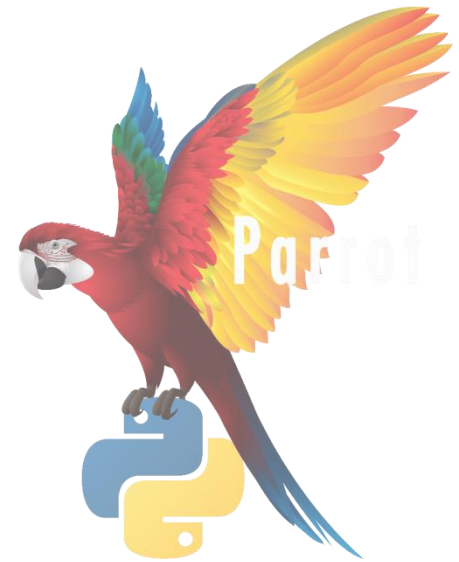
parrot

Data Science

Toxic comment classification : 안소윤, 이수정, 정상희

Contents



시도해본 것들

결과

최종 모델

시도해본 것들

1. 전처리

- Stopwords
- Text augmentation
- RE
- Lemmatization

2. 토큰화

- Tokenizer
- Sentencepiece

1. 전처리: 불용어 제거

```
from tqdm import tqdm
import re
from nltk.corpus import stopwords

def clean_text(text, remove_stopwords = True):
    output = ""
    text = str(text).replace("\n", "")
    text = re.sub(r'[^\w\s]','',text).lower()
    if remove_stopwords:
        text = text.split(" ")
        for word in text:
            if word not in stopwords.words("english"):
                output = output + " " + word
    else:
        output = text
    return str(output.strip())[1:-3].replace("  ", " ")

texts = []

for line in tqdm(train_x, total=train.shape[0]):
    texts.append(clean_text(line))
```

```
1 print('Original data:', train_x[1])
2 print('Length of original data:', len(train_x[1]))
3 print('Cleaned data:', texts[1])
4 print('Length of cleaned data:', len(texts[1]))
```

```
Original data: d aww he matches this background colour i am seemingly stuck with thanks talk       january         utc
Length of original data: 103
Cleaned data: ww matches background colour seemingly stuck thanks talk    january
Length of cleaned data: 72
```

1. 전처리: Text Augmentation for toxic data

(1) Synonym Replacement

```
from textaugment import EDA
e = EDA()

X_sr = toxic_X.map(lambda x: e.synonym_replacement(x))
```

```
 1 toxic_X[43]
```

```
'fuck your filthy mother in the ass dry'
```

```
 1 X_sr[43]
```

```
'shag your filthy mother in the ass dry'
```

```
1 toxic_X[43]
```

```
'fuck your filthy mother in the ass dry'
```

(2) Random Deletion

```
1 X_rd = toxic_X.map(lambda x: e.random_deletion(x))
```

```
1 X_rd[43]
```

```
'fuck your filthy mother in the ass dry'
```

(3) Random Swap

```
1 X_rs = toxic_X.map(lambda x: e.random_swap(x))
```

```
1 X_rs[43]
```

```
'fuck your filthy mother in the dry ass'
```

(4) Random Insertion

```
1 X_ri = toxic_X.map(lambda x: e.random_insertion(x))
```

```
1 X_ri[43]
```

```
'father fuck your filthy mother in the ass dry'
```

1. 전처리: Text Cleaning with re

```
def clean_text_1(text):
    text = text.lower()
    text = re.sub(r"it's\s","it is",text)
    text = re.sub(r"aren't","are not",text)
    text = re.sub(r"couldn't","could not",text)
    text = re.sub(r"didn't","did not",text)
    text = re.sub(r"doen't","does not",text)
    text = re.sub(r"don't","do not",text)
    text = re.sub(r"hadn't","had not",text)
    text = re.sub(r"hasn't","has not",text)
    text = re.sub(r"haven't","have not",text)
    text = re.sub(r"isn't","is not",text)

    text = re.sub(r"arent","are not",text)
    text = re.sub(r"couldnt","could not",text)
    text = re.sub(r"didnt","did not",text)
    text = re.sub(r"doesnt","does not",text)
    text = re.sub(r"dont","do not",text)
    text = re.sub(r"hadnt","had not",text)
    text = re.sub(r"hasnt","has not",text)
    text = re.sub(r"havent","have not",text)
    text = re.sub(r"isnt","is not",text)

    text = re.sub(r"\\n"," ",text)

    text = re.sub(r"mustn't","must not",text)
    text = re.sub(r"shadn't","shall not",text)
    text = re.sub(r"weren't","were not",text)
    text = re.sub(r"where's","where is",text)
    text = re.sub(r"who'd","who would",text)
    text = re.sub(r"won't","will not",text)
    text = re.sub(r"wouldn't","would not",text)
    text = re.sub(r"what's", "what is", text)

    text = re.sub(r"\'ve", " have", text)
    text = re.sub(r"can't", "can not", text)
    text = re.sub(r"let's","let us",text)
    text = re.sub(r"mightn't","might not",text)
    text = re.sub(r"i'm", "i am", text)

    text = re.sub(r"cant", "can not", text)
    text = re.sub(r"lets","let us",text)
    text = re.sub(r"mightnt","might not",text)
    text = re.sub(r"im\s", "i am", text)

    text = re.sub(r"\'re", " are", text)
    text = re.sub(r"tryin","trying",text)
    text = re.sub(r"\'ll", " will", text)
    text = re.sub(r'\W', ' ', text)
    text = re.sub(r'\s+', ' ', text)
    text= re.sub(r"@[A-Za-z0-9]+", ' ', text)
    text = re.sub(r"https?://[A-Za-z0-9./]+", ' ', text)
    text = re.sub(r"[^a-zA-Z.!?']", ' ', text)
    text = re.sub(r"-"," ",text)
    text = text.strip(' ')
    return text
```

1. 전처리: Text Cleaning with re

```
[ ]  1 print('Original data:', train_x[1], train_y[1])
     2 print('Length of original data:', len(train_x[1]))
     3 print('Cleaned data:', train_texts[1], train_y[1])
     4 print('Length of cleaned data:', len(train_texts[1]))
```

```
Original data: D'aww! He matches this background colour I'm seemingly stuck with. Thanks.  (talk) 21:51, January 11, 2016 (UTC) [0 0 0 0 0 0]
Length of original data: 112
Cleaned data: daww he matches this background colour i am  seemingly stuck with thanks  talk  january  utc [0 0 0 0 0 0]
Length of cleaned data: 93
```

1. 전처리: Text Lemmatization (표제어 추출)

https://wikidocs.net/21707

```python
from nltk.stem import WordNetLemmatizer

l = WordNetLemmatizer()

def lemma(text, lemmatization=True):
  output=""
  if lemmatization:
    text=text.split(" ")
    for word in text:
       word1 = l.lemmatize(word, pos = "n")
       word2 = l.lemmatize(word1, pos = "v")
       word3 = l.lemmatize(word2, pos = "a")
       word4 = l.lemmatize(word3, pos = "r")
       output=output + " " + word4
  else:
    output=text

  return str(output)
```

```
print('Cleaned data:', train_texts[1], train_y[1])
print('Length of cleaned cleaned data:', len(train_texts[1]))
print('Lemmatized data:', train_x_lemma[1], train_y[1])
print('Length of lemmatized data:', len(train_x_lemma[1]))
```

```
Cleaned data: daww he matches this background colour i am  seemingly stuck with thanks  talk  january  utc [0 0 0 0 0 0]
Length of cleaned cleaned data: 93
Lemmatized data:  daww he match this background colour i be  seemingly stick with thank  talk  january  utc [0 0 0 0 0 0]
Length of lemmatized data: 91
```

2. 전처리: Tokenizer

```
 1 from tensorflow.keras.preprocessing.text import Tokenizer
 2 from tensorflow.keras.preprocessing.sequence import pad_sequences
 3
 4 t = Tokenizer()
 5 t.fit_on_texts(X)
 6 t.fit_on_texts(test_X)
 7
 8 X_encoded = t.texts_to_sequences(X)
 9 test_X_encoded = t.texts_to_sequences(test_X)
10
11 X = pad_sequences(X_encoded, maxlen=200, padding='post')
12 test_X = pad_sequences(test_X_encoded, maxlen=200, padding='post')
```

2. 전처리: Sentencepiece

https://wikidocs.net/86657

```
[ ]  1 templates = '--input={} --model_prefix={} --vocab_size={}'
     2 cmd = templates.format(input_file, prefix, vocab_size)
     3 cmd
```

```
'--input=/content/drive/MyDrive/Parrot/spm_train.txt --model_prefix=/content/drive/MyDrive/Parrot/sentencepiece/toxic --vocab_size=25000'
```

```
[ ]  1 import sentencepiece as spm
     2 sp = spm.SentencePieceProcessor()
     3 sp.Load("/content/drive/MyDrive/Parrot/sentencepiece/toxic.model")
```

```
True
```

```
[ ]  1 for t in train_x_lemma[:5]:
     2   print(t)
     3   print(sp.encode_as_pieces(t))
     4   print(sp.encode_as_ids(t), '₩n')
```

```
 explanationwhy the edit ismade under my username hardcore metallica fan be revert they be not vandalism just closure on some gas after i vote at new york doll fac and please do not remove the template from the talk page si
['_explanation', 'why', '_the', '_edit', '_is', 'made', '_under', '_my', '_username', '_hard', 'core', '_metallic', 'a', '_fan', '_be', '_revert', '_they', '_be', '_not', '_vandalism', '_just', '_clo', 'sure', '_on', '_
[1182, 3719, 4, 48, 94, 2686, 279, 37, 1654, 432, 2902, 3996, 82, 1082, 5, 186, 63, 5, 17, 308, 60, 2865, 1911, 19, 49, 4216, 193, 8, 888, 41, 163, 1080, 6775, 2420, 10, 59, 21, 17, 117, 4, 420, 42, 4, 62, 33, 198, 8, 5, 39

 daww he match this background colour i be seemingly stick with thank  talk  january  utc
['_daw', 'w', '_he', '_mat', 'ch', '_this', '_background', '_', 'colour', '_i', '_be', '_seem', 'ingly', '_stick', '_with', '_thank', '_talk', '_january', '_utc']
[7212, 245, 57, 1730, 216, 23, 1168, 3, 3471, 8, 5, 168, 1524, 1474, 27, 101, 62, 1346, 460]

 hey man i be really not try to edit war it isjust that this guy be constantly remove relevant information and talk to me through edit isinstead of my talk page he seem to care more about the format than the actual info
['_hey', '_man', '_i', '_be', '_really', '_not', '_try', '_to', '_edit', '_war', '_it', '_is', 'just', '_that', '_this', '_guy', '_be', '_constant', 'ly', '_remove', '_relevant', '_information', '_and', '_talk', '_to',
[978, 453, 8, 5, 192, 17, 153, 7, 48, 210, 16, 94, 1209, 15, 23, 458, 5, 2172, 25, 117, 624, 138, 10, 62, 7, 43, 316, 48, 2278, 396, 9, 37, 62, 33, 57, 168, 7, 534, 71, 46, 4, 889, 140, 4, 626, 698]

 morei can not make any real suggestion on i amprovement  i wonder if the section statistic should be late on or a subsection of type of accident  i think the reference may need tidy so that they be all in the exact same fo
['_more', 'i', '_can', '_not', '_make', '_any', '_real', '_suggest', 'ion', '_on', '_i', '_amprovement', '_i', '_wonder', '_if', '_the', '_section', '_statistic', '_should', '_be', '_late', '_on', '_or', '_a', '_subsec
[71, 78, 40, 17, 55, 77, 522, 281, 74, 19, 8, 2852, 8, 674, 30, 4, 135, 1661, 70, 5, 485, 19, 35, 6, 2170, 9, 518, 9, 3214, 8, 61, 4, 143, 88, 125, 96, 1498, 50, 15, 63, 5, 54, 14, 4, 433, 183, 889, 8, 18, 474, 889, 513, 8,

 you sir be my hero any chance you remember what page thats on
['_you', '_sir', '_be', '_my', '_hero', '_any', '_chance', '_you', '_', 'remember', '_what', '_page', '_that', 's', '_on']
[11, 1847, 5, 37, 3688, 77, 1142, 11, 3, 739, 44, 33, 15, 31, 19]
```

- 사전 훈련된 Word Embedding (GloVe)

```
from numpy import array
from numpy import asarray
from numpy import zeros

embeddings_dictionary = dict()

glove_file = open('/content/drive/MyDrive/glove.6B.100d.txt.zip (Unzipped Files)/glove.6B.100d.txt', encoding="utf8")

for line in glove_file:
    records = line.split()
    word = records[0]
    vector_dimensions = asarray(records[1:], dtype='float32')
    embeddings_dictionary[word] = vector_dimensions
glove_file.close()

embedding_matrix = zeros((vocab_size, 100))
for word, index in tokenizer.word_index.items():
    embedding_vector = embeddings_dictionary.get(word)
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
```

- 모델링: CNN

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 250)]             0
_________________________________________________________________
embedding (Embedding)        (None, 250, 100)          3000000
_________________________________________________________________
conv1d (Conv1D)              (None, 250, 64)           32064
_________________________________________________________________
batch_normalization (BatchNo (None, 250, 64)           256
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 50, 64)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 50, 64)            20544
_________________________________________________________________
batch_normalization_1 (Batch (None, 50, 64)            256
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 16, 64)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 16, 64)            20544
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 64)            256
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 5, 64)             0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 5, 64)             20544
_________________________________________________________________
batch_normalization_3 (Batch (None, 5, 64)             256
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 1, 64)             0
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 390
=================================================================
Total params: 3,099,270
Trainable params: 98,758
Non-trainable params: 3,000,512
```

- 모델링: Bidirectional LSTM

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 200)]             0
_________________________________________________________________
embedding_4 (Embedding)      (None, 200, 128)          3200000
_________________________________________________________________
bidirectional_3 (Bidirection (None, 200, 100)          71600
_________________________________________________________________
global_max_pooling1d (Global (None, 100)               0
_________________________________________________________________
dropout (Dropout)            (None, 100)               0
_________________________________________________________________
dense_3 (Dense)              (None, 50)                5050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 306
=================================================================
Total params: 3,276,956
Trainable params: 3,276,956
Non-trainable params: 0
```

- 모델링: Bidirectional LSTM + GloVe

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 200, 100)          14843700
_________________________________________________________________
lstm_2 (LSTM)                (None, 200, 128)          117248
_________________________________________________________________
lstm_3 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 390
=================================================================
Total params: 15,010,746
Trainable params: 167,046
Non-trainable params: 14,843,700
```

- 모델링: CNN+GRU

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 250)]             0
_________________________________________________________________
embedding_1 (Embedding)      (None, 250, 100)          3000000
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 250, 32)           16032
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 83, 32)            0
_________________________________________________________________
gru (GRU)                    (None, 32)                6336
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 198
=================================================================
Total params: 3,022,566
Trainable params: 22,566
Non-trainable params: 3,000,000
```

- 모델링: NBSVM (Naive Bayes SVM)

https://nlp.stanford.edu/pubs/sidaw12_simple_sentiment.pdf

```
import re, string
re_tok = re.compile(f'([{string.punctuation}“”¨«»®´·º½¾¿¡§££‘’])')
def tokenize(s): return re_tok.sub(r' \1 ', s).split()
```

```
n = train.shape[0]
vec = TfidfVectorizer(ngram_range=(1,2), tokenizer=tokenize,
               min_df=3, max_df=0.9, strip_accents='unicode', use_idf=1,
               smooth_idf=1, sublinear_tf=1 )
trn_term_doc = vec.fit_transform(train_x_lemma)
test_term_doc = vec.transform(test['comment_text'])
```

re 전처리, TfidfVectorizer로 벡터화

- 모델링: NBSVM (Naive Bayes SVM)

```
x = trn_term_doc
test_x = test_term_doc
```

```
def get_mdl(y):
    y = y.values
    r = np.log(pr(1,y) / pr(0,y))
    m = LogisticRegression(C=4)
    x_nb = x.multiply(r)
    return m.fit(x_nb, y), r
```

```
import numpy as np
from sklearn.linear_model import LogisticRegression
preds = np.zeros((len(test), len(train.columns[2:])))

for i, j in enumerate(train.columns[2:]):
    print('fit', j)
    m,r = get_mdl(train[j])
    preds[:,i] = m.predict_proba(test_x.multiply(r))[:,1]
```

- 모델링: Ensemble (submission점수 기준 csv파일 앙상블)

```
p_lstm = pd.read_csv(f_lstm)
p_nbsvm = pd.read_csv(f_nbsvm)
```

```
label_cols = ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']
p_res = p_lstm.copy()
p_res[label_cols] = (p_nbsvm[label_cols] + p_lstm[label_cols]) / 2
```

```
p_res.to_csv('submission_nbsvm_lstm.csv', index=False)
```

- **정리**

- 불용어 처리와 augmentation은 정확도를 크게 올려주지는 않음.
  (오히려 처리 안했을 때가 더 높았음)
- 전처리에서는 정규표현식으로 cleaning하거나 표제어를 추출했을 때 더 좋은 결과를 보임.
- 모델 중에서는 GloVe를 포함한 모델이 대체로 성능이 좋게 나옴.
  + 기본 CNN이나 GRU 모델보다는 양방향 LSTM 모델이 성능 GOOD
- 앙상블이 꽤 효과가 좋았음.

최종 모델

- 결과

	Private score	Public score
final4.csv	0.98279	0.98252
final2.csv	0.98280	0.98246

수고하셨습니다