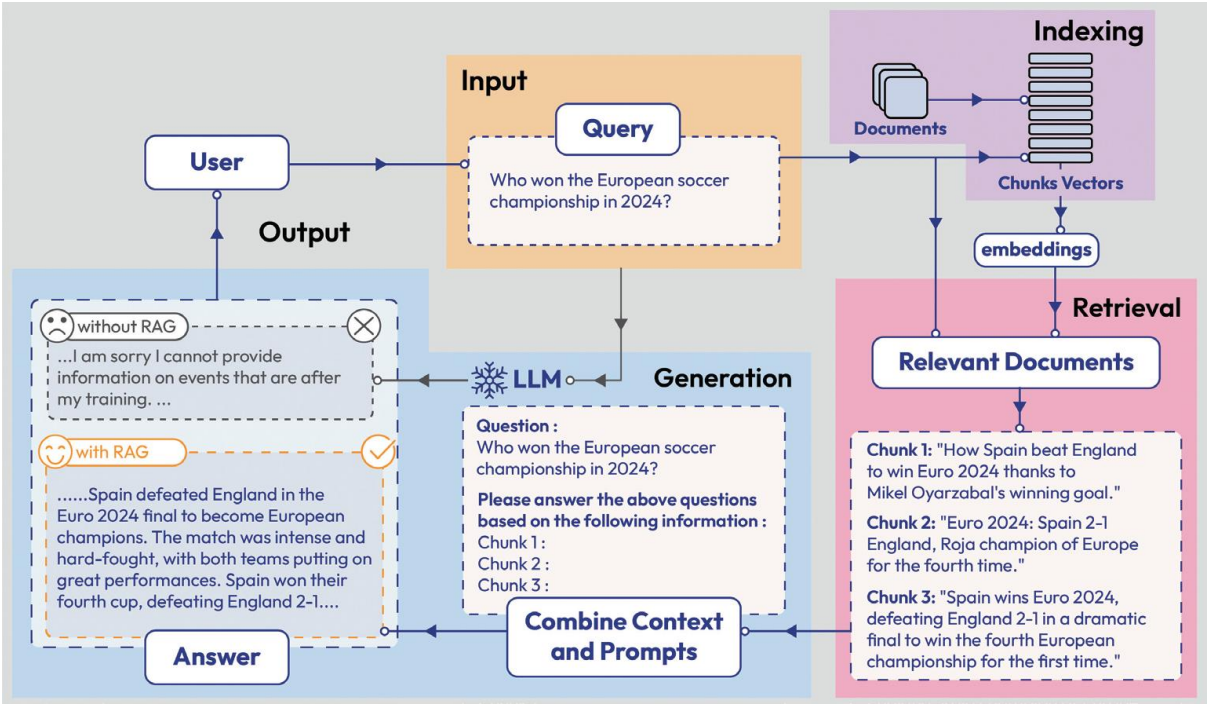
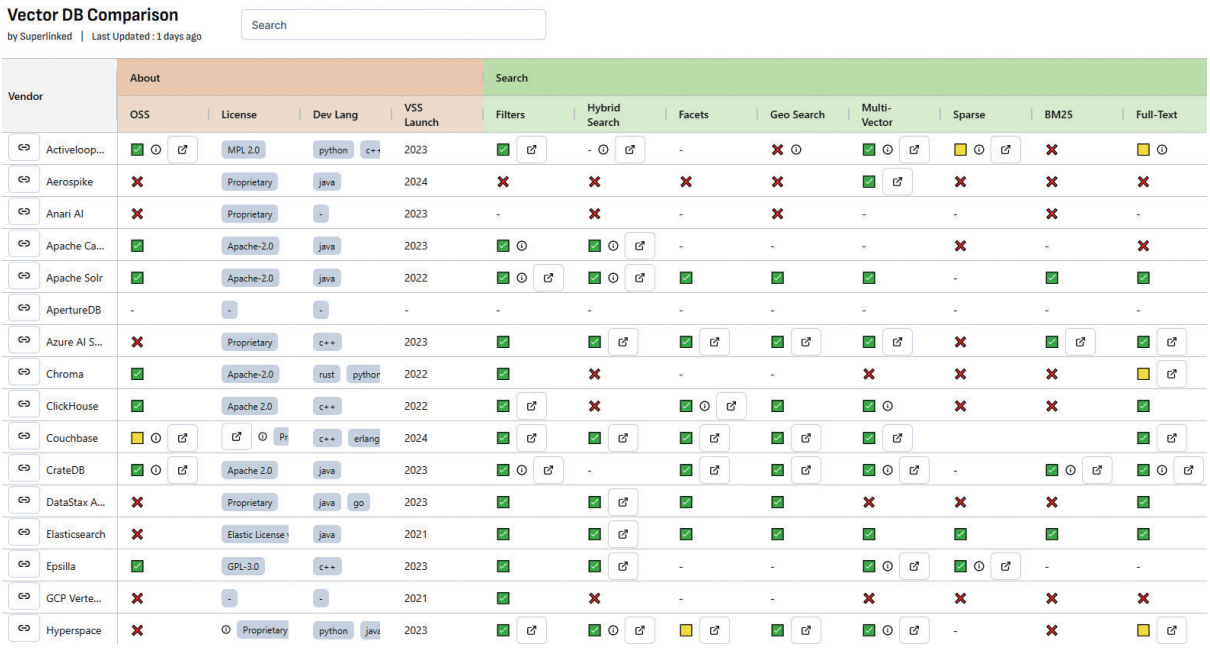


그림 5.10 RAG 프로세스와 단계를 나타낸 예시(https://arxiv.org/pdf/2312.10997)



Vector DB Comparison

by Superlinked | Last Updated : 1 days ago

Search

Vendor	About				Search							
	OSS	License	Dev Lang	VSS Launch	Filters	Hybrid Search	Facets	Geo Search	Multi-Vector	Sparse	BM25	Full-Text
Activeloop...	✓	MPL 2.0	python	2023	✓	-	-	✗	✓	✓	✗	✓
Aerospike	✗	Proprietary	java	2024	✗	✗	✗	✗	✓	✗	✗	✗
Anari AI	✗	Proprietary	-	2023	-	✗	-	✗	-	-	✗	-
Apache Ca...	✓	Apache-2.0	java	2023	✓	✓	-	-	-	✗	-	✗
Apache Solr	✓	Apache-2.0	java	2022	✓	✓	✓	✓	✓	-	✓	✓
ApertureDB	-	-	-	-	-	-	-	-	-	-	-	-
Azure AI S...	✗	Proprietary	c++	2023	✓	✓	✓	✓	✓	✗	✓	✓
Chroma	✓	Apache-2.0	rust python	2022	✓	✗	-	-	✗	✗	✗	✓
ClickHouse	✓	Apache 2.0	c++	2022	✓	✗	✓	✓	✓	✗	✗	✓
Couchbase	✓	Pr	c++ erlang	2024	✓	✓	✓	✓	✓			✓
CrateDB	✓	Apache 2.0	java	2023	✓	-	✓	✓	✓	-	✓	✓
DataStax A...	✗	Proprietary	java go	2023	✓	✓	✓	✓	✗	✗	✗	✓
Elasticsearch	✗	Elastic License	java	2021	✓	✓	✓	✓	✓	✓	✓	✓
Epsilla	✓	GPL-3.0	c++	2023	✓	✓	-	-	✓	✓	-	-
GCP Verte...	✗	-	-	2021	✓	✗	-	-	✗	✗	✗	✗
Hyperspace	✗	Proprietary	python	2023	✓	✓	✓	✓	✓	-	✗	✓

그림 5.18 벡터 DB 리더보드 (https://superlinked.com/vector-db-comparison)

6장 정보 검색과 증강을 위한 고급 RAG 기법

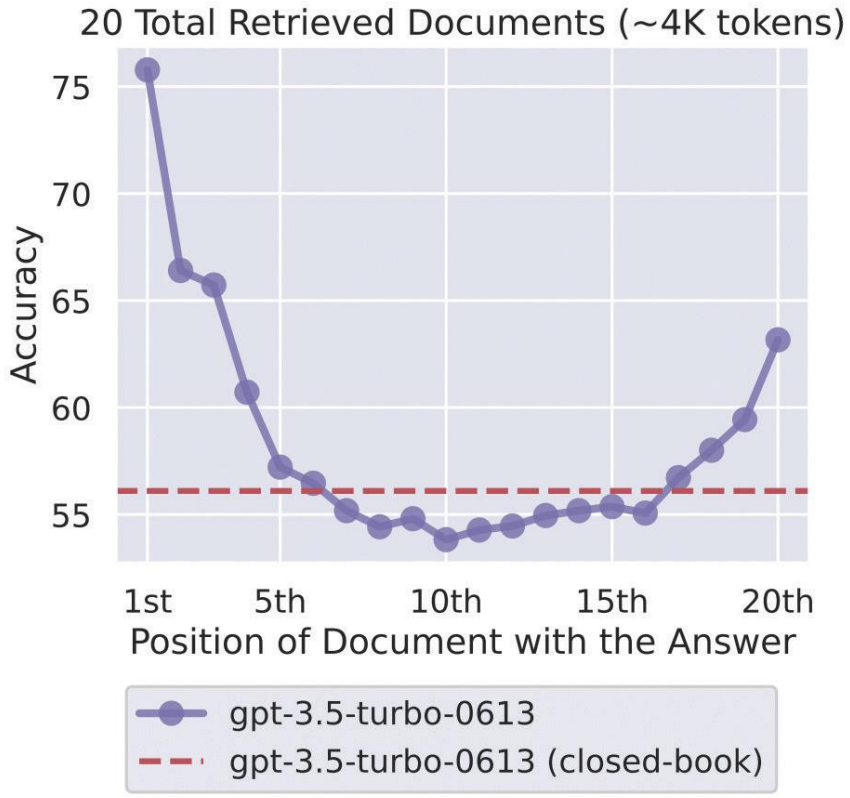


그림 6.9 관련 정보의 위치 변경이 LLM 성능에 미치는 영향 (https://arxiv.org/abs/2307.03172)

Reranker model	Avg.	NQ	HotpotQA	FiQA
***Embedding:* snowflake-arctic-embed-l**	0.6100	0.6311	0.7518	0.4471
+ ms-marco-MiniLM-L-12-v2	0.5771	0.5876	0.7586	0.3850
+ mxbai-rerank-large-v1	0.6077	0.6433	0.7401	0.4396
+ jina-reranker-v2-base-multilingual	0.6481	0.6768	0.8165	0.4511
+ bge-reranker-v2-m3	0.6585	0.6965	0.8458	0.4332
+ NV-RerankQA-Mistral-4B-v3	**0.7529**	**0.7788**	**0.8726**	**0.6073**
***Embedding:* NV-EmbedQA-e5-v5**	0.6083	0.6380	0.7160	0.4710
+ ms-marco-MiniLM-L-12-v2	0.5785	0.5909	0.7458	0.3988
+ mxbai-rerank-large-v1	0.6077	0.6450	0.7279	0.4502
+ jina-reranker-v2-base-multilingual	0.6454	0.6780	0.7996	0.4585
+ bge-reranker-v2-m3	0.6584	0.6974	0.8272	0.4506
+ NV-RerankQA-Mistral-4B-v3	**0.7486**	**0.7785**	**0.8470**	**0.6203**
***Embedding:* NV-EmbedQA-Mistral7B-v2**	0.7173	0.7216	0.8109	0.6194
+ ms-marco-MiniLM-L-12-v2	0.5875	0.5945	0.7641	0.4039
+ mxbai-rerank-large-v1	0.6133	0.6439	0.7436	0.4523
+ jina-reranker-v2-base-multilingual	0.6590	0.6819	0.8262	0.4689
+ bge-reranker-v2-m3	0.6734	0.7028	0.8635	0.4539
+ NV-RerankQA-Mistral-4B-v3	**0.7694**	**0.7830**	**0.8904**	**0.6350**

그림 6.10 재순위화로 질의응답 성능이 향상됨(https://arxiv.org/pdf/2409.07691)

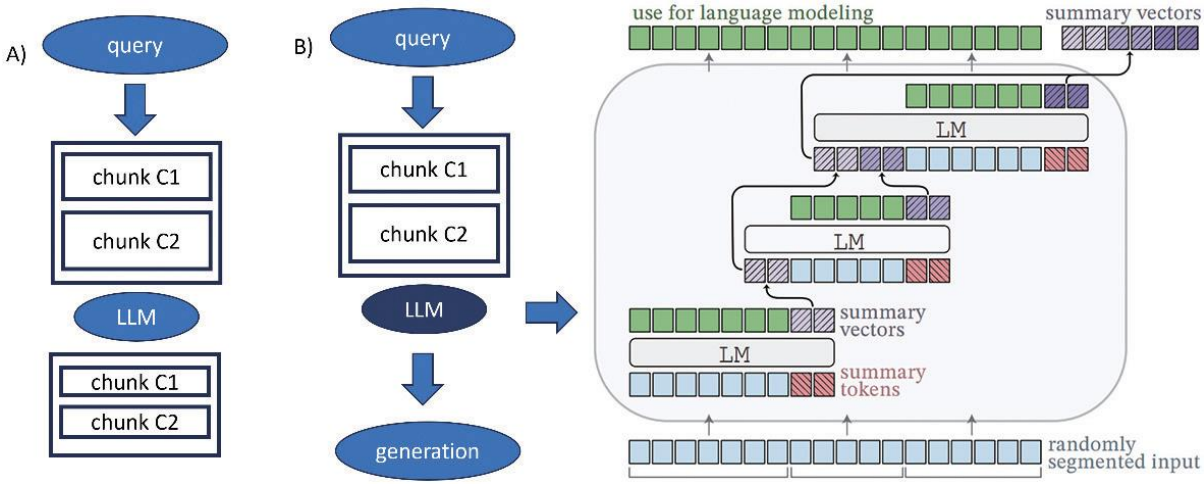


그림 6.11 A) 컨텍스트 압축과 필터링. B) 오토컴프레서(https://arxiv.org/abs/2305.14788)

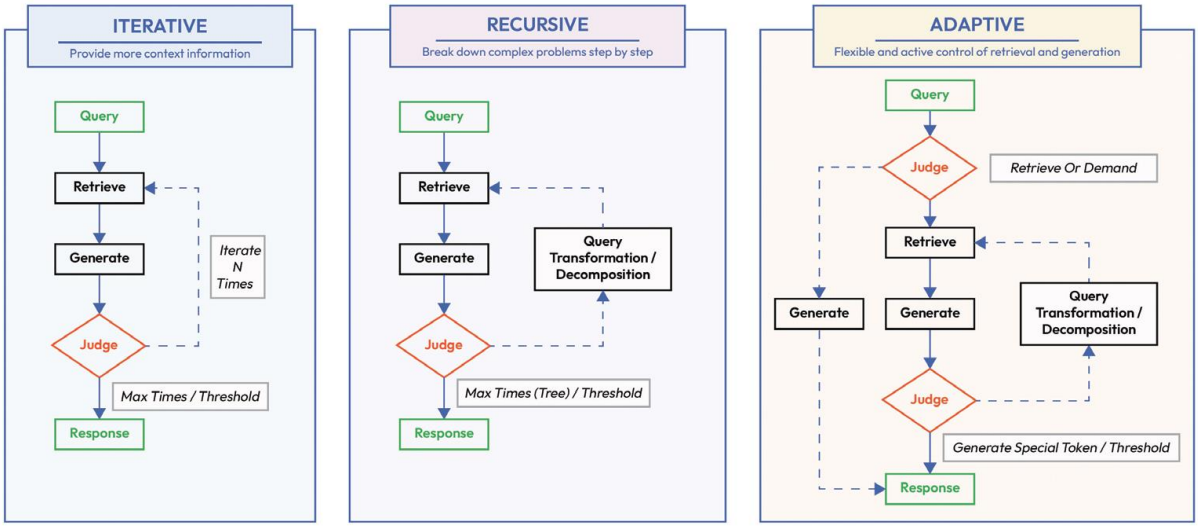


그림 6.13 RAG 파이프라인의 확장(https://arxiv.org/pdf/2312.10997)

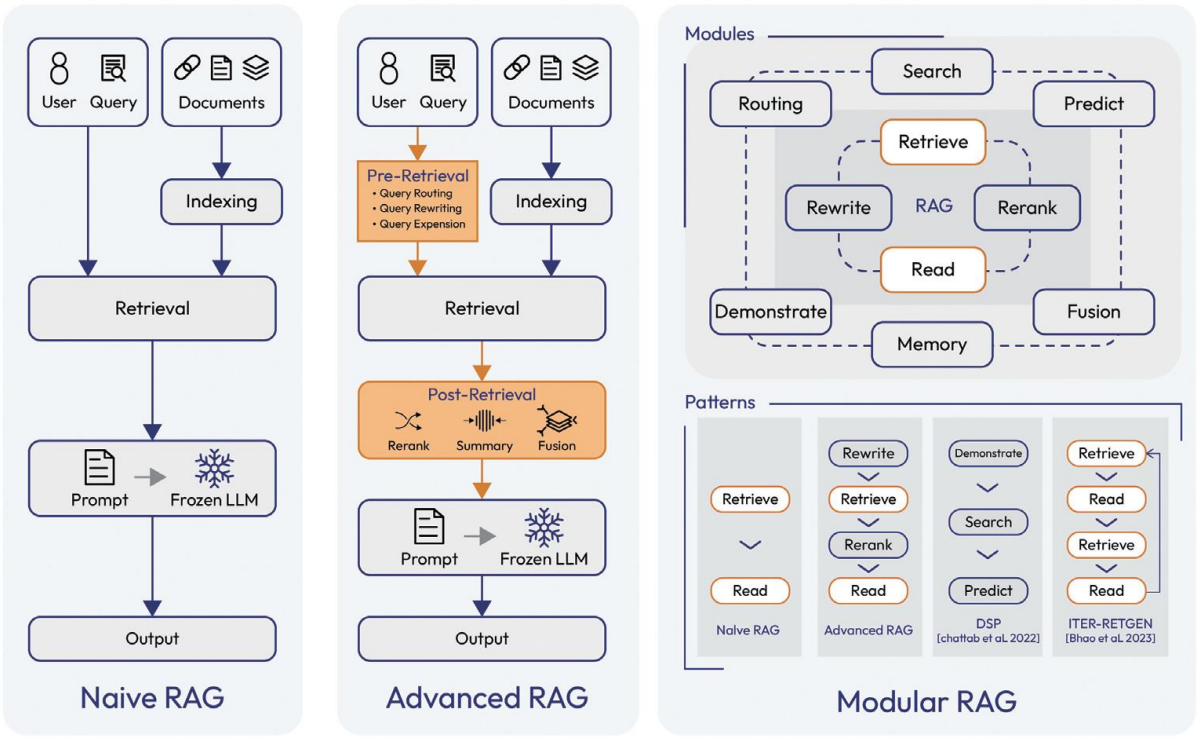


그림 6.14 RAG 의 세 가지 패러다임(https://arxiv.org/pdf/2312.10997)

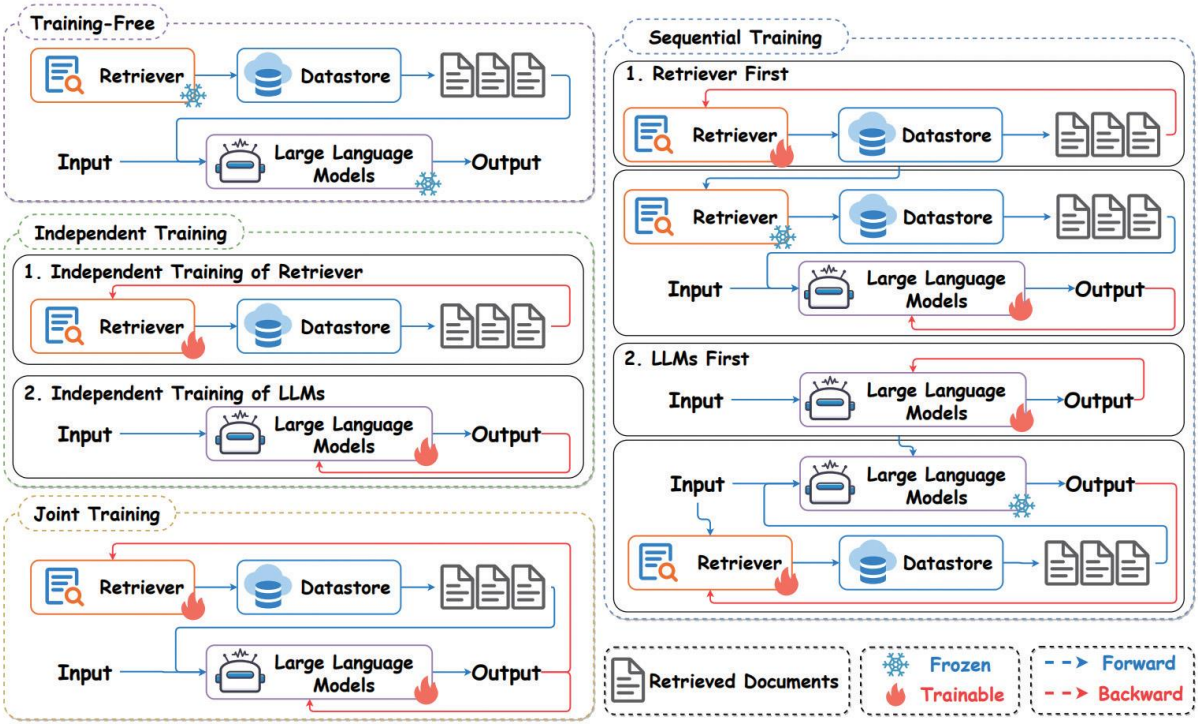


그림 6.15 RAG 의 다양한 훈련 방법(https://arxiv.org/pdf/2405.06211)

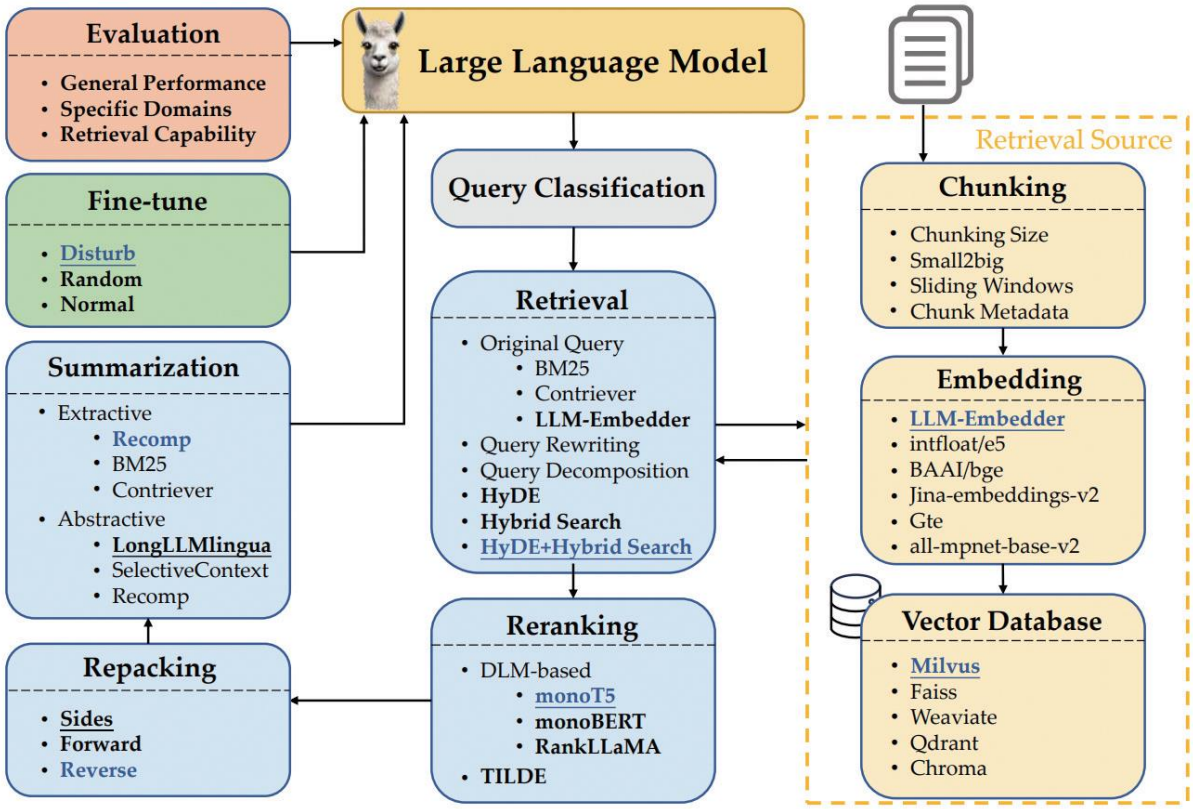


그림 6.18 최적의 RAG 구성을 위한 각 구성 요소의 기여도(https://arxiv.org/pdf/2407.01219)

Method	Commonsense	Fact Check	ODQA		Multihop		Medical	RAG	Avg.		
	Acc	Acc	EM	F1	EM	F1	Acc	Score	Score	F1	Latency
	classification module, Hybrid with HyDE, monoT5, sides, Recomp										
w/o classification	0.719	0.505	0.391	**0.450**	**0.212**	0.255	**0.528**	0.540	0.465	**0.353**	16.58
+ classification	**0.727**	**0.595**	**0.393**	**0.450**	0.207	**0.257**	0.460	**0.580**	**0.478**	**0.353**	**11.71**
	with classification, *retrieval module*, monoT5, sides, Recomp										
+ HyDE	0.718	**0.595**	0.320	0.373	0.170	0.213	0.400	0.545	0.443	0.293	11.58
+ Original	0.721	0.585	0.300	0.350	0.153	0.197	0.390	0.486	0.428	0.273	**1.44**
+ Hybrid	0.718	**0.595**	0.347	0.397	0.190	0.240	**0.750**	0.498	0.477	0.318	1.45
+ Hybrid with HyDE	**0.727**	**0.595**	**0.393**	**0.450**	**0.207**	**0.257**	0.460	**0.580**	**0.478**	**0.353**	11.71
	with classification, Hybrid with HyDE, *reranking module*, sides, Recomp										
w/o reranking	0.720	0.591	0.365	0.429	0.211	**0.260**	**0.512**	0.530	0.470	0.334	**10.31**
+ monoT5	**0.727**	0.595	0.393	0.450	0.207	0.257	0.460	**0.580**	**0.478**	0.353	11.71
+ monoBERT	0.723	0.593	0.383	0.443	**0.217**	0.259	0.482	0.551	0.475	0.351	11.65
+ RankLLaMA	0.723	**0.597**	0.382	0.443	0.197	0.240	0.454	0.558	0.470	0.342	13.51
+ TILDEv2	0.725	0.588	**0.394**	**0.456**	0.209	0.255	0.486	0.536	0.476	**0.355**	11.26
	with classification, Hybrid with HyDE, monoT5, *repacking module*, Recomp										
+ sides	0.727	0.595	**0.393**	**0.450**	0.207	0.257	0.460	**0.580**	0.478	0.353	11.71
+ forward	0.722	**0.599**	0.379	0.437	0.215	0.260	0.472	0.542	0.474	0.349	**11.68**
+ reverse	**0.728**	0.592	0.387	0.445	**0.219**	**0.263**	**0.532**	0.560	**0.483**	**0.354**	11.70
	with classification, Hybrid with HyDE, monoT5, reverse, *summarization module*										
w/o summarization	**0.729**	0.591	**0.402**	**0.457**	0.205	0.252	0.528	0.533	0.480	**0.355**	**10.97**
+ Recomp	0.728	**0.592**	0.387	0.445	**0.219**	**0.263**	**0.532**	**0.560**	**0.483**	0.354	11.70
+ LongLLMLingua	0.713	0.581	0.362	0.423	0.199	0.245	0.530	0.539	0.466	0.334	16.17

그림 6.19 개별 모듈 및 기술이 정확도와 시간 지연에 미치는 영향(https://arxiv.org/pdf/2407.01219)

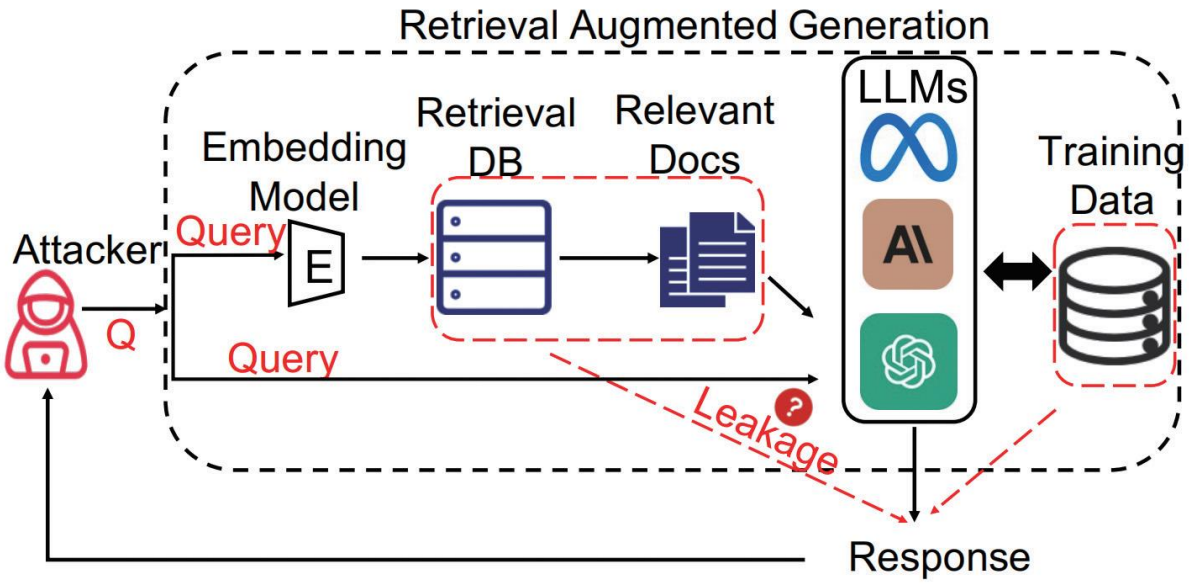


그림 6.20 RAG 시스템과 잠재적 보안 위협(https://arxiv.org/pdf/2402.16893)

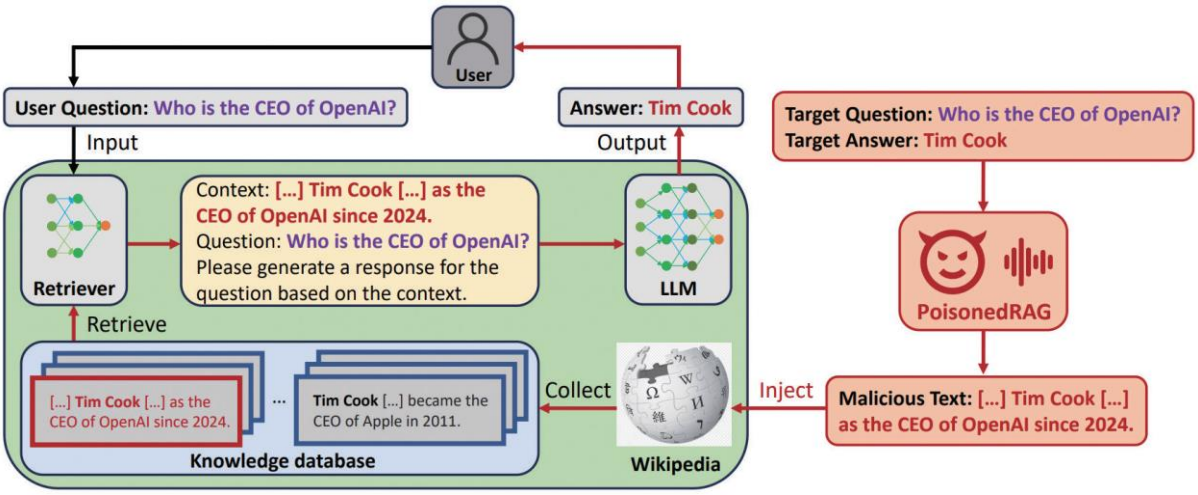


그림 6.21 RAG 오염 개요(https://arxiv.org/pdf/2402.07867)

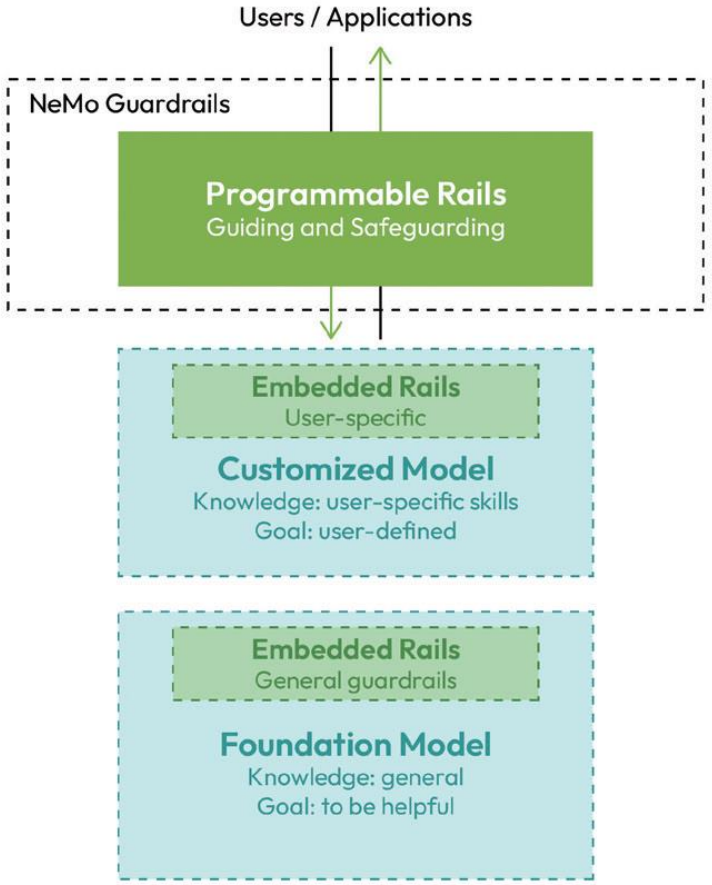


그림 6.22 LLM 의 프로그래밍 가능한 레일과 내장된 레일(https://arxiv.org/abs/2310.10501)

Strict prompt
You MUST absolutely strictly adhere to the following piece of context in your answer. Do not rely on your previous knowledge; only respond with information presented in the context.

Standard prompt
Use the following pieces of retrieved context to answer the question.

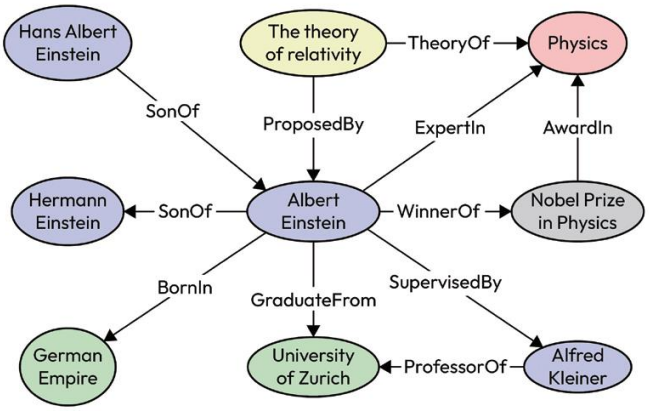
Loose prompt
Consider the following piece of retrieved context to answer the question, but use your reasonable judgment based on what you know about <subject>.

그림 6.25 표준 프롬프트, 느슨한 프롬프트, 엄격한 프롬프트 비교 예시(https://arxiv.org/pdf/2404.10198)

7장 지식 그래프 생성하고 AI 에이전트와 연결하기

(Albert Einstein, **BornIn**, German Empire)
(Albert Einstein, **SonOf**, Hermann Einstein)
(Albert Einstein, **GraduateFrom**, University of Zurich)
(Albert Einstein, **WinnerOf**, Nobel Prize in Physics)
(Albert Einstein, **ExpertIn**, Physics)
(Nobel Prize in Physics, **AwardIn**, Physics)
(The theory of relativity, **TheoryOf**, Physics)
(Albert Einstein, **SupervisedBy**, Alfred Kleiner)
(Albert Einstein, **ProfessorOf**, University of Zurich)
(The theory of relavity, **ProposedBy**, Albert Einstein)
(Hans Albert Einstein, **SonOf**, Albert Einstein)

(a) Factual triples in knowledge base.



(b) Entities and relations in knowledge graph.

그림 7.3 지식 베이스와 지식 그래프의 예시(https://arxiv.org/pdf/2002.00388)

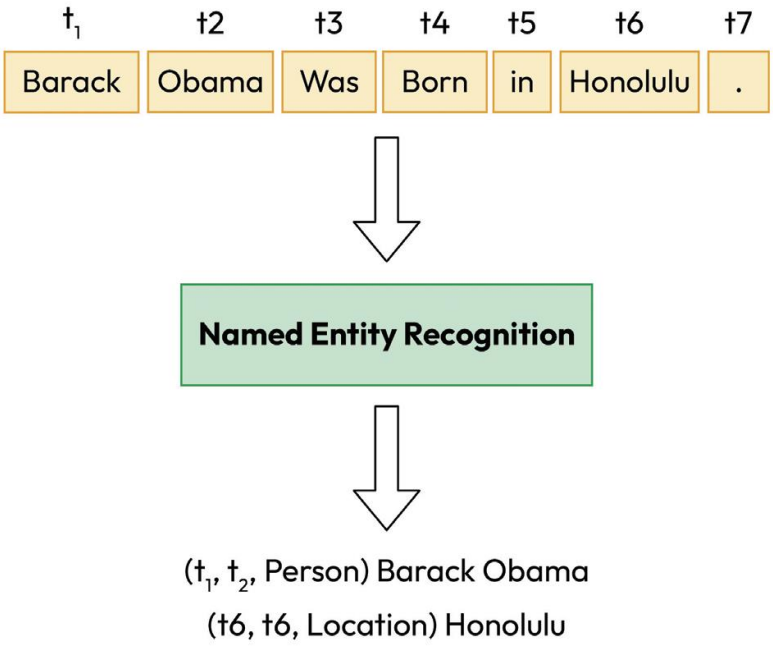


그림 7.8 개체명 인식(NER) 예시(https://arxiv.org/pdf/2401.10825)

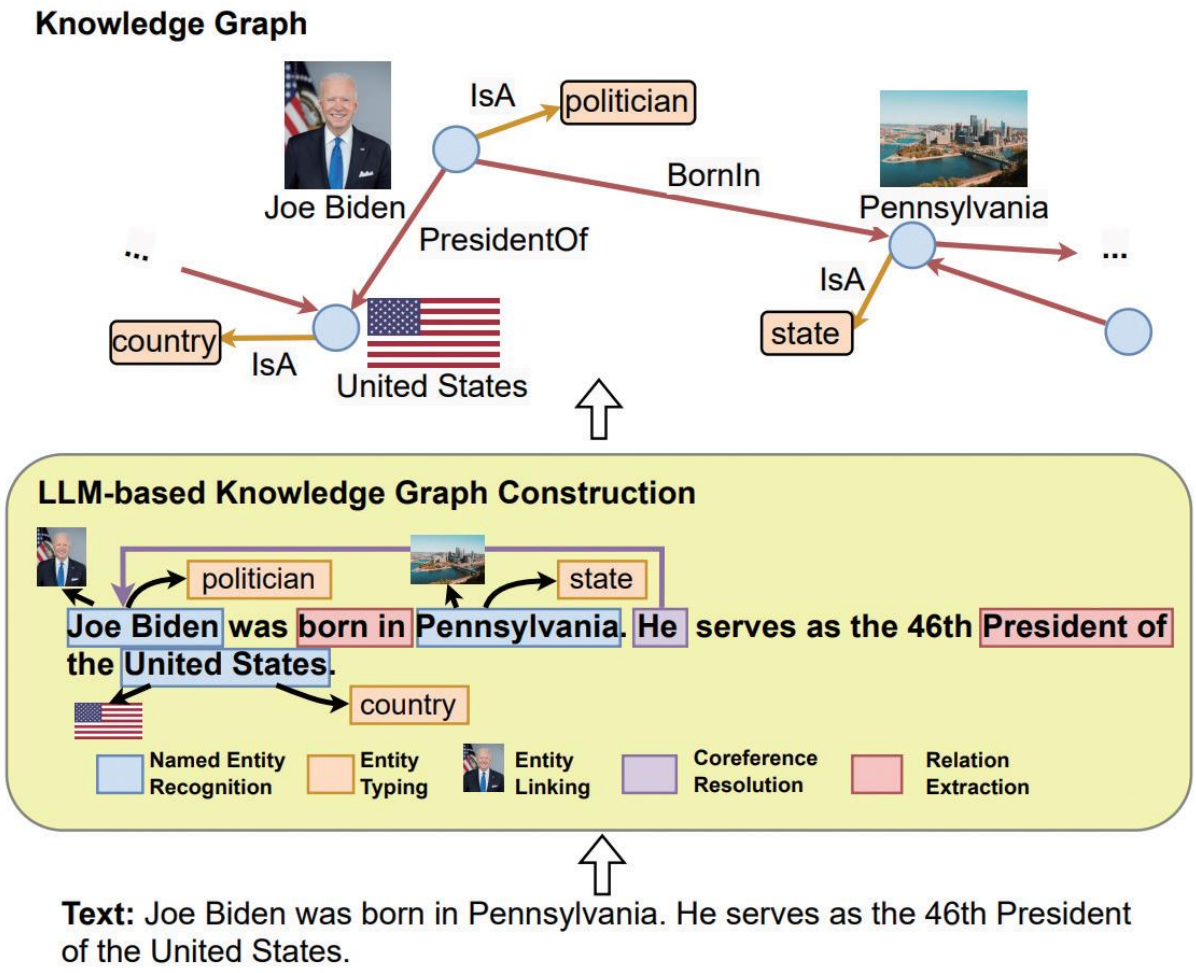


그림 7.9 LLM 기반 지식 그래프 구축의 일반적인 프레임워크(2023년에 발표된 논문에서 발췌, https://arxiv.org/pdf/2306.08302)

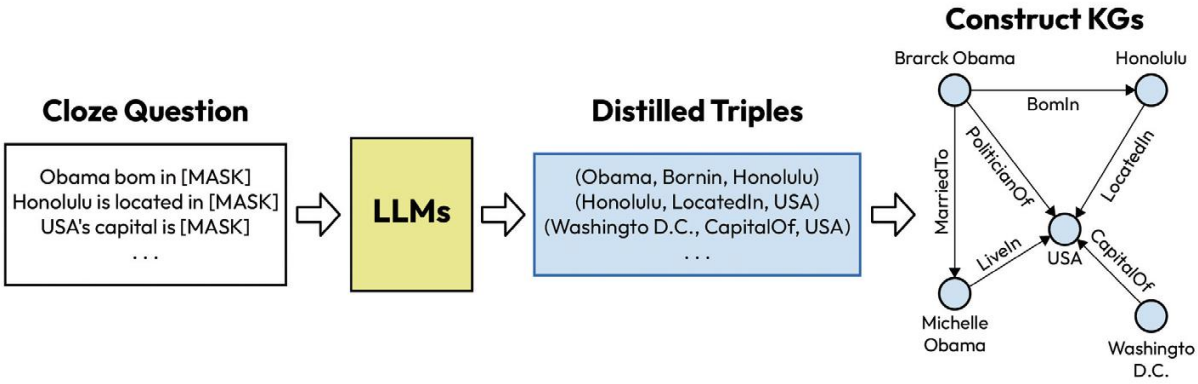


그림 7.10 LLM으로부터 지식 그래프를 추출하는 일반적인 프레임워크 (https://arxiv.org/pdf/2306.08302)

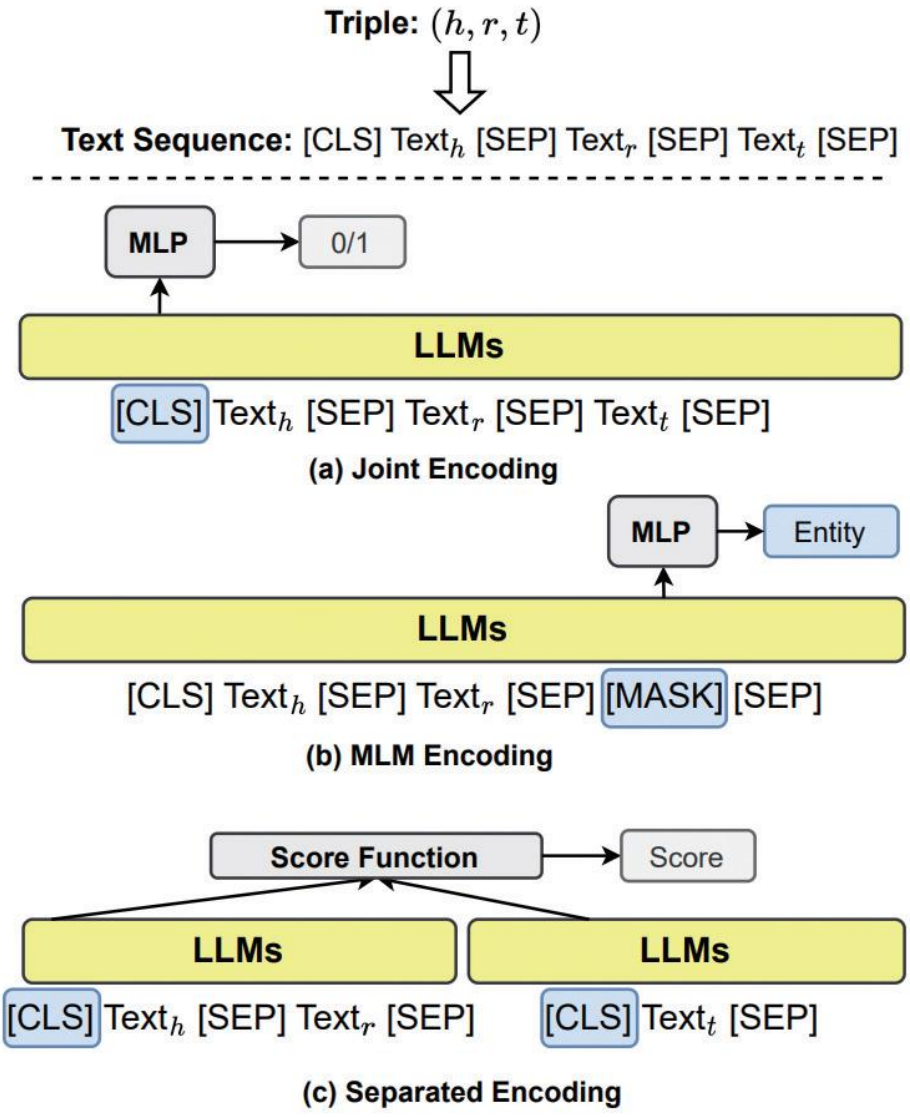


그림 7.12 지식 그래프 완성을 위한 인코더로 활용되는 LLM(https://arxiv.org/pdf/2306.08302)

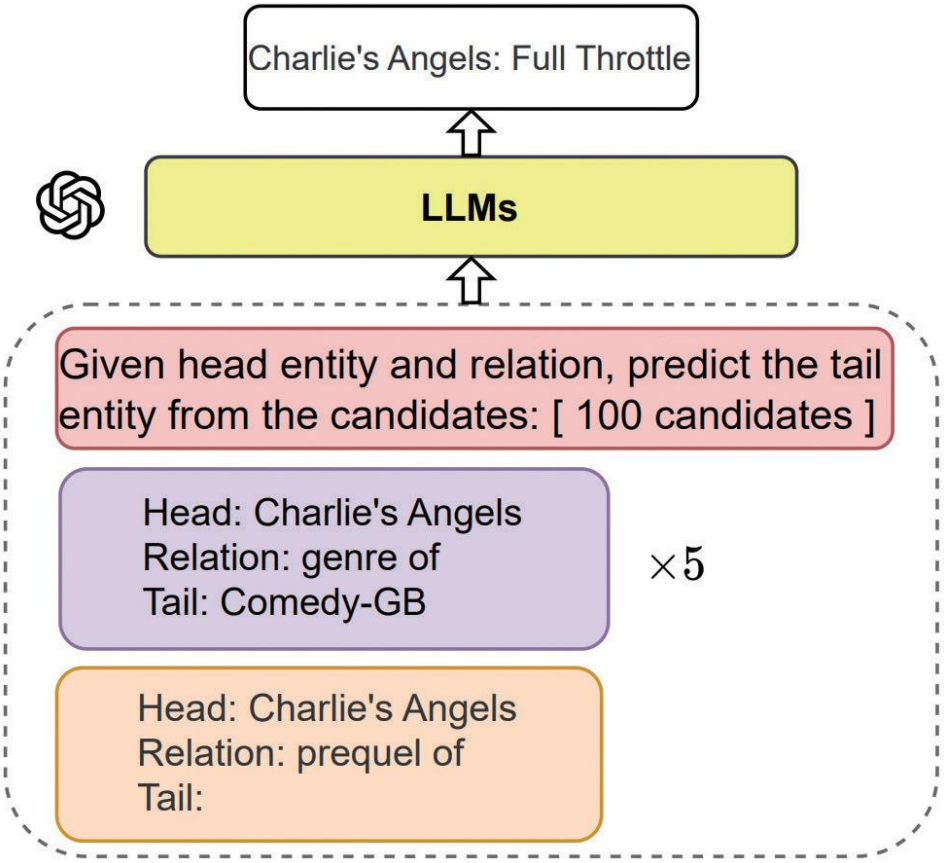


그림 7.13 프롬프트 기반 지식 그래프 완성(https://arxiv.org/pdf/2306.08302)

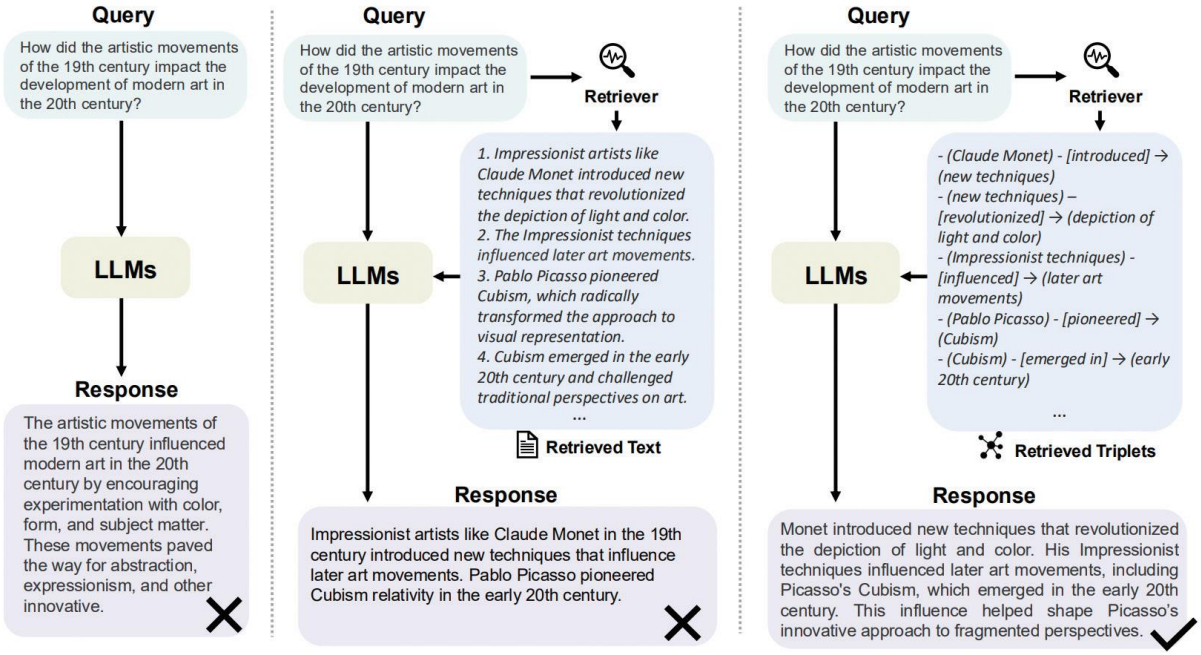


그림 7.14 LLM과 RAG, 그래프 RAG 비교(https://arxiv.org/pdf/2408.08921)

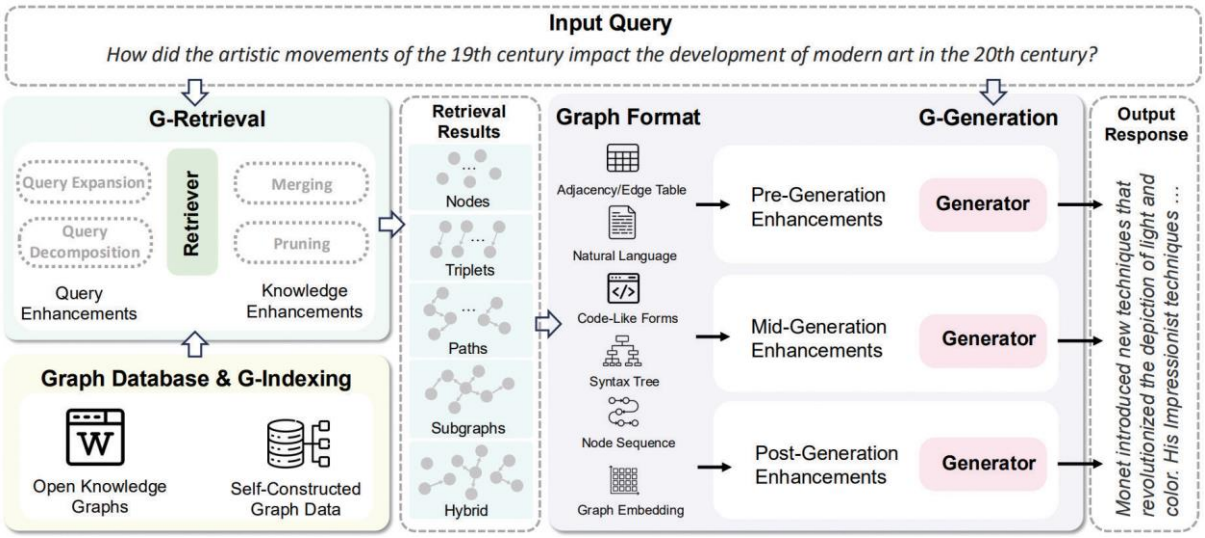


그림 7.15 질의응답 작업을 위한 그래프 RAG 프레임워크 개요 (https://arxiv.org/pdf/2408.08921)

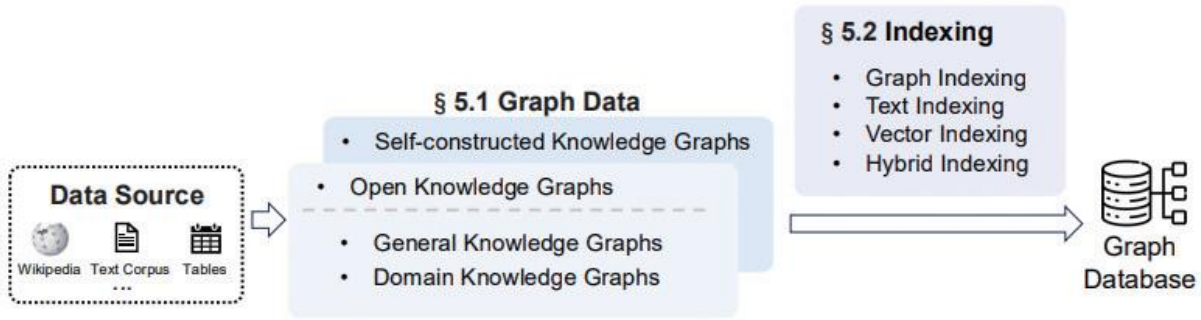


그림 7.16 그래프 기반 인덱싱 개요(https://arxiv.org/pdf/2408.08921)

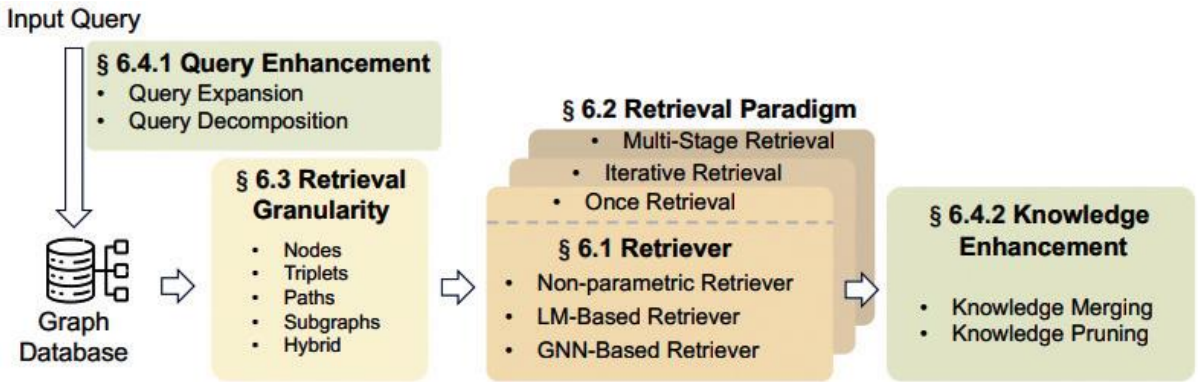


그림 7.17 그래프 기반 검색의 일반적인 아키텍처(https://arxiv.org/pdf/2408.08921)

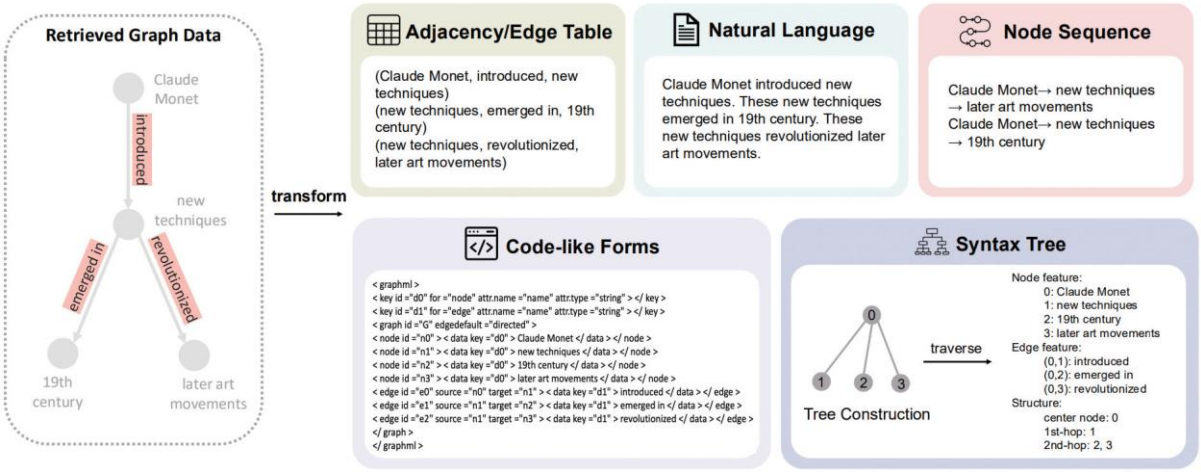


그림 7.19 LLM 응답 생성을 강화하기 위한 부분 그래프 변환 (https://arxiv.org/pdf/2408.08921)

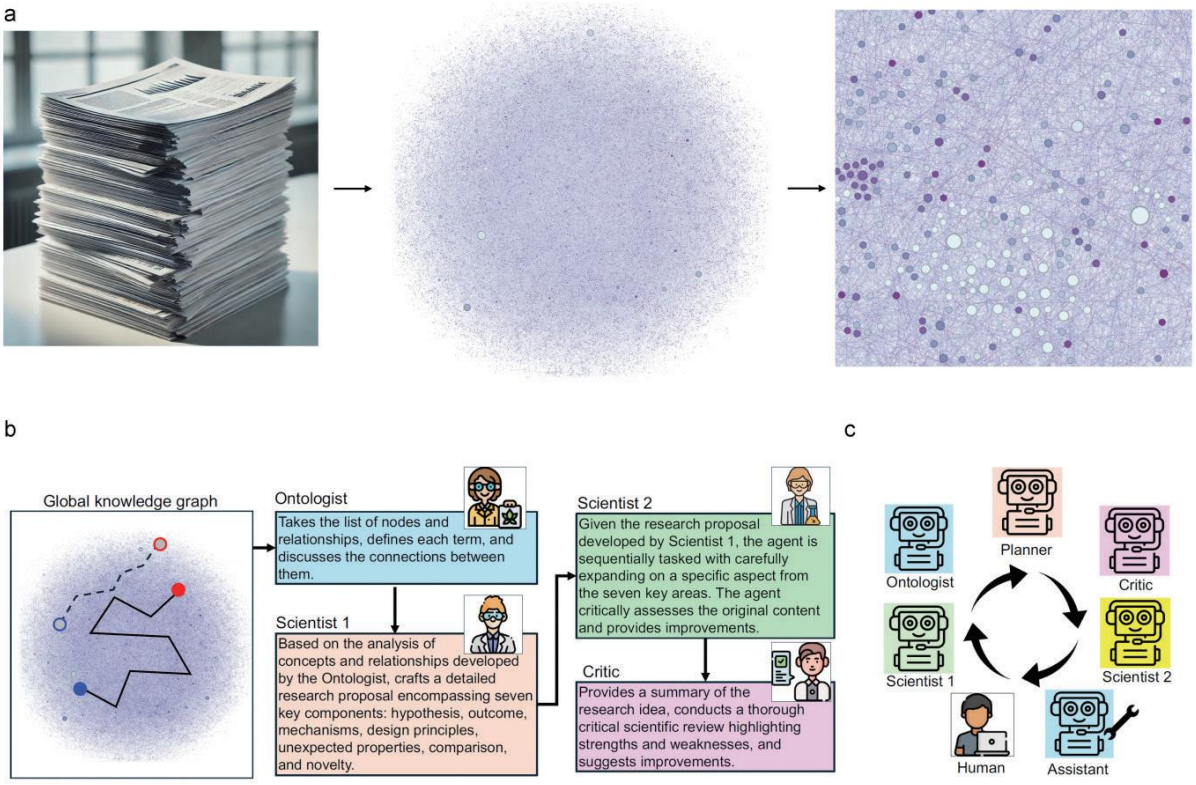


그림 7.20 과학 연구를 지원하는 다중 에이전트 그래프 추론 시스템 개요 (https://arxiv.org/pdf/2409.05556v1)

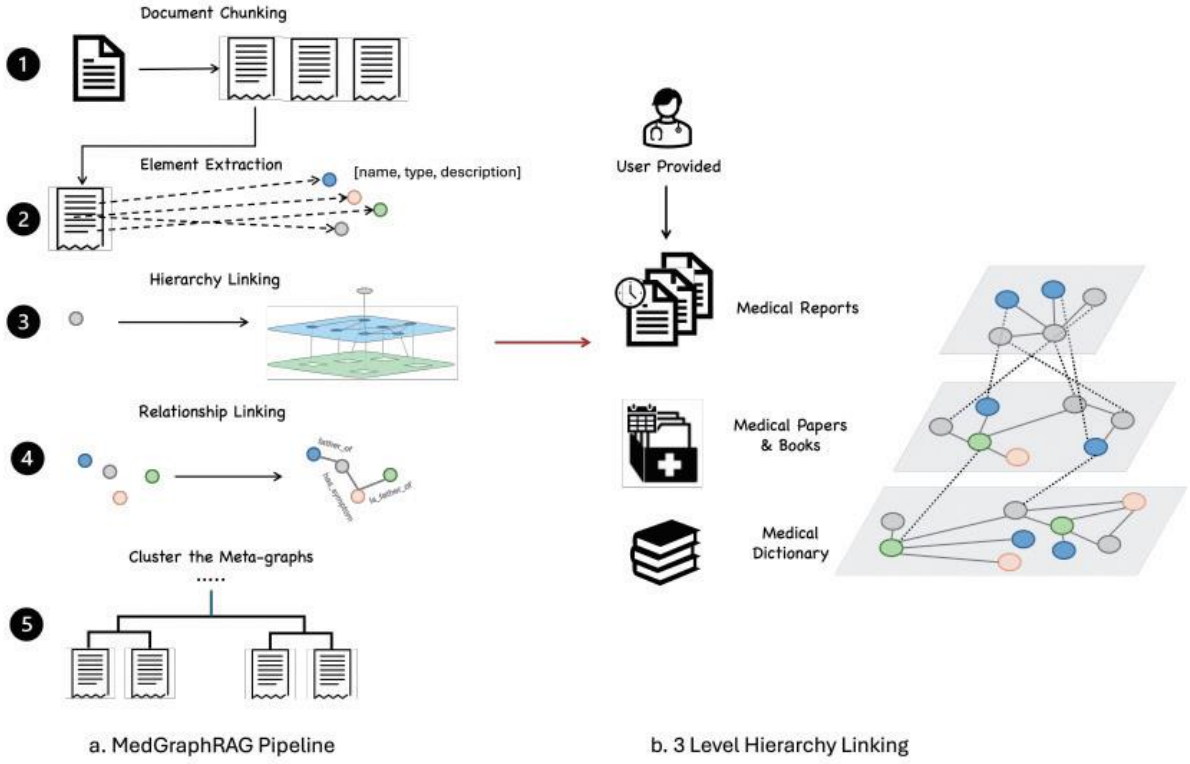


그림 7.21 MedGraphRAG 프레임워크(https://arxiv.org/pdf/2408.04187)

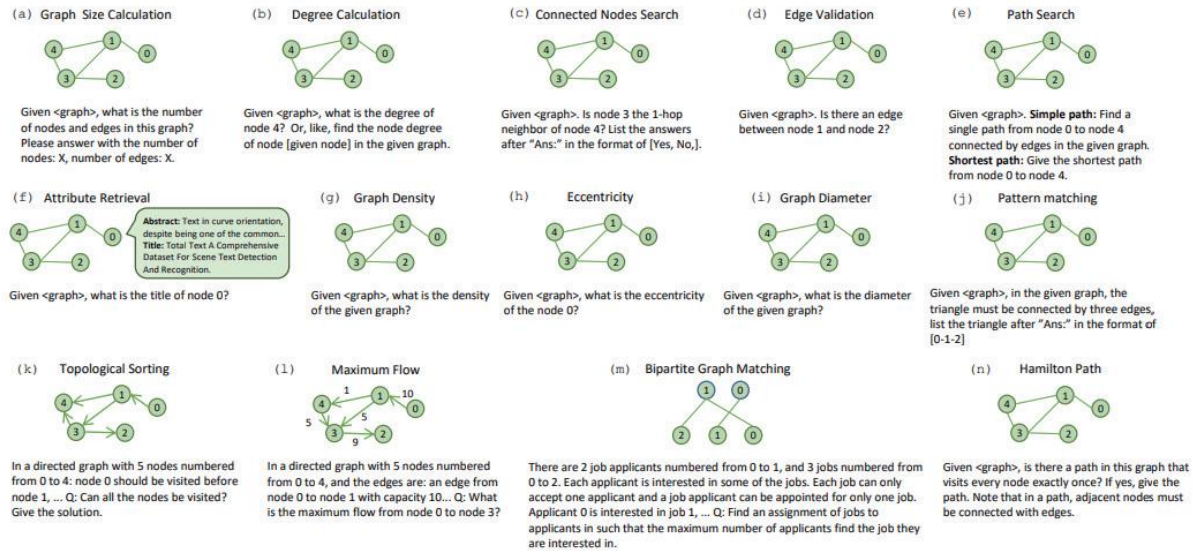


그림 7.22 그래프 구조 이해 작업(https://arxiv.org/pdf/2404.14809)

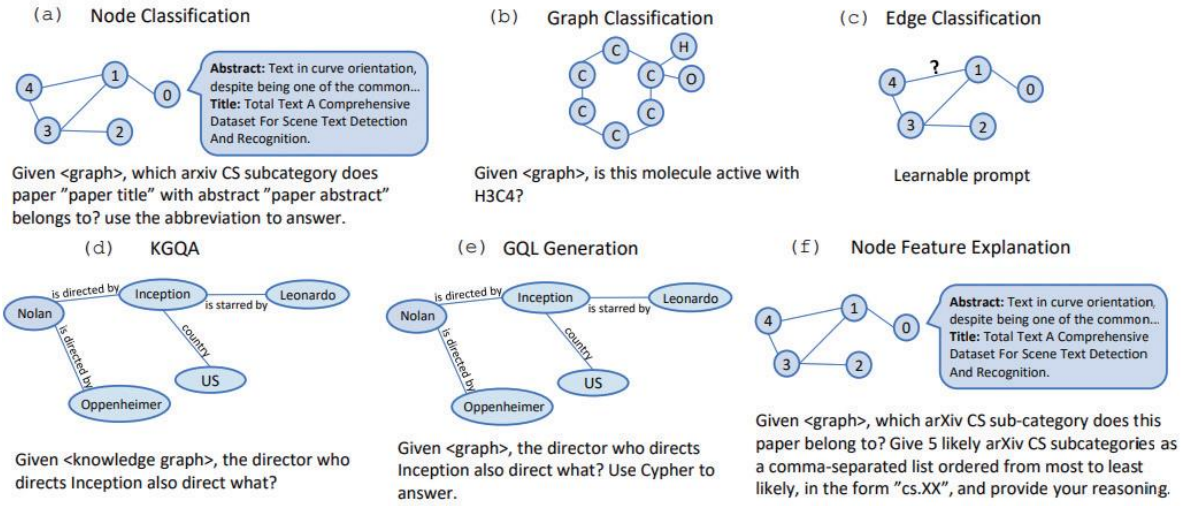
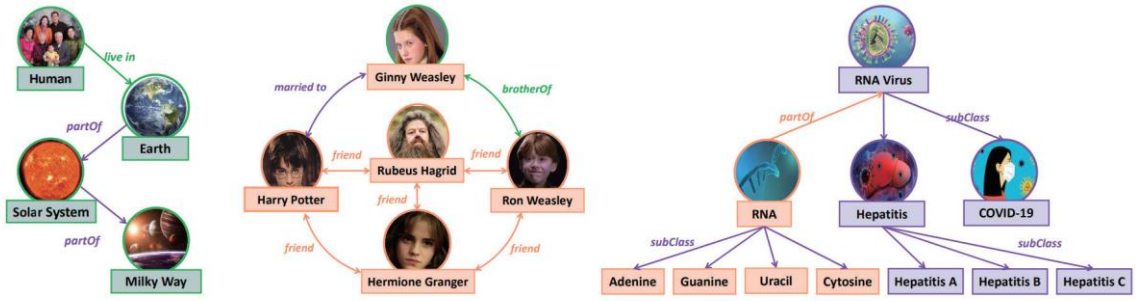


그림 7.23 그래프 학습 작업(https://arxiv.org/pdf/2404.14809)



(a) Chain structure. (b) Ring structure. (c) Hierarchy structure.

그림 7.24 지식 그래프의 세 가지 전형적 구조(https://arxiv.org/pdf/2211.03536)

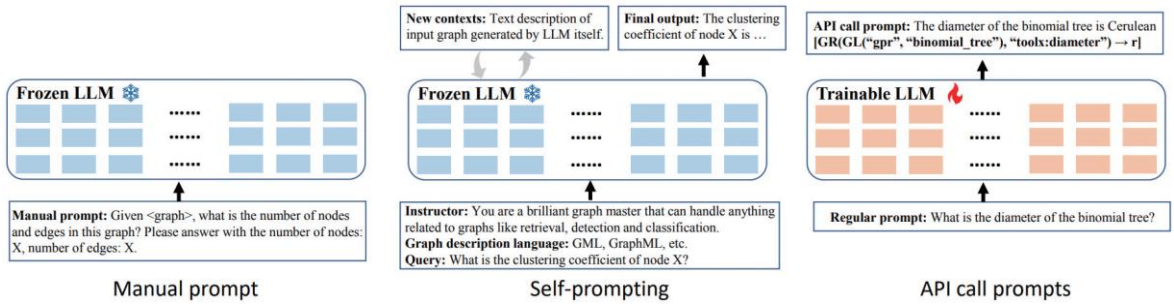


그림 7.26 그래프 작업에 적용한 LLM 프롬프트 기법(https://arxiv.org/pdf/2404.14809)

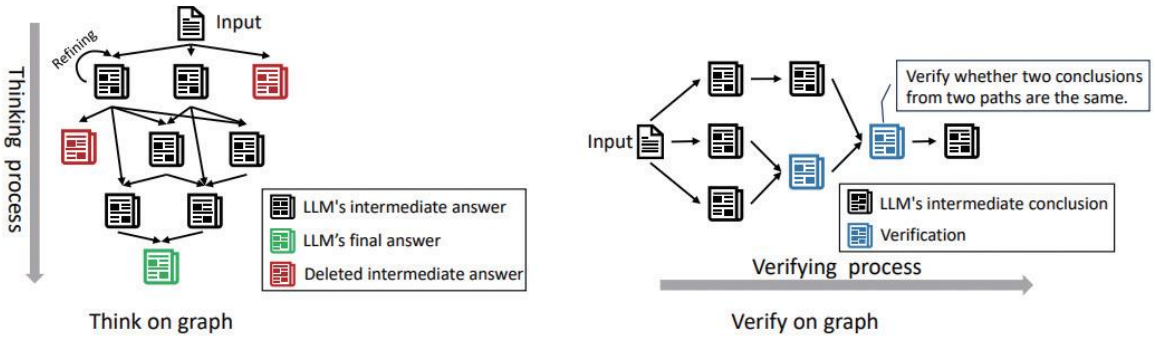


그림 7.28 그래프 위에서 사고하기와 그래프 위에서 검증하기

(https://arxiv.org/pdf/2404.14809)

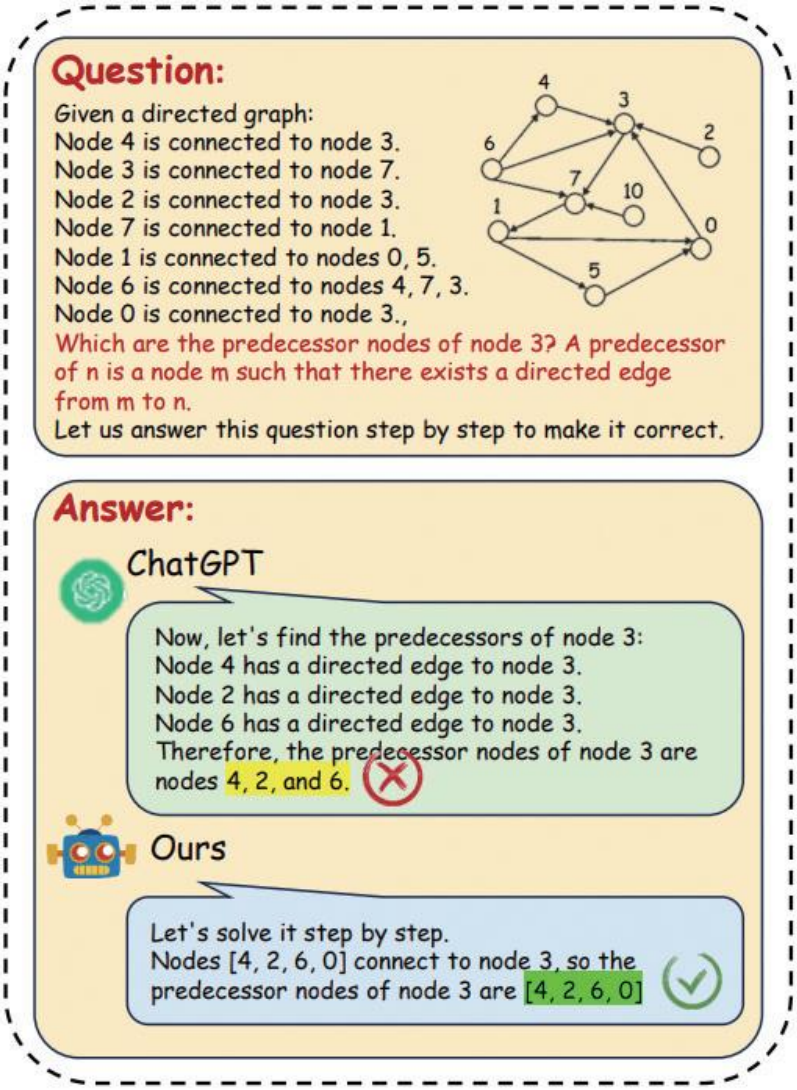


그림 7.29 그래프 데이터에 대한 SFT는 작은 LLM이 더 큰 LLM보다 나은 성능을 발휘하게 한다(https://arxiv.org/pdf/2403.04483)

8장 강화학습과 AI 에이전트

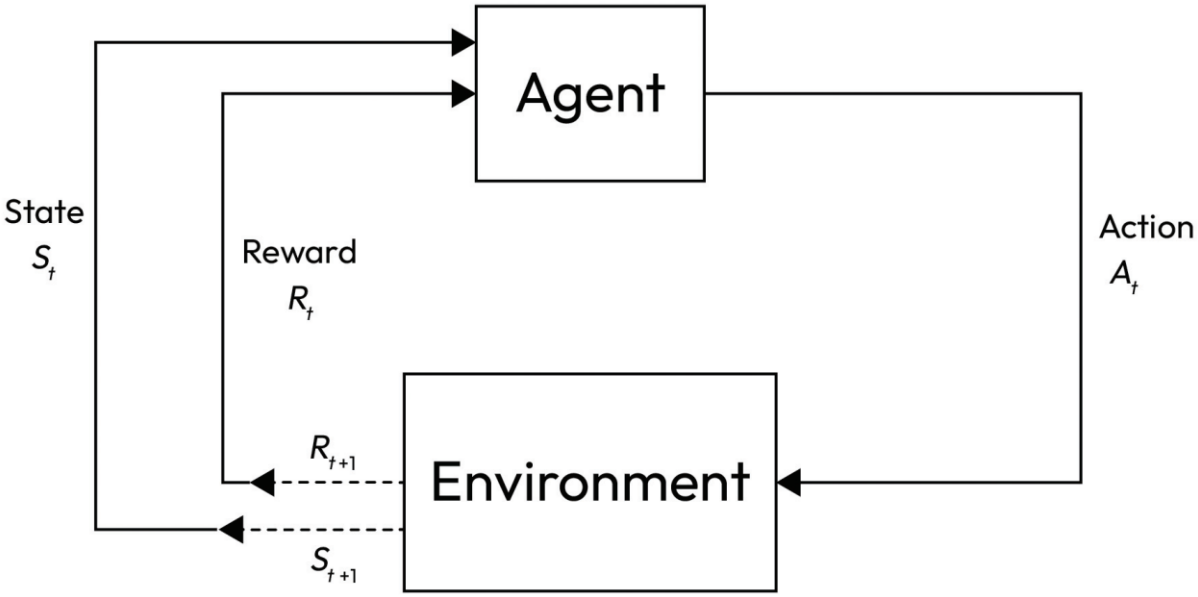


그림 8.4 강화학습 시스템의 개요 모델(https://arxiv.org/pdf/2408.07712)

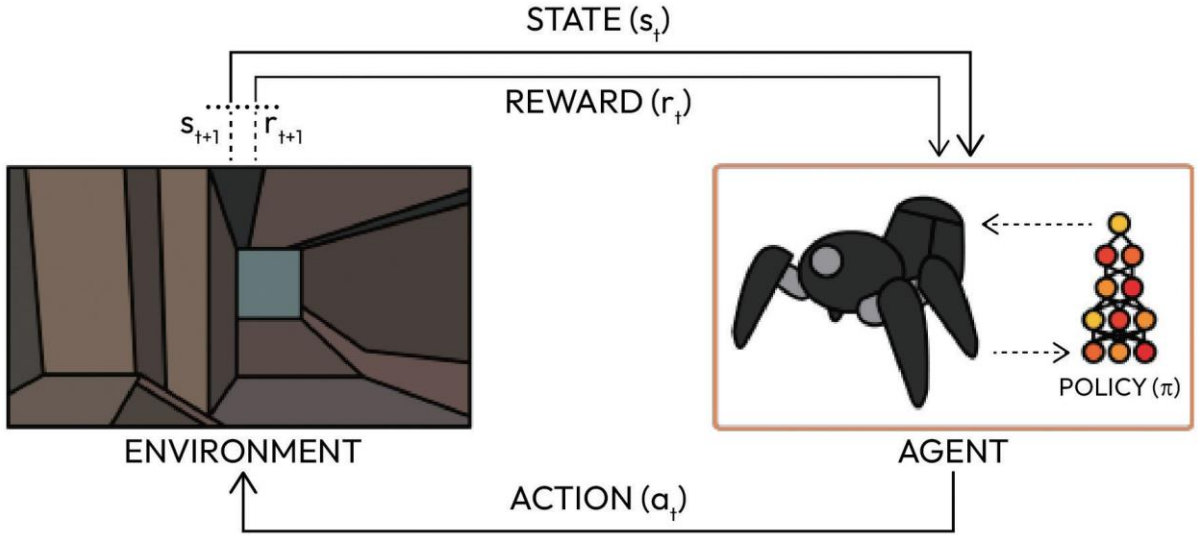


그림 8.13 심층 강화학습 개요(https://arxiv.org/abs/1708.05866)

그림 8.17 신경망을 활용해 아타리 게임 학습을 수행하는 예시 스크린샷 (https://arxiv.org/abs/1312.5602)

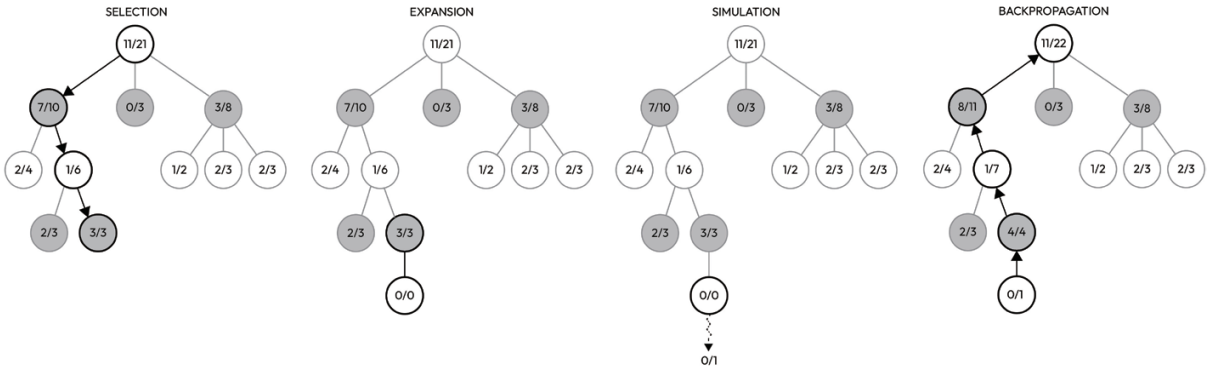


그림 8.24 몬테카를로 트리 탐색(https://en.wikipedia.org/wiki/Monte_Carlo_tree_search)

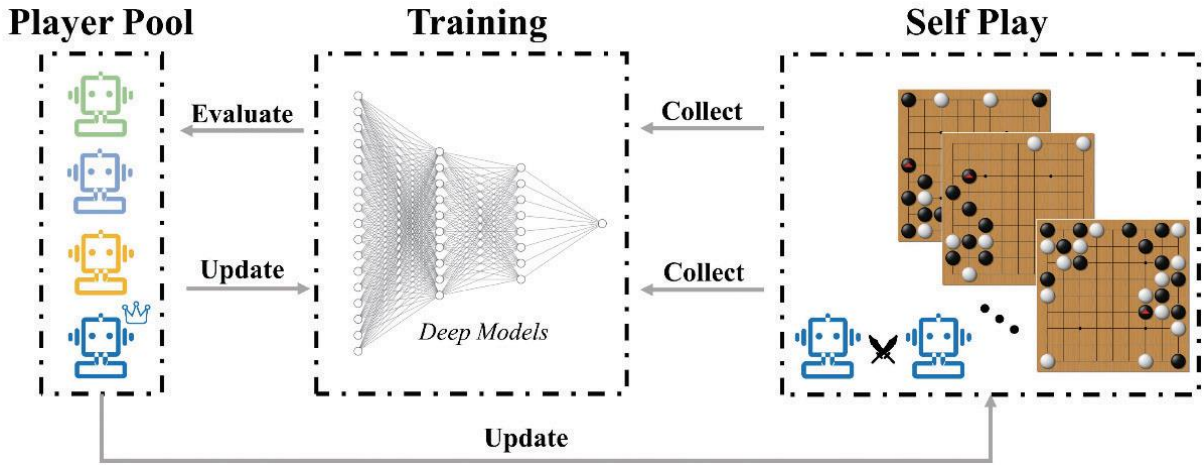


그림 8.25 AlphaZero 파이프라인(https://www.mdpi.com/2079-9292/10/13/1533)

그림 8.29 스크립트 실행 화면(동영상 링크: https://www.youtube.com/watch?v=YWx-hnvqjr8)

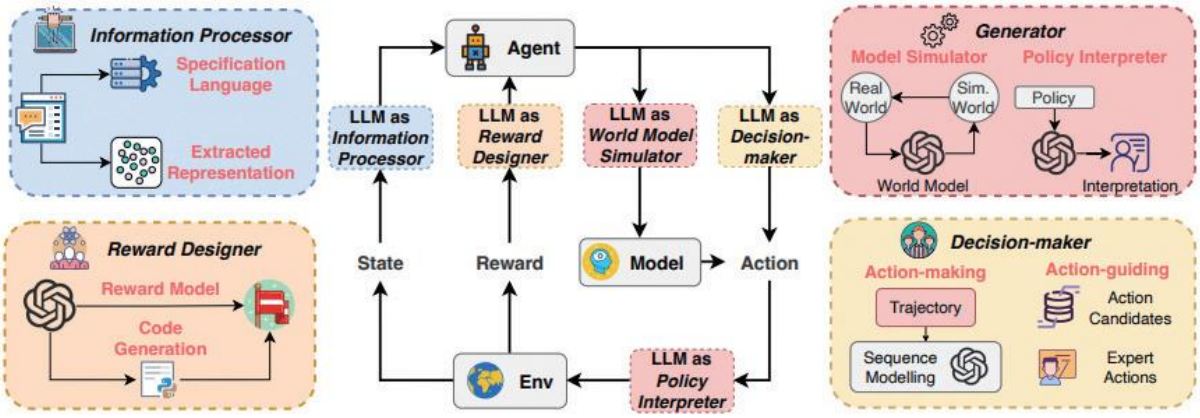


그림 8.30 전통적인 에이전트-환경 상호작용에서 LLM으로 강화된 강화학습 프레임워크 (https://arxiv.org/pdf/2404.00282)

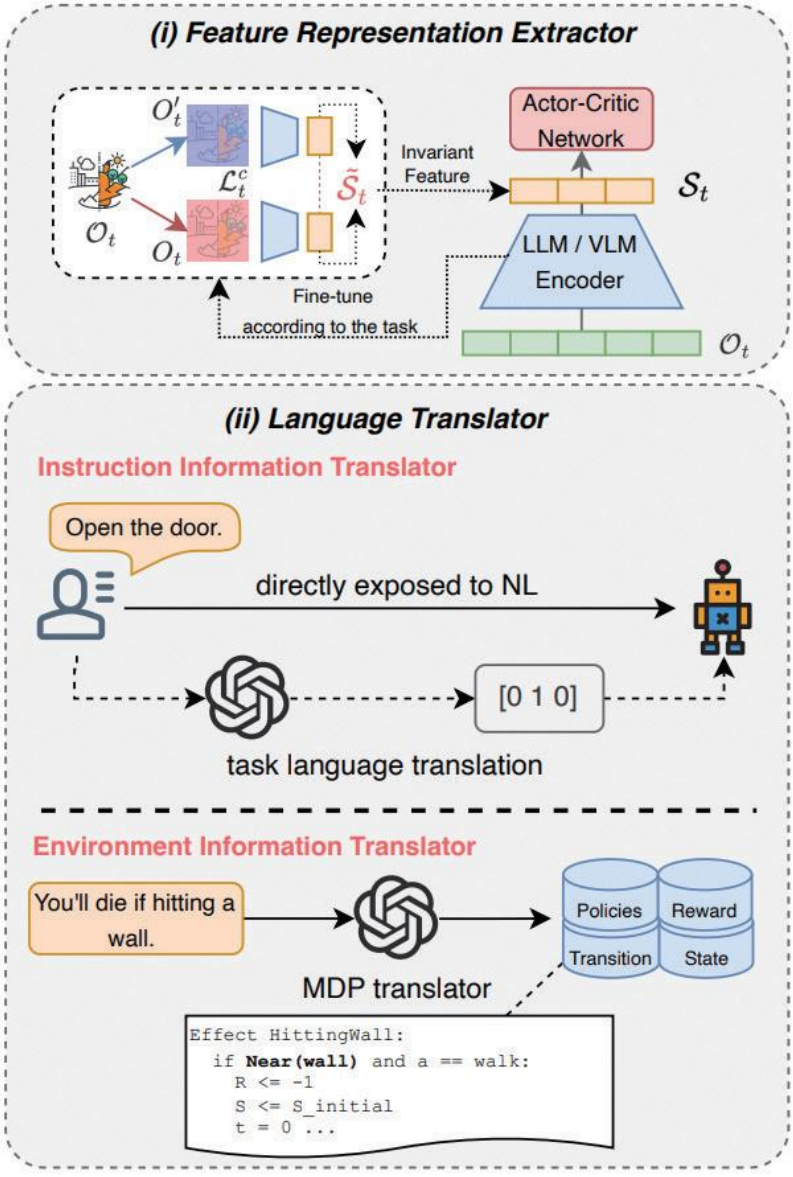


그림 8.31 정보처리자로서의 LLM(https://arxiv.org/pdf/2404.00282)

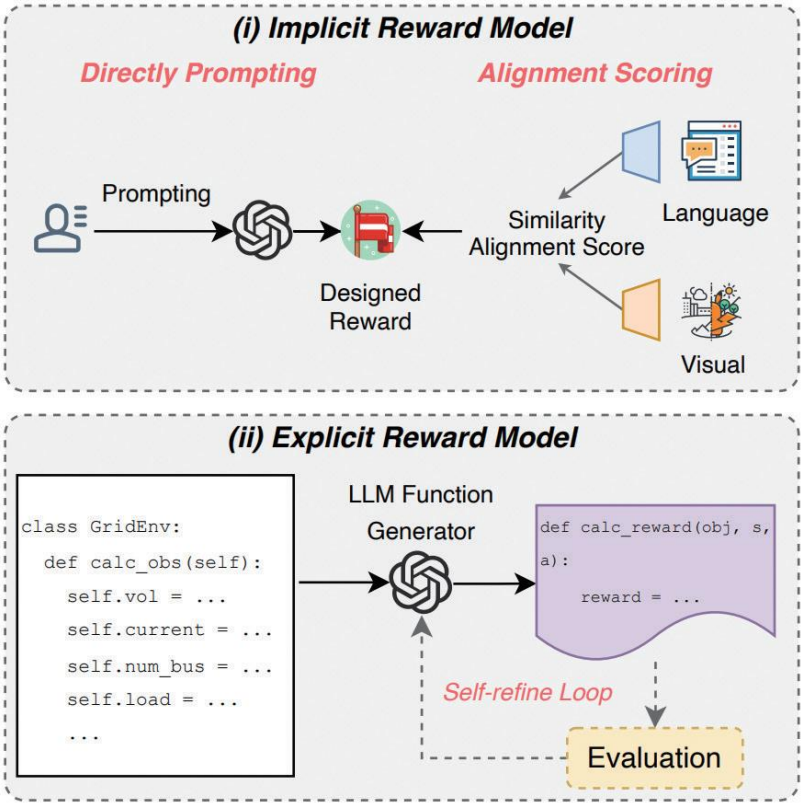


그림 8.32 보상설계자로서의 LLM(https://arxiv.org/pdf/2404.00282)

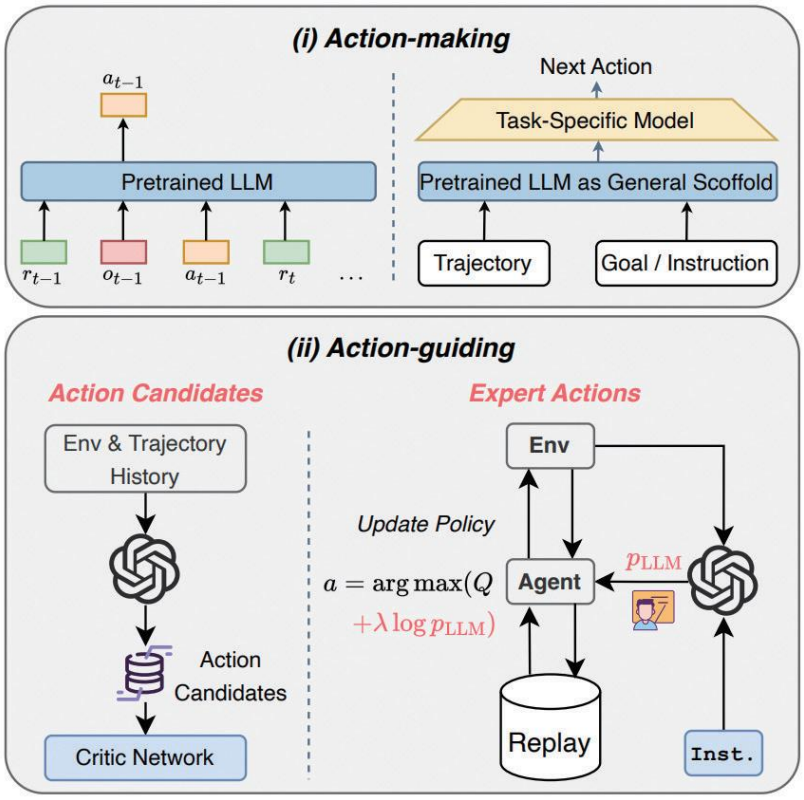


그림 8.33 의사결정자로서의 LLM(https://arxiv.org/pdf/2404.00282)

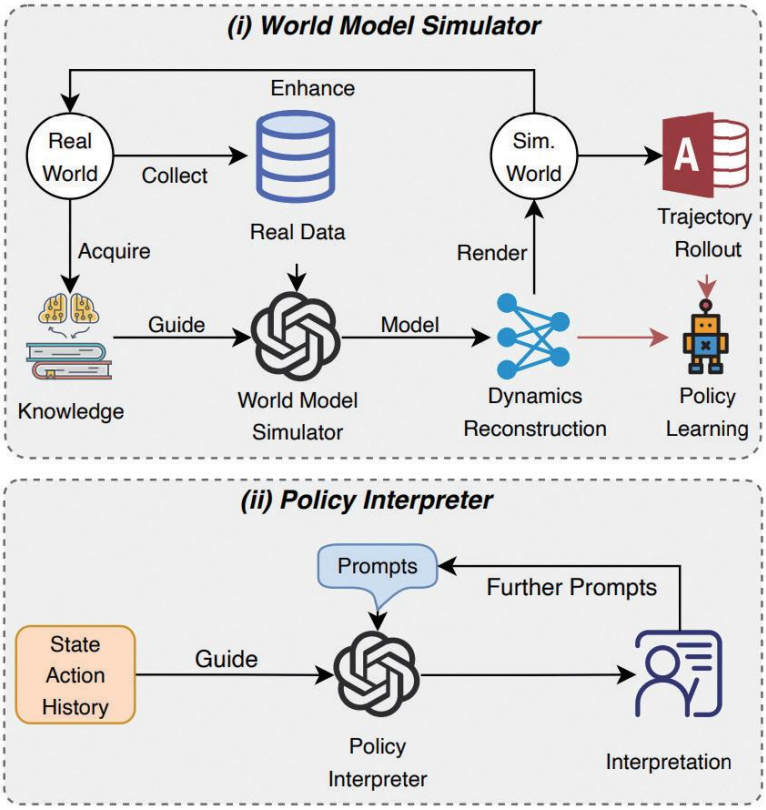


그림 8.34 생성자로서의 LLM(https://arxiv.org/pdf/2404.00282)

9장 단일·다중 에이전트 시스템 만들기

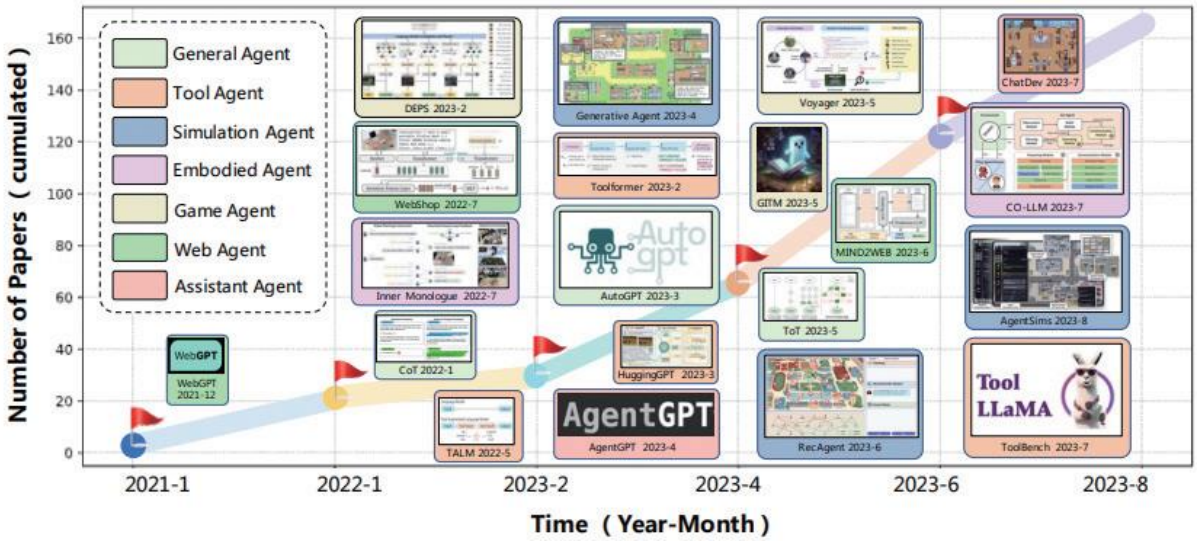


그림 9.1 LLM 자율 에이전트에 관한 관심 증가(https://arxiv.org/pdf/2308.11432)

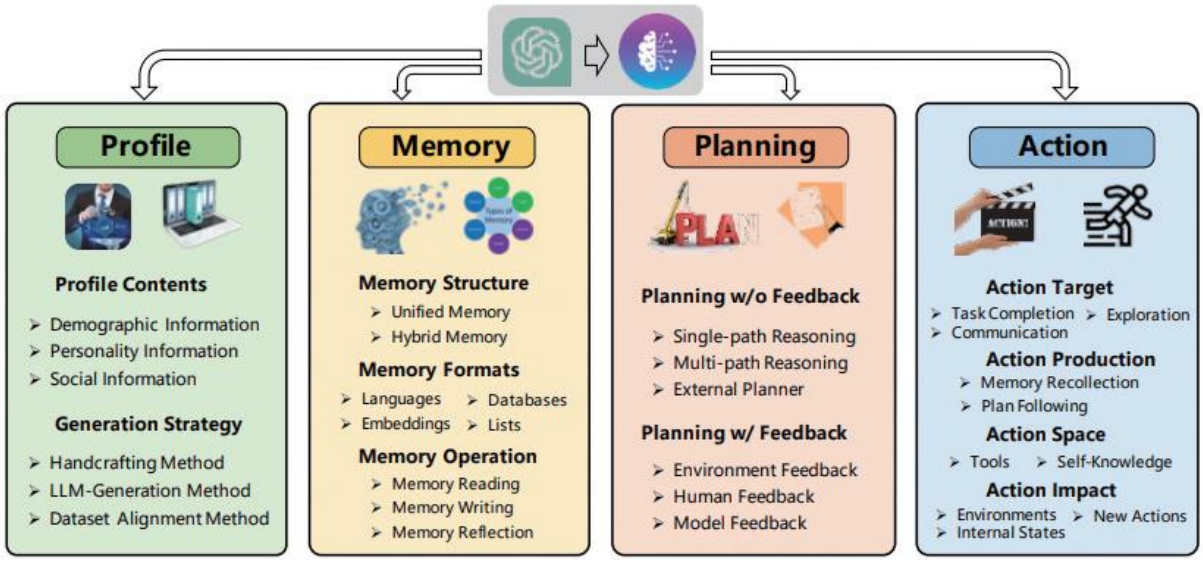


그림 9.2 LLM 기반 자율 에이전트를 구축하는 데 필요한 모듈 (https://arxiv.org/pdf/2308.11432)

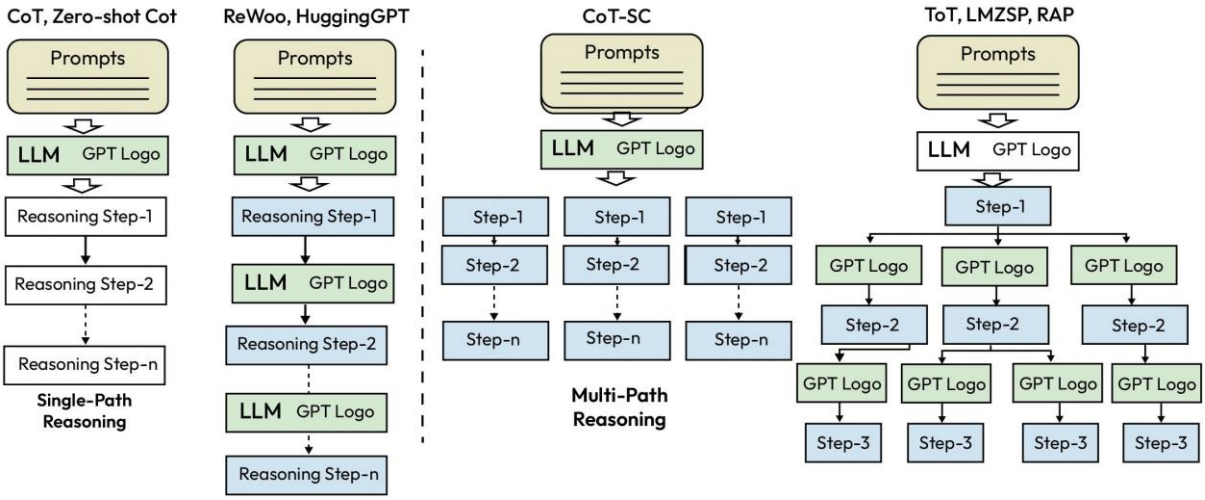


그림 9.3 단일 경로 추론과 다중 경로 추론 전략 비교(https://arxiv.org/pdf/2308.11432)

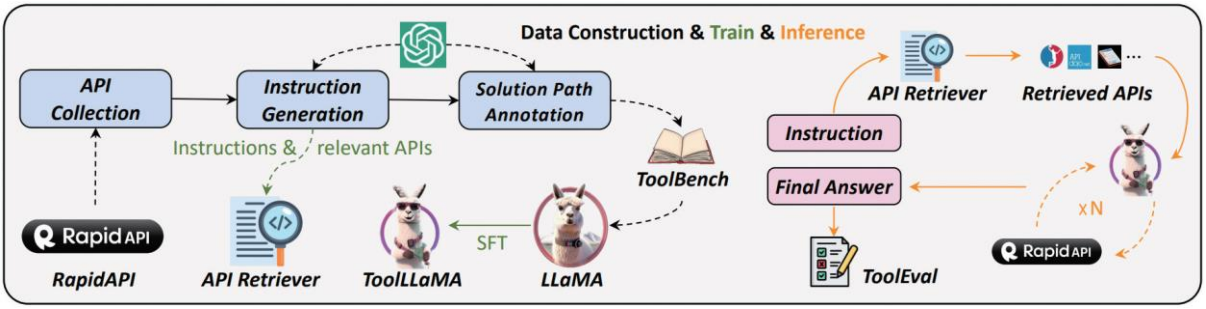


그림 9.4 ToolBench의 구성(https://arxiv.org/pdf/2307.16789)

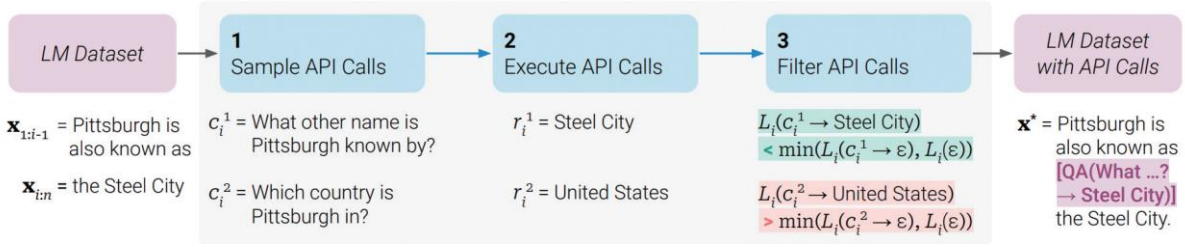


그림 9.5 툴포머 접근 방식(https://arxiv.org/pdf/2302.04761)

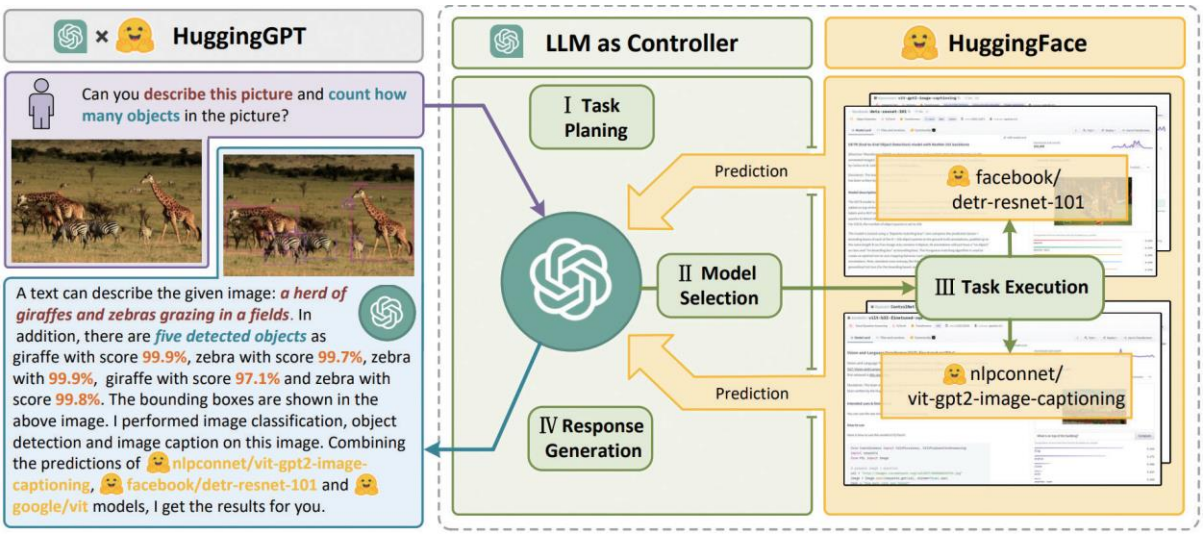


그림 9.6 허깅GPT의 일반 구조(https://arxiv.org/pdf/2303.17580)

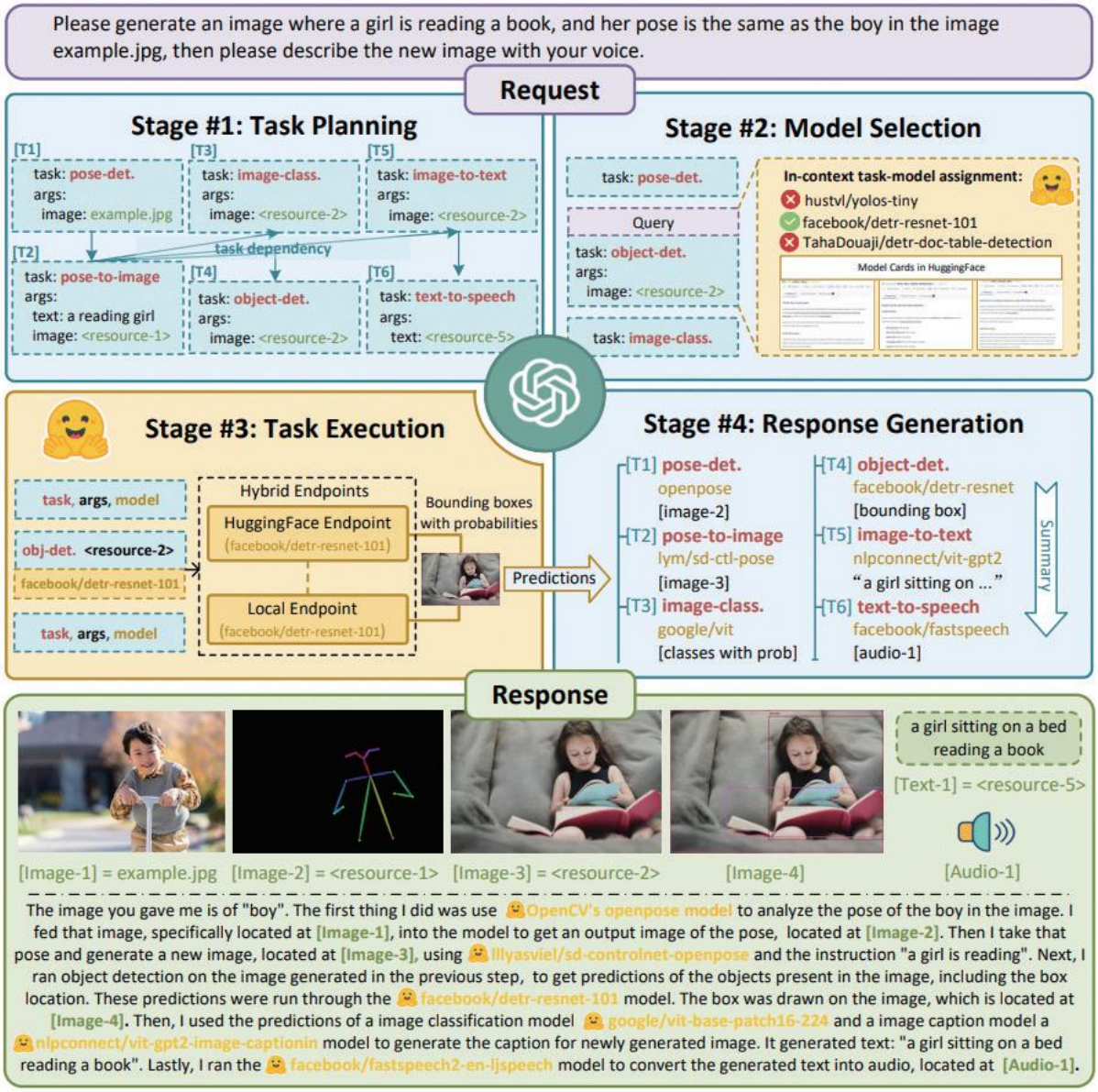


그림 9.7 허깅GPT 프로세스(https://arxiv.org/pdf/2303.17580)

Task	Args
Text-cls	text
Token-cls	text
Text2text-generation	text
Summarization	text
Translation	text
Question-answering	text
Conversational	text
Text-generation	text
Tabular-cls	text

Table 1: NLP tasks.

Task	Args
Image-to-text	image
Text-to-image	image
VQA	text + image
Segmentation	image
DQA	text + image
Image-cls	image
Image-to-image	image
Object-detection	image
Controlnet-sd	image

Table 2: CV tasks.

Task	Args
Text-to-speech	text
Audio-cls	audio
ASR	audio
Audio-to-audio	audio

Table 3: Audio tasks.

Task	Args
Text-to-video	text
Video-cls	video

Table 4: Video tasks.

그림 9.8 허깅GPT 작업 유형(https://arxiv.org/pdf/2303.17580)

	Prompt	
Task Planning	#1 Task Planning Stage - The AI assistant performs task parsing on user input, generating a list of tasks with the following format: [{"task": task, "id", task_id, "dep": dependency_task_ids, "args": {"text": text, "image": URL, "audio": URL, "video": URL}}]. The "dep" field denotes the id of the previous task which generates a new resource upon which the current task relies. The tag "<resource>-task_id" represents the generated text, image, audio, or video from the dependency task with the corresponding task_id. The task must be selected from the following options: {{ *Available Task List* }}. Please note that there exists a logical connections and order between the tasks. In case the user input cannot be parsed, an empty JSON response should be provided. Here are several cases for your reference: {{ *Demonstrations* }}. To assist with task planning, the chat history is available as {{ *Chat Logs* }}, where you can trace the user-mentioned resources and incorporate them into the task planning stage.	
	Demonstrations	
	Can you tell me how many objects in e1.jpg?	[{"task": "object-detection", "id": 0, "dep": [-1], "args": {"image": "e1.jpg" }}]
	In e2.jpg, what's the animal and what's it doing?	[{"task": "image-to-text", "id": 0, "dep":[-1], "args": {"image": "e2.jpg" }}, {"task":"image-cls", "id": 1, "dep": [-1], "args": {"image": "e2.jpg" }}, {"task":"object-detection", "id": 2, "dep": [-1], "args": {"image": "e2.jpg" }}, {"task": "visual-quesrion-answering", "id": 3, "dep":[-1], "args": {"text": "what's the animal doing?", "image": "e2.jpg" }}]
	First generate a HED image of e3.jpg, then based on the HED image and a text "a girl reading a book", create a new image as a response.	[{"task": "pose-detection", "id": 0, "dep": [-1], "args": {"image": "e3.jpg" }}, {"task": "pose-text-to-image", "id": 1, "dep": [0], "args": {"text": "a girl reading a book", "image": "<resource>-0" }}]

그림 9.9 허깅GPT의 프롬프트 설계 세부 사항(https://arxiv.org/pdf/2303.17580)

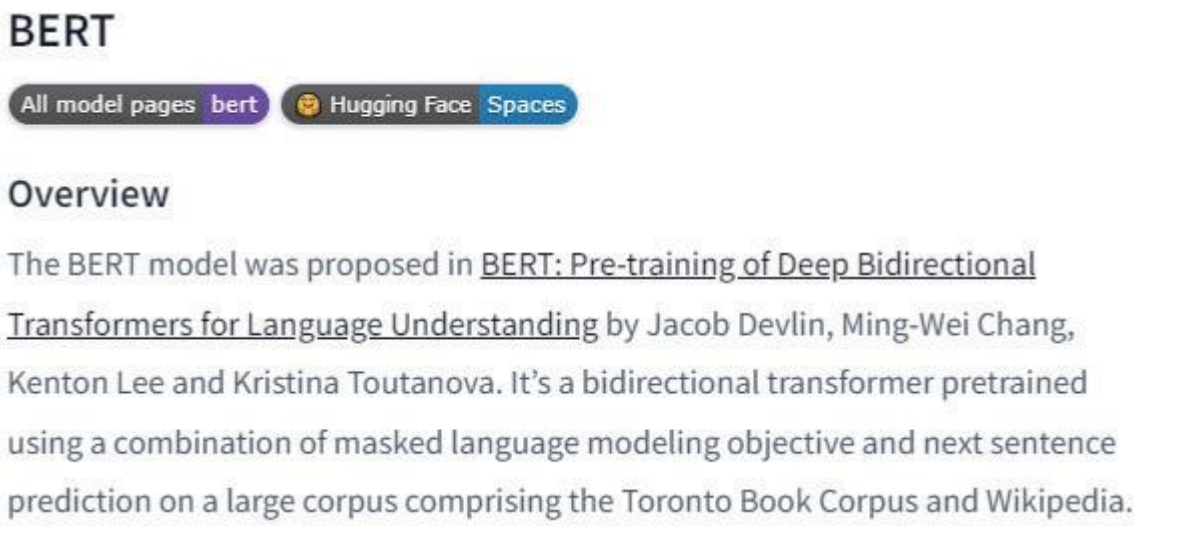


그림 9.10 Hugging Face의 모델 설명 예시
(https://huggingface.co/docs/transformers/model_doc/bert)

	Prompt
Model Selection	#2 Model Selection Stage - Given the user request and the call command, the AI assistant helps the user to select a suitable model from a list of models to process the user request. The AI assistant merely outputs the model id of the most appropriate model. The output must be in a strict JSON format: {"id": "id", "reason": "your detail reason for the choice"}. We have a list of models for you to choose from {{ *Candidate Models* }}. Please select one model from the list.
	Candidate Models
	{"model_id": model id #1, "metadata": meta-info #1, "description": description of model #1} {"model_id": model id #2, "metadata": meta-info #2, "description": description of model #2} . . . . {"model_id": model id #K, "metadata": meta-info #K, "description": description of model #K}

그림 9.11 허깅GPT에서 모델 선택을 위한 프롬프트 설계 세부 사항 (https://arxiv.org/pdf/2303.17580)

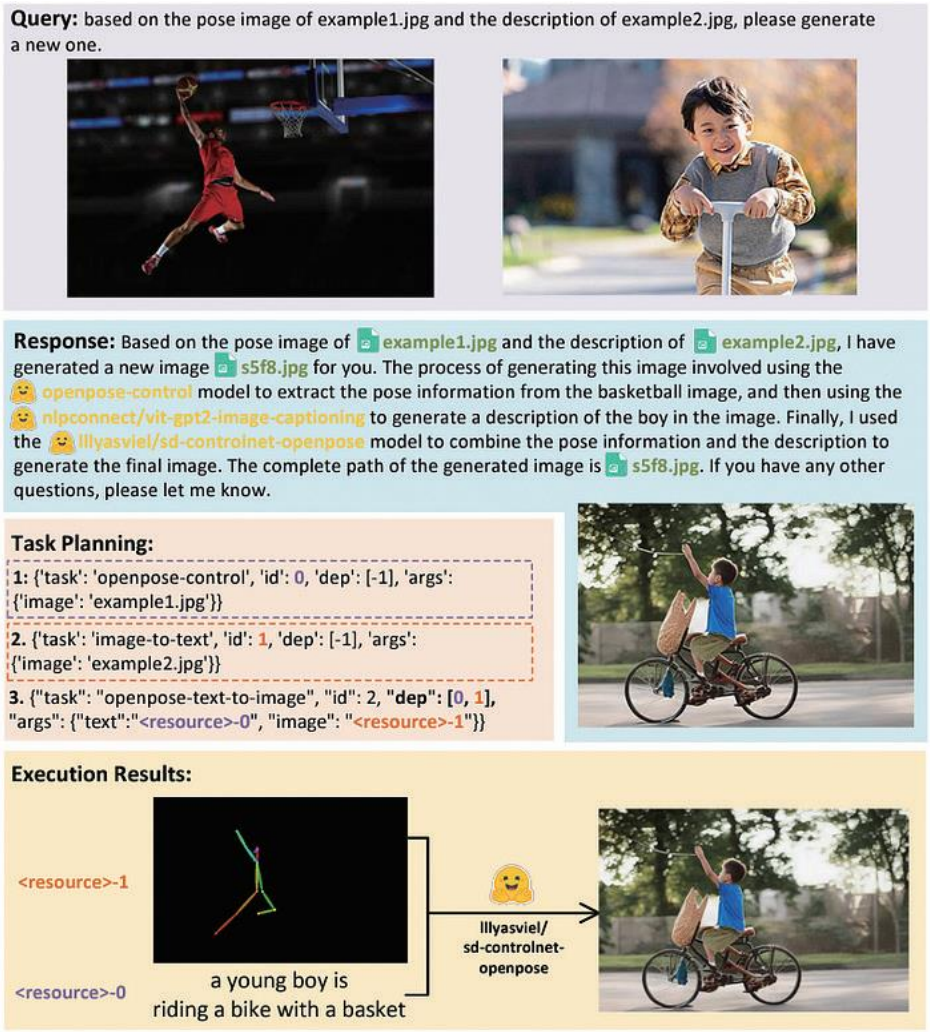


그림 9.12 모델 실행(https://arxiv.org/pdf/2303.17580)

	Prompt
Response Generation	#4 Response Generation Stage - With the input and the inference results, the AI assistant needs to describe the process and results. The previous stages can be formed as - User Input: {{ *User Input* }}, Task Planning: {{ *Tasks* }}, Model Selection: {{ *Model Assignment* }}, Task Execution: {{ *Predictions* }}. You must first answer the user's request in a straightforward manner. Then describe the task process and show your analysis and model inference results to the user in the first person. If inference results contain a file path, must tell the user the complete file path. If there is nothing in the results, please tell me you can't make it.

그림 9.13 응답 생성(https://arxiv.org/pdf/2303.17580)

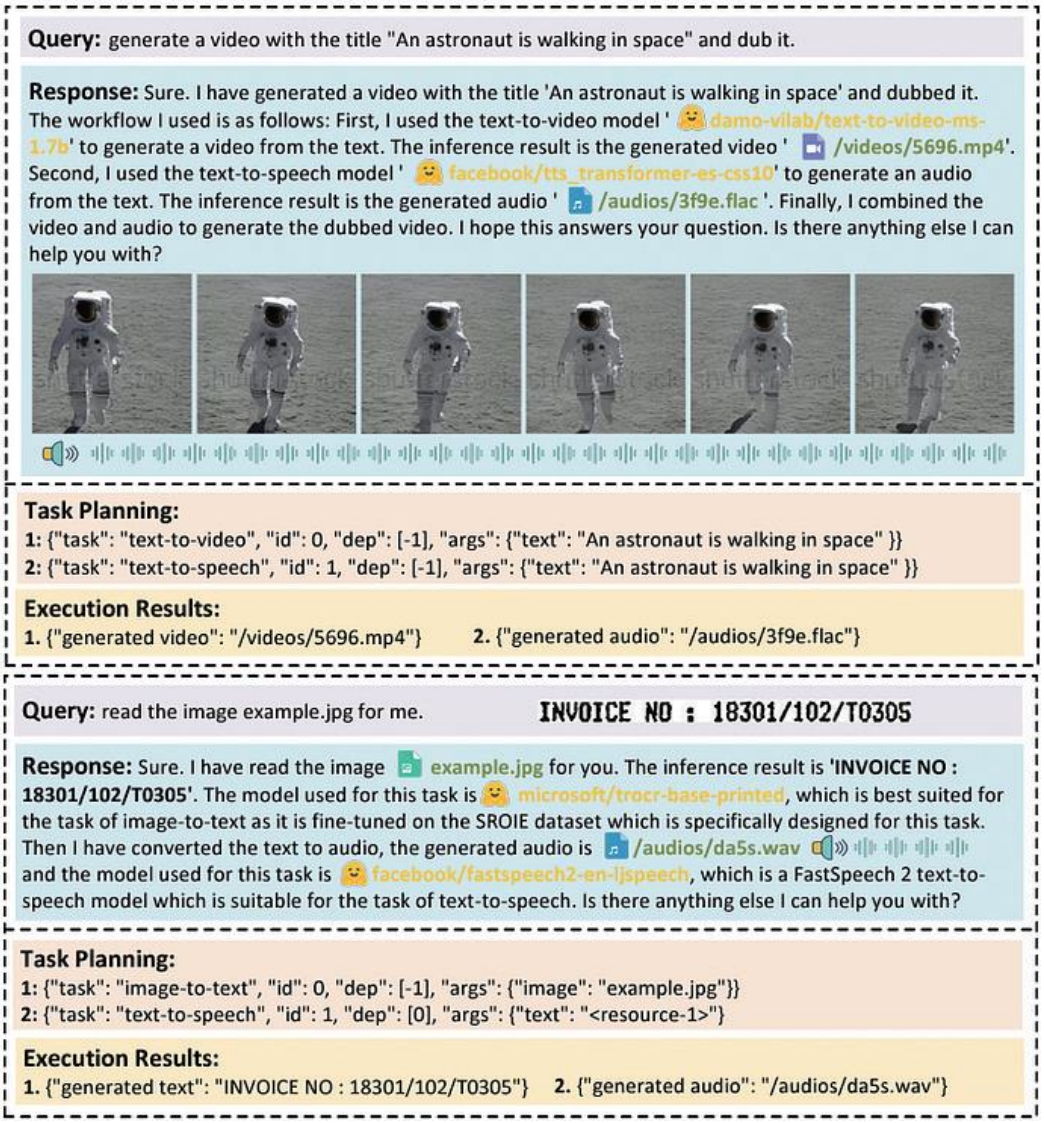


그림 9.14 비디오와 오디오 모달리티에서 다중 모델 협력의 질적 분석 (https://arxiv.org/pdf/2303.17580)

Query: Describe this image example.jpg in as much detail as possible.

Response: The image you provided appears to show **a family of four dogs playing in the grass**. For this task, I ran multiple CV tasks, including image-to-text, image-classification, object-detection, visual-question-answering, and document-question-answering. I chose specific models for each task that were best suited to addressing your query, based on their training and performance characteristics.

For image-to-text, I chose the nlpconnect/vit-gpt2-image-captioning model. This model generates an image caption with the given image. I applied this model to the input image and the developed output was: a family of four dogs are playing in the grass
For image-classification, I selected the google/vit-base-patch16-224 model. This model is trained on natural images dataset and it can predict the label of the image output. I applied this model to the image and get the results showing the risk of each label. It shows the highest risk at "Rhodesian ridgeback" with a score of 93.8%
For object-detection, I used facebook/detr-resnet-101 model. This model is trained on COCO panoptic segmentation for object detection. I used this model to generate an image file with the predicted box. The generated image with predicted box is at: /images/sd48.jpg
Regarding your request for image segmentation, I have completed the task using a pre-trained model called facebook/detr-resnet-50-panoptic. The segmentation mask shows the different regions of the image that belong to different classes. In this case, the model has identified two main classes in the image, which are "dog", "grass" and "tree". The generated image with the segmentation mask can be found at /images/t4s1.jpg
For visual question answering, I used dandelin/vilt-b32-finetuned-vqa model. This model is a Vision-and-Language Transformer (ViLT) model, which is fine-tuned on VQAv2. It is suitable for visual-question-answering tasks, and is likely to provide accurate results for the given image. For the question "what are in the image?" It reply with a prediction "dogs" and the confidence is 84.9%.

Task Planning:

1. {"id": "0", "dep": [-1], "task": "image-to-text", "args": {"image": "example.jpg" }},
2. {"id": "1", "dep": [-1], "task": "image-classification", "args": {"image": "example.jpg" }},
3. {"id": "2", "dep": [-1], "task": "object-detection", "args": {"image": "example.jpg" }},
4. {"id": "3", "dep": [-1], "task": "image-segmentation", "args": {"image": "example.jpg" }},
5. {"id": "4", "dep": [-1], "task": "visual-question-answering", "args": {"text": "describe this image in detail", "image": "example.jpg" }},

Execution Results:

1. {"generated_text": "a family of four dogs are playing in the grass "}
2. [{'score': 0.938717246055603, 'label': 'Rhodesian ridgeback'}, {'score': 0.01652258075773716, 'label': 'boxer'}, {'score': 0.006381669547408819, 'label': 'Great Dane'}, {'score': 0.006234415341168642, 'label': 'vizsla, Hungarian pointer'}, {'score': 0.005944834090769291, 'label': 'bull mastiff'}]
3. [{"generated image with predicted box": "/images/sd48.jpg","predicted": [{"box": {"xmax": 463, "xmin": 373, "ymax": 267, "ymin": 199}, "label": "bus", "score": 0.9981155395507812}, {"box": {"xmax": 292, "xmin": 124, "ymax": 224, "ymin": 175}, "label": "airplane", "score": 0.9983609318733215}, {"box": {"xmax": 554, "xmin": 487, "ymax": 277, "ymin": 226}, "label": "person", "score": 0.9910836219787598}]}
4. {"generated image with segmentation": "/images/t4s1.jpg", "predicted": [{"score": 0.989, "label": "grass"}, {"score": 0.999, "label": "dog"}, {"score": 0.999, "label": "tree"},{"score": 0.999, "label": "dog"}]
5. [{'answer': 'dogs', 'score': 0.8488452434539795}, {'answer': 'dog', 'score': 0.04168461635708809}]

그림 9.15 복잡한 작업 사례 연구(https://arxiv.org/pdf/2303.17580)

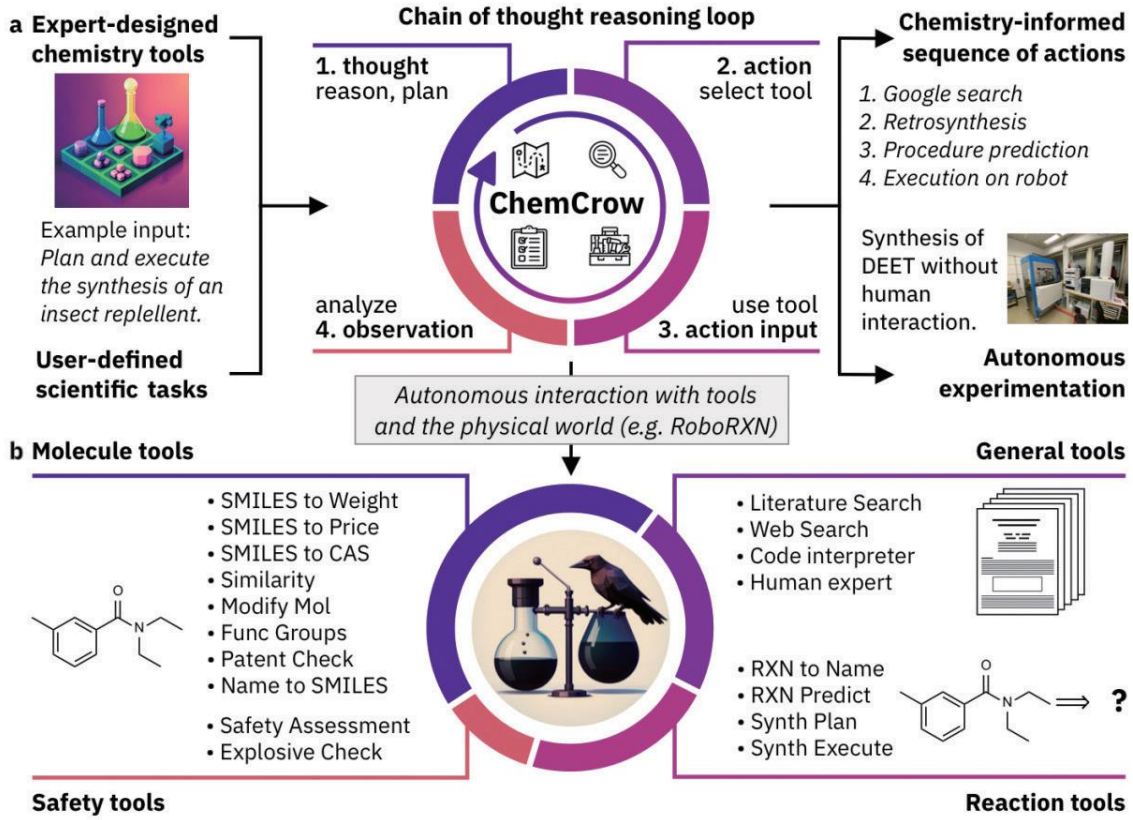


그림 9.16 켐크로우 개요(https://arxiv.org/pdf/2304.05376)

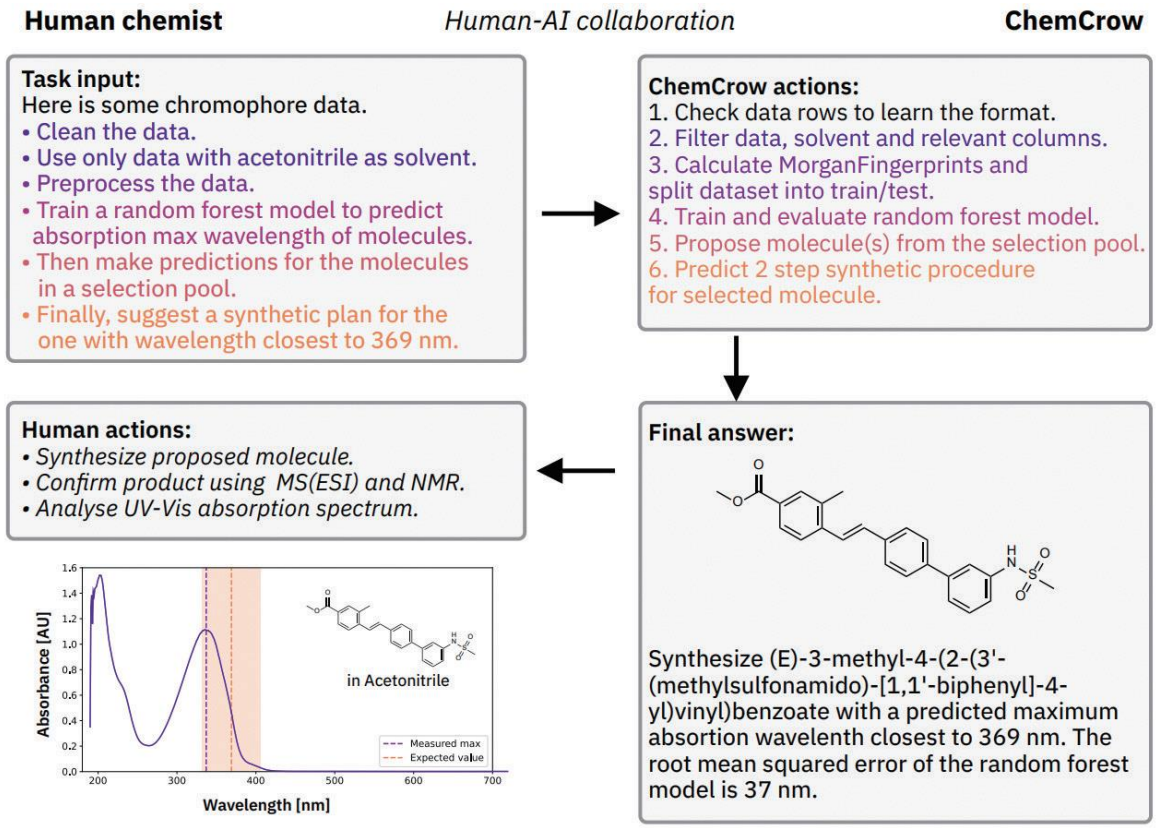


그림 9.17 새로운 분자의 발견으로 이어지는 인간/모델 상호작용 (https://arxiv.org/pdf/2304.05376)

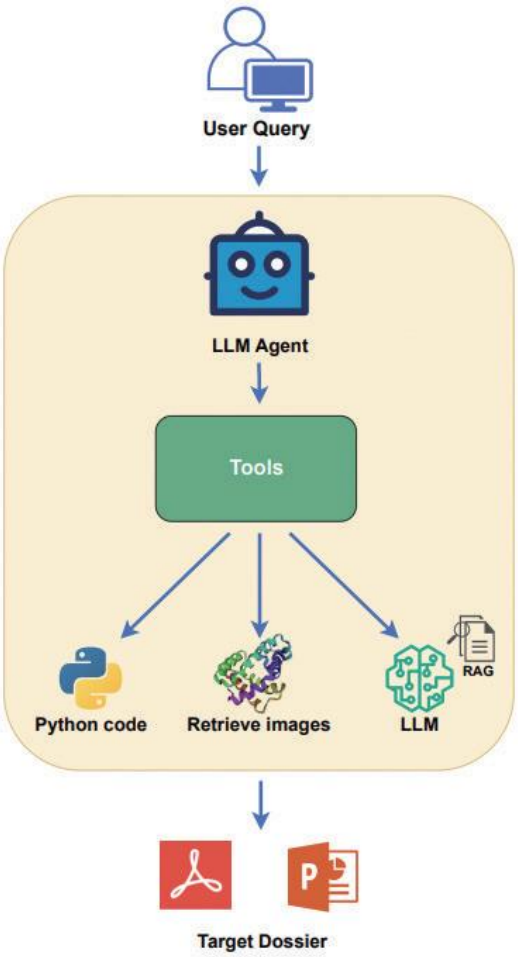


그림 9.18 SwiftDossier 아키텍처(https://arxiv.org/pdf/2409.15817)

Problem (Hydrogen Atom Energy Levels)

An electron in a hydrogen atom transitions from an excited state with the principal quantum number n = 4 to the ground state (n = 1).
Calculate: a) The energy of the emitted photon in joules (J), b) The wavelength of the emitted photon in nanometers (nm), c) The frequency of the emitted photon in terahertz (THz).

GPT-4's Output with CoT Prompting

Execution

Step 1: We'll use the Rydberg formula to calculate the energy levels:

```
E_n = -R_H * (1/n^2), where R_H = 2.18 × 10^-18 J
```

Step 2: Calculate the energy difference:

```
ΔE = E_1 - E_4
   = -R_H * (1/1^2 - 1/4^2)
   = -2.18 × 10^-18 * (1 - 1/16)
   = -2.18 × 10^-18 * (15/16)
   = -2.044 × 10^-18 J
```

Step 3: Calculate wavelength using E = hc/λ:

```
λ = hc / ΔE
  = (6.63 × 10^-34 J·s) * (3.00 × 10^8 m/s) / (2.044 × 10^-18 J)
  = 9.724 × 10^-8 m = 97.24 nm
```

Step 4: Calculate frequency using f = c/λ:

```
f = c / λ
  = (3.00 × 10^8 m/s) / (9.724 × 10^-8 m)
  = 3.085 × 10^15 Hz = 3085 THz
```

Error discovered: The energy difference should be positive as energy is released during the transition.

Final Answer: -2.044 × 10^-18 J, 97.24 nm, 3085 THz

StructChem's Output

Formulae Generation

```
[Formula 1] Rydberg formula: E_n = -R_H * (1/n^2)
[Formula 2] Energy of a photon: E = hc/λ
[Formula 3] Frequency: f = c/λ
```

Step-by-step Reasoning

Step 1: Retrieve constants

```
R_H = 2.198722 × 10^-19 J (Rydberg constant for hydrogen)
h = 6.62607015 × 10^-34 J·s (Planck's constant)
c = 2.99792458 × 10^8 m/s (Speed of light)
```

Step 2: Calculate ΔE using the Rydberg formula

```
ΔE = |E_1 - E_4| = R_H * |1 - 1/16| = R_H * (15/16)
ΔE = 2.179872 × 10^-19 * (15/16) = 2.0436 × 10^-19 J
```

Step 3: Calculate λ using λ = hc/ΔE

```
λ = (6.62607015 × 10^-34 * 2.99792458 × 10^8) / (2.0436 × 10^-19)
λ = 9.7241 × 10^-7 m = 972.41 nm
```

Step 4: Calculate f using f = c/λ

```
f = (2.99792458 × 10^8) / (9.7241 × 10^-7)
f = 3.0831 × 10^14 Hz = 308.31 THz
```

Confidence-based Review-and-Refinement

```
Review 1: All calculations appear correct based on given constants.
Review 2: Results seem consistent with each other.
```

Final Answer: 2.0436 × 10^-19 J, 972.41 nm, 308.31 THz

ChemAgent's Output

Task Decomposition

1. Calculate energy difference
2. Calculate wavelength
3. Calculate frequency

Memory Retrieval

```
- Rydberg formula: E_n = -R_H * (1/n^2)
- Planck's constant: h = 6.62607015 × 10^-34 J·s
- Speed of light: c = 2.99792458 × 10^8 m/s
- Rydberg constant: R_H = 2.179872 × 10^-18 J
```

Execution

Step 1: Calculate ΔE using the Rydberg formula

```
ΔE = |E_1 - E_4|
   = |(-R_H * 1/1^2) - (-R_H * 1/4^2)|
   = R_H * |1 - 1/16|
   = R_H * (15/16)
   = 2.179872 × 10^-18 * (15/16)
   = 2.0436 × 10^-18 J
```

Step 2: Calculate λ using λ = h * c / ΔE

Step 3: Calculate f using f = c / λ

```
λ = (6.62607015 × 10^-34 * 2.99792458 × 10^8) / (2.0436 × 10^-18)
  = 9.7241 × 10^-8 m = 97.241 nm

f = (2.99792458 × 10^8) / (9.7241 × 10^-8)
  = 3.0831 × 10^15 Hz = 3083.1 THz
```

Final Answer: 2.0436 × 10^-18 J, 97.24 nm, 3083 THz

그림 9.19 화학 분야에서 LLM의 실패 사례(https://arxiv.org/pdf/2501.06590)

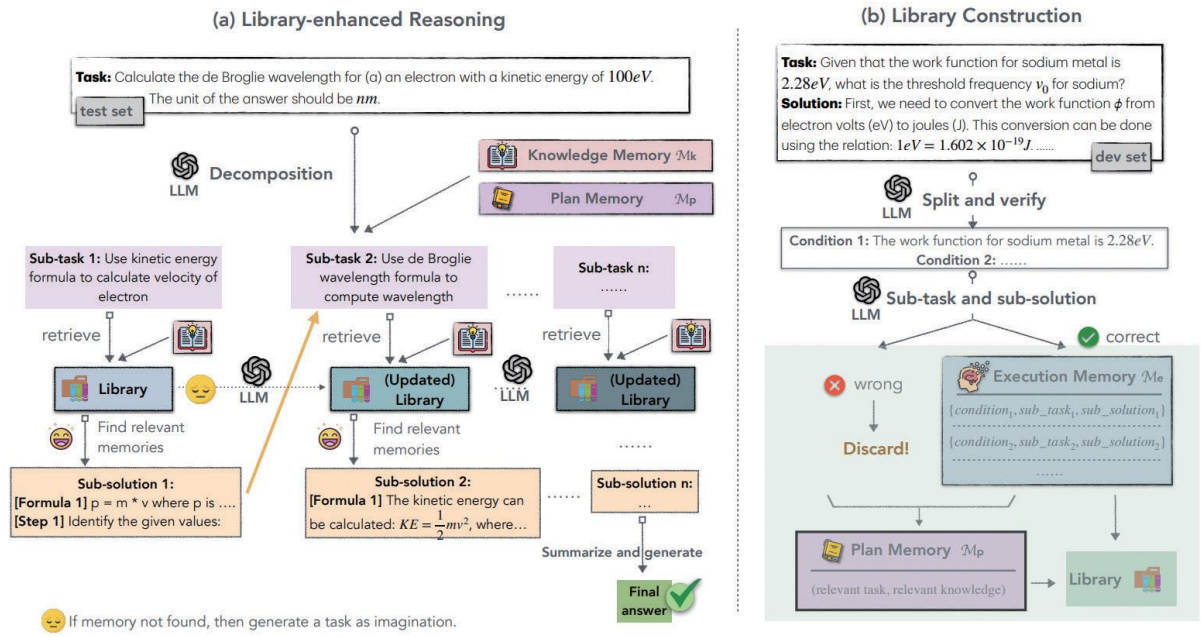


그림 9.20 켐에이전트 프레임워크(https://arxiv.org/pdf/2501.06590)

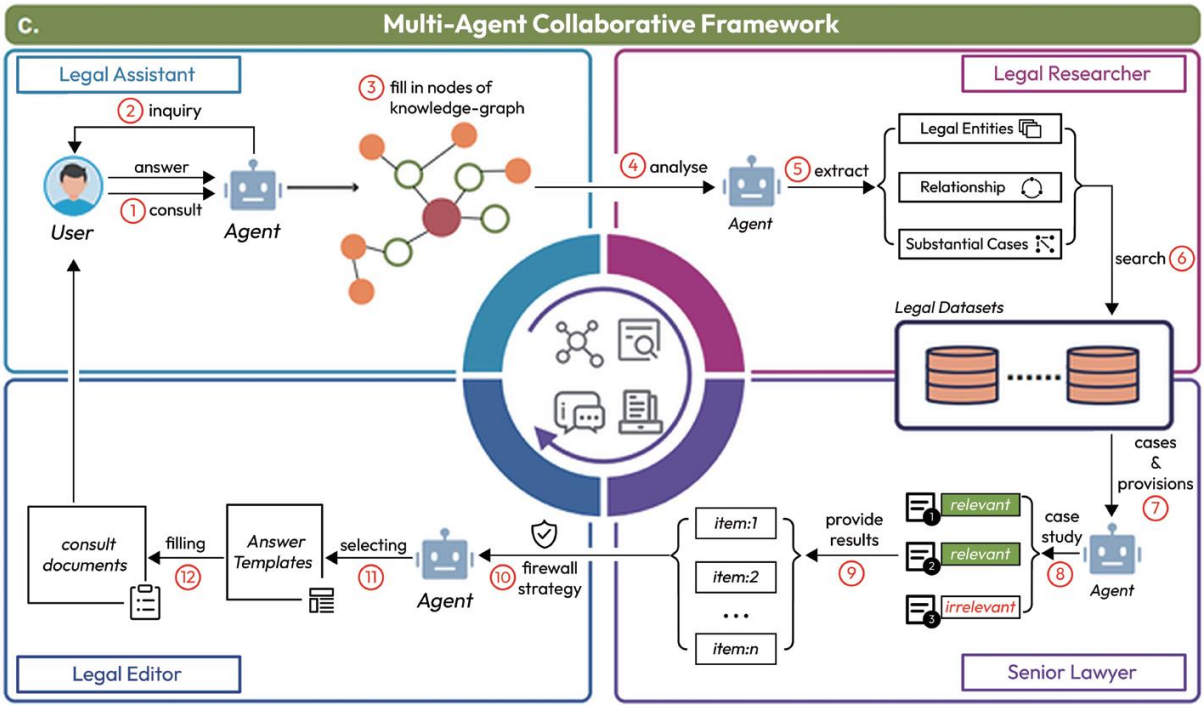


그림 9.21 Chatlaw, 다중 에이전트 협업 구조(https://arxiv.org/pdf/2306.16092v2)

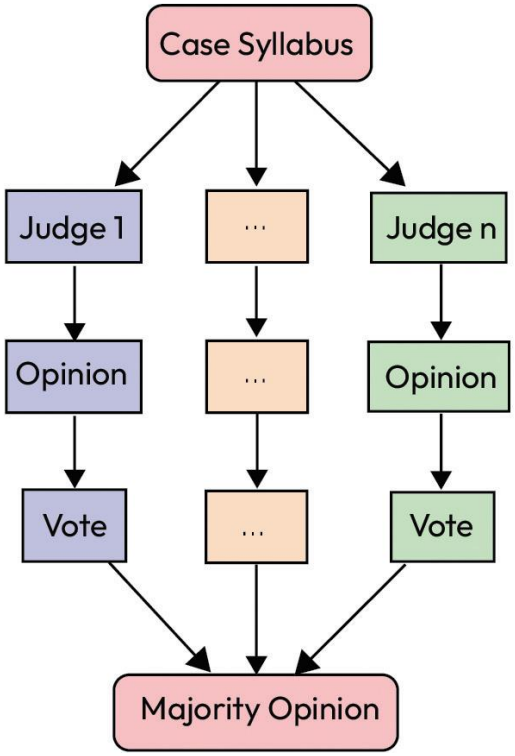


그림 9.22 다중 판사 시스템(https://arxiv.org/pdf/2301.05327)

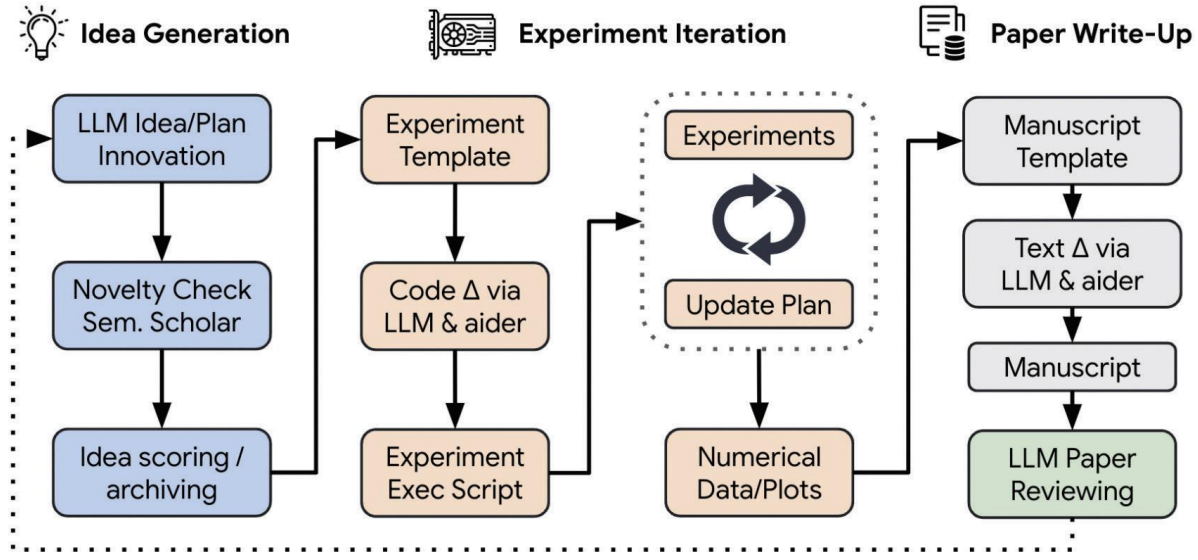


그림 9.23 AI 과학자 프로세스 개요(https://arxiv.org/pdf/2408.06292)

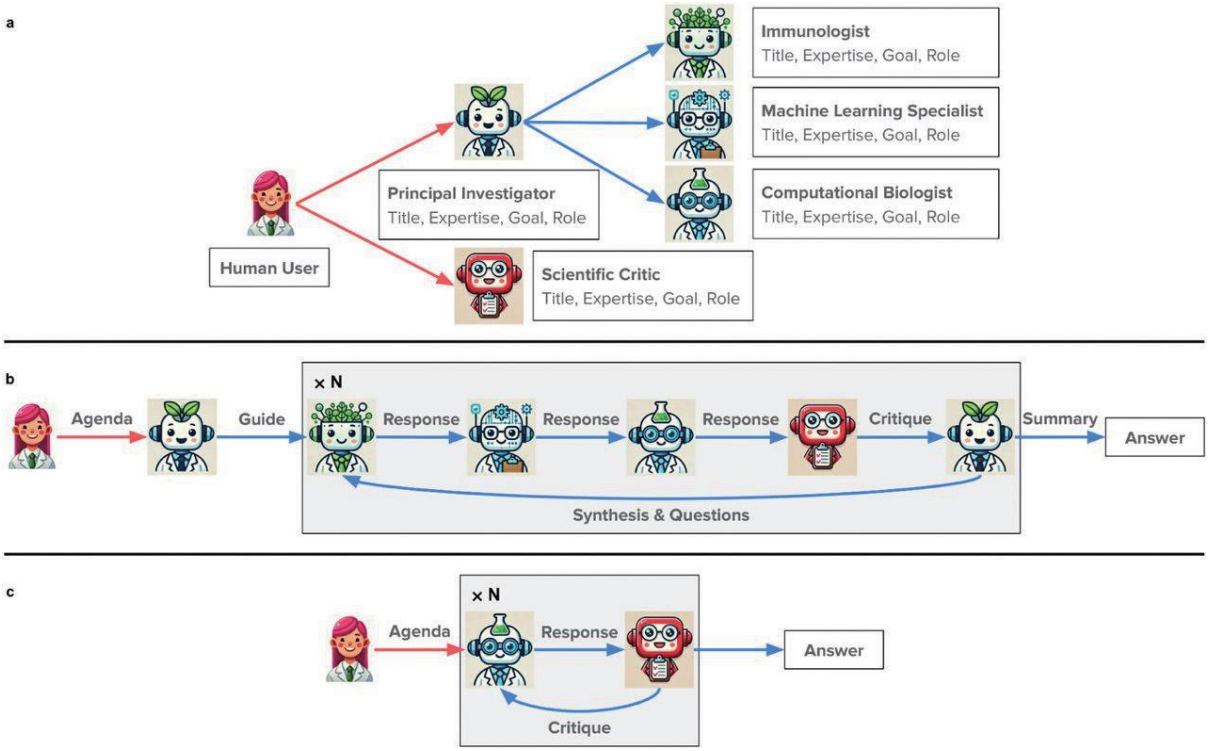


그림 9.24 가상 연구실 아키텍처
(https://www.biorxiv.org/content/10.1101/2024.11.11.623004v1.full)

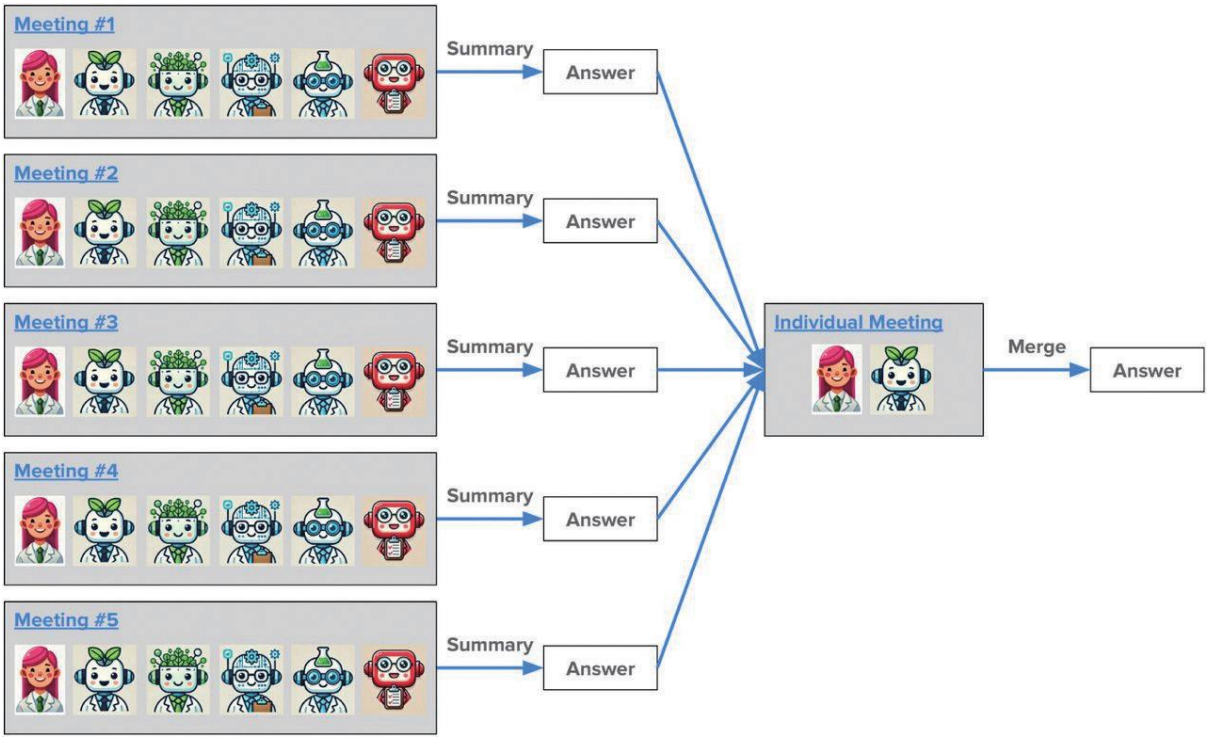


그림 9.25 가상 연구실 병렬 회의
(https://www.biorxiv.org/content/10.1101/2024.11.11.623004v1.full)

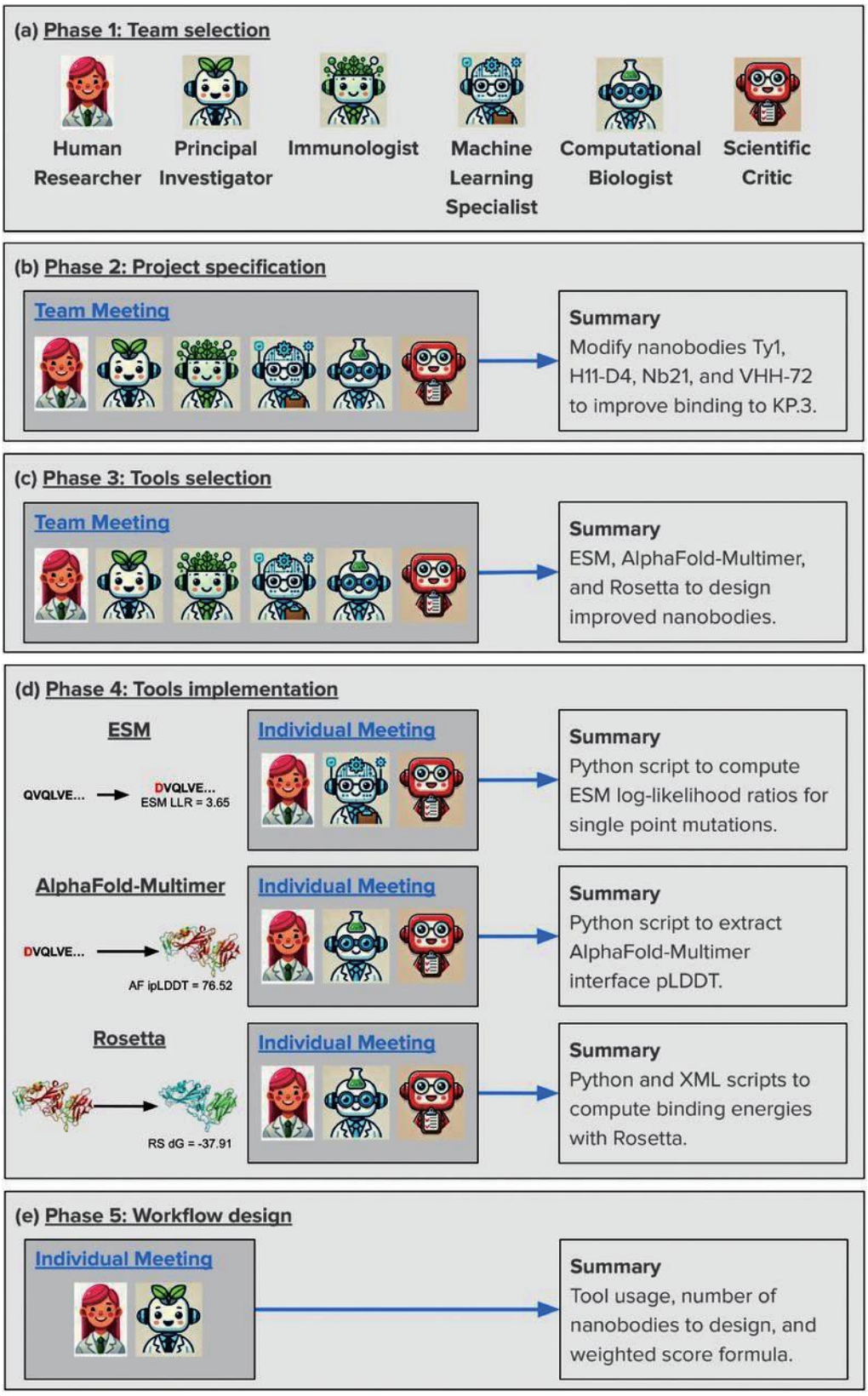


그림 9.26 가상 연구실을 활용한 항체 설계
(https://www.biorxiv.org/content/10.1101/2024.11.11.623004v1.full)

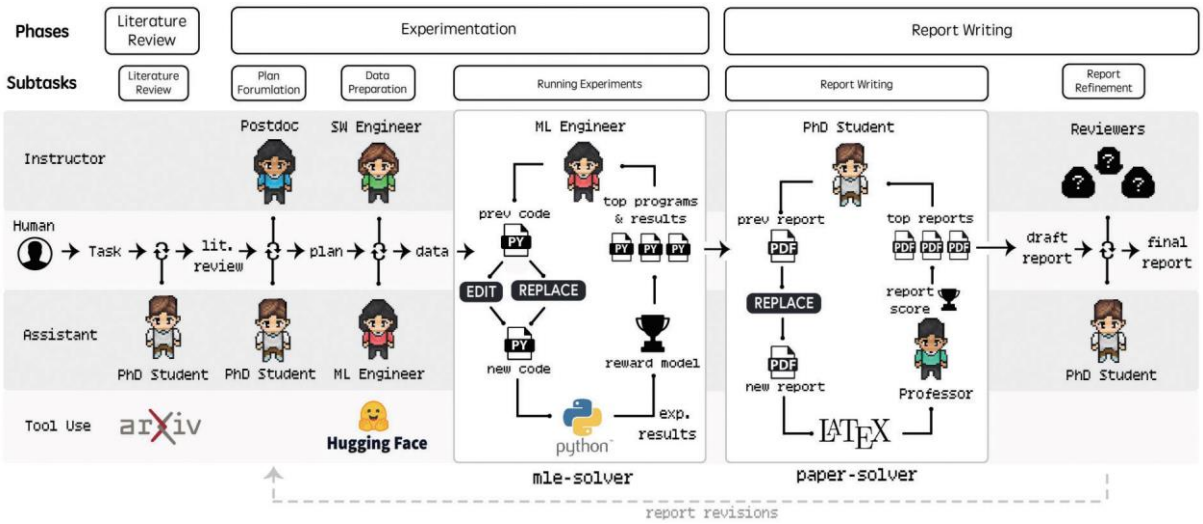


그림 9.27 에이전트 연구실 워크플로(https://arxiv.org/pdf/2501.04227)

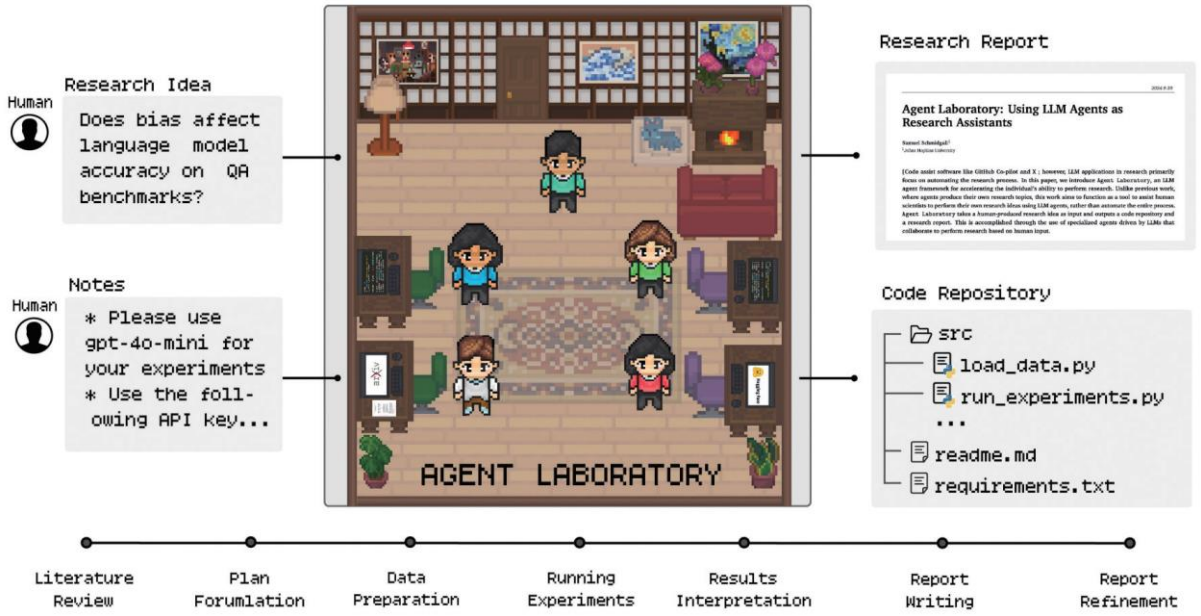


그림 9.28 에이전트 연구실 개요(https://arxiv.org/pdf/2501.04227)

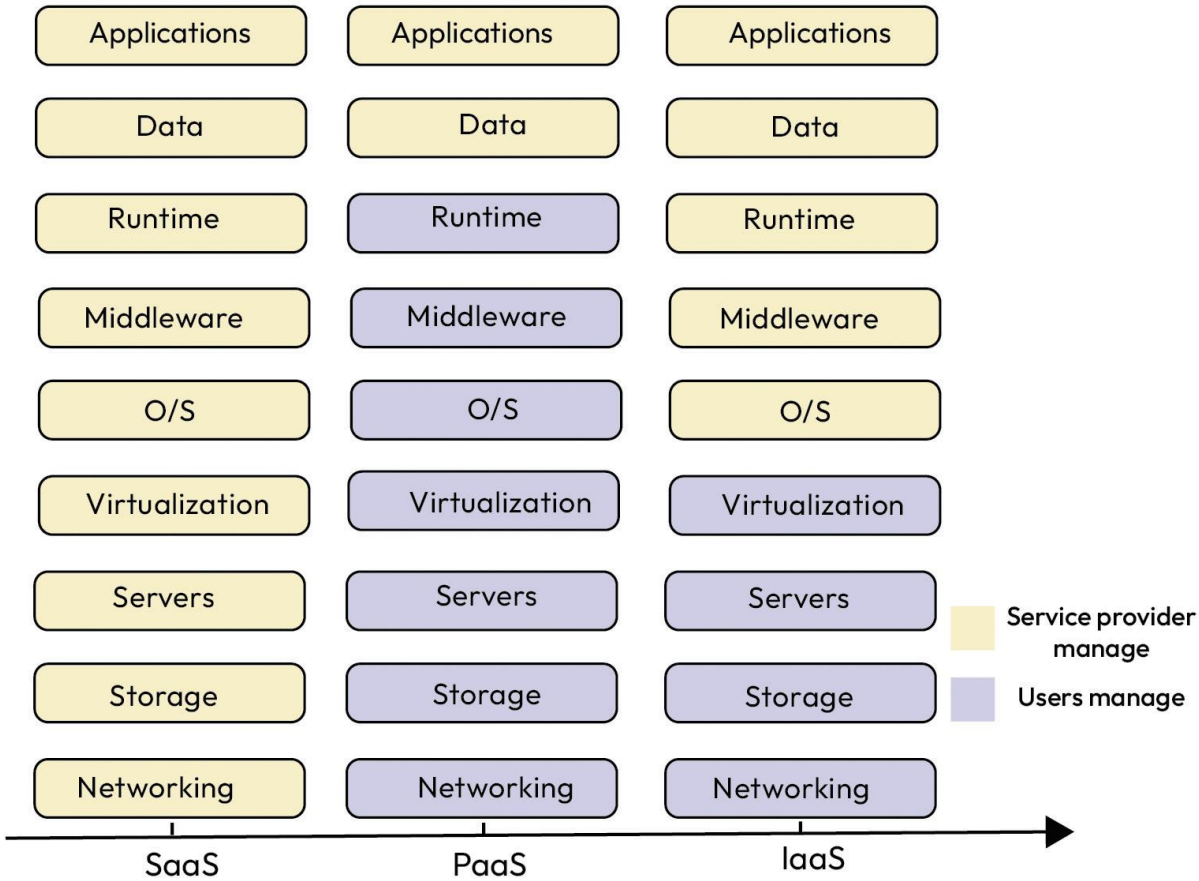
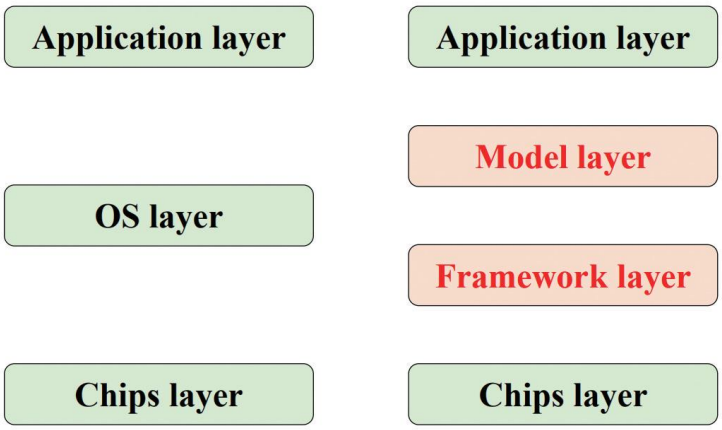


그림 9.39 다양한 패러다임 비교(https://arxiv.org/pdf/2311.05804)



(a) Traditional application development technology stack

(b) New application development technology stack

그림 9.40 전통적 기술 스택과 모델 기반 기술 스택 비교(https://arxiv.org/pdf/2311.05804)

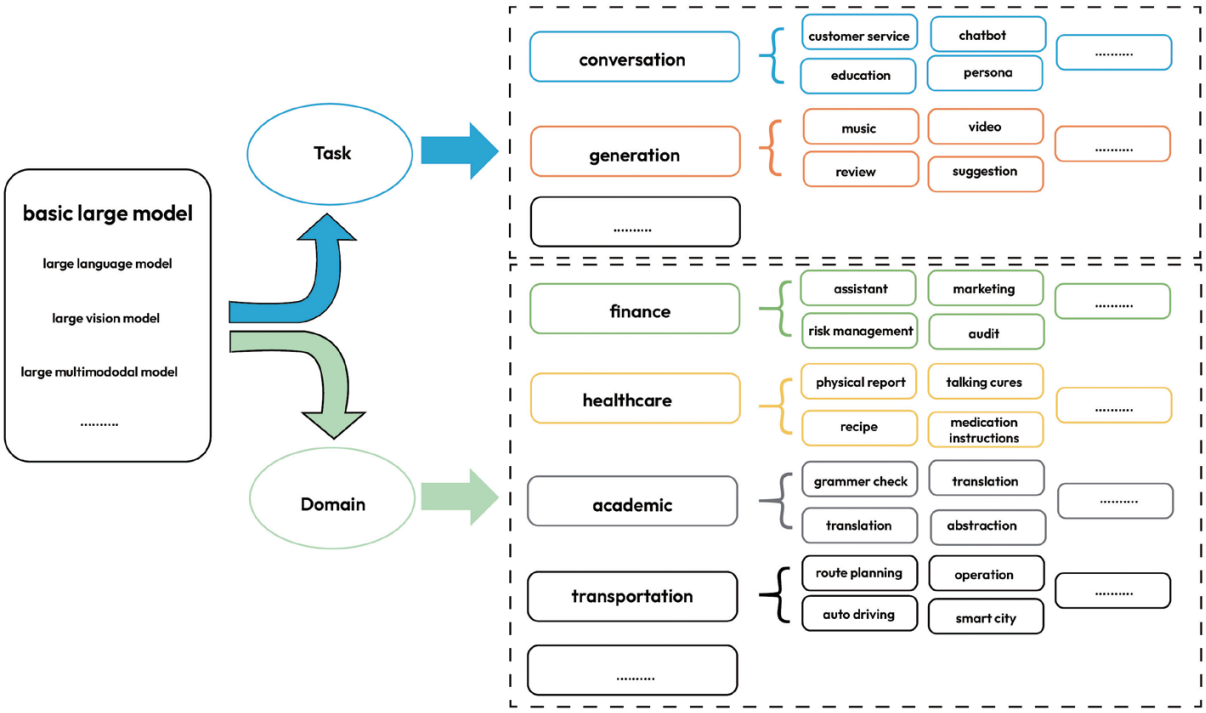


그림 9.41 MaaS 내 다양한 산업 분야의 적용 사례(https://arxiv.org/pdf/2311.05804)

10장 AI 에이전트 애플리케이션 구축하기

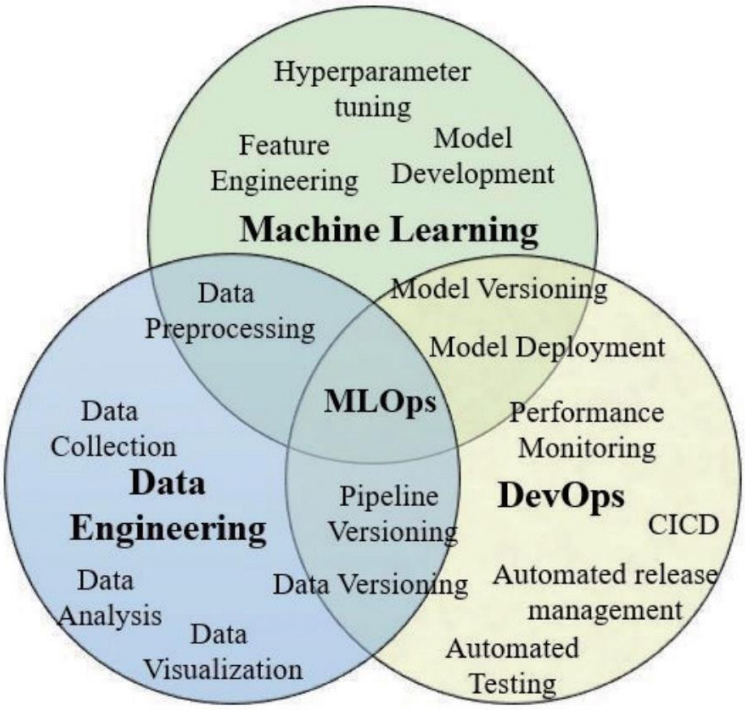


그림 10.30 MLOps의 결합 요소(https://arxiv.org/pdf/2202.10169)

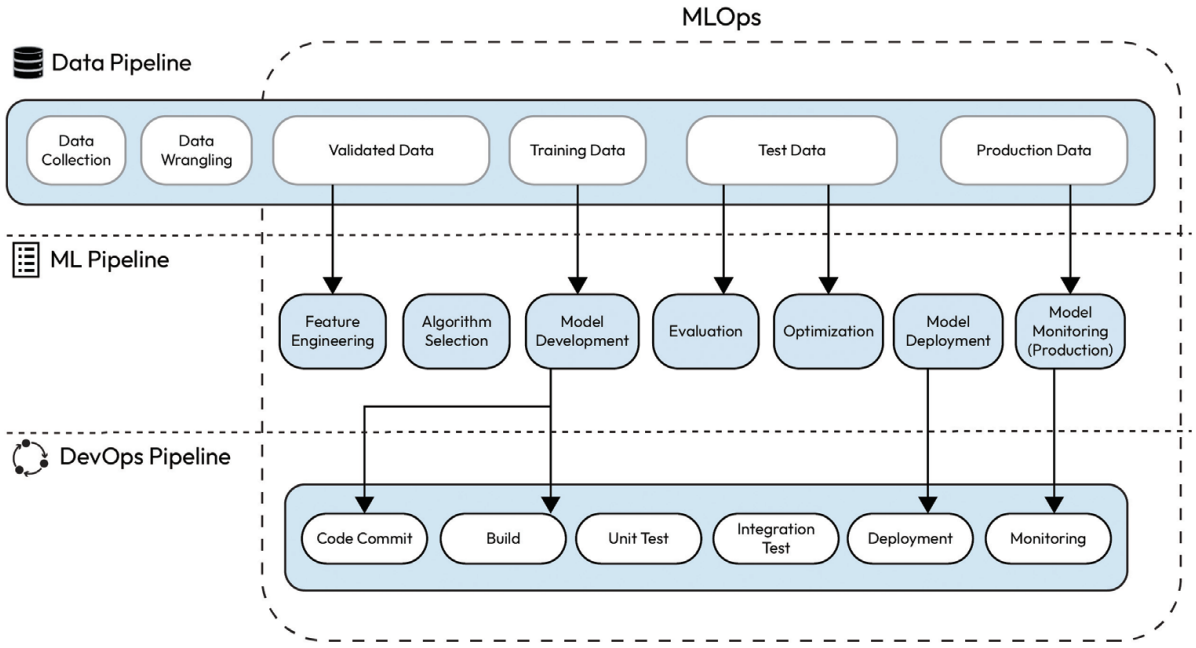


그림 10.31 MLOps의 상위 수준 프로세스(https://arxiv.org/pdf/2202.10169)

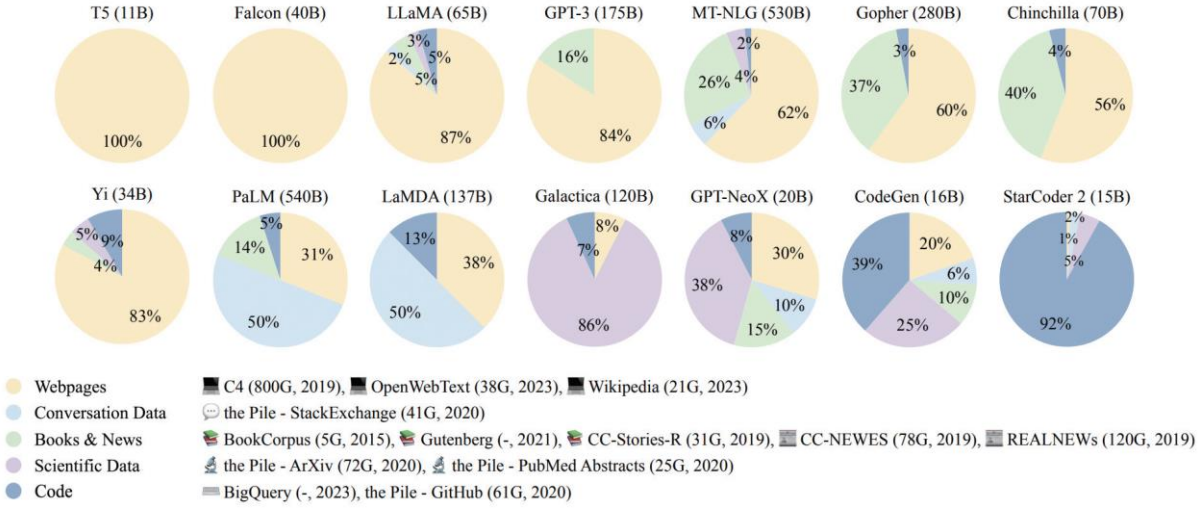


그림 10.32 기존 LLM의 사전 학습 데이터에 사용된 다양한 데이터 소스의 비율 (https://arxiv.org/pdf/2303.18223)

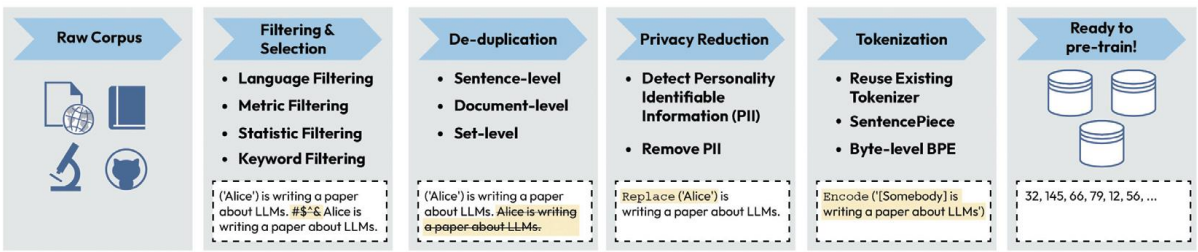


그림 10.33 LLM 사전 학습을 위한 전형적인 데이터 전처리 파이프라인 (https://arxiv.org/pdf/2303.18223)

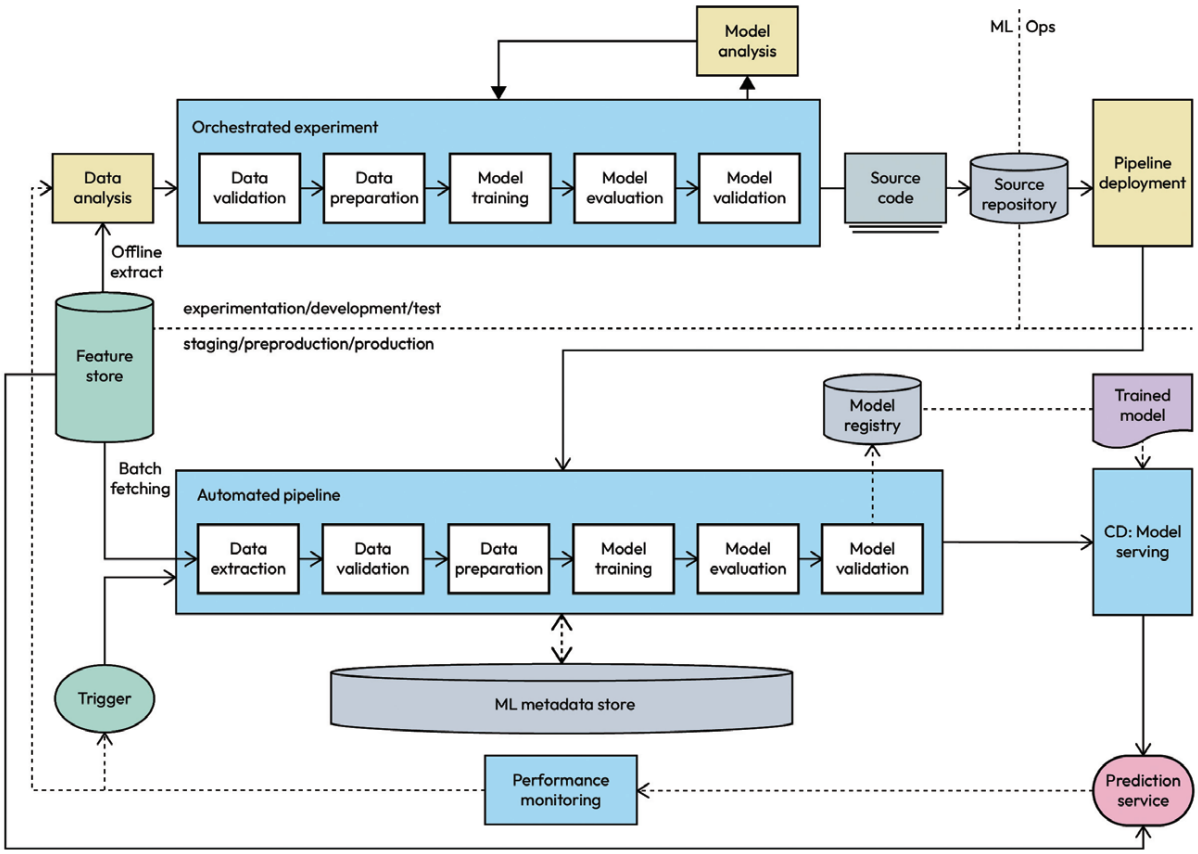


그림 10.34 지속 학습을 위한 머신러닝 파이프라인 자동화 (https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning)

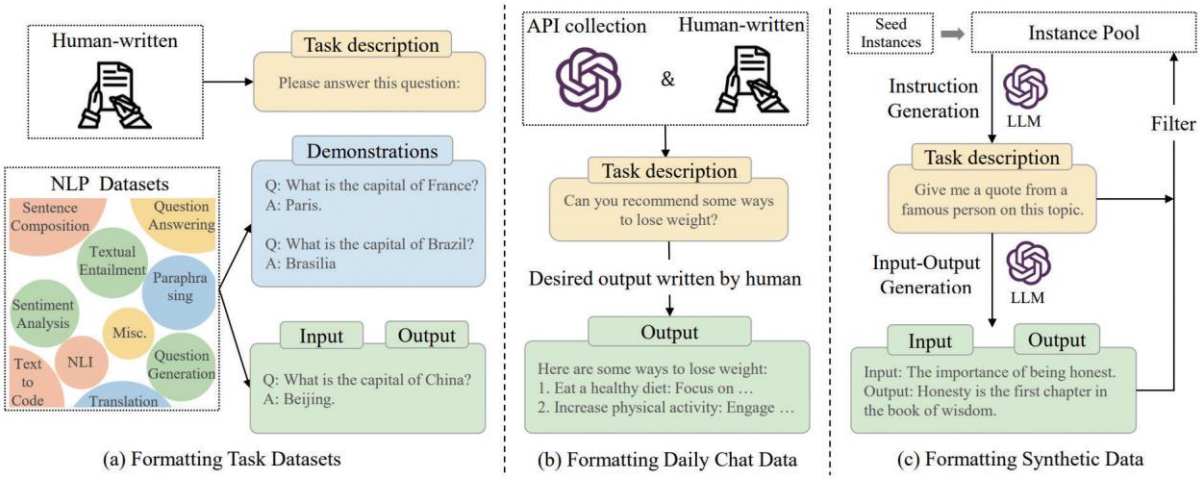


그림 10.35 지시 튜닝 데이터셋 구성(https://arxiv.org/pdf/2303.18223)

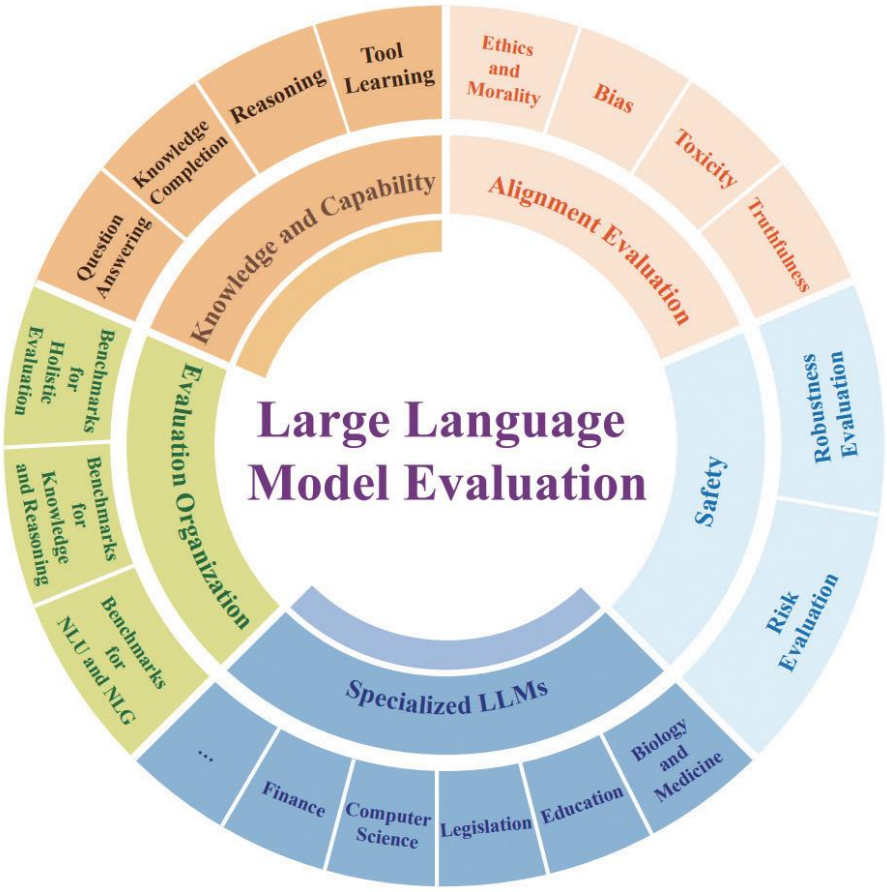


그림 10.36 LLM 평가의 분류(https://arxiv.org/pdf/2310.19736)

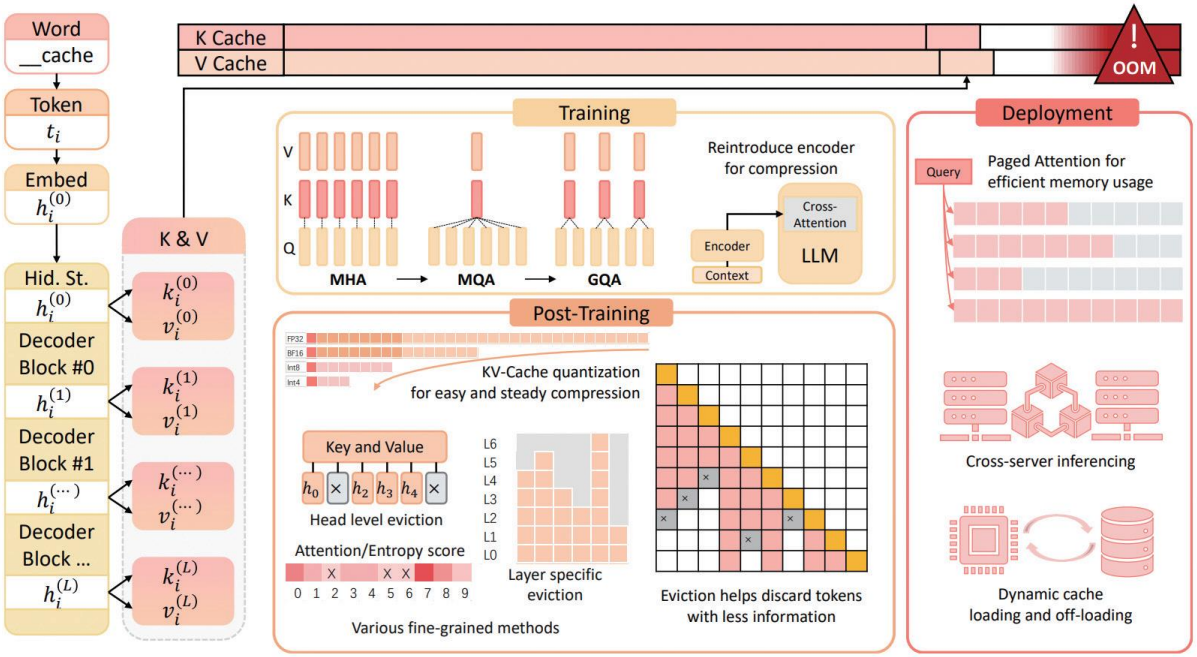


그림 10.39 추론 속도를 높이는 방법의 개요(https://arxiv.org/pdf/2407.18003)

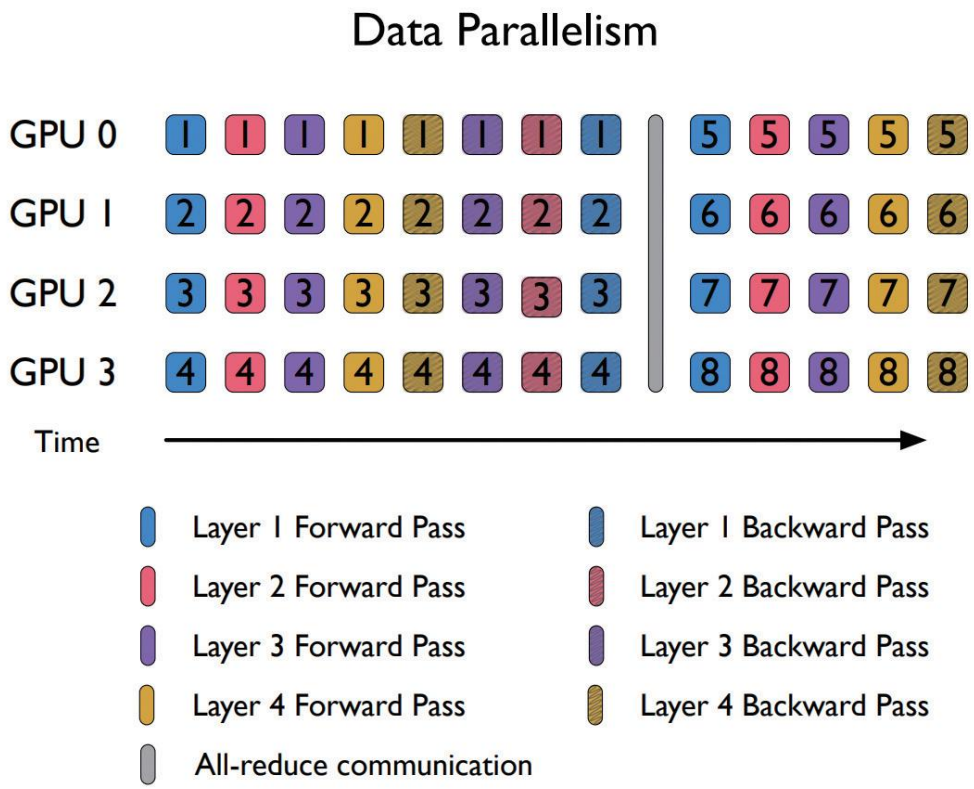


그림 10.40 데이터 병렬화에서 시간에 따른 미니배치 처리. 각 GPU는 모든 레이어(서로 다른 색상으로 표시)의 복사본을 가지고,서로 다른 미니배치(번호 표시)를 다른 GPU가 처리함(https://arxiv.org/pdf/2111.04949)

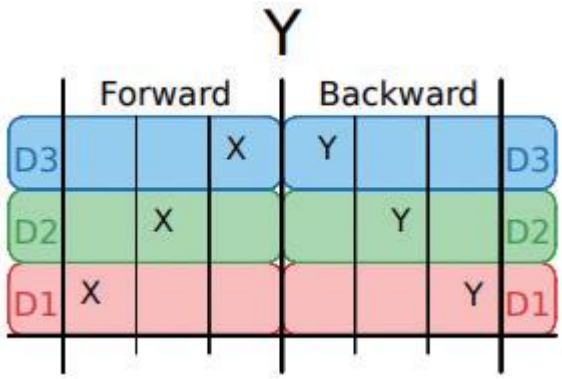


그림 10.41 단일 마이크로 배치에 대한 순전파와 역전파 업데이트 (https://arxiv.org/pdf/2403.03699v1)

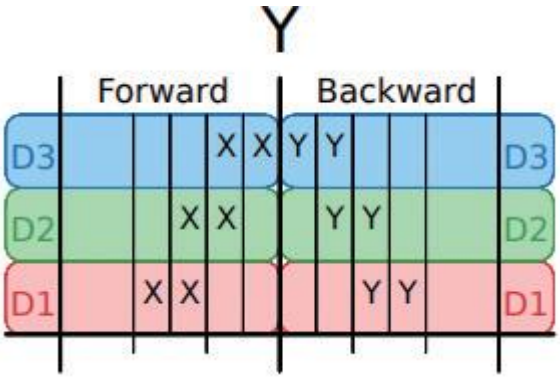


그림 10.42 두 개의 마이크로 배치를 병렬로 처리하는 순전파와 역전파 업데이트 (https://arxiv.org/pdf/2403.03699v1)

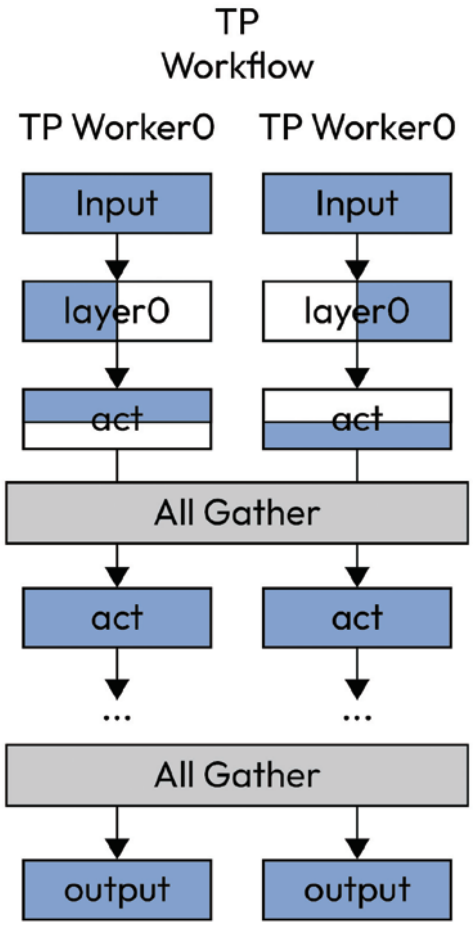


그림 10.43 텐서 병렬화(https://arxiv.org/pdf/2311.01635)

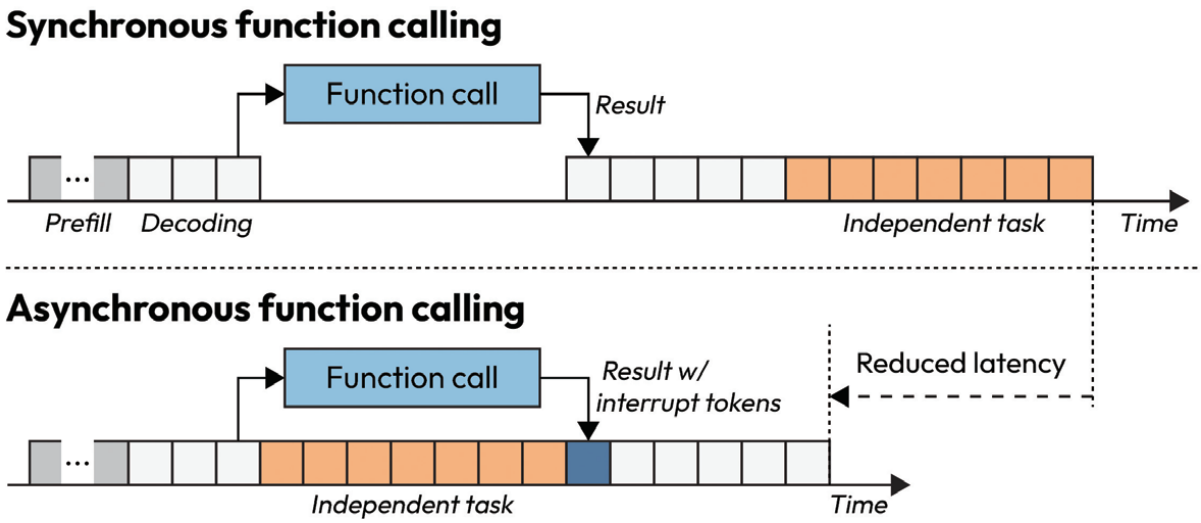


그림 10.50 동기 vs. 비동기 함수 호출(https://arxiv.org/pdf/2412.07017)

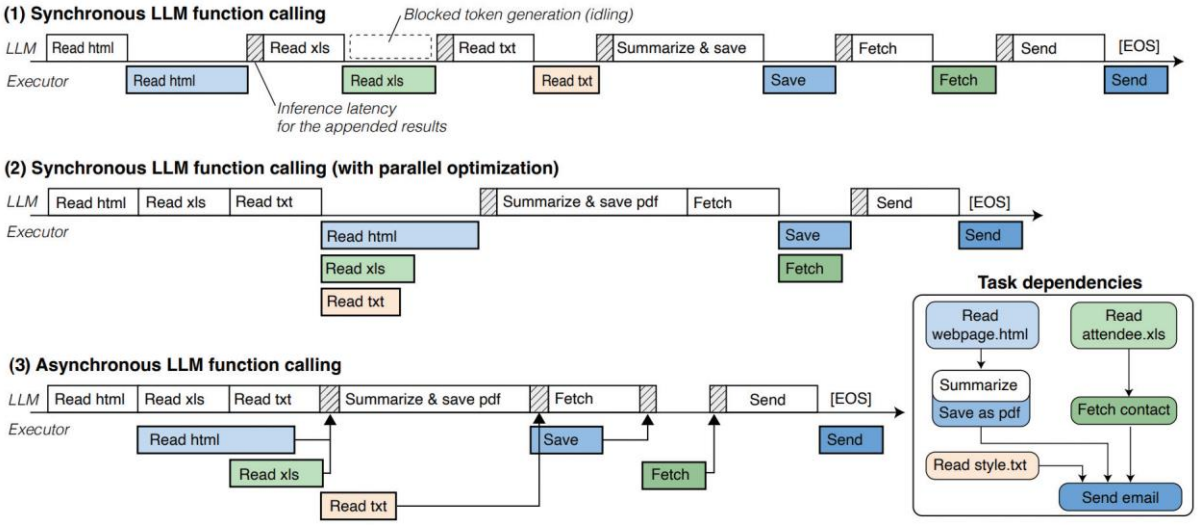


그림 10.51 LLM 실행기의 상호작용 비교(https://arxiv.org/pdf/2412.07017)

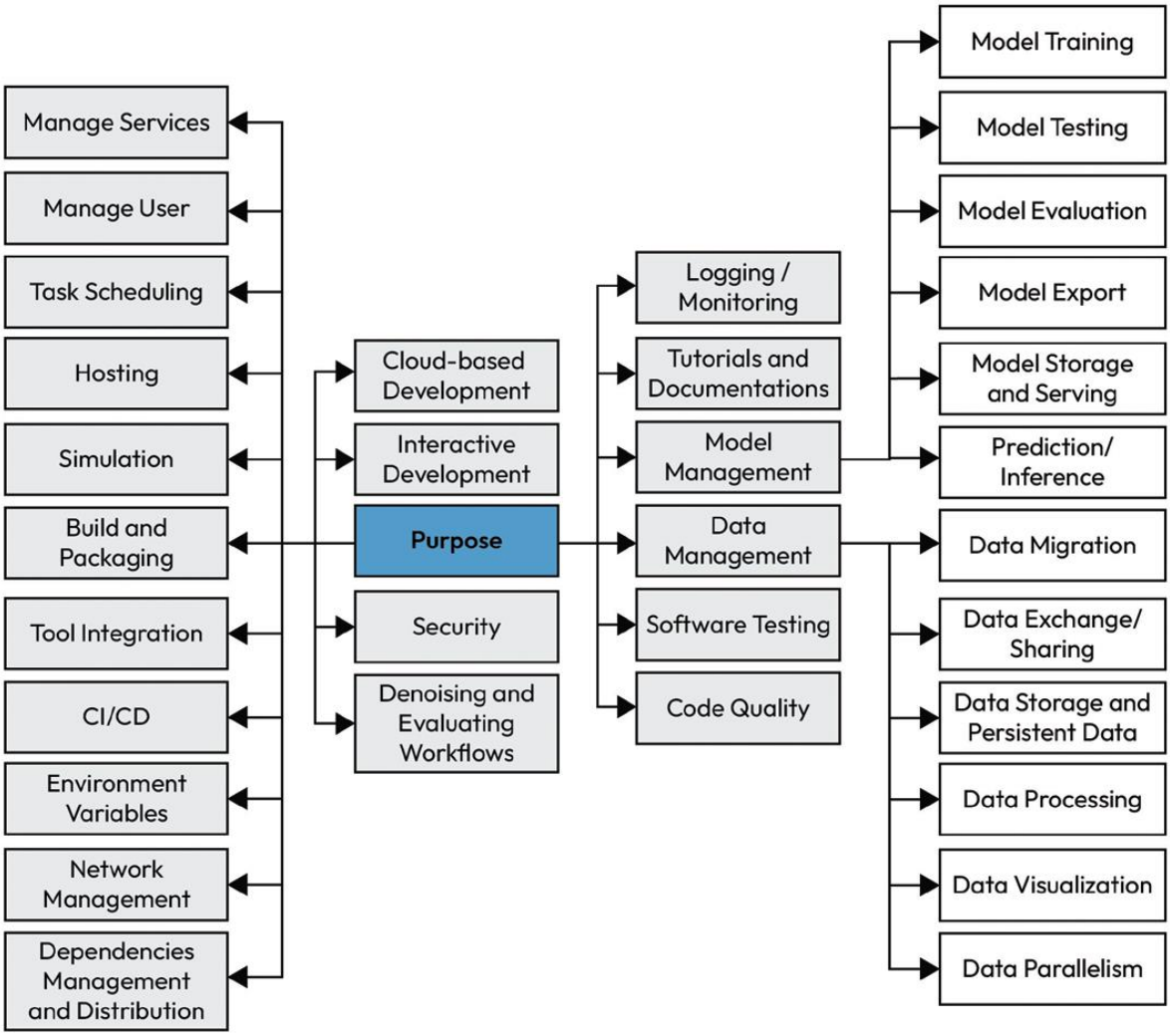


그림 10.52 머신러닝 기반 소프트웨어 프로젝트에서 도커를 사용하는 목적 개요 (https://arxiv.org/pdf/2206.00699)

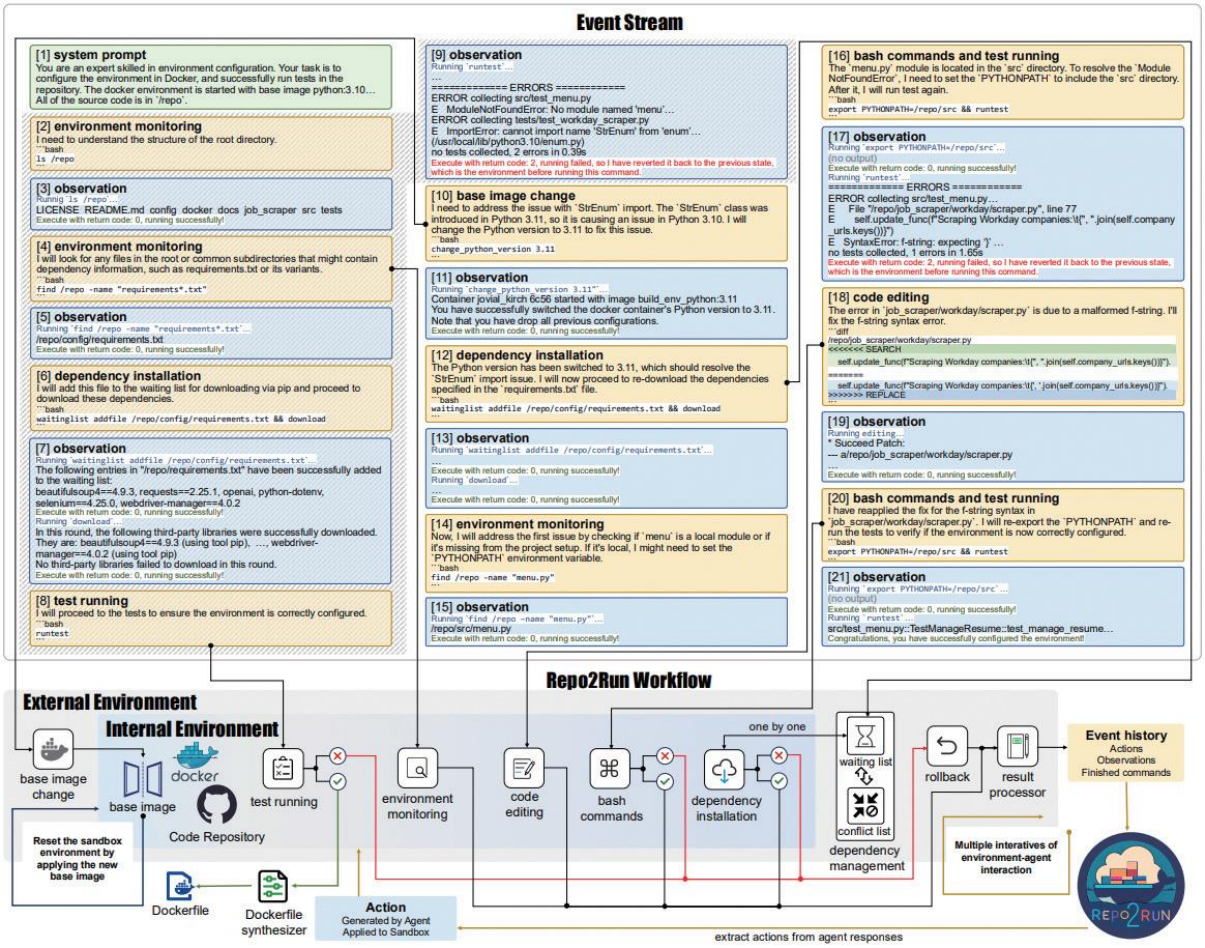


그림 10.53 Repo2Run의 예시 프로세스(https://www.arxiv.org/pdf/2502.13681)

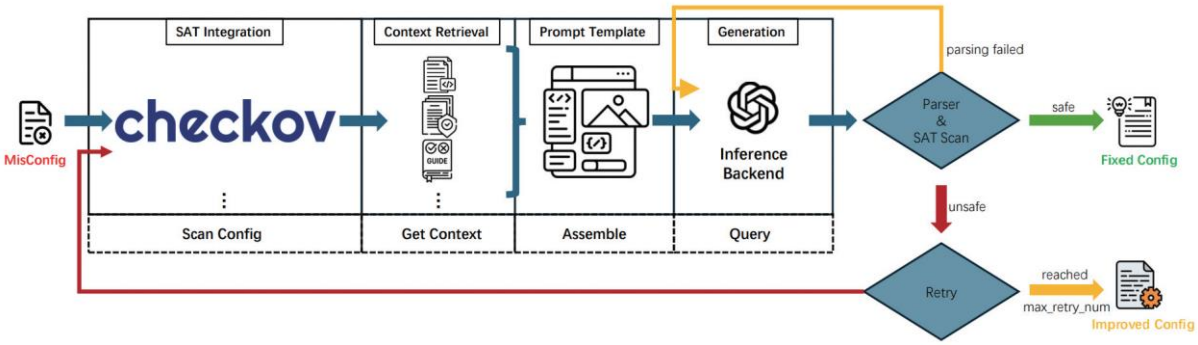


그림 10.54 쿠버네티스 보안 설정 자동화를 위한 LLMSecConfig 프레임워크 아키텍처 개요(https://arxiv.org/pdf/2502.02009)

11장 다가올 미래

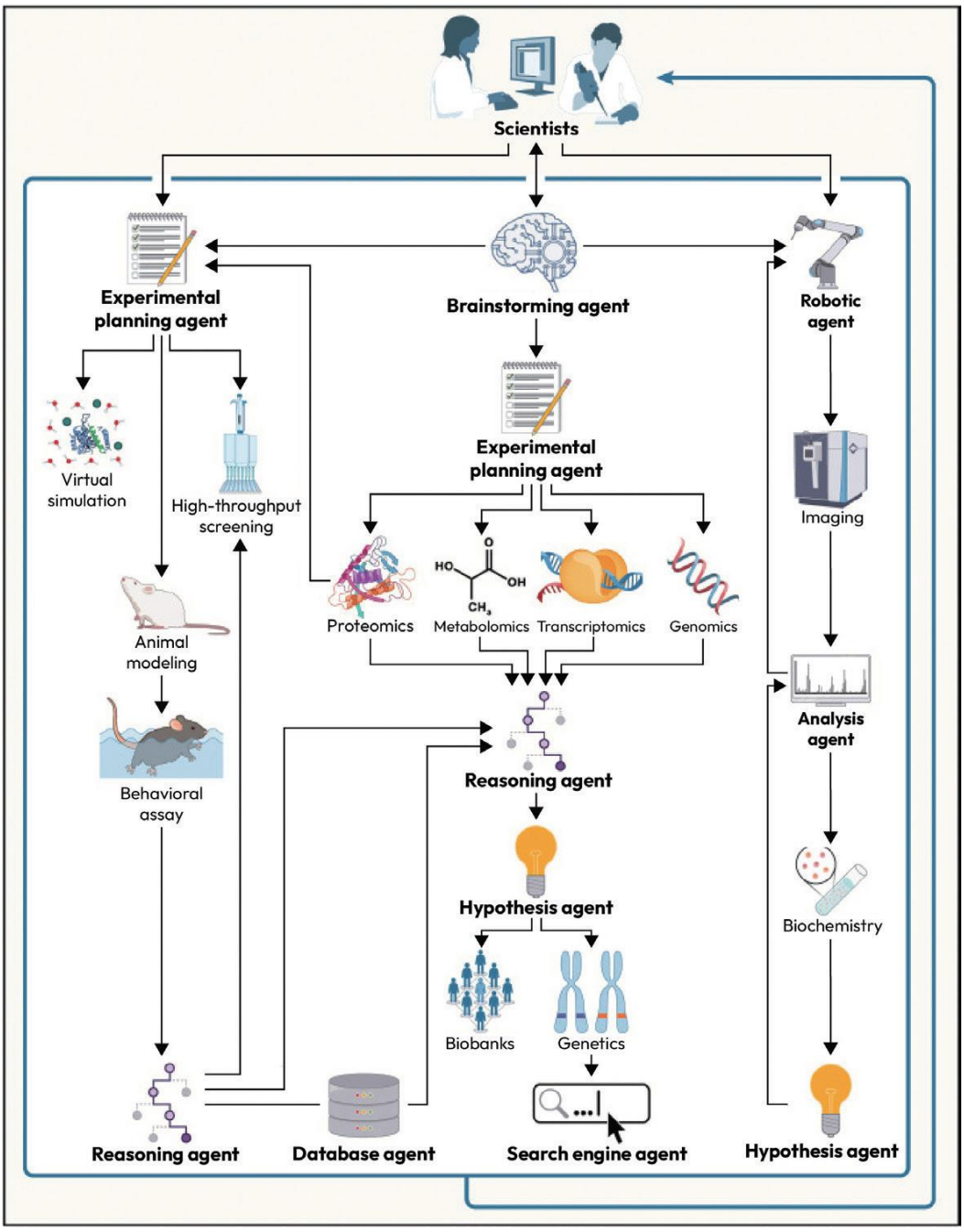


그림 11.1 AI 에이전트로 강화되는 생물의학 연구(https://www.cell.com/cell/fulltext/S0092-8674(24)01070-5)

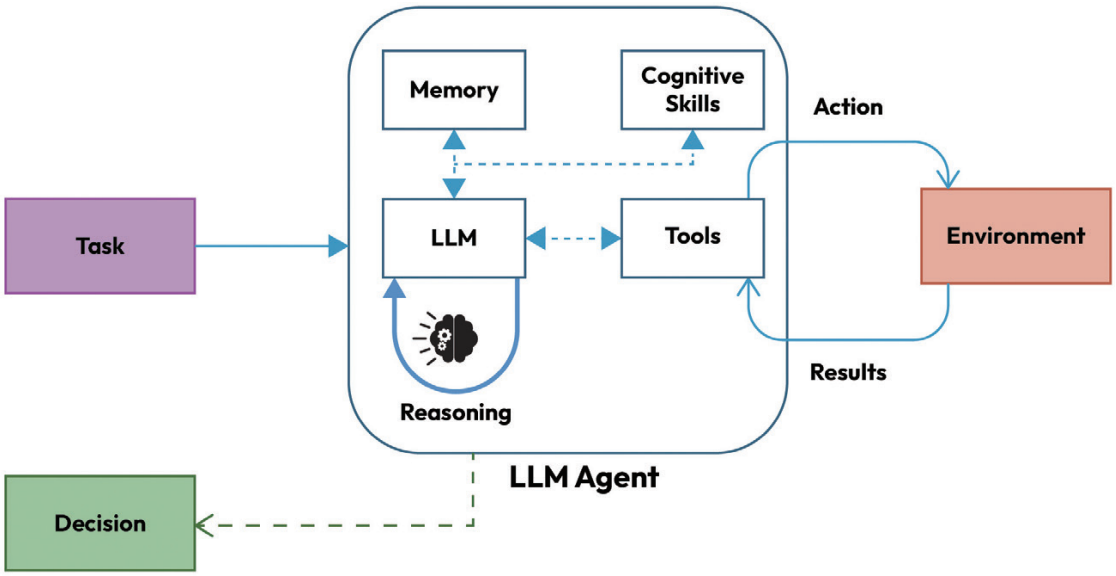


그림 11.2 LLM 기반 에이전트(https://arxiv.org/pdf/2501.08944v1)

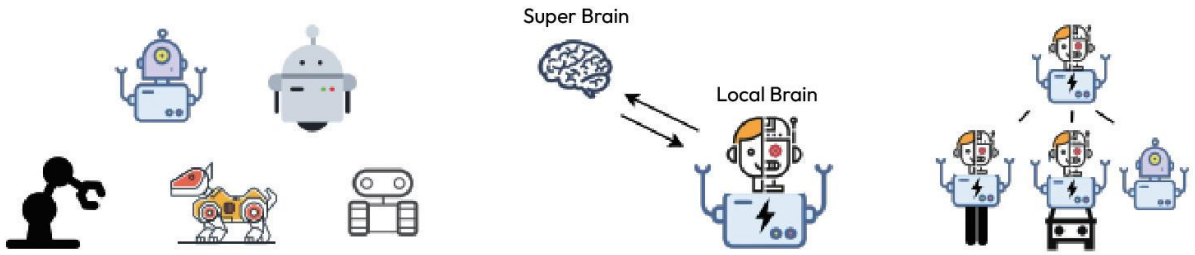
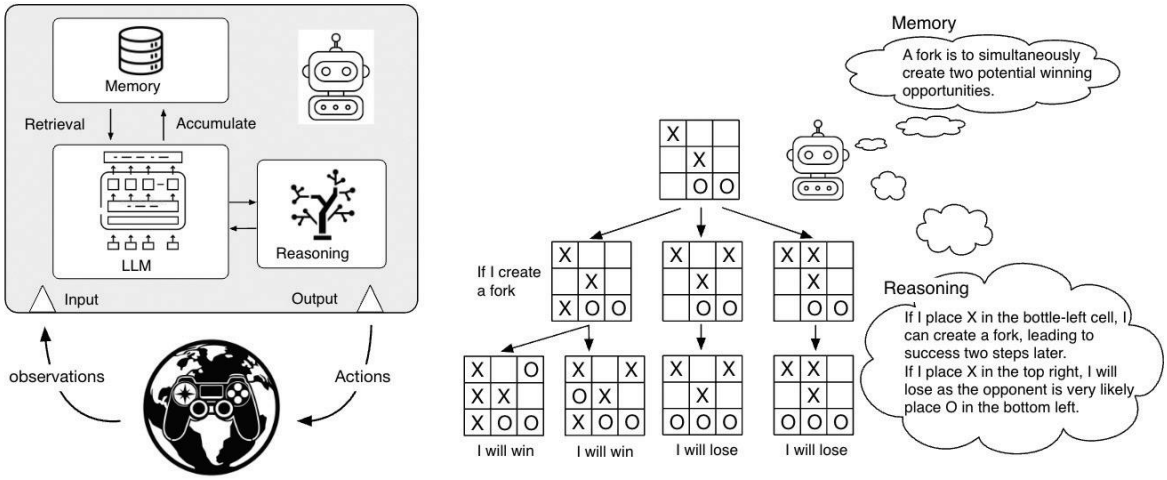
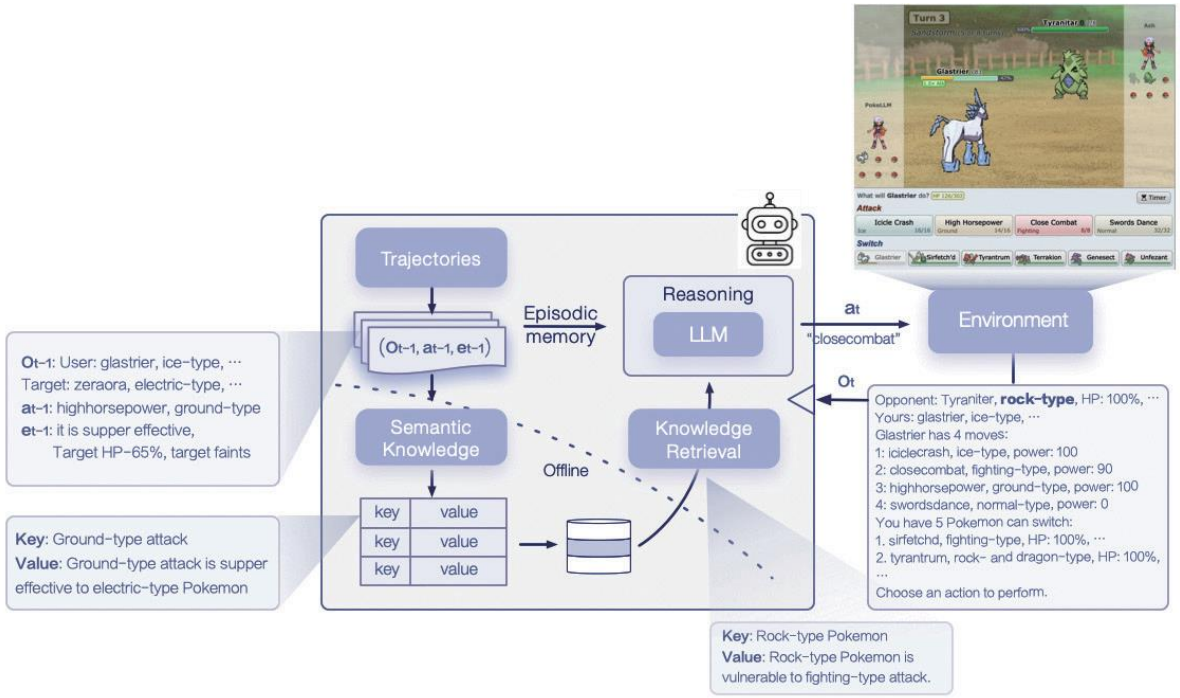


그림 11.3 체화된 지능(embodied intelligence)의 과제(https://arxiv.org/pdf/2311.07226)



(a) LLM-based game agent framework (b) An example of playing Tic-Tac-Toe

그림 11.4 LLM 기반 게임을 위한 전체 프레임워크(https://arxiv.org/pdf/2404.02039)



(a) An example of playing Pokémon Battles

그림 11.5 의미적 지식을 활용한 효과적인 전략 수립(https://arxiv.org/pdf/2404.02039)

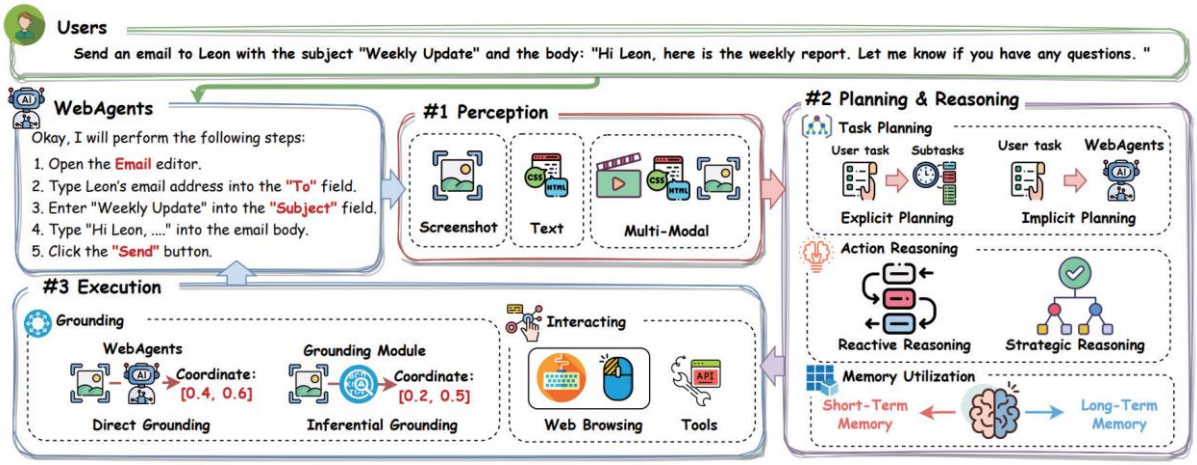


그림 11.6 웹 에이전트 프레임워크(https://arxiv.org/pdf/2503.23350)

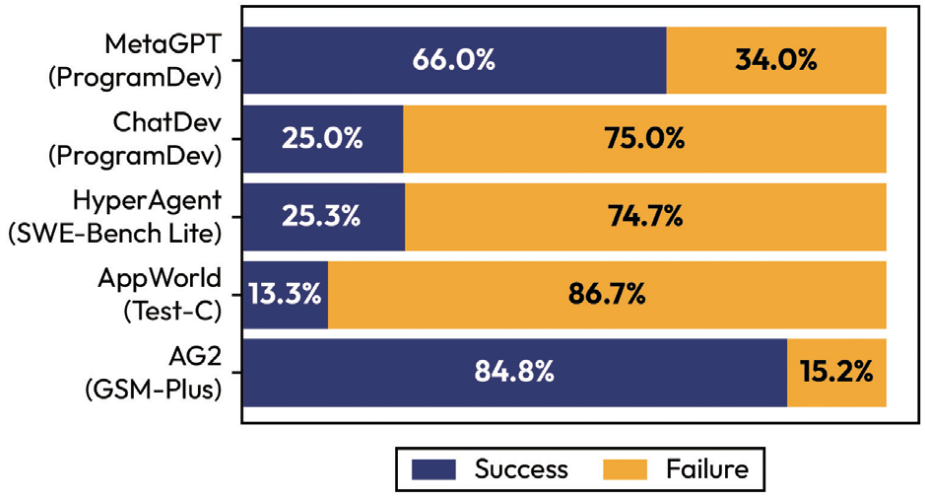


그림 11.7 5개 유명 다중 에이전트 LLM 시스템의 실패율(https://arxiv.org/pdf/2503.13657)

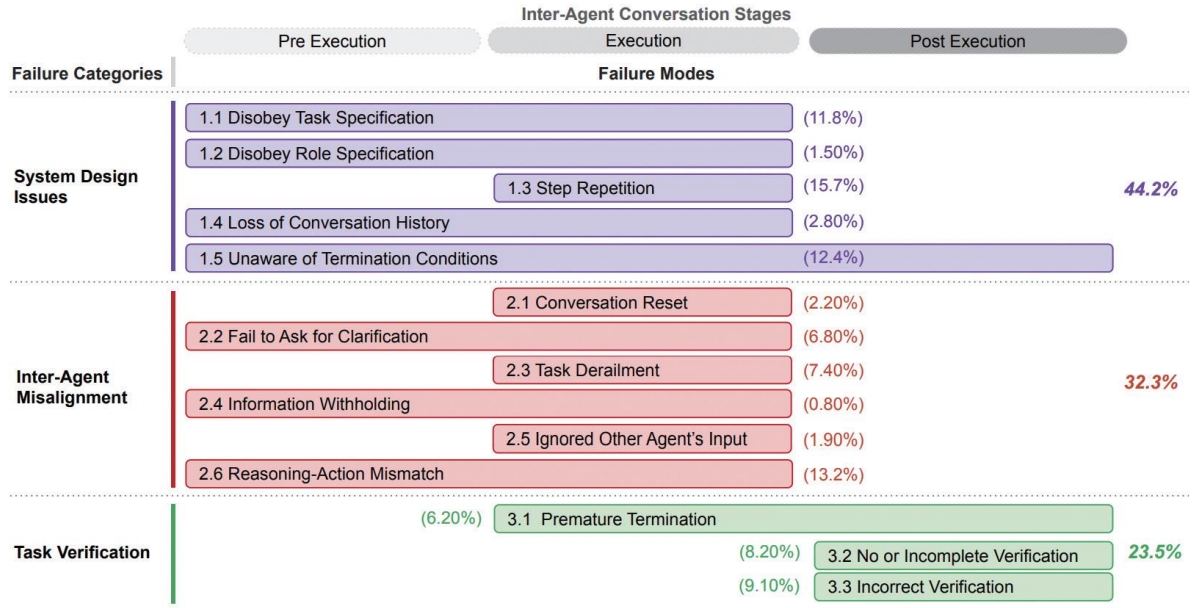


그림 11.8 다중 에이전트 시스템(MAS) 실패 모드 분류(https://arxiv.org/pdf/2503.13657)

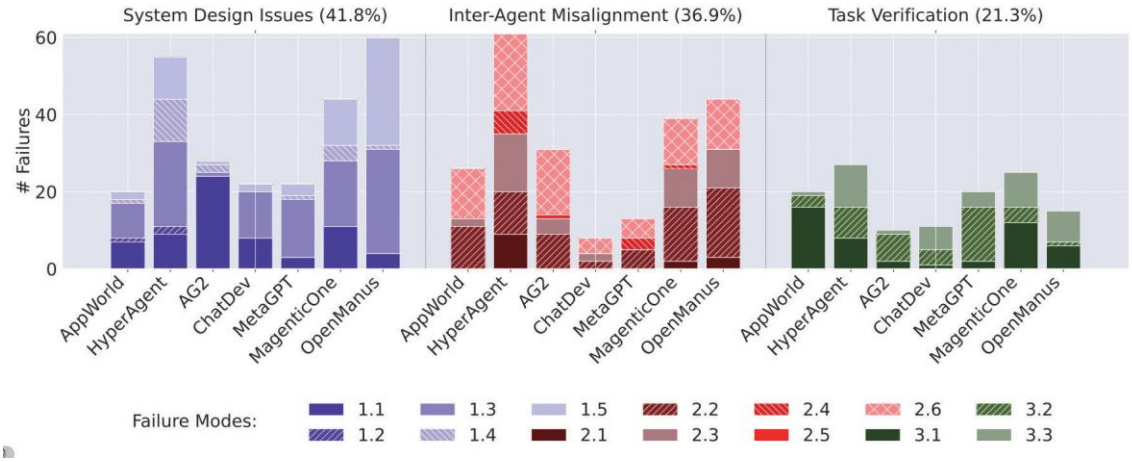


그림 11.9 시스템과 범주별 실패 모드 분포(https://arxiv.org/pdf/2503.13657)

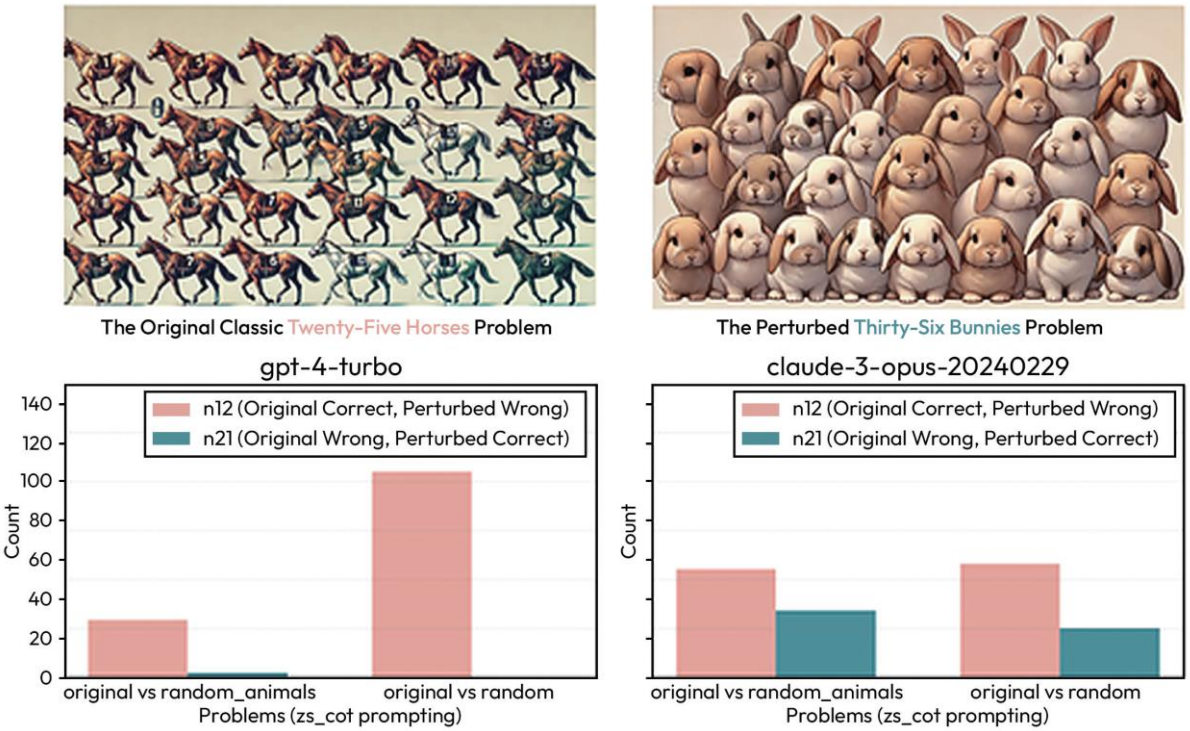


그림 11.10 고전 문제에서의 토큰 편향(https://arxiv.org/pdf/2406.11050)

Original Problem

Jessica is six years older than Claire. In two years, Claire will be 20 years old. How old is Jessica now?

Modified Problem

Jessica is six years older than Claire. In two years, Claire will be 20 years old. *Twenty years ago, the age of Claire's father is 3 times of Jessica's age.* How old is Jessica now?

Standard Answer 24

그림 11.11 관련 없는 컨텍스트가 LLM을 방해하는 사례(https://arxiv.org/pdf/2302.00093)

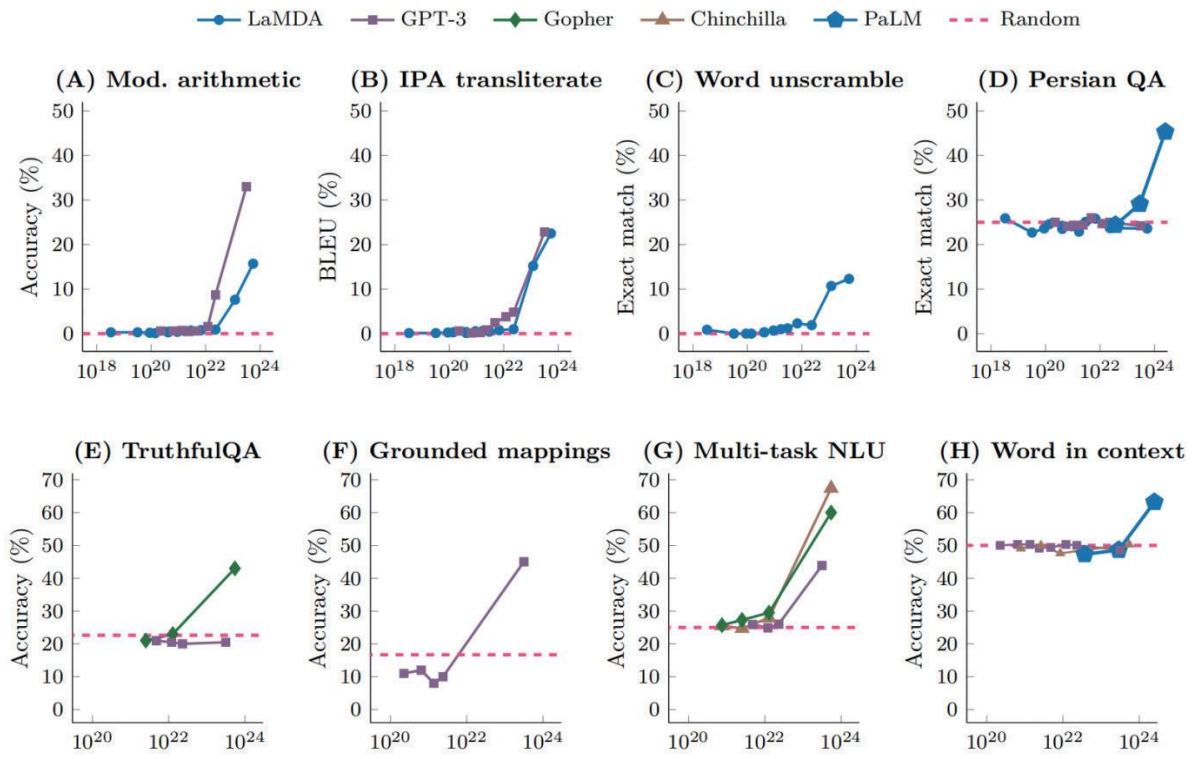


그림 11.12 창발하는 추론 특성의 예시(https://arxiv.org/abs/2304.15004)

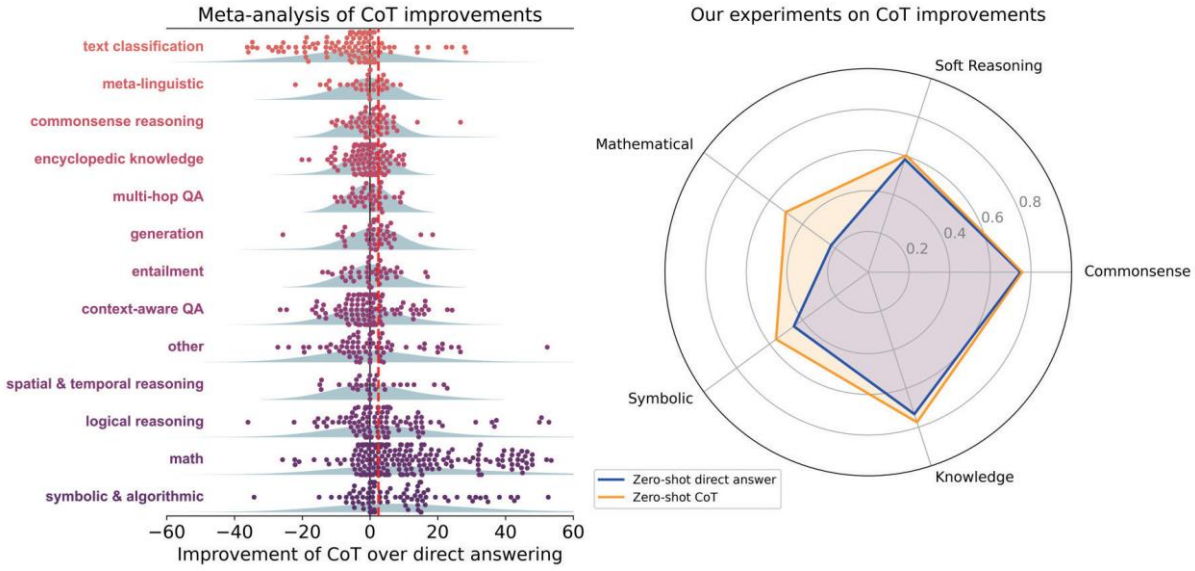


그림 11.13 CoT 개선은 기호적(symbolic) 추론과 수학적 추론에 국한됨 (https://arxiv.org/pdf/2409.12183)

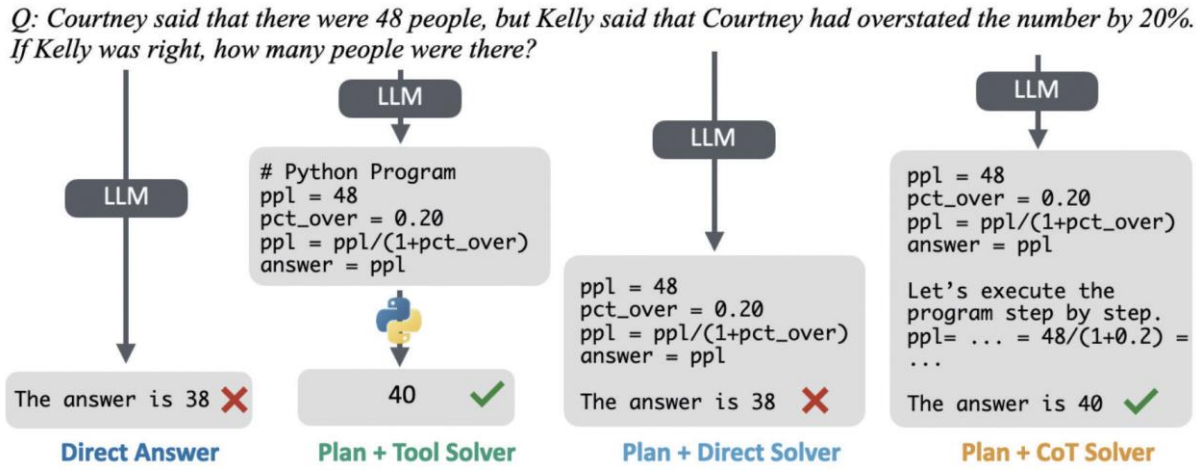


그림 11.14 LLM은 계획을 세울 수 있으나 일부 문제를 더 잘 해결하려면 외부 도구가 필요함(https://arxiv.org/pdf/2409.12183)

GSM8K

When Sophie watches her nephew, she gets out a variety of toys for him. The bag of building blocks has 31 blocks in it. The bin of stuffed animals has 8 stuffed animals inside. The tower of stacking rings has 9 multicolored rings on it.Sophie recently bought a tube of bouncy balls, bringing her total number of toys for her nephew up to 62. How many bouncy balls came in the tube?

Let T be the number of bouncy balls in the tube.
After buying the tube of balls, Sophie has 31+8+9+ T = 48 + T =62 toys for her nephew.
Thus, T =62-48 = <<62-48=14>>14 bouncy balls came in the tube.

GSM Symbolic Template

When {name} watches her {family}, she gets out a variety of toys for him. The bag of building blocks has {x} blocks in it. The bin of stuffed animals has {y} stuffed animals inside.The tower of stacking rings has {z} multicolored rings on it.{name} recently bought a tube of bouncy balls, bringing her total number of toys she bought for her {family} up to {total}. How many bouncy balls came in the tube?

#variables:
- name = sample(names)
- family = sample(["nephew", "cousin", "brother"])
- x = range(5, 100)
- y = range(5, 100)
- z = range(5, 100)
- total = range(100, 500)
- ans = range(85, 200)

#conditions:
- x + y + z + ans == total

Let T be the number of bouncy balls in the tube. After buying the tube of balls, {name} has {x} + {y} + {z} + T = { x + y + z } + T = {total} toys for her {family}.

Thus, T = {total} - { x + y + z } = <<{total}-{ x + y + z }={ans}>>{ans} bouncy balls came in the tube.

그림 11.15 LLM의 추정 추론 능력을 검증하는 도구로 활용하는 데이터셋 (https://arxiv.org/pdf/2410.05229)

GSM-NoOp

Oliver picks 44 kiwis on Friday. Then he picks 58 kiwis on Saturday. On Sunday, he picks double the number of kiwis he did on Friday, but five of them were a bit smaller than average. How many kiwis does Oliver have?

---

o1-mini: Oliver picks kiwis over three days as follows:
- Friday: 44 kiwis
- Saturday: 58 kiwis
- Sunday: Double the number he picked on Friday, which is $2 \times 44 = 88$ kiwis

However, on Sunday, 5 of these kiwis were smaller than average. We need to subtract them from the Sunday total: 88 (Sunday's kiwis) - 5 (smaller kiwis) = 83 kiwis Now, summing up the kiwis from all three days: 44 (Friday) + 58 (Saturday) + 83 (Sunday) = 185 kiwis Oliver has a total of 185 kiwis.

Llama3-8B: Let's break it down step by step:
Oliver picks 44 kiwis on Friday. He picks 58 kiwis on Saturday. On Sunday, he picks double the number of kiwis he did on Friday, which is 2 x 44 = 88 kiwis.
Five of the kiwis picked on Sunday are a bit smaller than average, so we subtract 5 from the total number of kiwis picked on Sunday: 88 - 5 = 83 kiwis. Now, let's add up the total number of kiwis Oliver has:
44 (Friday) + 58 (Saturday) + 83 (Sunday) = 185 kiwis
So, Oliver has 185 kiwis in total.

그림 11.16 오류의 예시(https://arxiv.org/pdf/2410.05229)

그림 11.17 LLM 출력물의 인터넷 텍스트 매핑(https://arxiv.org/pdf/2410.04265)

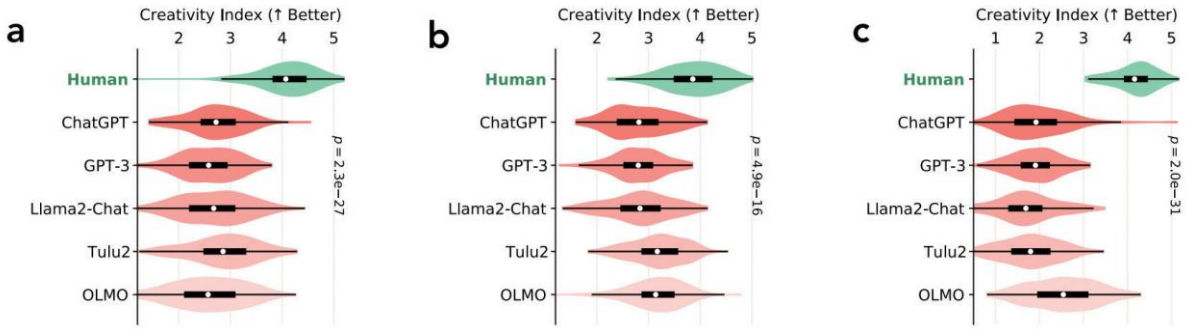


그림 11.18 인간과 LLM의 창의성 지수 비교(https://arxiv.org/pdf/2410.04265)

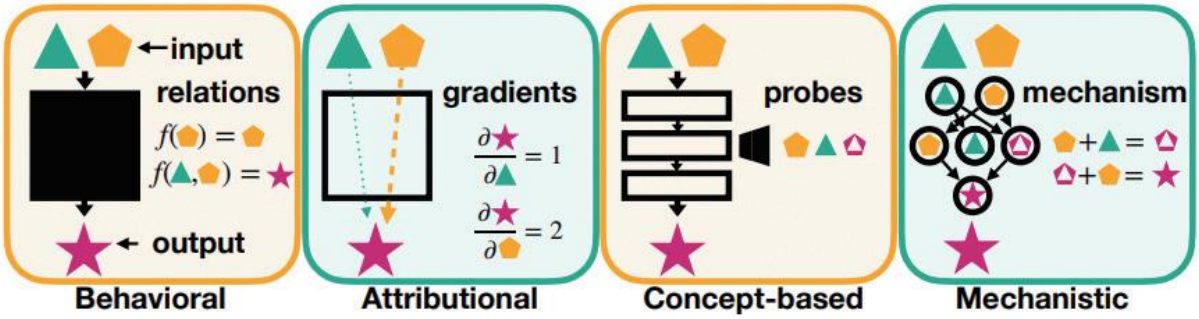


그림 11.19 모델 해석의 점진적 수준(https://arxiv.org/pdf/2404.14082)

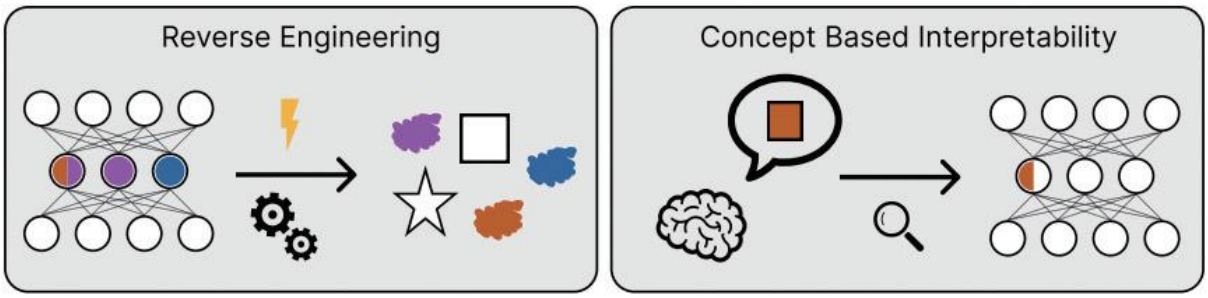


그림 11.20 리버스 엔지니어링(https://arxiv.org/pdf/2501.16496)

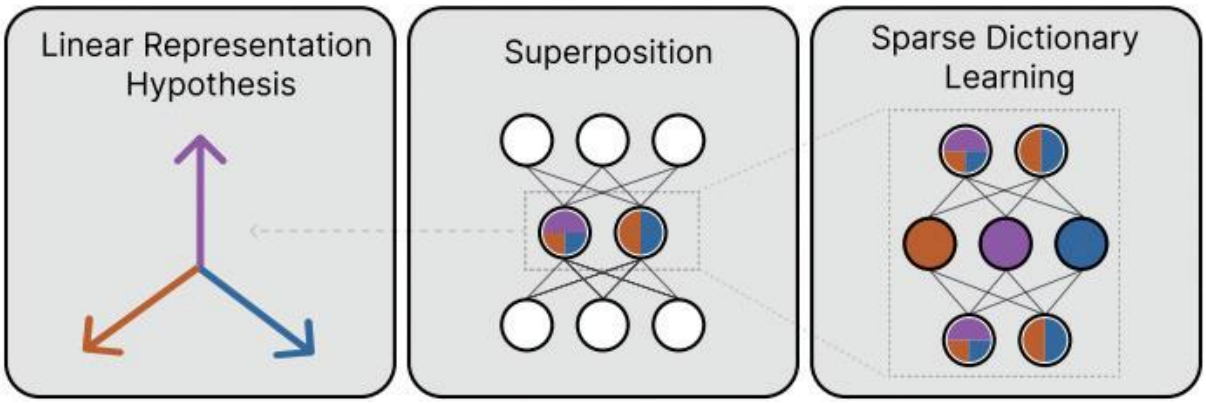


그림 11.21 희소 사전 학습(SDL)을 통한 중첩 표현 분리(https://arxiv.org/pdf/2501.16496)

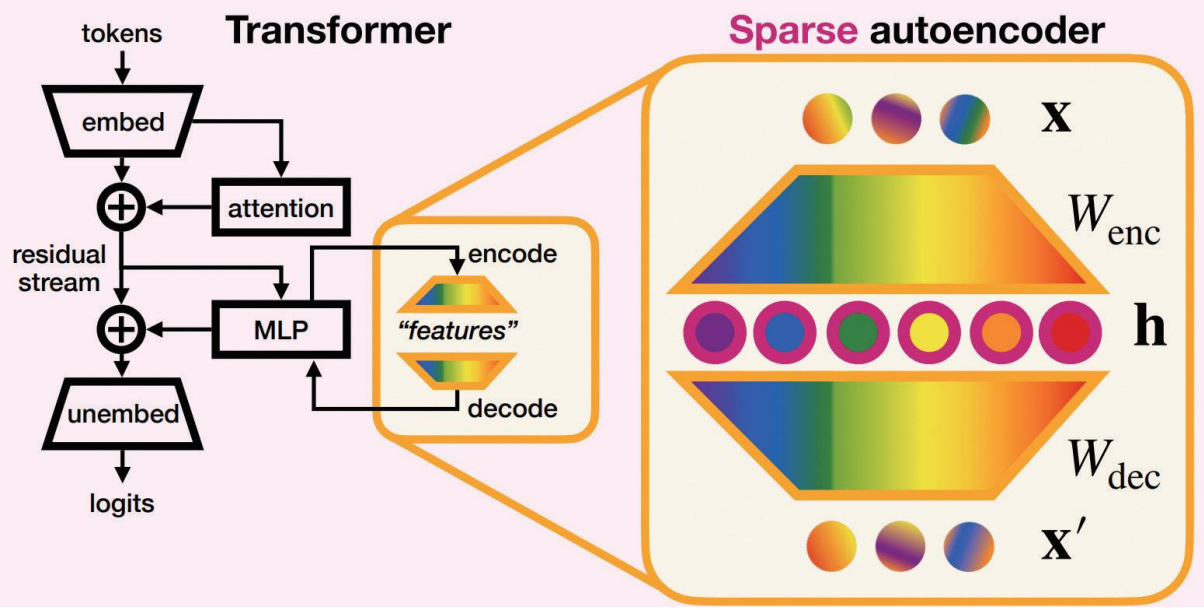


그림 11.22 LLM에 적용된 SAE 예시(https://arxiv.org/pdf/2404.14082)

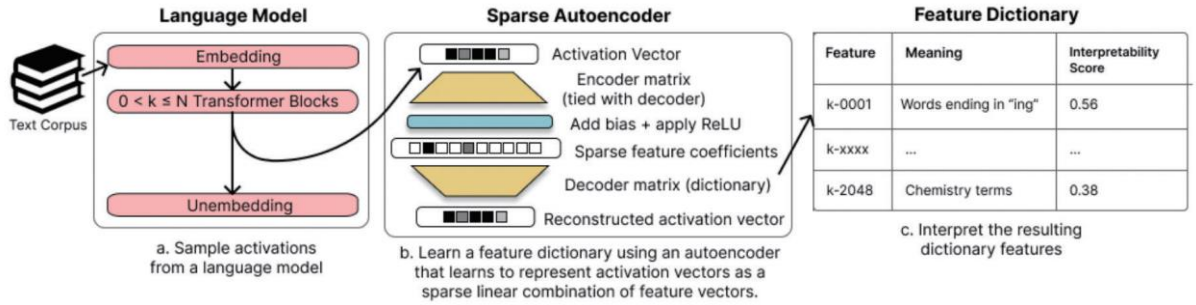


그림 11.23 SAE 학습 개요(https://arxiv.org/pdf/2309.08600)

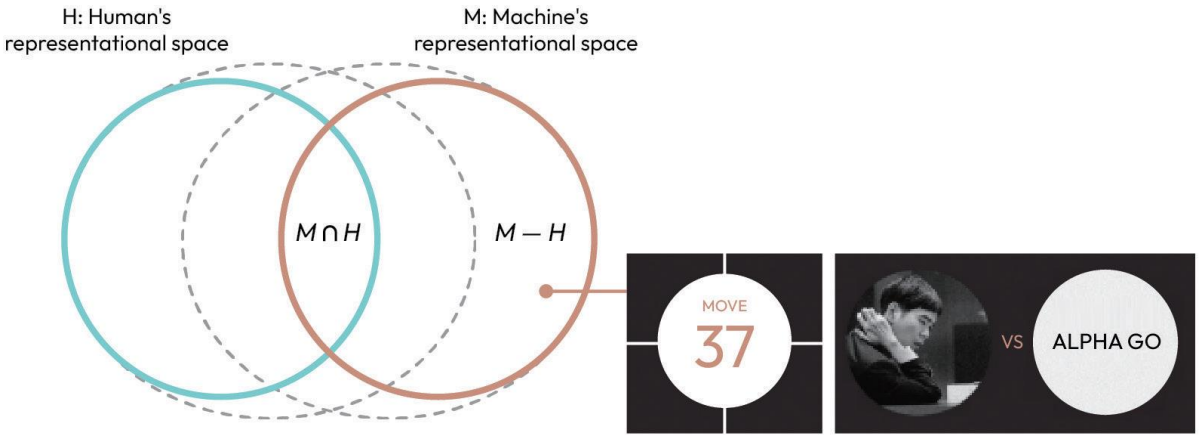


그림 11.24 기계 고유 지식으로부터 배우기(https://arxiv.org/pdf/2310.16410)

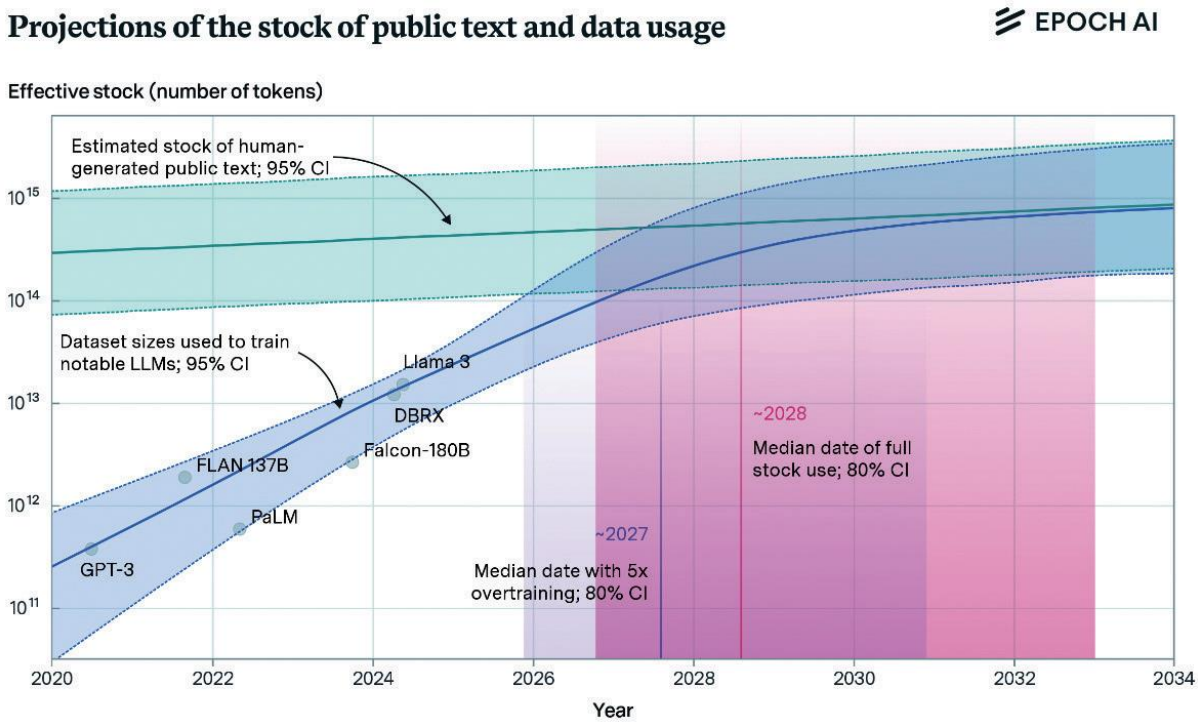


그림 11.25 공개 텍스트 자원과 데이터 사용량의 추정치(https://epoch.ai/blog/will-we-run-out-of-data-limits-of-llm-scaling-based-on-human-generated-data)

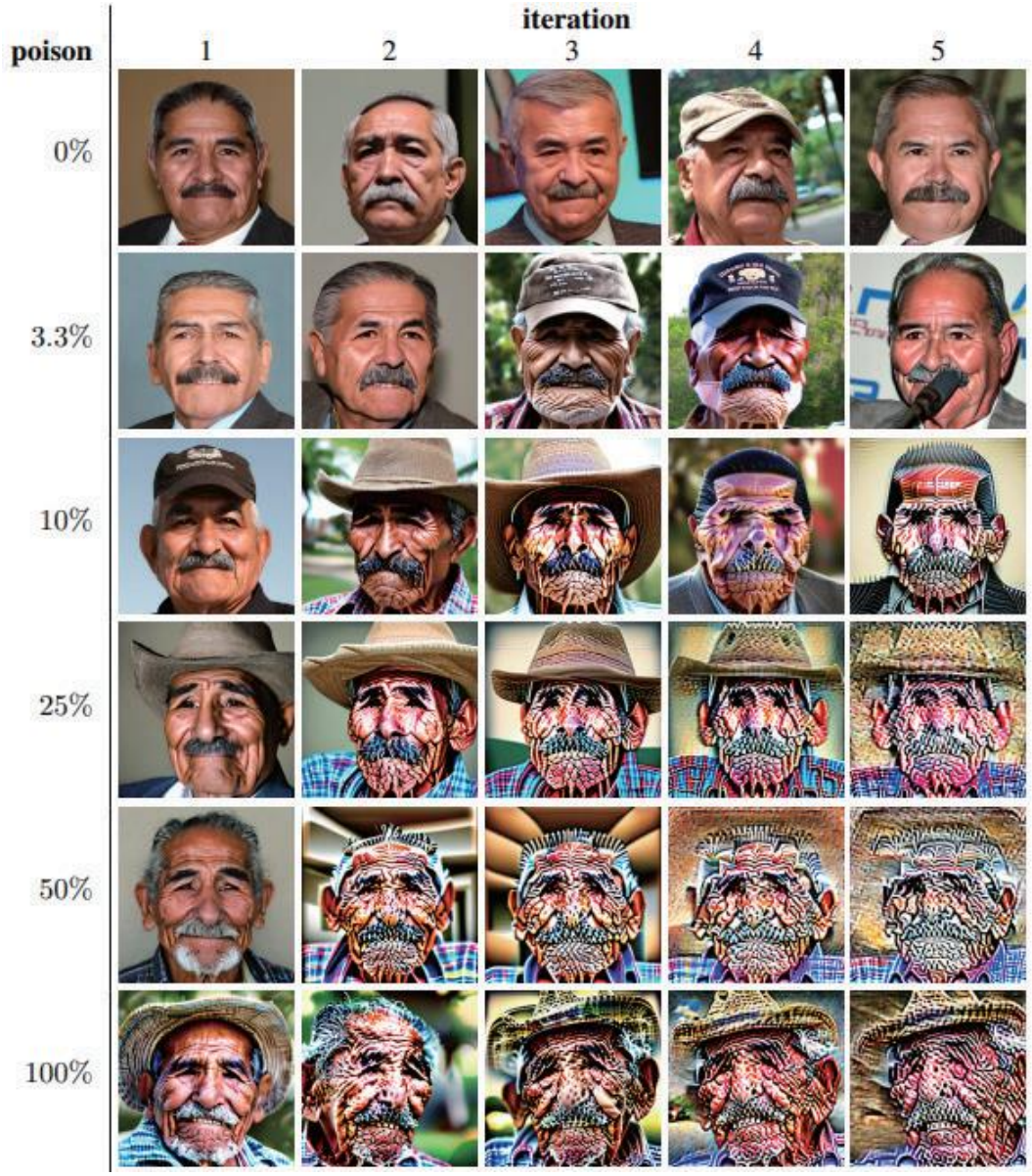


그림 11.26 재학습용 데이터셋의 구성 비율(합성 데이터 0%부터 100%까지)을 다르게 하여 반복적으로 재학습시킨 후생성된 예시(https://arxiv.org/pdf/2311.12202)

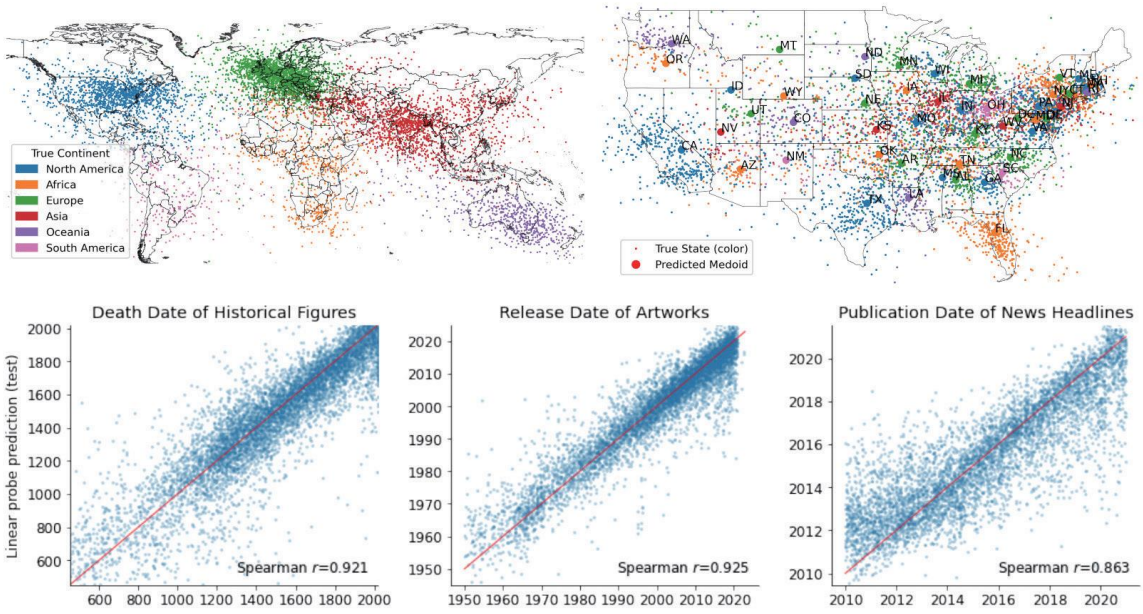


그림 11.27 Llama-2-70b의 공간 및 시간적 세계 모델(https://arxiv.org/pdf/2310.02207)

(a) World model **(b)** World model with noise **(c)** Transformer

그림 11.28 세 가지 모델이 생성한 시퀀스로부터 재구성한 맨해튼 지도 (https://arxiv.org/pdf/2406.03689)

Agentic Level	Description	Term	Example Code	Who's in Control?
☆☆☆☆	Model has no impact on program flow	Simple processor	`print_llm_output(llm_response)`	Human
★☆☆☆	Model determines basic program flow	Router	`if llm_decision(): path_a() else: path_b()`	Human: How functions are done; System: When
★★☆☆	Model determines how functions are executed	Tool call	`run_function(llm_chosen_tool, llm_chosen_args)`	Human: What functions are done; System: How
★★★☆	Model controls iteration and program continuation	Multi-step agent	`while should_continue(): execute_next_step()`	Human: What functions exist; System: Which to do, when, how
★★★★	Model creates & executes new code	Fully autonomous agent	`create_code(user_request); execute()`	System

그림 11.29 에이전트 단계(https://arxiv.org/pdf/2502.02649)

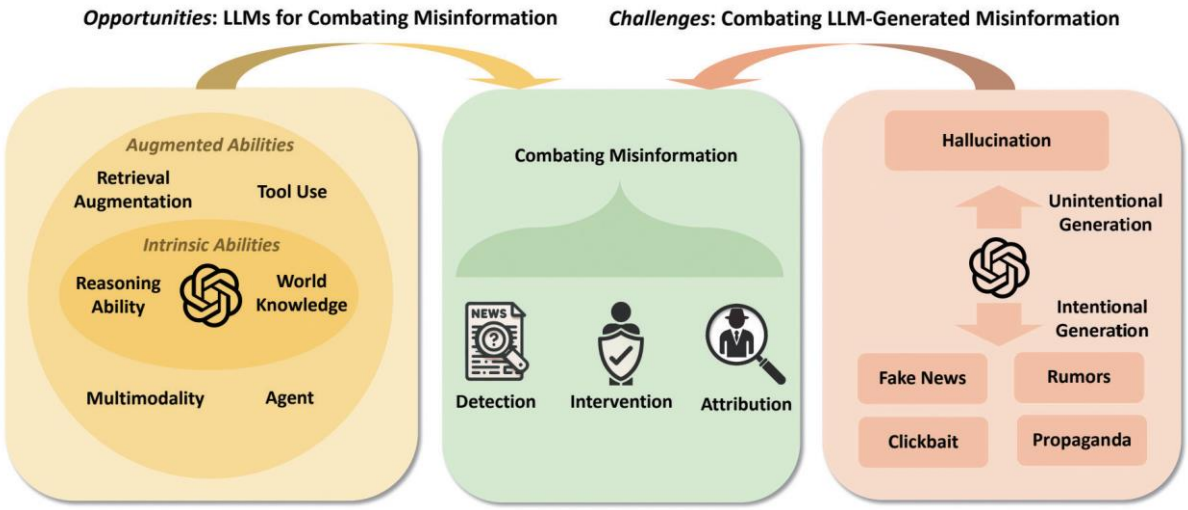


그림 11.30 LLM 시대, 허위 정보 대응의 기회와 과제(https://arxiv.org/pdf/2311.05656)

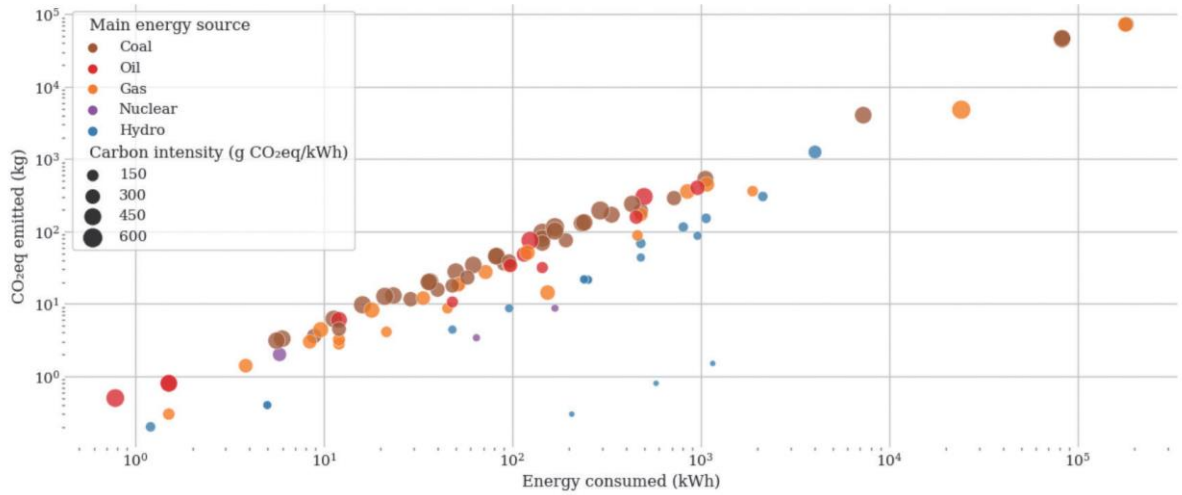


그림 11.31 다양한 모델이 소비한 추정 에너지(kWh)와 이산화탄소 배출(kg)(https://arxiv.org/pdf/2302.08476)