

김태훈

# LLM Reply Improvement

RAG LLM 답변 길이 문제 해결 방안

# 목차

## 01 프로젝트 설명

기존 프로젝트 제한: Financial Sellside Analyst Report 요약본에 대한 LLM 답변 길이 부족 현상  
현 프로젝트 방향성: 복/붙 아이디어 활용한 plan-and-execute 형식 개발

## 02 아키텍처 및 작동 원리

워크 플로우1: 사용자 질의 → LLM의 플랜 생성  
워크 플로우2: 생성된 플랜에 대한 RAG 실행  
워크 플로우3: 실행된 RAG에 대한 Summation 진행

## 03 구현 방법

환경 설정: VS Code와 Jupyter Extension을 활용한 개발 환경  
코드 구현: Python 기반의 단계별 라이브러리 호출 및 프로토타입 소스 코드 가이드

## 04 개선 방향

장단점 분석: 코드 제한 분석을 통한 추후 개선 방향 제시

## 01 프로젝트 설명

### AS-IS: RAG 기반 요약 구조의 한계

#### 현재 방식

- RAG(Retrieval-Augmented Generation) 기반 문서 요약 수행
- LLM 출력 길이 제한으로 긴 문서를 한 번에 요약 불가
- 문서 분할 후 반복 요약 방식 적용
  - ex.) 16장 길이의 에세이를 8장으로 요약해줘 → 답변: 1장 이내의 요약본
  - 전체 요약 생성을 위해 최소 n회 반복 수행

#### 주요 문제

- 반복 호출로 인한 운영 비효율 증가
- 정보 손실 리스크
- 전체 맥락 단절 가능성

## 02 아키텍처 및 작동 원리

### TO-BE: Plan-and-Execute: 목차 생성 이후 개별적인 RAG 실행

#### Self-Planning 기반 지능형 RAG 워크플로우:

##### Step 1. [Intent Analysis]:

- 사용자 질의의 의도와 형식을 분석하여 최적화된 리포트 구조(Plan) 설계 (Non-RAG 기반 추론)

##### Step 2. [Adaptive Planning]:

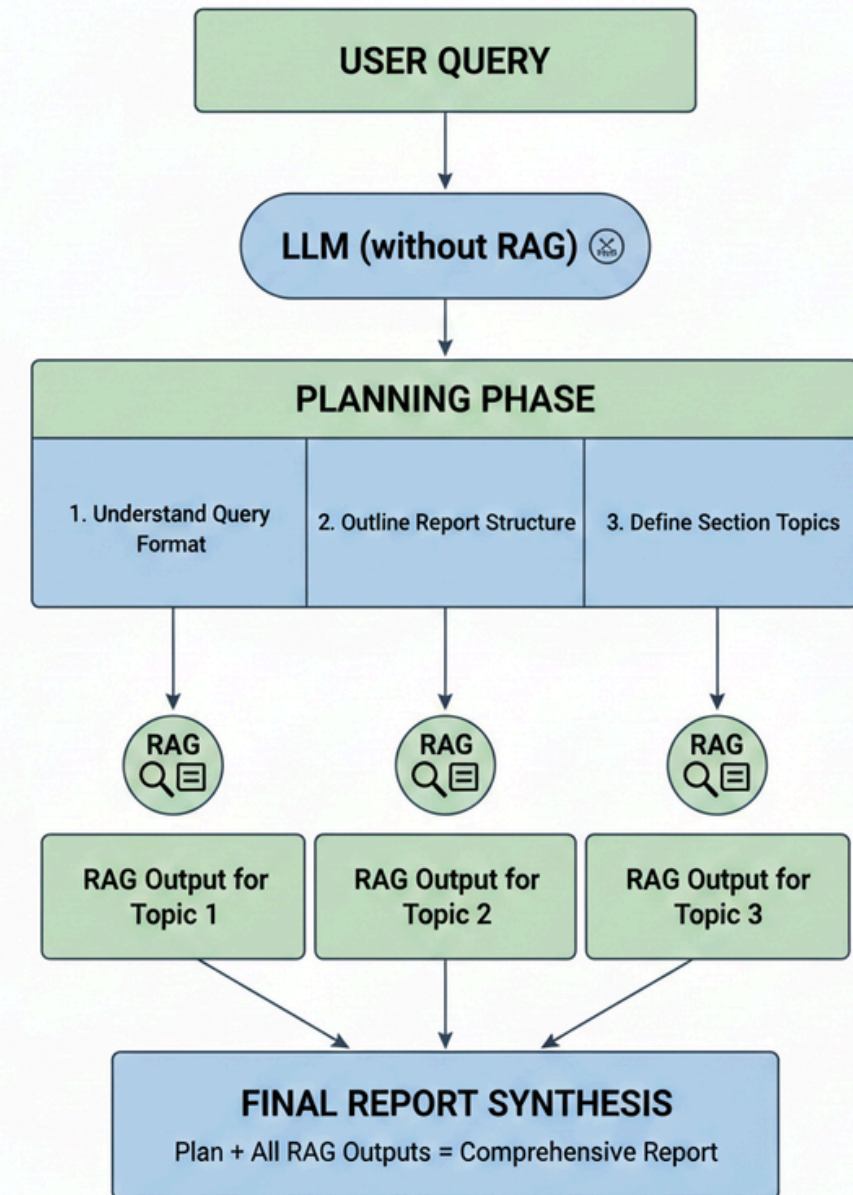
- 생성된 목차별 세부 주제 분할 및 각 섹션에 최적화된 검색 쿼리 생성

##### Step 3. [Iterative RAG Retrieval]:

- 각 목차별 독립적인 RAG 프로세스 가동을 통해 정보 밀도와 정확도 극대화

##### Step 4. [Contextual Synthesis]:

- 분산 생성된 콘텐츠를 통합하여 일관성 있는 최종 리포트 완성



## 03 코드 구현

[https://github.com/KimHoon98/RAG-Implementation/tree/main/3.\)%20PDF%20RAG%20\(LLM%20%2B%20improvement\)%20implementation](https://github.com/KimHoon98/RAG-Implementation/tree/main/3.)%20PDF%20RAG%20(LLM%20%2B%20improvement)%20implementation)

## 04 개선 방향

### RAG Document 인증 / 연관성 확인

LangSmith Trace: 각 VectorStoreRetriever 함수는 8개의 Document를 Fetch하게 됩니다. 다만 이 8개의 모든 Document가 필요한 내용인지, 그리고 연관성이 있는지에 대해서 불확실합니다.

#### 해결 방안:

- Advanced RAG deployment:
  - Advanced RAG의 첫부분: Document의 연관성 확인 및 Priority (우선 사항) 체크를 통해 필요한 Document (Splits)만 추출하도록 연결