



Wrap-up Report

Wrap-up Report 목차

- [1 프로젝트 개요](#)
- [2 프로젝트 팀 구성 및 역할](#)
- [3 프로젝트 수행 절차 및 방법](#)
- [4 프로젝트 수행 결과](#)
 - [4.1 데이터 분석](#)
 - [4.2 학습 설계 및 실험 결과 분석](#)
 - [4.2.1 강정완](#)
 - [4.2.2 김민선](#)
 - [4.2.3 서선아](#)
 - [4.2.4 이인구](#)
 - [4.2.5 이재협](#)
 - [4.2.6 임상엽](#)
- [5 최종 제출 결과](#)
- [6 자체 평가 의견](#)
- [Appendix](#)

1 프로젝트 개요

1. 과제 소개

Topic Classification은 주어진 자연어

문장이 어떤 주제나 카테고리에 속하는지를 분류하는 NLP Task이다. 모델이 자연어의 주제를 얼마나 잘 이해하는지 평가할 수 있는 기본적인 방법 중 하나로, 특히 KLUE-Topic Classification benchmark에서는 뉴스 헤드라인을 보고 해당 뉴스가 어떤 주제를 가지는지 분류하는 작업이 요구된다.

본 대회는 데이터 품질 개선을 통해 모델의 성능을 높이는 Datacentric AI 접근법에 초점을 두고 있기 때문에 모델 개선은 할 수 없으며, 데이터셋을 정제할 때에도 여러 제한 사항을 준수하여야 한다. 특히, 실제 KLUE-Topic Classification benchmark에는 생활문화(Society), 스포츠(Sports), 세계(World), 정치(Politics), 경제(Economy), IT과학(IT/Science), 사회(Society)라는 주제가 공개되어 있으나, 본 대회에서는 이러한 라벨 매핑 정보는 사용할 수 없으며 오직 정수 인코딩 정보만 사용하여야 한다. 또한 외부 데이터셋(크롤링 포함)을 사용할 수 없고, 유료 결제가 필요한 비공개 생성형 모델도 사용할 수 없다.

2. 평가 지표

평가 지표는 Macro F1 Score이며, 이는 모든 라벨에 동등한 중요도를 부여하기 위함이다. Macro F1 Score는 라벨 별 f1 score의 평균으로 계산한다.

$$f1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$Macro\ f1\ score = \frac{1}{num_label} \sum_{n=0}^{num_label} f1\ score$$

2 프로젝트 팀 구성 및 역할

이름	역할
강정완	텍스트 노이즈 클리닝 (by T5), 클러스터링 기법을 활용해 라벨 오류 보정 (by t-SNE, GMM)
김민선	노이즈 데이터 증강 (by LLM), 데이터 증강 (by Back Translation)
서선아	텍스트 노이즈 클리닝 (by LLM), 라벨 오류 보정 (by LLM), 데이터 증강 (by LLM)
이인구	데이터 증강(by Back Translation)
이재협	텍스트 노이즈 클리닝 (by LLM), 데이터 증강 (by LLM), 라벨 오류 보정 (by Cleanlab)
임상엽	텍스트 노이즈 클리닝 (by LLM), 라벨 오류 보정 (by LLM, Clustering), 데이터 증강 (by LLM), 작업류 구현

3 프로젝트 수행 절차 및 방법

- 팀원들이 각자 자신의 아이디어를 가지고 프로젝트 전반부터 후반까지 진행했습니다.

2024년 11월

일	월	화	수	목	금	토
27	28	29	30	31	11월 1일	2
	데이터, 베이스라인 코드 살펴보기			EDA	라벨 복원, 텍스트 클리닝, 데이터 증강 실험	
3	4	5	6	7	8	9
라벨 복원, 텍스트 클리닝, 데이터 증강 실험				데이터 병합 실험		

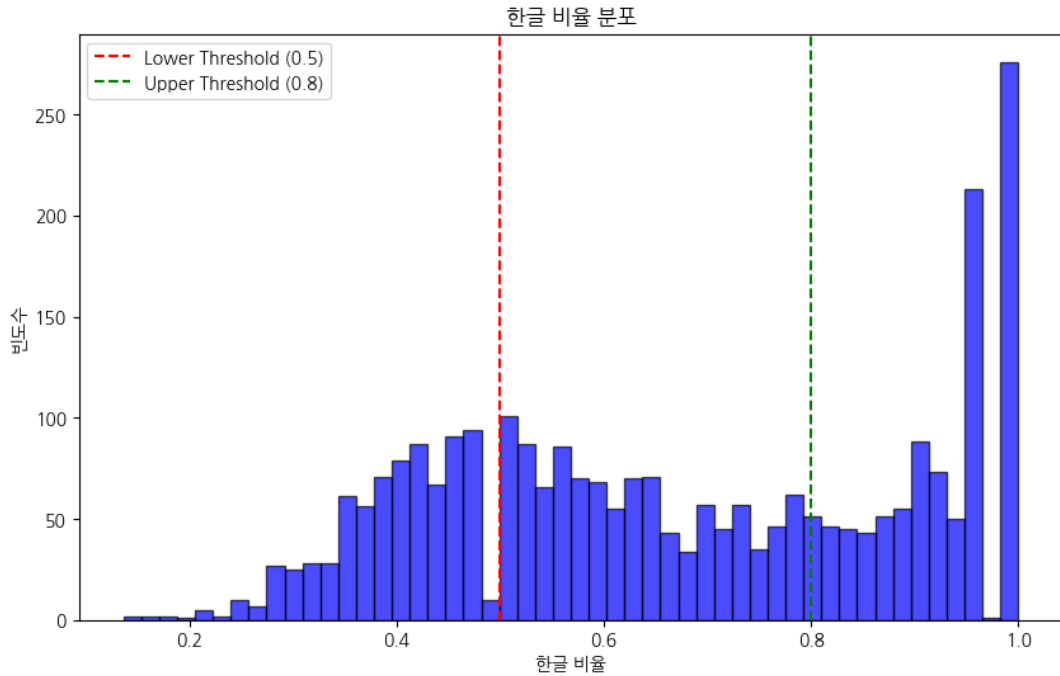
4 프로젝트 수행 결과

4.1 데이터 분석

대회 측에서 제공한 학습 데이터는 2,800개다. 이 중 1,000개에는 라벨 오류가 있고, 1,600개에는 임의로 텍스트 노이즈(무의미한 ASCII 코드)가 포함되었다. 따라서 정상 데이터는 단 200개만 존재한다.

텍스트 노이즈가 있는 데이터는 텍스트 클리닝을 통해, 라벨 오류가 있는 데이터는 라벨 보정을 통해 사용 가능하게 만들어야 한다.

아래 그래프는 학습 데이터셋의 텍스트 노이즈(한글을 제외한 문자) 비율 분포를 나타낸다.



위 그래프를 통해 텍스트 노이즈가 과도한 데이터와 적정 수준인 데이터의 비율을 직관적으로 파악할 수 있었다. 즉, 텍스트 클리닝이 어려운 데이터와 쉬운 데이터의 비율을 쉽게 확인할 수 있었다.

4.2 학습 설계 및 실험 결과 분석

4.2.1 강정완

1. 노이즈 텍스트 클리닝 (by T5)

기대 효과: T5 모델은 입력 텍스트 내에서 임의의 연속된 토큰을 선택하여 `<extra_id_#>` 토큰으로 마스킹하고, 해당 구간의 원래 텍스트를 예측하는 방식으로 사전 학습이 진행된다. 이러한 사전 학습 방식 덕분에, 문맥에 맞는 텍스트를 적절히 복원할 것으로 기대했다.

노이즈 텍스트 클리닝 과정:

1. ASCII 코드가 노이즈로 포함되었기 때문에, 한글과 한자를 제외한 모든 문자를 마스킹했다. (by 정규표현식)

`pI美대선I앞두고 R2fr단 발] $비해 감시 강화` →

`<extra_id_0> 美 대선 <extra_id_1> 앞두고 <extra_id_2> 단 발 <extra_id_3> 비해 감시 강화`

2. 마스킹된 입력을 모델에 전달해 원래 텍스트를 예측하도록 했다. (by mT5-xl)

`<pad> <extra_id_0> 트럼프 <extra_id_1> 을 <extra_id_2> 트럼프 <extra_id_3> 연에`

3. 모델이 예측한 텍스트를 각 마스킹 구간에 넣어준다.

`트럼프 美대선을 앞두고 트럼프에 대한 단 발언에 비해 감시 강화`

실험 결과: accuracy: 0.7684(+0.1445), f1-score: 0.7607(+0.1652)

결과 분석: 예상대로 mT5-xl 모델의 우수한 텍스트 복원 능력 덕분에 기사 주제에 맞는 텍스트 클리닝이 적절히 이루어졌고, 이로 인해 성능이 향상되었다고 판단했다.

▼ 클리닝 예시

北조국통!민주x#전선 결성 <0돌 중;보고회 열x → 北조국통진민주화전선 결성, 돌중간 보고회 열려

박항c 매직c베트남i 축구_표팀K.??# 쓴;*:d → 박항구 매직티켓베트남어 축구 경기 표팀 내 쏟아

추가 실험: 노이즈 텍스트를 클리닝한 데이터를 구글 번역기로 역번역하여 2배로 증강한 후 학습에 포함했다. 그 결과, 성능이 더욱 향상되었다.

	baseline	clean_by_mT5	clean_by_mT5 + BT
accuracy	0.6239	0.7684(+0.1445)	0.8214(+0.1975)
F1-score	0.5955	0.7607(+0.1652)	0.8174(+0.2219)

2. 클러스터링 기법을 활용해 라벨 오류 보정 (by t-SNE, GMM)

기대 효과: `clean_by_mT5 + BT` 데이터로 학습한 모델이 80% 이상의 정확도를 보였으므로, 텍스트의 표현 벡터를 비교적 적절히 나타낸다고 볼 수 있다. 즉, 동일한 라벨을 가진 텍스트들은 서로 유사한 표현 벡터를 형성한다고 추론할 수 있다. 따라서, 텍스트의 표현 벡터와 유사한 집합의 라벨로 보정하는 방식이 효과적일 것으로 예상했다.

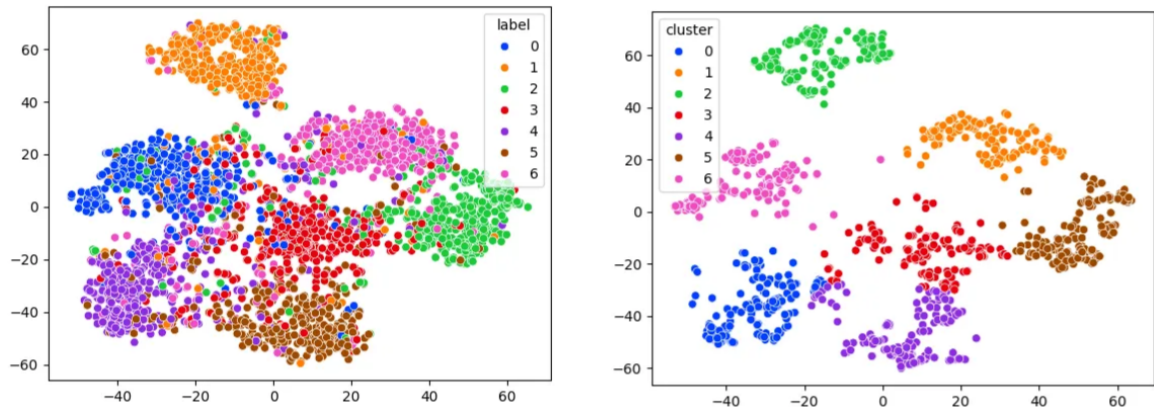
라벨 오류 보정 과정:

1. `clean_by_mT5 + BT` 데이터로 학습한 모델로 텍스트 표현 벡터(최종 CLS 벡터)를 추출한다.
2. t-SNE 기법으로 표현 벡터를 저차원으로 변환한다. (512차원 → 2차원)
3. GMM 기법으로 클러스터링 모델을 만들어, 라벨 오류가 있는 데이터의 라벨을 보정한다.

실험 결과: 라벨 오류를 보정한 데이터와 해당 데이터를 역번역하여 증강한 데이터를 기존 모델에 추가한 결과, 성능이 추가로 향상되었다. accuracy: 0.8398(+0.2159), f1-score: 0.8362(+0.2407)

	baseline	clean_by_T5	clean_by_T5 + BT	clean_by_T5 + BT & restore_label_by_cluster + BT
accuracy	0.6239	0.7684(+0.1445)	0.8214(+0.1975)	0.8398(+0.2159)
F1-score	0.5955	0.7607(+0.1652)	0.8174(+0.2219)	0.8362(+0.2407)

결과 분석: 정상 라벨 데이터로 클러스터링을 수행한 결과, 완벽하지는 않지만 라벨 별로 적절한 클러스터가 형성되었다. 따라서 모든 라벨 오류 데이터를 보정하지는 못했더라도, 대부분의 라벨 오류 데이터는 올바르게 보정되었을 것으로 판단된다.



4.2.2 김민선

1. 노이즈 데이터 증강(by LLM)

텍스트에 노이즈가 포함된 데이터는 라벨이 정상인 데이터이기 때문에 해당 데이터들을 증강하여 라벨이 정확한 데이터들로 학습할 수 있도록 하고자 하였다.

gemma2 모델을 활용하여 아래와 같은 프롬프트를 모델에 입력으로 제공했다.

```

"""
지시 사항:

{batch_size}개의 텍스트로 구성된 목록을 확인하고, 각 항목의 맥락과 유사한 3개의 뉴스 기사
스타일 문장을 생성하세요.
원본 텍스트에 아스키 코드 노이즈가 많이 포함된 경우, 생성된 텍스트에는 노이즈를 포함하지 않
원본의 주요 의미를 최대한 보존합니다.
생성된 텍스트는 한 단어로 구성되어서는 안 되며, 심표(,)가 포함되지 않도록 합니다.
추가 설명이나 부가적인 정보는 출력에 포함하지 않고, 아래의 출력 형식에 엄격히 맞춰 작성하세

{input}

출력 형식:
각 입력 항목에 대해 다음과 같은 형식으로 출력을 생성하세요:
1. 원본 텍스트1, 생성 텍스트 1-1, 생성 텍스트 1-2, 생성 텍스트 1-3
2. 원본 텍스트2, 생성 텍스트 2-1, 생성 텍스트 2-2, 생성 텍스트 2-3

...

"""

```

프롬프트를 보면 알 수 있듯이 한 데이터로부터 총 3개의 새로운 데이터를 생성하도록 했다. 노이즈의 정도를 기준으로 데이터를 필터링하지 않고 노이즈 여부로만 필터링했기 때문에 텍스트 노이즈를 잘 걸러낸 데이터를 생성하는 것이 힘들 수도 있겠다고 예상했지만 생성된 데이터를 보니 라벨을 예측하기에 문제없을 정도로 품질이 개선된 것을 확인할 수 있었다. 예를 들어, 원래 데이터에서 `정i :파1 미사z KT(이용기간 2e 단] Q분총U2보` 라고 적

허있던 데이터를 통해 다음과 같은 3개의 데이터를 생성했다. **KT 이용 기간 2년으로 확대 예정, KT 이용기간 연장 논의, KT 이용 기간 2년 연장 안 발표**

이렇게 생성된 데이터를 모두 학습에 사용하게 되면 의미가 유사한 데이터를 여러번 학습하게 되는 것이기 때문에 과적합을 우려하였고, 원본 데이터 기준으로 생성된 3개의 데이터 중 1개만 랜덤으로 선택하여 남겼다. 하지만 그렇게 생성한 데이터를 통해 학습한 결과, baseline보다 성능이 크게 떨어졌다.

	baseline	01
accuracy	0.6239	0.3289(-0.2950)
F1	0.5955	0.2987(-0.2968)

2. 데이터 증강(by Back Translation)

1번에서 새롭게 생성된 데이터만 가지고 학습했기 때문에 데이터의 양이 부족해서 성능이 나빠졌을 것이라고 판단되어 1번에서 생성한 데이터에 한글 → 영어 → 한글 단계를 거치는 Back Translation 과정을 적용했다. google 번역기를 활용하였으며 번역 예시는 다음과 같다.

KT 이용 기간 2년으로 확대 예정 → 2 년의 KT 사용 기간으로 확장 될 예정입니다.

1번, 2번 데이터를 병합하여 학습한 결과도 역시, baseline보다 성능이 크게 떨어졌다.

	baseline	02
accuracy	0.6239	0.3280(-0.2959)
F1	0.5955	0.2983(-0.2972)

3. 라벨 에러 복원(by Cleanlab)

위의 과정을 진행할 때 텍스트 데이터에 노이즈가 얼마나 포함되어 있는지만 판단한 후 학습을 진행했고, 그 결과 베이스라인보다 크게 떨어지는 성능을 확인했다. 이는 모델이 생성한 텍스트에 원본 데이터의 라벨을 가져와 붙여주는 작업이 잘못되어 라벨이 잘못된 데이터를 생성했기 때문이라고 예상했다. 실제로 확인하니 원본 데이터와 새로 생성한 데이터의 라벨이 달랐다. 따라서 그 라벨 에러를 복원할 필요성을 느꼈고, Cleanlab을 적용하여 문제를 해결하기로 했다. 그렇게 Cleanlab을 통해 1번 데이터의 라벨을 수정한 데이터(16)로 학습한 결과, 성능이 baseline보다 향상되었다.

	baseline	16
accuracy	0.6239	0.6735(+0.0496)
F1	0.5955	0.6967(+0.1012)

4. 라벨 에러 복원(by Clustering)

위와 같은 이유로 1번, 2번 데이터의 라벨을 복원하기 위해 Clustering도 활용하기로 했다. 1번 데이터만 Clustering을 통해 데이터의 라벨을 수정한 데이터로 학습한 결과, 성능이 baseline보다 성능이 떨어졌다.

	baseline	18
accuracy	0.6239	0.4899(-0.1340)
F1	0.5955	0.5375(-0.0580)

데이터의 양이 부족한 것이 원인이었다고 판단되어 1번, 2번 데이터를 병합한 후 Clustering을 적용한 데이터로 학습한 결과, 성능이 baseline보다 향상되었다.

	baseline	19
accuracy	0.6239	0.6581(+0.0342)
F1	0.5955	0.6947(+0.0992)

4.2.3 서선아

이번 프로젝트의 목표

- 노이즈가 섞인 데이터들을 가공하는 것
- 잘못 라벨링된 데이터를 찾아 수정하는 것
- 모델에게 정상적인 데이터들을 최대한 많이 주입하는 것

1. 텍스트 노이즈 복구(by LLM)

목표: '노이즈가 없는 정상 text들'의 라벨링 에러를 복원하는데 이용하기 위하여, 라벨링 에러가 없는 noised text에서 노이즈를 제거하고자 하였다.

gemma2:27B를 이용해 프롬프트 엔지니어링을 진행했다. 전체 텍스트 데이터 중에서 노이즈가 존재하는 텍스트를 분류한 후, chatgpt의 prompt maker를 사용하여 프롬프트를 작성해 노이즈를 제거하였다. 노이즈를 잘 제거하는 프롬프트의 경우, 텍스트가 '뉴스 기사 제목'임을 밝혀야 했다. 처음엔 노이즈가 상대적으로 적은 텍스트들이 복원이 잘 될 것으로 예상했으나, 노이즈가 심한 경우에도 LLM이 유추를 잘 해주었다. 예를 들어, `pI美대선I앞두고 R2fr단 발] $비해 감시 강화` → `미국 대선 앞두고 러시아 정보 단체 감시 강화` 와 같은 경우, 기존 내용을 사람도 유추하기 어려웠지만, LLM이 창의적으로 잘 생성해내었다. 이렇게 모든 noised 데이터를 LLM으로 복원하여 baseline을 새로 학습시킨 결과, 성능이 유의미한 정도로 향상된 것을 확인할 수 있었다.

	baseline	22
accuracy	0.6239	0.7613(+0.1374)
F1	0.5955	0.7591(+0.1636)

2. 라벨 에러 복구(by LLM)

목표: LLM을 사용하여 라벨 오류가 없는 데이터를 참조하여 오류가 있는 데이터의 라벨을 정정한다.

먼저, 위에서 텍스트 노이즈가 복구된 라벨 오류가 없는 데이터셋으로부터 LLM을 이용해 라벨 의미를 추론하였다.

```
# 프롬프트 생성 함수 (5개 문장씩)
def create_partial_meaning_prompt(target, texts):
    texts_joined = "\n".join([f"- {text}" for text in texts])
    return f"""
    다음은 라벨 {target}에 해당하는 뉴스 제목들입니다. 이 문장들로부터 이 라벨이 주로 어

    문장들:
    {texts_joined}

    이 라벨이 주로 나타내는 주제를 간단하게 설명해 주세요.
    """

# 각 라벨에 대해 누적된 의미를 바탕으로 대표 주제 도출
# 대표 주제 도출 프롬프트 함수
def summarize_label_meaning_prompt(target, partial_meanings):
    partial_meanings_text = "\n".join([f"- {meaning}" for meaning in partial_meanings])
    return f"""
    라벨 {target}에 대해 각 뉴스 제목에서 추론한 주제들이 아래와 같습니다.:
    """
```

```
{partial_meanings_text}
```

```
이 라벨이 나타내는 대표적인 주제를 한 문장으로 요약해 주세요.  
"""
```

추론한 라벨 의미를 참조하여 다시 LLM으로 라벨 오류가 있는 데이터의 라벨을 복구하도록 하였다. 아래는 해당 작업을 위해 사용한 프롬프트를 생성하는 함수이다.

```
def create_label_correction_prompt(text, target):  
    return f"""  
    뉴스 제목: "{text}"  
    현재 라벨: {target}  
  
    각 라벨의 의미는 다음과 같습니다:  
    {label_descriptions}  
  
    위 라벨 설명을 참고하여 7개의 라벨 중 뉴스 제목에 가장 잘 맞는 라벨을 추천해 주세요.  
    뉴스 제목의 키워드를 잘 활용하세요.  
    응답 형식은 반드시 아래와 같이 작성해 주세요. 추가 설명이나 불필요한 내용은 포함하지 마세요.  
  
    형식: "수정된 라벨: X" (X는 0~6 중 하나의 숫자만 입력)  
  
    예:  
    수정된 라벨: 5  
    """
```

베이스라인과 비교하여 성능이 올라간 것을 확인할 수 있지만, 텍스트 노이즈를 제거한 것보다 성능이 떨어진 것으로 나타났다. 그 이유는 LLM이 라벨링을 잘못하였고, 특히 LLM이 유추한 라벨의 의미들을 확인해 보았을 때, 라벨의 의미를 명확히 구분하지 못하며 각각 특정 세부 주제에 편향되게 라벨의 의미를 추론한 것을 확인할 수 있었다. 예를 들어 2번의 경우, 한반도 정세에 대한 것에 집중하여 해석하였고, 3번의 경우 사회와 정치를 포괄하여 해석하였다. 따라서, 라벨링 오류를 제거하기 위하여 데이터를 증강시킬 필요가 있었다.

	baseline	22	7 (라벨 에러 복구)
accuracy	0.6239	0.7613(+0.1374)	0.7375(+0.1136)
F1	0.5955	0.7591(+0.1636)	0.7199(+0.1244)

3. 데이터 증강(by LLM)

데이터 증강 시 2번과 3번뿐만 아니라 모든 라벨에서 다양한 텍스트를 생성할 수 있었던 것까지는 좋았지만, 증강한 데이터들이 올바른 라벨링으로 이어졌는지를 검토하는 것이 성능에 도움이 되었을 것으로 보인다. 아래 표를 보면 라벨링이 잘못된 라벨 의미를 유추하는 과정에서 여전히 문제가 있었고 특히 라벨2와 3번끼리 서로 의미가 여전히 잘 구분되지 않아 올바른 데이터를 생성하지 못한 것으로 보인다. 그러나 베이스라인보다는 성능이 조금 올랐기 때문에 완전히 실패했다고 보기는 어렵다. 그러므로 앞으로 현업에서는 데이터를 증강할 때 또는 증강한 후에 라벨링을 개선하는 작업에 있어서 어떤 방법을 사용하면 좋을지 추후 공부를 통해 개선해나갈 수 있을 것으로 보인다.

	baseline	22	23 (데이터 증강)
accuracy	0.6239	0.7613(+0.1374)	0.6842(+0.0603)

	baseline	22	23 (데이터 증강)
F1	0.5955	0.7591(+0.1636)	0.6883(+0.0928)

4.2.4 이연구

1. 데이터 증강(by Back Translation)

데이터 증강을 위해 Back Translation (BT) 기법을 사용하였다.

- **다양한 언어쌍의 선택 이유:** 한국어로만 된 데이터를 영어, 일본어, 중국어, 프랑스어 등 여러 언어로 번역한 후 다시 한국어로 역번역함으로써 번역 과정에서 나타날 수 있는 어휘적 변화나 문체의 변화를 유도하고자 했다. 이는 모델이 원본 데이터의 표현에만 의존하지 않고 다양한 표현을 학습하게 하려는 목적이 있다.
- **기대 효과:** 각 언어쌍의 특성에 따라 동일한 문장에 대해 미묘하게 다른 문체와 단어 선택이 반영된 데이터를 생성할 수 있다. 예를 들어, 한국어 → 일본어 → 한국어는 일본어의 간결한 표현이 반영될 수 있고, 한국어 → 프랑스어 → 한국어는 다소 복잡한 문체의 문장이 생성될 가능성이 있다.
 - 데이터 예시 결과
 - 원본: 해외 로밍과 금폭탄中등 차단 더 빨라진다
 - 영어: 해외 로밍 및 금 폭탄이 더 빨라집니다
 - 일본어: 해외 로밍 및 금 폭탄이 더 빠릅니다
 - 프랑스어: 해외 폭탄과 금이 더 빠릅니다
- **증강 데이터의 품질 분석:** 언어쌍별로 생성된 문장을 비교해 원본 문장과 일치도나 의미적 일관성을 분석했다. 또한 각 언어쌍별로 데이터 증강이 모델의 성능에 미친 영향을 실험하여, 특정 언어쌍이 더 효과적인지 또는 전체적으로 고르게 긍정적인 효과를 나타내는지를 파악했다.
- **실험 결과 요약:** 각 언어쌍을 통해 생성된 데이터가 모델의 일반화 성능을 높이는 데 얼마나 기여했는지를 요약하고, 성능 향상이 두드러진 언어쌍이 있다면 그 이유를 분석했다.

데이터 노이즈 제거 gemma2 를 이용하여 prompt를 사용하였다.

- **프롬프트 예시:** 예를 들어, "이 문장이 한국어에서 자연스럽게 읽히도록 수정해 주세요" 또는 "의미를 유지하면서 간결하고 일관된 표현으로 정제해 주세요"와 같은 프롬프트를 사용하여 gemma2가 텍스트를 개선하도록 유도했다.
- **프롬프트 적용 과정:** 각 데이터 샘플에 대해 gemma2의 프롬프트를 적용하여, 특정 패턴이나 의미 왜곡을 가진 문장을 필터링하거나 재구성했다. 이 과정을 통해 역번역된 문장이 원문과 의미적으로 일치하고 자연스럽게 읽히도록 했다.
- **프롬프트 사용의 효과:** gemma2의 프롬프트를 통해 노이즈를 줄임으로써, 데이터 증강 과정에서 발생할 수 있는 불필요한 변형을 최소화하고, 모델이 일관성 있고 의미 있는 패턴을 학습할 수 있도록 도움을 주었다.
- **결과:** 프롬프트로 작업하였지만 원하는 결과를 얻지 못하였다.

4.2.5 이재협

1. 텍스트 노이즈 복원 (* Appendix 08번 데이터)

적당한 수준의 노이즈가 포함된 텍스트를 LLM(`Gemma2:27b` 모델)을 활용하여 복원하였다. 적당한 수준의 노이즈는 그럴 듯하게 복원이 되지만, 사람도 원 문장이 뭔지 알 수 없을 만큼 너무 노이즈가 많으면 LLM의 복원 퀄리티가 떨어질 것을 우려하였기 때문에 적당한 수준의 데이터만 필터링하였다.

노이즈 수준을 판단하기 위해서 우선 아스키 코드 문자를 숫자/알파벳과 특수 문자로 구분하였다. 이 중 숫자/알파벳과 공백,

`.`은 정상적인 텍스트에도 들어 있을 가능성이 그 외 특수문자보다 높다고 생각해서 문장 내 숫자/알파벳 비율과 공백, `.`을 제외한 특수 문자 비율을 따로 계산한 뒤 두 값을 합친 것을 `noise_ratio`로 사용하였다. 이후 `noise_ratio`가 0.22 ~ 0.4이면서 아스키 특수 문자가 2개 이상 포함된 데이터에 대해서만 노이즈 복원을 시도하였다.

- 복원 예시

`pI美대선I앞두고 R2fr단 발] $비해 감시 강화` → `미대선 앞두고 러시아 정보 단 발표 비해 감시 강화`

`m 김정) 자주통일 새, ?r열1나가야1보` → `김정은, 자주통일 새 국민 정부 설립 나가야`

	baseline	08
accuracy	0.6239	0.504 (-0.1199)
F1-score	0.5955	0.4734 (-0.1221)
데이터 수	2800	482

점수가 많이 떨어졌으나, 이는 확보한 데이터 수가 482개로 매우 적은 것이 원인으로, 다른 데이터들과 병합하여 데이터 수를 늘리면 해결될 수 있다고 판단하였다.

2. 데이터 증강 (* Appendix 09번 데이터)

노이즈가 복원된 데이터는 대부분 라벨이 제대로 붙어 있는 데이터이므로, 이 데이터를 기반으로 LLM을 활용하여 데이터를 증강하였다. 라벨 별로 문장 샘플을 무작위로 20개 뽑은 뒤 이를 프롬프트에 `fewshot example`로 제공하여 유사한 카테고리의 뉴스 기사 제목을 생성하도록 하였다. 다양한 예시를 제공하기 위해 LLM 요청 1회당 라벨 별로 20개씩 문장을 생성하고, 생성된 문장들을 포함해 새로 예시 샘플을 뽑아서 다시 요청하는 과정을 5회 반복하였다.(총 라벨 별 100개씩 문장 생성 요청. 단, 파싱 오류 등 문장이 제대로 생성되지 않는 경우가 발생하여 총 645개 데이터가 생성되었음)

	baseline	09
accuracy	0.6239	0.6294 (+0.0055)
F1-score	0.5955	0.619 (+0.0235)
데이터 수	2800	645

이 데이터셋 역시 데이터 수가 645개로 적지만, 문장 자체가 `08_cleaned_filtered_train`보다 더 정제된 형태로 되어 있기 때문에 점수는 좀 더 높게 나왔다.

3. 라벨 복구 (* Appendix 10, 11번 데이터)

텍스트에 노이즈가 없는 데이터에는 라벨이 잘못된 데이터가 다수 섞여 있는데, 이를 복구하기 위한 방법으로 Cleanlab을 활용하였다. 우선 1, 2 과정을 통해 얻은 정상 데이터와 다른 팀원들이 얻어낸 정상 데이터를 합쳐 학습 데이터셋을 마련하였고(3, 4, 8, 9, 15번 데이터를 합쳐 사용), 이 데이터셋으로 베이스라인 모델(`bert` 모델)을 학습시킨 뒤 라벨 오류가 포함된 데이터에 대해 라벨 별 예측 확률 데이터를 산출하였다. 이후 가장 높은 확률이 `confidence_threshold` 값 이상인 데이터에 대해서만 Cleanlab을 활용해 라벨 오류를 탐지하고, 적절한 라벨로 수정해주었다.

```

error_data_logits = trainer.predict(error_data_encodings).predictions
error_data_probs = torch.softmax(torch.tensor(error_data_logits), dim=
1).numpy()

...

label_errors = find_label_issues(
    labels=high_confidence_labels,
    pred_probs=high_confidence_preds,
)

```

이후 복구된 데이터에 2번의 증강 방식을 적용하여 데이터를 증강하였다. (파일명: `11_aug_10`, 데이터 수: 652)

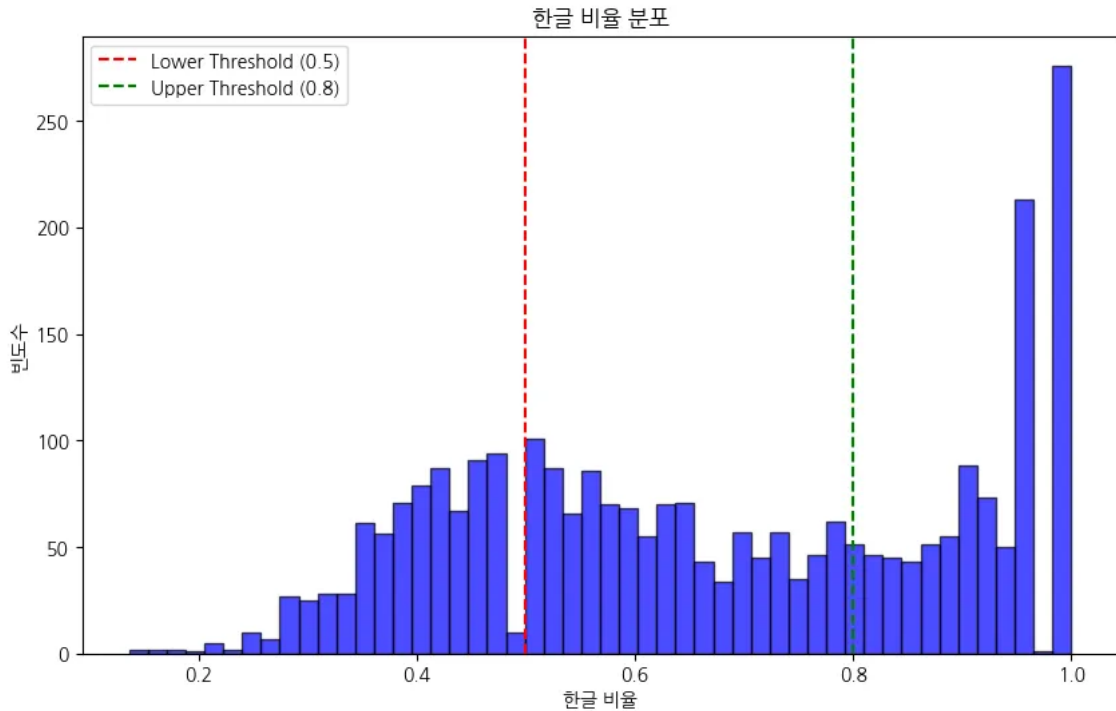
	baseline	10	11	08, 09, 10, 11 병합
accuracy	0.6239	0.7906 (+0.1667)	0.6785 (+0.0546)	0.831 (+0.2071)
F1-score	0.5955	0.778 (+0.1825)	0.6661 (+0.0706)	0.8254 (+0.2299)
데이터 수	2800	989	652	2768

1, 2, 3 과정을 통해 확보한 4개의 데이터셋 중에서 데이터 수가 비교적 많은 10번 데이터에서 성능이 가장 높게 나왔다. 또한 08, 09, 10, 11 데이터를 모두 합쳐 학습 데이터셋으로 사용한 경우 데이터 수가 2768개로 베이스라인 모델(원본 train 데이터셋 2800개)과 비슷해지는데, 성능이 훨씬 향상된 것을 볼 수 있다. 이를 통해 데이터 수가 어느 정도 확보되면, 텍스트 노이즈, 라벨 오류를 잘 제거한 데이터셋에서 성능이 더 잘 나온다는 것을 확인할 수 있다.

4.2.6 임상업

프로젝트 진행 계획

1. 대회 설명에서 봤을 때 노이즈가 아스키 코드(한글이 포함X)로 이루어져있다고 해서, 전체 문자 길이에서 한글의 비율로 노이즈의 적용 여부를 알 수 있지 않을까해서 시도했다.
2. 첫 번째 시도 결과 "... "이 비율의 많은 부분을 차지해서 "... "을 "... " 바꿔주었다.



3. 분류 결과: 너무 많은 노이즈 | 적당한 노이즈 | 정상 데이터 or 라벨 에러 데이터

- 너무 많은 노이즈는 복원이 힘들기 때문에 **제외**하기로 결정했다.
- 적당한 노이즈 복원이 가능 & 라벨 에러가 없기 때문에 노이즈를 제거하여 사용하기로 결정했다.
- 정상 데이터 or 라벨 에러 데이터의 경우 라벨 에러를 고쳐서 사용하기로 결정했다.

진행 순서

- 라벨 에러가 없는 적당한 노이즈의 데이터에 노이즈를 제거하여 label에 대한 정보를 추출한다.
- label에 대한 정보를 추출한 후 이를 바탕으로 라벨 에러 복구 또는 증강한다.

1. 텍스트 노이즈 복구

```
prompt_reasoning = ChatPromptTemplate.from_template(
    """
```

Prompt:

아래에 번호와 줄바꿈으로 나뉜 {batch_size}개의 텍스트는 임의의 character 중 20%~80%를 한글을 제외한 다른 아스키 코드로 대체한 텍스트가 끼어있습니다. 한글을 제외한 다른 아스키 코드로 대체한 텍스트는 다음과 같습니다.

예시:

- pI美대선I앞두고 R2fr단 발] \$비해 감시 강화
- oi 매력 R모h츠a열#w3약 >1·주가 고Q/진

아닌 예시:

- 美성인 6명 중 1명꼴 배우자·연인 빛 떠안은 적 있다
- 차대통령 얼마나 많이 놀라셨어요...경주 지진현장 방문종합

출력 형식:

1. 텍스트1, 아스키 코드로 대체했다고 생각하는 근거 또는 없다고 생각하는 근거
2. 텍스트2, 아스키 코드로 대체했다고 생각하는 근거 또는 없다고 생각하는 근거
- ...

아래 텍스트를 확인하고, 각 텍스트에 대한 아스키 코드로 대체했는지에 대한 근거를 논리적으로 밝혀주세요. (주어진 텍스트 갯수={batch_size}):

```
{input}
"""
)
```

```
prompt_label = ChatPromptTemplate.from_template(
"""
{response_reasoning}
```

내용들의 근거들을 바탕으로 번호에 따른 텍스트에 대한 아스키 코드 대체 여부를 1 또는 0으로 분류해 주세요.

아스키 코드로 대체되었다면 1, 대체되지 않았다면 0입니다.

출력 형식을 엄격히 지켜주세요.

출력 형식:

1. 텍스트1: 1
 2. 텍스트2: 0
 - ...
- ```
"""
)
```

두 단계의 프롬프트를 작성해서 텍스트에 노이즈가 있는지 없는지를 검증했다.

- 노이즈가 있다고 판단한 데이터의 갯수: 1666
- 대회 측에서 제공한 노이즈가 있는 데이터의 갯수: 1600

gemma2 오픈 소스 모델로 꽤나 좋은 성능으로 분류해낸 것을 확인했다.

## 2. 라벨 에러 복구

clustering으로 하는 방식과 llm을 사용해서 라벨 에러를 고치는 작업을 진행했다.

<https://www.youtube.com/watch?v=3m0wHz-PYVU>

## 3. Clustering Visualization with PCA

라벨 에러 복구 작업 전에 클러스터링 결과를 시각화하여 레이블 오류를 검토하고 데이터의 패턴을 파악하기 위해 **PCA**를 활용했다.

- PCA는 고차원의 데이터를 저차원 공간으로 투영하여 시각적 분석을 쉽게 할 수 있는 방법이다.

#### 4. Outlier Detection Using Silhouette Score

클러스터의 데이터 간의 응집도와 클러스터 간의 분리도를 평가할 수 있는 실루엣 점수로 아웃라이어를 감지해서 아웃라이어를 새로운 클러스터에 할당하였다.

결과는 기존에 사용했던 데이터의 성능보다 떨어졌다. 생각되는 이유는 클러스터링을 할 때 잘못된 라벨의 데이터가 있어서 클러스터링이 잘못되었을 경우와, 아웃라이어를 새로운 클러스터에 할당하면서 원래 클러스터에 잘 속해있던 데이터도 새로운 클러스터에 포함되었을 것으로 판단된다.

추가로 클러스터링을 하면서 3번에 해당하는 사회 주제가 매우 넓은 범위에 위치하고 있어서 모델이 사회 주제를 잘못추지 못하는 것을 발견했다.

#### 5. 데이터 증강

이상치 탐지에서 발견한 결과를 데이터 증강에 확인하기 위해 사회 주제의 데이터를 증강하였다.


증강의 과정은 LLM을 활용해서 얻어낸 라벨에 대한 정보와 주제에 해당하는 텍스트의 하위 카테고리 태그를 추출해서 이를 다시 LLM에게 제공해 뉴스 기사 제목을 생성해내는 작업을 수행하였다.


사회 주제 데이터를 1000개 증강하여 학습에 사용한 결과 0.8519 → 0.8446로 성능이 저하하였다.

비교를 위해 경제 주제 데이터를 1000개 증강하여 학습에 사용한 결과 성능이 그대로 유지 되었는데, 이것은 생성된 사회 주제의 텍스트의 품질이나 다양성이 원본 데이터와 일치하지 않아 모델 학습에 부정적인 영향을 미쳤기 때문이라고 생각된다.

## 5 최종 제출 결과

- 2주간의 프로젝트 기간 동안 다양한 실험과 분석을 거쳐, 최종적으로 public 순위 4위, private 순위 4위를 기록하며 프로젝트를 성공적으로 마무리했다.

| Rank      | Team Name | Team Member                                                                         | accuracy | f1     | Entries | Final |
|-----------|-----------|-------------------------------------------------------------------------------------|----------|--------|---------|-------|
| My Rank 4 | NLP_11조   |  | 0.8562   | 0.8516 | 33      | 1h    |

| Rank      | Team Name | Team Member                                                                         | accuracy | f1     | Entries | Final |
|-----------|-----------|-------------------------------------------------------------------------------------|----------|--------|---------|-------|
| My Rank 4 | NLP_11조   |  | 0.8562   | 0.8522 | 33      | 1h    |

- 최종 제출에 사용한 데이터 목록  
최종 제출에는 데이터의 크기 대비 성능 지표에서 상위 10개 이내에 포함되어 있는 데이터들을 활용하였다.  
최종 제출 데이터 ID: 3, 5, 10, 11, 15, 16, 22 (Appendix 참고)

## 6 자체 평가 의견

### 1. 잘한 점

- 깃허브 협업 규칙을 정하고 그 규칙을 지키며 프로젝트를 진행했다.
- 작업큐를 포함한 여러 유틸리티 도구를 제작하여 실험을 효율적으로 이루어지게 했다.

## 2. 아쉬운 점 & 개선점

- 전 프로젝트에 비해 다양한 아이디어를 떠올리고 프로젝트에 적용하는 부분이 부족했다.
- 프로젝트를 각자 진행하고 공유하는 방식으로 수행했기 때문에 동일한 코드 구현이 여러번 이루어졌다.
- 대회의 규칙이 수시로 변경되어 숙지하는데 어려움을 겪었다.

# Appendix

## 데이터 목록

| 데이터 ID | 데이터 설명                                                  | 데이터 크기 |
|--------|---------------------------------------------------------|--------|
| 01     | 노이즈 텍스트 데이터 증강(by LLM)                                  | 1507   |
| 02     | 1번 데이터 증강(by Back Translation)                          | 1943   |
| 03     | 노이즈 텍스트 데이터 클리닝(by T5)                                  | 1665   |
| 04     | 3번 데이터 증강(by Back Translation)                          | 1665   |
| 05     | 라벨 오류 데이터 보정(by clustering)                             | 1135   |
| 06     | 5번 데이터 증강(by Back Translation)                          | 1135   |
| 07     | 전체 데이터 클리닝(by LLM)                                      | 2800   |
| 08     | 노이즈 비율 0.22 ~ 0.4인 데이터 클리닝(by LLM)                      | 482    |
| 09     | 8번 데이터 증강(by LLM)                                       | 645    |
| 10     | (3, 4, 8, 9, 15) 노이즈 비율 0.2 미만인 데이터 레이블 복구(by cleanlab) | 989    |
| 11     | 10번 데이터 증강(by LLM)                                      | 652    |
| 12     | 3번 데이터 증강(by 일본어 Back Translation)                      | 1665   |
| 13     | 3번 데이터 증강(by 중국어 Back Translation)                      | 1665   |
| 14     | 3번 데이터 증강(by 프랑스어 Back Translation)                     | 1665   |
| 15     | label error가 없는 데이터 노이즈 제거(by LLM)                      | 1103   |
| 16     | 1번 데이터 라벨 복구(by cleanlab)                               | 828    |
| 17     | 2번 데이터 라벨 복구(by cleanlab)                               | 678    |
| 18     | 1번 데이터 라벨 복구(by clustering)                             | 1507   |
| 19     | 1번, 2번 데이터 라벨 복구(by clustering)                         | 3450   |
| 20     | 7번 데이터 증강(by LLM)                                       | 3867   |
| 21     | 20번 데이터 라벨 복구(by LLM, 증강 데이터 only)                      | 3275   |
| 22     | 7번 text 노이즈 복구 데이터 only(라벨 맞는 데이터)                      | 1665   |
| 23     | 22번 데이터 증강(by LLM)                                      | 1640   |