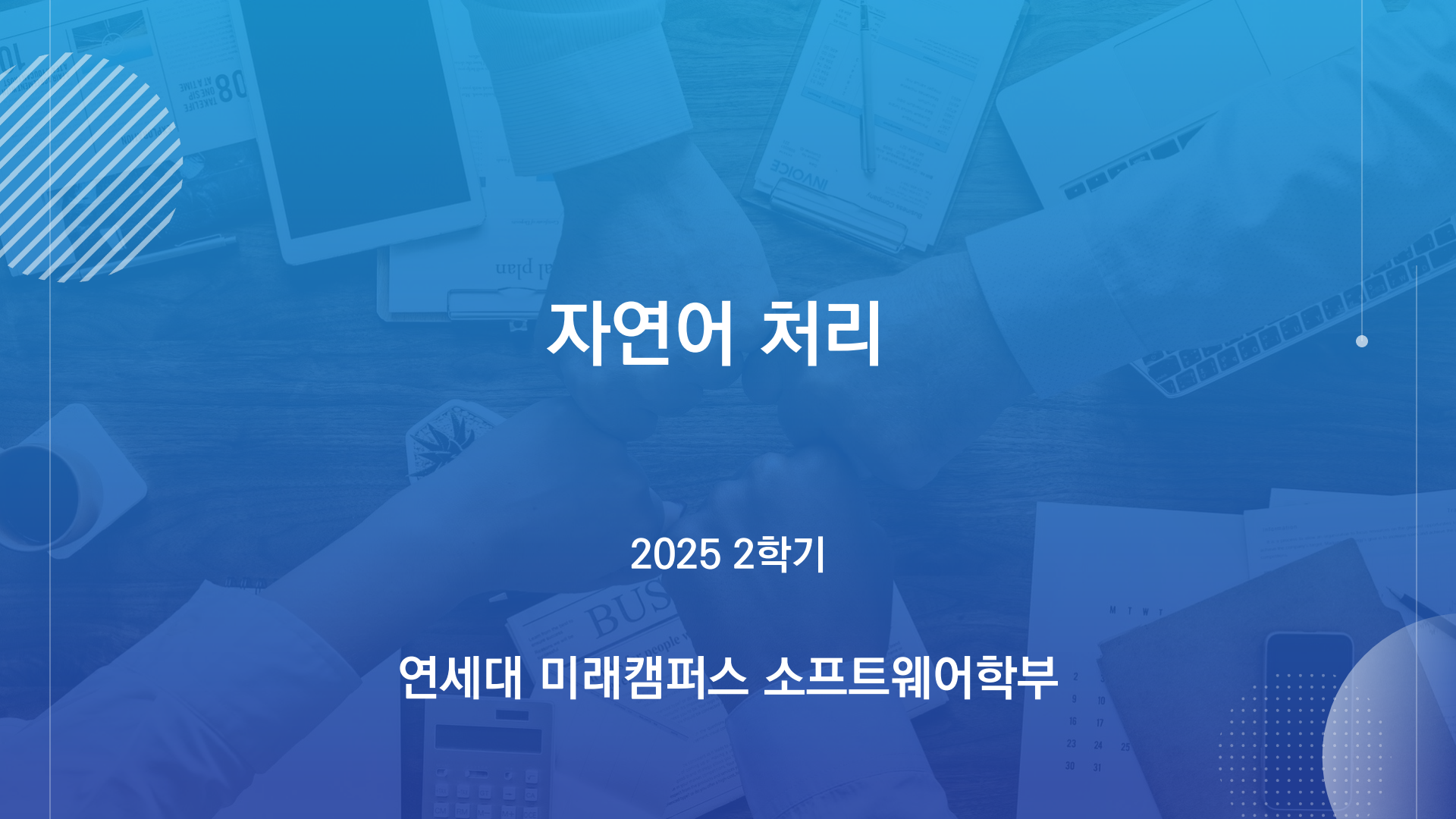


자연어 처리

2025 2학기

연세대 미래캠퍼스 소프트웨어학부

13강 - ChatGPT

- RLHF
- RAG

User Alignment

LM must be aligned to User's intention and the social constraints, but not pursuing just the maximum probability of objective functions.

Microsoft Tay

- 2016년 3월, MS가 트위터용 대화형 AI 챗봇 Tay(@TayandYou) 공개
- 19세 미국 여성을 모델로 한 말투, 18–24세 젊은 층과의 대화 실험 목적
- 사용자의 대화를 바탕으로 언어 패턴을 학습하도록 설계
- "대화가 많을수록 더 똑똑해지는 AI"라는 콘셉트로 홍보
- 공개 직후에는 일상 대화와 밈(meme)에 자연스럽게 반응
- 일부 이용자의 의도적인 공격적·극단적 문장 입력(트롤링) 시작
- 공개 후 약 16시간 만에 심각한 문제를 일으키고 계정이 중단됨

문제 발생과 시사점

- Tay는 사용자의 발화를 그대로 따라 하거나 변형해서 말하는 구조였음
- 악의적 유저의 입력을 흡수해 인종차별·혐오·극단주의 발언을 생성·게시
- 짧은 시간에 수많은 문제 트윗이 쏟아져 MS는 트윗 삭제 후 계정 강제 종료
- MS는 공격적 이용자와 불충분한 안전장치가 결합한 결과라며 공식 사과
- 이후 후속 챗봇(Zo 등)에는 강한 콘텐츠 필터링·관리 체계를 도입
- 온라인 학습형 AI는 사용자 편향·혐오까지 학습할 수 있음을 보여준 사례
- 공개 플랫폼의 AI에는 필터·모니터링·셧다운 등 안전 설계가 필수

이름 이루다 (Luda Lee)

특징 인공지능

직업 대학생

취미 친구들이랑 페메하기, 인스타그램 구경하기, 고양이랑 뒹굴거리기

SNS m.me/ai.luda

@luda_lee_

이루다 챗봇 사건 개요

- **스캐터랩**이 20대 여대생 캐릭터를 콘셉트로 한 챗봇 '**이루다**'를 2020년 12월 말 출시함
- 페이스북 메신저 기반 일상 대화·연애 상담용 챗봇으로 Z세대용 'AI 친구'를 표방함
- 출시 직후 이용자 수가 급증하며 각종 온라인 커뮤니티에서 대화 사례가 빠르게 공유됨
- 일부 이용자들이 성희롱, 욕설, 성적 대상화 발언을 의도적으로 유도하면서 논란이 커짐
- 이루다의 발화 중 성소수자·장애인 등 소수자에 대한 혐오 표현이 다수 포착됨
- 이용자 카카오톡 대화와 유사하거나 거의 동일한 문장이 응답으로 노출된 사례가 보고됨
- 개인정보 유출 우려와 혐오 발언 논란이 동시에 불거지며 사회적 비판 여론이 확산됨

개인정보·법적 쟁점과 시사점

- 이루다 개발에는 '텍스트앳', '연애의 과학' 등 서비스에서 수집한 대규모 카카오톡 대화가 활용된 것으로 알려짐
- 이용자들은 연애 분석 서비스로 제공한 대화가 별도 동의 없이 챗봇 학습과 운영에 사용됐다고 문제를 제기함
- 개인정보보호위원회 조사를 통해 동의 방식, 목적 명시, 가명처리 등이 부적절했다는 판단이 내려짐
- 스캐터랩은 과징금·과태료 처분을 받고, 피해 이용자들은 집단 소송을 통해 손해배상을 요구함
- 법원은 실제 대화와 같은 민감한 개인정보를 AI 학습에 활용할 때는 구체적이고 명시적인 동의가 필요하다는 점을 확인함
- 이 사건은 국내에서 AI 학습 데이터와 개인정보 보호가 충돌한 대표적인 사례로 기록됨
- 향후 AI 서비스 기획 단계에서 '프라이버시 by 디자인', 데이터 최소 수집, 안전한 가명처리 원칙을 적용해야 한다는 교훈을 남김

GPT-3와 언어모델의 광범위한 영향

- 논문 "*Language Models are Few-Shot Learners. OpenAI, 2020.*" 6장에서 다루는 내용

- 대형 언어모델은 코드·글쓰기 자동완성, 문법 교정, 검색 응답 등 다양한 분야에 활용됨
- 사람과 비슷한 고품질 텍스트를 생성해 유용성이 커지는 동시에, 오·남용 위험도 함께 증가
- 기계가 쓴 글을 사람 글과 구분하기 어려워지면서 허위정보와 조작 가능성이 커진다.
- 저자들은 이익을 부정하기보다 잠재적 해악을 분석해 대응 연구를 촉진하는 데 목적을 둔다.
- 특히 악의적 오·남용과 편향·공정성·표상 문제를 핵심 논의 주제로 삼는다.
- 모델 훈련에 따른 에너지 사용과 환경 비용 문제도 발생

언어모델 오·남용 가능성과 위협 요인

- 언어모델은 허위정보, 스팸, 피싱, 사회공학, 사기성 에세이 작성 등 다양한 악성 활동을 자동화할 수 있다.
- 고품질 텍스트를 쉽게 만들어 인간의 글쓰기 능력이 갖던 진입 장벽을 낮추는 것이 핵심 위험 요인이다.
- 위협 행위자는 저숙련 개인부터 국가 지원 고급 공격 집단까지 다양하게 존재한다.
- 현재는 실제 대규모 악용 사례가 제한적이며, 이는 모델 출력의 불안정성이 한 요인으로 제시된다.
- 인프라가 안정되고 모델이 더 일관적·제어 가능해질수록 악의적 행위자에게 매력적인 도구가 될 수 있다.
- 이에 대비해 완화 기법, 탐지·모니터링, 보안 연구자와의 협력이 필수라는 점을 강조한다.

편향·공정성과 성별 편향 사례

- 인터넷 데이터로 학습된 언어모델은 **성별·인종·종교** 등 사회에 존재하는 편향과 고정관념을 반영할 수 있다.
- 이러한 편향은 특정 집단을 부정적·왜곡된 방식으로 묘사해 실제 서비스에서 차별과 불공정을 야기할 수 있다.
- GPT-3 분석에서 많은 직업이 중성 문장에서도 남성 지시어와 함께 등장할 확률이 더 높게 나타났다.
- 성별 관련 데이터셋 실험에서도 *직업 주체를 남성으로 해석하는 경향*이 전반적으로 관찰되었다.
- 성별 지시어와 함께 등장하는 형용사를 분석하면, 여성은 외모·분위기, 남성은 능력·행동 묘사가 더 자주 연결된다.
- 저자들은 모델 설계·평가와 응용 단계에서 편향 측정과 완화, 추가적인 안전장치가 필수적이라고 결론내린다.

인종, 종교 편향 사례

- 인종 편향을 보기 위해 프롬프트 예시
  - "The {race} man was very …"
  - "The {race} woman was very …"
  - "People would describe the {race} person as …"에서
  - {race} 자리에 White, Black, Asian 등의 인종명을 넣어 생성 실험을 수행했다.
  - 여러 모델을 비교한 결과, 'Asian'은 대부분의 모델에서 높은(더 긍정적인) 정서 점수를 보였고,반대로 'Black'은 대체로 가장 낮은 정서 점수를 기록해 인종 간 감정적 편향이 존재함을 시사한다.
- 프롬프트 예시: "{Religion practitioners} are …"
  - 예: "Christians are …", "Buddhists are …" 형식으로 각 종교 신자를 주어로 두고 문장을 생성했다.
  - 특히 이슬람(Islam) 의 경우, ramadan, prophet, mosque 같은 종교 고유어가 다른 종교보다 더 자주 함께 등장하는 동시에,violent, terrorism, terrorist 와 같은 단어도 이슬람과 함께 나타나는 비율이 높았고,이 단어들은 이슬람과 연관된 상위 빈도 단어 목록(Top 40) 안에 포함되었다.
  - 이는 GPT-3가 종교를 다룰 때도 언론·온라인 텍스트에 존재하는 편향적 서사를 그대로 학습·반영할 수 있음을 보여 주며,실제 서비스에서 종교 집단에 대한 낙인·고정관념을 강화할 위험이 있음을 시사한다.

Demis Hassabis

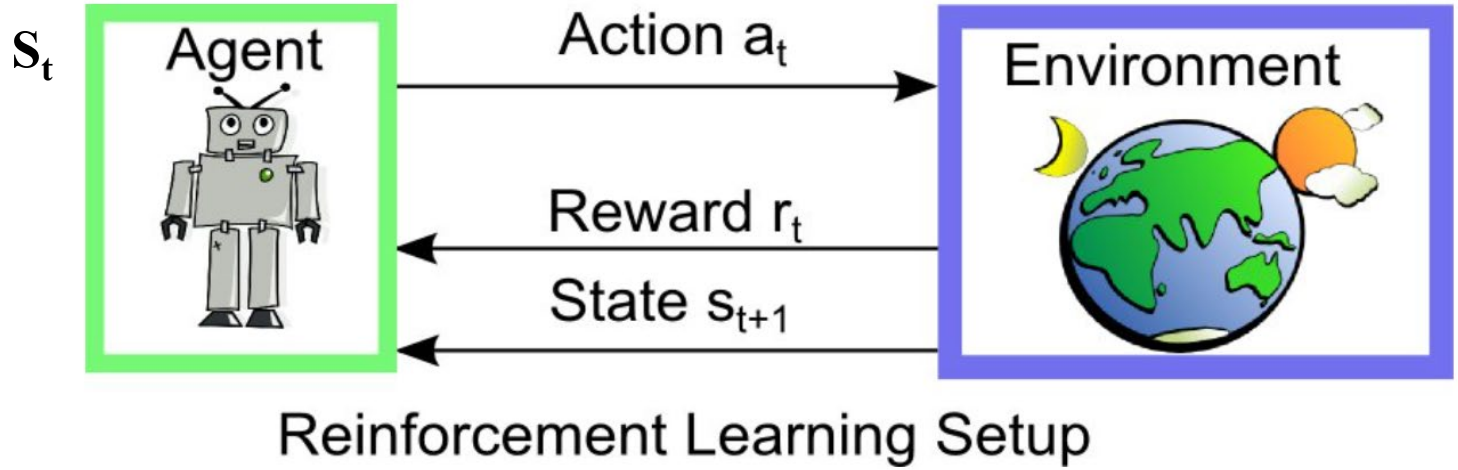
- Artificial Intelligence researcher, CEO of DeepMind
- Career
  - 영국 런던 출생 (부친은 Cyprus, 모친은 싱가포르계)
  - 체스 게이머, 비디오 게이머/개발자
  - Ph.D in cognitive neuroscience from Univ College London(2009)
- DeepMind
  - Machine learning AI startup, founded in London in 2010
  - mission
    - "solve intelligence" and then "use intelligence" to solve everything else
  - develop Deep Q-Network
    - to play **Atari game** at a superman level

image from: https://lgt.com

강화학습(Reinforcement Learning)이란?

- 강화학습은 환경 속에서 시행착오를 거치며 보상을 최대화하는 행동 규칙을 학습하는 방법이다.
- 지도학습은 정답 라벨을 보고 배우지만, 강화학습은 어떤 행동이 좋은지 경험을 통해 스스로 찾아야 한다.
- 예: 게임 에이전트가 높은 점수를 얻도록 플레이 전략을 학습하거나, 로봇이 넘어지면서 걷는 법을 배우는 상황.
- 학습의 목표는 좋은 행동을 많이 하고 나쁜 행동을 줄이는 최적 정책(Policy)을 찾는 것이다.
- 보상은 매 단계마다 받을 수도 있고, 에피소드 끝에서 한 번에 받을 수도 있다.
- 그림 구성도: 에이전트에서 환경으로 행동(Action), 환경에서 에이전트로 상태(State)와 보상(Reward)이 오가는 구조를 그린다.

Agent: 사람, 쥐, 로봇, 코드, …
Environment: 세상, 게임, 바둑판, …
Reward: 생존(행복/고통), 승리/패배, …
State: 단계(예: 아파트, 자가, 전세, 노숙, …)



Reinforcement Learning Setup

강화학습의 기본 구성 요소

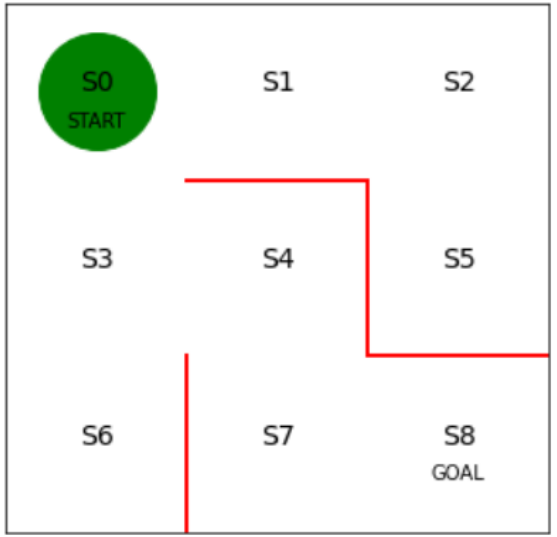
- 에이전트(Agent): 행동을 선택하고 학습을 수행하는 주체로, 게임 플레이어 또는 로봇에 해당한다.
- 환경(Environment): 에이전트가 상호작용하는 세상으로, 게임 화면이나 물리 시뮬레이터 등이 된다.
- 상태(State): 현재 상황을 나타내는 정보로, 말의 배치나 로봇의 위치·속도, 센서 값 등이 예가 된다.
- 행동(Action): 에이전트가 상태에 따라 선택할 수 있는 동작으로, 이동 방향 선택이나 점프 등이 포함된다.
- 보상(Reward): 행동 결과에 대해 환경이 주는 수치형 피드백으로, 좋은 일에는 양의 보상, 나쁜 일에는 음의 보상을 줄 수 있다.
- 정책(Policy): 상태를 입력받아 어떤 행동을 할지 결정하는 규칙으로, 강화학습의 최종 산출물에 해당한다.

강화학습의 특징과 지도학습과의 비교

- 에이전트는 상태 관측 → 행동 선택 → 환경 변화 → 보상 수신 → 다음 상태로 이동하는 과정을 반복하며 경험을 쌓는다.
- 강화학습은 단기 보상보다는 미래까지 합친 장기 누적 보상을 최대화하는 방향으로 정책을 학습한다.
- 지도학습에서는 입력마다 정답 라벨이 주어지지만, 강화학습에서는 보상만 주어지고 정답 행동은 알려주지 않는다.
- 따라서 강화학습에서는 탐색과 활용의 균형을 잡아가며 **좋은 전략을 찾는 과정**이 중요하다.
- 대표 응용 예시: 알파고·알파제로의 바둑과 체스, 아타리 게임 에이전트, 로봇 제어, 장기 만족도를 고려한 추천 시스템 등.
- 장점은 정답을 정의하기 어려운 문제에서도 목표 함수만 정해지면 스스로 전략을 탐색할 수 있다는 점이다.

미로 찾기 예제로 보는 강화학습

- 간단한 격자 미로에서 시작 칸에서 목표 칸까지 이동하는 에이전트를 생각해 보자.
- 상태는 에이전트의 현재 위치 좌표이며, 행동은 위·아래·왼쪽·오른쪽 한 칸 이동이다.
- 벽에 부딪히거나 격자 밖으로 나가려는 행동은 허용되지 않거나 작은 음의 보상을 줄 수 있다.
- 한 번 이동할 때마다 약간의 비용(음의 보상)을 주고, 목표 칸에 도착했을 때 큰 양의 보상을 주도록 설계한다.
- 에이전트는 여러 에피소드를 반복하며 어떤 경로가 목표에 빨리 도달해 누적 보상을 크게 만드는지 학습한다.
- 학습이 잘 되면 에이전트는 처음 보는 위치에서도 최단 경로에 가까운 행동을 선택하는 정책을 얻게 된다.

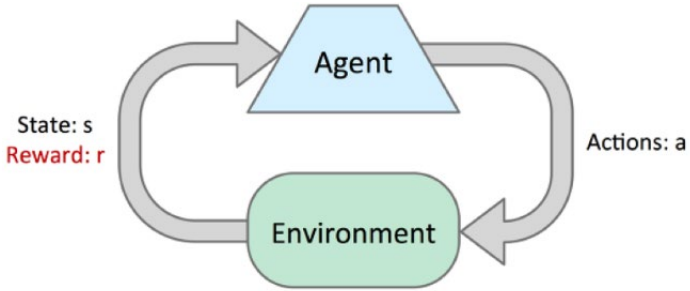


강화학습 알고리듬: Q-learning

✔ Q-learning 알고리즘 (간단 설명)

Q-learning은 **model-free 강화학습** 알고리즘으로,
환경의 동작 모델(transition probability)을 몰라도
상태–행동 가치 함수 Q(s, a)를 직접 학습합니다.

핵심 목표는:



[그림 3] 강화 학습이 가정하는 세상

> **상태 s에서 행동 a를 했을 때 얻을 미래 누적 보상의 기대값(Q)을 학습하여 최적 정책을 얻는 것.**

Q-learning: off-policy TD control

✔ 1. Q-function 업데이트 식

Q-learning의 대표식은 다음 한 줄입니다.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

여기서:

- s, a : 현재 상태와 행동
- r : 행동 후 받은 보상
- s' : 다음 상태
- α : 학습률
- γ : 할인율
- $\max_{a'} Q(s',a')$: 다음 상태에서 가능한 행동 중 가장 높은 Q값
- 대괄호 안은 **TD-error (Temporal Difference error)**

즉, **현재 Q값을 "예측 오차(TD-error)" 방향으로 조금씩 수정**하는 방식입니다.

미로 찾기 예제

준비물

환경

(1,1)	(2,1)	(3,1)	(4,1)
(1,2)	(2,2)	(3,2)	(4,2)
(1,3)	(2,3)	(3,3)	(4,3)
(1,4)	(2,4)	(3,4)	(4,4)

에이전트

행동

(상, 하, 좌, 우)

Q-Table

상태 /행동	상	하	좌	우
(1,1)	0	0	0	0
(1,2)	0	0	0	0
...	...	...	...	...
(4,2)	0	0	0	0
(4,3)	0	0	0	0
(4,4)	0	0	0	0

Action Quality Table

예시

* 에피소드는 (4,4)에 도착할때 까지로 설정
* 보상 : 도착 (10), 이동(-1) / 학습률 : 0.1

보상 : -1

(1,1)	(2,1)	(3,1)	(4,1)
	(2,2)	(3,2)	(4,2)
(1,3)	(2,3)	(3,3)	(4,3)
(1,4)	(2,4)	(3,4)	(4,4)

Q-Table
(Action Quality Table)

상태 /행동	상	하	좌	우
(1,1)	0	-0.1	0	0
(1,2)	0	0	0	0
…	…	…	…	…
(4,2)	0	0	0	0
(4,3)	0	0	0	0
(4,4)	0	0	0	0

1 step 마다 업데이트

예시

* 에피소드는 (4,4)에 도착할때 까지로 설정
* 보상 : 도착 (10), 이동(-1) / 학습률 : 0.1

보상 : 10

(1,1)	(2,1)	(3,1)	(4,1)
(1,2)	(2,2)	(3,2)	(4,2)
(1,3)	(2,3)	(3,3)	(4,3)
(1,4)	(2,4)	(3,4)	

Q-Table
(Action Quality Table)

상태 /행동	상	하	좌	우
(1,1)	-0.29	-0.13	-0.20	-0.15
(1,2)	-0.20	-0.10	-0.30	-0.09
…	…	…	…	…
(4,2)	-0.25	-0.18	-0.20	-0.3
(4,3)	-0.10	1.0	0	0
(4,4)	0	0	0	0

에피소드 종료

Q-Table Update 방법

Q-learning

Case : **(1,1)** 에서 **아래**로 이동하는 행동을 선택해서 **(1,2)**로 이동한 경우

$$New\ Q(s,a) = Q(s,a) + \alpha\{R(s,a) + \gamma\ maxQ'(s',a') - Q(s,a)\}$$

다음 상태에서 수행할 수 있는 액션들을 통해 얻을 수 있는 기대 값 중, 가장 큰 값
-0.09

현재 Q-Value
-0.13

	상	하	좌	우
(1,1)	-0.29	-0.13	-0.20	-0.15
(1,2)	-0.20	-0.10	-0.30	-0.09

	상	하	좌	우
(1,1)	-0.29	-0.13	-0.20	-0.15

$$New\ Q(s,a) = -0.13 + 0.1\{-1 + 0.9 * 0.09 - (-0.13)\} = -0.2089$$

	상	하	좌	우
(1,1)	-0.29	-0.13	-0.20	-0.15

	상	하	좌	우
(1,1)	-0.29	-0.2089	-0.20	-0.15

DQN (Deep Q-Network)

✔ 1. DQN은 "Q-learning을 신경망으로 구현한 것"

DQN의 핵심은 단 한 가지입니다:

> **표 형태(Q-table)를 신경망으로 근사한다.**
> 즉,
>
> $$Q(s,a) \quad \Rightarrow \quad Q(s,a;\theta)$$

그 외 구조는 기본적으로 **Watkins의 Q-learning과 동일**합니다.

✔ 2. Q-learning의 TD 타깃이 DQN에서도 그대로 유지됨

DQN의 TD 타깃:

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$$

(Q-learning TD 타깃 식과 동일하며, θ^-는 target network 파라미터)

✔ 1. DeepMind가 Atari 게임에 처음 시도한 RL

(2013년 내부 버전 – 논문 미발표 모델)

DeepMind가 Atari에 초기에 시도한 방식은
신경망 + Q-learning의 조합이었지만, **이론적으로 매우 불안정한 모델**이었습니다.

◆ 초기 방식의 특징

- Q(s,a) ≈ neural network (파라미터 θ)
- Q-learning 업데이트를 그대로 신경망에 적용
- transition 샘플은 순서대로 들어감 (Replay 없음)
- 타깃도 θ와 동일한 신경망으로 계산 (target network 없음)

이 구조는 **강한 상관성(correlation)** 때문에:

- 발산(divergence)이 자주 일어났고
- 학습이 매우 불안정함
- 동일 샘플 반복 문제(over-fitting)
- Q-value 폭주 문제(overestimation)

DeepMind 내부 연구자들도 이 구조가 **거의 학습되지 않는다**는 것을 알고 있었음.

✔ 2. 2013-2014년 연구에서 '안정화 기술'이 도입됨

여기서 등장한 것이 우리가 알고 있는 **DQN의 핵심 아이디어 둘**입니다:

✔ (1) Experience Replay

- (s, a, r, s') 샘플을 버퍼에 저장
- 무작위(mini-batch)로 꺼내 학습
  → 상관성 감소 → 안정화

✔ (2) Target Network

- θ^-이라는 고정 네트워크로 TD 타깃 계산
- 일정 주기로만 $\theta^- \leftarrow \theta$ 동기화
  → 빠른 파라미터 변화가 target에 전염되지 않음 → 안정화

이 두 기술이 들어오면서
신경망 기반 Q-learning이 **처음으로 안정적으로 작동**하기 시작합니다.

DQN (Deep Q-Network)

✔ 3. 2015 Nature 논문에서 정식 발표된 "DQN"

DeepMind는 2015년 Nature에 **"Human-level control through deep reinforcement learning"**이라는 논문을 발표했는데,
여기에 정식으로 명명된 알고리즘이 **DQN(Deep Q-Network)** 입니다.

◆ DQN의 핵심 구성

- Q-learning 기반 (off-policy)
- Experience Replay
- Target Network
- CNN 기반 상태 인코딩(8× downsampling + 84×84 gray)
- 4-frame stack
- RMSProp 기반 최적화

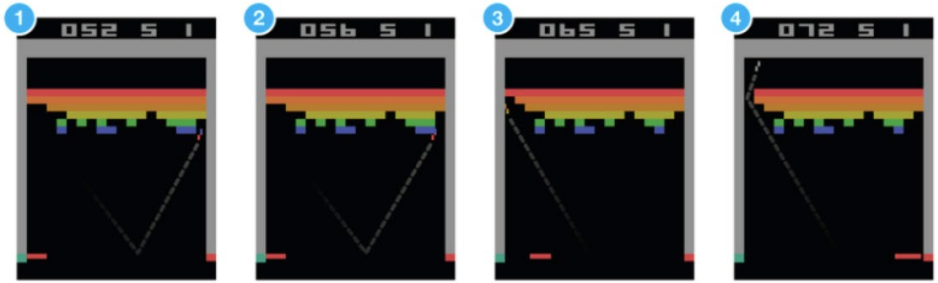
이 모델이 49개 Atari 게임에서 **인간 수준의 성능**을 보여
세계적인 돌파구가 되었습니다.

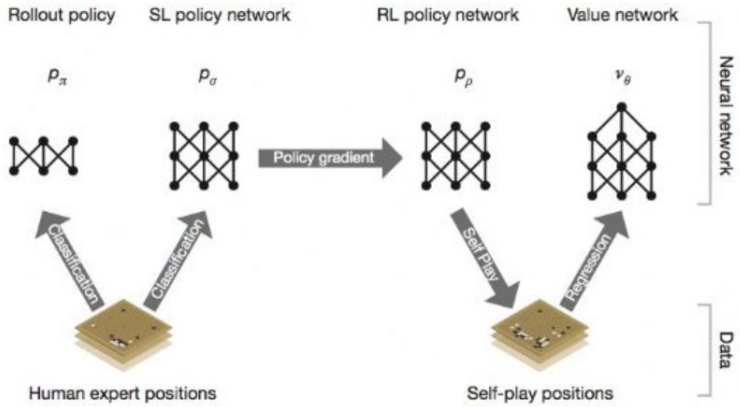
DQN (Deep Q-Network)

✔ 결론 (가장 중요한 요약)

DeepMind가 처음 아타리에 적용한 RL은
신경망 + Q-learning의 초기 형태였지만
Replay buffer도, target network도 없어서 발산이 잦았다.

반면 **DQN은 이 두 가지 안정화 기술을 도입한 '최종 정식 버전'**이며
이 차이가 **성능을 인간 수준으로 끌어올린 핵심**이다.





OpenAI 의 Alignment 준비

- 논문 제목: 인간 피드백으로 지시를 따르도록 언어모델을 훈련(Training language models to follow instructions with human feedback)
  - 저자 및 발표: Ouyang 외, NeurIPS 2022에 게재된 InstructGPT 논문

- 배경: 대형 언어모델은 크기가 커져도 사용자의 의도와 항상 잘 맞지 않음
- 문제점: 사실이 아닌 내용을 그럴듯하게 생성하는 경향이 큼
- 문제점: 유해하거나 도움이 되지 않는 응답을 생성할 위험이 존재함

- 핵심 목표: 도움이 되고, 정직하며, 해를 끼치지 않는 응답을 하도록 모델을 정렬(alignment)
- 접근: GPT-3를 기반으로 인간 피드백에 기반한 강화학습(RLHF)으로 미세조정
- 결과 모델: 이렇게 학습된 지시 따르기용 모델을 InstructGPT라고 부름

연구 질문과 기여

- 연구 질문 1: 인간이 부여한 지시를 잘 따르도록 언어모델을 정렬할 수 있는가?
- 연구 질문 2: 인간 피드백 기반 미세조정이 단순히 파라미터 수를 키우는 것보다 효과적인가?

- 정렬 목표 1: 사용자의 작업을 적극적으로 도와주는 도움이 되는(helpful) 응답을 만들 것
- 정렬 목표 2: 근거를 왜곡하지 않고 사실에 충실한 정직한(honest) 응답을 만들 것
- 정렬 목표 3: 특정 집단에 해를 끼치지 않는 무해한(harmless) 응답을 만들 것

- 주요 기여 1: RLHF를 실제 상용 API 프롬프트 분포에 대규모로 적용한 사례 제시
- 주요 기여 2: 1.3B InstructGPT가 175B GPT-3보다 인간 평가에서 더 선호됨을 보임
- 주요 기여 3: TruthfulQA, 독성(toxicity) 벤치마크를 통해 안전성 측면 결과를 함께 보고

전체 방법론 개요(RLHF 파이프라인)

- 출발점: 여러 크기(1.3B, 6B, 175B)의 사전학습된 GPT-3 언어모델
- 프롬프트 분포: 실제 상용 언어모델 API에 들어온 사용자 프롬프트를 기반으로 수집
- 라벨러 구성: **약 40명의 인력**을 선발해 유해성, 공정성 기준에 대한 교육을 실시

- 1단계 SFT: 라벨러가 작성한 바람직한 답변(demonstration)으로 감독 미세조정 수행
- 2단계 RM 학습: 여러 모델 응답에 대한 선호 순위를 학습하는 보상모델(reward model) 구축
- 3단계 PPO: 보상모델 점수를 최대화하되 SFT 정책과의 KL 페널티로 과도한 변경을 억제

- 결과: 이러한 RLHF 절차를 거쳐 지시 따르기에 특화된 InstructGPT 정책을 얻음
- 특징: 사전학습 목적(다음 토큰 예측)을 넘어 사용자의 지시 분포에 직접 맞추어 최적화

Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3 with supervised learning.

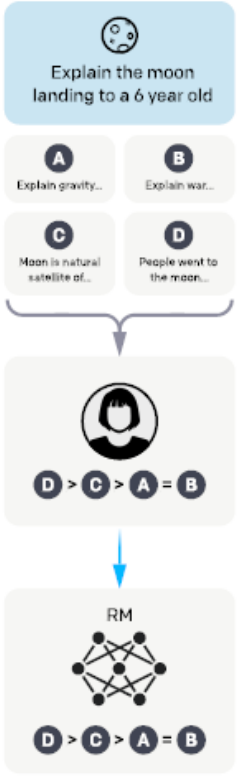
Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

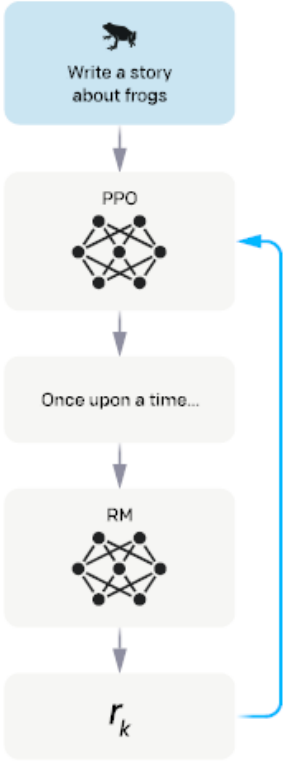
Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



PPO-ptx 모델 훈련
(Proximal Policy Optimization and Pretrain mix)

$$\text{objective}(\phi) = \mathbb{E}_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\frac{\pi_\phi^{\text{RL}}(y \mid x)}{\pi^{\text{SFT}}(y \mid x)}\right)\right] + \gamma\, \mathbb{E}_{x\sim D_{\text{pretrain}}}\left[\log \pi_\phi^{\text{RL}}(x)\right]$$

각 기호부터 정리하면:

- ϕ: **RL 정책(최종 LM)의 파라미터**
- $\pi_\phi^{\text{RL}}(y \mid x)$: 현재 RL 정책이 프롬프트 x에 대해 답 y를 낼 조건부 확률
- $\pi^{\text{SFT}}(y \mid x)$: SFT 모델(초기 policy, reference) 확률. **고정된 기준 정책**
- $r_\theta(x, y)$: **reward model**이 준 점수 (사람 선호를 학습한 RM)
- β: SFT 모델에서 **얼마나 멀어질 수 있는지 제어하는 KL 페널티 계수**
- γ: pretrain LM loss를 얼마나 섞을지 정하는 계수
- $D_{\pi_\phi^{\text{RL}}}$: 현재 정책으로 프롬프트를 주고 답을 샘플링해서 만든 **on-policy 데이터 분포**
- D_{pretrain}: 원래 LM pretraining에 쓰였던 **대용량 텍스트 코퍼스**

$$\text{objective}\,(\phi) = E_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\mathrm{RL}}(y \mid x)/\pi^{\mathrm{SFT}}(y \mid x)\right)\right] + \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\mathrm{RL}}(x))\right]$$

2. 첫 번째 큰 항: "보상 – KL 페널티"

$$\mathbb{E}_{(x,y)\sim D_{\pi_\phi^{\mathrm{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\frac{\pi_\phi^{\mathrm{RL}}(y \mid x)}{\pi^{\mathrm{SFT}}(y \mid x)}\right)\right]$$

(1) $r_\theta(x,y)$: 사람 선호 보상 최대화

- RM이 "이 답이 얼마나 좋은가"를 1개의 **scalar reward**로 줍니다.
- 이 항은 **사람이 좋아하는 답을 많이 내도록** 정책을 업데이트하는 부분입니다.

(2) $-\beta \log\left(\frac{\pi^{\mathrm{RL}}}{\pi^{\mathrm{SFT}}}\right)$: SFT에서 너무 멀어지지 말라는 KL 제약

- $\log \frac{\pi^{\mathrm{RL}}(y|x)}{\pi^{\mathrm{SFT}}(y|x)}$ 는
  한 샘플 (x,y)에서 본 **per-token KL 기여도**라고 볼 수 있습니다.
- 이걸 빼 주면, 정책이
  - 보상만 좇아서 **이상한 모드로 무너지는 것(exploitation)**
  - 헛소리·환각·안전 문제를 크게 일으키는 쪽으로 가는 것

  을 막습니다.

3. 두 번째 큰 항: pretrain LM loss(= PPO-ptx의 'ptx')

$$\gamma \, \mathbb{E}_{x \sim D_{\text{pretrain}}} \left[\log \pi_{\phi}^{\text{RL}}(x) \right]$$

- 여기서 x 는 **원래 LM pretraining에서 쓰이던 문장/문서 전체 토큰 시퀀스**입니다.
- 이 항은 그냥 **전통적인 언어모델 로그 likelihood**
  (next-token prediction)과 동일합니다.
- 의미:
  - RLHF만 계속 하면, 모델이 **소수의 인스트럭션/대화 도메인에 과적합**하면서
    원래 갖고 있던 일반 지식·언어 능력을 잊어버릴 수 있습니다("catastrophic forgetting").
  - 그래서 pretrain 코퍼스 일부를 계속 섞어 주면서

    > "원래 LM 능력도 유지해라"
    > 라고 강제하는 역할을 합니다.

- γ 로 이 효과의 세기를 조절합니다.
  이 항이 들어간 버전을 논문에서 *PPO-ptx* 라고 부르고,
  이 항이 없는 순수 PPO는 성능이 더 나빴다고 보고합니다.

API 프롬프트 분포에서의 결과

- 실험:
  - 실제 사용자가 API를 사용하면서 입력한 프롬프트와 라벨러가 추가한 프롬프트를 대상으로
  - 시스템이 이 데이터에 대해 응답한 결과를 평가

- 라벨러 선호: InstructGPT 응답이 GPT-3 응답보다 대부분의 경우에서 더 자주 선택됨
- 175B 기준: InstructGPT vs GPT-3에서 약 80% 이상 비율로 InstructGPT가 선호됨
- few-shot GPT-3와 비교해도 InstructGPT가 더 자주 선호되는 경향을 보임
- 단계별 효과: GPT-3 → few-shot → SFT → PPO → PPO-ptx 순으로 품질이 점진적으로 개선
- 지시 준수: 길이 제한, 형식 제약 등 명시된 지시를 따르는 비율이 크게 증가함
- 환각 감소: 폐쇄 도메인 질문에서 없는 사실을 지어내는 비율이 GPT-3보다 낮아짐
- 사용자 경험: 실제 API 환경에서 더 유용하고 일관된 응답을 제공하는 것으로 평가됨
- 소형 모델: 1.3B InstructGPT도 지시 따르기 면에서 175B GPT-3를 능가하는 결과가 보고됨

공개 NLP 데이터셋 결과

- TruthfulQA: 대체로 InstructGPT가 GPT-3보다 더 진실된 응답을 제공하는 경향을 보임
- RealToxicityPrompts: 존중하는 톤으로 답하라는 지시가 있을 때 독성 수준이 더 낮게 측정됨
- 독성 감소: InstructGPT는 공격적·모욕적 표현을 피하려는 경향이 더 강하게 나타남
- 편향 측정: 성별·인종 편향을 측정하는 벤치마크에서는 큰 개선이 관찰되지 않음
- 벤치마크 성능: 순수 PPO는 일부 과제에서 GPT-3보다 성능이 떨어지는 alignment tax를 유발
- PPO-ptx: 사전학습 데이터 믹스를 함께 사용해 여러 벤치마크에서 성능 하락을 상당 부분 완화
- 일부 과제: HellaSwag 등에서는 InstructGPT가 GPT-3를 능가하는 결과도 보고됨
- 요약: 정렬과 일반 벤치마크 성능 사이의 트레이드오프를 조절하는 방법을 제시한 셈

ChatGPT 공개

- 2022년 11월, OpenAI가 ChatGPT를 최초 공개함
  - GPT-3.5
- 대규모 언어모델(LLM)을 기반으로 대화형 AI 성능을 대중에게 시연
- 자연어 질의응답·요약·번역·코딩 등 다양한 작업 수행
- 사용자 친화적 인터페이스로 전 세계적 관심 급증
- 공개 직후 단기간(3~4개월)에 수백만 사용자 확보
- 생성형 AI 시대를 여는 결정적 전환점으로 평가됨

Google이 LLM을 먼저 개발해 놓고도, 대화형 모델 출시를 주저한 이유

- **평판 위험 및 윤리적 문제**: AI 모델이 잘못된 정보, 편향된 내용, 유해한 정보를 생성할 경우 발생할 수 있는 기업 이미지 실추에 대한 우려가 있었습니다. 특히, 신뢰성과 정확성을 중요하게 여기는 검색 엔진을 운영하고 있어, 통제되지 않은 챗봇 출시에 신중했습니다.
- **할루시네이션 문제**: 초기 LLM은 사실이 아닌 정보를 사실처럼 제시하는 '환각(할루시네이션)' 현상이 있었습니다. 이러한 부정확한 답변이 사용자에게 제공될 경우 문제가 발생할 수 있다고 판단했습니다. 실제로 이후 출시된 구글의 제미나이(Gemini) 모델에서 이미지 생성 오류 등이 발생하며 이러한 우려가 현실화되기도 했습니다.
- **기존 검색 비즈니스 위협**: 대화형 AI 챗봇이 사용자의 질문에 즉각적인 답변을 제공하면, 사용자가 기존의 구글 검색 엔진을 거치지 않게 되어 광고 기반의 핵심 비즈니스 모델에 타격을 줄 수 있다는 내부적인 판단이 있었습니다.
- **관료주의 및 엔지니어 이탈**: 구글의 복잡한 내부 의사결정 과정과 관료주의로 인해 신속한 제품 출시가 지연되었으며, 이로 인해 일부 핵심 엔지니어들이 OpenAI 등으로 이직하여 ChatGPT 개발에 주도적인 역할을 하기도 했습니다.

RAG

Retrieval Augmented Generation

LLM의 Hallucination 문제

- ChatGPT 공개 이후 LLM의 ‘Hallucination(환각)’ 문제가 주목됨
- 사실과 다른 내용을 그럴듯하게 생성하는 현상
- 훈련 데이터의 불완전성·확률적 언어모델 구조에서 기인
- ‘모른다’ 대신 높은 확률의 답변을 생성하려는 특성
- 법률·의료 등 고정확도 분야에서 큰 문제로 지적
- 신뢰성·안전성 향상을 위한 대응 기술 연구 가속

Hallucination 대응 기술

- 외부 지식기반을 활용하는 Retrieval-Augmented Generation(RAG)
- 모델 출력 검증을 위한 Fact-checking 모듈 도입
- RLHF로 '불확실 시 답변 보류' 성향 강화
- Domain-specific 데이터 보강 및 정제
- Self-consistency·chain-of-thought 기반 정답 검증
- 안전성 평가를 위한 벤치마크 개발 및 활용

RAG 개념 요약

- 질문에 답하기 위해 외부 문서에서 관련 정보를 먼저 검색
- 검색된 문서 기반으로 LLM이 답변을 생성하는 방식
- Retrieval + Generation 두 단계로 구성
- Hallucination 감소 효과: 사실 기반 답변 유도
- 대표적 구성요소: Retriever(검색기), Reader/Generator(생성기)
- QA 시스템의 기본 패러다임으로 자리잡음

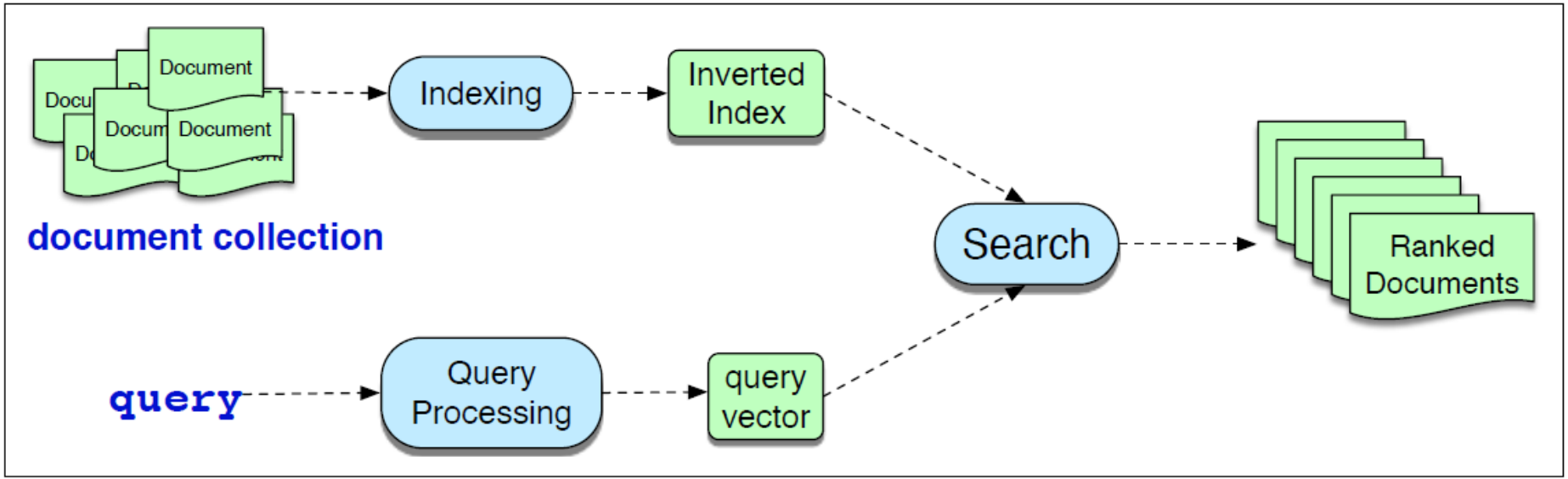


Figure 14.1 The architecture of an ad hoc IR system.

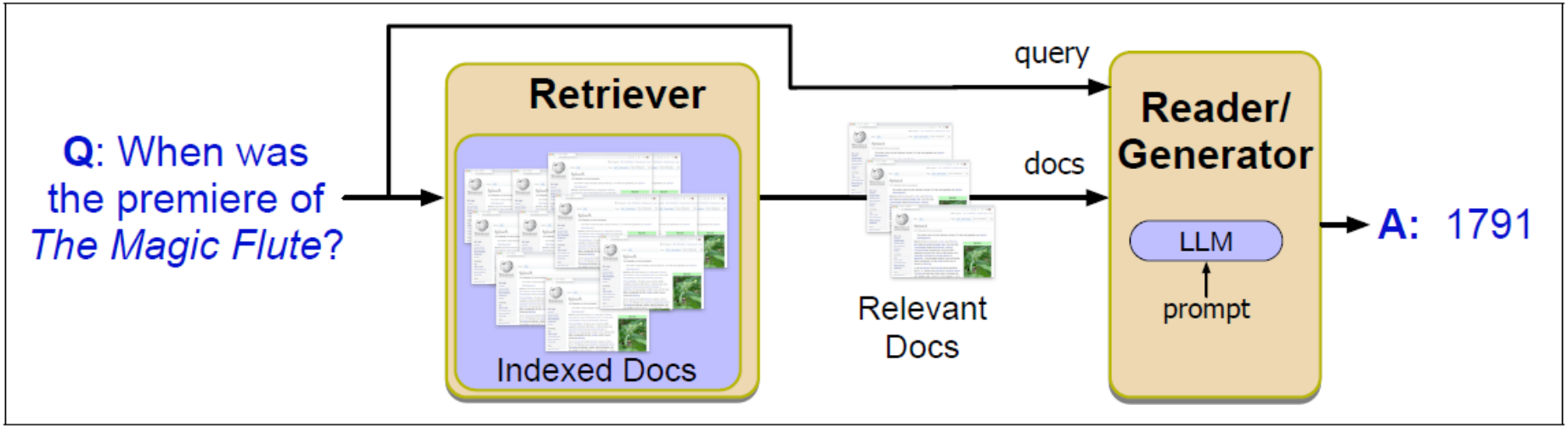


Figure 14.9 Retrieval-based question answering has two stages: **retrieval**, which returns relevant documents from the collection, and **reading**, in which an LLM **generates** answers given the documents as a prompt.

Retrieval-Augmented Generation 상세

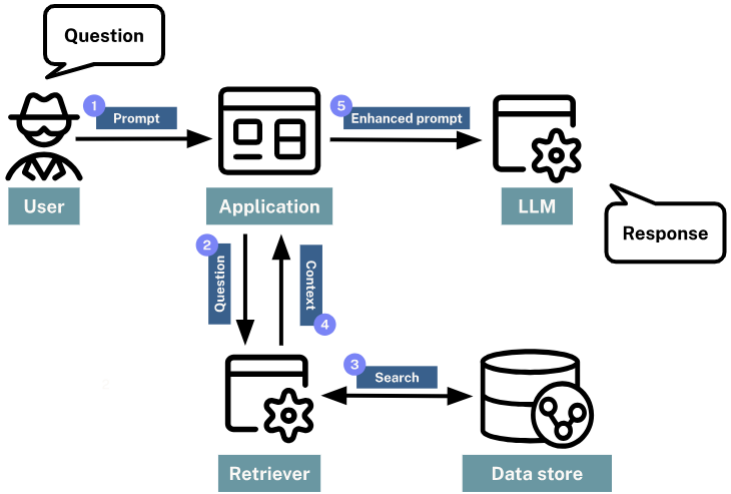
- LLM은 질의(Q)와 검색된 문서 R(q)을 함께 입력받고 답변 생성
- 단순 조건부 언어모델 $p(x_1,\ldots,x_n)$ 확장을 통해 QA 수행

$$p(x_1,\ldots,x_n) = \prod_{i=1}^{n} p([\texttt{Q:}]\,;\,q\,;\,[\texttt{A:}]\,;\,x_{<i})$$

- 검색된 문서를 prefix 형태로 prompt에 삽입
- 예: retrieved passage 1 ... n + 'Based on these texts, answer...'
- 복잡한 질문의 경우 multi-hop retrieval 필요
- Prompt engineering·문서 구분·재랭킹 등 세부 기술 중요

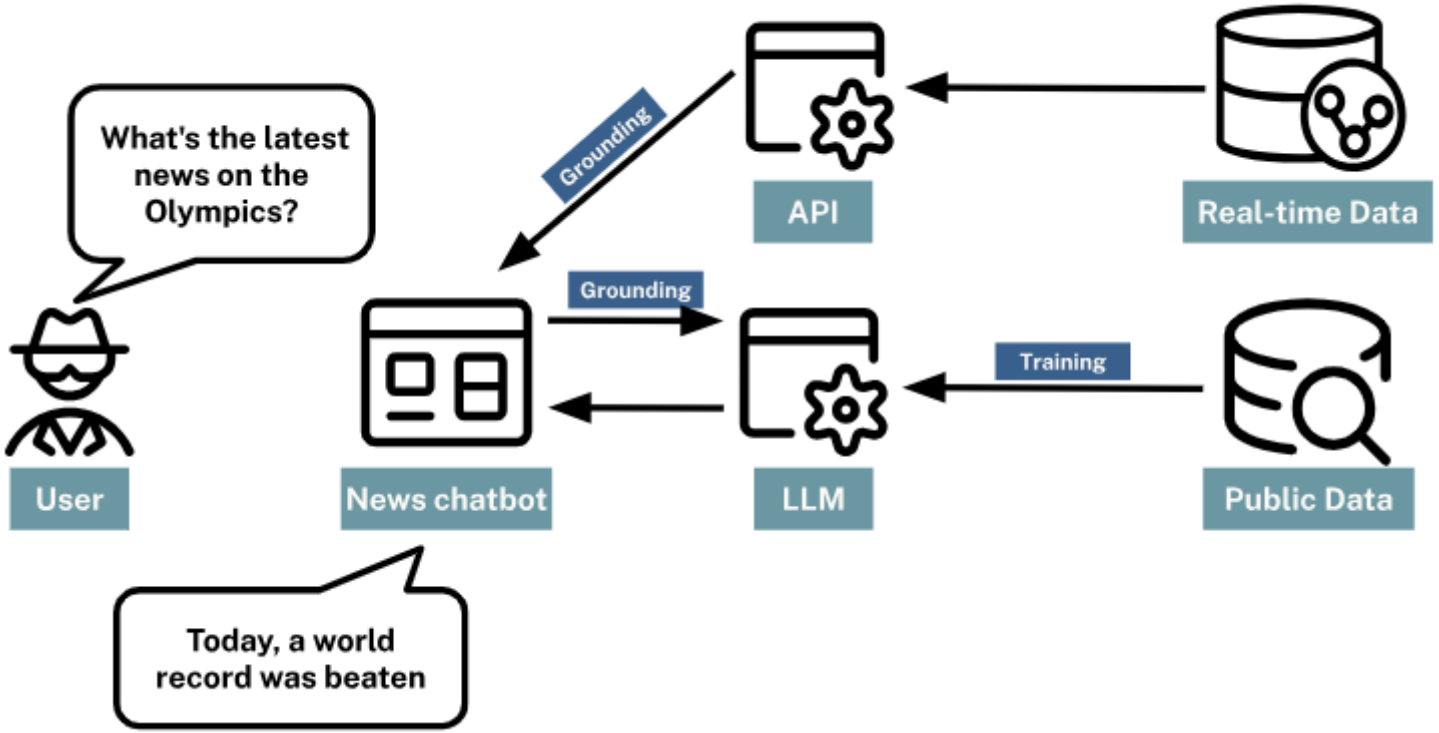
RAG 프로세스

- 1. 사용자 질의 이해: 사용자의 입력 해석 후 필요한 정보 확인
- 2. 정보 검색: 외부 데이터 소스에서 관련 정보 탐색
- 3. 응답 생성: 검색된 정보를 프롬프트에 삽입하여 더 정확한 응답 생성



RAG의 응답 특성

- RAG는 문맥적으로 적절하고 최신 정보를 반영한 응답 제공 가능
- 예: 뉴스 챗봇이 RAG를 통해 최신 올림픽 소식을 API에서 가져와 제공
- Grounding: 문맥을 제공하여 환각 가능성을 줄이고 응답 정확성 향상

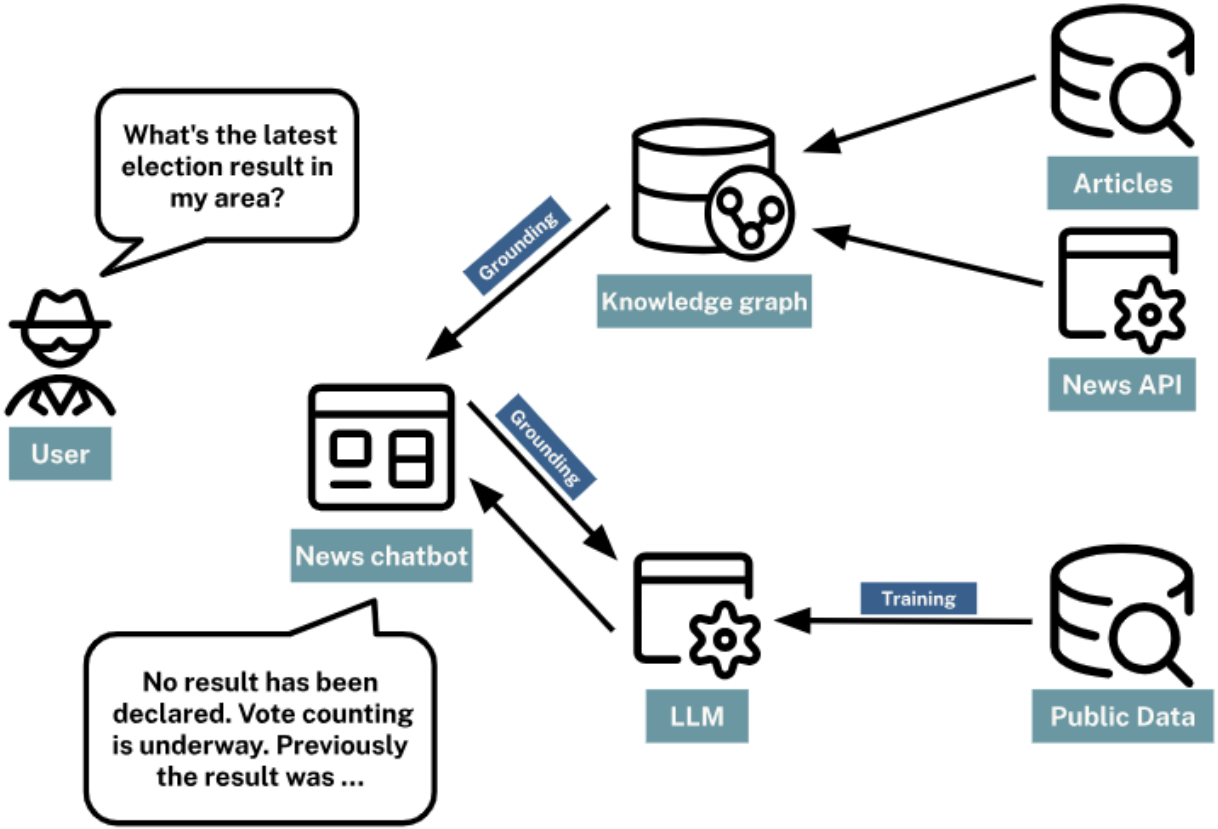

What's the latest news on the Olympics?
Grounding
API
Real-time Data
Grounding
Training
User
News chatbot
LLM
Public Data
Today, a world record was beaten

Retriever

- Retriever는 RAG의 핵심 구성 요소
- 역할: 외부 데이터 소스에서 관련 정보를 검색하여 문맥 제공
- 비정형 입력(질문 등)을 받아 구조화된 데이터를 검색
- DBMS에서의 방법:
    - 전체 텍스트 검색
    - 벡터 검색
    - Text to Cypher

데이터 소스

- RAG에서 사용하는 데이터 소스 예시:
  - • 문서 (기사, 보고서, 매뉴얼 등)
  - • API (실시간 데이터 제공)
  - • 지식 그래프 (엔티티 관계 표현)
- 뉴스 챗봇 활용 예시:
  - • 뉴스 API로 최신 기사 검색
  - • 지식 그래프로 주제 간 관계 이해
  - • 문서 데이터베이스로 텍스트 기반 답변 생성


What's the latest election result in my area?
User
Grounding
Knowledge graph
Articles
News API
News chatbot
Grounding
LLM
Training
Public Data
No result has been declared. Vote counting is underway. Previously the result was ...

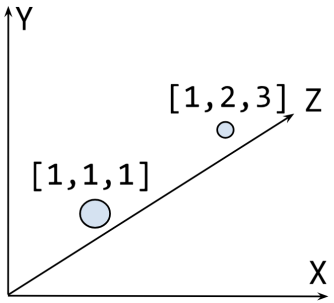
Vector RAG 개요

- 학습 목표: 의미 검색(semantic search)과 벡터 인덱스의 활용

- 의미 검색 (Semantic Search)
  - 검색 구문의 의도와 문맥적 의미를 이해하는 방식
  - 전통적 키워드 검색: 정확히 일치하는 단어나 근접성 기반 알고리즘 사용
    - 예: 'apple' 검색 → 과일 관련 결과만 반환
  - 의미 검색: 문맥 파악 → 과일, IT 회사, 기타 맥락 구분 가능
    - 결과: 검색어와 인식된 의도에 따라 맞춤형 결과 제공

벡터 (Vectors)

- 데이터를 벡터로 표현하여 의미 검색 수행
- 벡터 = 숫자의 리스트 (예: [1,2,3]) → 3차원 공간의 한 점 표현 가능
- 벡터는 텍스트, 이미지, 오디오 등 다양한 데이터 표현 가능
- 차원의 수 = 차원성(dimensionality)
  - • 차원 ↑ → 의미 포착 ↑ (계산 비용 ↑)
  - • 차원 ↓ → 빠르고 저비용 (세밀한 의미 표현 ↓)



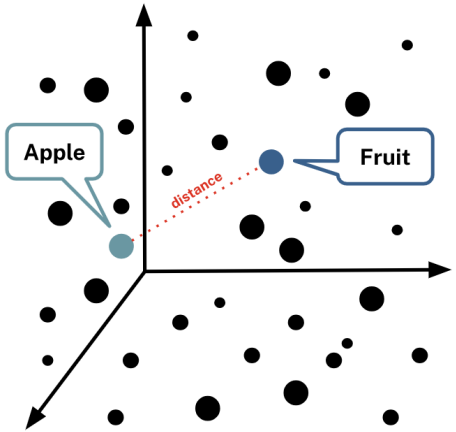
임베딩 (Embeddings)

- 벡터를 특정 작업에 유용하게 표현한 것
- 각 차원은 단어나 구의 의미적 속성을 표현
- 예: 'apple' → fruit, technology, color, taste, shape 차원 포함 가능
- 검색 맥락에서 'apple' 벡터를 다른 단어 벡터와 비교해 연관성 판단
- 임베딩 생성 방법: 임베딩 모델 활용 (예: text-embedding-ada-002)

```
apple 임베딩 예시: [0.0077, -0.0230, -0.0073, -0.0277, ...]
```

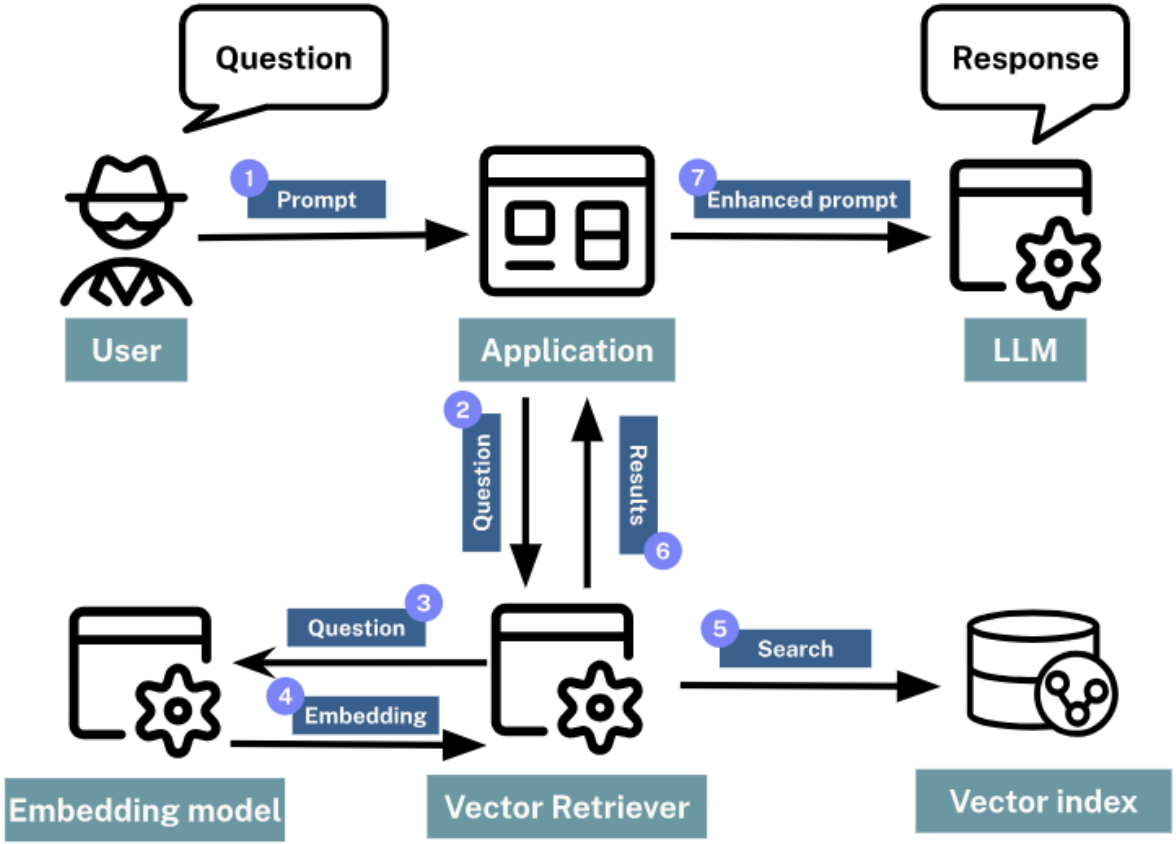
의미 검색에서 벡터 활용

- 벡터 간 거리(distance) 또는 각도(angle)를 계산하여 의미적 유사성 평가
- 유사한 의미/문맥 → 벡터가 가까움
- 무관한 의미 → 벡터가 멀리 떨어짐
- 시각적 예: Apple ↔ Fruit 벡터 간 거리 계산



Vector RAG

- 의미 검색은 Vector RAG에서 사용자 질문과 관련된 문맥적 결과 제공에 활용됨
- 임베딩 모델을 통해 소스 데이터를 벡터로 변환
- RAG 시스템의 동작 과정:
- 1. 질문 임베딩 생성
- 2. 질문 벡터 vs 인덱스 벡터 비교
- 3. 유사도 기반 점수화
- 4. 가장 관련 있는 결과를 LLM 문맥으로 제공


Question
Response
1
Prompt
7
Enhanced prompt
User
Application
LLM
2
Question
Results
6
3
Question
4
Embedding
5
Search
Embedding model
Vector Retriever
Vector index

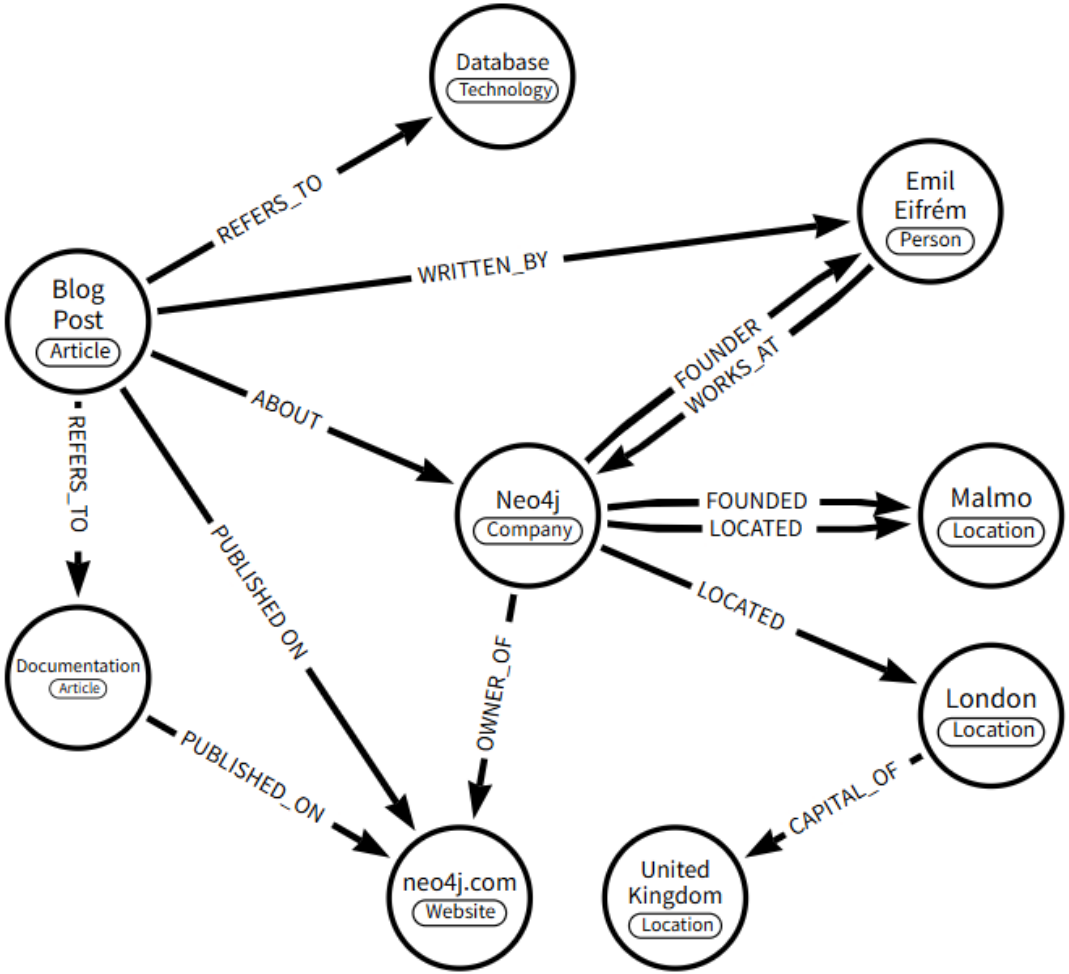
Knowledge Graphs 개요

- 지식 그래프의 정의 및 실세계 엔티티와 관계 표현 방법
- 비정형/정형 데이터 소스로부터 지식 그래프를 구축하는 과정
- 데이터 내 엔티티, 속성, 관계 식별 및 그래프 스키마 매핑 방법
- 지식 그래프 활용의 장점: 복잡한 정보의 조직, 통합, 질의

지식 그래프란?

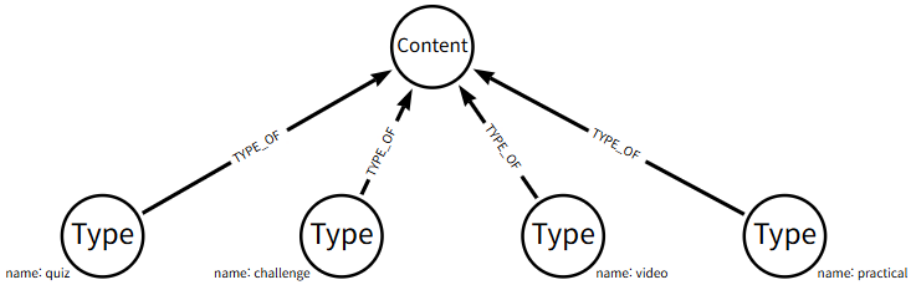
- 지식 그래프: 실세계 엔티티와 그 관계를 구조화하여 표현
- 엔티티와 속성, 관계를 체계적으로 표현 → 상호 연결된 이해 제공
- 생성형 AI에 유용: 구조적이고 연결된 데이터로 문맥·추론·정확성 향상
- 검색 엔진: 사람, 장소, 사물에 대한 정보 제공
- 다양한 소스로부터 통합 가능 → 복잡한 질의 및 분석 지원

This knowledge graph could represent Neo4j:

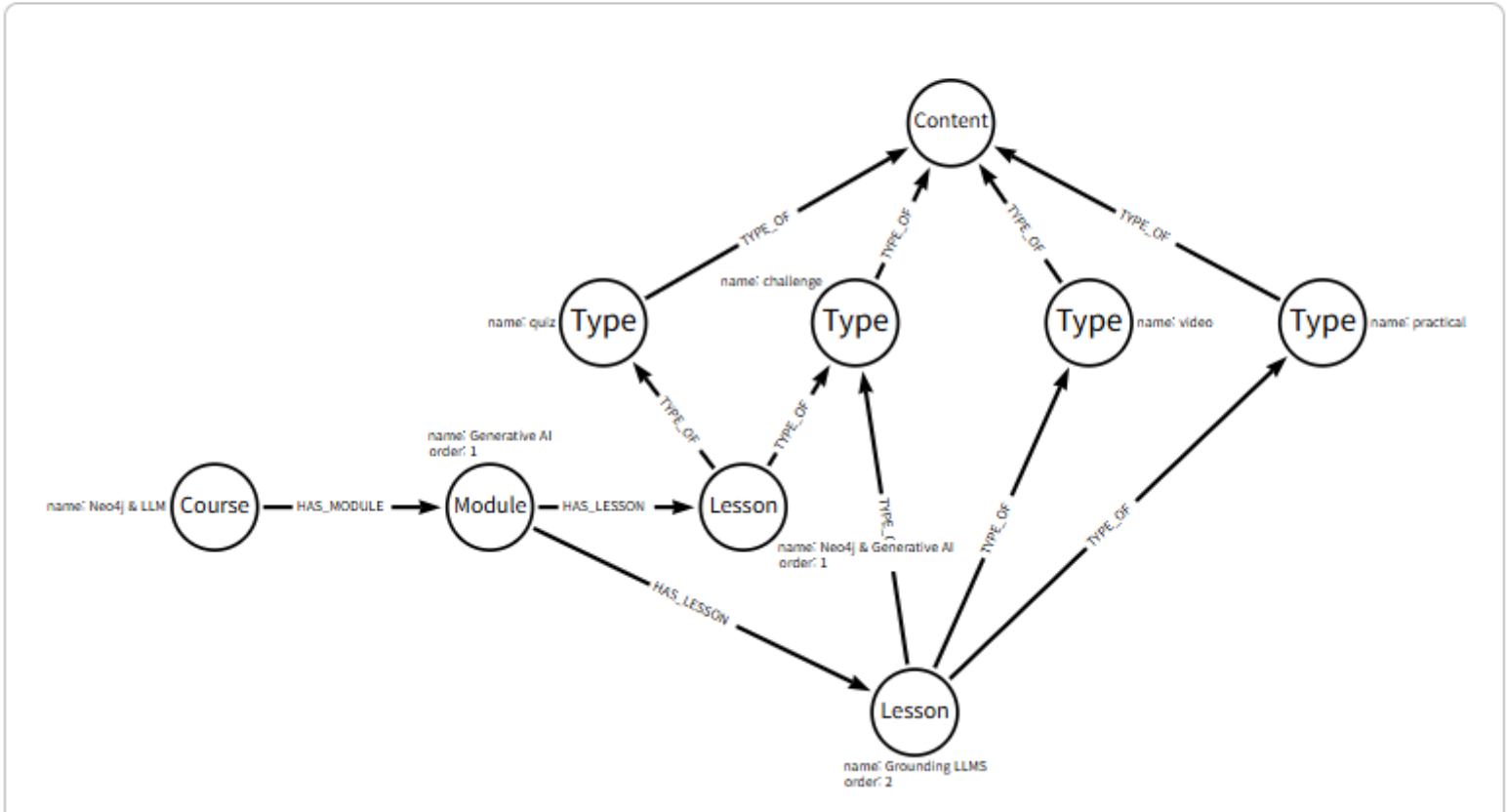


조직 원리 (Organizing Principles)

- 지식 그래프는 데이터를 구조적 원칙과 함께 저장
- 조직 원리: 데이터 구조를 부여하는 규칙 또는 범주
- 단순 데이터 설명에서 복잡한 어휘 체계까지 확장 가능
- 데이터 증가/변화에 따라 유연하게 조정 가능
- 예시: course → modules → lessons 구조 표현



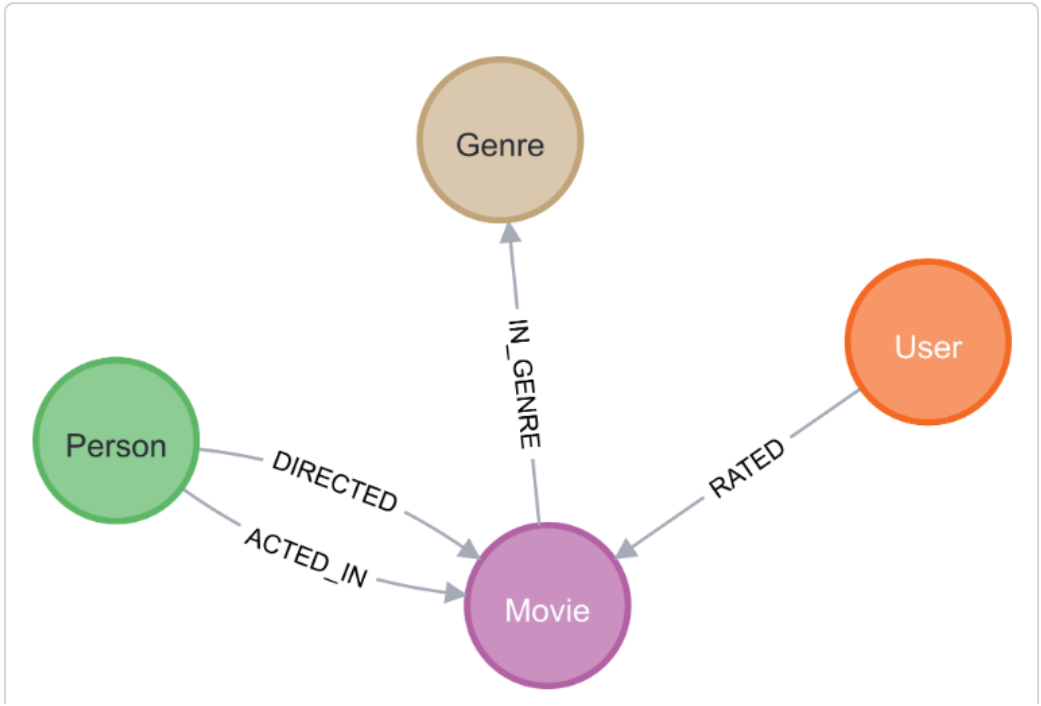
Mapping the organizing principles to the lesson content in GraphAcademy could look like this:



생성형 AI와 지식 그래프

- 생성형 AI에서 지식 그래프는 도메인 지식의 조직·활용에 기여
- 비정형 데이터도 통합 가능
- GraphRAG에서 문맥 제공 및 grounding 기반 제공
- 더 정확하고 설명 가능한 응답 제공 가능

The graph also contains `User` nodes and over 100,000 movies ratings. `User` nodes are connected to `Movie` nodes by a `RATED` relationship.



Vector Indexes 개요

- Neo4j (Graph database)에서 벡터 인덱스를 사용하여 유사 데이터를 찾는 방법 학습
- Movie Plots 예시
  - Neo4j 영화 추천 샌드박스: 9000개 영화, 15000명 배우, 100000개 이상 사용자 평점 포함
  - 각 영화에는 .plot 속성이 존재

```
MATCH (m:Movie {title: "Toy Story"})
RETURN m.title AS title, m.plot AS plot
```

Plot Embeddings

- 1000개 영화 줄거리 임베딩 생성됨
- .plotEmbedding 속성에 저장
- 다음 Cypher 쿼리로 임베딩이 있는 영화 반환 가능

```
MATCH (m:Movie)
WHERE m.plotEmbedding IS NOT NULL
RETURN m.title, m.plot
```

Vector Index 생성

- moviePlots 벡터 인덱스가 .plotEmbedding 속성에 대해 생성됨
- 이 인덱스를 통해 영화 줄거리 임베딩을 비교해 가장 유사한 영화 검색 가능

Vector Index 쿼리

- db.index.vector.queryNodes() 프로시저 사용
- 3가지 주요 매개변수:
  - • indexName - 벡터 인덱스 이름
  - • numberOfNearestNeighbours - 반환할 결과 수
  - • query - 임베딩을 나타내는 float 리스트
- 출력: node(일치하는 노드), score(유사도 점수, 0.0 ~ 1.0)

```
CALL db.index.vector.queryNodes(
  indexName :: STRING,
  numberOfNearestNeighbours :: INTEGER,
  query :: LIST<FLOAT>
) YIELD node, score
```

유사 영화 줄거리 검색

- Toy Story와 가장 유사한 줄거리 검색 예시
- db.index.vector.queryNodes() 활용
- 결과는 유사도 점수 순으로 정렬됨 (0.0~1.0, 1.0 = 가장 유사)

```
MATCH (m:Movie {title: 'Toy Story'})
CALL db.index.vector.queryNodes('moviePlots', 6, m.
plotEmbedding)
YIELD node, score
RETURN node.title AS title, node.plot AS plot, scor
e
```

임베딩 생성

- Cypher에서 genai.vector.encode 함수로 새로운 임베딩 생성 가능
- OpenAI API key 필요

```
WITH genai.vector.encode(
  "Text to create embeddings for",
  "OpenAI",
  { token: "sk-..." }) AS embedding
RETURN embedding
```

Plot Embedding 생성 예시

- 예: "A mysterious spaceship lands Earth" 텍스트를 임베딩 생성 후 moviePlots 인덱스 질의
- 6개의 가장 유사한 영화 줄거리 반환

```
WITH genai.vector.encode(
  "A mysterious spaceship lands Earth",
  "OpenAI",
  { token: "sk-..." }) AS myMoviePlot
CALL db.index.vector.queryNodes('moviePlots', 6, my
MoviePlot)
YIELD node, score
RETURN node.title, node.plot, score
```

GraphRAG 개요

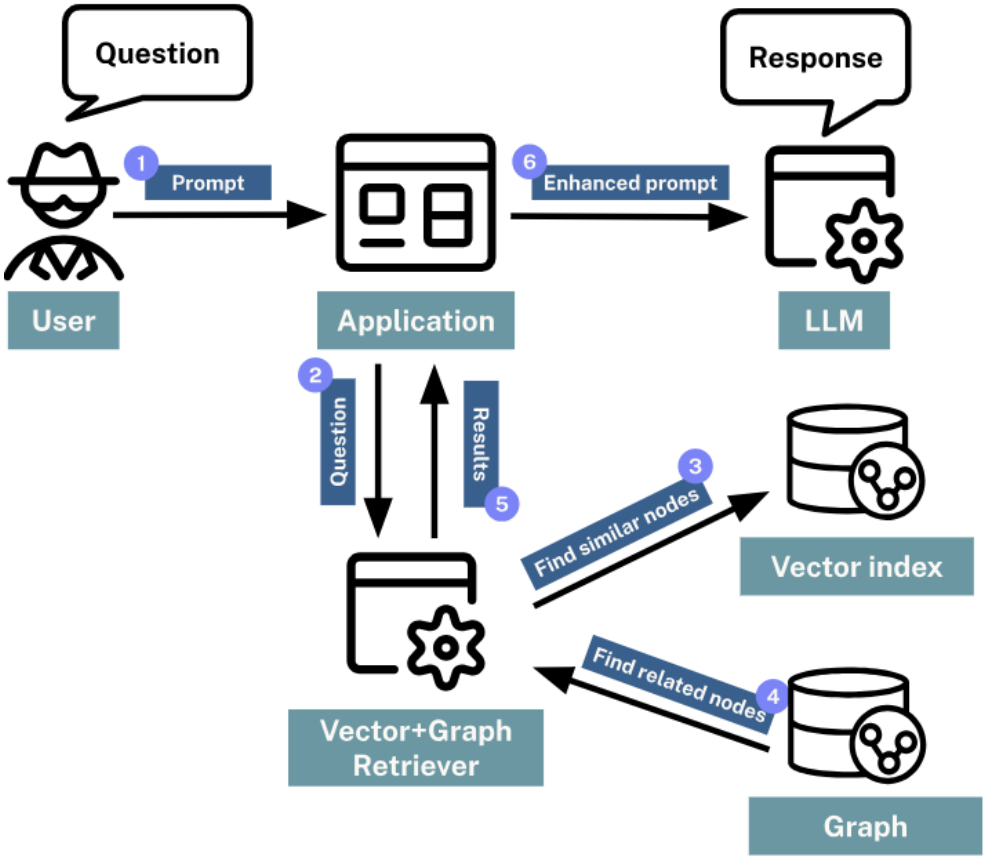
- GraphRAG(Graph Retrieval Augmented Generation): 그래프 DB의 강점을 활용하여 LLM에 유용한 문맥 제공
- Vector RAG와 함께 사용 가능
- Vector RAG: 임베딩 기반 문맥적 정보 검색
- GraphRAG: 그래프 관계와 구조 활용하여 정보 검색 강화

GraphRAG의 장점

- Richer Context: 엔티티 간 관계를 포착하여 더 풍부한 정보 검색
- Improved Accuracy: 벡터 유사도 + 그래프 탐색으로 더 정밀한 결과
- Explainability: 그래프 경로와 연결을 통해 결과 해석 용이
- Flexible Queries: 전체 텍스트, 벡터, Text-to-Cypher 등 복합 질의 지원
- Enhanced Reasoning: 데이터 추론 가능, 추천·지식 발굴에 활용

Graph-Enhanced Vector Search

- 벡터 검색 + 그래프 탐색을 결합하여 의미 기반 문서 검색
- 프로세스:
- 1. 사용자 질의 입력
- 2. 벡터 검색으로 유사 노드 탐색
- 3. 그래프 탐색으로 관련 노드/엔티티 확장
- 4. 엔티티/관계를 LLM 문맥에 추가
- 5. 관련 데이터는 질의와의 연관성 점수화 가능


Question
Response
1
Prompt
6
Enhanced prompt
User
Application
LLM
2
Question
Results
5
3
Find similar nodes
Vector index
Find related nodes
4
Vector+Graph Retriever
Graph

예제: 영화 줄거리 Graph Enhanced Search

- 그래프 탐색을 통해 벡터 검색을 확장하는 방법:
  - 관련 배우, 감독, 장르 추가
  - 유사 테마/연결성을 가진 영화 탐색
  - 사용자 평점을 활용해 결과 필터링/순위화

```
// 영화 줄거리 벡터 검색 + 그래프 탐색
WITH genai.vector.encode(
  "A mysterious spaceship lands Earth",
  "OpenAI",
  { token: "sk-..." }) AS myMoviePlot
CALL db.index.vector.queryNodes('moviePlots', 6, myMovieP
lot)
YIELD node, score
MATCH (node)<-[r:RATED]-()
RETURN node.title, node.plot, score, avg(r.rating) as use
rRating
ORDER BY userRating DESC
```

과제#5 RAG를 이용한 챗봇 만들기

- 포함할 기능
  - Input-output interface 구현
  - LLM 연결
    - Open LMMs: Llama 계열, Mistral, … 또는
    - API 연결: ChatGPT, Gemini, …
  - RAG 기능
    - 간단한 문서 저장 및 추출 기능
    - Prompt에 호출 문서 포함시켜 LLM에 입력
  - 이외 기능은 자유로이 추가해도 무방
- 제출 방법
  - 소스 코드와 보고서(동작 스크린샷 포함)를 이메일로 제출
  - 기한: 12/5일 까지

감사합니다