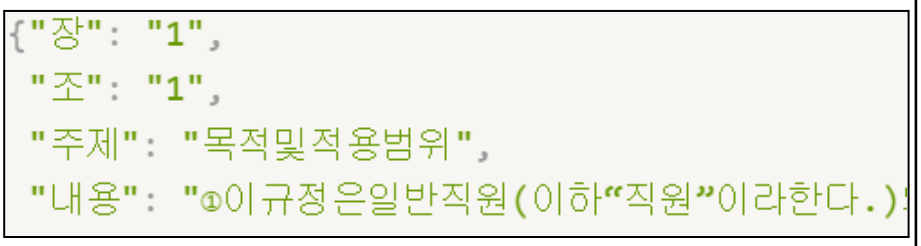
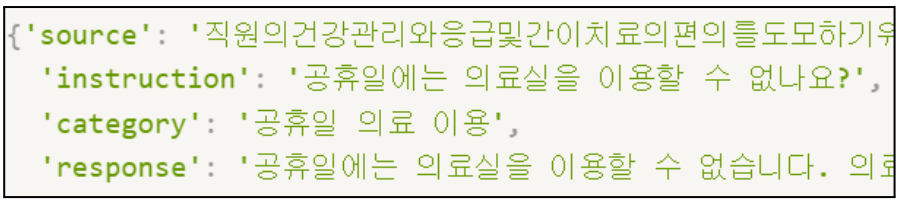
SK네트웍스 **Family AI**과정 **3**기

데이터 전처리 인공지능 데이터 전처리 결과서

☐ 개요

- 산출물 단계 : 데이터 전처리
- 평가 산출물 : 인공지능 데이터 전처리 결과서
- 제출 일자 : 12월 17일
- 깃허브 경로 : https://github.com/SKNETWORKS-FAMILY-AICAMP/SKN03-FINAL-3Team
- 작성 팀원 : 송영빈, 장수연

데이터 전처리 개요	**1.** 목적: • 원본 PDF 규정집을 구조화하여 텍스트 분석과 질의응답 데이터 생성에 용이한 데이터 추가 • 후속 FAQ 데이터 생성과 챗봇 모델 학습을 위한 기초 자료 마련 **2.** 대상 문서: • 사내 규정, 복지 정책, 휴가 정책 등을 포함한 PDF 문서 **3.** 결과물 형태: • JSONL(JSON Lines) 형태의 구조화된 데이터 • **PDF** 문서: {"장":"...", "조":"...", "주제":"...", "내용":"..."}  • **FAQ** 데이터: {"source":"...", "category":"...", "instruction":"...", "response":"..."} 

전처리 과정	**-** 전처리 순서 **1.** **PDF** 문서 전처리 **2.** 문서 기반 **FAQ** 데이터 생성 **3.** **FAQ** 데이터 전처리 **1. PDF** 문서 전처리 **(PDF → JSONL** 변환**)** **1.** **PDF** 텍스트 추출**:** PDF 파일에서 텍스트를 파싱하여 기본 텍스트 형태로 변환. **2.** 구조 파악 및 분할**:** ○ 문서 내 장, 조, 주제 항목 파악 ○ 해당 구조 단위별로 텍스트를 분리 **3.** 텍스트 정제**:** ○ 불필요한 공백, 특수문자 제거 ○ 문법 및 형식적 노이즈 제거 **4.** **JSONL** 변환**:** ○ 정제된 텍스트를 장, 조, 주제, 내용 필드로 구분 ○ 각 단위별 텍스트를 한 줄 한 레코드 형태의 JSON으로 작성 후 JSONL 파일 생성

5. 전처리 코드

```python
# PDF 경로 및 결과 파일 경로
pdf_path = "/content/drive/MyDrive/FINAL/복무규정.pdf"
output_path = "복무규정.jsonl"

# PDF 전체 텍스트 추출
doc = fitz.open(pdf_path)
full_text = ""
for page_num in range(doc.page_count):
    page = doc.load_page(page_num)
    full_text += page.get_text()

# 연속된 공백을 하나로 축소
full_text = re.sub(r'\s+', ' ', full_text).strip()

# "제N장" 패턴
chapter_pattern = r"(?m)제\s*(\d+)\s*장"
# "제N조 (주제)" 패턴
article_pattern = r"(?m)제\s*(\d+)\s*조\s*\((.*?)\)"

# JSONL 파일 쓰기 시작
with open(output_path, 'w', encoding='utf-8') as f:
    # 장 단위로 텍스트를 분리
    chapter_matches = list(re.finditer(chapter_pattern, full_text))
    for i, ch_match in enumerate(chapter_matches):
        ch_num = ch_match.group(1)  # 장 번호
        ch_start = ch_match.end()   # 장의 시작 위치

        # 다음 장의 시작 위치를 찾기
        if i < len(chapter_matches) - 1:
            ch_end = chapter_matches[i + 1].start()
        else:
            ch_end = len(full_text)

        # 해당 장의 텍스트 내용
        ch_content = full_text[ch_start:ch_end].strip()

        # 조 단위로 분리
        article_matches = list(re.finditer(article_pattern, ch_content))
        for j, art_match in enumerate(article_matches):
            art_num = art_match.group(1)    # 조 번호
            art_topic = art_match.group(2)  # 조의 주제
            art_start = art_match.end()     # 조 시작 위치

            # 다음 조의 시작 위치를 찾기
            if j < len(article_matches) - 1:
                art_end = article_matches[j + 1].start()
            else:
                art_end = len(ch_content)

            # 해당 조의 내용
            art_content = ch_content[art_start:art_end].strip()
            art_content = re.sub(r'\s+', ' ', art_content).strip()

            # JSON 형태로 데이터 기록
            entry = {
                "장": ch_num,
                "조": art_num,
                "주제": art_topic,
                "내용": art_content
            }
            f.write(json.dumps(entry, ensure_ascii=False) + "\n")
```

2. FAQ 데이터 생성

1. 질문 생성:
    - LLM 모델에 주제별 내용을 문맥으로 제공
    - 사내 상황을 가정한 질문 자동 생성
    - 다양한 어조, 표현 방식의 질문 확보
    - 질문 생성 시에는 temperature를 높게 설정하여(0.7~0.9) 창의적이고 다양한 표현의 질문을 유도

2. 질문 카테고리화:
    - 생성된 질문에 대해 LLM을 통해 간결한 형태의 카테고리(주제) 도출
    - 이 과정을 통해 질문을 의미별로 그룹핑하고, FAQ 탐색성을 향상
3. 답변 생성:
    - 동일한 문맥을 기반으로 LLM에게 친근하고 명확한 답변 작성 요청
    - 질문 텍스트 임베딩 생성
    - FAISS 인덱스 생성 및 임베딩 추가
    - FAISS를 사용해 유사한 규정 검색 후 답변 생성
    - 답변 생성 시에는 temperature를 낮게 설정하여(0.0~0.3) 정확하고 일관성 있는 응답을 확보
    - 규정 내용 범위를 벗어나면 “해당 질문은 대답할 수 없습니다.” 명시
    - 구조화된 EXCEL 형식의 FAQ 데이터셋 완성
4. 생성 코드

```
# 규정 문서를 OpenAI 임베딩으로 변환
def get_openai_embedding(text):
    response = openai.Embedding.create(
        input=text,
        model="text-embedding-ada-002"
    )
    return response['data'][0]['embedding']

rule_embeddings = np.array([
    get_openai_embedding(rule) for rule in tqdm(rules, desc="임베딩 생성
중")
])

# 질문 파일 읽기
questions_df = pd.read_excel(questions_path)
if "질문" not in questions_df.columns:
    raise ValueError("엑셀 파일에 '질문' 컬럼이 없습니다.")

# 질문 처리 및 답변 생성
def get_answer(question):
    # 질문 임베딩 생성
    question_embedding = get_openai_embedding(question)

    # FAISS 없이 가장 유사한 규정 찾기 (코사인 유사도 계산)
    similarities = np.dot(rule_embeddings, question_embedding) / (
        np.linalg.norm(rule_embeddings, axis=1) *
np.linalg.norm(question_embedding)
    )
    closest_rule_idx = np.argmax(similarities)
    closest_rule = rules[closest_rule_idx]
    prompt= (
                f"다음 규정을 바탕으로 질문에 답변하세요:\n\n규정:
{closest_rule}\n\n질문: {question}\n\n답변:"
                f"답변은 친근하고 명확한 존댓말로 작성하며 자세하게 해주세요. "
                f"인사는 생략하세요.\n"
                f"규정 내용에 해당하지 않는 질문은 '해당 질문은 대답할 수 없습니
다.'라고 대답하세요.\n\n"
                )
    # OpenAI API를 사용하여 답변 생성
    try:
        response = openai.ChatCompletion.create(
                model="gpt-4o-mini",  # 올바른 모델 이름으로 변경
                messages=[{"role": "system", "content": prompt}],
                temperature=0.0,
                max_tokens=3000,
                n=1,
                stop=None,
            )
        return response['choices'][0]['message']['content'].strip()
    except Exception as e:
        return f"답변 생성 중 오류 발생: {e}"
```

3. FAQ 데이터 전처리(EXCEL → JSONL 변환)

1. 문서 기반 정확성 검증:
    - 생성된 FAQ 답변이 원본 문서(규정 내용)와 상충되지 않도록 검증
    - LLM 답변 중 문서에 언급되지 않은 정보(환각)가 포함되어 있는지 확인
    - 문서와 불일치하는 내용은 수정하거나 "해당 질문은 대답할 수 없습니다."로 생성되었는지 확인
2. 데이터 형식 변환
    - 원본 Excel 파일을 JSONL 변환(코드 내 exel_to_json 함수 사용)

3. 데이터 추출 방식
    - Excel → JSONL 변환 후 Pandas DataFrame 로딩
    - 불필요 항목 제거 및 정규표현식(Regex) 사용한 텍스트 정제

4. 불필요한 데이터 제거 기준
    - 중복 (질문+카테고리 조합)
    - 필수 필드(참고 문맥, 질문, 카테고리, 답변) 중 하나라도 결측치 발생 시 제거

5. 정제 방법
    - Regex(정규표현식)를 통한 특수문자, 기호 제거
    - 문자열 전처리(Strip, Replace)
    - 전처리 함수(preprocess_item)로 각 항목 처리 후 리스트화

6. 전처리 코드

```
# Step 1: Convert a single Excel file to JSONL
def excel_to_jsonl_single_file(excel_file, output_file):
    """
    Converts a single Excel file to JSONL format.

    Args:
        excel_file (str): Path to the input Excel file.
        output_file (str): Path to save the output JSONL file.
    """
    try:
        # Read the Excel file
        df = pd.read_excel(excel_file)
        # Convert DataFrame to a list of dictionaries
        data = df.to_dict('records')

        # Write the data to a JSONL file
        with open(output_file, 'w', encoding='utf-8') as f:
            for item in data:
                json.dump(item, f, ensure_ascii=False)
                f.write('\n')

        print(f"Converted {excel_file} to JSONL format. Saved as
{output_file}")
    except Exception as e:
        print(f"Error processing {excel_file}: {e}")

# Step 2: Preprocess data and split into train/validation
def preprocess_item(item):
    """
    Preprocess a single JSON object to apply the required transformations.
    """

    # 1. 숫자 기호 (①, ②, ...)를 일반 숫자로 변환
    source = re.sub(r"①", "1", source)
    source = re.sub(r"②", "2", source)
    source = re.sub(r"③", "3", source)
    source = re.sub(r"④", "4", source)
    source = re.sub(r"⑤", "5", source)
    source = re.sub(r"⑥", "6", source)
    source = re.sub(r"⑦", "7", source)  # 추가: ⑦ -> 7
    source = re.sub(r"⑧", "8", source)  # 추가: ⑧ -> 8

    # 2. 특수문자 ●, □, ◦, ※ 제거
    source = re.sub(r"[●□◦※]", "", source)

    # 3. '1. 삭제', '2. 삭제' 등 번호와 함께 있는 '삭제' 제거
    source = re.sub(r"\d+\.\s*삭제", "", source)

    # 4. 기타 불필요한 문자 제거
    source = re.sub(r"\n", "", source)
    source = re.sub(r"\*\*.*?\*\*", "", source)  # Remove Markdown bold or
italic
    source = re.sub(r"•", "", source)
    source = source.strip()
    if source.endswith("."):
        source = source[:-1]

    def save_to_jsonl(data, file_path):
        with open(file_path, 'w', encoding='utf-8') as f:
            for item in data:
                json.dump(item, f, ensure_ascii=False)
                f.write('\n')

    save_to_jsonl(train_data, train_file)
    save_to_jsonl(val_data, val_file)
```

데이터 전처리 결과	**1. PDF** 전처리 결과 **(JSONL** 데이터**):** ● 문서 수: 4 ● 항목(조, 주제별) 수: 192 ● 데이터 구조: 장, 조, 주제, 내용 **2. FAQ** 데이터 전처리 결과: ● 각 주제별 평균 10~20개 질문 확보 (총 약 1600개) ● 구조화된 형식으로 구성된 FAQ 데이터셋 완성 ● 전처리 완료 후 데이터는 Train/Validation 세트로 분리 - Train 세트: 약 1300개 샘플 - Validation 세트: 약 300개 샘플 **3.** 향후 데이터 사용계획**:** ● 전처리된 데이터셋을 사용하여 sLLM 모델 파인튜닝 ● 정확도 향상을 위해 지속적으로 필요 데이터 생성 예정 ● 향후 신규 문서 확보 시 기존 JSONL 구조에 추가 가능 ● 문서 다양화에 따라 필요시 추가 계층 도입 가능