# SSAFY 학사 규칙 QA 챗봇 프로젝트 최종 보고서

# 1. 프로젝트 개요

## 1.1. 프로젝트 목적

- SSAFY 학생들의 학사 규정 접근성 향상
- 운영사무국의 반복적인 문의 응대 업무 감소
- 신속하고 정확한 규정 정보 제공
- 규정 해석의 일관성 확보

## 1.2. 개발 환경

- Backend: FastAPI, Python
- Frontend: Vue.js
- Deployment: AWS EC2
- Models: OpenAI text-embedding-ada-002, Cross-encoder/ms-marco-MiniLM-L-6-v2
- URL: https://ssafychat.kro.kr/

# 2. 시스템 설계

## 2.1. 데이터 구조

- 규정 데이터 (rule.json)
    - 계층적 구조의 학사 규정 정보
    - 재귀적 청크화를 통한 효율적인 검색 단위 구성
- QA 데이터셋 (qa_dataset.json)
    - 카테고리별 질문-답변 쌍 구조
    - 실제 사용 사례 기반의 예시 데이터

## 2.2. RAG 시스템 구현

1. **청크화 프로세스**

    - 규정 데이터: 계층 구조 보존 재귀적 청크화
    - QA 데이터: 개별 QA 쌍 단위 청크화
2. **2 단계 검색 프로세스**
    - Stage 1 (Bi-encoder):
        - OpenAI text-embedding-ada-002 활용
        - 코사인 유사도 기반 초기 검색
        - 상위 10 개 청크 선택
    - Stage 2 (Cross-encoder):
        - ms-marco-MiniLM-L-6-v2 모델 활용
        - 문맥적 관련성 평가
        - 최종 3 개 청크 선택
3. **성능 최적화**
    - 임베딩 캐시 구현
    - API 호출 최소화
    - 배치 처리 적용

# 3. 주요 기능 및 특징

## 3.1. 핵심 기능

- 학사 규정 관련 자연어 질의응답
- 규정과 실제 적용 사례 통합 제공
- 문맥 기반의 정확한 정보 검색
- 대화 이력 관리 (최근 20 개)

## 3.2. 기술적 특징

- 2 단계 검색을 통한 정확도 향상
- 임베딩 캐시를 통한 응답 속도 최적화
- 계층적 데이터 구조 활용
- AWS EC2 를 통한 안정적인 서비스 제공

# 4. 배포 환경

## 4.1. AWS EC2 구성

- Instance Type: t2.micro
- OS: Ubuntu 20.04 LTS
- Domain: ssafychat.kro.kr
- HTTPS 적용

## 4.2. 배포 아키텍처

- Nginx 웹 서버
- PM2 프로세스 관리
- SSL 인증서 적용
- CORS 설정

# 5. 성과 및 한계

## 5.1. 주요 성과

- 신속하고 정확한 규정 정보 제공
- 운영사무국 업무 효율화
- 규정 해석의 일관성 확보
- 데이터 기반 규정 개선 가능성 제시

## 5.2. 한계 및 개선점

- 데이터 확장 필요성
    - 실제 질의응답 데이터 부족
    - 더 다양한 규정 사례 필요
- 성능 모니터링 체계 부재
    - 응답 정확도 측정 필요
    - 사용자 피드백 수집 체계 필요
- 캐시 전략 개선 필요
    - LRU 캐시 도입 검토
    - 캐시 크기 최적화

# 6. 향후 발전 방향

## 6.1. 기술적 개선

- 한국어 특화 임베딩 모델 도입
- 실시간 성능 모니터링 시스템 구축
- 캐시 전략 고도화

## 6.2. 서비스 확장

- 사용자 피드백 시스템 구축
- 규정 변경 이력 관리 기능
- 다국어 지원 확장

# 7. 결론

본 프로젝트는 최신 RAG 기술을 활용하여 SSAFY 학사 규정 검색 문제를 효과적으로 해결했습니다. 2 단계 검색 프로세스와 효율적인 캐싱 전략을 통해 정확하고 신속한 응답을 제공할 수 있게 되었으며, AWS EC2 를 통한 안정적인 서비스 운영이 가능해졌습니다. 향후 데이터 확장과 성능 개선을 통해 더욱 발전된 서비스를 제공할 수 있을 것으로 기대됩니다.