SK네트웍스 **Family AI** 과정 **12**기

모델링 및 평가 LLM 활용 소프트웨어

산출물 단계	모델링 및 평가
평가 산출물	LLM 활용 소프트웨어
제출 일자	2025.08.01
깃허브 경로	https://github.com/SKNETWORKS-FAMILY-AICAMP/SKN12-FINAL-6TEAM
작성 팀원	이지복

*활용한 LLM 모델 파일명 기재하여 파일과 함께 제출

모델 파일 이름 : best_keyword_classifier.pth

1. 개요 (Overview)

- **1-1. 목표**
  - 그림 분석 후 얻은 감정 키워드를 활용하여 페르소나 유형화를 하는것에 있음
- **1-2. 배경**
  - 프로젝트의 목표에 맞게 특정 문서에서 필요한 감정 키워드를 추출해냄
  - 추출 및 정제 등의 전처리 과정을 거친 후 BERT 모델을 학습시킴

2. 활용 모델

모델명: KLUE BERT

특징: 한국어로 사전 학습된 BERT 모델로, 한국어에 특화된 다양한 언어이해 과제를 위한 벤치마크와 데이터셋, 그리고 해당 데이터셋에 맞춰 학습한 언어모델임

	YNAT	KLUE-STS		KLUE-NLI	KLUE-NER		KLUE-RE		KLUE-DP		KLUE-MRC		WoS	
Model	F1	R^P	F1	ACC	$F1^E$	$F1^C$	$F1^{mic}$	AUC	UAS	LAS	EM	ROUGE	JGA	$F1^S$
mBERT$_{BASE}$	81.55	84.66	76.00	73.20	76.50	89.23	57.88	53.82	90.30	86.66	44.66	55.92	35.46	88.63
XLM-R$_{BASE}$	83.52	89.16	82.01	77.33	80.37	92.12	57.46	54.98	89.20	87.69	27.48	53.93	39.82	89.61
XLM-R$_{LARGE}$	**86.06**	92.97	85.86	85.93	82.27	**93.22**	58.39	61.15	92.71	**88.70**	35.99	66.77	41.20	89.80
KR-BERT$_{BASE}$	84.58	88.61	81.07	77.17	74.58	90.13	62.74	60.94	89.92	87.48	48.28	58.54	45.33	90.70
KoELECTRA$_{BASE}$	84.59	92.46	84.84	85.63	**86.11**	92.56	62.85	58.94	92.90	87.77	59.82	66.05	41.58	89.60
KLUE-BERT$_{BASE}$	85.73	90.85	82.84	81.63	83.97	91.39	66.44	66.17	89.96	88.05	62.32	68.51	46.64	91.61
KLUE-RoBERTa$_{SMALL}$	84.98	91.54	85.16	79.33	83.65	91.14	60.89	58.96	90.04	88.14	57.32	62.70	46.62	91.44
KLUE-RoBERTa$_{BASE}$	85.07	92.50	85.40	84.83	84.60	91.44	67.65	68.55	93.04	88.32	68.67	73.98	47.49	91.64
KLUE-RoBERTa$_{LARGE}$	85.69	**93.35**	**86.63**	**89.17**	85.00	91.86	**71.13**	**72.98**	**93.48**	88.36	**75.58**	**80.59**	**50.22**	**92.23**

출처 : https://arxiv.org/abs/2105.09680

2. 모델 관리 (Model Deployment & Management)

- **2-1. Hugging Face Hub 활용**
  - 사전 학습된 모델의 버전 관리, 손쉬운 접근성, 다른 개발자와의 협업 용이성을 고려하여 Hugging Face Hub에 배포된 모델을 활용함
- **2-2. 활용 모델 정보**
  - 모델의 Hugging Face Hub 경로 : **Bokji/HTP-personality-classifier**
  - **허깅페이스 다운로드 주소 :** https://huggingface.co/Bokji/HTP-personality-classifier/resolve/main/best_keyword_classifier.pth
  - **모델 파일 이름 : best_keyword_classifier.pth**

3. 모델 활용 및 적용 (Model Utilization & Application)

- **3-1. 실행 환경**
  - 학습된 모델을 사용하기 위해 필요한 라이브러리 목록
    - torch == 2.1.0
    - transforemers == 4.35.0
    - huggingface_hub == 0.19.4

- **3-2. 핵심 소스 코드**
  - **1) 모델 불러오기**

```python
def _load_model(self):
    """허깅페이스에서 사전 학습된 BERT 모델 로드"""
    if not HF_TOKEN or not HF_MODEL_NAME:
        raise ValueError("HF_TOKEN과 HF_MODEL_NAME이 설정되어야 합니다")

    try:
        self.logger.info(f"허깅페이스에서 BERT 모델 다운로드 중: {HF_MODEL_NAME}")

        # 허깅페이스에서 모델 파일 직접 다운로드
        from huggingface_hub import hf_hub_download
        import tempfile

        model_file = hf_hub_download(
            repo_id=HF_MODEL_NAME,
            filename="best_keyword_classifier.pth",
            token=HF_TOKEN,
            cache_dir=tempfile.gettempdir(),
            force_download=False
        )

        self.logger.info(f"모델 파일 다운로드 완료: {model_file}")

        # 다운로드된 모델 로드
        checkpoint = torch.load(model_file, map_location='cpu')

        # 모델이 직접 객체인지 확인
        if hasattr(checkpoint, 'eval') and hasattr(checkpoint, 'forward'):
            self.model = checkpoint
            self.model.eval()
            self.logger.info("허깅페이스 모델 로드 완료 (직접 모델 객체)")
            self.hf_model = self.model
            return
```

  - **상세 설명**
    - **① 허깅페이스 모델 정보와 토큰 확인**
      - 모델을 다운로드하기 전, 허깅페이스 토큰(HF_TOKEN)과 모델 이름(HF_MODEL_NAME)이 올바르게 설정되어 있는지 확인함

        - 만약 둘 중 하나라도 누락되어 있으면 오류를 발생시켜 실행을 중단함
    - **② 허깅페이스에서 모델 다운로드 시도**
        - 로그를 남기면서, 허깅페이스 Hub에서 지정한 모델 파일을 다운로드함
        - 인증 토큰과 임시 폴더 경로를 사용하여 모델 파일을 저장함
    - **③ 다운로드 완료 및 파일 경로 확인**
        - 모델 파일이 정상적으로 다운로드되면, 파일의 경로를 로그에 기록하여 파일 위치를 확인할 수 있도록 함
    - **④ 다운로드된 모델 파일 로드**
        - PyTorch의 `torch.load`를 이용해 다운로드한 모델 파일을 메모리로 불러옴
        - 불러온 객체가 실제로 PyTorch 모델 객체인지 확인함
    - **⑤ 모델 객체 등록 및 평가 모드 전환**
        - 로드한 모델을 `self.model`과 `self.hf_model`에 할당하고, 평가 모드로 전환하여 추론할 준비함
        - 최종적으로 모델 로드가 완료되었다는 로그를 남김

- **2) 키워드 추출**

```python
"""텍스트에서 감정 키워드 추출 (GPT 키워드 섹션 우선 파싱)"""
def _extract_emotion_keywords(self, text: str) -> List[str]:

    extracted = []

    # GPT가 추출한 "주요 감정 키워드" 섹션을 직접 추출
    gpt_keywords = self._parse_gpt_keywords_section(text)
    if gpt_keywords:
        extracted.extend(gpt_keywords)
        self.logger.info(f"GPT 키워드 섹션에서 추출: {gpt_keywords}")


    return list(set(extracted))  # 중복 제거
```

- 상세 설명
    - **① GPT가 추출한 감정 키워드 섹션 파싱**

- 먼저 입력받은 텍스트에서 GPT가 이미 뽑아준 “주요 감정 키워드” 섹션이 있는지 확인하고, 이 부분을 우선적으로 파싱하여 감정 키워드를 추출함
- GPT 키워드가 존재할 경우, 추출된 키워드들을 리스트에 추가하고, 추출 결과를 로그에 기록함

- ■ **② 중복 제거 및 결과 반환**
  - 중복된 키워드가 있을 수 있으므로, set 자료형으로 변환하여 중복을 제거한 후 최종적으로 리스트 형태로 반환함

- **3) 페르소나 분류**

```python
def predict_keyword(keyword, model, tokenizer, label_encoder, max_length=64):
    """키워드에 대한 성격 유형을 예측합니다."""
    # 토크나이징
    encoding = tokenizer(
        keyword,
        truncation=True,
        padding='max_length',
        max_length=max_length,
        return_tensors='pt'
    )

    input_ids = encoding['input_ids'].to(device)
    attention_mask = encoding['attention_mask'].to(device)

    # 예측
    with torch.no_grad():
        outputs = model(input_ids, attention_mask)
        probabilities = F.softmax(outputs, dim=1)
        predicted_class = torch.argmax(probabilities, dim=1).item()
        confidence = probabilities[0][predicted_class].item()

    predicted_label = label_encoder.inverse_transform([predicted_class])[0]

    # 모든 라벨별 확률
    all_probabilities = {}
    for idx, prob in enumerate(probabilities[0]):
        label = label_encoder.inverse_transform([idx])[0]
        all_probabilities[label] = prob.item()

    return predicted_label, confidence, all_probabilities
```

- **상세 설명**
  - ■ **① 키워드 토크나이징**
    - 입력받은 키워드를 자연어 처리 모델이 이해할 수 있도록 토크나이저(tokenizer)를 이용해 토큰화함
    - 토큰화 과정에서 최대 길이(`max_length`)로 잘라내거나 패딩을 추가하여 입력 형태를 맞춤

- 토크나이즈된 결과를 파이토치 텐서(input_ids, attention_mask)로 변환함

■ **② 키워드 토크나이징**

- 토크나이즈된 입력을 모델에 넣고, 추론모드(torch.no_grad())에서 예측을 수행함
- 모델의 출력값에 소프트맥스(softmax)를 적용해 각 클래스별 확률을 계산함
- 확률이 가장 높은 클래스의 인덱스를 선택하여 예측 결과를 얻고, 해당 확률값을 신뢰도로 저장함

■ **③ 예측 결과 라벨 복원**

- 예측된 클래스 인덱스를 라벨 인코더(label_encoder)를 사용해 원래의 라벨명(성격 유형 등)으로 변환함

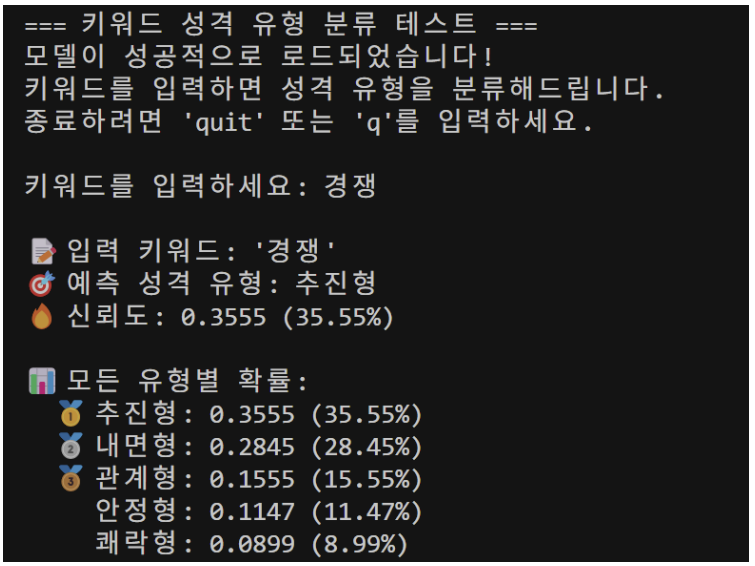
■ **④ 전체 라벨별 확률 계산**

- 각 클래스(라벨)별 확률값도 모두 라벨명과 함께 딕셔너리 형태로 저장함

■ **⑤ 최종 결과 반환**

- 예측된 라벨, 신뢰도(확률), 전체 라벨별 확률 정보를 반환함

- **3-3. 실행 예시 및 결과**



4. 결론 및 향후 과제

4-1. 결론

- Hugging Face Hub에 업로드된 사전학습 BERT 모델을 애플리케이션에 성공적으로 통합함
- 키워드 기반 성격 유형 분류기를 구축하여 사용자가 별도의 복잡한 과정 없이도 성격 유형 예측 기능을 활용할 수 있도록 구현함
- 본 시스템을 기반으로 실제 서비스에 키워드 분류 기능을 안정적으로 제공할 수 있는 기술적 기반을 마련함
- 키워드 분류 기능을 통해 사용자 경험의 향상 및 개인화 서비스 확대에 기여할 수 있는 가능성을 확인함

4-2. 한계점 및 향후 과제

- 현재 모델은 주어진 키워드에 대해 사전학습된 지식 내에서만 분류를 수행하므로, 의미적으로 유사하거나 새로운 키워드에 대한 대응력이 부족함
- 비슷한 의미의 다양한 표현, 복합 감정이 내포된 키워드 등에는 분류 정확도가 제한적임
- 이러한 키워드를 추가 수집하여 모델을 파인튜닝할 필요가 있음
- 데이터 다양성과 라벨의 정밀도를 높이기 위한 추가적인 자연어 처리 기법 및 데이터 증강 방법 도입이 요구됨
- 모델의 성능을 지속적으로 개선하고, 다양한 도메인 및 상황에 대응할 수 있는 시스템 고도화가 필요함