Transformer 기반 비윤리적 문장 탐지 모델

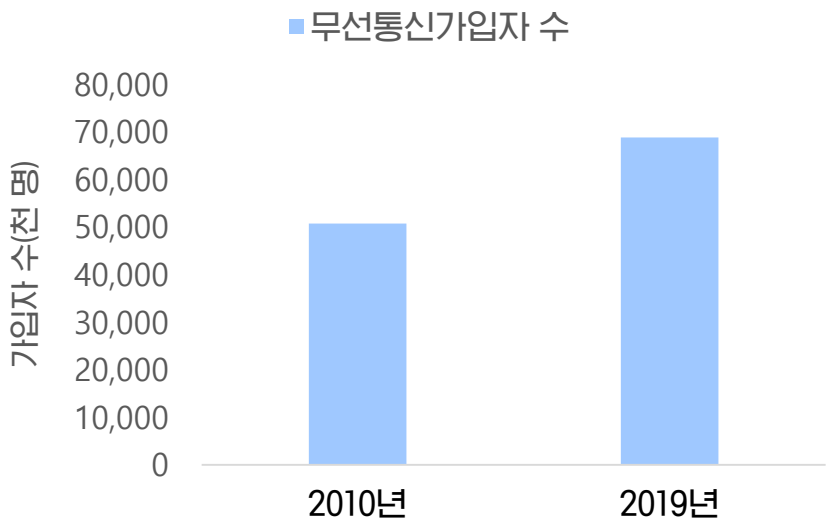
컴퓨터공학종합프로젝트2 최종 발표
컴퓨터정보통신공학과 182571 윤현서

목차

Introduction

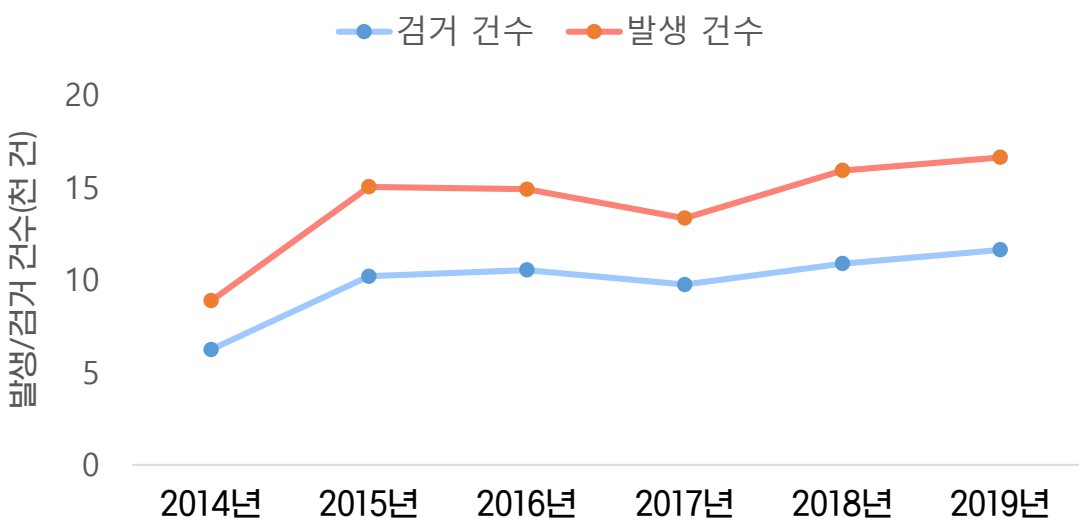
배경

무선통신가입자 현황



출처: 과학기술정보통신부

사이버 명예훼손 · 모욕 발생/검거 건수



출처: 경찰청

웹사이트와 SNS의 영향력 확대 → 심각한 사회적 문제인 **'악성 댓글'** 의 증가

Introduction

관련 연구

규제적 관점

네이버/다음의 연예 뉴스

댓글 폐지

타 포털 사이트

IP 블랙리스트, 비속어 필터

비속어 필터만으로는 비윤리적인 문장을
판별하기 어려움, 표현의 자유와 양립

인공지능 관점

CNN-LSTM 모델

정확도 80.94%

CNN-Attention 모델

정확도 70.32%

더 효율적인 학습 속도와 성능을 보유한
Transformer 모델을 활용하지 않음

‘Transformer 기반 비윤리적 문장 탐지 모델’ 제안

출처: 비윤리적 SNS댓글 분류를 위한 구문기반 CNN과 의미기반 LSTM의 앙상블 기법, 딥러닝 기술을 활용한 차별 및 혐오 표현 탐지 :어텐션 기반 다중 채널 CNN 모델링

Method

데이터셋

Korean HateSpeech Dataset
https://github.com/kocohub/korean-hate-speech
욕설 감지 데이터셋
https://github.com/2runo/Curse-detection-data
인공지능 윤리연구를 위한 비정형 텍스트 데이터셋
https://aihub.or.kr/keti_data_board/language_intelligence
Chatbot_data
https://github.com/songys/Chatbot_data

일반 댓글
26,769건

악성 댓글
24,481건

총 51,250건

학습 데이터	테스트 데이터
일반 댓글: 26,292건	일반 댓글: 477건
악성 댓글: 23,930건	악성 댓글: 551건

파이썬 라이브러리인
'SentencePiece' 를 사용하여 토큰화

출처:SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing

Method

토큰화된 문장
최대 길이 200
zero padding

Input

보고 싶은 영화입니다 좋은 영화 리뷰네요
'_보고', '_싶', '은', '**_영화**', '입', '니다', '_좋은', '**_영화**', '_리', '뷰', '네', '요'
1102, 2636, 3604, **339**, 3798, 1234, 1789, **339**, 160, 4397, 3857, 3760

Transformer
모델 구조

Position Embedding

토큰화된 데이터에 위치 정보를 포함하여 임베딩

Transformer Block

Multi-Head Attention + Feed Forward Network

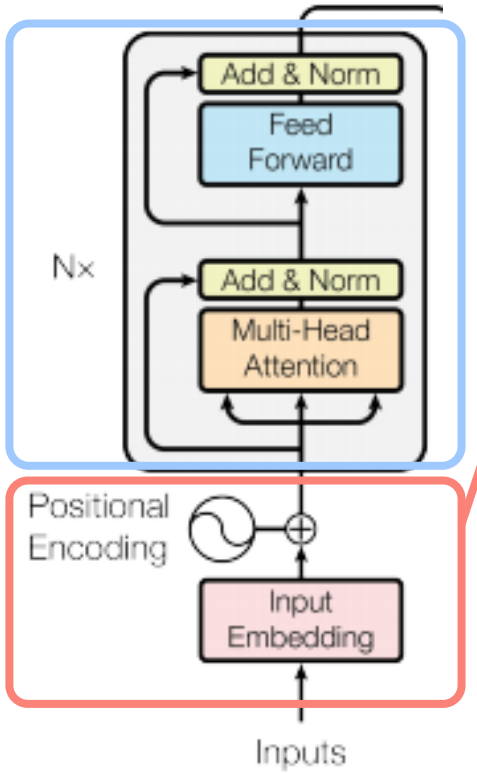
신경망 추가
Softmax 사용

Output

일반 댓글(0)과 악성 댓글(1)의 확률값 출력

Method

모델 코드

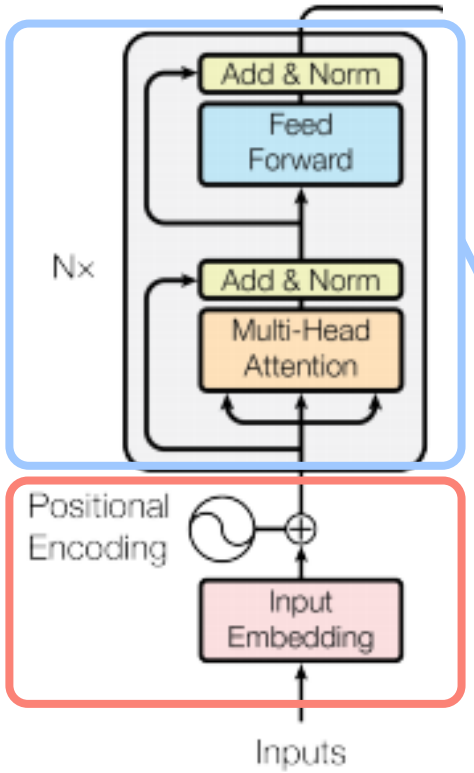



```
class TokenAndPositionEmbedding(layers.Layer):
    def __init__(self, maxlen, vocab_size, embed_dim):
        super(TokenAndPositionEmbedding, self).__init__()
        self.token_emb = layers.Embedding(input_dim=vocab_size, output_dim=embed_dim)
        self.pos_emb = layers.Embedding(input_dim=maxlen, output_dim=embed_dim)

    def call(self, x):
        maxlen = tf.shape(x)[-1]
        positions = tf.range(start=0, limit=maxlen, delta=1)
        positions = self.pos_emb(positions)
        x = self.token_emb(x)
        return x + positions
```

출처:Attention Is All You Need, https://keras.io/examples/nlp/text_classification_with_transformer/

Method

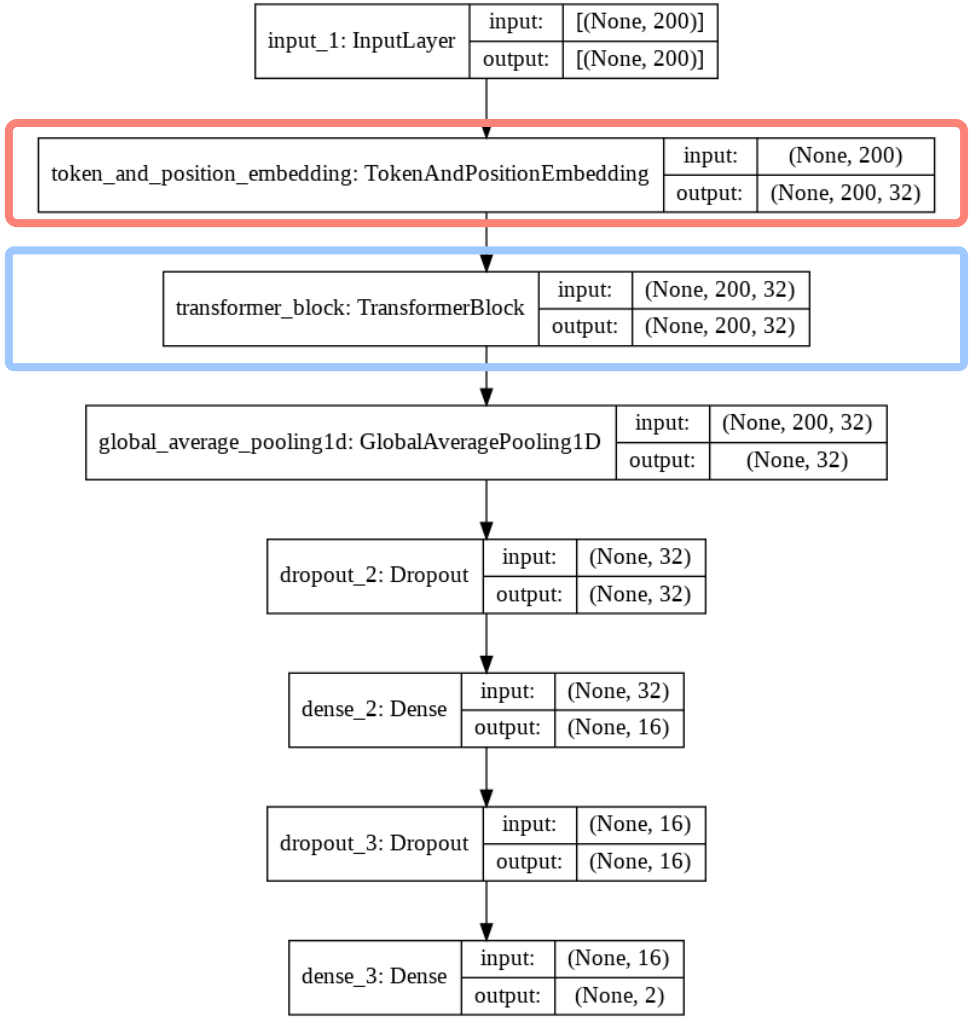



```
class TransformerBlock(layers.Layer):
    def __init__(self, embed_dim, num_heads, ff_dim, rate=0.1):
        super(TransformerBlock, self).__init__()
        self.att = layers.MultiHeadAttention(num_heads=num_heads, key_dim=embed_dim)
        self.ffn = keras.Sequential(
            [layers.Dense(ff_dim, activation="relu"), layers.Dense(embed_dim),]
        )
        self.layernorm1 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = layers.Dropout(rate)
        self.dropout2 = layers.Dropout(rate)

    def call(self, inputs, training):
        attn_output = self.att(inputs, inputs)
        attn_output = self.dropout1(attn_output, training=training)
        out1 = self.layernorm1(inputs + attn_output)
        ffn_output = self.ffn(out1)
        ffn_output = self.dropout2(ffn_output, training=training)
        return self.layernorm2(out1 + ffn_output)
```

출처:Attention Is All You Need, https://keras.io/examples/nlp/text_classification_with_transformer/

Method

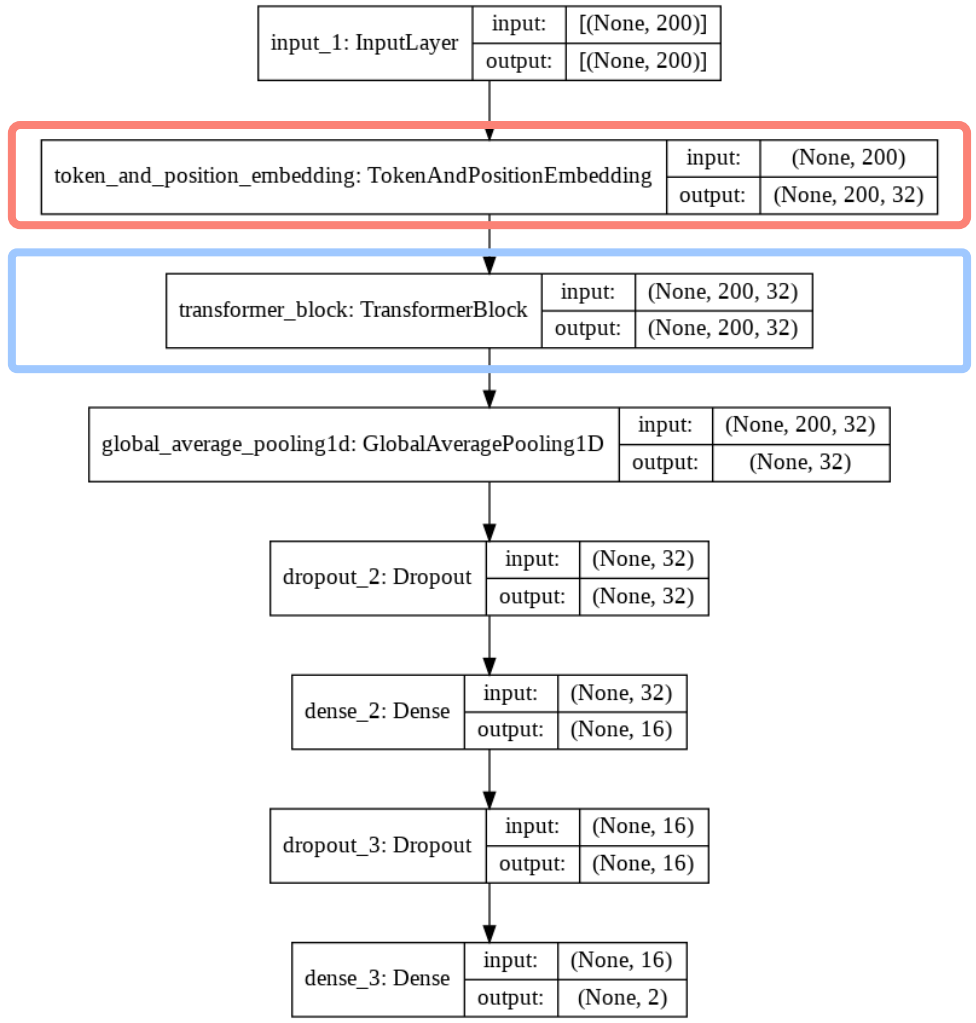



```
def make_model(maxlen, vocab_size):
    embed_dim = 32  # Embedding size for each token
    num_heads = 4  # Number of attention heads
    ff_dim = 32  # Hidden layer size in feed forward network inside transformer

    inputs = layers.Input(shape=(maxlen,))
    embedding_layer = TokenAndPositionEmbedding(maxlen, vocab_size, embed_dim)
    x = embedding_layer(inputs)
    transformer_block = TransformerBlock(embed_dim, num_heads, ff_dim)
    x = transformer_block(x)
    x = layers.GlobalAveragePooling1D()(x)
    x = layers.Dropout(0.1)(x)
    x = layers.Dense(16, activation="relu")(x)
    x = layers.Dropout(0.1)(x)
    outputs = layers.Dense(2, activation="softmax")(x)

    return keras.Model(inputs=inputs, outputs=outputs)
```

Method



최대 문장 길이 = 200

임베딩 크기 = 32 말뭉치 크기 = 20,000

Feed Forward Network 크기 = 32 Head = 4

Dropout 비율 = 0.1 활성화 함수 = ReLU

에폭 = 500 배치 크기 = 16

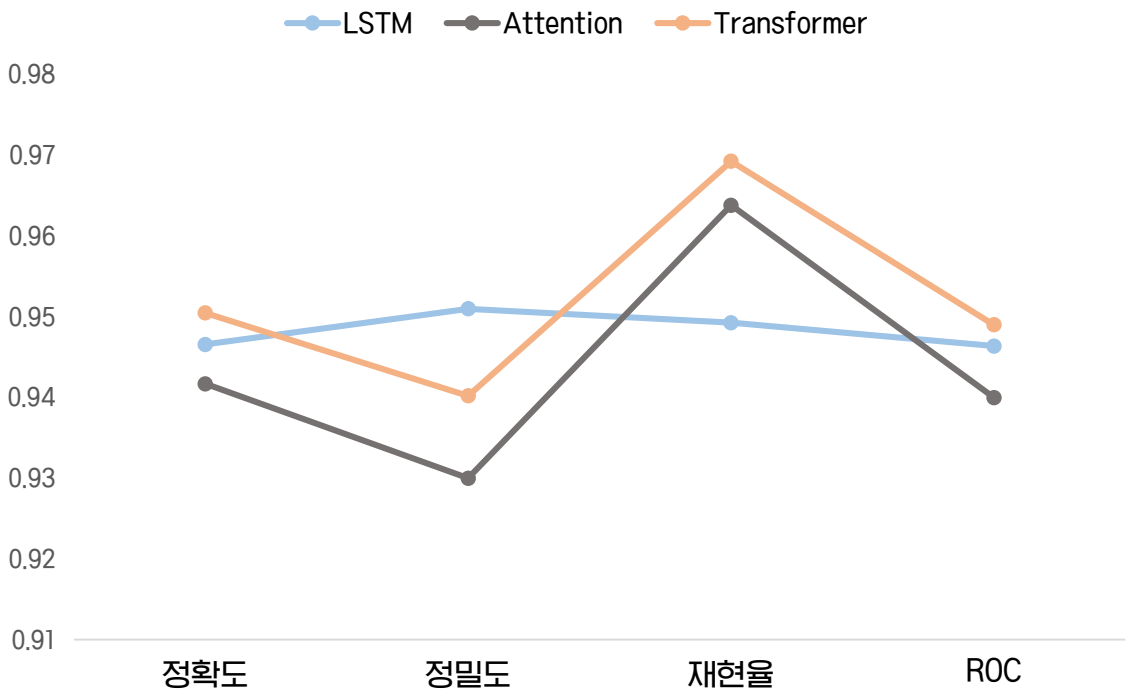
EarlyStopping, ReduceLROnPlateau 사용

검증 데이터 비율 = 0.2

옵티마이저 = Adam

Result

성능 평가



	정확도	정밀도	재현율	ROC
LSTM	0.95	**0.95**	0.95	0.95
Attention	0.94	0.93	0.96	0.94
Transformer	**0.95**	0.94	**0.97**	**0.95**

정확도와 재현율, ROC에서
Transformer를 활용한 모델이 우수한 성능을 보여줌

Result

성능 평가

“보고 싶은 영화입니다 좋은 영화 리뷰네요”	**일반 댓글(0): 95.75%** 악성 댓글(1): 4.25%
“LIX(흑인을 낮춰 부르는 영어 표현)는 거두면 안 된다.”	일반 댓글(0): 11.70% **악성 댓글(1): 89.30%**
“단맛의 라떼. 호불호 갈릴듯!”	일반 댓글(0): 30.36% **악성 댓글(1): 69.64%**
“어휴.. 너무 안타깝네요”	일반 댓글(0): 2.66% **악성 댓글(1): 97.34%**

일반 댓글을 악성 댓글로 예측하는 경우가 빈번

Discussion

기대효과

높은 성능과 빠른 학습 속도를 바탕으로 비윤리적 문장 탐지에 기여

댓글 뿐만 아니라 유튜브 등 스트리밍 서비스에서 활용 예상

한계

데이터셋의 크기가 작고 신뢰성이 다소 부족함

해당 모델은 이진 분류만을 수행하였지만
실제 환경에서는 다양한 인종, 지역, 성별, 나이 등 다양한 범주가 존재함

프로젝트 코드: https://github.com/0525hhgus/Detect-Bias-Sentence

감사합니다