# RAG-RFP 시스템 구축 프로젝트 보고서

Team2: 박병현, 배현석, 손원후, 이솔형

## 1. Overview

### 1.1 Project Summary

본 프로젝트는 공공기관·지자체에서 발행하는 RFP(Request for Proposal) 문서를 대상으로 핵심 정보 추출 및 질의응답을 자동화하는 Retrieval-Augmented Generation(RAG) 시스템을 구축하는 것을 목표로 한다. PDF/HWP 기반의 비정형 문서를 자동으로 분석하여 제출 마감일, 사업 규모, 예산, 수행기관 요건, 과업 범위 등의 정보를 빠르고 정확하게 조회할 수 있는 시스템을 설계하였다.

두 개의 시나리오를 통해서 비교 실험을 진행했으며, 시나리오 A는 GCP 기반으로 진행됐으며, 시나리오 B는 Cloud API 기반(OpenAI)으로 진행됐다.

### 1.2 Background

공공사업 제안서(RFP)는 문서 길이가 길고 구조가 기관마다 상이하여 사람이 직접 분석하기에 많은 시간과 비용이 소요된다. 따라서 대규모 문서로부터 빠른 정보 검색을 지원하는 AI 기반 문서 이해 시스템의 필요성이 증가하고 있다.

### 1.3 Objectives

- 비정형 PDF/HWP 문서에서 텍스트를 자동 추출한다.
- 문서 내 핵심 정보를 기반으로 청킹(Chunking) 및 임베딩(Embedding) 을 진행한다.
- FAISS 기반 벡터 데이터베이스를 구축한다.
- 사용자 질문에 대해 정확한 문서 기반 응답을 제공한다.
- 후보 청크를 재선정하기 위해 Reranker 모델을 추가한다.
- Retrieval 성능을 정량적으로 평가하여 최적화된 파이프라인을 구축한다.

## 2. Project Architecture

### 2.1 System Pipeline

- 데이터 수집 및 파일 구조 정리
- OCR + 텍스트 추출(pdfplumber, fitz, easyocr)
- 전처리 및 문서 구조 분석
- Chunking – multi-aspect chunking
- Embedding (시나리오 A: BAAI/bge-m3 /시나리오 B: OpenAI text-embedding-3-small)
- Vector DB 구축 (시나리오 A: FAISS HNSW index / 시나리오 B: FAISS IVFlat index)
- Retriever 개발 (Top-k Search)
- Hybrid Search (BM25 + Dense Retrieval)
- Reranker 추가 (Cross-encoder 기반, 시나리오 B에만 적용)
- 전체 Recall 평가 및 최적 조합 선정

# 3. Data Preparation

## 3.1 Raw Data

- 총 문서 수: 100개
- 파일 형식: PDF, HWP
- 크기: 100MB+

## 3.2 Preprocessing

전처리 과정은 다음과 같다:

- 텍스트 품질 검사(깨짐 감지, unique ratio 검사)
- 페이지 단위 텍스트 추출(pdfplumber → fallback to fitz → fallback to OCR)
- 헤더/푸터 제거
- 줄바꿈 정규화
- bullet point 복구
- 페이지 메타데이터 저장

# 4. Document Structure Analysis

문서 종류가 다양하므로, RFP의 공통 구조를 분석하여 Chunking 전략 설계의 기준으로 삼았다.

# 5. Chunking Strategy

## 5.1 Baseline: Fixed-size Chunking

chunk_size = 512/1024/2048 tokens
overlap = 64 tokens

## 5.2 Multi-Aspect Chunking

문단을 한 번씩만 자르는 대신, 서로 다른 목적의 청크를 여러 개 생성하여 다른 관점의 검색 신호를 강화하는 전략.

예:
구조 기반 청크
질문 기반 청크
목적 기반 청크
heading 기반 청크

## 5.3 Header-aware Chunking

LLM이 문서의 섹션 제목을 감지하여 자연스러운 단락 단위로 쪼개도록 한 방식.

### 5.4 Hybrid Chunking

Rule-based Chunking(패턴 기반)
LLM Header Chunking
Multi-aspect Chunking
세 가지 방식의 장점을 결합하여 Recall을 극대화
이 중에, Multi-Aspect Chunking이 성능이 가장 좋아 최종 청킹 전략으로 결정하였다.
(Baseline 전략의 성능을 훨씬 능가하였다.)

참고)
Baseline Chunking Recall@K 성능
- R@1: 0.3
- R:@3: 0.3667
- R@5: 0.4

Multi-Aspect Chunking Recall@K 성능
- R@1: 0.667
- R@3: 0.736
- R@5: 0.792

## 6. Embedding

### 6.1 Embedding Model

- 시나리오 A: BAAI/bge-m3
- 시나리오 B: OpenAI text-embedding-3-small / 1536-dim dense vector

### 6.2 Query Embedding

query 역시 동일 모델로 정규화(normalization) 하여 직접 비교가 가능한 cosine similarity 기반 검색 진행.

### 6.3 Embedding Output

- embeddings.npy
- metadata.pkl

## 7. Vector DB: FAISS 구축

### 7.1 Index types tested

FlatIP, FlatL2, IVFFlat, IVFPQ, HNSW을 각각 성능 테스트 진행.

### 7.2 Best performing index: HNSW

시나리오 A: FAISS IVFFlat
시나리오 B: FAISS HNSW

# 8. Retriever

## 8.1 시나리오 A – Pure Retriever Pipeline

- Dense Retrieval (FAISS): 임베딩 기반의 고정밀 검색 수행. 사용자 쿼리를 BAAI/bge-m3로 임베딩, 벡터 정규화 후 FAISS index에서 cosine similarity 기반 Top-k 검색

- Hybrid Search (BM25 + Dense Retrieval): 문서 내 특정 키워드 기반 검색(BM25)과 의미 기반 검색(Dense)를 결합해 Recall@K 향상을 극대화한다.
BM25로 후보 검색 -> Dense Retrieval 결과와 merge -> 점수 normalize 후 weighted sum -> 상위 후보 K개 선택

시나리오 A에서는 LangChain 없이 직접 코드로 구현하여 비용 최소화.

## 8.2 시나리오 B – LangChain 기반 모듈형 Retriever

시나리오 B는 시나리오 A의 검색 성능을 토대로, LLM + LangChain 기능을 활용한 실험 중심 구조로 확장된다.
주요 목적은 여러 Retriever 전략 비교, 다양한 파이프라인 자동화, LLM 기반 query expansion 실험 등을 수행하는 것.

- Dense Retrieval (LangChain Wrapper): LangChain의 Retriever 인터페이스를 활용하여 다양한 실험을 빠르게 구성할 수 있게 하는 것.

  OpenAIEmbeddings + FAISS wrapper 적용
  LangChain의 SimilaritySearch 및 max_marginal_relevance_search 실험 가능

  성능 개선 목적보다는 실험 자동화 및 조합 테스트가 핵심.

- Hybrid Search: 시나리오 A와 방식 동일

- LLM 기반 Reranker: 시나리오 B에서는 LLM reranking 실험까지 확장적 접근 가능

# 9. Generator

Generator 개선은 A → B → C 단계로 진행되었다.

## 9.1 Prompt 구조 개편 (A단계)

- system_prompt 완전 구조화
- user_content 세분화
- RFP 제안서 톤 반영
- evidence 기반 reasoning 강화
- hallucination 억제 규칙 추가
- JSON 실패 대비 fallback 로직 적용

**출력 안정성과 품질이 크게 향상됨**

### 9.2 JSON Output 표준화 (B단계)

- python에서 안전한 `json.loads`
- 기존 `evaluator` / `ask.py` 완전 호환
- key 누락 시 default 자동 세팅

### 9.3 확장 JSON 적용 (C단계)

| 필드명 | 설명 |
|---|---|
| confidence | LLM 답변 확신도(0-1) |
| risk_points | 리스크 항목 요약 |
| missing_info | 추가 필요한 정보 |
| structured_sections | 문서 구조 기반 섹션 요약 |
| tags | 자동태깅 |

CLI 반영: `ask.py`에서 전체 구조가 보기 좋게 출력되도록 리뉴얼됨.

## 10. Conclusion

본 프로젝트는 공공기관 RFP 문서를 대상으로 한 RAG 기반 자동 정보 검색 시스템을 구축하였다.
다양한 청킹 방식, 효율적 벡터 DB 설계, 재랭커 기반 모델 최적화를 통해 recall 성능을 크게 향상시켰으며, 문서 기반 QA 시스템을 위한 토대를 마련하였다. 향후 LLM 기반 구조 분석 강화 및 pipeline 자동화를 통해 실제 업무에서도 사용 가능한 수준으로 확장할 수 있다.