AI Day 11 정리(Mastering AI Agent)

✅ 학습목표

- **AI Agent의 정의 및** 핵심 **특성** 이해
- 대규모 언어 모델의 고정된 매개변수 메모리 **한계** → 에이전트형 **RAG 역할** (극복해결책)
- **LLM 에이전트** 추론 및 계획 능력을 **강화**하는 **프롬프트 기법** 적용
- 다중 에어전트 시스템의 필요성, 협업 유형, 커뮤니케이션 구조 등 **에이전트 시스템의 원리** 이해
- 에이전트 행동 정책을 형성하기 위한 **강화 학습 기반 훈련 방법**과 **표준화된 프로토콜 역할**

1. AI Agent

▼ 서론

- 대형 언어 모델(LLM)이란 무엇인가
  - [1980년대] Language Model (LM): 텍스트에서 단어, 부분 단어, 형태소 등의 의미있는 단위인 토큰의 순차적 사용을 모델링 하는 확률 모델
    - 이전 토큰들을 기반으로 다음 토큰의 가능성을 예측
    - 새로운 텍스트 시퀀스를 생성
  - [2017년] Transformer Architecture: 기존의 1) 긴 문장 학습 능력 제한, 2) 높은 훈련 비용을 획기적으로 개선한 모델
    - 긴 문장에서 선택적으로 집중
    - 대규모 사전 학습에 용이
  - [2018년~] Large Language Model (LLM): 대규모 데이터 (예, Book, Wikipedia)를 활용하여 언어적 지식과 패턴을 학습

        - 마스킹 언어 모델링 (Masked Language Modeling, MLM)
        - 다음 토큰 예측 (Next Token Prediction, NTP)
    - [2022년~] LLM의 확장과 진화
        - Multi-Modal Large Language Model (MLLM): LLM 기반 모델로 텍스트와 시각, 음성 등의 모달리티를 결합하여 작업할 수 있는 모델
        - Agentic AI: LLM이 단순한 '응답 생성 모델'을 넘어, 스스로 목표를 설정하고 계획, 실행하는 자율적 인공지능 (Agent)으로 진화

1. GPT 3.5(2022.11)
   범용 LLM (Generalist LLM)
    - 모든 도메인에 대해서 일정 수준 이상의 응답이 가능한 대형언어 모델
    - 가정의학과 전문의처럼, 넓은 지식 기반을 가지고 대부분의 문제에 1차 대응 가능
2. GPT 5(2025.08)
   의료 특화 LLM (Medical LLM)
    - 의료 지식에 특화되어 있으며 진단, 처방, 환자 응대 등에 전문화된 모델
    - 내과, 외과 등의 전문의처럼, 의료 특정 도메인에 깊이 있는 지식을 갖춘 LLM
3. Agentic Clinician AI(의료진 에이전트형 LLM)
    a. 같은 병이라도 의료진 마다 진단·치료 방식은 다름
    b. 환자 상황, 병원 환경, 의료진의 임상 경험, 언어 습관까지→ '의료진 개인'의 판단 맥락을 반영할 수 있어야 함
    c. 의료진 에이전트 AI는 단순 지식 제공을 넘어 의료진의 사고 흐름과 의사결정 과정을 능동적으로 재현→ 의료진 본인의 '디지털 트윈'처럼 동작
    d. 예시 : 의학 영상 판독, 의료 수술 보조 로봇

AI Agent의 개념과 특성

1. Introduction to AI Agent

- AI Agent(AI 에이전트)
    - 개념: AI를 사용해 목표를 추구하는 소프트웨어 시스템
    - 주요특성 : 계획, 실행, 결정을 통해 인간 개입 없이 자율적 문제 해결

- AI Agent 기술의 도전과제 : 보완, 실무 응용, 컨텍스트 관리 개선(학술연구) 등

  ▼ 산업 현장 예시

  - 삼성 SDS : AI 서비스 플랫폼 FabriX
    - 기업 내부 지식(문서) 학습 → AI Agent가 협업하도록 설계
    - 형태 : LLM + Copilot + Automation
  - MS : 365 Copilot

- 인공지능 발달 단계(위에서 아래로)

> **Perception AI(지각 AI)**
> 특화된 분야에서 해석 및 분석
> 현대 기계학습의 기반

> **Generative AI(생성형 AI)**
> 콘텐츠 제작(텍스트, 이미지 등)
> "the Creative Powerhouse"

> **Agentic AI (에이전트형 AI)**
> 능동적인 작업 수행
> "자율형 문제해결 시스템"

> **Physical AI (물리적 AI)**
> 현실 세계에서 작동
> AI 시스템이 물리적 실체를 얻음

- 에이전트
  - 다른 사람 또는 단체를 대신하여 행동하는 개인 또는 단체
  - 특정 작업을 수행하고, 결정을 내리며, 합의를 도출하는 것을 목표로 함
- AI 에이전트
  - AI를 사용하여 사용자를 대신해 목표를 추구하고 작업을 수행하는 소프트웨어 시스템
  - 작업 → **의사 결정(AI가 결정)** → 상황 인식
- AI VS AI Agent (구분 명확 X)
  - 자율성
    - AI : **사전에 설정된 의사결정** 과정 없이는 이러한 작업을 수행할 수없음

        - AI 에이전트 : **인간의 개입없이** 계획, 실행, 결정하여 **문제를 자율적으로 해결**할 수 있음
    - 인식
        - AI 에이전트 : 행동하기 전에 **외부 데이터를 감지하여 환경을 인식하고 해석**할 수 있음
    - 기억
        - AI 에이전트 : **기억에 따라 상황을 처리**하고 점점 더 정교한 도움을 제공할 수 있음
    - 추론(계획)
        - AI 에이전트 : **목표를 여러 개의 작은 작업들로 나누고**, 실행하기 전에 **합리적인 행동 계획을 구축**할 수 있음
    - 학습(적용)
        - AI 에이전트 : **시간이 지남에 따라 피드백 및 출력을 학습하여** 성능을 향상시키고 개인화함
    - 동작(도구)
        - AI 에이전트 : 기능 호출을 통해 외부 도구를 사용하여 작업을 처음부터 끝까지 완료할 수 있음
- Challenges in AI Agent
    - 학술 연구 영역

      ➡️
      - 다중 에이전트 협업 시스템
      - 메모리와 컨텍스트 관리
      - 작업 관리
      - 확장성과 성능
    - 실무 응용 영역

➡️

- 보안 및 개인정보 문제
- 자율성의 고위험 요소
- 추론 비용과 실시간 협업
- 윤리적 문제와 편향성에 대한 우려

---

2. Multi-Agent System

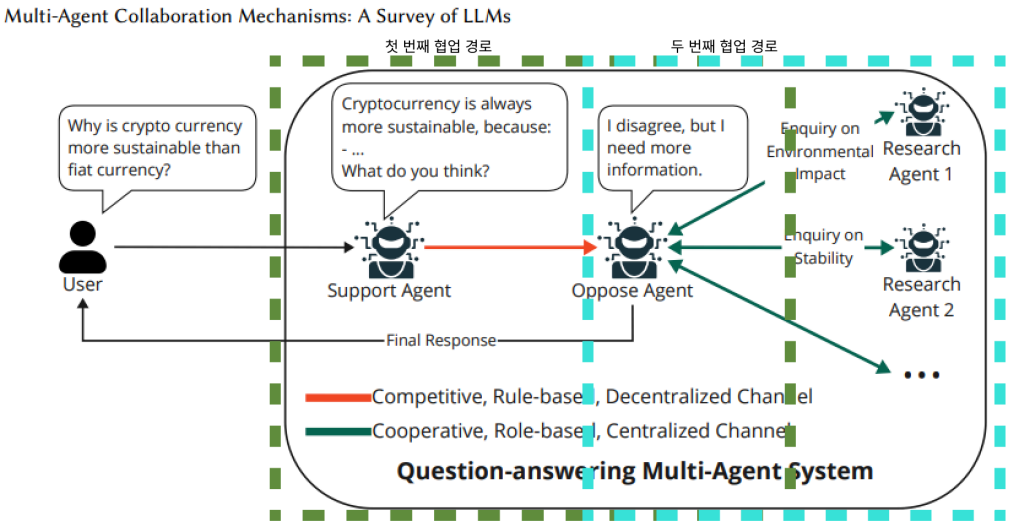
학습시작

- 대규모 언어 모델의 한계 : 환각 현상, 느린 추론 등 LLM의 본질적 한계
- 다중 에이전트 시스템 구성 : 다중 에이전트 시스템의 주요 구성 요소와 다중 에이전트 시스템 문제 정의
- 에이전트 협업 전략 : 협업 유형 및 전략과 커뮤니케이션 구조

1. Multi Agent System

- 대규모 언어 모델(LLM)
  - 최신 성과 : 창의적 글쓰기, 논리적 사고, 판단 능력 ← 인간 수준에 필적하는 성능
- 개별 LLM의 본질적 한계
  - **환각 현상**: 부정확하거나 조작된 정보 생성
  - **자기회귀적 성격**: 천천히 생각하거나 신중한 추론을 할 수 없음
  - **스케일링 법칙**: 모델 크기 증가에 따른 성능 향상 둔화
- **왜 다중 에이전트 시스템이 필요한가?**
  - **집단 지성**
    - 지능형 에이전트들이 팀을 이루어 협력하고 지식을 나누며, 함께 문제를 해결하는 능력에 초점을 맞춤 **(수평적 확장)**
    - 인간 사회는 일상 업무부터 과학 연구까지 팀워크와 역할 분담을 통해 공통 목표를 달성하는 데 탁월
    - 다중 에이전트 시스템도 이와 같은 원리를 적용하여, AI 에이전트들이 서로의 장점과 시각을 결합해 효율적으로 협업하도록 함

→ 개별 에이전트들의 능력의 총합을 뛰어넘는 진정한 집단 지성을 실현하고자 함



Multi-Agent Collaboration Mechanisms: A Survey of LLMs
LLM 기반 다중 에이전트 협업 시스템의 질의응답 활용 예시 (강의자료 아님, 논문 스크린샷)

- LLM 기반 Multi Agent System(다중 에이전트 시스템)
  - 협업 요소와 메커니즘 : 협업 유형, 전략, 소통 및 조직 구조
- 시사점 및 남은 문제들
  - 집단적 추론, 의사결정 등 다중 에이전트 시스템의 발전과정에서 아직 해결해야할 문제들이 존재
- 다중 에이전트 시스템의 주요 구성 요소
  - 에이전트 : 역할, 능력, 행동, 지식 모델을 가진 핵심 행위자들
  - 환경 : 에이전트가 존재하고 인식 및 행동할 수 있는 외부 세계
  - 상호작용 : 표준화된 통신 언어를 사용한 에이전트 간 소통
  - 조직 : 에이전트들은 계층 구조로 관리되거나 자발적인 행동으로 조직

---

2. 방법론 : 협업 유형

- 협업 유형
  1. 협업, 역할 분담, 문서 품질 향상
  2. 경쟁, 대립, 논증
  3. 협력 + 경쟁, 균형 잡힌 판단

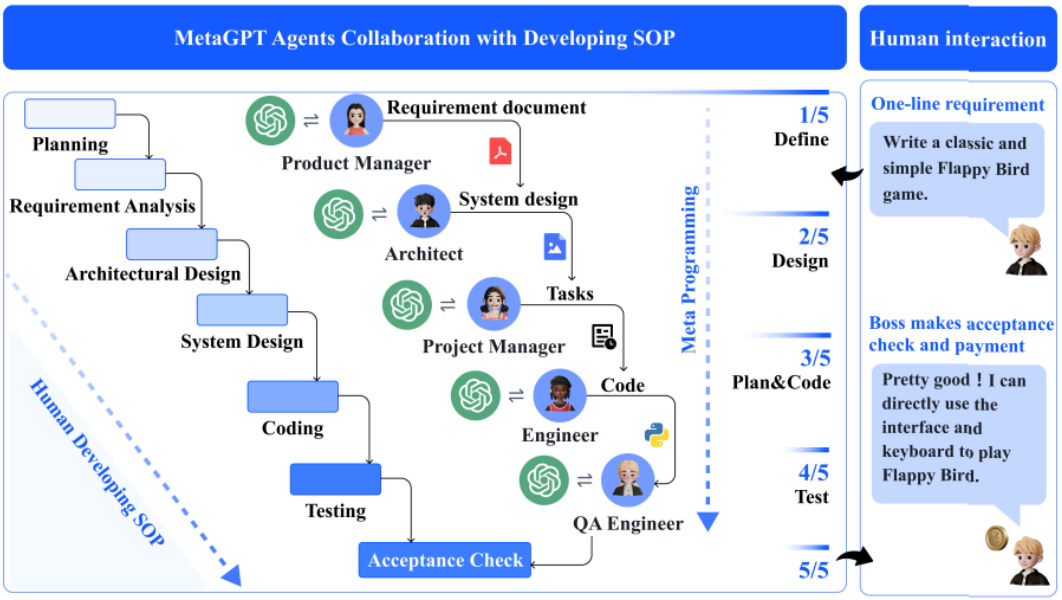
- 협력
  - 에이전트들은 각자의 개별 목표를 공동의 집단 목표와 일치시켜, 상호 이익이 되는 결과를 달성하기 위해 협력
  - 에이전트들은 각자의 전문 분야 내에서 특정 세부 작업에 집중하도록 활용할 수도 있음
  - 에이전트 간 빈번한 통신과 다중 협업 채널은 계산 비용과 복잡성 증가를 초래할 수 있음
  - 개별 에이전트의 신뢰성과 성능에 크게 좌우되어, 한 개 이상의 에이전트의 오류가 시스템 전체에 부정적 영향을 미칠 수 있음 (e.g., 무한 대화 루프, 환각 현상 증폭).
- 경쟁
  - 에이전트들은 각자의 개별 목표를 우선시하며, 이는 다른 에이전트들의 목표와 충돌하거나 대립할 수 있는 경쟁 요소를 도입
  - 에이전트들이 고도의 추론 능력과 더 창의적인 문제해결 방법을 개발하도록 촉진
  - 각 에이전트의 능력의 한계를 시험함으로써 시스템의 적응성을 강화
  - 경쟁적 협업 경로는 강력한 프롬프트를 가진 단일 에이전트에 의해 따라잡힐 수 있음
  - 끝나지 않는 논쟁
- 협력과 경쟁의 조합
  - 협력과 경쟁의 전략적 조합
    - 에이전트들이 공동 목표 달성을 위해 특정 작업에서는 협력하면서 동시에 다른 작업에서는 경쟁할 수 있도록 함
  - 전문가 혼합 모델(Mixture-of-Experts)
    - 다양한 전문가 모델들이 최종 결과물에 기여하려고 경쟁하고, 게이트 시스템이 각 입력에 맞는 최적의 전문가를 골라냄
    - 전문가들 사이의 협력 및 경쟁적 관계는 모델 학습 과정에서 먼저 형성되며, 이때 각자가 데이터의 서로 다른 측면에 대해 전문화하도록 훈련

3. 방법론 : 협업 전략

- 규칙 기반 프로토콜
  - 사전에 정의된 규칙 : 에이전트 간 상호작용은 미리 정의된 규칙에 의해 엄격히 통제

- 인간의 협력 방식 모방
  - 동료 검토에서 영감을 받은 협업 메커니즘은 사전 정의된 규칙을 사용하여 에이전트들이 서로의 출력을 비판하고, 수정하며, 개선할 수 있도록 함
  - 규칙 기반 전략은 시스템 동작과 특정 규칙을 쉽게 연결할 수 있으므로 구현과 디버깅을 용이하게 함
  - 합의 도출, 경로 찾기와 같이 절차가 명확히 정의되고 변동성이 적은 작업에 특히 효율적

- 역할 기반 프로토콜
  - 사전에 정의된 역할 : 세분화된 목표 하에서 각 에이전트가 고유한 역할 분담이나 업무 분할을 통해 작동하도록 함
  - 주로 도메인 지식에 기반
    - 각 에이전트에 특정 책임을 할당하여 인간과 유사한 협업을 시뮬레이션하고, 역할 준수를 통해 일치성 강화
    - 각 에이전트의 역할은 전문가 수준의 지식으로 정의되어, 에이전트는 서로의 결과를 검증할 수 있는 전문가의 역할을 할 수 있음
    - 개별 모듈의 재활용 가능성



https://arxiv.org/abs/2308.00352

MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework

4. 방법론 : 커뮤니케이션 구조

- 중앙집중형 구조
  - 참여자-서비스 제공자 구조 : 서비스 에이전트는 시스템 안에서 참여자들 사이의 소통이나 협력을 관리, 제어, 조율하는 역할을 담당
  - 중앙 에이전트
    - LLM-블렌더는 한 라운드에서 서로 다른 LLM들을 호출하고, 두 개씩 비교하여 순위를 책정한 후, 최상위 응답들을 결합
    - 판사 에이전트(토론), 최종 의사결정을 위한 의사(MDT 프로세스)
- 탈중앙화 구조
  - 에이전트에 대한 통제 및 의사결정 분산
    - 각 에이전트는 에이전트와의 제한적인 통신과 일부의 정보를 바탕으로 작동하므로, 상호작용과 의사결정을 위한 정교한 알고리즘이 필요
    - 에이전트들이 분산형으로 서로 직접 소통하는 방식으로 작동
    - 일부 에이전트가 실패해도 시스템이 계속 기능할 수 있음
    - 높은 확장성을 제공하며, 에이전튼들이 자율적으로 작동하고 시스템 변화에 적응할 수 있음

- 다중 에이전트 시스템에 대한 우려의 시사점
  - 확장성 고려사항
    - 성능 저하 없이 더 큰 에이전트 네트워크를 처리하기 위해서는 확장 가능한 아키텍처와 알고리즘 구현이 필수적
  - 최적 협업 전략
    - 효과적인 협업을 위해서는 작업 요구사항에 맞는 최적의 협업 전략을 선택하는 것이 주요
  - 적응가능한 역할 및 협업 경로 할당
    - 적응가능성은 시스템이 변화하는 환경과 목표에 효과적으로 대응할 수 있도록 함

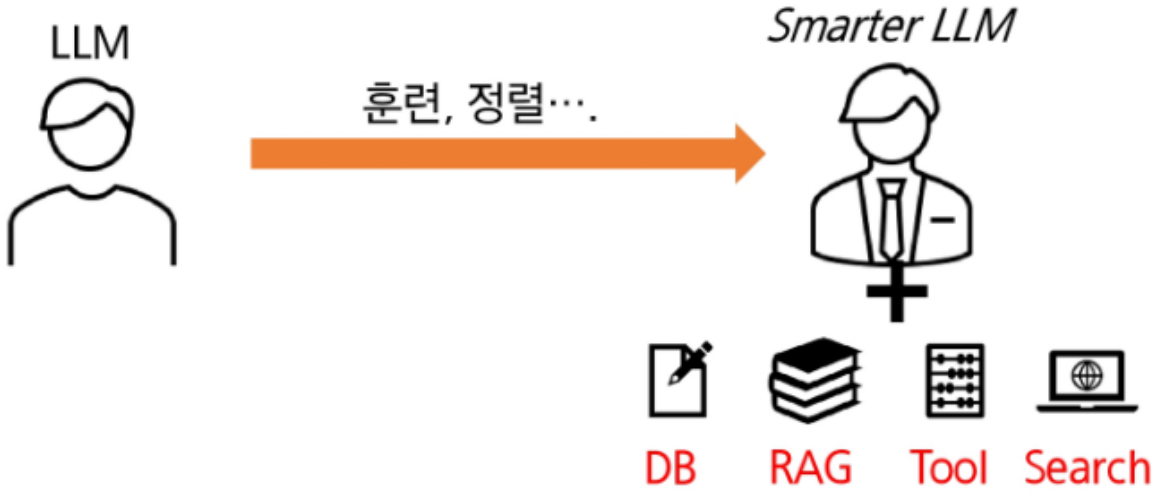
---

3. 다중 에이전트 시스템에서의 메모리 및 도구

전략적 도구 사용 정책 학습 및 MCP 프로토콜의 역할

1. Memory & Tool in Multi Agent System

• AI는 언어 모델(LLM)에서, 독립적 사고와 행동을 할 수 있는 자율적인 "AI 에이전트"의 **새로운 패러다임으로 발전**하고 있음
• **LLM** : 고정된 매개변수 메모리에 제한을 받아 수동적임
• **AI 에이전트** : 실시간 정보에 접근, 외부 도구를 활용, 물리적 또는 디지털 세계와 상호작용 → 환경을 인지하고, 계획을 수립하며, 특정 목표 달성을 위해 구체적인 행동을 실행할 수 있는 **능동적 개체**



- **패러다임의 전환 : 언어 모델에서 에이전트 시스템으로!**
  - **기억 / 지식** : 에이전트가 최신 정보를 어떻게 습득하는가? **(RAG, 메모리)**
  - **행동 / 능동성** : 에이전트가 구체적인 업무를 어떻게 완수하는가? **(도구 호출)**
  - **학습 / 적용** : 에이전트들이, 특히 공동 작업 시, 어떻게 지능을 향상시키는가? **(멀티 에이전트 학습)**

- 메모리 & 도구
  - **LLM** 내장 매개변수 메모리의 **한계**

    : 최신 정보를 반영할 수 없음, 잘못된 내용을 생성하는 경향이 있음, 기업 내부 정보를 검색할 수 없음

    → 이런 제약들로 인해 실제 핵심 업무에서 LLM을 믿고 사용하기 어려움

- **RAG (검색 증강 생성 (Retrieval-Augmented Generation))**
  - 핵심 아이디어 : LLM이 답변을 생성하기 전에 신뢰할 수 있는 외부 소스들을 이용하여 응답 구성

    → ex) 데이터베이스, 최신 웹 문서
  - 장점 : LLM의 자연스러운 표현력과 외부 지식의 정확성을 결합하여 신뢰할 수 있는 답변 생성

- 일반적 검색 증강 생성(RAG)의 한계점
  - **지식 소스의 한정성** — 보통 하나의 외부 데이터베이스만 사용하기 때문에, 여러 소스

    (예: 웹 검색, API 등)를 동시에 활용하기 어렵다.
  - **정보 검증 부재** — 검색된 내용을 한 번만 가져와 바로 사용하므로, 정보의 신뢰도나 정확성을 검토하지 않는다.
  - **복잡한 추론 한계** — 상태 정보를 저장하지 않아 다단계 추론이나 연속적인 탐구 작업에 적합하지 않다.
  - **맥락 유지 불가** — 이전 대화나 검색 맥락을 기억하지 못해, 마치 "기억하지 못하는 검색 엔진"처럼 동작한다.

  → RAG + 에이전트 = 에이전트 RAG

- **에이전트형 RAG란?**
  - AI 에이전트가 RAG의 각 단계를 조율하며 단순한 검색·생성을 넘어 **도구 활용과 판단 기능**을 수행하는 방식

    → 벡터 검색, 웹 검색, API 등 다양한 도구를 사용해 **검색 여부 판단·도구 선택, 쿼리 생성, 결과 평가**까지 수행

- **AI 에이전트의 주요 구성 요소**
  - **LLM** (특정 역할과 작업을 담당)
  - **메모리** (단기 기억 및 장기 기억)
  - **계획** (예 : 자기 성찰, 자기 비판, 쿼리 분배 등)
  - **도구** (예 : 계산기, 웹 검색 등)

- **멀티 에이전트 시스템을 위한 훈련 방법**

  —> 멀티 에이전트 시스템 훈련 핵심은 "언제, 무엇을, 어떻게"를 결정하는 것임
  - **언제:** 자체 지식으로 답할지, 도구를 쓸지 판단 (정확도 vs. 비용·지연시간의 균형)
  - **무엇을:** 상황에 맞는 도구 선택 (예: 계산기, 웹 검색 등)
  - **어떻게:** 선택한 도구에 올바른 인수를 전달

    → **결과나 오류를 분석해 다음 행동을 스스로 결정**

  —> 이러한 의사 결정 단계들이 모여서 에이전트의 행동 정책(policy) 형성

---

ex 1) Toolformer : 언어모델이 스스로 도구 사용법을 학습
- 소규모 API 호출 데모 세트로 시작
- 언어모델이 새로운 맥락에 API 호출 삽입
- 호출 실행, 응답 검색, 유용성에 따른 필터링, 언어모델 미세조정

ex 2) Gorilla : LLM을 대규모 API와 연결
- **접근 방식:**
  - APIBench 데이터셋을 활용해 **LLaMA·GPT 모델을 미세조정**
  - **RAFTA**(검색 증강 미세조정)로 API 호출 전 관련 문서를 검색하여 활용
- **주요 기능:**
  - 수십만 개의 API를 **정확한 형식으로 처리**
  - 문서 검색 + 지도 학습(SFT)을 통해 **잘못된(API 환각) 호출을 감소**시킴

ex 3) RetTool : LLM의 전략적 도구 사용을 위한 강화학습

—> 코드 인터프리터 등 외부 도구 사용 시점·선택·방법을 학습시켜 복잡한 계산 및 추론 작업에서 LLM의 성능 향상을 위함

- **접근 방식:**
  - 코드 추론 흔적을 활용한 **골드 스타트 SFT**
  - **PPO 강화학습**으로 실시간 코드 실행 및 피드백
- **주요 기능:** 결과 기반 보상으로 **도구 호출 타이밍, 도구 선택, 코드 개선, 자기 수정 능력**을 학습

ex 4) Search-R1 : LLM을 사용한 더 나은 웹 검색을 위한 강화 학습

→ LLM 에이전트가 **자율적으로 다단계 웹 검색**을 수행해 답변의 **정확성과 관련도**를 높이도록 함

- **접근**
  - 검색 증강 LLM 에이전트를 훈련 시켜 **검색 시점·방법·질의 구성·결과 선택·통합**을 스스로 결정하도록 학습
  - **인간 피드백 기반 온라인 강화 학습**으로 검색 전략을 지속 개선
  - 에이전트는 **검색 여부, 질의 구성, 반복 개선** 정책을 학습
- **강화학습**
  - **GRPO(Group Relative Policy Optimization)** 사용

    → PPO의 발전형으로, 여러 실행 결과를 비교해 **더 나은 검색 결과에 보상**을 부여함

- **모델 컨텍스트 프로토콜 (MCP)**
  - **정의:**

    MCP는 AI가 **외부 도구, 리소스, 환경과 표준화된 방식으로 상호 작용**하도록 하는 인터페이스로, **"AI용 USB-C"** 역할을 함.
  - **중요성:**

    MCP가 없으면 **M×N개의 개별 통합**이 필요하지만, MCP를 사용하면 이를 **M*N개의 연결만으로 단순화**할 수 있음.

  - **아키텍처 개요:**

        - **호스트:** AI 애플리케이션 환경 (예: 채팅 앱, IDE)
        - **클라이언트:** 호스트와 MCP 서버 간 통신 담당
        - **서버:** 표준 형식으로 도구·리소스·프롬프트 제공
    - **핵심 기능:**
        - **도구:** 실행 가능한 작업 (예: 쿼리, 날씨 정보 등)
        - **리소스:** 읽기 전용 데이터 소스 (문서, DB 등)
        - **프롬프트:** AI 행동을 안내하는 템플릿·워크플로
    - **효과:**
        - 연결 복잡도 감소
        - 도구·자료 재사용성 향상
        - 다도구 활용 가능한 **유연한 AI 에이전트 개발** 촉진

- **Agent2Agent 프로토콜**
    - **목표:**

      여러 AI 에이전트가 **내부 메모리나 도구를 직접 공유하지 않고** 맥락 & 작업 업데이트, 지시사항, 데이터만 교환하며 협업하도록 함.
    - **MCP와의 관계:**
        - **MCP:** 에이전트를 **외부 도구**에 연결
        - **A2A:** 에이전트 **간 협업**을 지원
    - **핵심 개념:**
        - **에이전트 서버:** 개별 에이전트를 호스팅
        - **클라이언트 에이전트:** 여러 에이전트 서버에 연결
        - **라우터:** 요청(쿼리)을 적절한 에이전트로 전달

---

4. AI Agent의 추론 및 계획

LLM의 추론 능력(Reasoning)

추론 능력이 있는 대규모 언어 모델

대규모 언어 모델은 논리 퍼즐, 수학적 문제, 코딩과 같은 작업을 해결 가능

- ex) 나탈리아는 4월에 48명의 친구들에게 클립을 판매, 5월에는 그 절반의 클립 판매 4월과 5월에 얼마나 많은 클립을 팔았는가?
- -추론을 사용 x →그녀는 72개의 클립을 판매
- - 추론 사용 → 그녀는 5월에 48/2 = 24개의 클립 판매 → 4월과 5월에 48+24 = 72개의 클립 판매 → 따라서 4월과 5월에 총 72개의 클립 판매

LLM은 프롬프트만으로도 추론 능력을 이끌어낼 수 있음

1. Few-shot promping(Chain-of Thought Prompting)
2. Zero-shot prompting(”Let’s think step by step’)

대규모 언어 모델의 추론 유도
• 프롬프트만으로도 추론 능력을 이끌어낼 수 있음

1. Few-shot prompting (Chain-of-Thought Prompting)
2. Zero-shot prompting ("Let's think step by step")
   (a) Few-shot

Q: 로저는 테니스 공 5개를 가지고 있다. 그는 테니스공이 들어 있는 캔 2개를 더 산다. 각 캔에는 테니스 공이 3개씩 들어 있다. 지금 그는 테니스 공을 몇 개 가지고 있는가?
A: 답은 11이다.

Q: 한 저글러는 16개의 공을 저글링할 수 있다. 공의 절반은 골프공이고, 그 골프공의 절반은 파란색이다. 파란색 골프공은 몇 개인가?
A:
(출력) 답은 8. X

(b) Few-shot-CoT
Q: 로저는 테니스 공 5개를 가지고 있다. 그는 테니스공이 들어 있는 캔 2개를 더 산다. 각 캔에는 테니스 공이 3개씩 들어 있다. 지금 그는 테니스 공을 몇 개 가지고 있는가?
A: 로저는 처음에 공 5개를 가지고 있었다. 3개들이 캔 2개는 6개의 테니스 공이다. 5 + 6 = 11. 답은 11이다.
Q: 한 저글러는 16개의 공을 저글링할 수 있다. 공의 절반은 골프공이고, 그 골프공의 절반은 파란색이다. 파란색 골프공은 몇 개인가?

(출력) 저글러는 총 16개의 공을 저글링할 수 있다. 공의 절반이 골프공이다. 따라서 골프공은 16 / 2 = 8개이다. 골프공의 절반이 파란색이다. 따라서 파란색 골프공은 8 / 2 = 4개이다. 답은 4이다. /

(c) 제로샷(Zero-shot)
Q: 한 저글러는 16개의 공을 저글링할 수 있다. 공의 절반은 골프공이고, 그 골프공의 절반은 파란색이다. 파란색 골프공은 몇 개인가?
A: 답은
(출력) 8 X

(b) Few-shot-CoT
Q: 로저는 테니스 공 5개를 가지고 있다. 그는 테니스공이 들어 있는 캔 2개를 더 산다. 각 캔에는 테니스 공이 3개씩 들어 있다. 지금 그는 테니스 공을 몇 개 가지고 있는가?
A: 로저는 처음에 공 5개를 가지고 있었다. 3개들이 캔 2개는 6개의 테니스 공이다. 5 + 6 = 11. 답은 11이다.
Q: 한 저글러는 16개의 공을 저글링할 수 있다. 공의 절반은 골프공이고, 그 골프공의 절반은 파란색이다. 파란색 골프공은 몇 개인가?

(출력) 저글러는 총 16개의 공을 저글링할 수 있다. 공의 절반이 골프공이다. 따라서 골프공은 16 / 2 = 8개이다. 골프공의 절반이 파란색이다. 따라서 파란색 골프공은 8 / 2 = 4개이다. 답은 4이다. /

(d) 제로샷-CoT
Q: 한 저글러는 16개의 공을 저글링할 수 있다. 공의 절반은 골프공이고, 그 골프공의 절반은 파란색이다. 파란색 골프공은 몇 개인가?
A: 단계별로 생각해 보자.
(출력) 총 16개의 공이 있다. 공의 절반이 골프공이다. 즉 골프공은 8개다. 골프공의 절반이 파란색이다. 즉 파란색 골프공은 4개다. /

대규모 언어 모델의 추론 능력 강화

- 자기 일관성 : 다양한 추론 과정을 생성하고 그 중 가장 일관성있는 답변을 고르는 디코딩 방

1. 프롬프트 입력
2. **Language model이 다양한 reasoning path를 생성**

   예시 추론 경로:

   - (1) `16 - 3 - 4 = 9` → `$2 × 9 = $18` → **The answer is $18.**
   - (2) `16 - 4 - 3 = 9` → `$2 × 9 = $18` → **The answer is $18.**
   - (3) 계산 실수로 `$26`이 나오는 경우도 있음 → **The answer is $26**

3. **최종 답 결정**
    - 여러 reasoning 중 가장 자주 등장하는 답은 `$18`.
    - 따라서 최종적으로 **"The answer is $18."**

대규모 언어 모델의 추론 능력 향상: Tree of Thoughts(ToT

1. 개념 요약

- Tree of Thoughts(ToT)는 언어 모델이 문제를 해결할 때

단순히 한 번의 추론을 하는 대신,

여러 가능한 "생각(thought)"을 트리(tree) 구조로 확장·탐색하면서
가장 합리적이고 정확한 결론을 찾는 방식이에요.

이 과정은 두 가지 탐색 방법으로 진행됩니다.

- **너비 우선 탐색 (BFS)**: 다양한 아이디어를 폭넓게 살펴본 후 좋은 가지를 확장
- **깊이 우선 탐색 (DFS)**: 한 아이디어를 깊게 탐구하며 가능성을 평가

---

2. ToT의 구성 요소

Tree of Thoughts는 크게 두 단계로 작동합니다.

(a) Propose Prompt — "생각 제안 단계"

- 모델이 **다음에 어떤 사고 경로로 나아갈 수 있을지** 여러 가지 후보 생각(thoughts)을 생성합니다.
- 예시에서 입력값은 `4, 9, 10, 13`이고,

  모델은 가능한 조합을 제시합니다.
    - 4 + 9 = 13 (남은 수: 10, 13, 13)
    - 10 - 4 = 6 (남은 수: 6, 9, 13)

즉, 모델이 여러 "가능한 다음 생각들"을 만들어내는 단계예요.

(b) Value Prompt — "생각 평가 단계"

- 생성된 각 생각(thought)에 대해 **유용성 또는 가능성**을 평가합니다.
- 예를 들어, 목표가 "24를 만드는 것"이라면

각 단계에서 남은 숫자로 24를 만들 수 있는지 평가합니다.

- 10 + 14 = 24 → 가능
- 13 × 3 = 39 → 불가능

이 과정에서 “유망한 가지(good branch)”는 계속 확장하고,

“가능성이 낮은 가지(bad branch)”는 탐색을 중단합니다.

---

3. 전체 흐름 정리

1. **입력(Input)**: 문제나 목표 주어짐 (예: 숫자 조합으로 24 만들기)
2. **생각 생성 (Thought Generation)**:
   - 여러 가능한 계산 또는 추론 경로를 제시 (Propose Prompt)
3. **생각 평가 (Thought Evaluation)**:
   - 각 경로의 유효성을 평가 (Value Prompt)
4. **탐색(Search)**:
   - BFS 또는 DFS를 사용하여 유망한 경로만 확장
5. **출력(Output)**:
   - 가장 타당한 추론 경로를 통해 최종 답을 도출

---

4. 기존 방식과의 차이

방식	구조	특징
IO (Input–Output)	단일 직선형	단순 질의응답 수준
CoT (Chain of Thought)	직선형 사고 경로	단계별 추론 가능
CoT-SC (Self-Consistency)	여러 직선 경로 중 다수결 선택	일관성 개선
ToT (Tree of Thoughts)	트리 구조로 여러 생각 확장·평가	BFS/DFS 탐색으로 **최적의 추론 경로 선택**

| LLM 에이전트의 Planning
• LLM은 여러 사고 단계를 생성할 수 있음. 즉, 목표 달성을 위한 하위 목표들의 순서를 구성할 수 있음
• 행동(실행) 명령을 통합하면 LLM이 결과를 되돌아보면서 도구를 사용하거나 행동할 수 있음

LLM 에이전트의 Planning 개요

1.ReAct란?

> ReAct (Reason + Act)는 LLM이 단순히 '생각(Reasoning)'만 하거나 '행동(Action)'만 하는 것이 아니라,
>
> **'생각(Thought)'과 '행동(Act)'을 번갈아 수행하며 문제를 해결하는 방식**이다.

즉, 모델이 **스스로 사고의 흐름을 설명하면서**, 필요한 정보를 탐색(Search)하거나 **도구를 사용**해 나가는 과정.

2.HuggingGPT란?

계획 수립을 담당하는 LLM이 복잡한 작업을 나누어 각각의 전문 모델들에게 분배하는 시스템

3.Plan-and-Act란?

계획 에이전트는 거시적인 계획을 생성, 실행 에이전트는 해당 계획에 따라 하위 단계를 실행

추론 시간 연산량 최적화 및 효율적 추론 접근법

추론 시간의 연산량 증대
-추론 시에 컴퓨팅 자원을 더 많이 투입할수록 LLM의 성능이 향상됨
- 이는 난이도가 낮은 작업에서 모델 매개 변수를 늘리는 것보다 더욱 효과적일 수 있다

-추론 시에 컴퓨팅 자원을 더 많이 투입할수록 LLM의 성능이 향상됨

-스케일링의 법칙도 적용됨

추론모델 개요

- **의미:** 추론을 위해 추가적인 컴퓨팅 자원이 활용된 모델들이 새롭게 등장하고 있음.

주요 추론 모델 종류

모델	개발사 / 출시일	특징
o1	OpenAI (2024.12)	OpenAI의 추론 특화 모델
Gemini Series	Google (thought)	'thought' 단위로 사고 과정을 분리
GPT Series	OpenAI (reasoning summary)	사고 과정을 요약해 결과를 제시
DeepSeek-R1	DeepSeek-AI (2025.01)	대규모 추론 중심 모델
Claude Series	Anthropic (ThinkingBlock)	사고 단위를 'ThinkingBlock'으로 구성
Open-source Models	오픈소스 커뮤니티	`<think></think>` 형태의 사고 구조 활용

추론모델 강화

- 정확성과 올바른 형식에 대해서만 보상을 제공함으로써 LLM이 chain-of-thought을 사용하는 방법을 학습하게 함
- • 정확성과 올바른 형식에 대해서만 보상을 제공함으로써 LLM이 chain-of-thought을 사용하는 방법을 학습하게 힘
- ex): " ...< think> reasoning process here </think> <answer> answer here </answer>"

효율적인 추론
• 과도한 사고 현상
• 더 긴 CoT 추론 과정은 실제로 성능을 향상시키지만, 장황하고 중복되는 출력으로 인해 계산 부담 크게 늘어나는 것은 여전히 문제

- **목표:** AI 모델이 추론(Reasoning)을 수행할 때 **불필요한 연산을 줄이고, 효율적으로 사고하도록 개선**
- **접근 방법:**
  - 모델 구조 자체를 개선하거나
  - 추론 결과를 활용하거나
  - 입력 프롬프트를 조정하여 효율을 높이는 방식으로 구분됨

효율적 추론의 세 가지 접근 방식

구분	설명	대표 기법 / 모델
(a) 모델 기반 접근	모델 내부에서 추론 길이나 방식을 제어함	🔷 *DeGRPO*: 제어 토큰(`<short>` , `<think>`)으로 추론 길이를 조절→ 불필요하게 긴 추론 방지
(b) 추론 결과 기반 접근	모델이 추론 도중 스스로 생각을 요약하며 동적으로 사고	🔷 *InftyThink*: 각 단계마다 생각을 정리(`Summary`)하며 누적 추론 수행→ 점진적 사고 및 메모리 효율성 향상
(c) 입력 프롬프트 기반 접근	질문 난이도에 따라 연산량(추론량)을 다르게 배분	🔷 *라우팅(Routing)*: 쉬운 질문은 간단히, 어려운 질문은 깊게 사고🔷 *Claude Sonnet 4*, *GPT-5*: 복합 문제일수록 더 많은 추론 자원 투입

핵심 기술 요약

기술	핵심 아이디어	효과
DeGRPO	`<think>` / `<short>` 등 제어 토큰으로 추론 길이 학습	불필요한 긴 연산 감소
InftyThink	단계별로 생각을 요약하고 누적 반영	동적·점진적 사고
Routing (Claude, GPT-5)	질문 난이도별로 추론 자원 자동 분배	효율적 자원 활용

Deep Research의 배경

| 패러다임의 전환
• 원 검색 → LLM 채봇 → 검색 증강 생성(RAG) → 에이전트 기반 심층 연구

에이전트 기반 심층 연구

| 핵심 개념
• LLM은 다단계 추론과 검색을 결합
• 동적 피드백 루프를 형성: 검색 결과에 대해 추론하고, 이 추론은 이후 검색 과정을 조정
• 간단한 응답 생성기가 아니라, 능동적인 정보 탐색 에이전트 역할
• 필요한 경우, 문서를 읽고 쿼리를 반복적으로 생성하고 수정하여 다른 도구나 에이전트와 협력

| 역동적인 환경에서의 어려움과 해결방안
• 검색과 추론의 결합에는 어려움이 존재

- 기존 RAG 파이프라인은 검색한 다음 추론하는 고정된 순서를 사용 → 오류 축적
- 에이전트 기반 접근법은 추론이 언제, 무엇을 검색할지 결정하도록 하여 오류 전파를 줄일 수 있음
  • 적응력 한계
- 프롬프트와 지도 학습에 대한 의존으로 인해 새로운 환경에 대한 유연한 전략을 개발하기가 어려움
- 시행 착오를 통해 최적의 전략을 배우려면 강화 학습이 필요
  • 효율적인 리소스 할당
- 검색과 추론 둘 다 토큰과 시간을 소비
- 작업 유형에 따라 검색/추론 리소스 비율을 동적으로 조절하는 기술이 필요

| 강화학습 기반 훈련 및 보상 설계
• 프롬프트

- 예: ReAct - 추론+행동 구조를 사용하여 프로세스를 검색 및 추론 단계로 분할
  • 지도학습
- 예: Toolformer - 모델은 훈련 데이터의 패턴을 기반으로 도구를 사용하는 법을 학습
  • 강화 학습
- 목표 : 모델이 독립적으로 최적의 검색 및 추론 전략을 발견하도록 유도
- 예: R1-Searcher, DeepResearcher - 복잡한 작업을 세분화하고, 쿼리 시퀀스를 계획하고, 정보를 검증하고, 전략을 조정

Test-Time Scaling (TTS)
• 정의: 추론/검색 단계 수를 늘리면 성능 향상으로 이어짐 (TTS 법칙)
• 추론 TTS: 추가적인 추론 단계 (CoT, Self-Refinement, Self-Consistency 등) 도입
→ 정확도 향상

- 예: 수학적 증명
  • 검색 TTS: 반복적, 다단계 검색 →정보 수집 범위 확장
- 예: 문헌 리뷰, 지식 그래프 구축
  · Trade-off:
- 검색과 추론 둘 다 자원을 소모 → 각 작업 유형에 맞는 최적 비율의 조정이 필수.

방법론 - 문제 해결 과정

- 생각 → 도구 사용 → 생각 → ... → 답변 사이클 반복
  - <생각> 단계에서 세부 목표를 세우고 검색할 내용을 결정함
  - <검색> 에이전트가 검색 API를 사용해서 상위 k개 결과(제목, 주소, 요약문)를 가져옴
  - <탐색> 에이전트가 웹페이지 내용을 읽고 유용한 정보를 단기 저장소에 모아둠
  - 읽는 과정에서 에이전트는 (a) 계속 읽을 건지 아니면 그만둘 건지, (b) 어떤 내용을 저장할 건지 판단
  - 해당 페이지가 별로 도움이 안 된다 싶으면 빠르게 다른 URL로 이동
  - 필요한 정보를 충분히 모았다면, <답변> 단계에서 최종 결론을 생성

| 역동적인 환경에서의 어려움과 해결방안

- 많은 요청 동시 처리
  - 한 번에 최대 4,096개 도구 호출이 발생 가능
  - 처리 지연을 막고 빠른 속도를 유지하기 위해 50개 노드의 분산 CPU 시스템에서 작업을 나누어 처리
    - 웹 크롤링 및 API 제약
  - 크롤링 방지 시스템과 요청 제한으로 인한 실패 가능성
  - 재시도 메커니즘과 캐시를 사용하여 호출 빈도와 비용 감소
    - 정보 추출 효율성 문제
  - 웹페이지를 작은 단위로 나누어서 하나씩 차례로 분석
  - 별로 중요하지 않다고 판단되면 바로 포기하고 핵심 페이지에 컴퓨팅 자원을 집중

| 강화학습 훈련 프레임워크 및 보상

- 알고리즘 : GRPO (Group Relative PPO)
  - 이전 정책 (π_old)을 사용한 여러 그룹의 실행 결과를 기준으로 삼음.
  - 새로운 정책(policy)이 이 결과들의 발생 확률을 높이는 정도에 따라 PPO 방식으로 업데이트
    - 보상 설계:
  - 형식 오류 : -1 페널티.

- 올바른 형식 : 단어 레벨의 F1 점수(0-1)를 부여
- 사실에 대한 정확도, 논리적 일관성 등에 대한 복합적인 보상을 도입할 수 있음

에이전트 기반 심층 연구 (DeepResearcher)

핵심 아이디어

- 단순 검색형 모델이 아닌, **스스로 계획하고 검증하며 사고하는 AI 에이전트**를 지향.

창발적 행동 유형

유형	설명
계획 능력(Planning)	End-to-End 강화학습으로 단계별 계획을 세우며 문제 해결
교차 검증(Cross Validation)	첫 결과에 의존하지 않고 여러 출처를 비교해 추가 확인
성찰적 재검색(Reflection)	검색 결과가 부족하면 방향을 수정해 다시 탐색
솔직한 회피(Honesty)	확실하지 않을 땐 추측 대신 “모른다”고 답변

요약 한 줄:

> DeepResearcher는 계획, 검증, 성찰, 정직성을 통해 사고 과정이 자율적이고 신뢰성 높은 AI 추론 에이전트를 구현한다.

1. 평가 및 발전 방향
   | 한계점 및 고려사항
   • 높은 비용 (토큰)

   -단일 에이전트는 표준 질의응답보다 4배 더 많은 토큰을 사용; 멀티 에이전트는 15배 더 많이 사용

   -고부가가치 작업에서만 비용-효율적임
   • 범용성의 한계

   -병렬 작업들로 분할될 수 있는 작업에서만 효과적

   -코딩, 디버깅 등 순차적인 단일 컨텍스트 처리가 필요한 작업에는 적합하지 않음
   • 에이전트 관리의 복잡성

   -너무 많은 숫자의 에이전트가 생성되거나, 무한 루프 또는 에이전트 간 간섭 위험

-명확한 정지 조건 및 프롬프트 가이드 라인과 같은 보호 조치 필요

한계점 및 고려사항
• 상태 관리 및 오류 전파

-작은 오류가 프로세스의 전체 방향을 왜곡시킬 수 있음

-체크 포인트 설정 및 복구를 위한 메커니즘 필요

• 평가 및 디버깅의 어려움

-실행 경로와 답변은 매 실행마다 다르므로 공평한 평가가 어려움

-LLM 평가 점수 (0-1 스케일)와 사람에 의한 평가를 함께 사용하여 평가

| 미해결 문제 및 발전 방향
• 인간 중심의 상호 작용

-AI를 자율적인 연구원이 아니라 강력한 조력자로 포지셔닝

-중간 사고 과정을 보여주고 설명 → 투명성 보장 및 신뢰성 향상

-사용자 피드백을 통해 지속적인 개인화
• 전문 분야 에이전트

-의학, 법, 생물학 등과 같은 분야에 대한 분야별 지식과 추론 구조를 결합

-도메인 별 데이터셋 및 평가 프레임워크 개발

-범용적인 에이전트와 및 전문적인 에이전트의 하이브리드를 운영
• 효율적인 TTS 구현

-더욱 효율적인 추론: 중요한 부분에 대해서만 깊이 추론하고, 적절하게 더 작은 모델을 사용

-더욱 효율적인 검색: 인덱싱 최적화, 비생산적 검색 조기 중단, 우선 순위가 낮은 경로 제거

---

5. 도메인 특화 AI Agent

전문 분야에 대한 지식 및 추론 구조 결합의 필요성

1. Deep Reasearch의 배경

- **패러다임의 전환**

  웹 검색 → LLM 챗봇 → 검색 증강 생성 (RAG) → 에이전트 기반 심층 연구

2. 에이전트 기반 심층 연구

- **핵심 개념:**
  - LLM이 **다단계 추론과 검색을 결합**
  - **검색 결과 → 추론 → 검색 조정**의 **동적 피드백 루프** 형성
  - 단순 응답 생성기가 아닌 **능동적 정보 탐색 에이전트**로 작동
  - 필요 시 **문서 분석·쿼리 수정·도구·다른 에이전트와 협력** 수행

- **역동적 환경에서의 어려움과 해결 방안**
  1. **검색-추론 결합의 어려움**
     - 기존 RAG: 검색 → 추론 순서로 고정 → 오류가 누적될 수 있음
     - 에이전트 기반: 추론이 검색 시점과 내용을 결정 → 오류 전파 감소
  2. **적응력 한계**
     - 프롬프트 지도학습 의존 → 새로운 환경 대응 어려움
     - 강화 학습 활용 시 시행착오로 최적 전략 학습 가능
  3. **효율적 리소스 할당**
     - 검색과 추론 모두 토큰·시간 소모
     - 작업 유형별로 리소스 비율을 동적으로 조절하는 기술 필요

- **강화학습 기반 훈련 및 보상 설계**
  1. **프롬프트 기반 접근**
     - 예: **ReAct** – 추론과 행동 구조를 결합하여 검색과 추론 단계를 분리

2. **지도학습 기반 접근**
    - 예: **Toolformer** – 훈련 데이터 패턴을 통해 도구 활용 방법 학습
3. **강화학습 기반 접근**
    - 목표: 모델이 스스로 최적 검색·추론 전략을 발견
    - 예: **R1-Searcher, DeepResearcher** – 복잡한 작업 세분화, 쿼리 계획, 정보 검증, 전략 조정

- **Test-Time Scaling (TTS)**
    1. **정의**
        - 추론 또는 검색 단계를 늘리면 성능 향상 (TTS 법칙)
    2. **추론 TTS**
        - 추가 추론 단계 적용 (CoT, Self-Refinement, Self-Consistency) → 정확도 향상
        - 예: 수학적 증명
    3. **검색 TTS**
        - 반복적·다단계 검색 → 정보 수집 범위 확대
        - 예: 문헌 리뷰, 지식 그래프 구축
    4. **Trade-off**
        - 검색과 추론 모두 리소스 소모 → 작업 유형별 최적 비율 조정 필요

- **방법론 : 문제 해결 과정**
    1. **사이클 구조**: 생각 → 도구 사용 → 생각 → ... → 답변 반복
    2. **생각 단계**: 세부 목표 설정 및 검색할 내용 결정
    3. **검색 단계**: 검색 API로 상위 k개 결과(제목, 주소, 요약) 수집
    4. **탐색 단계**: 웹페이지 내용 확인, 유용 정보 단기 저장
        - 읽을지 계속할지 결정
        - 어떤 정보를 저장할지 판단
        - 도움이 없으면 다른 URL로 이동

5. **답변 단계**: 충분한 정보 수집 후 최종 결론 생성

- **역동적 환경에서의 어려움과 해결방안**
  1. **동시 요청 처리**
     - 최대 4,096개 도구 호출 가능 → 분산 CPU(50개 노드)로 작업 분할, 지연 최소화
  2. **웹 크롤링 및 API 제약**
     - 크롤링 방지·요청 제한으로 실패 가능 → 재시도와 캐시 활용으로 호출 빈도·비용 절감
  3. **정보 추출 효율성**
     - 웹페이지를 작은 단위로 분석 → 중요하지 않은 페이지는 즉시 포기, 핵심 페이지에 자원 집중

- **강화학습 훈련 프레임워크 및 보상**
  1. **알고리즘: GRPO (Group Relative PPO)**
     - 이전 정책(π_old) 기반 여러 그룹 실행 결과 참고
     - 새로운 정책이 결과 발생 확률을 높이는 정도에 따라 PPO 방식으로 업데이트
  2. **보상 설계**
     - 형식 오류: -1 페널티
     - 올바른 형식: 단어 수준 F1 점수(0~1)
     - 필요 시 사실 정확성, 논리적 일관성 등 복합 보상 적용

- **성능 분석**
  - **벤치마크 성능**: In-domain(NQ, TQ, HotpotQA, 2Wiki) 및 Out-of-domain(Musique, Bamboogle, PopQA)에서 SOTA 달성
  - **특징**: 특히 Bamboogle에서 기존 웹 검색 기반 모델을 크게 능가

    → 실제 웹 환경 학습 효과 입증

- **창발적 행동**

- **계획 능력**: end-to-end 강화학습을 통해 다단계 계획 수행 가능
- **교차 검증**: 초기 결과에만 의존하지 않고 추가 확인으로 신뢰도 향
- **성찰적 재검색**: 불만족스러운 검색 결과 시 검색 전략 조정
- **솔직한 회피**: 확실하지 않은 경우 추측 대신 "모름"으로 응답

3. 평가 및 발전 방향 (pdf에는 4번이라 돼있음, 3번 없으유..)

- **한계점 및 고려사항**
  1. **높은 비용**
     - 단일 에이전트: 표준 QA 대비 4배 토큰 사용
     - 멀티 에이전트: 15배 토큰 사용 → 고부가가치 작업에서만 비용 효율적
  2. **범용성 한계**
     - 병렬 처리 가능한 작업에서만 효과적
     - 코딩, 디버깅 등 순차적 단일 컨텍스트 작업에는 부적합
  3. **에이전트 관리 복잡성**
     - 과도한 에이전트 생성, 무한 루프, 간섭 위험
     - 명확한 정지 조건과 프롬프트 가이드라인 필요
  4. **상태 관리 및 오류 전파**
     - 작은 오류가 전체 프로세스 방향을 왜곡할 수 있음
     - 체크포인트 및 복구 메커니즘 필요
  5. **평가 및 디버깅 어려움**
     - 실행 경로와 답변이 매번 달라 공평한 평가 어려움
     - LLM 점수(0~1)와 사람 평가를 병행하여 평가

- **미해결 문제 및 발전 방향**
  1. **인간 중심 상호작용**
     - AI를 자율적 연구원이 아닌 조력자로 포지셔닝

- 중간 사고 과정 공개 → 투명성과 신뢰성 향상
- 사용자 피드백 기반 지속적 개인화

2. **전문 분야 에이전트**
    - 의학, 법, 생물학 등 분야별 지식과 추론 구조 결합
    - 도메인별 데이터셋과 평가 프레임워크 개발
    - 범용 에이전트와 전문 에이전트 하이브리드 운영
3. **효율적인 TTS 구현**
    - 추론: 중요한 부분만 심층 분석, 적절히 작은 모델 활용
    - 검색: 인덱싱 최적화, 비생산적 검색 조기 중단, 낮은 우선 순위 경로 제거