RAG(검색증강생성) 실무 심층 가이드

> LLM에 "기억"이 아니라 "근거"를 붙이는 기술

---

1. 왜 RAG인가

대규모 언어모델은 학습 시점에 고정된 파라미터 안에 지식을 압축해 저장합니다. 이 구조는 세 가지 구조적 한계를 낳습니다.

1. **지식 컷오프(Knowledge Cutoff)** — 학습 이후 발생한 사실을 알지 못합니다.
2. **환각(Hallucination)** — 모르는 내용을 확률적으로 그럴듯하게 생성합니다.
3. **비공개 데이터 부재** — 사내 문서, 계약서, 개인 노트는 애초에 학습 대상이 아닙니다.

파인튜닝으로 이를 해결하려는 시도는 비용·갱신 주기·출처 추적 세 측면에서 모두 불리합니다. 문서 한 건이 바뀔 때마다 모델을 다시 학습시킬 수는 없기 때문입니다.

RAG(Retrieval-Augmented Generation) 는 이 문제를 다르게 풉니다. 지식을 파라미터에 넣는 대신 **외부 저장소에 두고, 질의 시점에 검색해서 프롬프트에 주입**합니다. 모델은 "기억하는 존재"가 아니라 "주어진 근거를 읽고 답하는 존재"가 됩니다.

이 전환의 실무적 의미는 큽니다.

- 문서를 갱신하면 즉시 반영됩니다(재학습 불필요).
- 답변에 인용 출처를 붙일 수 있어 **감사 추적(auditability)** 이 확보됩니다.
- 접근권한을 검색 단계에서 필터링해 **문서 단위 보안**을 구현할 수 있습니다.

---

2. 표준 파이프라인 해부

RAG는 크게 **인덱싱(오프라인)** 과 **질의(온라인)** 두 단계로 나뉩니다.

2.1 인덱싱 단계

```
원본 문서 → 파싱/정규화 → 청킹 → 임베딩 → 벡터 인덱스 저장
```

① **파싱/정규화** PDF·HWPX·DOCX·HTML을 텍스트로 변환합니다. 실무에서 RAG 품질이 무너지는 첫 지점이 바로 여기입니다. 2단 조판 PDF의 컬럼 순서가 뒤섞이거나, 표가 줄바꿈으로 뭉개지면 그 뒤 파이프라인이 아무리 정교해도 복구되지 않습니다. 레이아웃 인식 파서(예: 문서 레이아웃 모델 기반 추출)나 표 전용 추출기를 별도로 적용해야 합니다.

② **청킹(Chunking)** 문서를 검색 단위로 자릅니다. 핵심 설계 변수는 다음과 같습니다.

전략	방식	적합 상황
고정 길이	512~1,024 토큰, 10~20% 중첩	균질한 산문
재귀적 분할	문단 → 문장 → 토큰 순 하향 분해	범용 기본값
의미 기반(Semantic)	인접 문장 임베딩 유사도 하락 지점에서 절단	주제 전환이 잦은 문서
구조 기반	마크다운 헤딩·조문·섹션 경계	법령, 기술 매뉴얼, 논문
부모-자식	작은 청크로 검색, 큰 부모 청크를 LLM에 전달	정밀 검색 + 충분한 문맥 동시 필요

경험칙: **검색 정밀도는 청크가 작을수록, 답변 충실도는 문맥이 클수록 좋아집니다.** 이 상충을 해소하는 표준 해법이 부모-자식 검색기(Parent Document Retriever) 또는 문장 윈도우(Sentence Window) 방식입니다.

③ **임베딩** 청크를 벡터로 변환합니다. 한국어 실무에서는 다국어 임베딩 모델의 한국어 성능 편차가 크므로, 반드시 **자체 도메인 질의셋으로 벤치마크**한 뒤 선정해야 합니다. 영어 리더보드 순위는 한국어 성능을 보장하지 않습니다.

④ **인덱스 저장** HNSW, IVF-PQ 등 ANN(근사 최근접 이웃) 인덱스에 저장합니다. 문서 메타데이터(작성일, 부서, 접근등급, 문서유형)를 함께 저장해야 이후 필터링 검색이 가능합니다. 이 설계를 초기에 빠뜨리면 재인덱싱 비용이 발생합니다.

2.2 질의 단계

```
질의 → 질의 변환 → 검색(Hybrid) → 재순위화 → 문맥 압축 → 프롬프트 조립 → 생성 → 인용 검증
```

① **질의 변환(Query Transformation)** 사용자 질의는 대개 검색에 최적화되어 있지 않습니다.

- **Query Rewriting**: 대화 맥락을 반영해 독립 질의로 재작성("그건 얼마야?" → "2025년 3분기 매출액은 얼마인가?")
- **Multi-Query / RAG-Fusion**: 질의를 3~5개 변형으로 확장해 병렬 검색 후 RRF(Reciprocal Rank Fusion)로 결과 융합
- **HyDE**: LLM이 가상의 이상적 답변을 먼저 생성하고, 그 답변을 임베딩해 검색. 질의-문서 간 어휘 격차가 클 때 효과적
- **Query Decomposition**: 복합 질의를 하위 질의로 분해해 각각 검색

② **하이브리드 검색** 밀집 검색(임베딩)과 희소 검색(BM25)은 실패 양상이 다릅니다. 밀집 검색은 의미는 잘 잡지만 고유명사·모델명·조항번호에 약하고, BM25는 정확한 토큰 매칭에 강하지만 동의어에 취약합니다. **두 결과를 RRF로 융합하는 하이브리드가 대부분의 도메인에서 단일 방식을 상회합니다.** 한국어는 형태소 분석기 설정이 BM25 품질을 좌우하므로 별도 튜닝이 필요합니다.

③ **재순위화(Reranking)** 검색기는 재현율(recall) 확보를 위해 top-k를 넉넉히(30~100) 가져오고, **크로스 인코더 리랭커**가 질의-문서 쌍을 직접 채점해 상위 3~10건으로 압축합니다. 바이 인코더(임베딩)는 질의와 문서를 독립적으로 인코딩하지만, 크로스 인코더는 둘을 함께 입력해 상호작용을 계산하므로 정확도가 크게 높습니다. **RAG 품질 개선 대비 투입 노력이 가장 효율적인 지점이 리랭커 도입입니다.**

④ **문맥 압축** 불필요한 문장을 제거해 토큰 비용과 "중간 소실(lost in the middle)" 현상을 줄입니다. 긴 컨텍스트에서 중간 위치 정보가 무시되는 경향이 보고되어 있으므로, 핵심 근거는 프롬프트의 앞·뒤에 배치하는 것이 유리합니다.

⑤ **생성과 인용 검증** 프롬프트에 "제공된 문맥만 사용하고, 근거가 없으면 모른다고 답하라"는 제약을 명시합니다. 생성 후 별도 검증 단계에서 각 문장이 실제 근거 청크에 지지되는지 확인하면 환각률을 실질적으로 낮출 수 있습니다.

---

3. 고급 아키텍처

기법	핵심 아이디어
Self-RAG	모델이 검색 필요 여부, 검색 결과 관련성, 답변의 근거 지지도를 스스로 토큰으로 평가
CRAG (Corrective RAG)	검색 품질을 평가해 낮으면 웹 검색으로 대체하거나 질의를 재작성
Adaptive RAG	질의 복잡도를 분류해 무검색 / 단일검색 / 다단계검색 경로를 동적 선택
Agentic RAG	검색을 도구(tool)로 노출하고 에이전트가 반복적으로 검색-추론 루프 수행
RAPTOR	청크를 재귀적으로 군집·요약해 계층 트리를 만들고 모든 층위에서 검색

RAPTOR는 뒤에서 다룰 GraphRAG와 문제의식이 맞닿아 있습니다. 두 방식 모두 "청크 단위 평면 검색으로는 전역적 질문에 답할 수 없다"는 한계에서 출발합니다.

---

4. 평가 체계

RAG는 검색과 생성 두 축을 분리 평가해야 원인 진단이 가능합니다.

검색 품질 - Hit Rate@k: 정답 근거가 상위 k에 포함된 비율 - MRR(평균 역순위), NDCG: 정답의 순위 품질 - Context Precision / Recall: 가져온 문맥 중 유효 비율, 필요한 문맥의 확보 비율

생성 품질 - Faithfulness(충실도): 답변 주장들이 제공 문맥에 의해 지지되는 비율 — **환각 측정의 핵심 지표** - Answer Relevancy: 답변이 질의에 부합하는 정도 - Answer Correctness: 정답셋 대비 정확도

운영 지표 - p95 지연시간, 질의당 토큰 비용, 무응답률(적절한 "모름" 응답 비율)

실무 원칙: **골든 데이터셋 50~200문항을 먼저 만들고 시작하십시오.** 평가셋 없이 튜닝하면 개선인지 퇴행인지 판별할 수 없습니다. 이것이 RAG 프로젝트의 성패를 가르는 가장 큰 변수입니다.

---

5. 실패 유형과 처방

증상	원인	처방
답이 엉뚱함	검색 실패(recall 부족)	하이브리드 검색, 질의 확장, 청크 재설계
근거는 맞는데 답이 틀림	생성 단계 문제	프롬프트 제약 강화, 리랭커로 노이즈 제거
표·수치를 못 읽음	파싱 단계 붕괴	레이아웃 인식 파서, 표 전용 추출
고유명사 검색 실패	밀집 검색 한계	BM25 병행, 키워드 필터
"여러 문서를 종합하면?"에 답 못함	다중 홉 추론 불가	질의 분해, 에이전트 루프, **GraphRAG**
"이 자료 전체 주제는?"에 답 못함	전역 요약 불가	계층 요약(RAPTOR), **GraphRAG 글로벌 검색**

---

6. RAG의 구조적 한계

표준 RAG는 본질적으로 **"질의와 가장 유사한 k개의 청크"** 를 찾는 시스템입니다. 여기서 두 가지 질문 유형이 원리적으로 처리되지 않습니다.

첫째, 다중 홉(multi-hop) 질의. "A사에 부품을 공급하는 업체 중 B규제 대상인 곳은?"이라는 질의는 A-공급사 관계와 공급사-규제 관계를 연결해야 답할 수 있습니다. 그러나 이 두 정보가 서로 다른 문서에 있고 어느 쪽도 질의와 직접 유사하지 않다면, 유사도 검색은 실패합니다.

둘째, 전역 요약(global sensemaking) 질의. "이 1만 건 문서의 핵심 쟁점 5가지는?"이라는 질의에 대응하는 특정 청크는 존재하지 않습니다. 답은 코퍼스 전체에 분산되어 있으며, top-k 검색으로는 원리적으로 도달할 수 없습니다.

이 두 한계가 **GraphRAG**가 등장한 이유입니다. 관련 내용은 별도 원고에서 다룹니다.

---

7. 실무 도입 체크리스트

- [ ] 골든 평가셋 50문항 이상 구축 (최우선)
- [ ] 파싱 품질 육안 검수 (표·수식·다단 조판 샘플 20건)
- [ ] 청킹 전략 2~3안 A/B 비교
- [ ] 한국어 임베딩 모델 자체 벤치마크
- [ ] 하이브리드 검색(밀집 + BM25 + RRF) 적용
- [ ] 크로스 인코더 리랭커 도입
- [ ] 메타데이터 필터링(권한·기간·부서) 설계
- [ ] "모름" 응답 경로 명시
- [ ] 인용 출처 UI 노출
- [ ] 지연시간·토큰비용 모니터링

8. 정리

RAG는 화려한 기법의 경연장이 아니라 **파이프라인 각 단계의 누수를 막는 공학**입니다. 실무 개선 효과가 큰 순서는 대체로 다음과 같습니다.

① 평가셋 구축 → ② 파싱 품질 → ③ 청킹 전략 → ④ 하이브리드 검색 → ⑤ 리랭커 → ⑥ 질의 변환 → ⑦ 고급 아키텍처

거꾸로 접근하는 조직이 많습니다. Self-RAG나 에이전트 루프를 먼저 도입하고도 성능이 오르지 않는 대부분의 사례는, 실은 PDF 파싱 단계에서 표가 깨져 있었던 경우입니다.

작성: 주식회사 별의문 (Stargate Corporation) · 2026-07-25