Journal of KIIT. Vol. 24, No. 6, pp. 1-17, Jun. 30, 2026. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2026.24.6.1

심층 연구 에이전트의 컨텍스트 드리프트 완화를 위한 체크리스트 루브릭 중심 반복 제어

김민석*, 정유철**

Checklist Rubric-Driven Iterative Control for Mitigating Context Drift in Deep Research Agents

Minseok Kim*, Yuchul Jung**

이 연구는 국립금오공과대학교 대학 연구과제비(2024~2026)와 (2026)년도 교육부 및 경상북도의 재원으로 경북RISE센터의 지원을 받아 수행된 지역혁신중심 대학지원체계(RISE)-(지역성장 혁신LAB)의 결과입니다.(2026-rise-15-105)

요 약

최근 LLM 기반 심층 연구 에이전트는 웹 탐색과 근거 통합을 반복하며 장문 보고서를 생성하는 방향으로 발전하고 있다. 그러나 기존 반복 개선 방식은 명시적 기준 없이 탐색과 재작성이 이루어질 경우 컨텍스트 드리프트와 목표 누락이 누적될 수 있다. 본 논문은 이를 완화하기 위해 Checklist-based Rubric 기반 병렬 조사 멀티에이전트 프레임워크를 제안한다. 제안 방법은 질의로부터 전역·지역 명세를 생성해 이를 체크리스트 루브릭으로 고정하고, FAIL 항목만 선택적으로 재실행하여 요구사항 중심의 통제된 반복을 수행한다. ResearchQA 평가에서 제안 방법은 일부 조건에서 체크리스트가 없는 설정 대비 ORS 개선 경향을 보였으며(Qwen3-8B: 93.30%→94.62%, GPT-OSS-120B: 95.19%→97.77%), 그 효과는 모델과 조건에 따라 달랐다. 이러한 결과는 체크리스트 기반 보강이 특정 조건에서 목표 누락 완화와 ORS 개선에 기여할 가능성을 보였다.

Abstract

Recent LLM-based deep research agents have advanced toward generating long-form reports through iterative web exploration and evidence synthesis. However, when exploration and rewriting proceed without explicit criteria, context drift and target omission can accumulate. To address this issue, this paper proposes a checklist-based rubric-driven parallel investigation multi-agent framework. The proposed method derives global and local specifications from the query, fixes them as a checklist rubric, and selectively re-executes only failed items to enable controlled requirement-centered iteration. Evaluation on ResearchQA showed that the proposed approach exhibited a tendency toward improved ORS under some settings compared with the configuration without a checklist (Qwen3-8B: 93.30% →94.62%, GPT-OSS-120B: 95.19%→97.77%), although the effect varied by model and condition. These results suggest that checklist-based refinement can help mitigate target omission and improve ORS under certain conditions.

Keywords

deep research, multi agent systems, LLM, large language models, autonomous agents, research automation, RAG, retrieval augmented generation, knowledge synthesis

* 국립금오공과대학교 컴퓨터 · AI융합공학과
- ORCID: https://orcid.org/0009-0007-9180-7983
** 국립금오공과대학교 컴퓨터공학부 부교수(교신저자)
- ORCID: https://orcid.org/0000-0002-8871-1979

· Received: Feb. 23, 2026 Revised: Apr. 16, 2026, Accepted: Apr. 19, 2026
· Corresponding Author: Yuchul Jung
Dept. of Computer Engineering, Kumoh National Institute of Technology, 61 Daehak-ro (yangho-dong), Gumi, Gyeongbuk, [39177] Korea
Tel.: +82-478-7436, Email: jyc@kumoh.ac.kr

Ⅰ. 서 론

최근 LLM 기반 에이전트는 단순 대화형 응답을 넘어서, 광범위 탐색·근거·종합을 통해 장문 보고서를 생성하는 '심층 연구'의 방향으로 빠르게 발전하고 있다[1]. 예를 들어 ChatGPT의 Deep Research[2]는 복잡한 질문에, 다단계로 자료를 수집·추론·종합하여 리포트 형태의 결과를 생성하도록 설계되었다고 소개된다.

이렇듯 상용 기능의 확산과 동시에, 오픈소스 생태계에서도 유사한 흐름이 강화되고 있다. GPT-Research[3], Thinkdepth.ai DeepResearch[4]와 같은 오픈소스 프레임워크 또한 장문 보고서 생성 파이프라인(수집-정리-작성)과 반복적 개선 루프를 제공하며, 심층 연구 에이전트의 구현과 재현 가능성을 높이고 있다.

그러나 반복 실행 자체가 산출물의 안정성을 보장하지는 않는다. 다수의 시스템에서 최종 결과가 사용자의 요구 항목을 부분적으로 누락되는 '목표 누락 현상(Coverage gap)'이 반복적으로 관찰되며, 특히 반복을 제어하는 명시적 기준점이 부재할 경우, 탐색 방향의 불안정성과 누락의 고착화가 동시에 발생한다.

본 연구는 "사용자 목표 달성"을 체크리스트 기반 루브릭 충족으로 정의한다. 즉, 질의 Q가 요구하는 핵심 요소들이 체크리스트 $R = r_1, r_2, \ldots, r_m$로 주어졌을 때, 생성된 장문 응답 A가 각 항목을 얼마나 충족하는지로 목표 달성률을 규정한다.

이 관점에서 목표 누락 현상은 최종 산출물이 사전 정의된 요구 항목 집합을 완전하게 만족하지 못하는 경우로 이해된다. 목표 누락 현상이 발생하는 원인은 다음과 같이 정리할 수 있다. 첫째, 에이전트가 전역 목표를 유지하지 못하고 탐색 방향을 잃는 컨텍스트 드리프트(Context drift)[5]가 누적될 수 있다. 에이전트는 제한된 컨텍스트 내에서만 추론하므로, 수행 단계가 진행됨에 따라 초기 의도와 기준이 희석되거나 맥락에서 밀려나 목표가 약화될 수 있다. 둘째, 목표가 명시적으로 구조화되지 않으면, LLM이 매 반복마다 수집 방향을 임의로 확장하여 요구 항목과 관련 없는 정보가 증가하고, 정작 필요한 요소가 누락될 가능성이 커진다[6]. 셋째, 초안 갱신 자체가 누락을 자동으로 해소하지 않는다. 반복이 이루어지더라도, '무엇을 보완해야 하는가'에 대한 검증 신호가 없으면 누락 문제가 해결되지 않거나, 보강 과정에서 다른 항목이 새롭게 훼손되는 문제가 발생한다[7]. 이러한 한계를 완화하기 위해 다양한 기법[8][9]이 제안되어 왔으나, 반복을 구동하는 평가 신호 자체가 불안정할 수 있다는 점에서 평가 결과의 신뢰성에 대한 의문이 남는다.

이러한 목표 누락 현상을 줄이기 위해, 생성 과정의 기준점을 명시적으로 고정하는 '명세(Specification)기반 심층 연구 프레임워크'를 제안한다. 제안 방법은 사용자 질의 Q로부터 전역 명세(Global spec)을 생성하고, 전역 명세 내의 목표를 체크리스트 형태의 루브릭으로 구조화한다. 이후 루브릭을 단위 작업으로 분해하여 병렬 연구 에이전트가 각각의 목표 항목(체크리스트)를 충족하기 위한 자료를 수집/정리하고, 작성자가 이를 통합하여 초안을 작성한다.

프레임워크의 핵심은 평가 단계가 단순 채점에 머물지 않고, 체크리스트 항목 단위의 판정과 피드백이 다음 반복 실행을 결정하는 제어 신호로 동작한다는 점이다. 이 과정은 TTD-DR[10]의 "초안 중심 점진 개선" 아이디어를 계승하되, 반복을 구동하는 기준을 체크리스트 기반 명세로 고정하여 일반적인 Self-Refine[8]나 루브릭 기반 점수화[9] 계열의 주관적 자기평가 부분을 명확한 근거와 재현 가능 형태로 전환하여 설계한다. 그 결과, 반복 보강이 누락된 요구 항목을 선택적으로 보강하는 방향으로 정렬되며, 평가 신호가 점수나 자기서술에 의존하지 않기 때문에 평가 결과의 일관성과 신뢰성 또한 향상된다.

ResearchQA[11]에서 프레임워크를 평가한 결과, 목표가 명시적으로 고정된 환경에서는 제안 방법이 w/o checklist 대비 전반적으로 긍정적인 ORS 개선 경향을 보였으며, 그 효과는 조건과 모델 규모에 따라 다르게 나타났다. 이러한 결과는 체크리스트 기반 보강이 "무엇을 채워야 하는지"를 항목 단위로 고정·추적함으로써, 일부 설정에서 요구사항 누락 완화에 기여할 가능성을 보였다. 정리하자면, 본 연구의 기여점은 다음과 같다.

1) 추론 단계에서 생성된 체크리스트 루브릭을 실행 과정에서 참조되는 기준점으로 설계하여, 실행 단계 전반에서 목표 이탈과 드리프트를 억제할 수 있음을 보였다.

2) 반복 개선 프레임워크에서 실패 체크리스트에 대응되는 작업만 재실행하도록 함으로써 실행 시점에서 반복을 제어하는 신호로의 적용 가능성을 보였다.

II. 관련 연구

2.1 심층 연구 에이전트 프레임워크

심층 연구 에이전트 프레임워크는 일반적으로 (1) 과제 해석 및 계획 수립, (2) 외부 자료 탐색 및 수집, (3) 근거 요약 및 정리, (4) 장문 보고서 작성의 파이프라인으로 구성된다. 최근에는 이 과정을 "한 번의 검색 후 작성"으로 끝내기보다, 여러 단계의 탐색과 초안 갱신을 반복하여 보고서의 완성도를 높이기 위한 방향으로 발전하고 있다.

대표적인 상용 심층 연구 에이전트인 OpenAI의 Deep Research[2]는 온라인 자료를 찾고-분석하고-종합하여, 리서치 애널리스트 수준의 보고서를 생성하는 에이전트로 소개된다. 오픈소스 생태계에서도 유사한 흐름이 강화되고 있다. GPT-Researcher[3]는 depth/breadth를 갖는 tree-like exploration을 통해 하위 주제 탐색을 확장하고, 비동기 병렬 실행으로 여러 가지 연구 경로를 동시에 진행하는 구조를 채택한다. 또한 LangChain은 DeepAgents[13] 및 Open DeepResearch[14]와 같은 구성요소를 통해 계획 도구, 파일 시스템 기반 컨텍스트 관리, 서브에이전트(분할 정복)등 장기 작업을 위한 기본 아키텍쳐를 제공한다.

TTD-DR[10]은 장문 리포트 생성을 "초안에서 시작해 반복적으로 정제하는 과정"으로 해석하며, 각 단계에서 외부 검색을 통해 근거를 보강하면서 초안을 점진적으로 개선하는 매커니즘을 제안한다. 다만 이러한 프레임워크에서도 반복 자체가 곧바로 "요구사항 충족"을 보장하지는 않는다. 대부분의 시스템은 반복을 수행하더라도 "무엇을 만족해야 하는지"에 대한 기준이 내부적으로 안정화되어 있지 않다. 그 결과, 반복이 진행될수록, 초기 과제 해석과 목표가 점차 희석되는 컨텍스트 드리프트(Context drift)[5]가 누적되고, 탐색이 '그럴듯한 확장' 위주로 흐르면서 중요한 요구 항목이 끝까지 누락되는 coverage gap이 고착될 수 있다. 즉, 반복은 종종 더 많은 정보를 추가하지만, 그 추가가 사용자 목표의 완결성을 체계적으로 보장하는 방식으로 정렬되어 있지 않기 때문에, 목표 누락 문제가 지속적으로 발생한다.

국내에서도 LLM과 RAG, 웹 검색, 에이전트 워크플로우를 결합한 연구가 점차 보고되고 있다. 예를 들어, 감염병 분야에서는 RAG와 웹 검색을 결합한 AI 에이전트가 제안되었고[15], 주소 정보 분야에서는 질의 복잡도에 따른 단계적 추론과 근거 검증을 포함한 에이전틱 RAG 기반 챗봇이 제시되었다[16]. 또한 Graph Agent 및 LangGraph 기반 연구들은 검색된 정보의 신뢰성 평가, 정보 종합, 다중 에이전트 오케스트레이션을 통해 응답 품질을 높이고자 하였으며[17], RAG 기반 지식 에이전트나 보안 지능 에이전트와 같은 도메인 특화 응용[18]도 보고되고 있다. 이러한 국내 연구들은 검색 기반 LLM 시스템이 단순 질의응답을 넘어 자료 수집·검증·통합을 수행하는 방향으로 확장되고 있음을 보여준다. 다만 기존 연구들은 주로 도메인 특화 응용이나 RAG 파이프라인 고도화에 초점을 두고 있으며, 장문 심층 연구 보고서 생성을 위해 반복 탐색을 제어하고 목표 누락 및 컨텍스트 드리프트를 완화하는 문제는 충분히 다루지 않았다.

2.2 루브릭 기반 평가 및 LLM-as-a-judge

루브릭[19]은 특정 과제의 산출물을 평가하기 위해 평가 기준(Criteria)를 명시적으로 정의하고, 각 기준이 어느 수준으로 충족되었는지를 판단할 수 있도록 구성한 평가 틀이다. 최근 LLM 기반 장문 생성 과제에서는 '정답 Text'를 두기 어려운 경우가 많아, 정답과의 단순 유사도 보다 "요구 항목을 얼마나 충족했는지"를 루브릭 항목으로 평가하는 방식이 널리 사용된다.

LLM-as-a-judge[20] 패러다임은 LLM이 인간 대신 평가를 수행하도록 하여, 인간 평가의 비용을 줄이면서도, 루브릭에 따라 다차원적 품질을 비교적 유연하게 평가할 수 있다는 장점이 있다. 특히 장문 보고서처럼 정답이 고정되지 않는 설정에서 루브릭 기반의 LLM-as-a-judge는 평가자의 판단을 일정 수준 가이드하거나, 평가를 자동화하는 역할을 수행한다.

반면 루브릭 기반 LLM-as-a-judge[21]에는 몇 가지 구조적 제약이 보고된다. (1) 기준의 주관성, (2) Likert/등급 점수의 경계 모호성은 평가자(Judge) 모델 간 일치도 저하와 분산 증가로 이어질 수 있다. CheckEval[22]은 기존 LLM-as-a-judge에서 관찰되는 불안정성과 분산이 "주관적 기준 + 다단계 점수 부여"에서 기인한다고 분석하고, 이를 이진(Yes/No) 형태의 체크리스트로 변환하여 평가하도록 하는 방향을 제시한다. 또한 최근 RULERS[23]는 루브릭 텍스트가 프롬프트 민감도에 의해 흔들리거나(Rubric instability), 근거 없는 판정이 발생하는 문제를 지적하며, 루브릭을 잠금된 실행 가능한 명세로 다루는 것이 신뢰도 향상에 중요하다고 주장한다. 최근에는 루브릭을 단순한 사후 평가 기준이 아니라, 생성 과정에서 모델이 따라야 할 추론·행동 제약으로 통합하려는 시도도 등장하고 있다[9]. 그러나 이러한 접근이 실제 심층 연구처럼 반복 탐색 - 통합 작성이 결합된 장문 보고서 생성에서, (i) 점수 기반 루브릭이 신뢰성이 있는지, (ii) 항목별 FAIL 신호를 제어 신호로 사용해 관련 서브태스크만 선택적으로 재실행하는 방식까지 포괄적으로 연결되어 검증되었는지는 여전히 불분명하다.

2.3 명세 기반 멀티에이전트 프레임워크

최근 멀티에이전트 프레임워크는 역할 분담과 단계 분해를 통해 장기 작업을 수행하도록 설계된다. 이 계열의 핵심 아이디어는 "에이전트 간 협업이 가능해지려면, 각 에이전트가 공유하는 작업 기준(명세)과 산출물 형태(계약)가 필요하다"는 점이며, 이를 구현하기 위해 많은 프레임워크가 자연어 명세를 구조화된 아티팩트(문서/계약/절차)로 변환하고 이를 실행 기준으로 삼는다. 대표적으로 MetaGPT[24]는 소프트웨어 회사를 모사한 역할 집합(예: PM, Architect, Engineer 등)을 구성하고, 역할별 산출물을 표준 운영 절차(SOP; Standardized Operation Procedures)로 정의하여 요구사항, 설계, 구현이 문서 아티팩트를 통해 진행되도록한다. MetaGPT는 "Code = SOP(Team)"라는 철학 아래, 단일 프롬프트보다 절차화된 명세를 중간 산출물로 재작성하는 과정을 통해, 협업을 정렬시키는 방식을 강조한다. 유사하게 ChatDev[25]는 소프트웨어 개발을 여러 단계로 분해하고, 단계별로 특화된 에이전트가 대화 체인을 통해 협업하도록 설계한다. ChatDev의 핵심은 "무엇(what)을 소통할지"와 "어떻게(how) 소통할지"를 구조화하여 에이전트 간 상호작용을 안정화하는 것으로, 결과적으로 단계별 목표와 산출물이 자연어 명세로 공유되도록 한다. 한편 AutoGen[26]과 같은 범용 멀티에이전트 프레임워크는 특정 도메인(예; SW 개발)에 고정되지 않고, 여러 에이전트가 대화하며 도구를 호출하고 사람 입력을 혼합하는 conversation programming을 제공한다. 이러한 프레임워크는 "명세 기반 멀티에이전트"를 구현할 수 있는 실행 인프라로서, 개발자가 에이전트 간 상호작용 규칙(검토자/작성자, 계획자/실행자)을 설계해 명세-실행-검증과 유사한 루프를 구성하는데 활용된다.

다만 기존 프레임워크에서 명세는 주로 역할 분담과 커뮤니케이션을 정렬해 협업을 원활히 하는 워크플로우 기준으로 활용되는 반면, 본 연구의 프레임워크는 명세를 요구 항목(체크리스트)을 고정·추적하여 누락을 식별하고 보강을 구동하는 기준점으로 사용한다는 점에서 차이가 있다.

III. 제안 프레임워크

3.1 문제 설정 및 설계 개요

본 연구에서 해결하고자 하는 핵심 과제는 반복적 탐색 및 작성 과정에서 발생하는 임의적 수집 및 서술을 줄이고, 최종 산출물이 사용자의 요구항목을 안정적으로 충족하도록 생성 과정을 구조화하는 것이다. 이를 위해 사용자 질의로부터 도출되거나 외부에서 주어진 사용자 목표를 명세(Specification)로 표현하고, 이를 검증 가능한 체크리스트 형태로 구성한 프레임워크를 제안한다. 구성의 핵심은 자료

수집 및 보고서 생성 과정을 LLM의 즉흥적인 임의적 판단에 맡기지 않고, 정의된 명세와 체크리스트를 만족하는 것을 목표로 한다.

제안 프레임워크의 핵심 아이디어는 두 가지이다. 첫째, 시스템은 질의 Q로부터 보고서의 목표/범위/출력 조건을 고정하는 전역 명세(Global Specification)을 생성하며, 이는 이후 모든 단계가 참조하는 기준점으로 기능한다. 둘째, 전역 명세에 포함된 요구사항을 루브릭 단위로 분해하여 병렬 연구를 수행하고, 생성된 초안에 대하여 체크리스트 기반 평가를 통해 미충족(FAIL) 루브릭만 선택적으로 재실행함으로써 초안을 반복 보강한다. 이와 같은 루프를 통해 장문 보고서 생성 과정이 명세 준수 관점에서 체계적으로 개선되도록 한다.

3.2 명세 아티팩트

3.2.1 전역 명세

표 1. 전역 명세의 필드 구성 및 역할
Table 1. Field composition and roles of the global specification

Field	Description
objective	Topic/Purpose of the report
output_contract	Intended audience, output language, and deliverables/components to be included in the report
term_definitions	Meaning of key terms to avoid confusion due to homonyms/polysemy
coverage_rubrics	Structure the purpose/output contract into a rubric and checklist of requirements.

전역 명세(Global specification)는 사용자 질의 Q로부터 도출되는 전역 목표와 산출물 조건을 명시적으로 고정한 아티팩트로서, 프레임워크 실행 전 과정에서 일관되게 참조되는 단일 기준점 역할을 수행한다. 심층 연구 과정에서 자료 수집과 서술이 단계마다 임의적으로 변형될 경우 목표 누락이 발생하기 쉬우므로, 전역 명세는 보고서가 충족해야 할 목적·범위·형식 및 핵심 용어를 구조화하여 이후 모듈(병렬 연구, 초안 작성, 평가, 보강)이 동일한 방향성 하에서 동작하도록 한다. 표 1에 제시한 바와 같이, 전역 명세의 핵심 구성은 (i) 보고서의 목적을 한 문장으로 정의하는 objective, (ii) 예상 독자·출력 언어·필수 산출물 등을 규정하는 output_contract, (iii) 문서 전반에서 용어의 의미를 고정하여 동음이의어·다의어로 인한 해석 혼동을 방지하는 term_definitions, (iv) 사용자 목표를 루브릭 단위로 구조화한 coverage_rubrics로 구성된다.

3.2.2 지역 명세

지역 명세(Local specification)는 전역 명세에 포함된 단일 Coverage Rubric을 입력으로 생성되는 작업 단위 명세로, 각 병렬 연구 에이전트가 수행해야 할 연구 계획과 산출물 요구사항을 구체화한다. 전역 명세가 보고서 전체에 대한 목적·범위·출력 조건을 고정하는 상위 기준점이라면, 지역 명세는 그 중 특정 루브릭을 달성하기 위해 "무엇을 조사하고, 어떤 형태로 정리하여, 어떤 기준을 만족해야 하는가"를 연구 에이전트 관점에서 재서술한 실행 지침에 해당한다. 각 필드의 세부 역할은 표 2에 정리하였다.

표 2. 지역 명세의 필드 구성 및 역할
Table 2. Field composition and roles of the local specification

Field	Description
objective	Research topics/objectives corresponding to a single coverage rubric
deliverables	Output requirements that must be included in research notes
rubric	Checklist of things research agents must satisfy

3.3 에이전트 구성 및 입출력

3.3.1 명세 빌더

명세 빌더(Spec builder)는 사용자 질의 Q를 입력으로 받아, 전단계(병렬 연구-초안 작성-평가-보강)가 참조할 명세 아티팩트를 생성하는 모듈이다.

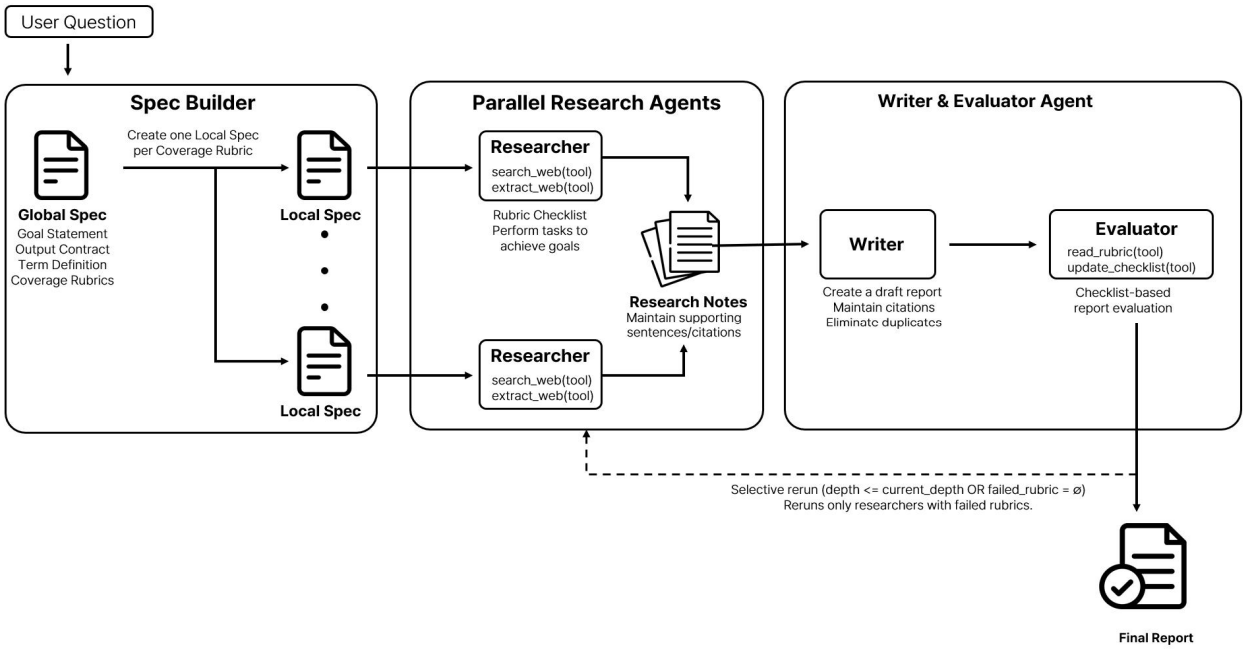

그림 1. Rubric checklist 기반 병렬 조사 프레임워크 개요
Fig. 1. Overview of a Rubric checklist-based parallel research framework

그림 1의 좌측에서 보듯이, 명세 빌더는 (1) 사용자 질의를 기반으로 보고서 전반의 기준점인 전역 명세, (2) 전역 명세의 루브릭을 연구 에이전트의 작업 단위로, 규정하는 지역 명세를 생성한다. 이를 통해 후속 모듈은 반복마다 즉흥적으로 탐색 방향을 재정의하는 대신, 사전에 고정된 명세를 기준으로 수집/통합/서술을 수행한다. CheckEval[22]이 제안하듯 점수 기반 루브릭은 해석이 모호한 반면, 체크리스트는 누락을 명시적으로 식별할 수 있으므로 미충족 루브릭을 선별하여 재실행을 트리거 해야하는 본 프레임워크의 요구에 적합하다고 판단하여 체크리스트 기법을 채택하였다.

정리하면 명세 빌더의 입력과 기능은 다음과 같다. 입력으로는 사용자 질의 Q가 주어지며, 출력으로는 전역 명세와 각 루브릭에 대응되는 지역 명세 집합이 생성된다. 이후 병렬 연구 에이전트는 지역 명세를 작업 계획으로 활용하고, 평가 에이전트는 루브릭의 체크리스트를 기준으로 초안을 판정함으로써, 명세 기반의 목표 충족 루프가 일관되게 유지한다.

3.3.2 병렬 연구 에이전트

병렬 연구 에이전트(Parallel research agents)는 전역 명세의 각 루브릭에 대응하는 Local Spec을 1:1로 할당받아 실행되는 모듈이다. 각 연구 에이전트는 LLM의 임의적 쿼리 생성 및 수집 대신 자신에게 할당된 지역 명세의 루브릭 체크리스트를 충족시키는 것을 목표로 자료를 수집하고, 해당 목표 달성에 기여하지 않는 정보의 확장을 억제하도록 설계되는 ReACT[27] Style 에이전트이다. 또한 연구 에이전트들은 병렬로 수행됨으로써, 동일한 총 조사량을 직렬로 처리하는 것 보다 전체 조사 지연(Latency)를 감소 시킬 수 있다는 이점을 가진다.

연구 에이전트의 주요 출력은 연구 노트(Research note)로, 이는 후속 단계의 작성자 에이전트가 보고서 초안을 구성하는데 사용하는 중간 산출물이다. 연구 노트는 지역 명세의 필수 산출물(Deliverables)를 포함하도록 지시되며, 연구 노트의 내용은 (1) 요약된 핵심 내용, (2) 이를 지지하는 근거 문장(또는 발췌 스니펫), (3) 해당 근거의 출처 URL을 함께 기록하도록 프롬프트를 설계하였다. 이와 같이, 연구 노트에 근거 문장과 출처 URL을 함께 기록함으로써, 작성자 에이전트가 보고서 서술 시 해당 근거를 명시적으로 인용·연결할 수 있도록 한다. 또한 반복 보강 단계에서는 평가 에이전트의 피드백(미충족 체크리스트 항목 및 보

강 지시)을 지역 명세에 반영하여, 해당 루브릭에 대해서만 추가 자료 수집을 수행하고 '보강 연구 노트(Revision note)'를 생성한다. 보강 연구 노트는 누락된 체크리스트 항목을 보완할 수 있게 작성자에게 추가 입력으로 제공된다.

정리하면, 병렬 연구 에이전트는 (1) 지역 명세 기반으로 목표를 한정하고, (2) 체크리스트 충족을 목표로 필요한 자료를 수집하며, (3) 필수 산출물 중심의 구조화된 연구 노트를 생성함으로써, 보고서 생성 과정 전반에서 목표 누락을 줄이기 위한 근거 기반 입력을 제공한다. 나아가 (4) 반복 보강 단계에서는 미충족 루브릭에 한해 보강 연구 노트를 생성하여, 초안 갱신이 누락 항목을 중심으로 이루어지도록 지원한다.

3.3.3 작성자

작성자(Writer)는 병렬 연구 에이전트가 생성한 연구 노트를 통합하여, 장문 보고서 형태의 초안을 생성/갱신하는 모듈이다. 작성자의 입력은 (1) 보고서의 목적/범위/출력 조건을 규정하는 output_contract, (2) 모든 연구 에이전트의 연구 노트 집합, (3) 반복 보강 단계에서의 이전 보고서 초안 $Draft_{d-1}$로 구성된다. 작성자는 이러한 입력을 바탕으로 최종 출력물인 보고서를 생성한다. 작성자는 특히 전역 명세의 output_contract에 명시된 대상 독자(Audience), 출력 언어(Output_language), 필수 산출물(Deliverables)을 준수하도록 설계되며, 연구 노트에 포함된 근거 문장과 출처 URL을 활용하도록 하여, 핵심 주장의 근거 및 인용을 포함하도록 유도한다.

반복 depth >= 2의 반복 보강 단계에서 작성자 에이전트는 초안을 상태로 유지하며 점진적으로 개선하는 초안 중심 갱신 방식을 따른다. 이 단계에서는 이전 초안 $Draft_{d-1}$을 기준 텍스트로 유지한 채, d-1 단계의 평가 단계에서 미충족(FAIL)으로 판정된 루브릭에 대응하는 연구 에이전트의 보강 노트를 추가 입력으로 받아 부족한 항목을 보완한다. 이 과정에서 연구 에이전트는 전체 문서를 매 반복마다 새로 생성하기 보다, 실패 루브릭을 기반으로 내용을 보강하여 $Draft_d$를 산출한다. 이러한 설계는 초안의 구조적 일관성을 유지하면서도, 누락된 요구 항목을 반복적으로 채우는 방향으로 보고서를 수렴시키는 것에 목적이 있다.

3.3.4 평가 에이전트

평가 에이전트(Evaluator agent)는 생성된 보고서 초안을 전역 명세(Global spec)에 정의된 체크리스트로 평가하는 모듈이다. 평가 에이전트의 입력은 현재 반복 단계의 초안 $Draft_d$와 전역 명세에 속한 체크리스트 형태의 루브릭이며, 출력은 각 체크리스트에 대한 이진 판정(is_satisified)와 미충족 항목에 한해 보강을 위한 피드백(Feedback)이 제공된다. 이러한 평가와 피드백은 단순한 평가를 넘어, 연구 에이전트가 다음 반복에서 어떤 관점을 갖고 연구 노트를 보완해야 하는지에 대한 근거로써 제공된다.

3.4 FAIL 루브릭 평가 재실행 알고리즘

3.4.1 초기 단계

초기 단계(Depth 1)는 사용자 질의 Q로부터 전역 목표와 평가 기준을 고정하고, 자료 수집을 수행해 첫 번째 보고서 초안 $Draft_1$을 생성한 뒤, 미충족 요구사항을 식별하는 단계이다. 그림 2의 Depth 1에 나타난 바와 같이, 명세 빌더는 Q를 입력으로 전역 명세를 생성하며, 이는 보고서의 목적 및 산출물 계약과 함께 목표 요구사항을 루브릭/체크리스트 형태로 명세화한다. 이후 전역 명세 내에 속한 루브릭에 대응되어 생성된 지역 명세가 병렬 연구 에이전트의 입력이 된다.

각 병렬 연구 에이전트는 지역 명세의 체크리스트를 충족하기 위한 자료를 수집/정리하여 연구 노트를 산출하고, 작성자 에이전트는 모든 연구 노트를 통합하여 $Draft_1$을 생성한다. 마지막으로 평가 에이전트는 $Draft_1$를 입력으로 체크리스트 항목별 충족 여부 및 피드백을 산출하여, 다음 반복 단계에서의 선택적 재실행 대상과 보강 방향을 결정 기준으로 사용된다.

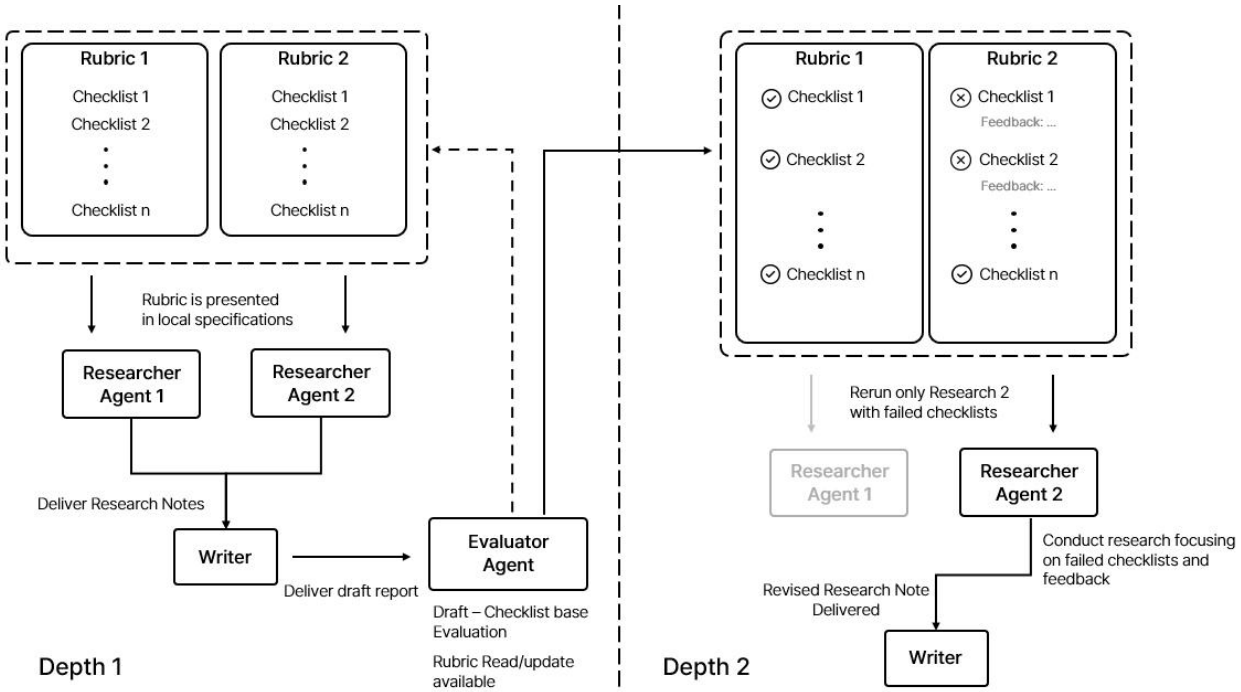

그림 2. FAIL 루브릭 평가 기반 선택적 재실행 알고리즘의 전체 흐름

Fig. 2. Overall workflow of the selective rerun algorithm based on FAIL rubric evaluation

3.4.2 반복 단계

반복 단계(Depth d≥ 2)에서는 초기 평가에서 도출된 미충족 루브릭을 중심으로 초안을 점진적으로 보강하는 단계이다. 그림 2의 Depth 2에 나타난 바와 같이, 이전 단계에서 산출된 FAIL 루브릭 집합 F_{d-1}에 포함된 루브릭에 대해서만 해당 연구 에이전트를 선택적으로 재실행하며, 이때 평가 에이전트가 제공한 체크리스트 항목별 피드백을 입력으로 함께 전달하여 보강 방향을 명시한다. 재실행된 연구 에이전트는 피드백에 기반해, 부족한 근거와 내용을 추가로 수집/정리하여 보강된 연구 노트를 생성하고, 작성자 에이전트는 이전 초안 $Draft_{d-1}$과 보강된 연구 노트를 입력으로 받아 $Draft_d$를 생성한다. 이 과정은 문서를 매시도마다 새로 작성하기보다 기존 초안 구조를 유지한 채 누락된 요구사항을 보완하는 방식으로 수행된다. 결과적으로 $Draft_d$는 $Draft_{d-1}$의 내용을 기반으로 하면서도, 실패 항목을 중심으로 보강된 개선본으로 수렴한다. 이후 평가 에이전트는 갱신된 초안인 $Draft_d$를 다시 d-1 단계에서 실패한 체크리스트 기준으로 판정하여 FAIL루브릭 집합 F_d를 갱신한다. 반복은 (1) 모든 루브릭이 통과되거나, (2) 사전에 설정한 최대 반복 깊이 max_depth에 도달하는 경우 종료된다.

IV. 실 험

본 절에서는 제안 프레임워크가 장문 심층 연구 과정에서 발생하는 요구사항 누락(Coverage gap)과 컨텍스트 드리프트를 얼마나 완화하는지 정량적으로 평가한다. 실험은 (1) 질문 q만 제공되는 환경 (2) 질문 q와 함께 ResearchQA[11]의 평가 루브릭 R을 제공하여 프레임워크 내/외부 루브릭 정렬 불일치(Alignment gap) 요인을 최소화한 조건을 평가한다. 모든 조건은 동일한 검색 환경과 동일한 평가 프로토콜을 사용하며, 공정한 비교를 위해 문제당 외부 검색/도구 호출 수, judge 모델 및 프롬프트/디코딩 설정을 동일하게 통제한다.

4.1 데이터셋

ResearchQA[11]는 학술 survey 논문을 기반으로 구축된 장문 연구 질의응답(long-form scholarly QA) 벤치마크로, 다양한 학문 분야에서 "연구자가 실제

로 묻는 수준의 질문"에 대한 장문 응답을 평가하는 것을 목표로 한다. ResearchQA는 총 21.4K개 질의와 160K개 루브릭 항목을 포함하며, 7개 연구 도메인(예: 의생명/보건, 생명·지구과학, 공학·컴퓨터과학 등)으로 구성된다. 각 샘플은 (i) 연구형 질문 q와 (ii) 해당 질문의 응답이 충족해야 할 질의-특화 루브릭 항목 집합 $R = r_1, r_2, ..., r_m$으로 구성되며, 루브릭 항목은 핵심 개념의 설명, 비교 관점 제시, 한계 기술, 핵심 문헌 인용 등 "답변에 반드시 포함되어야 할 요구사항"을 구체적으로 명시한다.

ResearchQA의 루브릭 항목 R을 "질문이 요구하는 핵심 요소들의 체크리스트"로 해석하고, 모델이 생성한 장문 응답이 이 항목들을 얼마나 충족하는지를 Overall Rubric Score로 측정한다. 또한 실험은 ResearchQA의 평가 비용을 고려하여 7개 상위 연구 도메인별 15개씩 균등 샘플링하여 총 105개 문제를 사용한다. 이러한 균등 샘플링은 도메인별 난이도 및 성능 편차 가능성을 고려하여 특정 분야의 비중에 의해 방법 간 비교 결과가 좌우되는 현상을 줄이기 위해 설계하였다. 따라서 본 실험 결과는 ResearchQA 전체 분포에 대한 절대적 성능 추정이라기보다, 도메인 편향을 완화한 조건에서 수행한 상대 비교 결과로 해석하는 것이 적절하다.

4.2 평가 지표

평가 지표는 ResearchQA[11]의 공식 평가 프로토콜을 따른다. 구체적으로, 각 질문 q에 대해 생성된 장문 응답 A가 ResearchQA가 제공하는 루브릭 항목 집합인 $R = r_1, r_2, ..., r_k$ 을 얼마나 충족하는지를 ORS(Overall Rubric Score)로 측정한다. ORS는 응답이 체크리스트(루브릭)의 각 요구사항을 어느 정도 충족했는지를 정량화한 값으로, 본 연구에서 정의한 "목표 달성률"과 동일한 관점의 평가 지표에 해당한다. ResearchQA는 답변과 루브릭 항목을 함께 입력으로 받아, 각 루브릭 항목에 대해 "Not at all, Barely, Moderately, Mostly, Completely"의 5단계 라벨 중 하나를 출력하도록 LLM judge를 프롬프트하며, 이를 0~1의 수치 점수로 변환 및 정규화 후 점수 평균으로 정의한다. 최종 성능은 평가 대상 질문 집합 Q에 대해 ORS(q)으로 평가한다. 또한, 실험 조건 간 공정한 비교를 위해, 모든 조건에서 동일한 LLM judge(gpt-4o-mini), 동일 프롬프트/출력 포맷, 동일 디코딩 설정으로 평가한다.

4.3 실험 조건

실험은 제안 프레임워크의 효과를 (1) 목표 추론(내부 명세 생성)이 포함된 end-to-end 환경과, (2) 평가 목표가 외부에서 명시된 환경으로 분리하여 분석하기 위해 두 가지 입력 조건을 설정한다. 두 조건은 각각 동일한 질문 q, 검색 환경, 평가 프로토콜을 사용하며, 평가 루브릭 R의 제공 여부를 차이로 둔다.

4.3.1 Cond-A: Question-only

Cond-A에서는 입력으로 질문 q만 제공하며, 시스템은 q로부터 내부적으로 Global Spec을 생성하여 루브릭/체크리스트를 구성한다. 이때 내부적으로 생성된 Spec이 ResearchQA가 제공하는 평가 루브릭 R과 완전히 일치한다는 보장이 없으므로, 평가 단계에서 루브릭 간 정렬 불일치(Alignment gap)가 성능에 영향을 줄 수 있다. 따라서 Cond-A는 "목표가 암묵적인 실제 사용 환경"을 모사하며, 제안 프레임워크가 목표를 추론하고(End-to-end) 이를 충족하는 능력을 종합적으로 측정하는 조건으로 사용한다.

4.3.2 Cond-B: Question + Rubric

Cond-B에서는 입력으로 질문 q와 함께 ResearchQA의 평가 루브릭 R을 제공한다. 이는 시스템이 참조해야 할 목표 요구사항을 외부에서 명시적으로 고정함으로써, Cond-A에서 발생할 수 있는 루브릭 간 정렬 불일치 요인을 최소화한다. 따라서 Cond-B는 "목표 정렬이 확보된 조건"에서 제안 프레임워크의 구조적 효과, 즉 - 루브릭 단위 병렬 분해, 체크리스트 평가 기반 FAIL 루브릭 선택적 재실행, 초안 보강 루프 - 가 목표 달성 성능에 미치는 영향을 분리하여 측정하는 조건으로 사용된다.

4.4 비교 방법(Baselines) 및 Ablations

제안 프레임워크의 효과를 검증하기 위해, 명세(spec)·체크리스트 기반 평가 - 선택적 재실행 루프가 장문 보고서 생성의 목표 달성률(Overall rubric score)에 미치는 영향을 중심으로 - 비교 실험을 구성한다. 모든 조건에서 동일한 입력 질의를 사용하며, 조건 간 차이는 (i) 체크리스트 적용 여부, (ii) 사용 모델의 규모, (iii) 반복 깊이(Depth)로 제한한다. 또한 검색 환경의 공정성을 확보하기 위해, 모든 방법에서 외부 검색 도구 호출 수를 문제당 총 100회로 제한하며, 검색엔진은 DuckDuckGo[28]를 사용한다.

비교 기준으로는 시스템 수준 baseline과 동일 프레임워크 내 ablation을 함께 사용한다. 시스템 baseline으로는 GPT-Researcher를 채택하였다. GPT-Researcher는 웹 탐색, 정보 수집, 반복적 통합 작성을 수행하는 공개 Deep Research형 장문 생성 프레임워크로서, 제안 방법과 구조적으로 가장 직접적인 비교가 가능하다. 본 연구의 목적은 다양한 외부 프레임워크 간 포괄적 성능 비교보다는, 유사한 반복 기반 구조에서 checklist rubric 기반 선택적 재실행 메커니즘의 효과를 검증하는 데 있으므로, 대표 시스템 baseline으로 GPT-Researcher를 사용하였다. 아울러 제안 요소의 기여를 보다 직접적으로 분석하기 위해 동일 프레임워크 내에서 체크리스트를 제거한 w/o checklist 조건을 함께 비교하였다.

모델은 실제 배포에서 자주 선택되는 소/중/대형의 세 크기의 모델를 대표하도록 구성하였다. 소형 모델은 비용·지연 제약이 큰 환경을 대표하면서도 툴 사용 및 장문 생성 안정성이 높은 Qwen3-8B를 사용하였다. 중·대형 티어는 동일 계열 내 용량 변화에 따른 영향을 비교할 수 있도록 gpt-oss-20b와 gpt-oss-120b를 채택하였다. 이 선택은 동일 패밀리 스케일링 자체를 정밀 추정하기보다는, 제안한 제어 메커니즘이 서로 다른 용량 티어에서도 유효하게 작동하는지와, 반복 깊이에 따른 효과가 어떻게 달라지는지를 평가하기 위한 설계이다.

4.4.1 System baseline

1) GPT-Researcher(Deep Research)

GPT-Researcher[3]의 Deep Research 모드는 tree-like exploration(depth×breadth) 형태로 주제를 재귀적으로 확장하며, 각 depth에서 다수의 검색 경로를 생성·병렬 실행(concurrent processing)하고, 수집된 컨텍스트를 통합해 보고서를 생성한다는 점에서 "병렬 수집 → 통합 작성"이라는 큰 흐름이 본 연구의 프레임워크와 구조적으로 유사하다. 또한 depth가 증가할 때의 탐색 구동은 이전 단계의 research goal과 follow-up questions를 결합해 다음 질의를 구성하는 방식으로 이루어지며, 이는 본 연구가 FAIL 루브릭의 feedback으로 다음 반복을 구동하는 것과 대비되는 핵심 차이점이다. 설정은 프레임워크에서 권장하는 기본 설정인 (depth=2, breadth=4)를 사용한다.

4.4.2 Ablations

제안 프레임워크의 핵심 설계 요소인 체크리스트 기반 평가 - 선택적 재실행 루프의 기여를 분리하여 확인하기 위해, 체크리스트를 제거한 ablation 설정을 추가한다. w/o checklist는 전역 명세(Global spec)의 루브릭 및 평가자(Evaluator)를 통한 체크리스트 판정·피드백 생성, 그리고 FAIL 루브릭 선택적 재실행을 모두 사용하지 않는 조건이다. 즉, 시스템은 사용자 질의로부터 보고서 초안을 단일 패스 $d = 1$로 생성하며, 반복 보강을 수행하기 위한 "실패 신호"가 존재하지 않으므로 $d > 1$의 반복 실행은 정의하지 않는다.

구체적으로 w/o checklist는 (1) 사용자 질의 Q로부터 문장 형태의 연구 질문(Research Question)을 구성한 뒤, 병렬 연구 에이전트가 각각의 연구 질문을 수집/정리하여 작성자 에이전트가 이를 통합하여 보고서를 생성하는 파이프라인으로 동작한다. 이때 연구 단계는 루브릭 단위 분해 및 항목 충족을 목표로 하는 통제가 없으므로, 수집·정리 과정은 모델의 일반적인 질의 확장 및 요약 전략에 의존한다. 본 ablation은 "루브릭/체크리스트가 실행을 구동할 때의 이득(FAIL 기반 보강)"을 외부 시스템 baseline과 구분되는 형태로 내부 통제 비교하기 위한 기준선으로 사용된다.

표 3. 실험 결과
Table 3. Experimental results

Setting	Cond-A		Cond-B	
	Overall rubric score(%)	Std(%)	Overall rubric score(%)	Std(%)
GPT-Researcher				
Qwen3-8B(d=2, b=4)	71.927	18.305	93.364	6.360
GPT-OSS-20B(d=2, b=4)	81.721	12.152	95.585	5.354
GPT-OSS-120B(d=2, b=4)	86.299	11.843	97.542	4.169
Ablations(w/o checklist)				
Qwen3-8B(d=1)	72.008	18.391	93.298	6.833
GPT-OSS-20B(d=1)	78.807	13.909	94.182	6.488
GPT-OSS-120B(d=1)	82.996	16.152	95.192	10.632
Ours(d=1)				
Qwen3-8B	77.653	13.352	94.623	5.625
GPT-OSS-20B	78.711	14.143	94.970	6.341
GPT-OSS-120B	82.459	13.873	97.766	4.185
Ours(d=2)				
Qwen3-8B	78.740	15.461	94.370	5.419
GPT-OSS-20B	79.390	14.167	94.663	6.581
GPT-OSS-120B	83.095	14.549	97.607	3.857
Ours(d=3)				
Qwen3-8B	78.343	15.027	95.428	7.438
GPT-OSS-20B	77.900	14.065	94.639	7.559
GPT-OSS-120B	80.419	14.271	97.295	3.981

4.4.3 Proposed: Ours(d∈{1,2,3})

제안 방법(Ours)는 (1)전역/지역 명세 생성, (2)병렬 연구 수행, (3) 초안 작성, (4) 평가, (5) 선택적 반복 루프 기반의 프레임워크로, 모든 설정에서 검색 환경은 동일하게 통제하며, 반복 횟수인 depth는 1,2,3회로 나누어서 실행한다.

V. 결 과

5.1 Cond-A: Question-only

Cond-A는 입력으로 질문 Q만 제공되는 설정으로, 시스템은 질문으로부터 내부적으로 루브릭/체크리스트를 생성한다. 이때 내부 루브릭이 ResearchQA의 평가 루브릭과 완전히 일치하지 않을 수 있으며, 이러한 루브릭 불일치(alignment gap)가 목표 달성률(ORS)에 영향을 미칠 수 있다. 실험 결과는 표 3의 좌측에 명시되어있다.

표 3과 표 4의 GPT-Researcher 결과를 보면, Cond-A에서는 모델 규모가 커질수록 ORS가 뚜렷하게 상승한다. 즉, Qwen3-8B에서 71.93%인 ORS가 GPT-OSS-20B에서 81.72%, GPT-OSS-120B에서 86.30%까지 증가한다. 이는 질문만 주어진 상황에서 대형 모델이 질문에 내재된 요구사항을 더 잘 추론하거나, 학술 보고서에서 기대되는 구성 요소를 보다 일관되게 채워 넣음으로써 평가 루브릭과의 간접적 정렬이 강화되는 효과로 해석할 수 있다.

이를 제안 방법(Ours d=2)와 비교해보면, Qwen3-8B에서는 Ours(d=2)가 78.74%로 GPT-Researcher의 71.93% 대비 +6.81 p.p. 향상되었고, 표준편차도 18.31% → 15.46%로 감소하여 목표 달성의 안정성까지 개선되는 경향을 보였다. 반대로 GPT-OSS-20B 및 GPT-OSS-120B에서는 Ours(d=2)가 각각 79.39%, 83.10%로 GPT-Researcher(81.72%, 86.30%) 대비 -2.33 p.p., -3.20 p.p. 낮았다. 이는 중·대형 모델에서 GPT-Researcher의 tree-like 탐색 및 통합 작성이 이미 높은 목표 달성률을 확보하고 있어, Cond-A에서 내부 spec/체크리스트를 추가하는 것이 외부 평가 루브릭(ResearchQA) 기준 ORS로는 직접적인 이득으로 이어지지 않을 수 있다.

표 4. 주요 비교쌍에 대한 ORS 평균 및 paired t-test p-value
Table 4. Mean ORS and paired t-test p-values for the main comparison pairs

COMPARISON	A_ORS(%)	B_ORS(%)	P_VALUE
Cond_A			
Qwen3-8B Ablation(d=1) vs Ours(d=2)	72.008	77.653	0.000191
GPT-OSS-20B Ablation(d=1) vs Ours(d=2)	78.807	78.711	0.928266
GPT-OSS-120B Ablation(d=1) vs Ours(d=2)	82.996	82.459	0.731
Qwen3-8B Ours(d=2) vs Ours(d=3)	78.74	78.343	0.716542
GPT-OSS-20B Ours(d=2) vs Ours(d=3)	79.39	77.900	0.263757
GPT-OSS-120B Ours(d=2) vs Ours(d=3)	83.095	80.419	0.043735
Cond_B			
Qwen3-8B Ablation(d=1) vs Ours(d=2)	93.298	94.623	0.059212
GPT-OSS-20B Ablation(d=1) vs Ours(d=2)	94.182	94.97	0.26847
GPT-OSS-120B Ablation(d=1) vs Ours(d=2)	95.192	97.766	0.017257
Qwen3-8B Ours(d=2) vs Ours(d=3)	94.37	95.428	0.171144
GPT-OSS-20B Ours(d=2) vs Ours(d=3)	94.663	94.639	0.975838
GPT-OSS-120B Ours(d=2) vs Ours(d=3)	97.607	97.295	0.469245

체크리스트의 기여를 w/o checklist ablation과 비교하면, Cond-A에서는 모델별로 통계적 양상이 상이하게 나타났다. Qwen3-8B의 경우 Ablation(d=1) 72.01% 대비 Ours(d=1) 77.65%로 평균 ORS가 +5.65 p.p. 증가하였고, 이 차이는 paired t-test에서 통계적으로 유의하였다(p=0.000191). 반면 GPT-OSS-20B는 78.81% → 78.71%(-0.10 p.p., p=0.928266), GPT-OSS-120B는 83.00% → 82.46%(-0.54 p.p., p=0.731376)로, 두 조건 모두 유의한 차이가 확인되지 않았다. 따라서 Cond-A에서는 체크리스트 기반 제어의 효과가 소형 모델(Qwen3-8B)에서는 뚜렷하게 관찰되었으나, 중·대형 모델에서는 평균 차이가 작고 문항별 분산이 존재하여 통계적으로 확정적인 개선 근거는 확보되지 않았다. 이에 따라 본 결과는, Cond-A와 같이 내부 spec과 외부 평가 루브릭 간 alignment gap이 존재할 수 있는 환경에서 체크리스트 기반 제어의 효과가 모델 규모와 정렬 정도에 따라 달라질 수 있음을 보인다.

마지막으로 반복 깊이 d에 따른 효과를 보면, 세 모델 모두 기술통계상 d=2 이후 d=3에서 ORS가 낮아지는 경향이 관찰되었다. 다만 paired t-test 결과, d=2와 d=3의 차이가 통계적으로 유의했던 경우는 GPT-OSS-120B(83.10% → 80.42%, p=0.043735)에 한정되었고, Qwen3-8B(78.74% → 78.34%, p=0.716542)와 GPT-OSS-20B(79.39% → 77.90%, p=0.263757)에서는 유의한 차이가 확인되지 않았다. 따라서 반복 증가에 따른 성능 저하는 모든 모델에서 일반적으로 성립하는 결론이라기보다, 일부 설정에서 관찰된 가능성 있는 현상으로 해석할 수 있다.

5.2 Cond-B: Question + Rubric

Cond-B는 질문 q와 함께 ResearchQA가 제공하는 평가 루브릭 R 을 입력으로 제공하는 조건이다. 즉, 시스템이 따라야 할 목표 요구사항이 외부에서 명시적으로 고정되며, Cond-A에서 발생할 수 있는 Spec - 루브릭 정렬 불일치(alignment gap) 요인을 최소화한다. 따라서 Cond-B는 "목표 정렬이 확보된 조건"에서, 제안 프레임워크의 구조적 효과(루브릭 단위 병렬 분해, 체크리스트 기반 평가 및 FAIL 선택 재실행, 초안 보강 루프) 가 ORS에 미치는 영향을 분리해 관찰하기 위한 설정으로 사용한다. 실험 결과는 표 3의 우측에 명시되어있다.

표 3과 표 4의 결과는 Cond-B에서 ORS가 전반적으로 높은 구간(약 93 - 98%)에 분포함을 보여준다. w/o checklist 대비 Ours(d=1)의 평균 ORS는 세 모델 모두에서 증가했으나, 통계적 유의성은 모델별로 다르게 나타났다. GPT-OSS-120B는 95.192%에서 97.766%로 +2.574 p.p. 증가하였고, 이 차이는 paired t-test에서 통계적으로 유의하였다 (p=0.017). 반면 GPT-OSS-20B는 94.182%에서 94.970%로 +0.788 p.p. 증가하였으나 유의하지 않았다 (p=0.268).

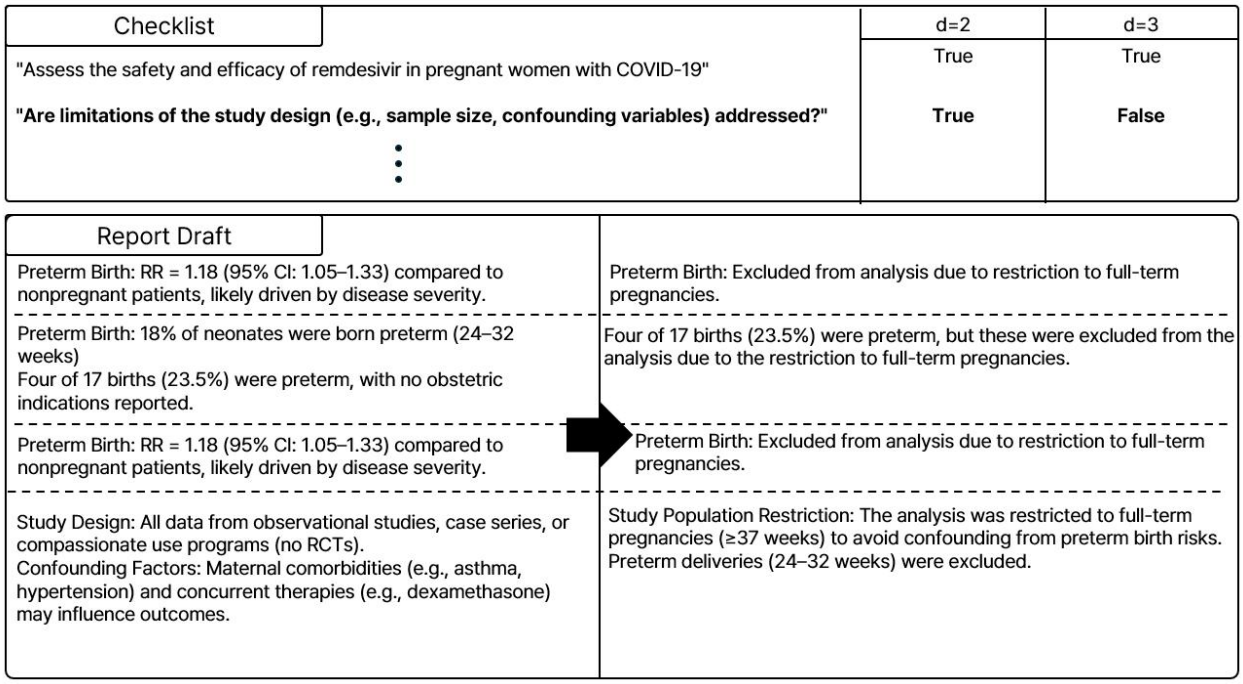

그림 3. d=2에서 d=3으로 반복이 증가할 때 연구 한계 기술이 약화된 사례
Fig. 3. Example of weakened handling of study limitations as iteration increases from d=2 to d=3

Qwen3-8B의 경우 평균 ORS는 93.298%에서 94.623%로 증가하였으며, paired t-test 결과는 경계적 수준에 머물러(p=0.059), 긍정적인 개선 경향을 보였다. 한편 반복 깊이의 효과는 모델 규모에 따라 상이하게 나타나, Qwen3-8B에서는 d=3에서 95.428%로 최고 성능을 보인 반면, GPT-OSS-20B 및 120B에서는 d=1이 각각 94.970%와 97.766%로 최댓값이며, d 증가에 따라 완만한 하락을 보였다. 이는 중·대형 모델의 경우 루브릭이 주어진 상황에서 초기 산출물만으로도 높은 커버리지를 확보하는 경향이 강해 추가 반복이 중복 서술이나 재작성에 따른 부분 회귀로 이어질 수 있음을 암시한다.

마지막으로 Cond-B에서 GPT-Researcher(d=2, b=4)와 Ours(d=2)를 직접 비교하면, 모델 규모에 따라 상대적 우열이 달라진다. 먼저 Qwen3-8B에서는 GPT-Researcher가 93.364% (std 6.360)인 반면, Ours(d=2)는 94.370% (std 5.419)로 +1.006 p.p. 높고 표준편차도 감소하여(6.360→5.419) 목표 달성의 안정성까지 개선되는 경향을 보였다. 반대로 GPT-OSS-20B에서는 GPT-Researcher가 95.585% (std 5.354)로 Ours(d=2) 94.663% (std 6.581) 대비 +0.922 p.p. 우세했으며, 표준편차 역시 더 낮아 GPT-Researcher가 더 일관된 성능을 보였다. GPT-OSS-120B에서도 GPT-Researcher 97.542% (std 4.169)가 Ours(d=2) 97.607% (std 3.857)과 거의 유사한 수준으로, ORS 차이는 +0.065 p.p.에 불과하지만(사실상 동급), Ours(d=2)가 더 낮은 표준편차(4.169→3.857)를 기록해 변동성 측면에서는 소폭 유리했다.

5.3 Case study

정량 결과에서 관찰된 바와 같이, 반복 깊이의 증가가 항상 ORS 향상으로 이어지지는 않았다. 특히 일부 설정에서는 d=2에서 d=3으로 반복을 한 단계 더 수행했을 때 성능이 오히려 소폭 하락하였으며, 이는 초안 갱신 과정에서 내용 정렬이 약화될 가능성을 시사한다. 이를 구체적으로 확인하기 위해, 본 절에서는 d=2와 d=3 초안을 직접 비교하여 반복 보강 과정에서 어떤 유형의 회귀가 발생하는지 사례 수준에서 분석한다.

대표 사례로, "What are the pregnancy and neonatal outcomes observed in hospitalized pregnant women with COVID-19 who received remdesivir treatment?" 질문에 대한 생성 결과를 분석하였다. 이 사례에서

d=2 초안은 관찰연구 중심의 근거, 교란 가능성, 조산 사례 포함 등을 연구의 한계로 정확히 기술하였다. 반면 d=3 초안은 실제로는 존재하지 않는 만삭 임신 제한(Full-term restriction) 을 새로 선언하고 조산 사례를 제외했다고 서술함으로써, 한계를 더 정확히 설명한 것이 아니라 근거와 불일치하는 방법론적 재서술로 덮어썼다. 따라서 d=3는 연구 설계의 한계를 적절히 다루었다고 보기 어렵다.

그림 3는 d=2에서 d=3으로 반복이 증가하면서 연구 설계의 한계에 대한 서술이 약화된 사례를 보여준다. 이 사례는 반복 증가에 따라 발생할 수 있는 두 가지 문제를 보여준다. 첫째, 정렬 불일치 누적(Alignment mismatch accumulation) 이다. 하위 초안 수준에서는 유지되던 제한사항이 최종 종합 단계에서 왜곡되면서, evidence와 summary 간의 일관성이 약화되었다. 둘째, 기존 충족 항목의 회귀(Regression of previously satisfied aspects) 이다. d=2에서는 "연구 설계의 한계를 정확히 기술한다"는 측면에서 비교적 안정적이었던 서술이, d=3에서는 오히려 unsupported methodological revision으로 대체되었다. 이는 반복 보강이 항상 누락 보완으로 이어지는 것이 아니라, 경우에 따라서는 이전 단계에서 유지되던 정확한 제약 정보를 덮어쓰며 새로운 오류를 만들 수 있음을 보여준다.

VI. 결론 및 향후 연구

본 연구는 LLM 기반 심층 연구(Deep research) 시스템에서 반복 탐색이 심화될수록 나타나는 목표 누락과 컨텍스트 드리프트 등의 핵심 문제를 완화하기 위한 루브릭/체크리스트 기반 심층 연구 프레임워크를 제안하였다. 프레임워크는 사용자 질의를 명세 및 체크리스트/루브릭으로 구조화하여 미충족 항목만 선택적으로 재실행하는 방식으로 반복을 제어한다. 이를 통해 단순히 탐색 횟수를 늘리는 접근이 아니라, 요구사항 중심의 통제된 반복을 통해 보고서의 목표 달성률을 체계적으로 향상시키는 것을 목표로 하는 프레임워크이다.

ResearchQA 벤치마크 실험 결과는 제안 프레임워크의 효과가 조건 및 모델 규모에 따라 상이하게 나타남을 보여준다. Cond-A(Question-only)와 Cond-B (Question+Rubric) 모두에서, 체크리스트가 없는 설정과 비교했을 때 소형 모델(Qwen3-8B)에서는 상대적으로 뚜렷한 성능 향상 경향이 관찰된 반면, 중·대형 모델(GPT-OSS-20B, GPT-OSS-120B)에서는 개선 폭이 제한적이거나 일부 조건에서 소폭 하락하는 양상이 나타났다. 이러한 결과는 체크리스트 기반 보강 메커니즘의 효과가 모든 설정에서 일관되게 통계적으로 확증되었다고 보기는 어렵지만, 특히 목표 정렬이 비교적 잘 확보된 조건에서는 ORS 개선에 기여할 가능성을 보인다.

또한, 반복 깊이(Depth) 관점에서는 반복 횟수에 따라 성능이 선형적으로 향상되지 않는 점을 확인하였다. 이러한 문제점은 반복을 하면서 (i) 불필요/중복 정보 축적, (ii) 재작성 과정에서 기존 충족 항목의 회귀 문제를 해결할 수 있는 기법이 추가로 연구되어야 함을 나타낸다.

5.3의 사례 분석은 반복 증가가 항상 누락 보완으로 이어지지 않음을 보여준다. d=3에서는 일부 경우 하위 근거와 불일치하는 방법론적 재서술이 최종 종합에 삽입되었으며, 이는 기존에 유지되던 한계 정보의 회귀와 evidence–summary alignment 약화로 연결될 수 있었다.

종합적으로, 본 실험을 통해 루브릭/체크리스트를 단순한 사후 평가 지표가 아니라 반복 탐색을 제어하는 운영 신호(Control signal)로 활용함으로써, 심층 연구 에이전트의 목표 달성률을 개선할 수 있음을 보여준다. 특히 (i) 목표가 명확히 주어지지 않는 환경에서는 소형 모델의 목표 누락을 완화하는 데 효과적이며, (ii) 목표가 외부 루브릭으로 주어진 환경에서는 체크리스트 기반 선택적 보강이 누락 감소에 기여할 수 있음을 확인하였다. 향후 연구에서는 내부 명세와 사용자의 목표 간의 정렬을 강화하는 방법뿐 아니라, 기존 충족 항목의 회귀를 방지하는 검증 절차, 근거–요약 정합성 점검 모듈 등을 현재 프레임워크 내에 추가함으로써 현재 프레임워크의 안정성과 실용성을 더욱 높일 필요가 있다. 또한 반복 깊이를 고정적으로 늘리기보다, 체크리스트 충족도 개선 추이에 따라 반복을 조절하는 적응적 제어 방식도 중요한 후속 과제이다.

References

[1] Y. Huang, Y. Chen, H. Zhang, K. Li, H. Zhou, M. Fang, L. Yang, X. Li, L. Shang, S. Xu, J. Hao, K. Shao, and J. Wang, "Deep Research Agents: A Systematic Examination And Roadmap", arXiv preprint arXiv:2506.18096, Jun. 2025. https://doi.org/10.48550/arXiv.2506.18096.

[2] "Deep research in ChatGPT", https://chatgpt.com/features/deep-research/. [accessed: Feb. 03, 2026]

[3] "GPT Researcher", https://github.com/assafelovic/gpt-researcher. [accessed: Feb. 03, 2026]

[4] "Thinkdepthai/Deep_Research", https://github.com/thinkdepthai/Deep_Research. [accessed: Feb. 03, 2026]

[5] F. Bousetouane, "AI Agents Need Memory Control Over More Context", arXiv preprint arXiv:2601.11653, Jan. 2026. https://doi.org/10.48550/arXiv.2601.11653.

[6] O. Weller, K. Lo, D. Wadden, D. Lawrie, B. Van Durme, A. Cohan, and L. Soldaini, "When do Generative Query and Document Expansions Fail? A Comprehensive Study Across Methods, Retrievers, and Datasets", Findings of the Association for Computational Linguistics: EACL 2024, St. Julian's, Malta, pp. 1987-2003, Mar. 2024. https://doi.org/10.18653/v1/2024.findings-eacl.134.

[7] Y.-J. Lee, S. Kim, B.-K. Lee, M. Moon, Y. Hwang, J. M. Kim, G. Neubig, S. Welleck, and H.-J. Choi, "RefineBench: Evaluating Refinement Capability of Language Models via Checklists", arXiv preprint arXiv:2511.22173, Nov. 2025. https://doi.org/10.48550/arXiv.2511.22173.

[8] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark, "Self-Refine: Iterative Refinement with Self-Feedback", Adv. Neural Inf. Process. Syst. 36 (NeurIPS 2023), New Orleans, LA, USA, Dec. 2023. https://doi.org/10.5555/3666122.3668141.

[9] Z. Wang, X. Wang, S. Lee, and X. Xu, "ARISE: Agentic Rubric-Guided Iterative Survey Engine for Automated Scholarly Paper Generation", arXiv preprint arXiv:2511.17689, Nov. 2025. https://doi.org/10.48550/arXiv.2511.17689

[10] R. Han, Y. Chen, Z. CuiZhu, L. Miculicich, G. Sun, Y. Bi, W. Wen, H. Wan, C. Wen, S. Maître, G. Lee, V. Tirumalashetty, E. Xue, Z. Zhang, S. Haykal, B. Gokturk, T. Pfister, and C.-Y. Lee, "Deep Researcher with Test-Time Diffusion", arXiv preprint arXiv:2507.16075, Jul. 2025. https://doi.org/10.48550/arXiv.2507.16075.

[11] L. S. Yifei, A. Chang, C. Malaviya, and M. Yatskar, "ResearchQA: Evaluating Scholarly Question Answering at Scale Across 75 Fields with Survey-Mined Questions and Rubrics", arXiv preprint arXiv:2509.00496, Aug. 2025. https://doi.org/10.48550/arXiv.2509.00496.

[12] "Gemini Deep Research — your personal research assistant", https://gemini.google/overview/deep-research/. [accessed: Feb. 03, 2026]

[13] "Deep Agents overview", https://docs.langchain.com/oss/python/deepagents/overview. [accessed: Feb. 03, 2026]

[14] "open_deep_research", https://github.com/langchain-ai/open_deep_research. [accessed: Feb. 03, 2026]

[15] S.-Y. Jung and B. Jang, "Research on Infectious Disease AI Agent Using Large Language Models: Focusing on Domestic Infectious Disease Data", Journal of Internet Computing and Services, Vol. 26, No. 2, pp. 73-84, Apr. 2025.

[16] J. Y. Lee and H. L. Kim, "Development of Agentic RAG-based Chatbot for Improving Address Information Accessibility", Journal of Cadastre & Land Information, Vol. 55, No. 2, pp. 71-86, Dec. 2025. https://doi.org/10.22640/lxsiri.2025.55.2.71.

[17] C. Jeong, "A Graph-Agent-Based Approach to Enhancing Knowledge-Based QA with Advanced

RAG", Knowledge Management Research, Vol. 25, No. 3, pp. 99-119, Sep. 2024. https://doi.org/10.15813/kmr.2024.25.3.005.

[18] J.-H. Kim and J.-Y. Kim, "LangGraph-Based Integrated Architecture for CrewAI Multi-Agent Systems", Journal of the Korea Institute Of Information and Communication Engineering, Vol. 29, No. 8, pp. 987-992, Aug. 2025.

[19] Cornell University Center for Teaching Innovation, "Using Rubrics", https://teaching.cornell.edu/teaching-resources/assessing-student-learning/using-rubrics. [accessed: Feb. 03, 2026]

[20] L. Zheng, W.-L. Chiang, Y. Sheng, T. Zhuang, Z. Wu, Y. Zhuang, J. Lin, Z. Li, D. Li, E. Xing, H. Zhang, J. Gonzalez, and I. Stoica, "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena", Advances in Neural Information Processing Systems 36 (NeurIPS 2023), New Orleans, LA, USA, pp. 46595-46623, Dec. 2023. https://doi.org/10.5555/3666122.3668142.

[21] A. Pathak, R. Gandhi, A. U. Nambiappan, S. Jayakumar, A. G. Vasantha, P. Mathur, N. Aggarwal, S. Agarwal, P. K. R., S. Manoharan, N. Krishnaswamy, and K. H. K. N., "Rubric Is All You Need: Improving LLM-Based Code Evaluation With Question-Specific Rubrics" Proc. of the 2025 ACM Conference on International Computing Education Research (ICER '25), Charlottesville, USA, pp. 181-195, May 2025. https://doi.org/10.1145/3702652.3744220.

[22] Y. Lee, J. Kim, J. Kim, H. Cho, J. Kang, P. Kang, and N. Kim, "CheckEval: A reliable LLM-as-a-Judge framework for evaluating text generation using checklists", Proc. of the 2025 Conference on Empirical Methods in Natural Language Processing, Suzhou, China, pp. 15771-15798, Nov. 2025. https://doi.org/10.18653/v1/2025.emnlp-main.796.

[23] Y. Hong, H. Yao, B. Shen, W. Xu, H. Wei, and Y. Dong, "RULERS: Locked Rubrics and Evidence-Anchored Scoring for Robust LLM Evaluation", arXiv preprint arXiv:2601.08654, Jan. 2026. https://doi.org/10.48550/arXiv.2601.08654

[24] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, and J. Schmidhuber, "MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework", Proc. of the International Conference on Learning Representations (ICLR 2024), Vienna, Austria, May 2024.

[25] C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, J. Xu, D. Li, Z. Liu, and M. Sun, "ChatDev: Communicative Agents for Software Development", Proc. of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Bangkok, Thailand, pp. 15174-15186, Aug. 2024. https://doi.org/10.18653/v1/2024.acl-long.810.

[26] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation", arXiv preprint arXiv:2308.08155, Aug. 2023. https://doi.org/10.48550/arXiv.2308.08155.

[27] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models", arXiv preprint arXiv:2210.03629, Oct. 2022. https://doi.org/10.48550/arXiv.2210.03629.

[28] "DuckDuckGo", https://duckduckgo.com/. [accessed: Feb. 03, 2026]

저자소개

김 민 석 (Minseok Kim)

2025년 2월 : 국립금오공과대학교 컴퓨터공학과(공학사)
2025년 7월 : 국립금오공과대학교 컴퓨터 · AI융합공학과 석사과정
관심분야 : Document Retrieval, RAG, Agent, Databases

정 유 철 (Yuchul Jung)

2011년 2월 : 한국과학기술원 (KAIST) 전산학과(공학박사)
2013년 7월 : 한국전자통신연구원 (ETRI) 선임연구원
2017년 8월 : 한국과학기술정보연구원(KISTI) 선임연구원
2022년 2월 : 국립금오공과대학교 컴퓨터공학과 조교수
2022년 2월 ~ 현재 : 국립금오공과대학교 컴퓨터공학부 부교수
관심분야 :생성형 AI, 거대언어모델, Agentic AI, 데이터퓨전, 지식그래프