SK네트웍스 **Family AI** 과정 **15**기

모델링 및 평가 수집된 데이터 및 전처리 문서

산출물 단계	모델링 및 평가
평가 산출물	수집된 데이터 및 전처리 문서
제출 일자	2025.10.17
깃허브 경로	https://github.com/SKNETWORKS-FAMILY-AICAMP/SKN15-FINAL-3TEAM
작성 팀원	노건우

1. 데이터 수집 개요

데이터 수집 기간: 2025.10.06 ~ 2025.10.16

수집 방식: 크롤링, API

1.1. 수집 목적

수집 목적:

- 논문/학술 데이터
  논문 데이터와 메타데이터를 수집하여 유사 논문 자동 추천 시스템을 개발하고, 신기술 동향 분석 및 연구자-특허 맵핑 모델 개발에 활용하기 위함.

- 법령/심사기준 데이터
  특허 심사기준, 법령 정보를 수집해 실시간 법령 매칭 서비스, 자동 법률 인용 및 특허 거절 사유 추론 모델 개발에 활용하기 위함.

- 특허 및 거절 이력 · 의견결정서
  특허 출원, 심사, 거절 사례 및 의견결정서 데이터를 축적하여 특허 거절 사유 예측 모델, 특허 취득 성공률 향상 시스템을 개발하기 위함.

1.2 데이터 출처

데이터명	출처	수집 방식	형식
논문 정보	arxiv	웹 크롤링	JSON
특허 정보	KIPRIS	API 연동 수집	CSV
특허 거절 이력	KIPRIS	API 연동 수집	CSV
법령	AI hub	웹 크롤링	CSV
심사 기준	지식 재산처	웹 크롤링	CSV
대화 로그	내부 시스템	DB 추출	CSV
의견 결정서	KIPRIS	API 연동 수집	CSV

1.3 수집 대상 및 범위

대상	총 수집 건수	개인정보 포함 여부	민감 정보 여부 및 조치 사항
논문 정보	약 210 건	X	X
특허 정보	약 61,000 건	X	X
특허 거절 이력	약 2,000 건	X	X
심사 기준	약 21 건	X	X
법령	약 9건	X	X
의견 결정서	약 8500건	X	X

2.저장 방식

- 저장 경로: /data/raw_data/consult_log.json
- 저장 형식: CSV (UTF-8)

2.1 중복 및 정합성 검증

- 중복 제거 기준: 출원 번호 기준 중복 제거
- 정합성 확보 방법:
    - 주요 식별자(특허번호)의 공식 데이터 소스 기준값 활용
    - 크로스체크(특허/거절이력 교차 검증)
    - 데이터 수집~정제 단계에서 자동 형식 및 중복·누락 체크

3. 데이터 전처리 절차

3.1 이상치 탐지 및 처리

- 이상치 기준:

구분	주요 이상치 유형
논문 데이터	발행일 오류(미래 시점), DOI 형식 불일치, 동일 논문 중복 수집
특허 데이터	존재하지 않는 출원번호, 등록/거절 상태 불일치, 출원인명 비정형 패턴
거절 이력	동일 출원번호 내 중복 기록, 심사일자 역전(등록일보다 빠름), 입력 누락
로그 데이터(챗봇,	비정상 질의/응답 길이, 유효하지 않은 사용자 ID, action_type 오탈자

검색, 관리자)	

- 처리 방법:

처리유형	기준	처리방법
단순 오류	입력단 타이포, 포맷 깨짐	정규식 변환 및 포맷 교정(OCR, replace rule)
중복 데이터	동일 키(ID·제목·출원번호 등)	유사도·발행일 기준 최신본만 유지
결측치 발생	DOI/심사일 등 누락	외부 API(학술·특허 DB) 재조회 또는 '결측' 플래그 설정
비정상 로그	action_type, 길이 이상	사전 정의 키워드 외 항목 삭제 및 예외 테이블 저장
모호값(다중 기록)	동일 키에 상이한 값 존재	검증 신뢰도가 높은 출처 우선 선택(공식 DB → 내부 DB → 사용자 입력 순)

3.2 결측치 처리

- 결측 필드: 출원 번호, 청구항 일부 누락
- 처리 방법: 누락 응답 → 해당 row 제거
- 적용 비율: 전체 데이터 중 약 10% 제거

3.3 데이터 변환 및 재현 가능성

- 텍스트 정제:
    - 특수문자 제거, 이모지 필터링, 유니코드/공백 정규화

- 변환 코드 예시:

```
BOILER_PATTERNS = [
    r"^\s*\d{2}-\d{4}-\d{7}\s*$",
    r"^\s*-+\s*\d+\s*-+\s*$",
    r"^\s*Page\s+\d+\s+of\s+\d+\s*$",
    r"^\s*발\s*송\s*번\s*호.*$",
    r"^\s*발\s*송\s*일\s*자.*$",
    r"^\s*제\s*출\s*기\s*일.*$",
    r"^\s*YOUR INVENTION PARTNER\s*$",
    r"^\s*\[.*서식.*\]\s*$",
]
BOILER_RX = re.compile("|".join(BOILER_PATTERNS))


def normalize_text(s: str) -> str:
    if s is None:
        return ""
    s = unicodedata.normalize("NFKC", str(s))
    s = s.replace("\u00A0", " ")
    s = re.sub(r"[ \t]{2,}", " ", s)
    s = "\n".join(ln.rstrip() for ln in s.splitlines())
    return s

TERM = re.compile(r"[\.。!? ！ ？…)]$")
START_OK = re.compile(r"^[가-힣A-Za-z0-9\(]")

def join_broken_lines(s: str) -> str:
    lines = s.splitlines()
    if not lines:
        return s
    out = [lines[0]]
    for ln in lines[1:]:
        prev = out[-1]
        if prev and not TERM.search(prev.strip()) and START_OK.search(ln.strip()):
            out[-1] = (prev.rstrip() + " " + ln.lstrip()).strip()
        else:
            out.append(ln)
    return "\n".join(out)

def remove_boilerplate(text: str) -> str:
    kept = []
    for ln in (text or "").splitlines():
        if not BOILER_RX.search(ln):
            kept.append(ln)
```

```
    return "\n".join(kept)

def clean_page_text(raw: str) -> str:
    t = normalize_text(raw)
    t = remove_boilerplate(t)
    t = join_broken_lines(t)
    t = re.sub(r"\n{3,}", "\n\n", t).strip()
    return t
```

- GitHub 또는 Colab 링크: https://github.com/SKNETWORKS-FAMILY-AICAMP/SKN15-FINAL-3TEAM/tree/main/preprocessing

3.4 전처리 절차 요약 (프로세스 플로우)

1. 데이터 수집
2. 중복 및 결측 제거
3. 텍스트 정제
4. 이상치 제거
5. 토큰화
6. 저장 (train.csv / test.csv)

5. 데이터셋 분리

- 학습/검증/테스트 비율:

구분	건수	비율
Train	42,700	70%
Validation	9,150	15%
Test	9,150	15%

- 분리 기준: 무작위(Random State=42), 클래스 비율 균형 유지 (Stratified Split)