Assignment 04 — NIST RAG & Agent 실험 리포트

> **작성일**: 2025년 12월 1일
> **프로젝트명**: nist-rag-project (LangSmith)

1. 프로젝트 개요

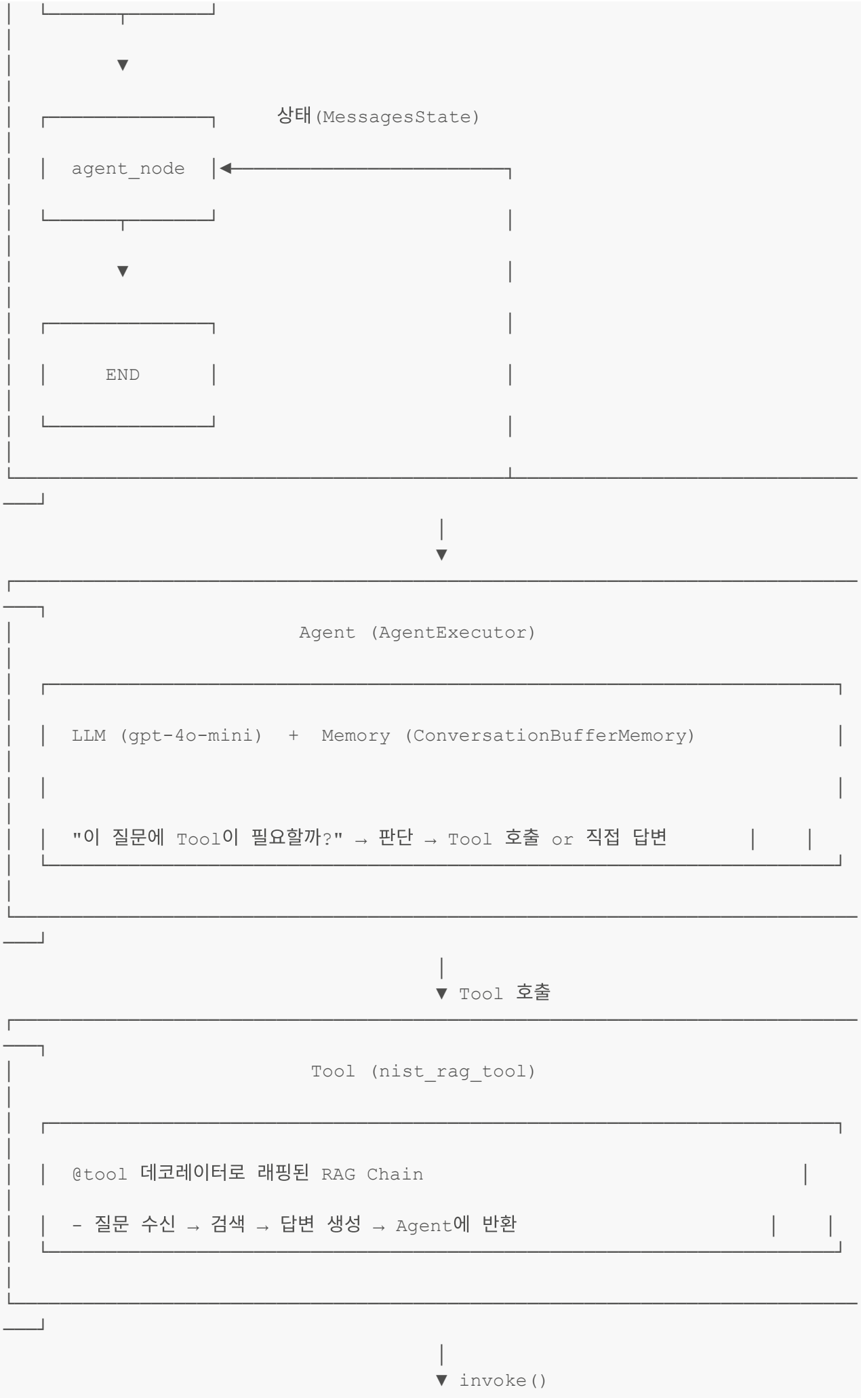
본 과제는 NIST(미국 국립표준기술연구소)에서 발행한 3종의 보안/AI 관련 PDF 문서를 기반으로 **RAG(Retrieval-Augmented Generation)** 시스템과 **Agent 기반 대화형 컨설팅 시스템**을 구축하고, 이를 **LangGraph**로 워크플로우화하여 **LangSmith**에서 모니터링하는 것을 목표로 한다.

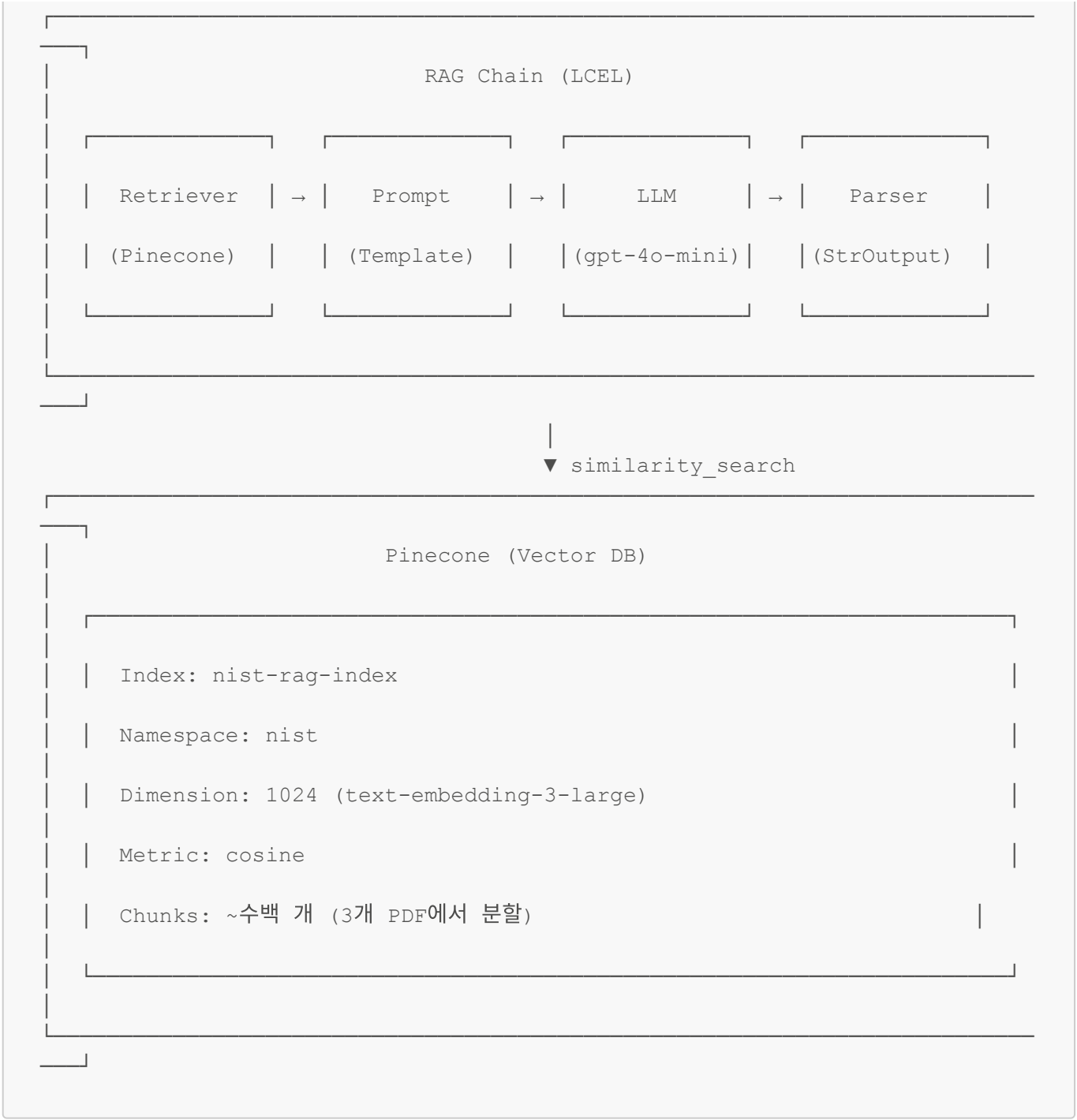
사용 데이터

문서명	약칭	주제
NIST AI Risk Management Framework 1.0	`ai_rmf`	AI 위험 관리 프레임워크
NIST Cybersecurity Framework 2.0	`csf_2_0`	사이버보안 프레임워크
NIST Zero Trust Architecture	`zero_trust`	제로 트러스트 아키텍처

2. 시스템 아키텍처

2.1 전체 구조 다이어그램


상태(MessagesState)
agent_node
END
Agent (AgentExecutor)
LLM (gpt-4o-mini) + Memory (ConversationBufferMemory)
"이 질문에 Tool이 필요할까?" → 판단 → Tool 호출 or 직접 답변
▼ Tool 호출
Tool (nist_rag_tool)
@tool 데코레이터로 래핑된 RAG Chain
- 질문 수신 → 검색 → 답변 생성 → Agent에 반환
▼ invoke()



2.2 구성요소 요약

계층	구성요소	역할
모니터링	LangSmith	전체 실행 과정 트레이싱, 성능 분석
워크플로우	LangGraph	상태 기반 실행 흐름 관리, 노드 간 전이
판단 주체	Agent	Tool 사용 여부 판단, 대화 기억, 최종 답변 생성
기능 단위	Tool	RAG Chain을 호출 가능한 함수로 래핑
검색+생성	RAG Chain	Retriever → Prompt → LLM → Parser 파이프라인
저장소	Pinecone	문서 청크의 벡터 임베딩 저장 및 유사도 검색

3. 실험 결과

3.1 RAG Chain 단독 실행 (01_nist_rag.ipynb)

5개의 테스트 질문에 대해 RAG Chain을 실행한 결과:

#	질문	참조 문서	결과 요약
1	AI RMF의 코어 기능을 간단히 설명해줘	AI RMF p.24	GOVERN, MAP, MEASURE, MANAGE 4가지 기능 설명
2	CSF 2.0에서 식별(Identify) 기능의 주요 항목은?	CSF 2.0 p.19	ID.AM(자산관리), ID.RA(위험평가), ID.IM(개선) 정확히 언급
3	Zero Trust에서 자산 접근 제어의 원칙은?	Zero Trust p.9	최소 접근, 지속적 인증/승인 원칙 설명
4	AI 위험 평가에서 데이터 거버넌스 관련 지침은?	AI RMF p.27, 43	개인정보 보호, 다양성 팀 구성, 역할 정의 등 상세 설명
5	CSF 2.0의 보호(Protect) 영역에서 권장 통제는?	CSF 2.0 p.7	PROTECT 영역의 접근 제어, 데이터 보안 등 설명

평가: 모든 질문에 대해 **정확한 문서/페이지 출처**를 언급하며 답변함. 한국어 답변 품질 양호.

3.2 Agent + LangGraph 실행 (02_nist_rag_agent.ipynb)

시나리오 1: AI Risk Management 컨설팅 (3턴 대화)

- Agent가 `nist_rag_tool`을 호출하여 AI RMF 문서 기반 답변 생성
- Memory를 통해 이전 대화 맥락 유지

시나리오 2: Zero Trust 전환 컨설팅 (3턴 대화)

- Zero Trust 원칙, 자산 접근 제어, 지속적 검증에 대한 연속 질의
- 각 턴에서 Tool 호출 후 근거 기반 답변 생성

LangGraph 실행:

```
graph.invoke({"messages": [{"role": "user", "content": "AI RMF의 핵심 목표는?"}]})
```

- 상태(`MessagesState`) 기반으로 agent_node 실행 → END 도달
- LangSmith에서 `nist_agent_graph` run_name으로 트레이스 확인 가능

---

4. 기술적 세부사항

4.1 임베딩 및 인덱스 설정

```
embeddings = OpenAIEmbeddings(model="text-embedding-3-large",
dimensions=1024)
# Pinecone Index: nist-rag-index (1024차원, cosine metric, AWS us-east-1)
```

4.2 청킹 전략

```
splitter = RecursiveCharacterTextSplitter(
    chunk_size=900,
    chunk_overlap=150,
    separators=["\n\n", "\n", " "]
)
```

4.3 LangSmith 트레이싱 설정

```
os.environ["LANGSMITH_TRACING"] = "true"
os.environ["LANGSMITH_PROJECT"] = "nist-rag-project"
```

- 모든 RAG Chain 및 Agent 호출에 `tags`와 `run_name` 부여
- 예: `tags=["nist", "rag", "tool"]`, `run_name="nist_rag_tool"`

---

5. 자기 평가

5.1 잘 된 점

1. **정확한 출처 인용**: RAG Chain이 문서명/페이지 번호를 포함한 답변 생성에 성공
2. **시스템 통합**: RAG → Tool → Agent → LangGraph 계층 구조가 깔끔하게 연결됨
3. **LangSmith 트레이싱**: 모든 실행이 프로젝트별로 태깅되어 추적 가능
4. **다국어 처리**: 영문 PDF 기반이지만 한국어 질의/답변이 자연스럽게 동작

5.2 아쉬운 점

1. **Memory 활용 한계**: ConversationBufferMemory를 사용했으나, 긴 대화에서 컨텍스트 윈도우 초과 가능성 미검증
2. **에러 핸들링 부족**: PDF 로딩 실패, Pinecone 연결 실패 등 예외 상황 처리 미흡
3. **평가 지표 부재**: 답변 정확도, 지연 시간, 토큰 사용량 등 정량적 평가 미실시
4. **단순한 그래프 구조**: 현재 그래프가 `START → agent_node → END` 단일 경로로 분기/루프 없음

5.3 다음에 개선하고 싶은 점

1. **멀티 에이전트 구조**: 문서별 전문 에이전트 + 라우터 에이전트 구성
2. **스트리밍 응답**: `stream()` 메서드를 활용한 실시간 답변 출력
3. **정량적 평가 파이프라인**:
    - 정답 데이터셋 구축 후 Precision/Recall 측정
    - LangSmith API로 지연 시간/토큰 사용량 자동 수집

4. **그래프 고도화**:
    - 조건부 분기 (예: Tool 필요 여부에 따른 경로 분기)
    - 루프 구조 (예: 답변 검증 후 재검색)
5. **Hybrid Search**: BM25 + Dense Embedding 조합으로 검색 품질 향상

---

6. 결론

본 과제를 통해 **LangChain 기반 RAG 시스템**의 기본 구조부터 **Agent를 통한 자율적 Tool 사용**, **LangGraph를 통한 워크플로우 관리**, **LangSmith를 통한 모니터링**까지 LLMOps의 핵심 구성요소를 실습할 수 있었다.

특히 각 계층(RAG Chain → Tool → Agent → Graph → Tracing)이 어떻게 유기적으로 연결되는지 이해하는 것이 가장 큰 학습 포인트였다. 향후 프로덕션 환경에서는 평가 파이프라인 자동화와 그래프 구조 고도화를 통해 더욱 견고한 시스템을 구축할 수 있을 것이다.