비속어 포함 문장 판별 AI 모델 설계 및 구현

-CNN과 KoBERT를 중심으로-

2022105767 인공지능학과 권주명
2022105777 인공지능학과 도윤서
2022105778 인공지능학과 방채빈

목차

요약

비속어 포함 문장 판별 알고리즘은 주어진 문장의 비속어 포함 여부를 판별해주는 알고리즘이다. 본 프로젝트에서는 딥러닝 모델 중 CNN과 KoBERT를 이용하여 욕설뿐만 아니라 변형된 비속어까지 판별할 수 있도록 한다.

1. 서론

게임이나 댓글에서는 갖은 욕설과 가족의 안부를 묻는다든가, 특정 성별을 비하한다든가, 특정 집단을 혐오하는 등의 비윤리적 표현을 볼 수 있다. 현재 이를 저지하기 위해 대부분의 온라인 게임에서는 욕설을 필터링하는 서비스를 지원한다. 하지만 사람들은 유의어를 이용해 비속어 필터링을 피하려 한다. 교묘하게 필터링을 피해 간 비속어까지 차단할 수 있다면 쾌적한 인터넷 환경을 만드는 데에 유용할 것이라 생각하였다. 텍스트를 입력받으면 욕설이 포함된 문장인지 검출하고, 딥러닝 모델을 이용해 유의어까지 잡아낼 수 있도록 할 것이다. 수업 시간에 CNN을 활용한 자연어 처리에 대해 배워 CNN 모델을 선택했고, 프로젝트를 진행하기 위해 논문을 찾아보니 자연어 처리 분야에서는 KoBERT 모델을 사용하는 경우가 많아 이 모델을 선택했다. CNN, KoBERT 모델을 비교하여 가장 성능이 뛰어난 알고리즘 모델을 만들고자 한다.

2. 기존연구

2.1 딥러닝 기반 비속어 필터링 채팅 프로그램 설계 및 구현

이 논문에서는 'Text-CNN'을 활용한 딥러닝 기법에 기반하여 비속어 필터링 채팅 프로그램을 제안한다. 데이터의 자질을 '자모' 단위로 전처리하여 학습시키고 어느 부분이 비속어인지 검출하여 마스킹 처리하는 'LIME 알고리즘'을 사용한다.

이 논문에서는 직접 특정 악성 커뮤니티 사이트들에서 댓글들을 크롤링하여 수집 후, 각각의 댓글마다 분류 기준을 두어 레이블링 처리를 하여 데이터를 만들어 사용했다. 총 수집한 데이터 수는 약 400,000개이고, 이 데이터를 형태소 단위와 자모 단위 두 가지 방법으로 전처리한다. 자모 단위로 분해한 각각의 심볼들은 크기가 127인 One-hot Vector로 표현되며 입력의 길이의 최대 크기는 200으로 두었다. 200보다 짧은 길이의 데이터는 Zero-padding을 붙여 크기가 200이 되게 맞추었다.

데이터들을 학습시킬 딥러닝 모델로는 CNN 모델을 사용했다. CNN 모델은 보통 CV tack에 이용되지만, 최근 CNN을 NLP(Natural Language Processing, 자연어 처리) 문제에 활용하기 위한 많은 시도가 있었고, CNN 모델이 NLP 문제에 적합하다는 것이 여러 실험을 통해 밝혀졌다. CNN의 가장 큰 장점은 입력 데이터의 특징을 추출하는 데 적합하다는 것이다. 합성곱은 컴퓨터 그래픽의 핵심적인 부분이며, GPU 레벨로 구현되어 있다. 본 논문에서 사용하는 CNN 모델의 구조는 다음과 같다.

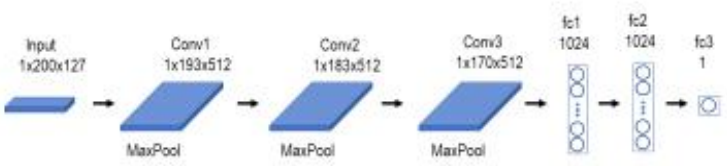

그림 1 CNN 구조 (딥러닝 기반 비속어 필터링 채팅 프로그램 설계 및 구현, 2019)

총 3개의 Convolution Layer와 3개의 FC Layer를 사용하였다. Convolution의 kernel size는 순서대로 6, 9, 12이다. 모든 Convolution Layer의 뒤에는 Max Pooling을 사용하였으며 Kernel size는 3으로 동일하다. 각 FC Layer의 마지막 단에는 Overfitting을 방지하기 위해 Drop out을 두었다. Loss function은 Binary Cross Entropy Loss를 사용하였고, Optimizer는 Adam을 사용했다. Learning rate는 0.0001로 두고 20 epoch마다 Learning rate를 0.1씩 감소시켰다.

마스킹 처리는 LIME(Local Interpretable Model-Agnostic Explanations) 알고리즘을 사용하였다. 이는 입력 문장 내 단어들을 각각 다르게 조합한 샘플 단어 벡터에 따라 예측값이 달라짐을 이용하여 측정한 결과에 대해 모델을 해석하며, 해석 가능한 모델을 이용하여 데이터의 지역적인 영역을 해석하는 것을 목표로 한다. LIME 알고리즘을 이용해 input data를 어절 단위로 나누어 욕으로 판단된 어절을 '*'로 마스킹한다.

Input	**존나 배고프다 시~~발**	
Analysis	모든 경우의 수	결과(욕일 확률)
	배고프다 시~~발	99.98
	존나 시~~발	99.98
	존나 배고프다	97.68
	시~~발	99.99
	배고프다	00.02
	존나	97.89
Output	** 배고프다 ****	

표 2.1 마스킹 처리 과정 (딥러닝 기반 비속어 필터링 채팅 프로그램 설계 및 구현, 2019)

자모 단위의 CNN과 형태소 단위의 LSTM 총 두 가지에 대해서 실험을 해 보았고, 성능은 자모 단위의 CNN은 94.13%, 형태소 단위의 LSTM은 85.82%로 자모 단위의 CNN의 성능이 약 8% 정도 높게 측정되었다. 웹사이트의 댓글이라는 비정형 데이터에 대해서 자모 단위로 했을 때 성능이 더 뛰어난 것을 볼 수 있다.

기존 비속어를 필터링할 수 있는 기술력은 사람들이 수동으로 금칙어를 지정해야 하는 '금칙어 기반'에 그쳤다. 이 논문에서는 '딥러닝' 기술을 활용하여 모델이 스스로 데이터를 학습

하여 데이터를 분류할 수 있는 프로그램을 소개했다. 이렇게 되면 사람이 직접 금칙어들을 제작할 필요가 없어지게 되고, 우회된 데이터와 비정형 데이터들에 대한 필터링 정확도도 증가한다. 이 아이디어가 올바른 인터넷 언어 문화를 조성하고, 올바른 언어습관을 갖게 하는 데 도움을 줄 것으로 기대된다.

2.2 BERT

1) 트랜스포머

트랜스포머 모델은 모델의 인코더와 디코더에서 RNN을 사용하지 않고 어텐션 기법만을 허용하는 단순한 네트워크 구조를 제안한다. 기존 모델에 비해 품질이 우수하고 병렬 처리가 가능하며, 학습에 더 적은 시간을 사용한다.

2) BERT

BERT는 트랜스포머의 인코더를 쌓아 올려 양방향 인코딩 표현을 사용하는 모델이다. 기본적으로 대량의 학습 데이터로 사전 학습을 진행 후에 필요한 문제 해결을 위해 더 적은 데이터를 사용하여 전이 학습을 진행한다.

BERT(Bidirectional Encoder Representations from Transformer)는 자연어 처리 분야 범용 모델을 제안한 것으로, 다양한 자연어 처리 임무를 동일한 구조의 학습된 모델로 해결할 수 있다. 트랜스포머에서는 활성화 함수로 ReLU함수를 사용하지만, BERT에서는 GELU함수를 사용한다. GELU는 ReLU보다 0 주위에서 부드럽게 변화해 학습 성능을 높인다는 장점이 있다.

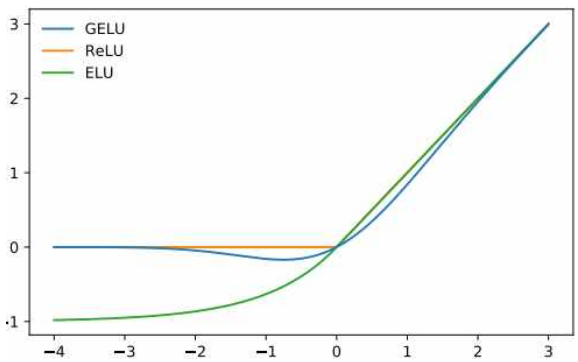

그림 2 GELU, ReLU, ELU 활성화 함수

(Hendrycks & Gimpel, 2016, p. 2)

BERT의 사전 훈련 방법은 크게 두 단계로 나뉜다. MLM(Masked Language Model, 마스크 언어 모델)과 NSP(Next Sentence Prediction)이다. 먼저, MLM 수행을 위한 데이터 생성은 입력 테스트 단어 중 15%를 랜덤으로 마스킹한다. 그런데 BERT는 마스킹의 대상이 되는 모든 단어를 [MASK]로 대체하는 것이 아니다. 10%의 단어들은 랜덤으로 다른 단어로 변경하고, 10% 단어들은 그대로 두어 모델은 해당 위치의 단어가 무엇인지 예측하도록 추가적인 조

건을 부여한다.

구분	텍스트 데이터
입력값	대학원 박사 과정은 매우 재밌고 유익하다
마스킹 후보	대학원 **박사** 과정은 **매우** 재밌고 **유익하다**
마스킹 후 입력값	대학원 [MASK] 과정은 **컴퓨터** 재밌고 **유익하다**

표 2.2 텍스트 데이터의 입력과 마스킹 결과 예시(권순보, 2022)

BERT에서는 각 문장이 토큰으로 나누어져 BERT 층을 통과한다. 이때 입력되는 문장 길이가 매번 다르기 때문에, 최대 문장 길이를 사전에 설정하고 문장의 끝 이후부터는 배치 데이터의 길이를 맞춰 주는 토큰인 [PAD]를 통해 길이를 맞춘다. [PAD]에서는 토큰이 실제로 존재하지 않는 부분을 0으로 채워주는 패딩 작업을 수행한다. 딥러닝 과정을 거친 데이터 형태가 축소되어 일부 데이터가 소실되는 문제점을 막기 위함이다. BERT 모델에서는 어텐션 마스크를 사용하여 0으로 패딩된 부분은 수행하지 않음으로써 학습 속도를 높일 수 있다. (서혜진, 신정아, 2020; 이기창, 2021)

이 논문에서는 모델을 학습하기 위해 Adam Optimizer을 사용했고 최대 길이 512, 배치 크기 16, 초기 학습률 0.00005(5e-5), 학습은 총 6 epochs를 수행하였다. 전체 데이터 기준으로 시험 데이터의 정확도는 0.79로 나타났다. 또한, 추론 기반의 언어 모델인 BERT는 지도 학습 방법으로 짧은 상담 문장을 비교적 높은 정확도와 정밀도, 재현율을 보여주며 분류하였다.

연구에서는 BERT의 한국어 모델 중 KoBERT를 활용하였지만, 현재 한국어 BERT 모델은 KorBERT, KR-BERT, HanBERT, KalBERT 등 다양한 모델들이 공개되어 있다.

3. 프로젝트

3.1 데이터 분석

3.1.1 데이터 소개

본 프로젝트에서는 AI Hub의 '텍스트 윤리검증 데이터'를 사용한다. 이 데이터는 1) 인공지능 윤리검증 데이터 구축의 실용적인 사례 및 기본 지침 개발 2) 대화형 에이전트 및 온라인 대화 등에 대한 윤리검증 AI 모델 개발을 위한 학습데이터 구축 3) 비윤리 데이터 사전 구축 선도사례 제시 및 인공지능 윤리 연구 자료 제공을 목적으로 구축되었다. 2022-07-12에 콘텐츠가 최초로 등록되었고, 이후 변경이력은 없다. 한국어 텍스트이며 형식은 JSON이고, 크라우드워커가 직접 생성한 데이터이다. 2021년에 데이터가 구축되었으며, 453,340문장, 데이터 세트 132,807건으로 이루어져 있다. 데이터는 train, val, test 세트로 나누어져 있다.

3.1.2 데이터 EDA

#	column	Non-Null Count	Dtype
0	talker	113948 non-null	int64
1	unethical text	113948 non-null	bool
2	intensity	113948 non-null	float64
3	chat	113948 non-null	int64
4	type	113948 non-null	int64
5	text	113948 non-null	object

표 3.1 train 데이터셋의 data info

#	column	Non-Null Count	Dtype
0	talker	12665 non-null	int64
1	unethical text	12665 non-null	bool
2	intensity	12665 non-null	float64
3	chat	12665 non-null	int64
4	type	12665 non-null	int64
5	text	12665 non-null	object

표 3.2 validation 데이터셋의 data info

train과 validation 데이터셋은 각각 talker(int64), unethical text(bool), intensity(int64), chat(int64), type(int64), text(object)의 총 여섯 가지의 칼럼을 가지는 것을 알 수 있다. talker는 화자에게 임의의 번호를 부여한 것이고, unethical text는 해당 텍스트의 비윤리성을 true/false로 나눈 것이다. intensity는 비윤리성의 강도를 의미한다. (unethical text가 false라면 intensity는 자동적으로 0에 해당한다.) chat은 채팅에 임의의 번호를 부연한 것이며 type은 텍스트의 비윤리 유형, text는 해당 텍스트를 의미한다. 여기서 우리는 unethical text type, text만을 사용할 것이다. 이들을 확인하기 위하여 비윤리적 유형이 각각 몇 가지 데이터를 갖고 있는지 알아보았다.

비윤리 유형	개수
IMMORAL_NONE	31499
DISCRIMINATION	31499
SEXUAL	18850
ABUSE	16445
VIOLENCE	15655

표 3.3 train 데이터셋의 type 칼럼의 class별 개수

비윤리 유형	개수
IMMORAL_NONE	3501
DISCRIMINATION	3501
SEXUAL	2095
ABUSE	1828
VIOLENCE	1740

표 3.4 validation 데이터셋의 type 칼럼의 class별 개수

데이터에는 5개의 클래스가 존재한다. 'IMMORAL_NONE', 'DISCRIMINATION', 'SEXUAL', 'ABUSE', 'VIOLENCE' 5개의 비윤리 유형을 각각 0, 1, 2, 3, 4로 라벨링 해 주었다.

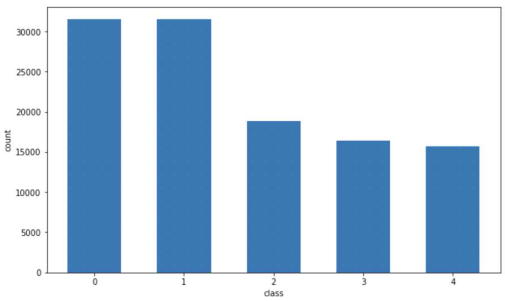

그림 3 train 데이터셋

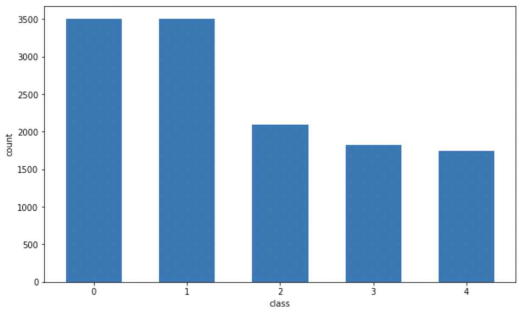

그림 4 validation 데이터셋

그래프를 통해 클래스별 개수 시각화해 본 결과, train 데이터셋과 validation 데이터셋의 비윤리 유형 분포 비율은 각 클래스별로 유사한 것을 알 수 있다.

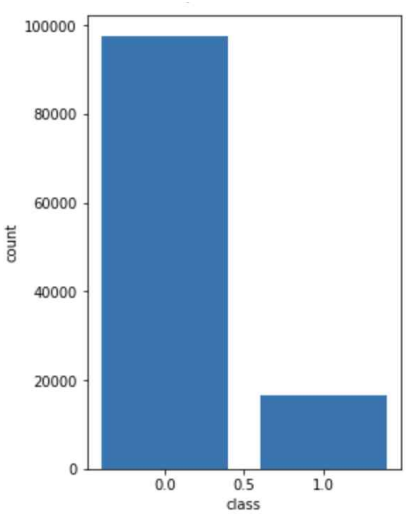

그림 5 train 데이터셋

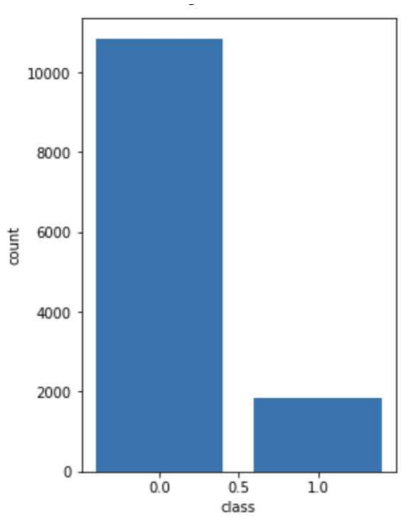

그림 6 validation 데이터셋

text의 욕설 여부를 판단하기 위해 사용할 욕설 class와 나머지 class의 분포 비율 또한 시각화해보았다. 위와 동일하게 train, validation의 데이터셋 분포 비율이 유사한 것을 알 수

있다.

	talker	unethical text	intensity	chat	type
talker	1.000000	0.043837	0.064822	0.224552	0.074139
unethical text	0.043837	1.000000	0.878142	0.060231	0.708670
intensity	0.064822	0.878142	1.000000	0.061456	0.654568
chat	0.224552	0.060231	0.061456	1.000000	0.065021
type	0.07413	0.708670	0.654568	0.065021	1.000000

표 3.5 train 데이터셋의 상관계수

train 데이터셋의 상관계수를 구한 결과, intensity와 unethical text의 상관계수가 0.878142로 가장 높게 나타났다. 하지만 text 칼럼은 각 샘플들의 길이가 모두 다른 object 형이기 때문에 다른 칼럼들과의 상관계수를 구하지 못하였다. 본 프로젝트에서는 type과 text 칼럼만 사용할 것이기 때문에 의미는 없어 보인다.

3.1.3 데이터 전처리

먼저 데이터를 dataframe형식으로 사용하기 위해 엑셀에서 train, validation, test 데이터를 불러와 csv형식으로 변환했다. 데이터 eda에서 확인해본 결과 데이터의 결측치는 존재하지 않아 결측치 처리는 따로 필요하지 않았다. 비윤리 데이터셋 중 우리가 사용해야 하는 type이라는 column은 데이터 eda에서 확인해본 결과 object 데이터타입으로 이루어져 있었다. 이 column을 모델에 넣어 사용하려면 데이터 전처리가 필요했다. 각 상황에 맞게 이진분류 문제를 수행할 때에는 type이라는 column의 5가지 클래스 중 욕설을 1, 나머지를 0으로 라벨 인코딩 후 int형으로 형변환 시켜주었다. 여기서 라벨 인코딩은 n개의 범주형 데이터를 0부터 n-1까지의 연속적 수치 데이터로 표현하는 것을 말한다.

변환 전	변환 후
IMMORAL_NONE(도덕/무도덕)	0
DISCRIMINATION(차별)	0
SEXUAL(선정)	0
VIOLENCE(폭력)	0
ABUSE(욕설)	1

표 3.6 이진분류 type 라벨 인코딩

다중 분류 문제를 수행할 때에는 type이라는 column의 class는 총 5개로 이루어져 있기 때문에 0은 도덕/무도덕, 1은 차별, 2는 선정, 3은 욕설, 4는 폭력으로 라벨 인코딩 후 int형으로 형변환 시켜주었다.

변환 전	변환 후
IMMORAL_NONE(도덕/무도덕)	0
DISCRIMINATION(차별)	1
SEXUAL(선정)	2
VIOLENCE(폭력)	3
ABUSE(욕설)	4

표 3.7 다중분류 type 라벨 인코딩

3.2 CNN 모델

3.2.1 이진 분류

CNN 모델을 활용해 문장의 욕설 포함 여부를 판별하는 이진 분류 모델을 구현했다.

1) 시행착오

	pooling filter (세로길이)	dropout	epoch	learning rate	batch size	train loss	val loss	train acc	val acc
시행착오 1	5	x	5	0.001	64	0.2909	0.5241	0.1443	0.1443
시행착오 2	34	o	5	0.0001	64	0.2655	0.5524	0.1443	0.1443

표 3.8 CNN 이진 분류 시행착오 비교

시행착오 1, 2에서는 activation 함수로 softmax, loss 함수로 binary crossentropy를 사용하였다. 풀링 필터의 세로 길이, 드롭아웃 유무, 학습률을 조정해가며 학습을 시켜본 결과 각각의 정확도가 터무니없이 낮은 값을 가지고, val loss 또한 높은 값을 가지는 것을 알 수 있다.

2) 재시도

이를 해결하기 위해 활성화 함수와 loss 함수를 기존 softmax, binary crossentropy에서 각각 sigmoid, mean_squared_error로 바꾸어 주었다. 함수 변화에 의한 차이를 알아보기 위해 시도 1에서는 위의 시행착오 2와 같은 조건으로 학습을 진행하였다.

	max features	learning rate	batch size	train loss	val loss	train acc	val acc
시도 1	150000	0.0001	64	0.0391	0.1580	0.9487	0.8009
시도 2	500	0.0001	64	0.1091	0.1277	0.8613	0.8480

표 3.9 CNN 이진 분류 재시도 비교

그 결과, train loss, val loss는 줄어들었고 train acc, val acc는 정상적인 수치로 나타나게 된 것을 알 수 있다. max features의 개수도 매우 높았기 때문에 이를 조정한 수치, 500으로 학습을 진행하였다.

그 결과로 시도 2에서는 시도 1에 비해 train loss는 늘어났지만, train acc와 val acc가 유사하게 나타나 과적합을 방지할 수 있었고, val loss 또한 줄일 수 있었다.

3.2.2 다중 분류

다음은 CNN 다중 분류 학습 결과이다. activation 함수와 loss 함수는 이중 분류 시 사용했던 함수와 동일하게 각각 sigmoid, mean squared error를 사용했다. 다중 분류를 진행할 때는 그리드 서치를 활용하여 batch size가 16, 32, 64, 128인 경우를 비교하였다.

	batch size	epoch	test acc mean	std
1	16	10	0.3034	0.0586
2	32	10	0.3070	0.0616
3	64	10	0.3087	0.0566
4	128	10	0.3132	0.0644

표 3.10 CNN 다중 분류 학습 결과

다중 분류에서는 batch size가 높아짐에 따라 test acc의 평균이 높아지는 경향을 보였으나, 전체적인 test acc의 평균은 낮다는 점을 알 수 있다.

3.3 KoBERT 모델

3.3.1 이진 분류

KoBERT 모델을 활용하여 주어진 문장의 욕설 포함 여부를 판별하는 이진 분류 모델을 구현했다.

모델의 성능에 큰 영향을 미칠 것이라 판단한 batch_size와 learning_rate에 한해 하이퍼파라미터 튜닝을 시도했다. batch_size는 32, 64, 128로 시도하고, learing_rate는 5e-6, 5e-5, 5e-4로 시도했다.

batch size / learning rate	32	64	128
5e-4	epoch=5 train acc=0.8556 test acc=0.8555	epoch=4 train acc=0.8554 test acc=0.8555	epoch=6 train acc=0.8549 test acc=0.8555
5e-5	epoch=5 train acc=0.9258 test acc=0.9058	epoch=2 train acc=0.9084 test acc=0.8643	epoch=5 train acc=0.9598 test acc=0.8596
5e-6	epoch=2 train acc=0.8982 test acc=0.8755	epoch=7 train acc=0.9228 test acc=0.9059	epoch=5 train acc=0.9127 test acc=0.9027

표 3.11 KoBERT 이진 분류 학습 결과

그 결과 도출된 KoBERT 이진 분류 모델에서의 최적의 학습환경은 다음과 같다. learning rate=5e-6, batch size=64일 때 test acc가 성능이 제일 높았다. 과대적합이나 과소적합이 이루어지지 않아 이 경우가 최적의 하이퍼 파라미터라고 판단하였다.

parameters	size
max_len	64
batch_size	64
warmup_ratio	0.1
num_epochs	7
max_grad_norm	1
log_interval	200
learning_rate	5e-6

표 3.12 KoBERT 다중 분류 최적의
하이퍼 파라미터

3.3.2 다중 분류

다음은 KoBERT 다중 분류 학습 결과이다. 모델의 성능에 큰 영향을 미칠 것이라 판단한 batch_size와 learning_rate에 한해 하이퍼 파라미터 튜닝을 시도했다. batch_size는 32, 64, 128로 시도하고, learing_rate는 5e-6, 5e-5, 5e-4로 시도했다.

batch size / learning rate	32	64	128
5e-4	학습 시간이 오래 소요되고, 정확도가 낮게 측정되었다.	epoch=4 train acc=0.2770 test acc=0.2779	epoch=4 train acc=0.4898 test acc=0.1655
5e-5		epoch=5 train acc=0.8654 test acc=0.7206	epoch=2 train acc=0.72 test acc=0.717
5e-6		epoch=7 train acc=0.7706 test acc=0.7204	epoch=7 train acc=0.7138 test acc=0.6834

표 3.14 KoBERT 다중 분류 학습 결과

그 결과 도출된 KoBERT 다중분류 모델에서의 최적의 학습환경은 다음과 같다. test acc는 learning rate=5e-5, batch size=64일 때 가장 높았지만, test acc보다 train acc가 0.14 높기 때문에 과대적합되었다고 볼 수 있다. 과대적합이나 과소적합되지 않은 모델 중 가장 성능이 나은 learning rate=5e-5, batch size=128일 때가 최적의 하이퍼 파라미터라고 판단하였다.

parameters	size
max_len	64
batch_size	128
warmup_ratio	0.1
num_epochs	7
max_grad_norm	1
log_interval	200
learning_rate	5e-6

표 3.15 KoBERT 다중 분류 최적의
하이퍼 파라미터

4. 결과

4.1 성능 비교

모델		test acc
CNN	이진분류	0.8480
	다중분류	0.3132
KoBERT	이진분류	0.9059
	다중분류	0.7170

표 4.1 각 모델 성능 비교

각 모델의 성능을 비교해 본 결과, KoBERT 이진 분류가 성능이 가장 높음을 알 수 있다.

4.2 예측

4.2.1 KoBERT 이진 분류

이진 분류 중 KoBERT가 가장 성능이 좋았기에 이 모델로 새로운 문장을 입력받아 문장에서의 욕설 포함 여부를 판별해 보았다.

욕설 O	욕설 X
개년아	캐논
개뇬	
ㄱHㅅHㄲ1	새끼를 꼬다
새끼야	새끼를 낳다
너는 재수가 없다	우리에게 재수는 없다
재수없다	재수 하지 말자
야이 개나리같은 아이야	개나리색 양말
ㅂ1ㅅ	방수
및힌	마침내

표 4.2 KoBERT 이진 분류 욕설 포함 여부 예측(성공)

성능이 가장 좋은 하이퍼 파라미터를 적용한 KoBERT 이진 분류 모델로 새로운 문장을 입력해 예측을 시도해 보았다. 예측 결과 욕과 같은 의미를 갖지만, 형태를 교묘하게 바꾼 단어가 포함된 문장도 욕설이 포함된 문장이라고 잘 예측하는 것으로 보인다. 또한, 욕과 비슷한 형태를 가졌지만 다른 의미를 가진 단어가 포함된 문장을 욕설이 포함되지 않은 문장이라고 예측하였다.

욕설 O	욕설 X
시바견	졸라
시발점	띠바
슈바이처	

표 4.3 KoBERT 이진 분류 욕설 포함 여부 예측(실패)

그러나 욕과 비슷한 형태이지만 다른 의미를 지닌 단어를 욕설이 포함된 문장이라 예측한 결과도 있었다. 또한 욕설과 같은 의미를 지닌 단어임에도 제대로 예측하지 못한 결과도 있었다. 그래서 이러한 문제점을 정확히 파악하기 위해 단어만 넣는 것이 아니라 이 단어가 포함

된 문장으로 다시 한번 예측을 진행해 보았다.

욕설 O	욕설 X
이 사건이 시발점이 되었다.	내 강아지는 시바견이야.
슈바이쳐 박사는 똑똑하다	아 망했다 띠바
졸라 맛있다.	

표 4.4 욕설 포함 여부 예측(예측에 실패한 단어가 포함된 문장으로 수정)

그 결과 '시바견'과 '졸라'라는 단어가 들어간 문장에 대해서는 앞의 단어만 넣은 경우와는 다르게 정확한 class로 분류하였다. 하지만 '시발점', '슈바이쳐', '띠바'가 포함된 문장의 경우에는 앞의 경우와 같이 잘못된 class로 분류하였다. 이러한 결과를 통해 이 모델은 특정 욕설 단어를 보고 욕설이라고 판단하는 것이 아니라, 문맥을 고려하여 문장에서의 욕설 포함 여부를 판단한다는 것을 알 수 있다. 하지만 문장으로 입력하였을 때에도 잘못 판별한 문장이 있듯이 이 모델의 성능이 완벽하지는 않다는 것을 알 수 있다.

4.2.1 KoBERT 다중 분류

다중 분류 중에서는 KoBERT가 가장 성능이 좋았기에 이 모델로 새로운 문장을 입력받아 문장에서의 욕설 포함 여부를 판별해 보았다.

predict	text
욕설	시발
	쭤빠지게 힘드네
	너 머리가 왜그러냐 (실패)
폭력	죽여버린다
	이제 보기만 해도 면상 그냥 아스팔트에 갈아버리고 싶어..
차별	저렇게는 살지 말아야지
	노키즈존 법적으로 전국 실시 해야됨;
	남녀평등시대라서 그런거 없어~ 그런게 어딧냐 ㅋㅋ (실패)
선정	진짜 엉덩이 죽이더라
	오늘도 즐떡~
도덕/무도덕	미안하다. 이해좀 해줘

표 4.5 KoBERT 다중 분류 예측

test 데이터셋에 있는 문장을 직접 입력해보았다. 예측에 실패한 도덕/무도덕인 두 문장을 제외하면 예상한 바와 같이 예측에 성공하였음을 알 수 있다.

4.3 한계

4.3.1 성능

앞서 제시한 알고리즘의 문제를 해결하기 위해서는 더 많은 데이터를 모델에 학습시키고, 단일 모델만 사용하는 것이 아니라 특정 모델의 한계를 극복할 수 있도록 다른 모델과 결합하는 방안을 사용한다면 좋은 결과가 있을 것이라 생각된다.

4.3.2 LIME 알고리즘

```
#예측 함수
predict_fn = model.predict()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-26-0683e02dc8e9> in <module>
----> 1 predict_fn = model.predict()

/usr/local/lib/python3.8/dist-packages/torch/nn/modules/module.py in __getattr__(self, name)
   1175             if name in modules:
   1176                 return modules[name]
-> 1177         raise AttributeError("'{}' object has no attribute '{}'".format(
   1178             type(self).__name__, name))
   1179

AttributeError: 'BERTClassifier' object has no attribute 'predict'
```

그림 7 LIME 알고리즘 실패1

```
#예측 함수
def predict_fn(text)):
  return model.run(probabilities, feed_dict={processed_texts: texts})
```

그림 8 LIME 알고리즘 실패2

본 프로젝트에서 구현한 네 가지 모델 중 가장 성능이 좋았던 KoBERT 이진 분류 모델을 이용한 LIME 알고리즘으로 욕설을 '*'로 마스킹 처리를 시도해 보았다. LIME에 예측 모델을 넣어주기 위해서는 예측 모델을 함수 형태로 만들어 주어야 한다. 예측 모델을 scikit-learn으로 만들었다면 모델의 predict_proba 함수를 쓰는 것으로 충분하지만, KoBERT 모델은 scikit-learn을 활용한 모델이 아니기 때문에 새로운 함수를 선언하여 예측 모델을 만들어 주어야 했다. 하지만 이 과정에서 막혀서 LIME 알고리즘을 구현하지는 못했다.

5. 결론

본 프로젝트에서는 딥러닝 모델 중 CNN과 KoBERT를 각각 이용하여 비속어 포함 문장을

판별해주는 알고리즘을 구현하였다. 이는 욕설이 포함된 문장을 거를 수 있어 깨끗한 인터넷 환경을 만들어 나가는 데 활용될 수 있다. 매번 동일한 데이터셋을 활용했던 수업 시간과는 달리, 실제 데이터를 기반으로 CNN 자연어 처리 모델에서 나아가 KoBERT 모델까지 구현해 보았다는 데에 의의가 있다. 예측에 실패한 경험을 토대로 더 많은 데이터를 학습시키고, 모델을 결합하는 방법으로 더 높은 성능을 가진 알고리즘을 만들 수 있을 것이라 기대된다.

참고문헌

[1] 이건환 (한국기술교육대학교 컴퓨터공학부) ; 박주찬 (한국기술교육대학교 컴퓨터공학부) ; 최동원 (한국기술교육대학교 컴퓨터공학부) ; 이연경 (한국기술교육대학교 컴퓨터공학부) ; 최호빈 (한국기술교육대학교 대학원 컴퓨터공학과) ; 한연희 (한국기술교육대학교 첨단기술연구소), 딥러닝 기반 비속어 필터링 채팅 프로그램 설계 및 구현, 한국정보처리학회 2019년도 추계학술발표대회, 2019년, p.998 - 1001

[2] 최준환, 딥 러닝 기반의 비속어 감지 시스템 설계 및 구현 = Design and implementation of abuse sentence detecting system based on Deep Learning, 서울 : 한양대학교 공학대학원, 2020

[3] 권순보, BERT를 활용한 진로상담 텍스트데이터 분석, 청주 : 한국교원대학교 대학원, 2022

[4] velog, “KoBERT”, https://velog.io/@seolini43/KOBERT%EB%A1%9C-%EB%8B%A4%EC%A4%91-%EB%B6%84%EB%A5%98-%EB%AA%A8%EB%8D%B8-%EB%A7%8C%EB%93%A4%EA%B8%B0-%ED%8C%8C%EC%9D%B4%EC%8D%ACColab, 2022.12.01.

[5] github, “LIME 알고리즘”, https://dreamgonfly.github.io/blog/lime/, 2022.12.02.