KWU
오픈소스소프트웨어설계및실습
Journal of Digital Contents Society
Vol. 23, No. 8, pp. 1349-1359, Aug. 2022

딥러닝 기반 비속어 필터링 프로그램

송 채 영[1] · 황 세 영[1] · 최 승 호[2*]
[1]광운대학교 컴퓨터정보공학부 학사과정
[2*]광운대학교 소프트웨어사업단 초빙교수

Bad Word Detection System

Chae-Young Song[1] · Se-Young Hwang[1] · Seoung-Ho Choi[2*]
[1]Bachelor Course, Department of Computer Engineering, Kwangwoon University, Seoul 01867, Kore
[2*]Visting Professor, National Program for Excellence in Software, Kwangwoon University, Seoul 018967, Kwangwoon University, Seoul 018967, Korea

[요 약]

각종 SNS, 게임 내에서는 비속어가 무분별하게 사용되고 있다. 이는 사이버 폭력으로도 이어져 이와 관련한 사이버 폭력 사례들이 급증하고 이를 문제로 삼아 사람들이 비속어 필터링과 관련된 프로그램을 만들고 있으나, 문맥상 욕설을 의도하지 않음에도 불고하고 욕설로 인지하여 채팅에 불편함을 겪게 되는 점을 문제로 삼아 이를 개선하는 프로그램을 제안한다. 각종 사이트에서 비속어를 추출하여 데이터 셋을 모은다. 이때, 예외처리로 비속어로 분류될 수 있으나 비속어가 아닌 데이터를 찾아준다. 이를 바탕으로 Fasttext 기법으로 임베딩 후 비속어와의 유사도를 추출해내도록 해준다. 벡터화 된 데이터는 CNN 모델에 넣어 학습시키며 비속어와 비속어가 아닌 단어들을 분류해주도록 해준다.

[Abstract]

Bad Slang is indiscriminately used in various SNS and games. This also leads to cyber violence, which leads to a surge in related cyber violence cases, and people are making programs related to slang filtering, but we propose a program to improve it by taking it as a problem because it is perceived as abusive even though it is not intended in the context. It collects data sets by extracting slang from various sites. In this case, it may be classified as a slang as an exception processing, but data that is not a slang is found. Based on this, the Fasttext technique allows the extraction of similarities with slang after embedding. The vectorized data is learned by putting it in a CNN model, and allows it to classify slang and non-sabotage words.

색인어 : 비속어, 비속어 유사도 분석

Keyword : Bad slang , Analysis of similarities between slang

http://dx.doi.org/10.9728/dcs.2022.23.1.1 (작성 금지)

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-CommercialLicense(http://creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Received 30 November 2022; **Revised** 30 November 2022
Accepted 05 November 2022 (작성 금지)

***Corresponding Author; Chae-Young Song, Se-Young Hwang**

Tel: +82-4842-3930, +82-4177-1429
E-mail: dkdl0806@naver.com, tpdudabc@naver.com

Ⅰ. 서 론

각종 SNS, 게임 내에서는 비속어가 무분별하게 사용되고 있다. 스마트폰 이용이 당연해지면서 많은 사람들이 인스타 그램, 페이스북과 같은 SNS를 사용하게 됐는데 이러한 SNS 댓글창을 살펴보면 비속어를 쉽게 발견할 수 있다. 이는 악성 댓글, 소위 악플이라고 하는 사이버 폭력으로도 이어지며 이로 인해 명예 훼손, 모욕죄 등과 같은 문제들이 시간이 지날수록 급증하고 있음을 확인하였다.

표 1을 살펴보면 2021년도에 사이버 명예훼손 부분에서 사이버 폭력 피해 경험률을 확인할 수 있는데, 1년에 한두 번 이상은 50프로가 넘는 인터넷 이용자가 사이버 명예훼손을 경험해 보았다고 답변하였다.

표 1. 2021년도 사이버 명예훼손 피해 경험률(출처 : 통계청)

Table 1. Experience rate of cyber defamation in 2021

(단위: %)

특성별(1)	특성별(2)	2021				
		1년에 한두 번	6개월에 한두 번	한 달에 한두 번	일주일에 한두 번	거의 매일
전체	소계	51.1	26.7	8.2	7.3	6.7
성별	남성	54.0	24.5	8.2	6.3	7.0
	여성	48.4	28.6	8.3	8.3	6.4
연령별	20~29세	49.9	28.6	5.1	7.7	8.5
	30~39세	49.2	25.4	10.5	10.2	4.8
	40~49세	54.9	23.5	18.2	3.4	-
	50~59세	49.1	25.4	6.2	7.6	11.8
	60~69세	58.4	28.0	4.5	4.2	4.9
학력별	초졸이하	100.0	-	-	-	-
	중졸	62.4	37.6	-	-	-
	고졸	57.8	21.8	6.6	5.1	8.7
	대졸이상	47.4	28.9	9.2	8.6	5.9

이를 문제로 삼아 예방하고자 사람들이 비속어 필터링과 관련된 프로그램을 만들어지고 있으며, 문장에 욕설이 포함되어 있으면 해당 단어를 “ * “과 같은 특수 언어로 마스킹 처리를 하거나 욕설 사용으로 경고를 주는 등 여러 프로그램들이 만들어져 적용되고 있다.

하지만 문맥상 욕설을 의도하지 않음에도 불구하고 욕설로 인지하여 마스킹 처리와 경고를 먹는 등 채팅에 불편함을 겪게 되는 점이 생겼다. 그림 1와 같이 게임 내에서 문맥상 욕설을 의도하지 않은 문장인 “아저씨 발냄새 나요”와 같은 채팅을 쳤을 때에도 욕설로 판단하며 그림 2에서 보이는 것과 같이 이를 마스킹 처리를 하여 “아저**냄새 나요” 처럼 출력이 되는 것을 확인할 수 있다.

이를 문제로 삼아 이를 개선하는 프로그램을 제안한다. “아저 씨발 냄새 나요”는 욕설을 의도한 것일 수도 있으나 “아저씨 발냄새 나요”같은 경우는 욕설을 의도하지 않은 문장으로 분류되어야 한다고 판단하였다.

사용자가 입력한 문장의 의도에 따라 비속어 사용을 의도한 것인지 아닌지를 판별하여 비속어로 판단되면 마스킹 처리가 필요하다. 같은 단어임에도 의도가 다를 수 있으므로 이를 예측하고 판단한다는 점에서 사용자의 불편함을 줄여줄 수 있다. 따라서 본 논문에서의 공헌도는 아래와 같다.

첫 번째, 문장에서 비속어와 비속어가 아닌 단어들을 분류할 수 있는 시스템을 제안한다.

두 번째, 비속어로 분류될 수 있으나 비속어가 아닌 데이터를 찾아준다.

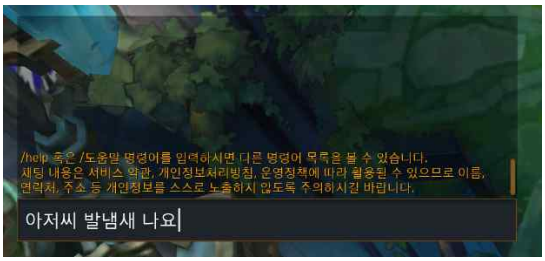


그림 1. 욕설을 의도하지 않은 문장
(사용 환경 : league of legends)

Fig 1. an unintended sentence

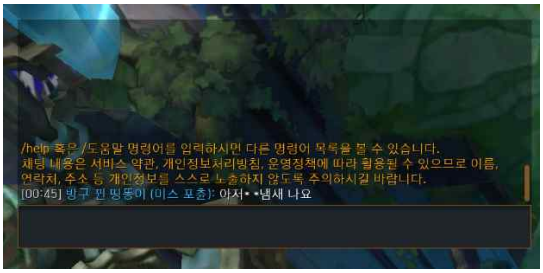


그림 2. 비속어 필터링 시 문제

Fig 2. Problems with filtering slang

Ⅱ. 관련 연구

1) 단어 자모 분리

단어를 분석하기 위해선 단어를 자음 모음을 분리한 형태로 나누어 분류해야 한다. 각 단어를 n-gram하여 임베딩을 해주는데 한글 같은 경우에는 음절단위로 나누어져 의미를 제대로 담지 못하기 때문에 자모 분리를 해주어야 할 필요성이 있다. 여기서 형태소를 기준으로 하여 분리를 하지 않았는데, 이는 비속어는 형태소가 적절하게 분리되지 않는 것이 비중을 많이 차지하기에 어절 단위로 적용해주었다. 단어 임베딩은 좌우 단어를 보고 임베딩하므로 전체 단어를 넣어 돌려준다. 단어의 자모 분리는 fasttext를 활용해야 하기에 입력값을 만들어주기 위하여 사용해준다.

2) Fasttext, Word2Vec

단어를 딥러닝 모델의 입력값으로 넣어주기 위해선 데이터를 벡터와 같은 수치형 데이터로 바꿔줘야한다. 문장을 벡터로 표현하는 방법 중에 하나로 Word2Vec을 예로 들 수 있는데, 이는 원래의 문장을 입력값으로 받아서 그 문장에서 단어의 위치로부터 의미와 유사도를 좌표축상에 나타내고 이를 기반으로 단어를 임베딩하고 단어 간의 거리를 통해 유사도를 구할 수 있다. 이의 특성으로 문장이 입력값으로 들어갈 때, 좋은 성능이 나오고 문장이 아닌 단순 쉼표로 나누어진 단어의 경우는 좋은

성능을 기대하기 어렵다. Fasttext는 본질적으로 word2Vec 모델을 확장한 것이지만 단어를 문자의 ngram 조합으로 취급한다. 그래서 한 단어에 대한 벡터는 이들 ngram의 합으로 만들어진다. Fasttext는 모르는 단어 (Out Of Vocabulary, OOV)에 대한 대응으로 학습 후 모든 데이터 셋의 모든 단어의 각 n-gram에 대해 워드 임베딩이 되므로 데이터 셋만 충분하다면 내부 단어를 통해 OOV에 대해서도 다른 단어와의 유사도를 계산할 수 있다.

비속어는 변형된 단어들이 많아 OOV를 해결하기 위해 char 단위로 학습하는 fasttext를 사용한다. 기존 단어 임베딩 모델처럼 같은 단어라도 좌우 문맥에 따라 의미를 달리하는 경우도 뽑기 위해 사용한다.

3) purifier model

purifier 모델은 입력된 문장 내에 욕설이 있는 지 없는 지에 대한 classification만 가능했다. 이 자체만으로도 문맥적인 욕설을 잡아낼 수 있다는 점에서 기존 rule-based model 모다 우세하다 할 수 있지만, 문장 내 모든 단어를 순차적으로 조합하는 캐스캐이드 방식으로 마스킹 알고리즘을 구현할 경우 2의 n 승 번의 예측이 필요하여 욕설이 있는 부분을 찾아 마스킹하는 데에 부적합하다.

이에 모델이 classification을 할 때 어떤 위치 혹은 정보를 기반으로 판단을 하는지를 알아내고 파악된 위치를 마스킹하는 방식으로 한다. 주어진 Query(Q)에 대해서 모든 Key(K)와의 유사도를 구하는 attention 함수는 이 유사도를 가중치로 하여 각각의 Value(V)에 반영해 준다. BERT는 Query, Key, Value가 모두 동일한 셀프 어텐션을 사용하고 있으므로 이는 입력 문장의 모든 단어 벡터들이 서로를 바라보고 서로를 반영한다고 할 수 있다

.BERT는 classification에 pooler를 통과한 CLS 토큰만을 사용한다. 이는 임베딩 완료된 문장이 12개의 어텐션 레이어를 통과하는 동안 문장의 앞뒤 문맥에 대한 정보가 CLS 토큰에 담기게 되기 때문이다.

Puri attention layer의 첫 번째 핵심은 모든 문맥 정보를 담고있는 CLS 토큰으로 아직 문맥 정보들이 뒤섞이지 않은 유사도를 구하게 하는 데에 있다.

Query는 CLS 토큰, Key와 Vaule는 임베딩 출력이 된다는 뜻이다. 이렇게 나온 각 단어 토큰마다의 어텐션 프로브(AP) 를 비교하여 일정 이상의 확률일 경우 욕설로 판단하고 해당 단어를 마스킹한다.

Purifier model은 퓨리 어텐션을 통해 fine tunning 동안 CLS 토큰과 임베딩 처리된 입력 문장의 유사도를 계산하여 그중 값이 높은 토큰을 욕설로 학습해 나간다고 정리할 수 있다.

4) N-gram

한국어의 일반적인 띄어쓰기 규칙과 달리 자동 띄어쓰기는 간단한 문제가 아니다. 잘못된 띄어쓰기는 형태소 분석에서 치명적인 오류를 불러오는 결과를 발생시킬 수 있다. N-gram 언어 모델은 SLM 의 일종으로 카운트에 기반한 통계적 접근을 사용하고 있다. 이는 등장한 모든 단어를 고려하지 않고 일부 단어만을 고려하는 방법을 사용하는데, 이 일부 단어가 몇 개이냐에 따라 n이 결정 된다.

N은 연속적인 단어 나열을 의미하며, 단어를 묶음 단위로 스플릿 하여 이를 하나의 토큰으로 다룬다. N-gram은 주어진 문장에서의 다음에 나올 단어를 예측하고 싶을 때, 이를 이용한 언어 모델을 사용한다. 하지만 N-gram은 앞의 몇 개의 단어만을 보다 보니 어쩔 수 없는 한계가 발생한다. 의도하고 싶은 문장의 끝맺음을 하지 못하는 경우도 생겨 전체 문장을 고려한 언어 모델보다는 정확도가 떨어진다는 점이다.

Ⅲ. 제안방법

각종 데이터를 모으기 위하여 에브리타임, 인스타 그램, 디씨인사이드, 유튜브, 뉴스 내의 댓글에서 비속어와 비속어가 아닌 데이터를 직접 하나하나 수작업을 통해 얻어주었다. 사이트에서 비속어를 포함한 단어를 모은 후 csv 파일에 저장해준다. 이외에도 네이버 금칙어와 인스티즈 금칙어를 데이터 셋에 저장해준다. 비속어로 분류될 수 있으나 비속어가 아닌 단어나 문장을 예외처리 해주기 위해 그에 맞는 데이터도 찾아준다. 이에 대한 예시로는 “무지개 같은 사장님”이 있다. 이 문장은 “무지 개 같은 사장님”으로 인식하여 욕설로 분류될 가능성이 있음을 알 수 있다. 이 문장을 어떻게 읽느냐에 따라 무지개 같은 사장님이 될 수도, 무지 개 같은 사장님이 될 수도 있다.

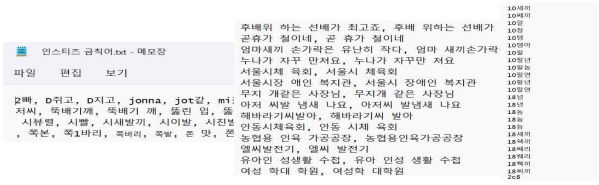


그림 3. 데이터

Fig.3 Data

csv파일

	A
1	10새끼
2	10쎄끼
3	10알
4	10창
5	10탱
6	10탱아
7	10팔
8	10팔년
9	10팔놈
10	10팔연
11	10할년
12	10할연

1505	후배 위하는 선배가 최고죠
1506	곧 휴가 철이네
1507	엄마 새끼손가락은 유난히 작다
1508	누나가 자꾸만 져요
1509	서울시 체육회
1510	서울시 장애인 복지관
1511	무지개 같은 사장님
1512	아저씨 발냄새 나요
1513	해바라기씨 발아
1514	안동시 체육회
1515	엘씨 발전기
1516	유아 인성 생활 수첩
1517	여성학 대학원
1518	의뢰인을 내 가족같이 모시겠습니

그림 4. 데이터 csv파일로 변환

Fig.4 Convert data to csv file

Fasttext를 이용하여 embedding 기법을 단어에 적용해준다. Jamosplit 모듈을 가져와 그림 8처럼 자모분리를 해준다. (그림 7 참고) 이는 각 단어를 n-gram하여 임베딩하는데 한글 같은 경우는 음절 단위로 나누어져 의미를 제대로 못 담기에 적용해 준 것이다. 형태소 단위로 분리를 하지 않은 이유는 embedding 해줄 단어들이 비속어이기에 "ㅋㅋ ㅇㅈ ㅗ" 과 같이 형태소가 적절하게 분리되지 않는 것이 많은 비중을 차지하고 있어 그림 6과 같이 어절을 단위로 하여 사용하였다. 단어 임베딩은 한 단어의 좌우 단어를 보고 임베딩하므로 전체 단어를 넣은 후에 돌린다. Fasttext는 본질적으로 word2Vec 모델을 확장한 것이지만 단어를 문자의 ngram 조합으로 취급한다. 그래서 한 단어에 대한 벡터는 이들 ngram의 합으로 만들어진다. Fasttext는 모르는 단어(Out Of Vocabulary, OOV)에 대한 대응으로 학습 후 모든 데이터 셋의 모든 단어의 각 n-gram에 대해 워드 임베딩이 되므로 데이터 셋만 충분하다면 내부 단어를 통해 OOV에 대해서도 다른 단어와의 유사도를 계산할 수 있다.

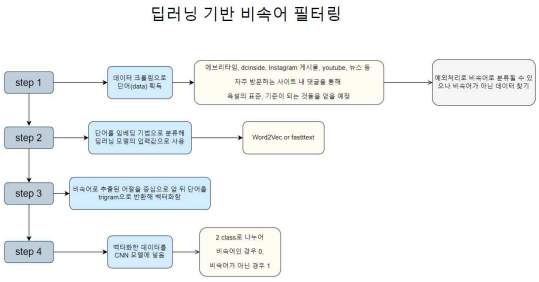


그림 5. 시스템 구성도

Fig.5 System Configuration

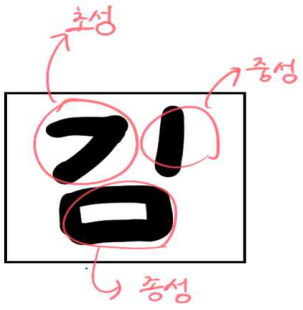


그림 6. 어절 분류 기준

Fig. 6 Word classification criteria

```
CHOSUNGS = [u'ㄱ',u'ㄲ',u'ㄴ',u'ㄷ',u'ㄸ',u'ㄹ'
JOONGSUNGS = [u'ㅏ',u'ㅐ',u'ㅑ',u'ㅒ',u'ㅓ',u'
JONGSUNGS = [u'_',u'ㄱ',u'ㄲ',u'ㄳ',u'ㄴ',u'ㄵ'
TOTAL = CHOSUNGS + JOONGSUNGS + JONGSUNGS
```

그림 7. 자모 분리 예시 코드

Fig. 7 Jamo Isolation Example Code

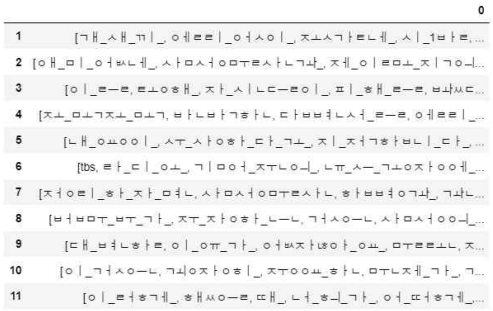


그림 8. 자모 분리

Fig.8 Consonant vowel separation

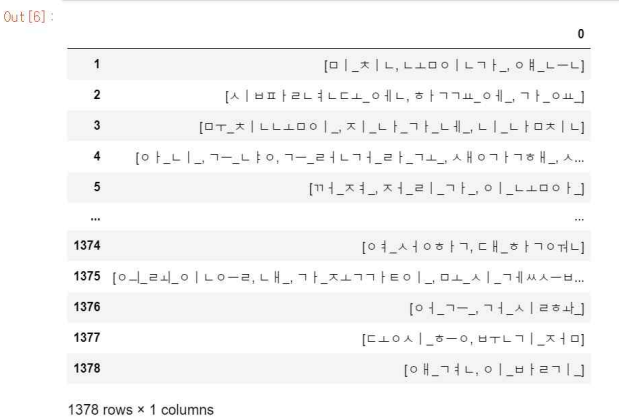
비속어는 변형된 단어들이 많아 OOV를 해결하기 위해 char 단위로 학습하는 fasttext를 사용한다. 기존 단어 임베딩 모델처럼 같은 단어라도 좌우 문맥에 따라 의미를 달리하는 경우도 뽑기 위해 사용한다.

벡터화한 데이터는 Random Foreset 1D CNN 모델의 입력값으로 사용해준다. 벡터화된 데이터를 불러와 train data와 test data를 분리시켜 준다. 그리고 데이터를 keras data 형식에 맞게 변환 시켜준 후 실행시킨다.

Ⅳ. 실험결과

1) 자모 분리 형태

문장을 단위로 하여 자음 모음을 분리한 데이터들이 만들어 졌다.



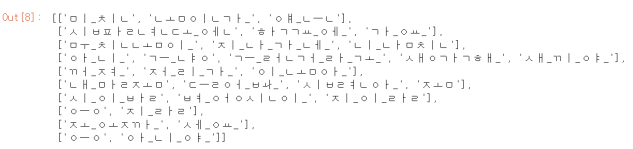
Out [6]:

	0
1	[ㅁㅣ_ㅊㅣㄴ, ㄴㅗㅁㅇㅣㄴㄱㅏ_, ㅇㅐ_ㄴㅡㄴ]
2	[ㅅㅣㅂㅍㅏㄹㄴㅓㄷㅗ_ㅇㅔㄴ, ㅎㅏㄱㄱㅗ_ㅇㅔ_, ㄱㅏ_ㅇㅛ_]
3	[ㅁㅜ_ㅊㅣㄴㄴㅗㅁㅇㅣ_, ㅈㅣ_ㄴㅏ_ㄱㅏ_ㄴㅔ_, ㄴㅣ_ㄴㅏㅁㅊㅣㄴ]
4	[ㅇㅏ_ㄴㅣ_, ㄱㅡ_ㄴㅑㅇ, ㄱㅡ_ㄹㅓㄴㄱㅓ_ㄹㅏ_ㄱㅗ_, ㅅㅐㅇㅏㄱㅎㅐ_, ㅅ...
5	[ㄲㅓ_ㅈㅓ_, ㅈㅓ_ㄹㅣ_ㄱㅏ_, ㅇㅣ_ㄴㅗㅁㅇㅏ_]
...	...
1374	[ㅇㅕ_ㅅㅓㅇㅎㅏㄱ, ㄷㅐ_ㅎㅏㄱㅇㅝㄴ]
1375	[ㅇㅣ_ㄹㅐ_ㅇㅣㄴㅇㅡㄹ, ㄴㅐ_, ㄱㅏ_ㅈㅗㄱㄱㅏㅌㅇㅣ_, ㅁㅗ_ㅅㅣ_ㄱㅔㅆㅅㅡㅂ...
1376	[ㅇㅓ_ㄱㅡ_, ㄱㅓ_ㅅㅣㄹㅎㅐ_]
1377	[ㄷㅗㅇㅅㅣ_ㅎㅡㅇ, ㅂㅜㄴㄱㅣ_ㅈㅓㅁ]
1378	[ㅇㅐ_ㄱㅓㄴ, ㅇㅣ_ㅂㅏㄹㄱㅣ_]

1378 rows × 1 columns

그림 9. 자모 분리

Fig.9 Consonant vowel separation



Out [8]:

```
[['ㅁㅣ_ㅊㅣㄴ', 'ㄴㅗㅁㅇㅣㄴㄱㅏ_', 'ㅇㅐ_ㄴㅡㄴ'],
 ['ㅅㅣㅂㅍㅏㄹㄴㅓㄷㅗ_ㅇㅔㄴ', 'ㅎㅏㄱㄱㅗ_ㅇㅔ_', 'ㄱㅏ_ㅇㅛ_'],
 ['ㅁㅜ_ㅊㅣㄴㄴㅗㅁㅇㅣ_', 'ㅈㅣ_ㄴㅏ_ㄱㅏ_ㄴㅔ_', 'ㄴㅣ_ㄴㅏㅁㅊㅣㄴ'],
 ['ㅇㅏ_ㄴㅣ_', 'ㄱㅡ_ㄴㅑㅇ', 'ㄱㅡ_ㄹㅓㄴㄱㅓ_ㄹㅏ_ㄱㅗ_', 'ㅅㅐㅇㅏㄱㅎㅐ_', 'ㅅㅐ_ㄱㅣ_ㅇㅑ_'],
 ['ㄲㅓ_ㅈㅓ_', 'ㅈㅓ_ㄹㅣ_ㄱㅏ_', 'ㅇㅣ_ㄴㅗㅁㅇㅏ_'],
 ['ㄴㅐ_ㅁㅏㄹㅈㅗㅁ', 'ㄷㅡㄹㅇㅓ_ㅂㅆㅏ_', 'ㅅㅣㅂㄹㅓㄴㅇㅏ_', 'ㅈㅗㅁ'],
 ['ㅅㅣ_ㅇㅣ_ㅂㅏㄹ', 'ㅂㅓ_ㅇㅣㅇㅅㅣㄴㅇㅣ_', 'ㅈㅣ_ㅇㅣ_ㄹㅏㄹ'],
 ['ㅇㅡㅇ', 'ㅈㅣ_ㄹㅏㄹ'],
 ['ㅈㅗ_ㅇㅗㅈㅋㅏ_', 'ㅅㅐ_ㅇㅛ_'],
 ['ㅇㅡㅇ', 'ㅇㅏ_ㄴㅣ_ㅇㅑ_']]
```

그림 10. Fasttext 입력 데이터

Fig.10 Fasttext input data

2) 유사 형태 단어 출력

입력으로 단어를 넣어주면 유사한 형태를 가진 단어들이 출력되는 것을 확인할 수 있다.

```
model.wv.most_similar(jamo_split("개새끼"))
```

```
Out[10]: [('ㄱㅐ_ㅅㅐ_ㄲㅣ_', 0.8398352265357971),
 ('ㄱㅐ_ㅅㅐ_', 0.7305082082748413),
 ('ㄱㅐ_ㅅㅐ_ㄲㅏ_', 0.7271333336830139),
 ('ㅈㅗㅈㅅㅐ_ㄲㅣ_', 0.7021294832229614),
 ('ㅎㅗ_ㄹㅗ_ㅅㅐ_ㄲㅣ_', 0.697624146938324),
 ('ㄱㅐ_ㅎㅜ_ㄹㅏ_ㅅㅐ_ㄲㅣ_', 0.6936761140823364),
 ('ㄱㅐ_ㄸㅐ_ㄲㅣ_', 0.6807937026023865),
 ('ㄱㅐ_ㅅㅐㅅ', 0.6679026484489441),
 ('ㄷㅗㄴㅅㅐ_ㄲㅣ_', 0.6532419323921204),
 ('ㄱㅡ_ㅈㅣ_ㅅㅐ_ㄲㅣ_', 0.6515879034996033)]
```

그림 11. 유사 단어 출력

Fig 11. Similar Word Output

3) 데이터 분포

단어 데이터를 plot으로 나타낸 것이며 오른쪽 위일수록 비속어 단어를, 오른쪽 아래일수록 자주 나오지 않는 단어를 나타낸다. 그림 12와 같이 데이터의 분포가 출력되어야 하지만, 그래프가 출력되지 않아 기대한 예시 그래프를 첨부하였다.

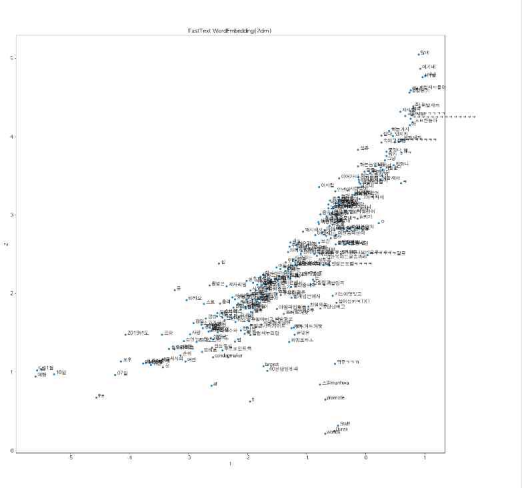


그림 12. 데이터 plot

Fig.12 Data plot

넣어준 단어들을 욕설인 단어는 1로 욕설이 아닌 단어는 0으로 라벨링 후에 그래프를 출력해주었다. 그림 13에서 볼 수 있듯, 욕설인 단어는 bad word로 욕설이 아닌 단어는 not bad word로 분류되어 있다.

```
In [20]: plt.bar(['bad word', 'not bad word'], data['label'].value_counts(), width=0.7)
         print(data['label'].value_counts())
         # 약 7대1의 비율
```

```
1    1464
0     127
Name: label, dtype: int64
```

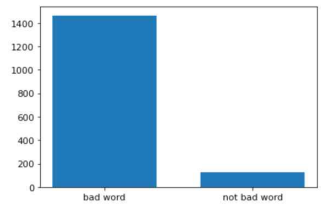


그림 13. 데이터 분포

Fig.13 Data Distribution

비속어 단어 데이터의 분포를 그래프로 나타내어 시각화하였다. 데이터의 시각화 과정에서 data count의 수정 부분에서 오류가 나 빈도수의 차이를 눈에 띄게 나타내지는 못하였다. 또한 한글이 깨져 어떤 단어의 빈도수를 나타낸 것인지 알 수 없다.

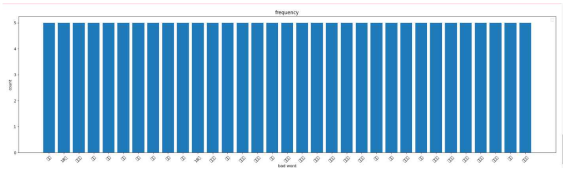


그림 14. 비속어 데이터 분포

Fig.14 Slang data distribution

4) 비속어 탐지

문장을 입력 값으로 넣어주면 비속어로 탐지된 부분이 *로 마스킹 처리되어 출력된다. 서론에서 언급한 “아저씨 발냄새 나요”와 같은 비속어를 의도하지 않으나 비속어로 분류되던 문장은 걸러지지 않고 알맞게 출력된다. “시1발”과 같은 욕설 감지를 피해가기 위한 단어 또한 비속어로 분류한다.

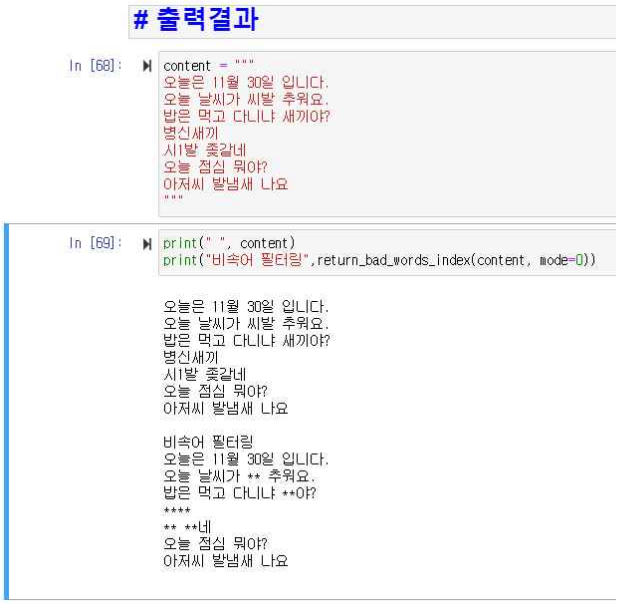


그림 15. 마스킹 처리

Fig 15. Masking treatment

그림 16. 마스킹 처리

Fig 16. Masking treatment

V. 결 론

욕설과 같은 비속어를 판별해내는 프로그램의 동작을 보면서 아쉬운 점들이 많았다. 아쉬움을 느낀 부분들을 보완하기 위해 비속어를 의도하지 않은 문장을 고려하도록 시스템을 제안했다. 욕설들을 masking 처리하며 사이버 폭력을 줄이는 것에 도움이 되고 욕설 사용을 의도하지 않고 채팅을 보냈으나 욕설로 처리되어 경고를 먹는 억울한 상황들을 줄일 것을 기대한다.

시스템을 만들고 간단하게라도 채팅창을 만들어내어 상황을 재현해내는 것까지 목표로 잡았으나 처음 접하는 개념들이 대부분이라 이해하고 코드를 해석하고 응용해내는 것에 있어서 시간이 부족하고 어려움을 많이 겪었다. 처음에 목표로 잡았던 것을 완벽하게 해내지 못했다는 아쉬움을 느꼈으나 추후에는 채팅 프로그램에 적용해보는 시스템을 만들어보고 다뤄보고자 한다.

참고문헌

[1] Y. T. Yoon " Internet Vulgarity Detecting System by Using Algorithm to Transform Modified Vulgar words to Basic Type

[2] G. H. Lee, ". Design and Implementation of Profanity Filtering Chat Program Based on Deep Learning ", Dept of Computer Engineering, Graduate School of KoreaTech University,Columbus,2019.

[3] Ente, L e e, “Project-Purifier” https://url.kr/xwmrya

[4] Avidale, “Compress-fastText”Dec 14, 2021. https://github.com/avidale/compress-fasttext

[5] T. J . Yoon, H.G. Hwan, “The Online Game Coined Profanity Filtering System by using Semi-Global Alignment”, in *Busan University*, Oct 30, 2009. https://url.kr/vg69ad

송채영(Chae-Young Song)

2021년~ 현재
: 광운대학교 컴퓨터정보공학부 학부과정

황세영(Se-Young Hwang)

2021년~ 현재
: 광운대학교 컴퓨터정보공학부 학부과정

최승호(Seoung-Ho Choi)

2018년 : 한성대학교 전자정보공학과 (공학사)
2020년 : 한성대학교 대학원 (공학석사)
2021년~현 재: 한성대학교 기초교양학부 시간강사
2022년~현 재: 숭실대학교 컴퓨터공학과 시간강사
2022년~현 재: 광운대학교 SW중심대학사업단 초빙교수
※관심분야 : 딥러닝, 컴퓨터 비전등