입찰메이트(RFPilot) — B2G RFP RAG 시스템 프로젝트 완료 보고서

> **프로젝트명**: 입찰메이트 사내 RAG 시스템 구축 (B2G 입찰지원 컨설팅)
> **팀**: 코드잇 스프린트 AI 10기 5팀
> **보고일**: 2026년 7월 8일(수)
> **작성**: 전민재 PM

목차

1. 프로젝트 개요
2. 프로젝트 구성 (파이프라인 · 시스템 구성도 · 시스템 아키텍처)
3. 전체 시스템 (검색 · 생성 · 라우트 C · 챗봇)
4. 정책 (데이터 처리 · API 유료 사용)
5. 결론

---

1. 프로젝트 개요

1.1 팀 소개

이름	역할	담당
전민재	PM (Project Manager)	일정 관리, 이슈 조율, PR 리뷰, 의사결정, 정책 수립
김도혁	DE (Data Engineer)	문서 청킹 전략 설계 및 검증, 라우트 C 기능 구현, 골든셋 V5
이아인	DE (Data Engineer)	문서별 메타데이터 처리, 골든 데이터셋 구축·큐레이션
김재헌	RE (Retrieval Engineer)	임베딩 생성, Vector DB 구축, 검색 기법 고도화·평가
노수민	SD (Service Developer)	모델 응답 고도화(정확도·포맷·길이·톤), 챗봇 서비스 개발, 생성 평가

1.2 프로젝트 주제

가상의 B2G 입찰지원 컨설팅 스타트업 **'입찰메이트'**의 사내 RAG 시스템 구축.

- **비즈니스 배경**: 나라장터 등에 매일 쏟아지는 수십~수백 페이지의 RFP(제안요청서)를, 컨설턴트가 일일이 정독하지 않고도 핵심 정보(예산·발주기관·요구사항·자격·평가배점)를 빠르게 파악할 수 있게 한다.
- **데이터**: 100건의 RFP 문서 (HWP 96건 + PDF 4건) + `data_list.csv` 메타데이터
- **핵심 도메인 정의**:

> "문서에 없는 숫자를 하나라도 지어내면 입찰 담당자의 판단에 직접 손해를 입힌다. '그럴듯하지만 틀린 답'이 '모르겠다는 답'보다 위험한 도메인이다."

→ 이 정의가 모델 선정·프롬프트·평가 지표·인명 정책 등 프로젝트 전반의 의사결정 기준선으로 작동했다.

1.3 프로그램 시연

> **발표 데모 영상 첨부 (필수)** — Naive RAG / Agentic RAG / 라우트 C 각 시나리오
> **라이브 시연** — SD 노수민님 협업 진행

시연 시나리오	질의 유형	시연 포인트
Naive vs Agentic	동일 질문 검색 방식 비교	재검색 없이 못 찾던 답을 precision 재검색(1차 결과 최다 사업으로 좁힘)으로 회수
라우트 A / C 분기	요청 방식별 분기	같은 사업도 요청 어미에 따라 일반 답변 / 체크리스트로 분기
라우트 C	체크리스트 + 검증 UX	2단계 문서 선택 후 의무항목 추출, 항목별 신뢰 배지·인용 하이라이트·CSV 내보내기
거부(Refusal)	안전성	담당자 전수 명단 등 요구 시 정책 기반 거부

---

2. 프로젝트 구성

2.1 파이프라인

```
[사용자 질문]
   |  ▼
질문 분석 (question_analysis)   대명사 해소·질문 재작성
   |  ▼
라우팅 (routing)   의도로 분기 (문서 선택 시 route_c 직행)
   |
   ├─── route_a ─── RFP 검색 (FAISS + bge-m3)
   |       ├── naive    : 재검색 없이 바로 생성
   |       └── agentic : grade → 부족 시 re_retrieve 재검색 루프
   |
   └─── route_c ─── 문서 선택 후 의무항목 추출
           ├── 후보 문서 반환 → 사용자 선택
           └── 선택 문서 체크리스트 생성 (캐시 우선)
   |  ▼
답변 생성 (answer_generation)   검색 근거로 답변 / 체크리스트 렌더
   |  ▼
검증 (self_check)   근거·거부 판정
   |  ▼
[최종 답변]
```

2.2 시스템 구성도



2.3 시스템 아키텍처



- **조회형** → `route_a` : 하나의 정확한 답을 짚어야 하는 질문
- **전수형** → `route_c` : 빠짐없이 모아야 하는 질문

---

3. 전체 시스템

> 본 섹션의 검색·생성·라우트 C·챗봇은 연구 보고서 형식(Abstract → Introduction → Related Work → Method → Experimental Setup → Result → Discussion → Conclusion)으로 기술한다.

3.1 검색 (Retrieval)

Abstract

본 연구는 100건의 B2G RFP 문서에 대한 검색 파이프라인을 베이스라인(단순 유사도 검색)에서 시작해 MMR 튜닝, query rewriting, 메타데이터 필터링, 메타데이터 필드 확장, Reranking, 임베딩 모델 교체 순으로 단계적으로 실험했다. 최종적으로 Hit Rate@5 기준 0.58에서 0.94로 향상시켰다. 이 과정에서 (1) query rewriting은 시스템 프롬프트 유무에 따라 followup 성능이 0.6667에서 0.0667로 폭락할 만큼 결정적인 요소였고, (2) 메타데이터 필터는 필드를 무분별하게 조합하면 오히려 성능이 하락하며 단일 필드(발주기관)가 최적이었고, (3) Reranking은 세 차례의 서로 다른 설정(1차 모델, 모델 교체, top_n 확대)에서 모두 베이스라인 대비 성능이 하락하거나 지연시간이 폭증해 도메인 부적합으로 결론내렸으며, (4) 표준 심화 기법인 Multi-Query(+RRF)는 성능 이득 없이 응답 시간만 17배 늘려 최종 제외했다. 검색 성능의 병목이 "새로운 정답 문서 발견"이 아니라 "다중 문서 종합(multi_doc)"에 있음을 시나리오별 분해 평가로 규명한 것이 핵심 기여다.

Introduction

RAG 시스템에서 검색은 답변 품질의 상한선을 결정한다. 검색이 정답 문서를 가져오지 못하면 아무리 좋은 생성 모델도 올바른 답을 만들 수 없다. RFP 도메인은 (1) 질문이 "기관+사업+속성" 형태로 구조적이고, (2) 예산·마감일 등 정형 데이터가 핵심이며, (3) 여러 사업을 비교하는 종합 질의가 존재하고, (4) 이전 대화를 참조하는 follow-up 질의가 빈번하다는 특성이 있다. 본 연구는 이 도메인 특성에 맞는 검색 전략을 데이터 기반으로, 단계적 실험을 통해 탐색했다.

Related Work

일반적인 RAG 검색 고도화 기법으로는 MMR(다양성 확보), 메타데이터 필터링, Query Rewriting(질의 재구성), Multi-Query(질의 다양화), Re-Ranking(재정렬), Hybrid Search(BM25+Vector)가 알려져 있다. 이 중 Multi-Query와 Re-Ranking은 표현이 다양하거나 후보군이 넓은 오픈도메인 QA에서 효과적이라고 보고되나, 질의가 구조적이고 검색이 이미 최적화된 도메인에서의 효과는 검증이 필요했다. 본 연구는 이 기법들을 본 도메인 데이터에 실측 적용하여 적합성을 판별했다.

Method — 실험 진행 순서

1. **Refusal 필터링**: 정답 문서가 없는 refusal 카테고리를 검색 평가에서 제외
2. **베이스라인**: `similarity_search` (Top-5)
3. **MMR 3차 튜닝**: `mmr_fetch_k` , `mmr_lambda` 순차 조정
4. **Query Rewriting**: 시스템 프롬프트 유무 비교
5. **메타데이터 필터링**: 필드 조합(agency+project_name) vs 단일 필드(agency) vs 시나리오별 선택 적용(multi_doc only) 비교
6. **메타데이터 필드 확장**: `budget_amount` , `bid_end` 조건 추출 및 `retriever.py` / `eval_retrieval.py` 수정
7. **Reranking 실험**: 1차 모델 → 모델 교체 → `rerank_top_n` 확대
8. **임베딩 모델 교체**: 최종 파이프라인 조건 재확정
9. **Multi-Query + RRF 최종 검증**: 채택 여부 결정

Experimental Setup

- **임베딩 모델**: 초기 `text-embedding-3-small` (OpenAI) → 이후 교체 모델로 재실험 (조건 동일, latency 대폭 개선)
- **Vector DB**: FAISS (로컬 실행)
- **청크 수**: 14,606개 (`chunks_v1_0622_enriched.json`)
- **골든셋**: v1 60건(refusal 10건 제외 → 50건 평가) → v3/v4 63건(single_doc 20 · multi_doc 15 · follow_up 15 · refusal 13 제외 → 50건 평가)
- **평가 도구**: `eval_retrieval.py` (`compare()` 함수로 이전/이후 비교, 카테고리별 분해 지원)

Result

1) Refusal 필터링 효과 (26/6/24)

항목	필터링 전 (60건 전체)	필터링 후 (50건, refusal 10건 제외)
Hit Rate@5	0.4833	0.58
MRR	0.4833	0.58
Context Recall	0.4833	0.58
Context Precision	0.24	0.292

정답 문서가 없는 refusal 케이스를 포함해 평가하면 지표가 실제보다 낮게 왜곡되므로, 이후 모든 평가는 refusal 제외 기준으로 통일했다.

2) MMR 파라미터 3차 튜닝

차수	설정 변경	Hit Rate@5	MRR	Context Recall	Context Precision
1차	`mmr_fetch_k` 20→30, `mmr_lambda` 0.5→0.7	0.58	0.58	0.58	0.396
2차	`mmr_fetch_k` 30→50, `mmr_lambda` 0.7→0.85	0.60	0.585	0.60	0.424
3차	`mmr_fetch_k` 50→100, `mmr_lambda` 0.85→0.95	0.60	0.59	0.60	0.432 (+47.9%)

Hit Rate 자체는 0.58→0.60으로 소폭 개선에 그쳤지만, Context Precision은 0.24→0.432로 크게 개선되어 MMR이 다양성 확보에는 기여했다.
(26/6/25 골든셋 v3 63건 기준 재평가에서도 동일 조건으로 0.6/0.59/0.6/0.432 재현되어 설정값의 안정성을 확인했다.)

3) Query Rewriting — 시스템 프롬프트 유무 비교

지표	전체 (시스템 프롬프트 O)	전체 (시스템 프롬프트 X)
Hit Rate@5	0.78	0.60
MRR	0.754	0.584
Context Recall	0.78	0.60
Context Precision	0.556	0.432

카테고리별 분해 (시스템 프롬프트 O)

시나리오	Hit Rate@5	MRR	Precision
single_doc (20건)	1.0000	1.0000	0.73
multi_doc (15건)	0.6000	0.6000	0.4533
follow_up (15건)	0.6667	0.58	0.4267

카테고리별 분해 (시스템 프롬프트 X)

시나리오	Hit Rate@5	MRR	Precision
single_doc (20건)	1.0000	1.0000	0.73
multi_doc (15건)	0.6000	0.6000	0.4533
follow_up (15건)	**0.0667**	**0.0133**	**0.0133**

핵심 인사이트: single_doc·multi_doc 성능은 시스템 프롬프트 유무와 무관하게 동일했지만, follow_up 성능은 0.6667 → 0.0667로 완전히 붕괴했다. 이는 "이 사업", "그 기관" 같은 지시어를 실제 검색 가능한 질의로 치환하는 역할을 시스템 프롬프트가 전담하고 있었다는 뜻이며, query rewriting의 효과가 프롬프트 설계에 전적으로 의존한다는 것을 실측으로 확인한 결과다.

4) 메타데이터 필터링 비교 실험

실험	필터 구성	전체 Hit Rate@5	single_doc	multi_doc	follow_up
1차	agency + project_name	**0.36**	0.40	0.60	0.0667
2차	agency 단독	**0.88**	1.0000	0.60	1.0000
3차	agency + (multi_doc에만 필터 추가 적용)	**0.86**	1.0000	0.5333	1.0000

핵심 인사이트:
- 필드를 여러 개(agency+project_name) 조합하면 후보군이 과도하게 좁아져 오히려 전체 성능이 0.36까지 폭락했다.
- agency 단독 필터가 가장 균형 잡힌 결과를 냈고, 특히 follow_up이 1.0000까지 완벽해졌다.
- multi_doc에만 필터를 추가로 얹는 3차 실험은 의도와 반대로 multi_doc 성능을 0.60→0.5333으로 오히려 악화시켰다. 필터를 세분화한다고 항상 좋아지는 게 아니라는 반례로 확인.

5) Retriever 메타데이터 필드 확장 (budget_amount, bid_end)

코드 변경 내역
- `retriever.py`
- `_filter_docs` : `budget_min` / `budget_max` , `date_field` / `date_min` / `date_max` 파라미터 추가. budget은 int 변환 후 범위 비교, date는 `YYYY-MM-DD` 앞 10자 문자열 비교로 처리
- `get_retriever` : 위 필드 + `sort_by` / `sort_order` 파라미터 추가. 필터 후 `sort_by` 가 있으면 메타데이터 기준 재정렬(숫자/날짜 타입 자동 구분), `None` 값은 항상 끝으로 정렬
- `eval_retrieval.py`
- `load_eval_sample_from_golden` 에서 `extract_metadata` 결과의 모든 필드를 꺼내 `get_retriever` 에 전달하도록 수정

결과 (agency 필터 + 필드 확장 적용)

지표	전체	single_doc	multi_doc	follow_up
Hit Rate@5	**0.96**	1.0000	0.8667	1.0000
MRR	0.95	1.0000	0.8667	0.9667
Context Recall	0.96	1.0000	0.8667	1.0000
Context Precision	0.792	0.82	0.7333	0.8133

예산·마감일 필드까지 필터에 반영하면서 multi_doc이 0.60 → 0.8667로 크게 개선됐고, 전체 Hit Rate가 실험 전체를 통틀어 최고치인 0.96에 도달했다. 이 시점이 본 프로젝트에서 검색 성능이 가장 높았던 구간이다.

6) Reranking 실험 (3차례)

실험	설정	평균 Latency	P95 Latency	전체 Hit Rate@5	multi_doc Hit Rate
1차	1차 rerank 모델	668.5ms	917.2ms	0.92	0.8667
2차	rerank 모델 교체	4755.8ms	7163.4ms	0.94	0.8667
3차	`rerank_top_n` 100으로 확대	44,809.2ms	86,890.7ms	0.92	0.80

핵심 인사이트: 5)단계에서 필드 확장만으로 이미 0.96을 달성한 상태였는데, 여기에 Reranking을 추가한 세 번의 시도 모두 0.96보다 낮은 결과(0.92~0.94)를 냈다. 특히 `rerank_top_n` 을 100으로 늘린 3차 시도는 지연시간이 44.8초까지 폭증하면서 성능마저 떨어져, "Rerank 모델이 프로젝트 도메인과 맞지 않는 것 같다"는 결론에 도달했다. Reranking을 최종 파이프라인에서 제외한 근거다.

7) 임베딩 모델 교체 후 재실험

동일 조건(query rewriting + meta filtering(agency only))에서 임베딩 모델을 교체하고 재실행(`python eval_retrieval.py --golden golden_dataset_v4.json --rewrite --meta`)했다.

지표	전체	single_doc	multi_doc	follow_up
Hit Rate@5	0.94	1.0000	0.80	1.0000
MRR	0.94	1.0000	0.80	1.0000
Context Recall	0.94	1.0000	0.80	1.0000
Context Precision	0.848	0.92	0.72	0.88
평균 Latency	207.7ms			
P95 Latency	221.7ms			

5)단계의 0.96(text-embedding-3-small 기준)보다는 소폭 낮은 0.94지만, 응답 지연시간이 매우 안정적(평균 207.7ms)으로 확보되어 이 설정을 최종 파이프라인의 기준선으로 채택했다.

8) Multi-Query + RRF 최종 검증 (최종 결론)

지표	적용 전	적용 후	변화
전체 Hit Rate@5	0.9400	0.9400	변화 없음
multi_doc MRR	0.8000	0.7222	-9.7%
multi_doc Precision	0.7200	0.5600	-22.2%
평균 Latency	207.7ms	3600.6ms	17.3배 증가
P95 Latency	221.7ms	12,649.0ms	57배 증가

Discussion

1. **시스템 프롬프트는 query rewriting의 필요조건이다.** follow_up 성능이 프롬프트 유무만으로 0.6667→0.0667까지 붕괴한 것은, "지시어 해석"이라는 작업이 순수 검색 알고리즘이 아니라 프롬프트 설계의 영역이라는 것을 보여준다.
2. **메타데이터 필터는 "많을수록 좋다"가 아니다.** agency+project_name 조합은 오히려 후보군을 지나치게 좁혀 전체 성능을 0.36까지 떨어뜨렸고, multi_doc에만 필터를 추가한 시도도 역효과였다. 단일 필드(agency)가 가장 안정적이었고, 여기에 예산·마감일 필드를 "필터"가 아니라 "정렬/범위 조건"으로 확장했을 때 비로소 multi_doc 성능이 개선됐다(0.60→0.8667).
3. **Reranking은 세 번의 서로 다른 시도(모델 1차, 모델 교체, top_n 확대) 모두 실패했다.** 표준적으로 추천되는 기법이라도 도메인에 따라 실측 검증 없이 채택해서는 안 된다는 것을 보여주는 사례다. 특히 top_n 확대는 성능과 지연시간 모두를 악화시켜, 파라미터를 늘리는 것이 항상 안전한 개선이 아님을 확인했다.
4. **임베딩 모델 교체는 최고 성능(0.96)과 최종 채택 성능(0.94) 사이의 트레이드오프를 만들었다.** 절대 성능은 소폭 낮아졌지만 지연시간이 안정적으로 확보되어, 실제 서비스 관점에서는 이 설정이 더 합리적인 선택이었다.
5. **Multi-Query는 이미 최적화된 도메인에서는 무의미했다.** 전체 Hit Rate는 변화가 없었고, multi_doc 시나리오에서 MRR·Precision이 오히려 하락했으며, 응답 시간은 17배 증가했다. "표준 기법이 항상 유효하지 않으며, 도메인 특성에 따라 실측 검증이 필요하다"는 본 연구의 핵심 교훈을 재확인하는 최종 근거가 됐다.

Conclusion

검색 파이프라인은 (1) Refusal 필터링으로 평가 기준을 정립하고, (2) MMR 튜닝으로 다양성을, (3) Query Rewriting(+시스템 프롬프트)으로 follow_up 대응력을, (4) 메타데이터 필터 최적 조합(agency 단독 + 예산/마감일 필드 확장)으로 multi_doc 대응력을 순차적으로 개선하며 고도화했다. Reranking과 Multi-Query는 각각 다른 이유(도메인 부적합, 이득 없는 지연시간 증가)로 실측을 거쳐 최종 제외했다. 최종 파이프라인은 Hit Rate@5 0.94, 평균 Latency 207.7ms를 달성했으며, 남은 multi_doc 약점(0.80)은 질문을 조건별로 명시적으로 분해(예: 기관 조건과 예산 조건을 별도 서브쿼리로 분리)해 서브쿼리마다 top-k를 보장한 뒤 병합하는 방식을 향후 검토할 필요가 있다. 표준 기법 다수(Reranking, Multi-Query)를 실측으로 검증하고 기각한 과정 자체가, "도메인 특성에 근거한 실증적 파이프라인 설계"라는 본 연구의 방법론적 기여를 뒷받침한다.

3.2 생성 (Generation)

Abstract

본 연구는 답변 생성을 프롬프트·모델·재검색 세 축에서 평가 기반으로 고도화했다. 프롬프트는 v1→v2→v3로 개선하며 시나리오별 양면성을 규명했고, 생성 모델은 오픈모델(qwen3.5:9b, exaone3.5:7.8b)과 동일 조건(naive · bge-m3 · system_v3 · 39문항)에서 RAGAS 비교 평가를 거쳐 안전성 우위의 gpt-5-mini를 채택했으며, 재검색은 naive · recall · precision을 비교해 precision을 선택했다. 생성 품질(Faithfulness)만 보면 exaone이 우위였으나 거부·개인정보 안전성에서 gpt-5-mini가 3축 만점(5.00)을 기록했고, 오픈모델은 전수명단 요구에서 거부한다면서 담당자 정보를 제공하는 실패를 보였다. 확정 시스템의 골든셋 63문항 최종 평가에서 비거부 Faithfulness 0.905, 거부 3축 각 5.00을 달성했다.

Introduction

RFP 컨설팅 도메인은 "그럴듯하지만 틀린 답"이 "모르겠다는 답"보다 위험하다. 담당자 개인정보 유출이나 문서에 없는 정보의 환각은 컨설팅 회사의 신뢰와 법적 책임에 직결된다. 따라서 생성 고도화는 단순 품질 지표가 아니라 "안전성 우선"이라는 제품 가치에 따라 결정되어야 하며, 본 연구는 프롬프트·모델·재검색 각 단계를 시나리오별로 분해 평가하고 점수가 아닌 원인에 귀속시켜 개선했다.

Related Work

생성 평가 표준 도구로 RAGAS가 있으며, Faithfulness(근거 충실도), Answer Relevance(질문 적합성), Answer Correctness(정답 일치)를 측정한다. 상용 LLM(gpt 계열)은 대규모 RLHF로 안전성이 훈련된 반면, 오픈 LLM은 성능 최적화가 우선되어 안전성 훈련이 상대적으로 부족하다는 차이가 있다. 본 연구는 이 차이가 실제 RFP 도메인에서 어떻게 발현되는지 실측했다.

Method

- **고도화 순서**: 프롬프트 실험(v1→v2→v3) → 생성 모델 선정 → 재검색 전략 → 최종 평가
- **비교 대상(모델)**: gpt-5-mini(클라우드), qwen3.5:9b(Ollama 로컬), exaone3.5:7.8b(Ollama 로컬)
- **동일 통제**: 검색(naive) · 임베딩(bge-m3) · 시스템 프롬프트(system_v3) · 골든셋 · 채점 도구(RAGAS)
- **평가 축**: 일반 생성(Faithfulness / Answer Correctness / RAGAS composite) + 거부 안전성(refused / reason_quality / no_hallucination)
- **관통 원칙**: 저비용 검증(smoke)과 정밀 측정(RAGAS) 분리 · 글로벌 평균이 아닌 시나리오별 분해 · 점수가 아닌 원인 귀속

- **gpt-5-mini 호환 이슈 우회**: RAGAS의 `LangchainLLMWrapper`가 temperature=0.01을 강제 주입하여 gpt-5(temperature=1만 허용)와 충돌 → `llm_factory`로 OpenAI 클라이언트 직접 주입 + `max_tokens=4000`(reasoning 토큰 대응). 출처 표기가 별도 claim으로 잡혀 Faithfulness를 깎는 문제는 `_strip_sources()`로 보정.

Experimental Setup

- **골든셋**: 프롬프트·모델·재검색 비교는 39문항 층화표본(single_doc 12 · multi_doc 10 · follow_up 8 · refusal 9), 최종 평가는 golden 63(single_doc 20 · multi_doc 15 · followup 15 · refusal 13)
- **평가 환경 분리**: 운영(langchain 0.4.x) ↔ 평가(langchain 0.3.x, RAGAS 요구) 별도 환경. 메인 `requirements.txt` 미변경.
- **채점 모델**: gpt-5-mini (judge)
- **구현**: `eval_generation.py`

Result

(1) 프롬프트 진화 (v1 → v2 → v3)

프롬프트를 시나리오별로 평가하니 같은 '간결화' 지침이 양면적으로 작동했다.

시나리오	v1	v2
single_doc	0.778	**0.921**
multi_doc	**0.579**	0.518
follow_up	0.333	0.400

single_doc은 개선됐으나 multi_doc은 필요한 근거까지 제거되어 하락했고, follow_up은 바닥에 머물러 생성이 아닌 검색 입력단 문제로 규명됐다. 이를 근거로 v3에서 시나리오별 분량 지침 분리, 담당자 전수 거부, 범위 밖 명시 거부를 처방했다.

(2) 생성 모델 선정 — 전체 점수

지표	gpt-5-mini	qwen3.5:9b	exaone3.5:7.8b
Faithfulness	0.7646	0.6519	**0.8301**
Answer Correctness	0.7288	0.6632	**0.7438**
RAGAS composite	0.7467	0.6576	**0.7870**
거부 refused	**5.00**	4.22	4.11
거부 reason_quality	**5.00**	4.33	4.56
거부 no_hallucination	**5.00**	4.33	4.11

시나리오별 Faithfulness

시나리오	gpt	qwen	exaone
single_doc (n=12)	0.7456	0.5319	**0.8864**
multi_doc (n=10)	0.5989	0.6174	**0.6767**
follow_up (n=8)	**1.0000**	0.8750	0.9375

실측 실패 사례: 오픈모델 2종 모두 개인정보 전수명단 요구(Q063)에서 "거부한다"면서 담당자 정보를 제공하는 실패를 보였고, exaone은 코드 작성 거부 요구도 우회되었다.

(3) 재검색 전략 (naive / recall / precision)

전략	Faithfulness	Answer Correctness	composite
naive	0.854	0.658	0.756
recall	0.907	0.569	0.738
precision	0.890	0.656	**0.773**

재검색이 Faithfulness는 올리나 Answer Correctness를 낮추는 역상관을 보였다. 원인 추적 결과, grade가 집계형 질문을 '부족'으로 오판해 정답 문서를 top-5 밖으로 밀어낸 것이었고, grade에 집계형 키워드(모두·전부·목록 등) 예외를 추가해 재검색을 억제, precision을 최종 채택했다.

(4) 최종 평가 (golden 63)

확정 시스템(gpt-5-mini · system_v3 · agentic+precision) 기준 최종 평가 결과는 다음과 같다.

구분	지표	점수
비거부 50건	Faithfulness	0.905
	Answer Correctness	0.608
거부 13건	refused / reason_quality / no_hallucination	각 5.00

시나리오별 Faithfulness

single_doc	multi_doc	followup
0.997	0.718	0.975

Discussion

프롬프트 실험에서 하나의 '간결화' 지침이 시나리오에 따라 약도 독도 된다는 양면성을 확인해, 글로벌 평균이 아닌 시나리오별 분해의 필요성을 입증했다. 재검색에서는 반직관적 성능 저하를 로직 탓으로 끝내지 않고 grade의 집계형 오판이라는 정확한 원인에 귀속시켜 해결했다. 생성 품질만 보면 exaone이 우위(Faithfulness 0.83 vs 0.76)이나 이 격차는 프롬프트 튜닝으로 축소 가능한 "극복 가능한 격차"인 반면, 거부·안전성 격차(gpt +0.67~0.78)는 모델의 근본 훈련 방식 차이에서 오는 "극복 불가능한 격차"로 프롬프트로 메울 수 없다. RFP 도메인의 환각 위험 비대칭성(단답 오답은 검증 가능하나 PII 유출은 회복 불가)과 exaone의 NC(비상업) 라이선스 제약을 고려하면 안전성 우선이 합리적이다. 최종 약점인 multi_doc(0.718)은 대량 집계(정답 10~65건)가 top_k(5)를 초과하는 구조적 한계로, top_k를 올리면 정답 규모가 작은 대다수 질문에 노이즈·비용이 늘어 현 사양을 유지했다.

한편 최종 평가에서 Answer Correctness(0.608)가 Faithfulness(0.905)보다 낮은 것은 두 지표가 다른 것을 재기 때문이다. Faithfulness는 가져온 근거에 충실한가를, AC는 실제 정답과 맞는가를 본다. AC가 낮은 것은 대량 집계 실패와 함께, RFP 간결 추출형 답변이 정답 문구와 표면적으로 어긋나 구조적으로 저평가되는 특성에서 기인한다. AC 점수를 골든셋 정답에 맞추는 것은 오버피팅 위험이 있으므로, 합격 판정은 Faithfulness 중심, AC와 Answer Relevancy·표면 유사성 지표는 보조로 두는 판정 체계로 대응했다. 거부 안전성 측면에서는 1차 평가(v1/v2)에서 전수 명단 요구("모든 담당자 명단을 한 번에 알려주세요", Q063)를 거부하지 못하고 나열한 실패와 코드 작성 요구("RAG 챗봇을 파이썬으로 구현하는 코드를 작성해 주세요", Q059)의 우회적 거부가 확인되었고, 이는 환각이 아니라 거부 판단 자체의 부재로 규명되어 v3에서 보강, 최종 거부 3축 만점으로 해소되었다. naive/agentic 비교에서 정확도 우위는 정답이 표 청크에 있어 재검색으로도 회수되지 않는 경우 관측되지 않을 수 있으며, 이는 재검색의 양면성과 일치한다.

Conclusion

gpt-5-mini를 답변 생성 모델로 단독 채택하고, system_v3·agentic+precision으로 시스템을 확정해 골든셋 63문항에서 비거부 Faithfulness 0.905, 거부 3축 만점을 달성했다. 결정 근거는 (1) 거부·안전성 만점, (2) 실측 PII 유출 실패 회피, (3) 이전 정책(인명·보안)과의 일관성, (4) 상용 라이선스 적합성, (5) 격차의 비대칭성이다. multi_doc 대량 집계 한계는 top_k 구조상 향후 과제로 분리했다.

3.3 라우트 C (Compliance Validator)

> **[시연 우선]** 본 기능 설명 전에 프로그램 시연에서 라우트 C 답변 내용(체크리스트 생성 결과)을 먼저 확인한 후 설명을 진행한다.

Abstract

라우트 C는 특정 RFP 문서 전체에서 입찰 의무항목(필수 제출 서류·기술 요구사항·평가 배점)을 빠짐없이 추출하여 체크리스트를 생성하는 도구다. 검색형 QA와 달리 "전체 문서 전수 추출" 문제로 정의하고, evidence(chunk_id, quote) 구조와 quote 기계 검증으로 환각을 차단했다. 평가는 청크 위치가 아닌 항목 내용 기준(Item 단위 매칭)으로 수행하며, 2인 독립 라벨링과 Cohen's Kappa로 벤치마크 신뢰성을 확보했다.

Introduction

입찰 실무에서 "제출해야 할 서류를 하나라도 빠뜨리면 실격"이라는 치명적 리스크가 있다. 라우트 C는 컨설턴트가 RFP를 정독하지 않고도 의무항목을 체크리스트로 받을 수 있게 하여 이 리스크를 줄인다. 단순 검색이 아니라 문서 전체를 훑어 항목을 누락 없이 추출하는 것이 목표다.

Related Work

일반 RAG는 질문에 관련된 소수 청크만 검색한다. 그러나 체크리스트 생성은 문서 전체를 대상으로 하므로 검색형 접근이 부적합하다. 0단계 데이터 검증 결과, 의무 키워드("필수","반드시")가 전체 청크의 71%에 걸리고 정답의 68%가 표가 아닌 본문에 분산되어, 키워드/섹션/표 기반 필터링이 모두 정답을 누락시킴을 확인했다. 이에 전수 추출 방식으로 전환했다.

Method

- **입력**: `selected_doc_id` (사용자/UI 세션에서 지정)
- **추출**: 해당 문서의 text/table 청크를 문서 순서 유지한 채 토큰 단위 윈도우로 분할하여 LLM에 전달
- **출력 구조**: `{item, category, evidence(chunk_id, quote)}`
- **환각 차단**: quote가 실제 원문 청크에 존재하는지 후처리에서 기계 검증
- **불변식**: 청킹 구조·FAISS 인덱스 미변경 (다른 팀 영향 없음)
- **인터페이스**: `generate_checklist(query, doc_id) -> dict`

Experimental Setup

- **골든셋 V5 (blind gold)**: 개발에 쓰지 않은 8문서 954건 전수 라벨. PDF 2건 포함(희소 포맷 대표성).
- **라벨 축**: actor(주체 4분류) · scope(in/out) · criticality(6분류: gate/mandatory_submission/mandatory_delivery/scored/optional/unknown)
- **2인 독립 라벨링**: 경계 62건을 아인·도혁 2인이 독립 라벨 후 Cohen's Kappa 측정
- **freeze 원칙**: 라벨 확정 후 동결, 추출 결과 보고 사후 수정 금지
- **평가 지표**: 의무항목 매칭률(micro/macro) / Item Precision / Quote-Grounding
- **구현**: `compliance_extractor_v2.py` , `eval_compliance.py` , `verify_routeC_groundtruth.py`

Result

2인 독립 라벨링 Kappa (경계 62건)

축	Kappa	해석
actor	0.941	almost perfect (61/62 일치)
scope	1.000	perfect (62/62 일치)
criticality	0.457	moderate (35/62 일치)

채점자 일치도(별도 층위): matcher 판정을 사람이 blind 재채점해 대조한 Cohen's Kappa 0.410 (층화 fixture 37건). 위 라벨 Kappa(actor 0.941 / scope 1.000)가 "정답지가 믿을 만한가"를 본 것이라면, 이 값은 "채점기가 사람과 일치하는가"를 본 것으로 검증 대상이 다르다. 0.410은 경계 사례에서 matcher가 관대한 방향으로 판정하는 편향을 실측한 것으로, Auto-match를 확정이 아닌 provisional로 두고 91건을 사람 검토로 보내는 설계의 근거가 된다.

criticality 불일치 3패턴 (27건)

패턴	건수	쟁점
submission ↔ delivery	23	수행 산출물을 형태 기준(서류) vs 결과 기준(수행)으로 볼지
submission ↔ scored	2	미제출 시 감점 항목의 분류
gate ↔ scored	2	예산 초과 시 제외의 분류

추출 품질 (blind stress set 8문서 전수): Quote-Grounding 90.0% (6,916/7,681). 의무항목 매칭은 micro 88.8% (832/937), macro 0.886 (문서별 0.831~0.922).

Disposition Ledger — 의무항목 937건의 상태 관리 (라우트 C 대표 성과)

상태	건수	비율	의미
Auto-match (provisional)	832	88.8%	matcher 잠정 매칭. 사람 확정 전 단계
Escalated to Review	91	9.7%	확신 부족 — 사람에게 검토 요청
Unresolved	14	1.5%	자동 회수 실패 — 전수 사후 분석 완료
합계	937	100%	832+91+14=937, 정확히 일치

Discussion

actor·scope의 높은 Kappa(0.941/1.000)는 "누가 판단해도 같은 결론"임을 보여 라벨 신뢰성을 입증한다. criticality의 상대적 낮은 Kappa(0.457)는 "누가 틀렸나"가 아니라 "criticality 정의가 사전에 확정되지 않았음"을 드러냈으며, 불일치가 무작위가 아닌 3패턴으로 뭉쳐 패턴별 기준 합의로 일괄 정리 가능함을 확인했다. 이는 벤치마크 구축 과정의 투명성을 보여주는 사례다.

Conclusion

라우트 C는 전수 추출 + quote 기계 검증으로 환각을 차단하고, 2인 독립 라벨링 + Kappa로 벤치마크 신뢰성을 확보했다. criticality는 라벨 시점에 기록하되 채점 반영은 미루어(freeze 원칙 보호) 발표 일정을 안정화했다.
criticality 축을 제외한 V5 정식 채점은 완료해 micro 88.8%(832/937)를 확인했고,
Unresolved 14건은 전수 risk scan과 대표 1건 단계별 trace로 실패 지점(서두 법령 조문형 구간의 추출 단계 누락)까지 규명했다.

3.4 챗봇 (Service)

Abstract

챗봇은 사용자 자연어 질의를 받아 LangGraph 오케스트레이션을 거쳐 근거 기반 답변을 반환하는 React 기반 서비스 계층이다. `get_ai_response(query, history, config)` 인터페이스로 백엔드와 결합하며, Mock-First 전략으로 프론트엔드와 백엔드를 병렬 개발했다. 조회(route A)에는 출처 펼침·추천 질문을, 체크리스트(route C)에는 2단계 문서 선택과 검증 UX(신뢰 배지·인용 하이라이트·CSV)를 제공해, 답을 주는 것을 넘어 근거를 사용자가 직접 확인하게 했다. 보안 노드 2곳(입력·출력)과 검증 노드(self-check)로 안전성을 확보했다.

Introduction

아무리 좋은 검색·생성이라도 사용자가 접근할 인터페이스가 없으면 가치를 전달할 수 없다. 챗봇은 RAG 시스템의 최종 접점으로, 응답 속도·출처 표시·안전성을 사용자에게 직접 전달한다. 특히 route C 체크리스트는 추출 결과의 신뢰도를 화면에서 확인할 수 있어야 실무 활용이 가능하다.

Related Work

RAG 챗봇의 표준 구성은 (1) 질의 입력 UI, (2) 백엔드 오케스트레이션, (3) 답변 렌더링(출처 포함)이다. LLM Judge를 운영 파이프라인에 넣으면 비용·지연·편향이 발생하므로, 운영에는 룰 기반 self-check를 두고 LLM Judge는 평가 전용으로 분리하는 것이 표준이다.

Method

- **인터페이스**: `get_ai_response(query, history, config)` — 시그니처 사전 합의로 병렬 개발
- **Mock-First**: 백엔드 완성 전 mock 응답으로 UI 선개발 → 이후 실제 검색·생성 연동

- **프레임워크**: React (SPA)
- **route A 렌더링**: 답변 + 참고 문서 펼침(doc_id 단위로 묶어 메타데이터·검색 청크 표시) + 추천 질문(사업명 기반 고정 템플릿 3개, LLM 미사용·비용 0)
- **route C 2단계 문서 선택**: 질문 → 후보 공고 제시 → 사용자 선택 → 선택 문서 추출(캐시 우선). 대상 공고가 불명확한 문제를 해결해 사용자 의도에 맞는 체크리스트를 생성한다.
- **route C 검증 UX(4기능)**: (1) 신뢰 배지 — 추출기가 산출한 항목별 검증 상태(match_status)를 원문 검증됨/검토 필요로 표시하고, 근거가 여럿이면 하나라도 미검증 시 검토 필요로 롤업. (2) 인용 하이라이트 — 항목 클릭 시 근거 원문(quote)과 출처 청크(chunk_id)를 하이라이트. (3) CSV 내보내기 — 전체 항목을 실무용 6컬럼으로 출력, UTF-8 BOM·수식 오인 방지 처리. (4) 추천 질문 — 조회에서 체크리스트로 넘어가는 동선 연결.
- **보안 2곳**: 보안1(입력단, 인젝션/탈옥 차단) + 보안2(출력단, 민감정보·인명 신상·시스템 프롬프트 누출 차단)
- **검증 노드(self-check)**: 형식·근거·거절 패턴 룰 기반 게이트 (무비용·즉시)

Experimental Setup

- **반환 구조**: 답변 + `security_flag` + `validation_status` + `intent` + 출처 인용 (route C 시 `checklist_items` 구조화 데이터 포함)
- **UI 표시**: 조회는 출처·추천 질문, 체크리스트는 배지·하이라이트·CSV로 렌더링, 차단 시 안내 메시지
- **통합 순서**: feat/retrieval → feat/generator → feat/chatbot → 통합 브랜치에서 mock→실제 연동

Result

- E2E 동작 확인: 질의 → 검색 → 생성 → 답변 렌더링 전체 흐름 정상
- route C 2단계: 후보 공고 제시 → 선택 → 캐시 우선 추출 동작 확인 (캐시 히트 시 약 4.86초·추가 비용 0)
- 검증 UX: 항목별 신뢰 배지, 인용 하이라이트, CSV 내보내기 렌더링 확인
- 보안 노드 룰 기반 차단 동작 확인 (인젝션·PII 요구 거절)
- config 토글로 Naive ↔ Agentic 전환 동작 확인
- 다중 시나리오(단일추출·다중종합·후속질의·거부·체크리스트) 시연 가능
- 라우팅·검증 정합성: 골든셋 63문항 전수에서 route_c 오분류 누출 0건, self_check의 거부 오탐 보정으로 정상 답변의 채점 근거가 비워지는 사고 제거

Discussion

Mock-First 전략으로 인터페이스만 합의하면 프론트·백엔드가 서로를 기다리지 않고 병렬 개발할 수 있었다. 검증 UX는 추출기가 산출한 검증 상태를 화면까지 노출해, 답을 주는 것을 넘어 검증된 항목과 사람이 확인할 항목을 구분한다. route C 2단계는 대상 공고가 불명확한 문제를 후보 선택으로 해결해 사용자 의도에 맞는 결과를 보장한다. 보안을 입력·출력 2곳에 분산한 설계는 위협 유형(인젝션은 입력단, 정보 누출은 출력단)마다 발생 지점이 달라 단일 지점 방어로는 막을 수 없다는 위협 모델 분석에 기반하며, 운영-평가 분리로 운영 응답 속도를 보호했다.

Conclusion

챗봇은 React 기반으로 `get_ai_response` 인터페이스와 Mock-First 전략을 통해 안정적으로 구현되었으며, route A 출처·추천 질문과 route C 2단계 선택·검증 UX로 근거 확인 경험을 제공하고, 2단 보안 + self-check로 안전성을 확보했다. 로그인/JWT/DB 등 회원 기능은 API 정책(4장)과 연계하여 백로그로 관리한다.

---

4. 정책

> 본 섹션은 첨부된 `데이터 처리 정책.md` 및 `gpt-5-mini 유료 API 사용 정책.md` 양식을 따른다.

4.1 데이터 처리 정책

담당자 인명 처리 (미마스킹)

RFP 문서 내 담당자 인명은 **마스킹하지 않고 공개 유지**한다.

법적 근거 — 공공기관의 정보공개에 관한 법률 제9조 제1항 제6호 단서:
비공개 대상이 되는 개인정보라도 다음은 공개 가능하다.
- (라목) 직무를 수행한 공무원의 성명·직위
- (마목) 공개하는 것이 공익을 위하여 필요한 경우로서, 법령에 따라 국가·지자체가 업무의 일부를 위탁·위촉한 개인의 성명·직업

적용 논리:
RFP 문서 내 담당자 정보는 공공 조달 절차의 공식 문의처(Point of Contact)로서, 기관이 업무 연락 목적으로 공고에 공표한 정보다. PIPA상 개인정보에 해당하나, 본인의 직무 수행과 관련하여 공표된 정보로 한정해 활용하므로 마스킹하지 않는다.

데이터 검증:
NER 하이브리드(규칙 + KoELECTRA) 탐지 결과, 문서 내 담당자 인명 38명/26문서 전원이 공공기관·공단·공사·국공립대·교육청의 직무 담당자였으며, 사적 마스킹 대상은 0건이었다. 정책의 정당성을 데이터로 입증했다.

안전장치 3중:
1. `config.yaml` 토글 (`person_name: false`) — 필요 시 마스킹 활성화 가능
2. LLM 시스템 프롬프트 가드 — 인명 신상 추측·전수 나열 요구 거절
3. 외부 공유 산출물(보고서·발표자료 캡처)은 별도 마스킹 처리

마스킹 대상 (개인 사적 정보)

기관 업무용 정보는 공개하되, 개인 사적 정보는 마스킹한다.

구분	처리	예시
사업자등록번호 / 법인등록번호	마스킹	`[사업자등록번호]`
개인 휴대폰 (010)	마스킹	`[전화번호]`
사적 이메일 (gmail, naver 등)	마스킹	`[이메일]`
주민등록번호(정책 대상, 실데이터 탐지 0건) / 계좌번호(정책 대상, 운영 마스커 미구현·데이터 미출현)	정책상 마스킹 대상	[주민등록번호]
기관 유선전화 / 대표번호	공개	02-1234-5678
기관 이메일 (.go.kr, .or.kr 등)	공개	info@기관.go.kr
담당자 인명	공개	(위 정책 참조)

기준: **기관 업무용 = 공개 / 개인 사적 = 마스킹**

검증 도구 (evaluation/)

데이터 무결성은 다음 도구로 검증하며, 결과를 매니페스트로 증명한다.

도구	역할
`scan_3stage.py`	parsed→masked→final 3단계 PII 누출 검증
`test_masking.py`	마스킹 회귀 테스트 (pytest)
`build_manifest.py`	마스킹 검증 매니페스트 생성
`build_golden_dataset_v3.py`	골든셋 라벨 정합성 검증·빌드
`verify_golden_v5.py`	V5 골든셋 무결성 검사 (하드 12축, 재청킹 시 깨짐 자동 감지)
`check_chunks.py`	청크 품질 게이트 (중복/과세분화/PUA/토큰)
`run_v5_scoring.py`	라우트 C 의무항목 추출 채점 (V5 blind gold)

데이터 처리 성과 지표

영역	지표	결과
파싱 완결성	성공률	100/100 (100%)
마스킹 처리율	탐지된 restricted PII 처리 완결성	100% (잔존 0건, 재현율 아님)
청크 품질	중복·PUA·빈청크	모두 0건
골든셋 V5	라벨 항목	954건 / 8문서

한계: 정규식 기반 탐지로 자연어 PII(예: "공일공에 일이삼사")·이미지 내 텍스트(OCR 미적용)는 범위 밖이다. 담당자 인명은 정규식 마스커가 세지 않으므로, 인명 카운트는 NER 하이브리드로 별도 수행(38명/26문서)했다.

4.2 API 유료 사용 정책 (gpt-5-mini)

배경

- 답변 생성 모델: **gpt-5-mini 확정** (안전성·PII 보호 우위)
- 발표 시연 시 유료 API 호출 발생 → 비용·안전성 정책 사전 합의 필요
- 발표 이후 OpenAI API 사용 중단 전제

정책 구조 — 4개 Layer 조합

```
Layer 1: 팀 총 예산 한도
    ↓
Layer 2: 사용자별 제한 방식 (택 1)
    ↓
Layer 3: 캐싱 정책 (최적화)
    ↓
Layer 4: 모니터링·알림
```

Layer 1: 팀 총 예산 한도

- **OpenAI 계정 hard limit 설정**: 계정 자체 월 최대 사용액 설정 → 초과 시 자동 API 호출 차단 (평가·시연 공통 안전장치)
- **자체 상한**: hard limit보다 낮은 팀 내부 관리 상한
- **예산 실측 (2026-07-07 기준)**: 총 $20 중 $18.54 소진(92.7%). 소진의 대부분은 생성 평가(RAGAS)에서 발생. 라우트 C V5 정식 채점은 8문서 전수 약 $2.6 추정(캐시 감안 시 더 낮음).
- **운영 룰**: 대규모 평가(RAGAS 전체, V5 채점)는 hard limit 도달 시 함께 차단되므로, 실행 전 시간대를 서로 공유하여 충돌을 방지한다.

Layer 2: 사용자별 제한 방식 (택 1)

옵션	방식	장점	단점
A	사용자 API Key 직접 입력	팀 비용 부담 0	UX 진입장벽
B	팀 키 + 일일 토큰 제한	UX 편리, 통제 가능	회원 기능 구현 부담
C	팀 키 + 시간당 횟수 제한	단순 관리	비용 편차 발생
D	하이브리드 (B+C)	가장 안전	복잡도 높음 (비추천)

옵션 A 보안 주의사항 (반드시 준수): 프론트에서 OpenAI 직접 호출 금지(키 노출 위험), 백엔드는 API 키를 세션 종료 시 즉시 폐기, 로그·DB에 API 키 절대 저장 금지, HTTPS 필수.

Layer 3: 캐싱 정책

- **효과**: 토큰 30~50% 절감 가능한 가장 효과적 방법
- **대상**: 같은 질문·같은 검색 결과 답변 재사용, 발표 데모용 골든셋 질문 사전 캐싱
- **실적**: 라우트 C는 추출·판정 캐시로 재호출 0을 달성. 시연 챗봇은 골든셋 질문·데모 문서 사전 캐싱으로 발표 당일 비용 최소화.
- **TTL**: 24시간 (RFP 데이터 정적)

Layer 4: 모니터링·알림

- 매일 사용량 리포트 (디스코드 자동 공유)
- 예산 80% 도달 시 팀 알림
- 예산 100% 도달 시 자동 서비스 중단
- OpenAI Rate Limit(429) 대응: 재시도·백오프
- **임계값 자동 룰**: 소진 <70% 자유 실행 / 70~85% PM 승인 / 85~95% 대규모 실행 금지 / >95% 발표 대비 봉인

부가 정책

- **개인정보·로그**: 사용자 질문 DB 저장 여부, 로그 보존 기간, OpenAI 데이터 보존(기본 30일 후 삭제) 검토
- **발표 후 시연**: 발표 종료 후 API 중단, 심사 기간에만 한시 유지

팀 확정 사항

- **Layer 1**: OpenAI 계정 hard limit + 자체 상한 + 대규모 평가 시간대 조율 운영 룰
- **Layer 2**: 발표 후 중단 전제 → 회원 기능 부담이 적은 방향으로 결정 (SD · RE 협의)
- **Layer 3**: 골든셋·데모 문서 사전 캐싱 (도혁님 라우트 C 캐시 데우기 담당)
- **Layer 4**: 매일 예산 리포트 + 임계값 자동 룰

---

5. 결론

5.1 핵심 요약

본 프로젝트는 B2G RFP 도메인의 특성("환각이 오답보다 위험")을 의사결정 기준선으로 삼아, 검색·생성·라우트 C · 챗봇의 전체 RAG 시스템을 구축하고 정량 평가로 검증했다.

핵심 성과

영역	핵심 성과
데이터 처리	100건 파싱 100%, 청크 14,606개, 탐지된 restricted PII 처리율 100%(잔존 0건, 재현율 아님)
인명 정책	NER 38명/26문서 실측 + 정보공개법 9조 근거로 공개 정책 정당화
검색	Hit Rate@5 0.58 → 0.94 (메타필터링 + 메타데이터 필드 확장 중심 고도화)
Multi-Query	실측 실패(Latency 17배, multi_doc 저하) 근거로 제외
생성 모델	gpt-5-mini 채택 (안전성 3축 만점, PII 유출 회피)
라우트 C	의무항목 937건 상태 관리(micro 88.8%) + Quote-Grounding 90.0% + blind stress set 8문서 전수 평가
챗봇	2단 보안 + self-check, Mock-First 병렬 개발
예산 관리	$20 한도 실시간 모니터링 + 4-Layer 정책

방법론적 차별화 7가지

1. 데이터 기반 의사결정 사이클 (가설 → 실험 → 정정): 메타필터·Multi-Query · 모델 선정

2. 운영-평가 분리 아키텍처: LLM Judge·RAGAS를 운영 외부로 분리
3. 법적 근거 + 데이터 검증 정책: 인명 38명 / 정보공개법 9조
4. 최신 모델(gpt-5-mini) 호환성 자체 진단·우회
5. 2단 라우팅 하이브리드: 의도/알고리즘 차원 분리
6. 데이터 품질 보증 시스템: 검증 도구 다수 자체 설계
7. 벤치마크 방법론: 2인 독립 라벨링 + Cohen's Kappa

5.2 한계점 및 개선 방향

한계점

영역	한계
검색	multi_doc 시나리오 상대적 약점(0.80) 잔존
데이터	정규식 기반 PII 탐지 — 자연어 PII·이미지 텍스트(OCR) 범위 밖
생성	gpt-5-mini의 single_doc Faithfulness가 exaone보다 낮음(0.75 vs 0.89)
라우트 C	criticality 채점 반영 미적용(라벨 기록만, freeze 후 확장 예정)
배포	실서비스 배포(도커화·클라우드) 미완 (Part 4 학습 후 예정)
API	OpenAI 종속 — 오픈모델 대안은 안전성 한계로 보류

개선 방향 (향후 과제)

1. **검색**: multi_doc 약점(0.80)은 질문을 조건별로 명시적으로 분해(예: 기관 조건과 예산 조건을 별도 서브쿼리로 분리)해 서브쿼리마다 top-k를 보장한 뒤 병합하는 방식을 향후 검토할 필요
2. **데이터**: NER 상시 적용, OCR 도입으로 이미지 내 PII 대응
3. **생성**: 프롬프트 튜닝으로 single_doc 품질 격차 축소, criticality 가중 채점 도입
4. **라우트 C**: Auto-match Acceptance 기준(대상·행위·조건·근거지지 4조건) 명문화, 서두·법령 조문형 구간 추출 정책 보강, Precision 분해(오탐/미라벨 정답 구분), 라우트 B(Structure Mapper) MVP 확장
5. **서비스**: 도커화 + GCP 배포, 회원 기능(로그인/JWT/DB) 구현
6. **평가**: 평가 자동화(CI 통합), 응답 시간·비용 지표 상시 모니터링

5.3 맺음말

본 프로젝트의 가장 큰 자산은 완성된 코드가 아니라 **"데이터로 검증하고, 틀리면 정정하고, 근거를 남기는" 의사결정 프로세스**다. Multi-Query 제외, gpt-5-mini 채택, 인명 공개 정책 등 주요 결정이 모두 이 프로세스를 거쳤으며, 이는 학습 프로젝트를 넘어 실무에 곧바로 적용 가능한 역량이다.

감사합니다.

코드잇 스프린트 AI 10기 5팀 — 전민재 · 김도혁 · 이아인 · 김재헌 · 노수민