TITLE

NLP PROJECT

Prj name: Naive Bayeji

5조

김예지 이기중 정하윤 최재철

CONTENTS

TABLE OF CONTENTS

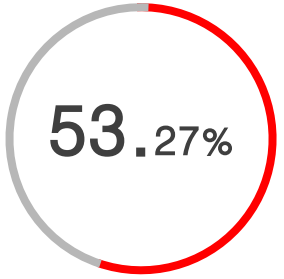
Machine Learning: NLP Project

프로젝트 목적

'건설적인 댓글이 많아져서 클린봇이 할 일이 점점 적어졌으면 좋겠다'

'나는 좀 더 아름다운 세상을 만들려고 노력하는 플랫폼 개발자'

-네이버 클린봇 개발자 이규호 리더



Prj Kaggle 최종 스코어

Bert Baseline : 52.5%

CONTENTS

개요

배경 및 목적

심각한 사회 문제로 대두되고 있는 온라인 '악성 댓글'

"삼류 XXX 같은 애. 덜 떨어진 X. 일부러 벗네
"그냥 노출증 환자."
"완전 노리고 했던데?ㅋㅋ"
"일부러 저러는 듯."

[줌인]영혼까지 갉아먹는 `악플`, 누가 왜 다는 것일까

악성루머·댓글에 활동 쉬었던 설리, 끝내 극단적 선택
대부분 평범한 악플러...다만 상대방 감정에 무관심해

Newsprime

[10년 전 오늘] 통신첨단국의 어두운 그림자 살인
이젠 ...

이러한 악플로 인한 피해는 언론을 통해 쉽게 접할 수 있습니다. 연예인들이 악플로 인해 우울증에 시달리다 극단적인 선택을 하는 것은 물론, 일반

[MT리포트] 설리도 최진실도 종현도...악플에 무너진 그들

[영상] "성희롱 등 댓글 내용 매우 심각"...악플러 고소한 김연경 소속사

먼트사인 라이언앳은 25일 "김연경에 관한 악성 댓글 등을 작
및 모욕죄에 해당하는 행위를 한 소위 '악플러'들을 고소 ...

"연예인들 자살 사건에 악플이 영향 미쳤다", 98%

한국언론진흥재단, "댓글 폐지, 실검 폐지에 대한 국민 인식" 발간
"다음 연예뉴스 댓글 폐지...지지한다", 80.8%
실시간 검색어 폐지, '지지한다' 46.7%, '반대한다' 26.8%, '관심 없다' 26.5

뉴스핌

악플에 쓰러지는 연예인들, 정부는 책임 없나

협회는 지난달 29일 국회와 문화체육관광부, 포털사이트를 향해 "더 이상의 악성 댓글로 인한 피해가 없어야 한다"고 대책 마련을 촉구했다.

2019. 12. 9.

CONTENTS

개요

배경 및 목적

악플을 쓰는 이유

○ 숨어 있는 열등감, 공격본능 자극

악플러들은 일상생활에서 자신이 없고 심리적 열등감으로 위축돼 있는 경우가 많다. 그 때문에 익명성이 보장돼 있는 인터넷 공간에서 마음속 억압된 감정을 발산하면서 순간순간 긴장감과 짜릿한 느낌을 맛보려 하는 것이다.

을지병원 정신과 신홍범 교수는 '자신은 특별하므로 특별한 대우를 받아야 한다고 생각하는 '자기애적 인격 장애'인 사람들이 악플을 달기 쉽다'면서 '이들은 다른 사람을 자주 부러워하며 오만하고 건방진 태도를 지니기 쉬우며 자신의 악플로 상대방이 어떤 느낌을 갖는지에 대해서는 신경을 쓰지 않는다'고 말했다.

SBS '그것이 알고싶다' 방송화면 캡처

CONTENTS

데이터 셋 설명

배경 및 목적

데이터셋을 구성한 사람의 말:

많은 사람들이 알고 있는 것처럼, 최근 악성 댓글로 인한 큰 사건들이 많았습니다. 특히 연예계에 종사하는 분들이 이로 인한 정신적인 피해를 많이 입고 있습니다. 결국 다음에 이어 네이버도 연예 뉴스의 댓글을 폐지하는 방식을 선택하였지만 이는 근본적인 해결이 될 수 없습니다.

이미 twitter, facebook 등의 플랫폼에서는 혐오 발언을 필터링하기 위해 자체적으로 데이터를 구축하고, 그 데이터로 학습한 모델을 활용하고 있습니다. 좋은 데이터가 있다면 이 데이터를 활용한 더 나은 해결책이 있을 것이라고 생각합니다.

인터넷 상에서의 무분별한 악성 댓글로 인한 불쾌감 및 인격적 모독에 대한 피해를 예방하고자 하는 목적으로 기획되었습니다. 추후 혐오 발언을 구분할 수 있는 모델이 개발될 수 있도록 좋은 데이터를 만들고자 합니다.

CONTENTS

데이터 셋 설명

배경 및 목적

- **hate : 글의 대상 또는 관련 인물, 글의 작성자 또는 댓글 등에 대한 강한 혐오 또는 모욕이 코멘트에 표시되었는가?**

comments	title
김종국 아빠 인생최대 행복했던 순간은 근육빵빵 아들내미 군대안갔을때? ㅎㅎ	홍선영 20kg 감량 성공…'미우새' 적수 없는 시청률 1위
조인성 방귀 잘낄거 같아 냄새도 독하다던데	조인성 "5년 전만 해도 교만..행복 뭔지 깨달았다"[화보]
문재인 ㄱH만도 못한것 ㅉㅉ	임청하, 결혼 24년 만에 이혼·위자료 2900억?…직접 입장 밝힐까 [종합]

- **offensive : 비록 논평이 상기만큼 혐오스럽거나 모욕적이지는 않지만, 그것이 대상이나 독자를 불쾌하게 만드는가?**

comments	title
그러게 거미가 어떻게 조정석을 잡았지 거미줄로 잡았나;;	조정석, ♥거미 언급하며 눈물 "뒷바라지해준 지연아, 쑥스럽지만 사랑해" [종합]
한가인 다리인가요? 실하네요	연정훈♥한가인, 귀요미 둘째 발 사진..앙증맞고 깜찍해
어디 여자가 팔에다 문신을 해?	씨잼, '해킹' 아닌 당당한 '럽스타그램' 中 "난 네가 필요해♥" [종합]

- **none : 어떠한 증오나 모욕도 내포하지 않는 논평**

comments	title
내용이 잼없다~~	[종합] '시크릿 마더' 김소연 누가 죽였나…송윤아와 갈등
잘 하실 거예요. 화이팅 입니다~	[종합] "열심히 할게요"…'해투4' 조윤희, MC 데뷔전도 성공(ft.♥이동건)
근래 들어 넘 재밌게 봤고 경치 좋은 하와이 티비에서 불수 있어 좋았어요.	[리뷰IS]'나 혼자 산다' 두 여자의 겨울나기 "힐링과 재미"

CONTENTS

데이터 셋 설명

트레인 데이터 / 검증 데이터 / 테스트 데이터(캐글 제출)

Train data : 7896 rows

	comments	bias_label	gender_labe	hate_label	news_title	comment_pos	title_pos
7891	힘내세요,응원합니다	none	False	none	허지웅 허투루 넘길 말 없었다,솔직하게 드러냈던 속 사정	[('힘내세요', 'Verb'), (',', 'Punctuation'), ('응원'...	[('허지웅', 'Noun'), ('허투루', 'Noun'), ('넘길', 'Ver...
7892	힘내세요,삼가 고인의 명복을 빕니다	none	False	none	이혜경 오 캐롤 공연 중 남편 오정욱 부 고 오열 속 발인 종합	[('힘내세요', 'Verb'), (',', 'Punctuation'), ('삼가'...	[('이혜경', 'Noun'), ('오', 'Noun'), ('캐롤', 'Noun'...
7893	힘내세용 항상 응원합니닷	none	False	none	설경구 송윤아 아들과 즐거운 하루 전 엄마니까요	[('힘내세용', 'Verb'), ('항상', 'Noun'), ('응원', 'Nou...	[('설경구', 'Noun'), ('송윤아', 'Noun'), ('아들', 'Nou...
7894	힘내 소연기로 답해요,나도 53살 인데 이런 일 저런 일 다 있더라구 요인격을 믿습...	none	False	none	SC 현장 연예인 인생 협박 유감 미소 잃은 최민수 보복운전 형의 2차 공판 종합	[('힘내', 'Verb'), ('소', 'Modifier'), ('연기', 'No...	[('SC', 'Alpha'), ('현장', 'Noun'), ('연예인', 'Nou...
7895	힘들면 관뒀어야지 그게 현명한 거다	none	False	none	단독 스태프 사망 사고 서른 이지만 결국 오늘 촬영 취소	[('힘들면', 'Adjective'), ('관', 'Noun'), ('뒀어야지',...	[('단독', 'Noun'), ('스태프', 'Noun'), ('사망', 'Noun...

Validation data : 471 rows

	comments	hate_label
466	지현우 범죄 저지르지 않았나요?	offensive
467	여자인생 망칠 일 있나 ㅋㅋ	hate
468	근데 전라도에서 사고가 났는데 굳이 서울까지 와서 병원에 가는 이유는?	offensive
469	할매젖x, 뱃살x, 몸매 s라인, 유륜은 적당해야됨(너무크거나 너무 작아도 x), ...	hate
470	남자가 잘못한거라면... 반성도 없다면...나였다면 ... 여자처럼 아주 못되게 할...	none

Test data : 974 rows, NO LABEL

	comments
969	대박 게스트... 꼭 봐야징~ 컨셉이 바뀌니깐 재미지넹
970	성형으로 다 뜯어고쳐놓고 예쁜척. 성형 전 니 얼굴 다 알고있다. 순자처럼 된장냄새...
971	분위기는 비슷하다만 전혀다른 전개던데 무슨ㅋㅋㄱ 우리나라사람들은 분위기만 비슷하면 ...
972	입에 손가락이 10개 있으니 징그럽다
973	난 조보아 이뻐서 보는데 백종원 관심무

CONTENTS

최종 목표

배경 및 목적

CONTENTS

TABLE OF CONTENTS

Machine Learning: NLP Project

CONTENTS

데이터 전처리

기본적인 전처리 진행

1. 특수문자 제거

```
import re
def cleanse(text):
    pattern = re.compile(r'\s+')
    text = re.sub(pattern, ' ', text)
    text = re.sub('[^가-힣ㄱ-ㅎㅏ-ㅣa-zA-Z0-9]', ' ', text)
    return text
train['comments'] = train['comments'].apply(cleanse)
```

2. 문장 분리

```
import kss
train['comments'] = train['comments'].apply(kss.split_sentences)
train['comments'] = [','.join(map(str, ls)) for ls in train['comments']]
```

3. 띄어쓰기

```
from pykospacing import spacing
spacing("아버지가방에들어가신다.")
```

```
'아버지가 방에 들어가신다.'
```

4. 중복 제거

```
from soynlp.normalizer import * # 중복 제거
print(repeat_normalize('믿고 있었다구우 쥐엔자아아아아아앙 ㅋㅋㅋㅋㅋㅋ ', num_repeats=2))
```

```
믿고 있었다구우 쥐엔자아아앙 ㅋㅋ
```

*적용 모델에 따라 전처리를 다르게 진행하거나,

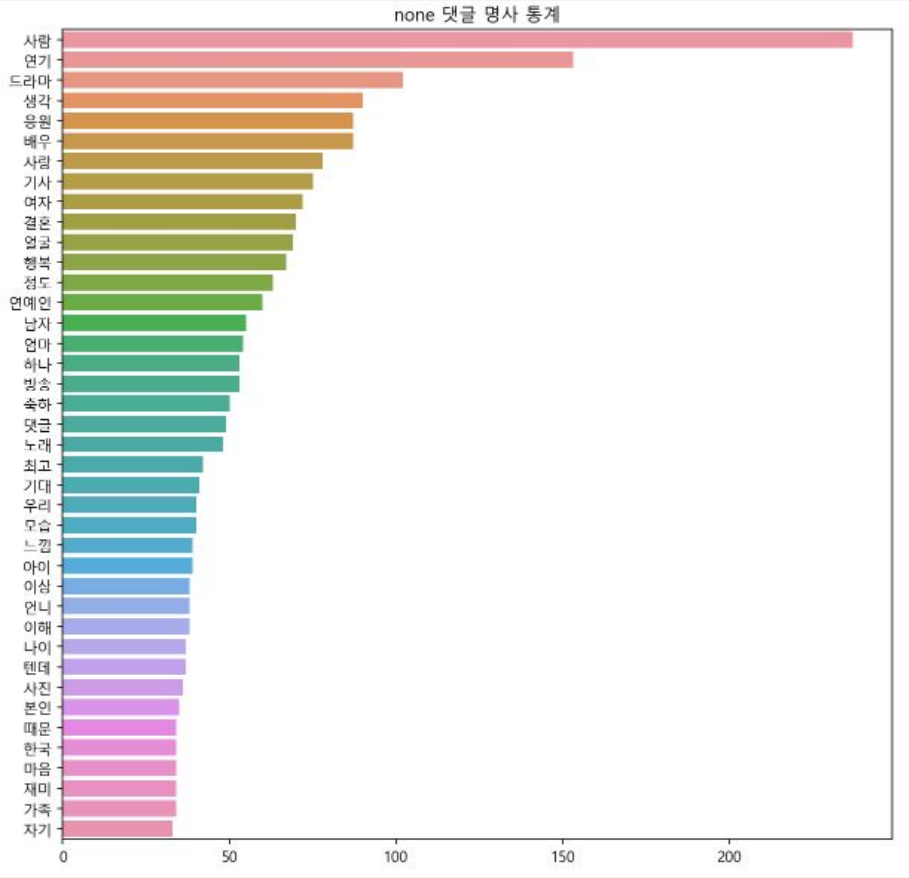
아예 전처리를 하지 않은 부분도 있음

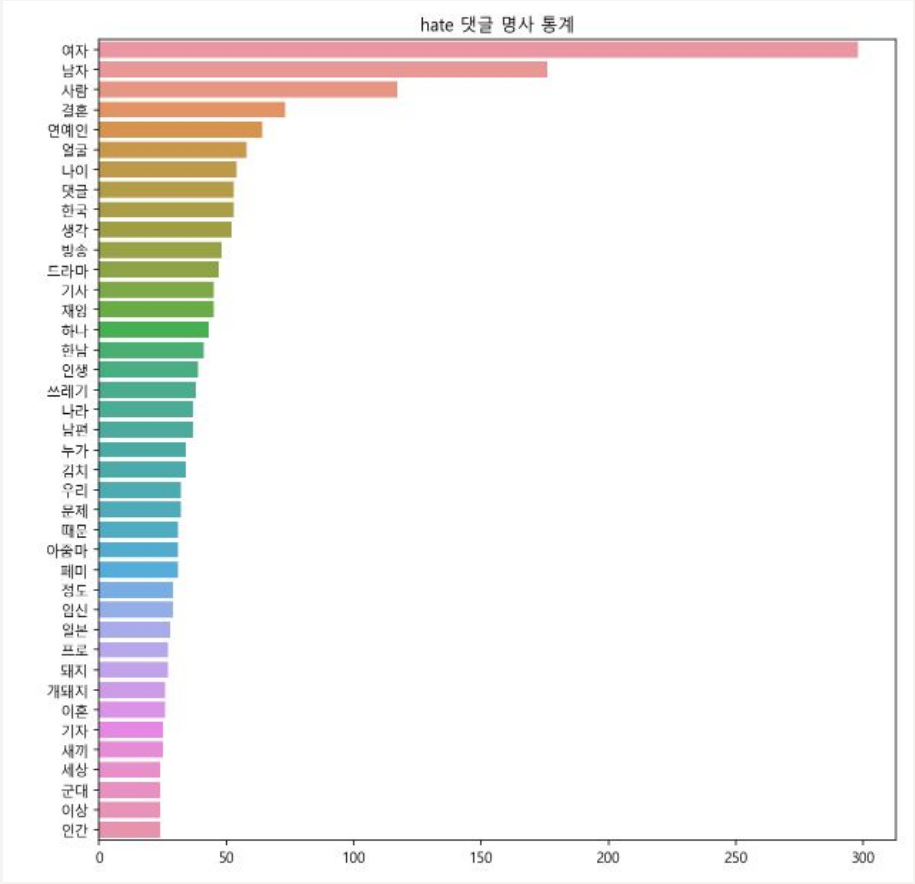
CONTENTS

EDA

2글자 이상 명사 통계

hate 댓글에서 성별, 비속어, 혐오 표현 단어의 빈도가 높음



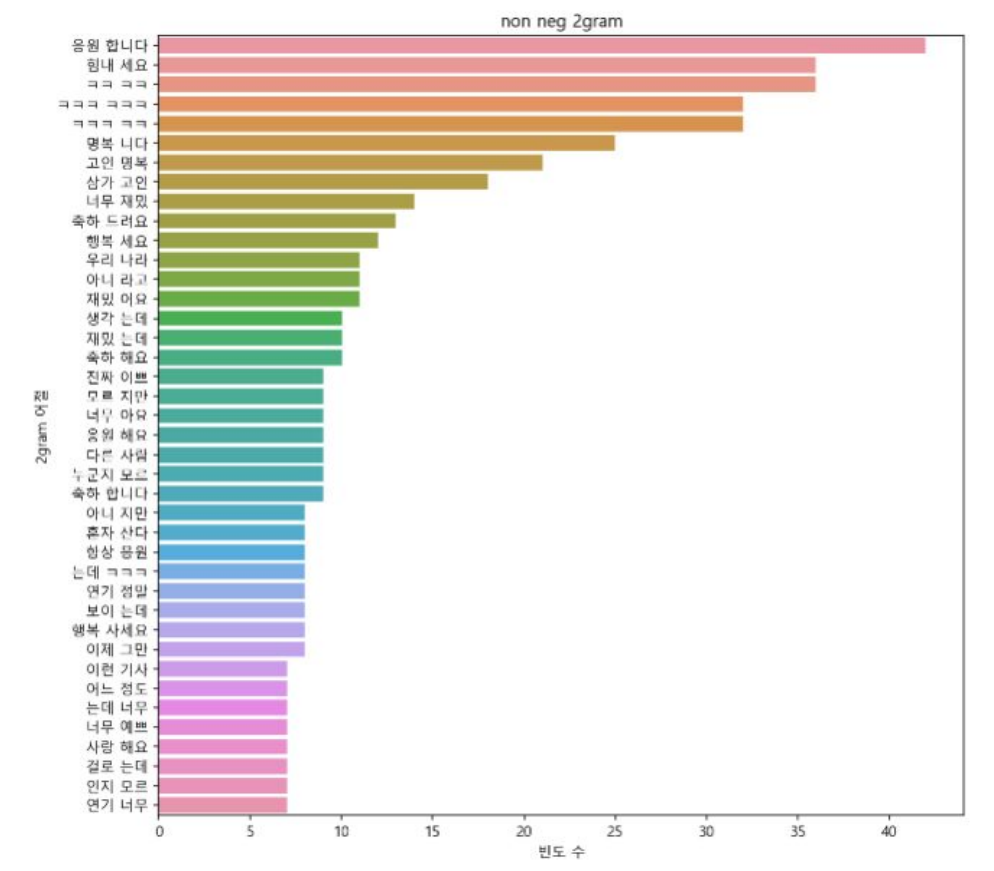


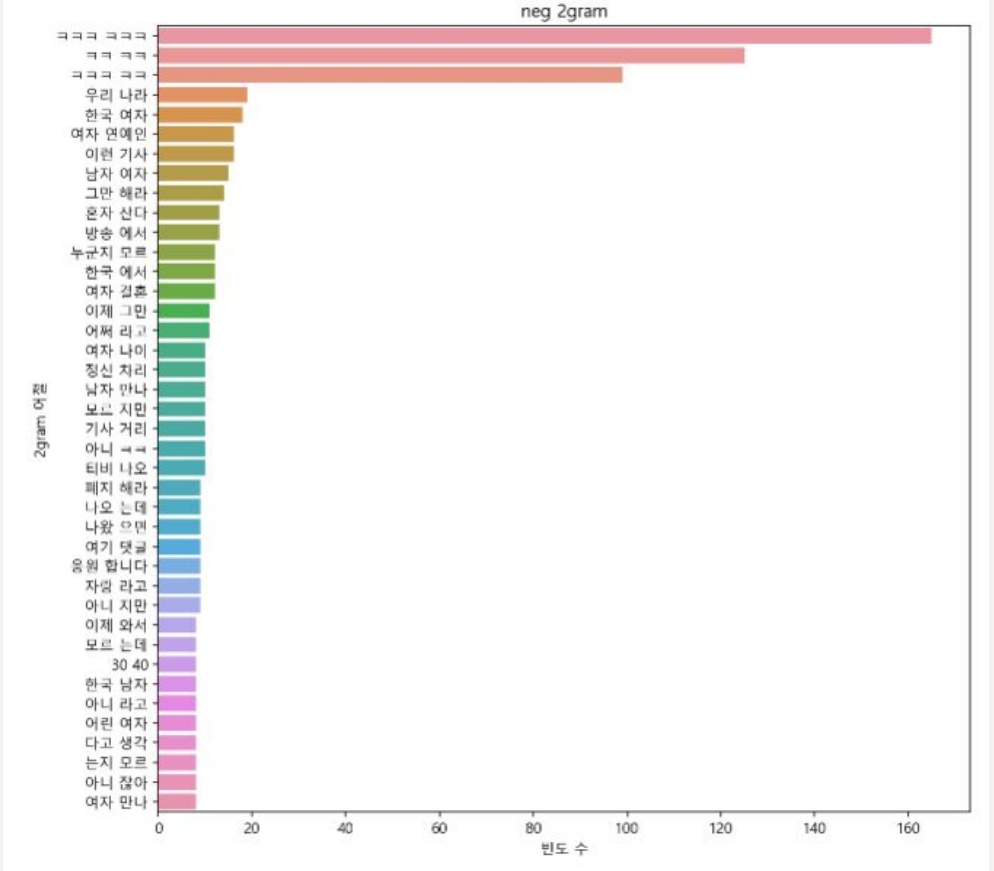
CONTENTS

EDA

형태소 기준 2-gram 통계

hate 댓글에서 성별, 비속어, 혐오 표현 단어의 빈도가 높음



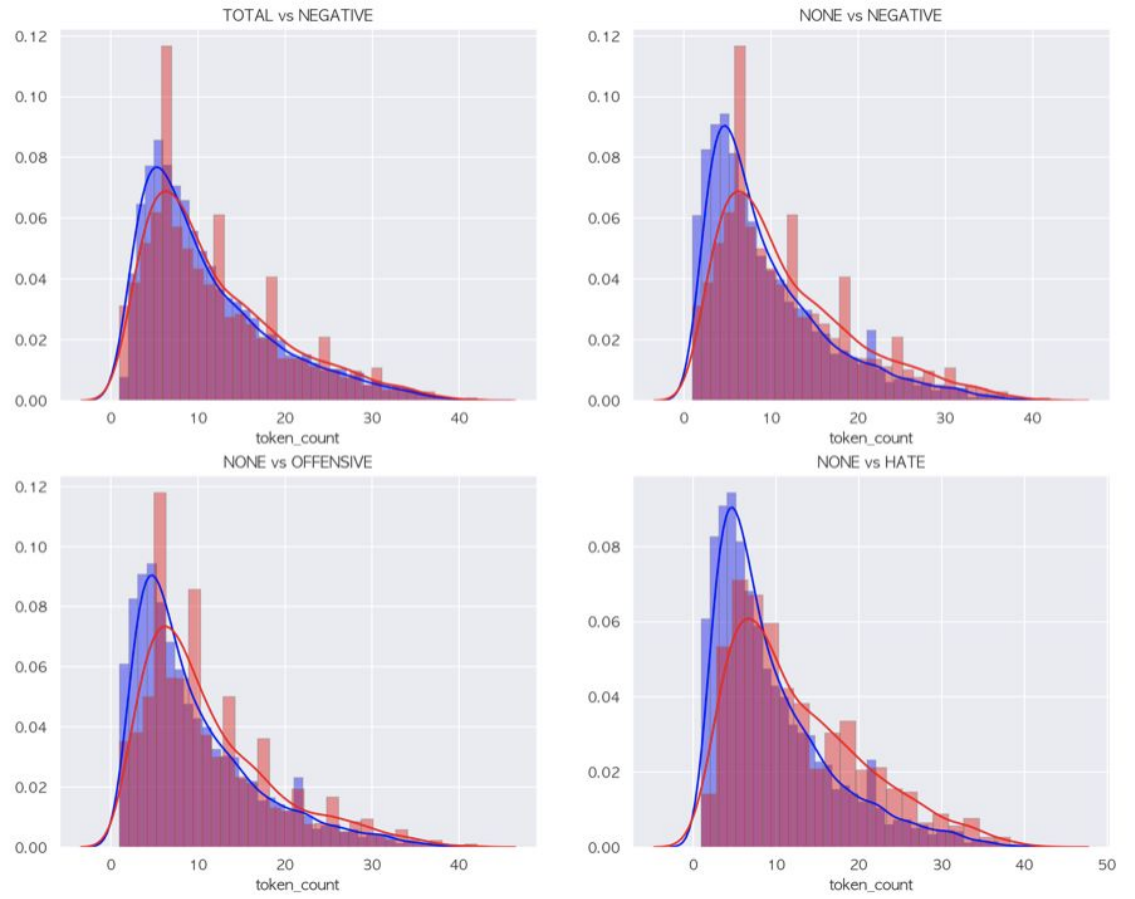


CONTENTS

EDA

형태소 기준 2-gram 통계

부정적 댓글일수록 토큰 수가 많고, 문장의 길이가 김



CONTENTS

Tokenizer

토크나이징 실험

```
models = [LogisticRegression(), RandomForestClassifier(), SVC(), LGBMClassifier()]
tokenizers = [None, k_tokenizer, t_tokenizer, m_tokenizer]
```

```
def get_score(model, tokenizer):
    # 훈련: train 전체 / 테스트: dev 전체
    X_train = train['comments']
    X_test = test['comments']
    y_train = train['label']
    y_test = test['label']

    # Setting up the pipeline
    vec_pipe = Pipeline([
                         ("vec", TfidfVectorizer(tokenizer=tokenizer)),
                         ("model", model)
                         ])

    # Setting the VEC hyperparameters
    vec_pipe_params = {"vec__ngram_range" : [(1,2)],
                       "vec__stop_words"  : [None],
                       "vec__min_df" : [3],
                       "vec__max_df" : [0.9]}

    # Instantiating the grid search
    vec_gs = GridSearchCV(vec_pipe,
                          param_grid=vec_pipe_params,
                          cv=3)

    # Fitting the model to the training data
    vec_gs.fit(X_train, y_train);

    # Predicting
    train_pred = vec_gs.predict(X_train)
    test_pred = vec_gs.predict(X_test)

    # Score
    result = ["train : ", f1_score(train_pred, y_train, average='macro'),
              "test : ", f1_score(test_pred, y_test, average='macro')]

    return result
```

Model : LogisticRegression()

Tokenizer : None
{'train': 0.75610, 'test': 0.41997}
Tokenizer : Khaiii
{'train':0.77985, 'test':0.41844}
Tokenizer : Okt
{'train': 0.83228, 'test': 0.57977}
Tokenizer : Mecab
{'train': 0.84135, 'test': 0.54039}

Model : SVC()

Tokenizer : None
{'train': 0.92808, 'test': 0.38061}
Tokenizer : Khaiii
{'train': 0.94814, 'test': 0.38643}
Tokenizer : Okt
{'train': 0.97399, 'test': 0.53625}
Tokenizer : Mecab
{'train': 0.98057, 'test': 0.52770}

Model : RandomForestClassifier()

Tokenizer : None
{'train': 0.96648, 'test': 0.38967}
Tokenizer : Khaiii
{'train': 0.98436, 'test': 0.34880}
Tokenizer : Okt
{'train': 0.99867, 'test': 0.50862}
Tokenizer : Mecab
{'train': 0.99932, 'test': 0.49737}

Model : LGBMClassifier()

Tokenizer : None
{'train': 0.59843, 'test': 0.34790}
Tokenizer : Khaiii
{'train': 0.65474, 'test': 0.34662}
Tokenizer : Okt
{'train': 0.78796, 'test': 0.54627}
Tokenizer : Mecab
{'train': 0.82722, 'test': 0.50974}

=> 전반적으로 test(검증) 데이터에서 Okt 스코어가 높음

CONTENTS

TABLE OF CONTENTS

Machine Learning: NLP Project

CONTENTS

Count Vectorizer vs. TF-IDF Vectorizer

모델 실험

KFold StratifiedKFold 사용

```
from sklearn.model_selection import KFold
from sklearn.model_selection import StratifiedKFold

kf = KFold(n_splits=10)
skfold = StratifiedKFold(n_splits=10)
```

KFold TF vs CV

```
KFold f1_macro :: TF :  [0.54409835 0.5751475  0.55495904 0.53784724 0.5603153  0.52843219
 0.56598094 0.53796904 0.55379309 0.56143291]

KFold f1_macro :: CV :  [0.54380155 0.54360996 0.53902544 0.53090814 0.54776016 0.52127145
 0.55323764 0.54966987 0.53721223 0.52919632]

KFold f1_macro :: TF :  0.5519975598739248 , CV :  0.5395692760168908
```

StratifiedKFold TF vs CV

```
SKFold f1_macro :: TF :  [0.54163117 0.56082713 0.5771245  0.53130162 0.55013692 0.53510416
 0.58301114 0.54874383 0.54916122 0.5544463 ]

SKFold f1_macro :: CV :  [0.53854502 0.53605738 0.55034544 0.5346379  0.54060879 0.5322481
 0.56368439 0.55686064 0.53945839 0.51794944]

SKFold f1_macro :: TF :  0.5531487999460587 , CV :  0.5410395492065031
```

dev 파일 TF vs CV

```
dev 파일 :: TF :  0.5773105429455988
dev 파일 :: CV :  0.5594804815636172
```

CONTENTS

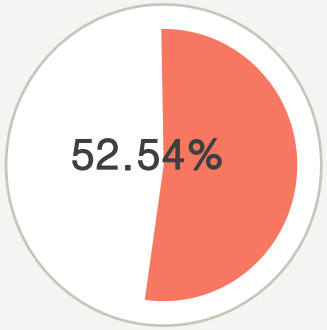
Naive Bayes

모델 실험

전처리: 특수문자와 공백 모두 제거 -> pykospacing.spacing 이용한 띄어쓰기

```
Most Informative Features
                    년 = True             hate : none   =     37.8 : 1.0
                   재앙 = True             hate : offens =     36.2 : 1.0
                   돼지 = True             hate : none   =     31.0 : 1.0
                   새끼 = True             hate : none   =     29.8 : 1.0
                    쳐 = True             hate : none   =     26.3 : 1.0
                  000 = True             hate : none   =     24.9 : 1.0
                   페미 = True             hate : offens =     24.8 : 1.0
                  개돼지 = True             hate : offens =     24.0 : 1.0
                   미투 = True             hate : none   =     20.1 : 1.0
                    ㅈ = True             hate : none   =     18.8 : 1.0
```

Final F1-score:



전처리: 특수문자 제거 -> 띄어쓰기 -> 문장 분리
F1-score: 50.97%

CONTENTS

Cosine Similarity 1

모델 실험

1. 훈련 데이터를 각 레이블별로 분류해 TFIDF Vectorize 한 뒤, 각각의 평균 벡터값 산출

```
# 각각의 평균 벡터값(위치) 산출
none_vec = []
offensive_vec = []
hate_vec = []

for i in range(none_matrix.shape[1]):
    none_vec.append(none_matrix[:,i].mean())

for i in range(offensive_matrix.shape[1]):
    offensive_vec.append(offensive_matrix[:,i].mean())

for i in range(hate_matrix.shape[1]):
    hate_vec.append(hate_matrix[:,i].mean())

# 벡터라이즈 잘 되었는지 길이 확인
len(none_vec), len(offensive_vec), len(hate_vec)
```

2. 코멘트의 벡터값 <-> 각 레이블 평균벡터값 간의 유사도 측정 후, 가장 유사도가 높은 레이블 리턴

```
from sklearn.metrics.pairwise import cosine_similarity

preds = []

for i in range(test_matrix.shape[0]):
    distances = {cosine_similarity(test_matrix[i,:], none_vec)[0][0] : "none",
                 cosine_similarity(test_matrix[i,:], offensive_vec)[0][0] : "offensive",
                 cosine_similarity(test_matrix[i,:], hate_vec)[0][0] : "hate"}
    labels.append( distances[max(distances.keys())] )
```

=> 검증 데이터에 적용했을 때,
Acc : 0.503184
F1 Score : 0.498306
로 우수한 성능을 보여주진 못함

CONTENTS

Cosine Similarity 2

모델 실험

댓글를 입력시 코사인 유사도 Top3를 출력하는 함수 생성

1. 모든 라벨값이 일치 할 경우, 그것으로 최종 예측

2. 2가지 종류 라벨이 나왔을 경우는 하나의 라벨이 높은 값을 보이고 나머지 두 개가 유사도가 굉장히 낮을때를 대비하여 같은 라벨 값끼리 유사도를 더함.

   bias가 존재 하면 +0.1, gender_bias는 +0.2 를 더해주어서 최종적으로 제일 높은 라벨으로 예측

3. 모두 다른 라벨 값이 나왔을때는 부정적인 덧글이 최소 2개 있다는 뜻이기 때문에, offensive나 hate 값중에 유사도가 더 높은 라벨을 예측, 2번과 마찬가지 bias 가중치를 적용

하지만 동의어나 문맥에 취약해서 눈에 띄는 결과는 보이지 못함

comments	bias	contain_gender_bias	hate	score
민상아 처낟 잘보내고 와	none	False	none	0.345514
마흔이 요즘은 늦은 나이 아니예요힘내시고 몸 잘 추스리시길	none	False	none	0.269861
제일 예쁜 마음 원호 오빠... 제일 귀여운 토끼 원호 오빠 제일 착한 사람 원호 ...	none	False	none	0.266653

comments	bias	contain_gender_bias	hate	score
듣보잡. 노잼	none	False	offensive	0.398685
그건 모르겠고 갑자기 이 듣보잡이 쳐나대는 게꼴보기싫음	others	False	hate	0.238098
결혼도안하고 뭐하는 짓?	others	False	offensive	0.211598

comments	bias	contain_gender_bias	hate	score
ㅋㅋㅋ 개꼬시다	none	False	offensive	0.212234
폭풍 짜리몽땅 됐노!!!!!!	others	False	hate	0.202414
엄마닮아서 천만다행~!!	none	False	none	0.195253

CONTENTS

Machine Learning Classifier

Logistic Reg. VS RandomForestClassifier VS SVC VS LGBMClassifier

```
models = [RandomForestClassifier(), LogisticRegression(), SVC(), LGBMClassifier()]

def get_score(model):
    # 훈련: train 전체 / 테스트: dev 전체
    X_train = train['words']
    X_test = test['comments']
    y_train = train['label']
    y_test = test['label']

    # Setting up the pipeline
    vec_pipe = Pipeline([
                        ("vec", TfidfVectorizer(tokenizer=t_tokenizer)),
                        ("model", model)
                        ])

    # Setting the VEC hyperparameters
    vec_pipe_params = {"vec__ngram_range" : [(1,2)],
                       "vec__stop_words"  : [None],
                       "vec__min_df" : [3],
                       "vec__max_df" : [0.9]}


    # Instantiating the grid search
    vec_gs = GridSearchCV(vec_pipe,
                          param_grid=vec_pipe_params,
                          cv=3)

    # Fitting the model to the training data
    vec_gs.fit(X_train, y_train);

    # Predicting
    train_pred = vec_gs.predict(X_train)
    test_pred = vec_gs.predict(X_test)

    # Score
    f1 = [f1_score(train_pred, y_train, average='macro'),
          f1_score(test_pred, y_test, average='macro')]
    acc = [accuracy_score(y_train, train_pred),
           accuracy_score(y_test, test_pred)]
    score_df = pd.DataFrame(data={'F1 Score' : f1, 'Accuracy' : acc},
                            index=['Train', 'Test'])
    return score_df
```

Model : RandomForestClassifier()

	F1 Score	Accuracy
Train	0.998677	0.998734
Test	0.487739	0.513800

Model : LogisticRegression()

	F1 Score	Accuracy
Train	0.832283	0.841565
Test	0.579778	0.585987

Model : SVC()

	F1 Score	Accuracy
Train	0.973995	0.974924
Test	0.536260	0.552017

Model : LGBMClassifier()

	F1 Score	Accuracy
Train	0.787965	0.796479
Test	0.546274	0.552017

=> 전처리 없이 Okt로 토큰화만 진행했을 때, 전반적으로 Logistic Regression이 우수한 성능을 보임

CONTENTS

TABLE OF CONTENTS

Machine Learning: NLP Project

CONTENTS

Logistic Regression with Word2Vec (1)

벡터라이즈 방식 변경

1. 레이블 되지 않은 200만 여 개의 raw comment data 중, 100만개의 코멘트를 랜덤 샘플링하여 Word2Vec 학습

```
model = Word2Vec(tokenized_data, size=100, window=5, min_count=5, sg=1)

model.save('/content/drive/MyDrive/Colab Notebooks/million_comments.model')
model.most_similar(positive=["연예인"], topn=100)
```

```
[('연옌', 0.857076108455658),
 ('일반인', 0.8052691221237183),
 ('옌', 0.7688726782798767),
 ('유명인', 0.7578760385513306),
 ('정치인', 0.7113410234451294),
 ('스포츠스타', 0.7022125720977783),
 ('연예', 0.6956661939620972),
 ('방송인', 0.6952584981918335),
 ('직종', 0.686843752861023),
 ('옌예', 0.6867480278015137),
 ('옌예인', 0.6753039360046387),
 ('인들', 0.672042191028595),
 ('공인', 0.6688129305839539),
```

2. 각 단어별 벡터를 더한 평균값을 이용해 문장 벡터 산출

```
def get_features(words):
    # 출력 벡터 초기화
    feature_vector = np.zeros(100, dtype=np.float32)
    num_words = 0

    # 어휘 사전 준비
    index2word_set = set(model.wv.index2word)

    for w in words:
        if w in index2word_set:
            num_words = 1
            # 사전에 해당하는 단어에 대해 단어 벡터를 더함
            feature_vector = np.add(feature_vector, model[w])

    # 문장의 단어 수만큼 나누어 단어 벡터의 평균값을 문장 벡터로 함
    feature_vector = np.divide(feature_vector, num_words)
    return feature_vector
```

```
get_features('아 진짜 짜증난다')
```

```
array([-0.07574642, -0.0988207 , -0.48576212, -0.4776301 ,  1.0557545 ,
        1.638185  , -0.52016985, -0.10827143, -0.6731411 ,  0.5465597 ,
       -1.2510335 ,  0.7781881 ,  0.86021054, -1.5637627 , -1.1958218 ,
        0.48384106, -0.5856546 ,  0.3942331 ,  0.58857167, -0.8910247 ,
       -0.86850727,  1.3836409 ,  0.14626695,  0.72884285,  0.14109324,
       -0.9455564 , -1.2405579 , -0.9072786 ,  0.1578528 , -0.3662238 ,
        0.9585849 ,  0.27965495, -0.5208655 ,  0.61863285,  0.80589837,
       -0.7552631 , -0.7169019 , -0.13069305,  0.49273053,  0.58437   ,
       -0.9490637 ,  0.21465805, -1.1149619 , -2.0163708 , -0.53309685,
```

CONTENTS

Logistic Regression with Word2Vec (2)

벡터라이즈 방식 변경

3. Logistic Regression 사용하여 예측

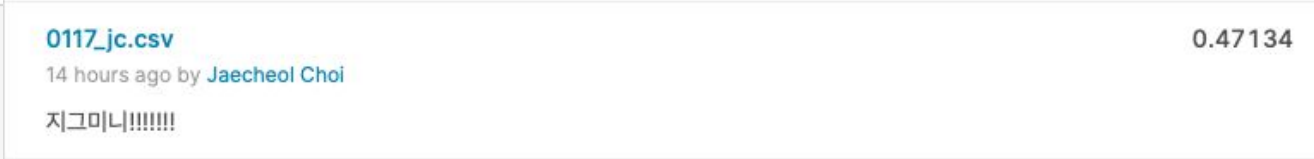
```
def get_dataset(comments):
    dataset = []

    for s in comments:
        dataset.append(get_features(s))

    commentFeatureVecs = np.stack(dataset)
    return commentFeatureVecs
```

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(multi_class='multinomial', class_weight='balanced')

X_train_vecs = get_dataset(X_train)
lr.fit(X_train_vecs, y_train)

X_test_vecs = get_dataset(X_test)

preds = lr.predict(X_test_vecs)
```

```
from sklearn.metrics import accuracy_score, f1_score
print("Accuracy : {}".format(accuracy_score(preds, y_test)),
    "F1 Score : {}".format(f1_score(preds, y_test, average='macro')))
```

```
Accuracy : 0.5520169851380042 F1 Score : 0.5457512516497584
```

4. Kaggle 업로드 결과



CONTENTS

Logistic Regression with Doc2Vec (1)

벡터라이즈 방식 변경

	comments	hate_label
8362	지현우 범죄 저지르지 않았나요?	offensive
8363	여자인생 망칠 일 있나 ㅋㅋ	hate
8364	근데 전라도에서 사고가 났는데 굳이 서울까지 와서 병원에 가는 이유는?	offensive
8365	할매젖x, 뱃살x, 몸매 s라인, 유륜은 적당해야됨(너무크거나 너무 작아도 x), ...	hate
8366	남자가 잘못한거라면... 반성도 없다면...나였다면 ... 여자처럼 아주 못되게 할...	none

```
train.shape
```

```
(8367, 2)
```

```
train_tagged = train.apply(lambda r: TaggedDocument(words=t_tokenizer(r['comments']), tags=[r.hate_label]), axis=1)
test_tagged = test.apply(lambda r: TaggedDocument(words=t_tokenizer(r['comments']), tags=[r.hate_label]), axis=1)
```

```
(TaggedDocument(words=['저', '는', '애초', '에', '믿지도', '않았어요', ',', '걱정', '마시고', '힘내세요', ',', '응원', '합니다', ',',
'나', '영석', 'PD', '님', '정유미', '님'], tags=['none']),
 TaggedDocument(words=['이시언', '집은', '또', '그', '새', '짐', '이', '많아졌어', '뭘', '저렇게', '집안', '에', '살림', '으로', '채우
는', '지하', '는', '짓', '이', '꼭', '노인', '같', '애'], tags=['offensive']),
 TaggedDocument(words=['웬', '지', '페미', '에', '문슬람', '같도', 'ㅋㅋㅋ'], tags=['hate'])
```

CONTENTS

Logistic Regression with Doc2Vec (2)

벡터라이즈 방식 변경

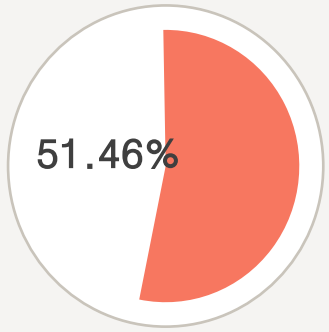
DBOW – 'Distributed Bag of Words'

```
model_dbow = Doc2Vec(dm=1, vector_size=500, negative=5, hs=0, min_count=10, sample=0, workers=cores)
model_dbow.build_vocab([x for x in tqdm(train_tagged.values)])
```

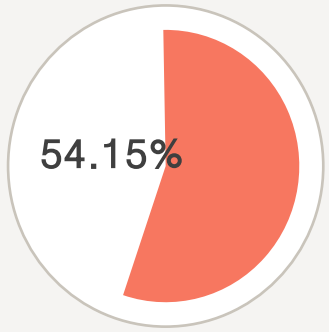
DM – 'Distributed Memory'

```
model_dmm = Doc2Vec(dm=1, dm_mean=1, vector_size=500, window=5, negative=5, min_count=10, workers=8, alpha=0.025, min_alpha=0.00025)
model_dmm.build_vocab([x for x in tqdm(train_tagged.values)])
```

F1-score: **DBOW**

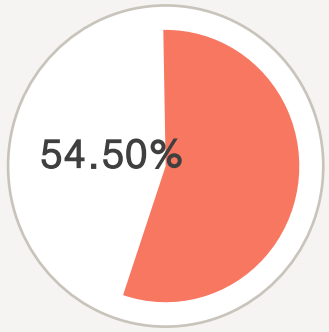


DM




```
new_model = ConcatenatedDoc2Vec([model_dbow, model_dmm])
```

Final F1-score:



CONTENTS

Parameter trials

Logistic Regression

실험 01 : 전처리X 모델 학습

1. 과정 : Tfidf 벡터화 - Logistic Regression 모델 학습
   dev 데이터로 f1-score 검증
   kaggle 제출

2. 검증 :
   가) max_features = 5000 / f1_score = 0.5773
   나) max_features = 10000 / f1_score = 0.5910

3. 결과 :
   가 결과) kaggle 0.515
   나 결과) kaggle 0.5165

4. 결론 :
   max features값 올릴수록 f1score 올라간다?

```
from sklearn.feature_extraction.text import TfidfVectorizer



vec = TfidfVectorizer(min_df=0.0, analyzer='char', ngram_range=(1,3),
                      sublinear_tf=True, max_features=5000)
X_tf = vec.fit_transform(X)
print(X_tf.shape)
```

```
(7893, 5000)
```

```
from sklearn.linear_model import LogisticRegression
lgs = LogisticRegression(multi_class='multinomial', solver='lbfgs', C=1,
                         class_weight='balanced',
                         max_iter=6000, random_state=10)
lgs.fit(X_tf, y)
X_test_tf = vec.transform(X_test)
pred =  lgs.predict(X_test_tf)
```

가 f1 score

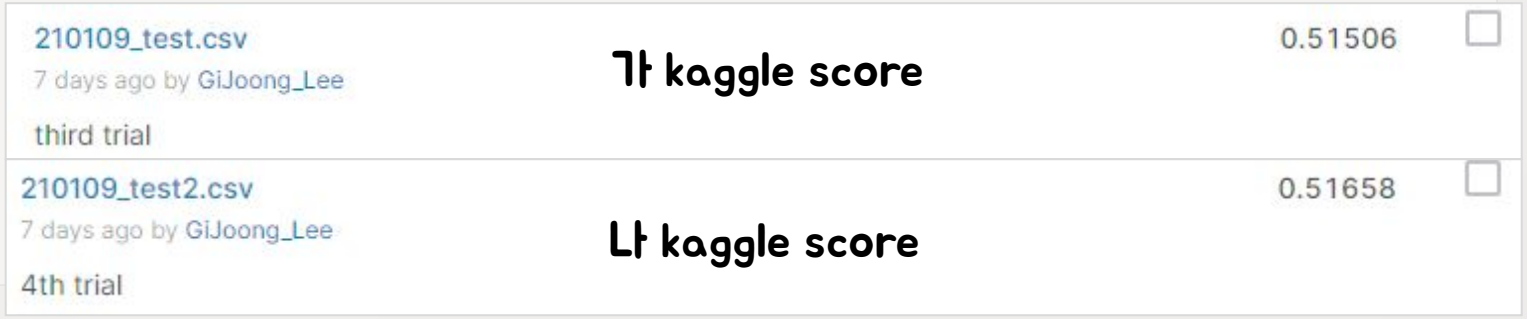
```
from sklearn.metrics import f1_score
f1_score(y_test, pred, average='macro')
```

```
0.5773105429455988
```

나 f1 score

```
from sklearn.metrics import f1_score
f1_score(y_test, pred, average='macro')
```

```
0.5910663394900268
```



CONTENTS

Parameter trials

Logistic Regression

실험 02 : 전처리X 모델 학습

-목적 : max_feature 값 변경

1. 과정 : Tfidf 백터화 - Logistic Regression 모델 학습
   dev 데이터로 f1-score 검증
   kaggle 제출

2. 검증 :
   baseline:
   max_feature = 10000
   f1_score = 0.5910 / kaggle = 0.5165

   해당 옵션의 최대 max_features = 146,518
   max_feature = 10000 ~ 150,000
   - top 3 : 70000, 140000, 100000

3. 결과 :
   a. 70000 : kaggle = 0.5249 -- 과적합 의심
   b. 140000 : kaggle = 0.52893
   c. 100000 : kaggle = 0.52899

```python
from sklearn.feature_extraction.text import TfidfVectorizer

vec = TfidfVectorizer(min_df=0.0, analyzer='char', ngram_range=(1,3),
                      sublinear_tf=True, max_features=100000)
```

f1 scoreTop 3

score	max_feature
0.620896	70000
0.619016	140000
0.619016	150000
0.617150	80000
0.617150	100000
0.591066	10000

CONTENTS

Parameter trials

Logistic Regression

실험 03 : 전처리X 모델 학습

-목적 : ngram_range와 max_feature 값 변경

1. 과정 : Tfidf 백터화 - Logistic Regression 모델 학습
   dev 데이터로 f1-score 검증
   kaggle 제출

2. 검증 :
   baseline:
   ngram_range = (1,3)
   max_feature = 100000
   f1_score = 0.61715 / kaggle = 0.52899
   ngram_range :
   (2, 3), (1, 4), (2, 4), (1, 5), (2, 5), (1, 6)
   max_features : range(최대 mf/2, mf+1, 10000)

3. 결과 :
   a. (1,4) 240000 : kaggle = 0.52661 -- 과적합 의심
   b. (1,4) 190000 : kaggle = 0.52096 -- 과적합 의심

4. 결론 : 전처리 안한 상태에서 tfidf 설정값만으로는 한계 보임
   전처리 시도 해보자.

```
from sklearn.feature_extraction.text import TfidfVectorizer

vec = TfidfVectorizer(min_df=0.0, analyzer='char', ngram_range=(1,3),
                      sublinear_tf=True, max_features=100000)
```

각 ngram range 별 최대 maxfeatures

(2, 3) : (7893, 144860)
(1, 4) : (7893, 337257)
(2, 4) : (7893, 335599)
(1, 5) : (7893, 572846)
(2, 5) : (7893, 571188)
(1, 6) : (7893, 824542)

f1 score Top 5

score	max_feature	ngram_range
0.619642	240000	(1, 4)
0.619401	230000	(1, 4)
0.619127	190000	(1, 4)
0.617765	430000	(1, 6)
0.617755	180000	(1, 4)

CONTENTS

Parameter trials

Logistic Regression

실험 04 : 다양한 전처리 시도후 f1score 비교

1. 과정 : 전처리 - Tfidf 백터화 - Logistic Regression 모델 학습
   dev 데이터로 f1-score 검증
   kaggle 제출

2. 검증 :
   baseline:
   ngram_range = (1,3), max_feature = 100000
   f1_score = 0.61715 / kaggle = 0.52899

   전처리 리스트
   가) konlpy.tag Okt
   나) soynlp.normalizer repeat_normalize
   반복되는 텍스트 축소
   다) soynlp.tokenizer MaxScoreTokenizer
   띄어쓰기가 지켜지지 않은 문장에서 익숙한 단어부터 확인

중복 제거 예시

```
1999
원본 :  역시 경규옹ㅋㅋㅋㅋㅋㅋ
반복제거 :  역시 경규옹ㅋ
```

```
2032
원본 :  인버마아 ㅠㅠㅠㅠㅠㅠㅠㅠㅠㅠㅠㅠㅠ 보고싶었어 ㅠㅠㅠㅠㅠ
반복제거 :  인버마아 ㅠ 보고싶었어 ㅠ
```

OKT

```
56
원본 :  이던이 인기를 끌면 무조건 현아는 버려진다.
okt :  이 던 이 인기 를 끌 면 무조건 현아 는 버려진다 .
```

```
52
원본 :  현아가 눈이 상당히 낮아 ㅋ
okt :  현아 가 눈 이 상당히 낮아 ㅋ
```

MaxScoreTok

```
8
원본 :  에이미.. 넘 이상한것 같음..
max :  에이미 .. 넘 이상한것 같음 ..
```

```
11
원본 :  통장에 십원밖에 없나? 시집가고 드라마도 안팔리고 .어쩌누
max :  통장에 십원밖에 없나? 시집가고 드라마 도 안팔리고 .어쩌누
```

CONTENTS

Parameter trials

Logistic Regression

실험 04 : 다양한 전처리 시도후 f1score 비교

전처리 리스트

가) konlpy.tag Okt

나) soynlp.normalizer repeat_normalize
반복되는 텍스트 축소

다) soynlp.tokenizer MaxScoreTokenizer
띄어쓰기가 지켜지지 않은 문장에서 익숙한 단어부터 확인

테스트 진행 순서:

가

가 - 나

나

나 - 가

다

다 - 가

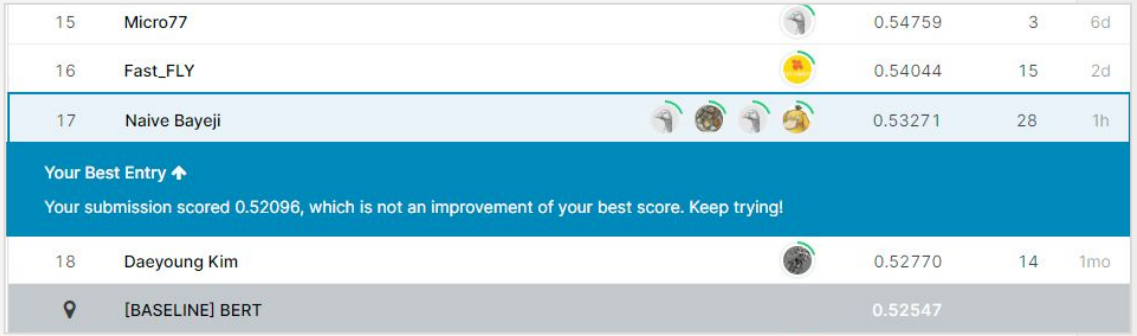
다 - 나

다 - 가 - 나

f1 score 결과

score	test
0.625458	repeat_normalize--maxscore_tokenizer
0.619240	repeat_normalize
0.618095	maxscore_tokenizer
0.580209	maxscore_tokenizer--okt
0.580209	maxscore_tokenizer--okt--repeat_normalize
0.575550	okt--repeat_normalize
0.573148	okt

top1 kaggle 결과 0.5327 상승!!



CONTENTS

TABLE OF CONTENTS

Machine Learning: NLP Project

CONTENTS

딥러닝 모델 소개

Kobert the Transformer

어느날 SQuAD (질의응답) 리더보드에 낯선 모델이 등장했다. BERT라는 이름으로 지금껏 state-of-the-art 였던 앙상블 모델을 single model로 가볍게 누르며 1위를 차지 했다. Pre-Trained Bert모델은 TOEIC 역시 정답률이 76%에 도달하며, 한국어 학습데이터 KorQuad도 2019년 기준, 인간의 문장 판별 및 독해 능력 수준을 넘어섰다. 그외 아래와 같은 다양한 테스크들을 수행할수 있다.

- 감성 분석
- 문장 분류
- 기계 독해
- 문법적으로 맞는 문장 판단
- 현재 문장 다음에 이어질 문장 선택
- 개체명 인식
- 내용 요약
- 관계 추출
- 기계 번역


아이뉴스24

카카오 AI, 한국어 독해능력 평가 인간 앞질렀다

민혜정 입력 2019. 01. 31. 11:23 댓글 44개


데이터뱅크…토플 스피킹 자동채점기 출시…자연어 처리 AI 기술력 증명

문병도 기자 | 승인 2020.12.15 14:29

CONTENTS

딥러닝 모델 소개

Kobert the Transformer

구글의 BERT를 기반으로 한국어에 최적화시킨 자연어 처리 모델 (한국전자통신연구원 2019년 공개) Kobert를 사용해서 악플 데이터 분류를 도전

Base 모델과 Small 모델이 있지만 로컬 환경에서 실행시키기 위해서 Small 모델을 선정

CPU 사용 기준 걸리는 시간은 약 6~7시간

아래는 모델의 config 설정 : 댓글의 글자 길이는 최대 300자 이기때문에 토큰 개수를 최대 128개로 설정

max_seq_len = 128 + epochs = 10 + adam_epsilon = 1e-8 + train_batch_size = 32 + eval_batch_size = 64 + learning_rate = 5e-5

CONTENTS

딥러닝 모델 소개

Kobert the Transformer

사전 학습 데이터

34G 한국어 말뭉치 학습

1. 뉴스
2. 위키
3. 나무위키
4. 구어, 메신저
5. 웹

전처리

1. 한영, 띄어쓰기, 일부 특수 문자 제거
2. 한국어 문장 분리기 (kss)
3. 반복되는 글자 제거 (soynlp)
4. 여러 개의 문장에서 중복 제거

토큰화

1. 미리 학습된 Kobert tokenizer
2. 문장 시작과 끝에 [CLS], [SEP]
3. 토큰 분리시 중간 글자에서는 ##

ex: '[CLS']', '안녕', '##하세요', '[SEP]'

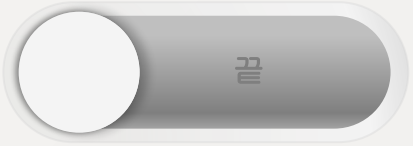
CONTENTS

딥러닝 모델 소개

Kobert의 결과

```
01/17/2021 02:27:02 - INFO - __main__ -   ***** Eval results on test dataset *****
01/17/2021 02:27:02 - INFO - __main__ -     f1 = 0.6536566038928818
01/17/2021 02:27:02 - INFO - __main__ -     precision = 0.6590550674945721
01/17/2021 02:27:02 - INFO - __main__ -     recall = 0.6553354584092289
```

#	Team Name	Notebook	Team Members	Score	Entries	Last
1	sanl			0.60815	44	2mo
2	eunjin kim			0.60784	7	2mo
3	Ji Hyung Moon			0.60419	1	5mo
4	kakao brain			0.60394	2	6mo
5	Yeoun Yi			0.59716	13	6mo
6	Giantpanda			0.58508	1	1mo
7	ohjuhyun			0.58214	14	6mo
8	Naive Bayeji			0.57696	29	~10s



감사합니다.