2018131425 고수현_최종결과보고서

1. Task 설명

- 온라인 커뮤니티 및 소셜 미디어의 이용자 수는 매년 증가하고 있습니다. 이와 함께, 참여자들의 욕설 및 혐오 표현 게시가 급증하고 있으며, 남녀노소 접근성이 뛰어난 소셜 미디어에서는 이러한 부적절한 표현들이 무분별하게 노출되고 있습니다. 따라서 플랫폼 자체에서 자동으로 이러한 표현들을 필터링하는 시스템의 필요성이 높아지고 있습니다.
- 본 프로젝트의 제목은 "Offensive Language Identification in Korean"으로, 단순히 욕설을 감지하는 것에 그치지 않고, 문장 내에 욕설이 없더라도 그 문장이 offensive한 의도를 가지고 있는지를 감지하고 분류하는 것까지에 목표를 두고 있음을 의미합니다. 예를 들어, "기다려라 찾아간다 ㅋㅋ 평생 누워있게 해줄게" 와 같은 문장에는 직접적인 욕설이 포함되지 않았지만, 공격적인 의도를 내포하고 있어 감지의 대상이 됩니다.
- 한국어의 언어적 특성상 필터링이 쉽지 않으며, 이러한 부적절한 표현들은 다양한 사회문화적 맥락을 고려하여 감지해야 합니다. 따라서 본 프로젝트에서는 문장의 맥락을 이해하고, 그 문장이 누군가에게 모욕적이거나 불쾌감을 줄 수 있는지를 추론할 수 있는 모델을 개발하고자 합니다. 이를 통해 건강한 온라인 커뮤니티 환경을 조성하는 데 기여할 수 있을 것으로 기대합니다.

2. 관련 연구동향

관련 논문 1: KOAS (Korean Text Offensiveness Analysis System)

- KOAS는 한국어에서 공격적 언어를 식별하는 시스템으로, 멀티-태스크 러닝을 사용하여 욕설 감지와 감정 분석을 동시에 수행합니다. 아래에서 KOAS 시스템의 핵심적인 구성 요소와 방법론에 대해 설명하겠습니다.

  시스템 개요

  - KOAS는 한국어 텍스트의 공격성을 분석하기 위해 두 가지 주요 작업을 수행합니다.
    1. **욕설 감지 (Abusive Language Detection)**: 문장에서 욕설이나 비속어를 감지하여 문장이 abusive인지 non-abusive인지를 분류합니다.
    2. **감정 분석 (Sentiment Analysis)**: 문장의 감정을 긍정(Positive), 중립(Neutral), 부정(Negative)으로 분류합니다.
  - 이 두 가지 작업을 통해 얻어진 점수를 종합하여 문장의 공격성(offensiveness)을 평가합니다. 공격성 점수는 다음과 같은 공식을 통해 계산됩니다.

$$O = \sigma(\alpha \times (y^{\text{neg}} - \max(0, y^{\text{pos}})) + \beta \times y^{\text{ab}})$$

→ 여기서 y**neg와 y**pos는 각각 부정적, 긍정적 문장 임베딩에 대한 출력 값이며, y**ab는 욕설 감지 결과이며, σ는 시그모이드 함수입니다.

  데이터 처리 및 전처리

  - KOAS는 한국어의 형태소 수준의 토큰화를 사용하여 데이터를 처리합니다. 이 과정에서 Mecab-ko와 같은 형태소 분석기를 활용하며, 데이터 증강(data augmentation)과 서브워드 정규화(subword regularization)를 통해 모델의 학습 효과를 극대화합니다.

  모델 구조

  - KOAS의 모델 구조는 두 부분으로 나뉩니다.
    1. **Shared Convolution Layer**: 감정 분석과 욕설 감지 작업을 위한 공통 레이어
    2. **Task-specific Layers**: 각 작업에 특화된 레이어. 감정 분석을 위한 레이어와 욕설 감지를 위한 레이어가 별도로 존재.
  - 입력 문장은 먼저 형태소 수준의 토큰화 과정을 거쳐 embedding layer에 전달됩니다. embedding layer의 출력은 shared convolution layer를 통과한 후, 각 작업별로 나누어져 해당 task-specific layers로 전달됩니다. 이 과정을 통해 각 작업별로 필요한 특성을 학습하고, 최종적으로 두 작업의 결과를 종합하여 공격성 점수를 산출합니다.

  시스템 설계

  - KOAS는 문장 내의 복잡한 상호작용을 효과적으로 학습하기 위해, 다양한 활성화 함수(activation functions), 드롭아웃(dropout), 맥스 풀링(max pooling) 기법을 사용합니다. 모델의 최종 출력은 소프트맥스(softmax) 함수를 통해 각 작업의 예측 확률로 변환되며, 이를 기반으로 문장의 공격성을 평가합니다.

  평가 및 결과

  - KOAS는 유튜브, 네이버 영화 리뷰, DC인사이드와 같은 다양한 출처에서 데이터를 수집하여 약 4만 5천 개의 문장을 학습 데이터로 사용했습니다. 이 데이터는 세 명의 어노테이터에 의해 욕설 포함 여부와 감정에 따라 라벨링 되었습니다.

- 이러한 구조와 방법론을 통해 KOAS는 한국어 텍스트의 공격성을 효과적으로 분석할 수 있으며, 기존의 단일 작업 기반 접근법에 비해 더 정교하고 신뢰성 높은 결과를 도출할 수 있습니다.

관련 논문 2: KODOLI ("Why do I feel offended?" Korean Dataset for Offensive Language Identification)

- KODOLI는 한국어에서 공격적 언어를 식별하기 위한 데이터셋과 이를 활용한 multi-task learning프레임워크를 제안하는 논문입니다.

  데이터셋 구성

  - KODOLI 데이터셋은 다양한 소스에서 수집된 댓글들로 구성되어 있으며, 총 38,525개의 댓글을 포함합니다. 주요 데이터 소스는 다음과 같습니다:
    1. **커뮤니티 사이트**: DC인사이드에서 수집된 데이터.
    2. **뉴스 플랫폼**: 네이버 뉴스 댓글에서 수집된 데이터. 상위 랭크 기사들의 댓글을 추출하여 각 기사로부터 최대 500개의 댓글을 수집했습니다.
    3. **기타 플랫폼**: 쇼핑 및 게임 플랫폼에서 수집된 데이터.
  - 데이터 수집 과정에서는 중복 데이터와 의미 없는 특수문자를 제거하여 데이터의 품질을 향상시켰습니다. 또한, 데이터가 특정한 편향을 가지지 않도록 다양한 소스로부터 균형 잡힌 데이터를 수집하는 데 중점을 두었습니다.

  태스크 정의

  - KODOLI는 멀티-태스크 러닝을 통해 세 가지 주요 작업을 동시에 수행합니다:
    1. **Offensive Language Detection**: 댓글이 공격적 언어를 포함하고 있는지 여부를 판단합니다. 세 가지 라벨(OFFEN, LIKELY, NOT)로 분류됩니다.
       - **OFFEN**: 명백히 공격적 언어를 포함하는 댓글.
       - **LIKELY**: 잠재적으로 공격적일 수 있는 댓글.
       - **NOT**: 공격적이지 않은 댓글.
    2. **Abusive Language Detection**: 댓글이 욕설을 포함하고 있는지 여부를 판단합니다. 두 가지 라벨(ABS, NON)로 분류됩니다.
       - **ABS**: 욕설을 포함하는 댓글.
       - **NON**: 욕설을 포함하지 않는 댓글.
    3. **Sentiment Analysis**: 댓글의 감정을 판단합니다. 세 가지 라벨(POS, NEU, NEG)로 분류됩니다.
       - **POS**: 긍정적인 댓글.
       - **NEU**: 중립적인 댓글.
       - **NEG**: 부정적인 댓글.

  모델링 및 학습

  - KODOLI는 사전 학습된 언어 모델(PLM)을 활용하여 멀티-태스크 학습을 수행합니다. 주로 사용된 모델들은 다음과 같습니다:
    1. **BiLSTM**: 양방향 LSTM을 사용하여 문장의 순서 정보를 학습합니다.
    2. **CNN**: 합성곱 신경망을 사용하여 문장의 지역적 패턴을 학습합니다.
    3. **KoBERT**: 한국어에 특화된 BERT 모델을 사용하여 문장의 깊은 의미를 학습합니다.
    4. **KoELECTRA**: 한국어에 특화된 ELECTRA 모델을 사용하여 문장의 양방향 정보를 학습합니다.
  - 데이터 전처리 과정에서는 형태소 수준의 토큰화(Mecab-ko)와 WordPiece 토크나이저를 사용하여 문장을 적절한 형태로 변환하였습니다. 또한, 하드 파라미터 공유 기법을 통해 각 작업의 관련성을 최대한 활용하여 모델의 성능을 향상시켰습니다.

  실험 결과

  - 실험 결과, KoELECTRA 모델이 대부분의 작업에서 가장 높은 성능을 보였습니다. 특히 offensive language detection 작업에서 KoELECTRA는 Macro Average 성능이 가장 우수하였으며, 다음과 같은 성능을 기록했습니다:
    - **Offensive Language Detection**: 높은 정확도와 F1 스코어.
    - **Abusive Language Detection**: 높은 정확도와 F1 스코어.
    - **Sentiment Analysis**: 높은 정확도와 F1 스코어.

- 모든 모델이 "Likely Offensive" 항목에서 성능이 떨어지는 경향을 보였지만, KoELECTRA는 다른 모델들에 비해 상대적으로 높은 성능을 보였습니다. 이는 모델이 공격적인 의도를 숨기려는 단서를 찾는 데 어려움을 겪기 때문입니다.

결론

- KODOLI는 한국어 공격성 언어 식별을 위한 중요한 데이터셋과 모델링 접근법을 제안합니다. 멀티-태스크 러닝을 통해 다양한 작업을 동시에 학습함으로써, 기존의 단일 작업 기반 접근법보다 더 정교한 분석을 가능하게 합니다. 이 논문은 한국어 NLP 분야에서 중요한 기여를 하였으며, 향후 다양한 응용 분야에서 활용될 수 있을 것입니다.

3. 제안하는 모델

선정 모델

- KoBERT
- KoElectra

선정 이유

- **Pre-trained Language Model** : 대규모 데이터셋에서 학습되어 다양한 언어적 특성을 잘 이해하고 있는 PLM을 사용한다면, 제한된 데이터 환경에서도 높은 성능을 낼 수 있는 모델을 구축할 수 있을 것이라는 아이디어에서 모델 선정 과정을 시작하였습니다.
- **관련 연구동향에서의 언급**: PLM 모델 중, 앞서 살펴본 관련 연구동향에서의 KoBERT와 KoELECTRA가 각각 한국어 텍스트의 공격성 식별 및 감정 분석에서 높은 성능을 보인 바 있습니다. 특히 KODOLI 데이터셋을 활용한 연구에서 이 두 모델의 유효성이 입증되었습니다.
- **한국어에 특화된 모델**: KoBERT와 KoELECTRA는 한국어 데이터셋을 기반으로 사전 학습되었기 때문에, 한국어의 언어적 특성을 잘 반영하고 있습니다.

→ 이와 같은 요소를 고려하여 KoBERT, KoElectra 두 모델을 선정해 실험을 진행하도록 하였습니다.

모델 설명

- **BERT (Bidirectional Encoder Representations from Transformers)**
  - BERT는 Transformer 인코더 구조를 기반으로 한 모델로, 문장을 양방향으로 이해할 수 있게 설계되었습니다. 이는 기존의 단방향 모델과 달리, 문맥을 보다 깊이 이해할 수 있도록 합니다.
    1. **Transformer란?**
       - **인코더-디코더 구조**: Transformer는 인코더와 디코더 구조로 이루어져 있습니다. 인코더는 입력 시퀀스를 이해하고, 디코더는 출력 시퀀스를 생성합니다.
       - **BERT의 인코더 구조**: BERT는 인코더 구조만을 활용하여 문장을 양방향으로 이해할 수 있도록 설계되었습니다.
    2. **사전 학습 과정 (Pre-training task)**
       - **Masked Language Model (MLM)**: 시퀀스에서 일부 토큰을 마스킹하고, 모델이 이 마스킹된 토큰을 예측하도록 학습합니다.
       - **Next Sentence Prediction (NSP)**: 두 문장이 연속된 문장인지 아닌지를 예측하는 이진 분류 작업을 수행합니다. 이를 통해 문장 간의 관계를 학습합니다.
- **ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately)**
  - ELECTRA는 BERT와 유사한 Transformer 인코더 구조를 가지고 있지만, 학습 방식에서 차별점을 보입니다. ELECTRA는 Replaced Token Detection 방식을 사용하여 더 효율적인 학습을 수행합니다.
    1. **ELECTRA란?**
       - Transformer 인코더 구조를 기반으로 한 모델로, 문장을 양방향으로 이해할 수 있게 설계되었습니다.
    2. **BERT와 다른 점**
       - **Replaced Token Detection**: ELECTRA는 Generator와 Discriminator를 함께 학습시킵니다.
         - **Generator**: 시퀀스에서 [MASK]로 교체된 토큰을 예측합니다.
         - **Discriminator**: 시퀀스 내의 모든 토큰에 대해 그것이 원래 시퀀스와 동일한지 여부를 예측합니다.
       - 최종적으로 Discriminator만 사용하여 모델의 효율성을 높입니다.

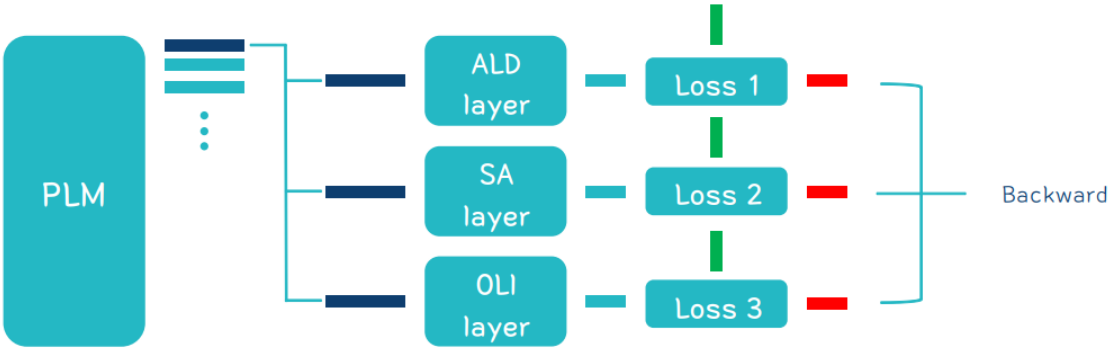
4. 실험 및 개발

Multi-task Learning

- 여러 작업 간의 파라미터를 공유한다는 아이디어.

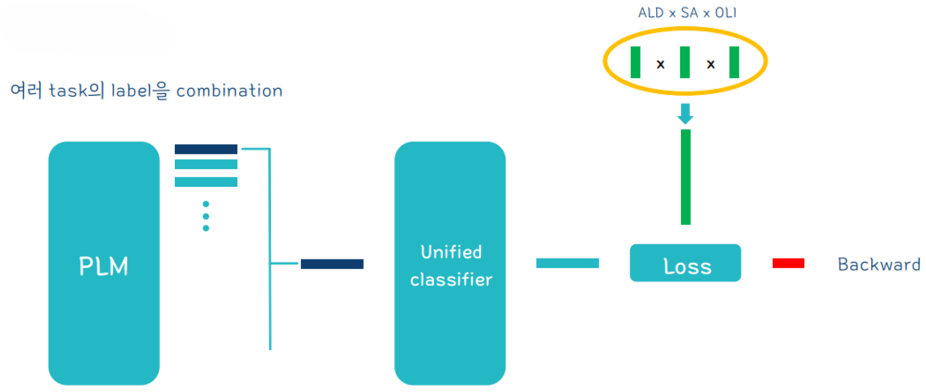
  → 한 작업이 다른 작업의 성능 향상에 기여함을 기대할 수 있습니다.

기존 논문의 방법론



- **Shared Layers**: 모든 작업에 공통으로 사용되는 인코더 레이어를 의미합니다.
- **Task-specific Layers**: 각 작업에 특화된 레이어를 의미합니다.

Unified Classifier 를 통한 Multi-task Learning 제안



- 우리는 한 단계 나아가 여러 작업들의 레이블을 결합하여 Unified Classifier를 사용해 학습하는 방법을 제안했습니다. 이 방법은 각 작업의 상관 관계를 최대한 활용하여 모델의 성능을 더욱 향상시킬 수 있을 것이라 가설을 세웠습니다.

Mecab_ko 토크나이저 사용

- 본격적인 실험에 앞서 KoBERT 모델에 기본토크나이저인 sentencepiece 를 사용했을 때와 Mecab_ko를 사용했을 때의 성능을 간단히 비교하여 어떠한 토크나이저를 사용할지에 대한 고민도 해보았습니다.

Epoch	Mecab_ko Train Acc	Mecab_ko Test Acc	SentencePiece Train Acc	SentencePiece Test Acc
1	0.4737	0.4635	**0.5951**	**0.6001**
2	0.5247	0.4721	**0.6568**	**0.6006**
3	0.5737	0.4840	**0.7142**	**0.6029**
4	0.6029	0.4871	**0.7549**	**0.6029**
5	0.6106	0.4805	**0.7549**	**0.6024**

5. 실험 결과 및 분석

Hyperparameter

- 이하와 같이 다양한 하이퍼파라미터들로 실험을 진행했습니다.

  전처리

시퀀스 최대 길이	64, 32 ...

배치 크기	64, 128 ...
Train / Test 비율	80/20, 90/10 ...
Tokenizer	Basic, Mecab_ko ...

모델링

Classifier Hidden Dimension	no, 512, 256 ...
Classifier Activation F	ReLU, Sigmoid, Tanh ...
Classifier Initialization	normal, xavier, he ...
Classifier Dropout	0, 0.1, 0.5 ...

학습

Optimizer	AdamW ...
Scheduler	Cosine ...
Gradient Clipping	1, 5 ...
Learning Rate	1e-3 ...
Epochs	5, 10 ...

Classifier modeling

KoBERT – Classifier Modeling

Metric	ALD	SA	OLI
Unified Classifier	**91.47%**	75.91%	**81.38%**
Specific Classifier for each Task	90.62%	**79.68%**	78.12%

KoELECTRA – Classifier Modeling

Metric	ALD	SA	OLI
Unified Classifier	**88.86%**	**74.24%**	**80.49%**
Specific Classifier for each Task	65.##%	70.##%	78.##%

왜 unified 가 대체로 성능 더 좋은가?

- 클래스 상호 작용과 데이터 특성의 통합
  - 부정적인 감정이 표현될 때 욕설과 공격성 수반될 확률이 높습니다. Label Combination 이 모델로 하여금 클래스를 상호 고려할 수 있도록 학습했을 가능성이 존재합니다. 3가지 task 는 독립적이지 않고 어떠한 상관관계가 분명 존재하기 때문입니다.
- 데이터 불균형
  세 클래스를 독립적으로 평가하므로 데이터 불균형이 학습과정에서 극대화됐을 가능성이 존재합니다.

Freezing strategy

KoBERT – Freezing (Unified)

Model	ALD	SA	OLI
BERT no freezing	**91.47%**	**75.91%**	**81.38%**
BERT 12th, 11th, 10th layer train	88.99%	72.24%	78.61%
BERT 12th, 11th layer train	87.95%	70.58%	77.28%
BERT 12th layer train	84.83%	67.53%	74.71%
BERT no train	67.12%	40.59%	64.39%

KoBERT – Freezing (Specific)

Model	ALD	SA	OLI
BERT no freezing	64.06%	35.93%	81.38%

Model	ALD	SA	OLI
BERT 12th, 11th, 10th layer train	89.06%	65.62%	70.31%
BERT 12th, 11th layer train	**90.62%**	**79.68%**	**78.12%**
BERT 12th layer train	81.25%	70.31%	70.31%
BERT no train	87.50%	60.93%	81.25%

KoELECTRA – Freezing (Unified)

Model	ALD	SA	OLI
ELECTRA no freezing	65.##%	35.##%	65.##%
ELECTRA 10th, 11th layer train	**88.86%**	**74.24%**	**80.49%**
ELECTRA 11th layer train	88.58%	73.58%	79.09%
ELECTRA no train	81.##%	57.##%	71.##%

KoELECTRA – Freezing (Specific)

Model	ALD	SA	OLI
ELECTRA no freezing	65.##%	35.##%	65.##%
ELECTRA 10th, 11th layer train	65.##%	70.##%	78.##%
ELECTRA 11th layer train	**65.##%**	**71.##%**	**78.##%**
ELECTRA no train	65.##%	65.##%	74.##%

KoBERT(Unified)와 KoELECTRA의 동결전략 차이

- KoBERT, KoELECTRA 모두 한국어 위키 데이터로 사전학습을 수행한 모델입니다. KoELECTRA의 경우 '모두의 말뭉치'를 써서 추가 학습을 수행했습니다. 실제 사람들의 수필, 블로그 글 등이 담겨있습니다.
- KODOLI 데이터가 KoBERT보다 KoELECTRA의 사전학습 데이터에 더 유사하다고 모델이 판단했을 가능성이 존재합니다.

KoBERT with Unified Classifier와 with Specific Classifier for each Task의 동결전략 차이

- 앞서 Label Combination이 모델로 하여금 클래스를 상호 고려할 수 있도록 학습했을 가능성에 대해 언급한 바 있습니다. 이와 같은 작업은 feature 간의 복잡한 상호작용을 모델링해야할 필요가 있습니다. 보다 깊은 모델의 tuning이 필요합니다.

  → 따라서 모델 전체를 재학습하는 전략이 유효할 수 있습니다.
- 반면 Specific Classifier for each Task의 경우 abuse, sentiment, offensiveness 에 대해 독립적으로 고려합니다. 상대적으로 낮은 복잡성을 가집니다. 그렇기에 상대적으로 사전 학습 구조를 유지하며 학습하는 것이 용이합니다.

  → 따라서 상위 layer + classifier만 재학습시키는 전략이 유효할 수 있습니다.

Task combination

KoBERT

Metric	ALD	SA	OLI
ALD + SA + OLI	**91.47%**	75.91%	**81.38%**
SA + OLI	Non	**76.91%**	80.95%
ALD + OLI	91.33%	Non	80.98%
OLI	Non	Non	80.23%

Metric	Pr - not	R - not	F1 - not	Pr - likely	R - likely	F1 - likely
ALD + SA + OLI	89.67	**92.71**	91.17	**41.89**	36.08	38.77
SA + OLI	**90.56**	92.00	**91.27**	40.77	36.24	38.37
ALD + OLI	90.20	92.21	91.19	40.82	37.51	39.09
OLI	90.36	91.35	90.85	39.27	**42.11**	**40.64**

Metric	Pr - off	R - off	F1 - off	Pr - avg	R - avg	F1 - avg
ALD + SA + OLI	75.01	74.41	74.71	68.85	67.73	68.21

Metric	Pr - off	R - off	F1 - off	Pr - avg	R - avg	F1 - avg
SA + OLI	74.03	**76.12**	75.06	68.45	**68.12**	68.23
ALD + OLI	75.80	74.57	**75.18**	68.94	68.09	**68.48**
OLI	**77.28**	70.19	73.56	**68.97**	67.88	68.35

KoElectra

Metric	ALD	SA	OLI
ALD + SA + OLI	**88.86%**	**74.24%**	**80.49%**
SA + OLI	Non	72.92%	79.55%
ALD + OLI	88.19%	Non	80.21%
OLI	Non	Non	77.19%

Metric	Pr - not	R - not	F1 - not	Pr - likely	R - likely	F1 - likely
ALD + SA + OLI	**89.45**	91.40	**90.41**	**37.92**	**28.38**	**32.46**
SA + OLI	87.15	92.23	89.62	31.94	15.79	21.13
ALD + OLI	86.38	93.80	89.94	37.13	17.16	23.47
OLI	82.51	93.55	87.68	23.91	5.03	8.32

Metric	Pr - off	R - off	F1 - off	Pr - avg	R - avg	F1 - avg
ALD + SA + OLI	70.62	**75.80**	**73.12**	**66.00**	**65.19**	**65.33**
SA + OLI	68.84	75.93	72.22	62.64	61.31	61.00
ALD + OLI	**70.90**	73.36	72.11	64.80	61.44	61.84
OLI	65.04	66.80	65.90	57.15	55.13	53.97

→ 단독 OLI 사용시보다, Task Combination 수행했을 때 성능이 높게 나오는 경향을 확인할 수 있었습니다.

→ 이는 KoBERT 보다 KoElectra에서 더 두드러집니다.

6. 한계

- Resource에 한계가 있어 Hyperparameter Tuning 에 어려움이 있었습니다.
- PLM이 큰 모델이라 학습 중 gradient flow가 원활하게 이루어지지 않아서 freezing하지 않았을 때 성능이 덜 나온 부분이 존재했던 것으로 추측되는 문제 또한 있었습니다.
- 가설 검증을 위한 충분한 시간이 부족해 정확도가 높고 낮게 나오는 확실한 이유에 대해 알기는 어려웠습니다.

→ 지면 여유상 첨부는 못했지만 Optuna 와 같은 라이브러리로 주어진 resource 내에서 해결하려는 시도를 하였으며

→ 충분한 시간을 가지고 재시도하는 방안으로 문제를 어느 정도 해결할 수 있을 것으로 보입니다.

7. 해당 모델 서비스 방향 또는 사회적 영향

- 고파스, 에브리타임 댓글 숨기기 기능
- 저연령대를 위한 댓글 필터링
- LLM 모델의 출력에 대한 Model Alignment 확보

위와 같은 방향으로 활용이 가능할 것입니다. 이를 통해

→ 무분별하게 욕설 및 혐오 표현에 노출될 위험성 감소

→ 소통과 즐거움의 창구여야 할 인터넷 속에서 스트레스 받을 확률 감소

→ 언어 습득기에 있는 저연령층의 올바른 언어 사용능력 함양에 도움.

→ AI 윤리 준수

와 같은 효과를 기대할 수 있습니다.