



# 실험 보고서

## 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 사전 학습 모델 선택
- 5 학습 설계 및 실험 결과 분석
  - 5.1 Data processing
    - 5.1.1 데이터 전처리
    - 5.1.2 데이터 증강
  - 5.2 Modeling
- 6 최종 제출 결과
- 7 자체 평가 의견

## 1 프로젝트 개요

### 1. 과제 소개

본 대회는 '생성형 AI(LLM)와 인간 : 텍스트 판별 챌린지'라는 주제로 진행되며 문단 단위(Paragraph)의 글(Text)이 사람(Human)이 작성한 것(0)인지, 생성 AI가 작성한 것(1)인지 판별하여, 각 문단(샘플)이 AI가 작성했을 확률을 예측하는 AI 모델을 개발하는 태스크를 부여한다. 본 문제에 사용되는 데이터는 대부분 한국어(Korean)로 구성된 텍스트 데이터이며 학습 데이터는 전체 글(Full Text)에 대해, 일부 문단이나 문장만 AI가 작성된 경우에도 글 전체에 'AI 작성' 라벨(1)이 부여되며, 문단 단위 라벨은 제공되지 않는다.

평가 데이터는 1개 문단 단위로 샘플이 구성되어 있으며, 샘플이 AI가 작성했을 확률(0~1 사이)을 예측해야 한다. 평가 데이터에는 `title` 과 `paragraph_index` 정보가 포함되어 있으며, 같은 `title` 을 가진 문단들은 하나의 글에 속하는 문단들이므로, 이를 바탕으로 동일 글 내의 다른 문단 정보를 추론에 활용하는 것이 허용된다. 이는 일반적인 평가 환경에서의 데이터 누수(Data Leakage)와는 다른 구조로, 하나의 글 내 문단 간 상호 참조는 허용되며, 서로 다른 글 간에는 정보가 공유되지 않도록 해야 한다.

### 2. 평가 지표

평가 지표는 ROC-AUC이다. 여기서 TPR (True Positive Rate)은 참 양성 비율, 즉 실제 양성 중에서 모델이 양성이라고 맞게 예측한 비율, FPR (False Positive Rate)은 위양성률, 즉 실제 음성인데 모델이 양성으로 잘못 예측한 비율이다.

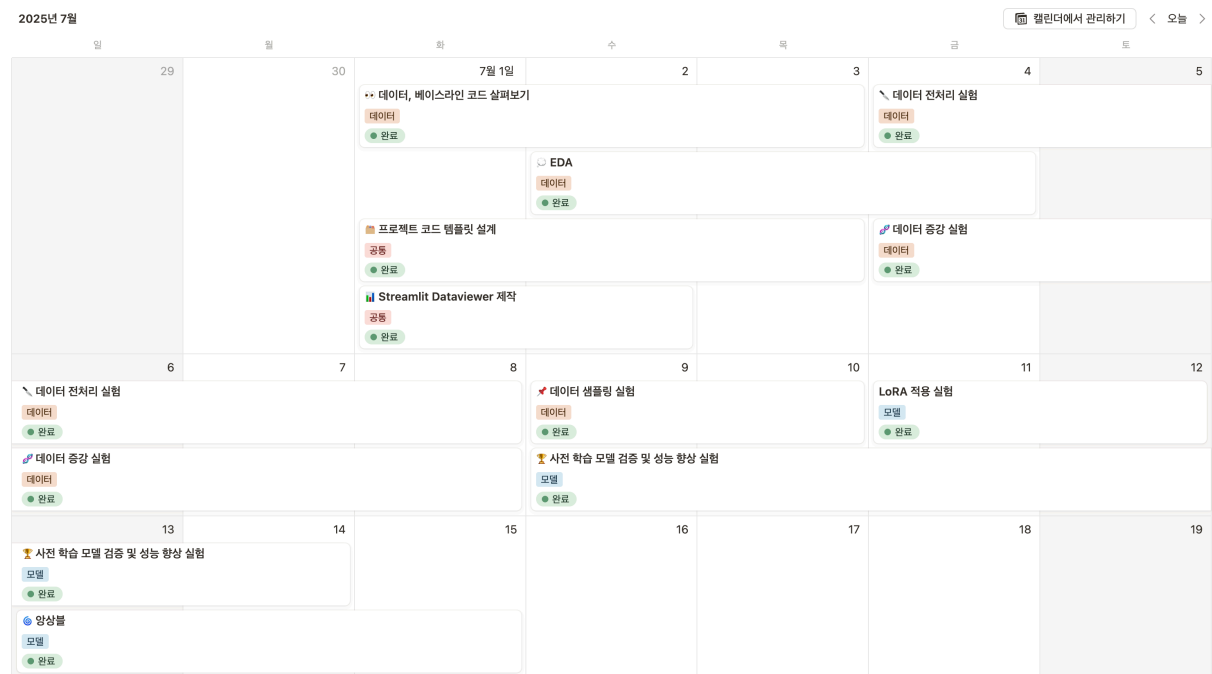
$$AUC = \int_0^1 TPR(FPR) dFPR$$

## 2 프로젝트 팀 구성 및 역할

이름	역할
김민선	EDA, 사전 학습 모델 선택 실험, 데이터 증강 - 라벨 1 증강 비율 실험, 데이터 샘플링 - 전체 데이터 비율 유지 후 데이터 양 줄이기, LoRA 적용 파이프라인 구축, 앙상블 실험, 프로젝트 템플릿 구축, Streamlit Data Viewer 제작
최건웅	EDA, 사전 학습 모델 선택 실험, 데이터 전처리 - 연속 공백 / 유니코드 제어문자 제거, 데이터 증강 - 라벨 1 학습 모델로 증강 데이터 생성, 형태소 분석 실험, 허깅페이스 모델 성능 개선 실험
황솔희	EDA, 사전 학습 모델 선택 실험, 데이터 전처리 - 극단적으로 긴 데이터 제거, 데이터 증강 - Back Translation, 허깅페이스 모델 성능 개선 실험

## 3 프로젝트 수행 절차 및 방법

프로젝트는 EDA, 데이터 전처리, 데이터 증강, 데이터 샘플링, 사전학습 모델 검증 및 성능 향상 실험, LoRA 적용 실험, 앙상블 순서로 진행되었다.



## 4 프로젝트 수행 결과

## 4.1 데이터 분석

### 1. 클래스 비율

전체 학습 데이터는 약 97,000개이며 라벨 0(Human)이 89,177개, 라벨 1(AI)이 7,995개로 약 11:1 수준의 심각한 클래스 불균형을 보인다.

이로 인해 모델이 사람 글(0)에 과도하게 편향되어 학습될 수 있으며 극단적인 경우 “항상 사람”이라고만 예측해도 약 92%의 정확도를 기록할 수 있다.

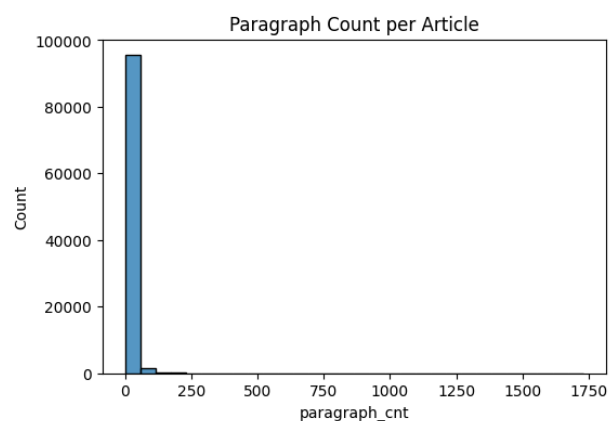
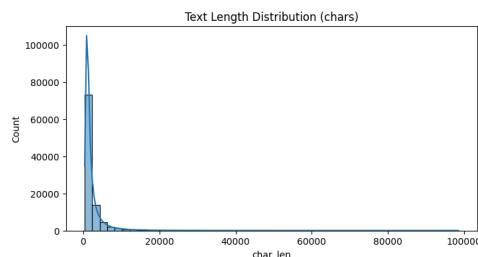
이러한 분포는 학습에 부정적인 영향을 미칠 수 있으므로 라벨 간 데이터 균형을 맞추는 사전 작업이 필요하다는 인사이트를 얻었다.

```
라벨 분포
generated
0      89177
1       7995
Name: count, dtype: int64
```

### 2. 텍스트 길이 & 문단 수 분석

텍스트 길이와 문단 수 분포를 확인한 결과, 대부분의 데이터는 텍스트 길이 20,000자 이하, 문단 수 50개 이하로 확인되었다. 전체 분포는 long-tail 형태를 띠며 극단적으로 긴 데이터가 일부 존재했는데 이는 학습 과정에서 잡음으로 작용할 가능성이 있다.

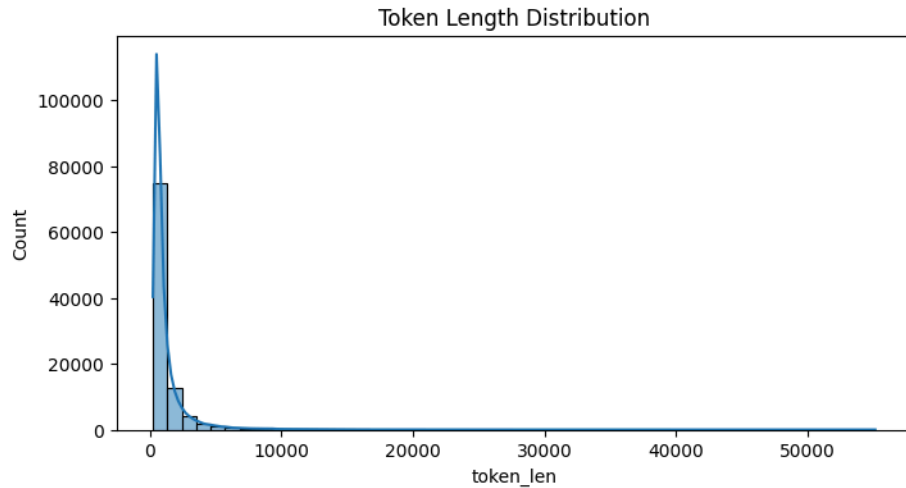
따라서 전처리 단계에서 텍스트가 과도하게 긴 샘플은 제거하거나 별도로 처리할 필요가 있다고 판단했다.



### 3. 토큰 길이 분석

`klue/bert-base` 토크나이저 기준으로 토큰 길이를 분석한 결과, 512 토큰을 초과하는 샘플이 다수 존재했으며, 최장 길이는 55,143 토큰까지 나타났다.

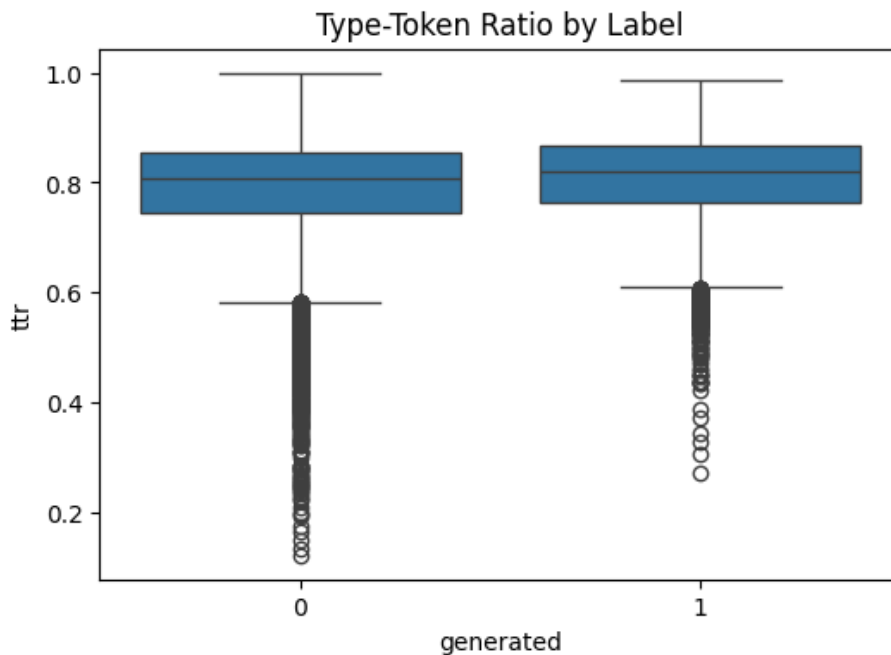
이로 인해 학습 시에는 토큰 분할이 필수적이며 sliding window, truncation, stride 등 적절한 전처리 전략이 필요하다.



#### 4. 어휘 다양성 (TTR)

“LLM이 작성한 글은 어휘가 반복될 것이다”라는 가설을 세우고 TTR(Type-Token Ratio)을 기준으로 분석을 진행했다. 하지만 실제 결과는 예상과 달랐다. AI 문장(1)의 TTR 중앙값이 사람 문장(0)보다 오히려 약간 높게 나타났다.

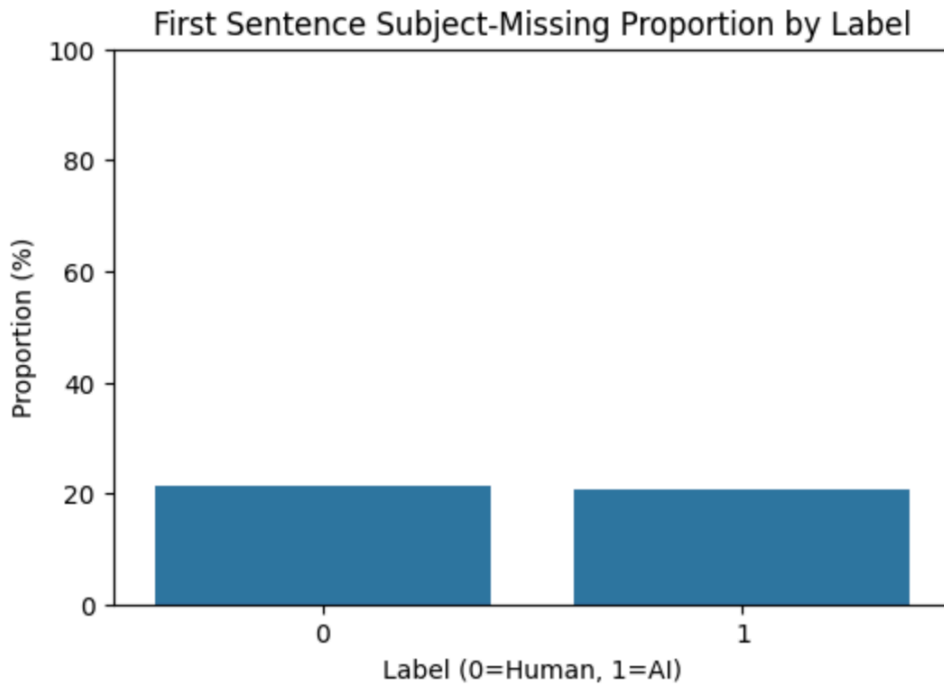
이는 최근 LLM의 표현력이 인간 수준에 도달하거나 일부 영역에서는 초과할 수도 있음을 시사하는 흥미로운 결과였다.



#### 5. 첫 문장 주어 유무

데이터 탐색 중 AI가 쓴 글에서 주어가 생략된 문장이 처음에 오는 경우가 많다는 인상을 받아 이를 정량적으로 분석해보았다. 하지만 실제로 분석한 결과, 사람 글과 AI 글 모두 첫 문장에 주어가 없는 비율이 비슷해 의미 있는 차이는 발견되지 않았다.

직관과는 달랐던 분석 결과였지만 직접 가설을 세우고 확인했다는 점에서 의미 있는 시도였다.

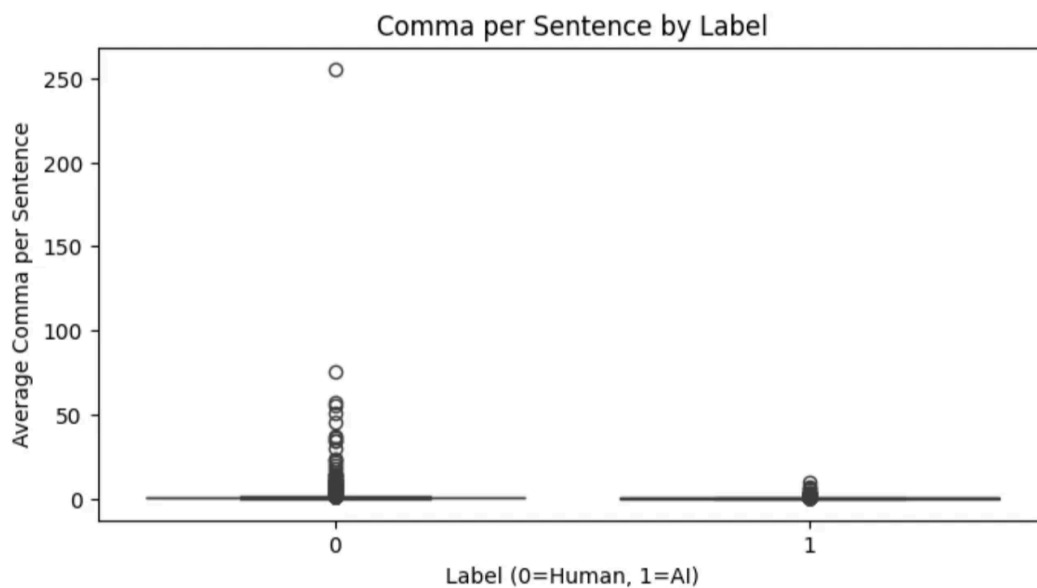


## 6. 쉼표 개수

LLM이 생성한 문장에서는 쉼표 사용이 많을 것이라는 가설을 바탕으로 문장당 쉼표 평균 개수를 라벨별로 분석했다.

사람 문장(0)은 평균 0.72개, AI 문장(1)은 평균 0.41개로 사람 문장이 쉼표를 더 많이 사용하는 것으로 나타났다. 특히 사람 문장은 이상치가 많고 분포 자체가 훨씬 넓은 경향을 보였다. 상위 25% 구간(Q3) 기준에서도 사람 문장이 쉼표를 더 자주 사용하는 등 쉼표 사용량 차이는 꽤 뚜렷했다.

기준에 알고 있던 LLM의 문체 특성과는 다소 다른 결과였으며 사람 문장이 한 문장당 더 많은 정보를 담는 구조이기 때문에 이런 차이가 발생했을 수도 있다는 생각이 들었다.



## 4.2 사전 학습 모델 선택

### 1. klue/roberta-base

**선택 이유:** KLUE 벤치마크를 기반으로 학습된 RoBERTa 계열의 한국어 특화 언어모델로, 다양한 한국어 자연어 처리 태스크에서 우수한 성능을 보여주는 대표적인 베이스라인 모델이다.

특히 한국어 문장의 구조, 어순, 조사 체계 등 고유 문법적 특성을 잘 반영하고 있어, 문장이 자연스럽게 작성되었는지 여부를 효과적으로 구분할 수 있다. 또한 RoBERTa 구조는 BERT에 비해 학습 방식이 개선되어 분류 문제에서 더 안정적인 성능을 보이며, 다양한 실험의 기준점 역할로 활용되기에 적합하다.

### 2. monologg/koelectra-base-v3-discriminator

**선택 이유:** 한국어 말뭉치를 대규모 수집 정제해 학습한 버전으로 단순 번역체나 기계 생성 문장의 뉘앙스를 더 잘 구별한다.

Base 모델임에도 학습된 Discriminator만 사용하므로 실제 추론 시 속도가 빠르며 개발 환경을 고려하여 GPU 리소스가 제한적인 환경에서도 효율적으로 운용할 수 있는 모델을 선택했다

### 3. beomi/KcELECTRA-base-v2022

**선택 이유:** KorQuAD, Namuwiki, 뉴스, 블로그 등 다양한 도메인을 아우르는 대규모 말뭉치로 학습되어, 한글 표현의 폭넓은 변주를 이해한다

최근 코퍼스와 토큰나이저 업데이트, 하이퍼파라미터 튜닝을 반영해, 이전 버전 대비 문장 이해력 및 분류 성능이 개선된 모델로 이전 버전 모델인 monologg/koelectra-base-v3-discriminator의 결과가 좋았던 것을 고려해 선택했다.

### 4. monologg/kobigbird-bert-base

**선택 이유:** 이 모델은 긴 문서를 처리할 수 있는 BigBird 아키텍처 기반의 한국어 특화 모델이다.

이번 대회에 문단은 최대 수만 토큰에 달하는 긴 텍스트로 구성되며 이를 효과적으로 인코딩하기 위해 sparse attention 구조를 활용하는 KoBigBird가 적합하다. 또한 한국어에 특화되어 있어 사람과 AI가 작성한 문장의 미묘한 차이를 정밀하게 구분하는 데 유리하다고 판단하여 선택했다.

### 5. daekeun-ml/koelectra-small-v3-nsmc

**선택 이유:** ELECTRA 구조를 기반으로 하며, 네이버 영화 리뷰 데이터셋(NSMC)을 포함한 감성 분류 태스크로 사전 학습되었다.

ELECTRA는 기존의 마스킹 기반 예측 방식과 달리 실제/가짜를 판별하는 방식으로 학습되기 때문에, 문장이 기계적으로 생성된 것인지 여부를 판단하는 데 유리한 구조를 갖는다. 특히 사람이 작성한 자연스러운 표현에 대한 노출이 많아, 인간 문체에 대한 분별력이 높을 것으로 기대된다. 또한 small 모델로 상대적으로 경량화되어 있어, 다양한 모델 간의 앙상블 구성 시 계산 효율성과 다양성 확보 측면에서도 장점이 있다.

### 6. classla/xlm-roberta-base-multilingual-text-genre-classifier

**선택 이유:** XLM-R 구조를 기반으로 하며, 다국어 텍스트의 장르(예: 뉴스, 블로그, 학술 문서 등)를 분류하는 데 특화되어 학습된 모델이다.

사람과 AI가 작성한 문장은 종종 문체나 어조, 정보 전달 방식 등의 측면에서 차이를 보이기 때문에, 이러한 문체적 특성을 구분할 수 있는 모델은 본 과제에 유의미하게 기여할 수 있다. 비록 한국어 전용 모델은 아니지만, 문

체 구분이라는 태스크 유사성을 바탕으로 일반화된 표현 감지에 유리하며, 앙상블 구성 시 문체 기반 판별 관점을 보완해줄 수 있는 요소로 활용될 수 있다.

## → 자체 성능

본 실험에서는 모델 구조 자체의 성능을 비교하기 위해 모든 베이스 모델에 대해 동일한 하이퍼파라미터(learning rate = `2e-05`, batch size = `1`)를 적용하여 fine-tuning을 진행하였다. 또한, 각 모델의 사전학습(pretrained)된 표현력만으로 어느 정도 구분이 가능한지 확인하고자 일부 모델에 대해서는 학습을 수행하지 않은 상태에서 대회 제공 데이터셋에 대해 inference를 진행하였다. 이를 통해 fine-tuning 여부와 관계없이 모델 구조 및 사전학습 표현력의 상대적인 우수성을 확인하고자 하였으며 동일 조건에서의 실험은 비교의 공정성과 해석 가능성을 높이는 데 목적이 있다.

base model	learning rate	batch size	BEST ROC-AUC
klue/roberta-base	2e-05	1	0.8062
monologg/koelectra-base-v3-discriminator	2e-05	1	0.8402
beomi/KcELECTRA-base-v2022	2e-05	1	0.8236
monologg/kobigbird-bert-base	2e-05	1	0.5
daekeun-mi/koelectra-small-v3-nsmc	2e-05	1	0.8068
classla/xlm-roberta-base-multilingual-text-genre-classifier	2e-05	1	0.5028

# 5 학습 설계 및 실험 결과 분석

## 5.1 Data processing

### 5.1.1 데이터 전처리

#### 1. 극단적으로 긴 데이터 제거

label = 0(사람 작성) 샘플에서만 상위 1%지점 이상의 극단치가 자주 관찰되었다. (문장 수 기준: 사람샘플 최대 98,549자, AI 샘플 최대 약 46,814자 )

label = 0 샘플에만 임계값 문자수 = 50,000자를 적용하여 지나치게 긴 텍스트가 모델에 미치는 영향을 최소화하고자 했다.

이 과정을 통해 최종 데이터 셋 크기는 7,172 → 96,207건(문장 수 필터) 및 97,172 → 97,128건(문자 수 필터)으로 정제되었다.

이를 통해 AI 생성 샘플 분포를 그대로 유지하며 클래스 불균형 영향을 최소화하고 모델 일반화 성능과 학습 안정성 향상을 기대한다.

#### 2. 연속 공백 / 유니코드 제어문자 제거

텍스트 데이터에는 보이지 않는 제어문자(예: `\x00`, `\u200b`)나 연속된 공백 등이 포함되어 있는 경우가 많다. 이러한 노이즈는 불필요한 토큰을 생성하거나 모델 입력 길이를 비효율적으로 늘릴 수 있어 제거가 필요하다. re 정규표현식을 활용해 제어문자를 제거하고, 연속된 공백을 하나로 줄이며, 문장의 앞뒤 공백도 제거하였다. 이 과정을 통해 텍스트 정제 품질을 높이고 모델 학습 안정성을 확보하였다.

### 5.1.2 데이터 증강

## 1. 라벨 1 증강 비율 실험

### • 기대 효과

- 극심한 클래스 불균형(약 11:1 비율)으로 인해 모델이 라벨 0(사람)에 과도하게 치우치는 경향을 보인다. 이에 따라 라벨 1(AI)을 증강하거나 샘플링 비율을 조정함으로써 모델의 판별력을 높일 수 있을 것으로 기대한다.
- 특히 이진 분류 문제의 특성상 minority class(라벨 1)에 대한 recall 향상은 전체적인 F1 score와 ROC-AUC 성능을 함께 개선할 수 있는 여지를 제공한다고 판단한다.
- 또한, 해당 실험 결과를 생성형 AI 텍스트 탐지 성능을 강화하기 위한 데이터 증강 전략 수립에 활용하기 위해 진행한다.

### • 실험 방법

- TF-IDF 벡터 기반으로 변환한 텍스트 피처를 입력으로 사용하는 XGBoost 모델을 기반으로 실험을 진행한다. (베이스라인 코드)
- 클래스 불균형을 보정하는 XGBoost의 핵심 하이퍼파라미터인 `scale_pos_weight` 를 1부터 15까지 변경하며 grid search 방식으로 성능을 비교한다.
- 각 설정에 대해 F1 score와 ROC-AUC 값을 측정하며, 이를 통해 모델의 정밀도-재현율 균형과 전반적인 판별 성능을 평가한다.

### • 실험 결과

scale_pos_weight	F1	ROC-AUC
1	0.9569	0.9112
2	0.9568	0.9122
3	0.9573	0.9104
4	0.9579	0.9074
5	0.9564	0.9072
7	0.9545	0.9043
9	0.9522	0.9045
11	0.9490	0.9035
13	0.9454	0.9040
15	0.9397	0.9005

- `scale_pos_weight = 3` 일 때 F1 score가 0.9573로 가장 높게 나타났다.
- `scale_pos_weight = 2` 일 때 ROC-AUC가 0.9122로 가장 높게 나타났다.
- `scale_pos_weight` 가 커질수록 성능이 점차 하락하는 경향을 확인하였으며, 예를 들어 `scale_pos_weight = 15`일 경우 F1은 0.9397, ROC-AUC는 0.9005로 감소하였다.

### • 해석

- 성능 최적 지점에서의 라벨 비율을 역산하면, `scale_pos_weight` 3은 라벨 1이 약 29,700개, 2는 약 44,500개 일 때의 비율에 해당한다. 따라서 현재 라벨 1 데이터 수(7,995개)를 기준으로 약 3~5.5배 증강하는 것이 성능 향상에 가장 효과적인 전략임을 확인했다.

## 2. Back Translation



- 기대 효과

- 다양성 증가 : 원문이 가지는 표현의 한계를 넘어, mBART를 통한 번역-역번역 과정에서 어휘 선택, 문장 구조, 어투 등이 자연스럽게 변형된다. 모델이 학습할 때 다양한 문체, 표현을 접하게 되어 테스트 환경에서 마주칠수 있는 변형된 입력에도 대응한다.
- 오버피팅 완화 : 한정된 원본 데이터만 학습할 경우 특정 문장 패턴에 과도하게 적응할 수 있는데 backtranslation을 통해 유사하지만 고유한 샘플을 추가함으로써 모델이 보다 일반적인 특징을 학습하도록 유도한다
- 클래스 불균형 완화 : label = 1 에 대해 추가 샘플을 확보함으로써 샘플 수 차이를 줄여 분류 성능을 개선한다

- 증강 방법

- label = 1인 샘플만 선별
- 한글 → 영어

```
en_text = mbart_translate(text, "ko_KR", "en_XX")`
```

- 영어 → 한글

```
ko_back = mbart_translate(en_text, "en_XX", "ko_KR")
```

- mBART 모델(facebook/mbart-large-50-many-to-many-mmt)을 활용해, 강제 BOS 토큰 (forced\_bos\_token\_id)으로 목적 언어 지정했다

- 증강 결과

라벨 0	라벨 1
89177	10989

- 실험 결과

- 베이스라인 모델인 XGBoost 결과

backtranslation 개수	ROC-AUC
0	0.6309
1000	0.5916
5000	0.5583
all	0.5552

- label =1 데이터의 backtranslation 개수에 따른 성능 측정 결과 backtranslation을 하지 않을 때가 0.6309로 성능이 가장 좋음을 확인할 수 있다.

- 해석

테스트셋에는 backtranslation 텍스트가 없으므로 학습 분포가 실제 분포 간 차이가 커져 일반화 성능이 저하된 것으로 보인다.

### 3. 라벨 1 학습 모델로 증강 데이터 생성

- 기대 효과

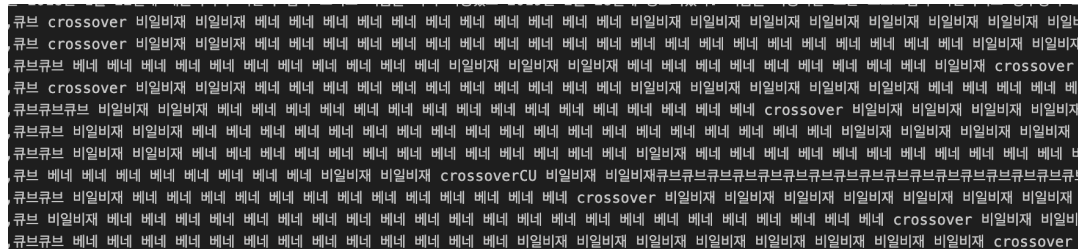
- 기존의 label=1(AI 생성 텍스트)은 데이터 수가 적고, 문장 구성에도 제한이 존재하기 때문에 모델이 충분히 다양한 패턴을 학습하기 어렵다. 이를 해결하기 위해, T5 모델을 활용하여 label=1 데이터를 기반으로 한 새로운 문장을 생성하면, 기존 AI 텍스트의 특성을 보존하면서도 보다 다양한 어휘, 구조, 길이를 포함하는 문장이 추가될 수 있다.
- 클래스 불균형 문제를 완화하고, label=1에 대한 일반화 성능을 높이며, 모델이 실제 AI 생성 문장의 특성을 보다 풍부하게 학습할 수 있도록 유도하는 것이 목적이다.

#### • 증강 방법

- label=1로 라벨링된 원본 데이터셋을 기반으로 사전학습된 T5 모델을 파인튜닝하였다. 학습 데이터는 full\_text와 label=1만을 필터링하여 입력값으로 활용하였고, 출력은 동일한 스타일의 새로운 문장을 생성하는 방식으로 설계했다.
- 학습 후 생성된 텍스트는 label=1로 간주하고 원본 데이터와 합쳐서 모델 학습에 사용하고자 했다.

#### • 증강 결과

- 증강 문장이 생성되었지만, 생성 결과를 확인해본 결과 아래 이미지와 같이 "베네", "큐브", "비일비재" 등 동일한 단어가 과도하게 반복되며, 실질적인 데이터 다양성과 품질이 매우 낮게 나타났다.



- 이러한 문제로 인해 증강 데이터는 실제 학습에 사용되지 않았다.
- 증강 실패에 따른 대안 검토 및 판단
  - T5 기반 텍스트 생성 모델을 이용해 label=1(AI 생성 텍스트) 데이터를 증강하고자 하였으나, 생성된 결과에서 의미 없는 단어 반복과 단조로운 어휘 사용 등 품질 저하가 심각하게 나타났다. 해당 증강 결과는 학습에 투입될 경우 오히려 모델 성능 저하를 유발할 수 있으며, 실제 테스트셋과의 분포 괴리가 심화되어 일반화 성능이 저하될 가능성이 높았다. 특히, 평가셋에는 label=1의 샘플 수 자체가 적고 label=0이 압도적으로 많은 경우가 많다. 이처럼 평가셋의 레이블 분포를 명확히 알 수 없는 상황에서, label=1 데이터를 과도하게 증강할 경우 모델이 label=1을 과대표현하게 학습하고, 평가 시에는 오히려 오탐률 이 증가하는 문제로 이어질 수 있다. 따라서 생성된 label=1 증강 데이터를 실제 학습에는 사용하지 않기로 결정했다.

## 5.2 Modeling

### 1. 형태소 분석

#### • 기대 효과

- LLM 기반 모델을 사용하기 이전, 제한된 환경에서도 성능 향상을 꾀하고자 형태소 분석 기반의 전처리 방식을 적용했다. 한국어는 조사, 어미 등 다양한 어형 변화가 존재하여 어절 단위 벡터화만으로는 모델이 적절한 표현을 포착하기 어렵다. 형태소 단위로 분석하면 어휘 표현의 다양성을 줄이고 핵심 의미를 보존할 수 있어, TF-IDF 벡터 밀도가 높아지고 학습 안정성 및 일반화 성능 향상이 기대된다.

#### • 실험 방식

- Okt 형태소 분석기를 사용해 full\_text 컬럼의 문장을 형태소 단위로 변환한 뒤, TfidfVectorizer를 통해 벡터화하였다. 여기에 문장 길이, 구두점 수, 숫자·대문자 비율 등 텍스트 통계 기반 피처를 추가하여 입력 데이터를 구성하였다. 학습은 XGBoost 분류기로 진행하였으며, AI:사람 비율은 1:2로 샘플링하여 밸런스를 맞췄다. 평가 지표는 ROC AUC를 기준으로 하였다.

#### • 실험 결과

모델명	TFIDF-XGBoost
Accuracy	0.61812719

#### • 해석

- 형태소 분석 기반의 전처리를 적용하여 단어 단위 표현력을 높이고자 하였으나, 최종 ROC AUC는 **0.6181**로, **기존 베이스라인 모델(0.6309)** 대비 오히려 낮은 결과를 보였다. 이는 형태소 분석이 오히려 불필요한 단어 분절이나 맥락 정보 손실을 유발했을 가능성이 있으며, TF-IDF 기반의 벡터화만으로는 문장의 구조나 의미 흐름을 충분히 반영하지 못했음을 시사한다. 따라서 해당 실험은 제한된 상황에서 시도해볼 수 있는 보완 전략이긴 하나, 실질적인 성능 개선에는 크게 기여하지 못한 것으로 판단된다.

## 2. 허깅페이스 모델 성능 개선 실험

#### • 기대 효과

- AI 생성 텍스트가 소수인 환경에서 적정량 정도의 증강을 통해 사람 vs 생성형 AI 비율을 맞추면 분류 모델의 경계가 안정적으로 학습된다
- 더 많은 epoch 동안 학습하면 모델 손실 곡선을 더 낮은 지점까지 내릴 기회가 늘어난다

#### • 실험 방식

1. 증강 데이터 입력
2. epoch 증가

#### • 실험 결과

모델명	데이터 증강 방법	epoch	ROC-AUC
KcELECTRA-base-v2022	none	5	0.8236
KcELECTRA-base-v2022	backtranslation	5	0.7944
koelectra-base-v3-discriminator	none	5	0.8402
koelectra-base-v3-discriminator	none	30	0.8136
klue/roberta-base	none	5	0.8062
koelectra-small-v3-nsmc	none	5	0.8068
xlm-roberta-genre-classifier	none	5	0.5028

#### • 해석

- backtranslation 데이터가 정보 중복, 노이즈를 과도하게 늘려 모델이 진짜 AI 생성 텍스트를 구분하는 신호를 희석시킨 것으로 보인다
- 학습을 많이 진행할수록 과적합이 발생해 테스트 셋에 대한 분류 능력이 저하되었다.

## 3. LoRA + 데이터 샘플링 적용

#### • 기대 효과

- 파라미터 효율적 학습 기법인 LoRA와 데이터 샘플링을 동시에 적용함으로써 기존 모델의 성능을 유지하면서도 메모리와 계산 자원을 절약하고 제한된 환경에서도 학습이 가능해져 실험 접근성이 높아질 것으로 기대한다.

#### • 실험 방식

- monologg/koelectra-base-v3-discriminator 모델을 기반으로 전체 학습 데이터의 라벨 비율 (11:1)을 유지한 채 20000개 데이터만 샘플링하여 LoRA 적용 모델로 학습을 진행한다.
- 이후 동일한 모델 구조에서 LoRA만 적용한 실험, 학습없이 inference만 수행한 결과와 비교하여 상대적인 성능을 평가한다.

#### • 실험 결과

- 데이터 샘플링(20000개)과 LoRA를 함께 적용한 경우, 성능이 가장 낮게 나타난다.
- LoRA만 적용한 경우, 샘플링+LoRA 동시 적용보다 성능이 다소 향상되었지만 학습을 전혀 하지 않은 모델의 inference 결과보다 낮은 성능을 보인다.

	inference만 진행	데이터 샘플링 + LoRA	LoRA
ROC-AUC	0.84020	0.67124	0.75178

#### • 해석

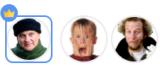
- 데이터 샘플링과 LoRA가 모두 모델 성능 저하에 기여한 것으로 판단한다.
- 특히 원래의 긴 문서를 압축하는 과정에서 중요한 정보가 손실되었을 가능성과 LoRA 적용 시 적절한 하이퍼파라미터 탐색이 부족했을 가능성이 존재한다.
- 제한된 환경에서 LoRA를 활용한 학습이 실제 성능 향상으로 이어지기 위해서는 데이터 처리 방식 개선과 정교한 파라미터 튜닝 등 추가 실험 설계가 필요함을 확인할 수 있다.

## 6 최종 제출 결과

#### • 리더보드 결과

최종 제출 결과, public score는 **0.85619**, private score는 **0.85712**를 기록했다.

Public 리더보드 기준으로는 **272팀 중 92위**, private 리더보드 기준으로는 **93위**를 기록하며 전체적으로 **상위 34% 이내의 성적**을 유지했다. Public과 private 점수 간 편차가 거의 없었던 점에서 모델이 과하게 leaderboard에 overfitting 되지 않고 일관된 일반화 성능을 보여주었음을 확인할 수 있었다.

#	팀	팀 멤버	최종 점수	제출수
93	203호		0.85712	50

#### • 앙상블에 사용한 모델과 성능(Weighted Voting)

	weights	Public Score	Private Score
klue_roberta_base , koelectra-base-v3-discriminator , daekeun-ml_koelectra-small-v3-nsmc	[0.36, 0.33, 0.31]	0.85619	0.85712

## 7 자체 평가 의견

### 1. 잘한 점

- a. 제한된 환경에서 자원을 최대한 활용할 수 있는 방법으로 실험을 설계 및 진행했다.
- b. 실험을 순서대로 진행하며 이전 실험의 결과를 근거로 하여 다음 실험을 설계하는 과정을 적용했다.

### 2. 아쉬운 점 & 개선점

- a. 리소스 제약으로 인해 모델을 단순히 실행하는 수준에 그쳤으며 본격적인 파인튜닝 및 하이퍼파라미터 튜닝을 적용하지 못한 점이 아쉬움으로 남는다.
- b. 만약 충분한 GPU 자원이 확보되었다면 기존에 학습한 내용을 기반으로 지식 증류, 샘플링 전략 고도화, 데이터 밸런싱 개선 등 보다 다양한 실험을 수행했을 것이다.