Metin Madenciligi Dersi

Nefret Soylemi ve Ofansif Dil Problemi

- Oguzhan KANDAKOGLU
- Elif Yildirim

  Bu calisma iki farkli yontem uzerinde denenmistir.

- Logistic Regression
- Support Vector Machine -> Linear SVC (Support Vector Classification)

Son kisimda iki farkli siniflandirma algoritmasinin (classifier) da performansi gosterilmektedir.

Gerekli Araclarin Dahil Edilmesi

In [ ]:

```
import numpy as np
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.svm import LinearSVC

import nltk
from nltk.stem.porter import *

import re

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer as VS

from textstat.textstat import *

import matplotlib.pyplot as plt
import seaborn

%matplotlib inline
```

Veri setinin yuklenmesi

In [ ]:

```
with open("data/labeled_data.p", 'rb') as f:
    df = pd.read_pickle(f)
```

In [ ]:

```
df
```

Out[ ]:

	count	hate_speech	offensive_language	neither	class	tweet
0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
...	...	...	...	...	...	...
25291	3	0	2	1	1	you's a muthaf***in lie “@LifeAsKing: @2...
25292	3	0	1	2	2	you've gone and broke the wrong heart baby, an...
25294	3	0	3	0	1	young buck wanna eat!!.. dat nigguh like I ain...
25295	6	0	6	0	1	youu got wild bitches tellin you lies
25296	3	0	0	3	2	~~Ruffled | Ntac Eileen Dahlia - Beautiful col...

24783 rows × 6 columns

In [ ]:

```
df.describe()
```

Out[ ]:

	count	hate_speech	offensive_language	neither
count	24783.000000	24783.000000	24783.000000	24783.000000
mean	3.243473	0.280515	2.413711	0.549247
std	0.883060	0.631851	1.399459	1.113299
min	3.000000	0.000000	0.000000	0.000000
25%	3.000000	0.000000	2.000000	0.000000
50%	3.000000	0.000000	3.000000	0.000000
75%	3.000000	0.000000	3.000000	0.000000
max	9.000000	7.000000	9.000000	9.000000

In [ ]:

```
df.columns
```

Out[ ]:

```
Index(['count', 'hate_speech', 'offensive_language', 'neither', 'class',
       'tweet'],
      dtype='object')
```

Kolon Yapisi

count = Her tweetin kac adet CrowdFlower kullanicisi tarafindan etiketlendiginin sayisi. (Her tweet en az 3 adet kullanici tarafindan etiketlenmistir.)

hate_speech = Kac CF kullanicisinin tweeti 'Nefret Soylemi' olarak etiketlediginin sayisi.

offensive_language = Kac CF kullanicisinin tweeti 'Ofansif Dil' olarak etiketlediginin sayisi.

neither = Kac CF kullanicisinin tweeti 'Ne Nefret Soylemi ne de Ofansif Dil' olarak etiketlediginin sayisi.

Siniflandirma Yapisi

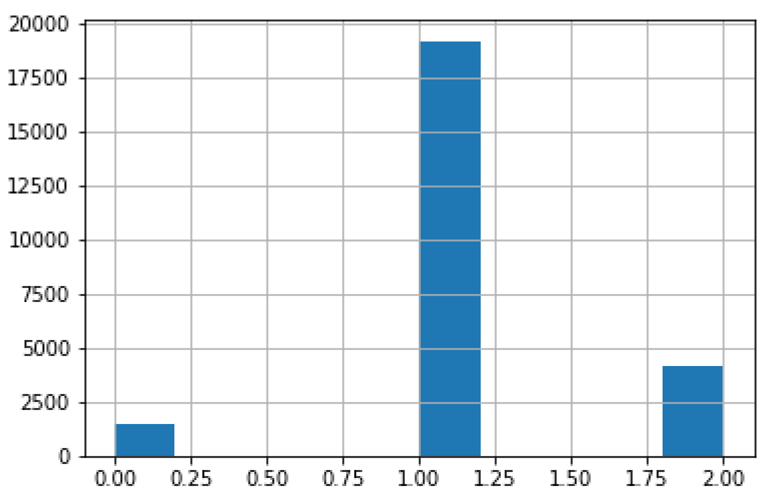
```
0 - Nefret Soylemi
1 - Ofansif Dil
2 - Neither
```

In [ ]:

```
df['class'].hist()
```

Out[ ]:

```
<AxesSubplot:>
```



Metin Koleksiyonunun Belirlenmesi

Histogram uzerinden de goruldugu gibi 'Ofansif Dil' kullanilmis tweetlerin sayisi cok daha fazla. 'Ne Ofansif Dil ne de Nefret Soylemi' barindiran tweetler ise 'Nefret Soylemi' barindiran tweetlerden daha fazla.

Bu durum 'Metin Madenciligi' icin cok da olumlu bir durum degil. Ancak buradan soyle bir sonuc cikarabiliriz. Ancak aciklamaya baslamadan once bir adim oncesine gitmemiz gerekiyor. Yani:

Verilerin Toplanmasi Metin Madenciligi yaparken dikkat etmemiz gereken hususlardan birisi olusturacagimiz modelin hangi alanda calisacagidir. Ornek vermek gerekirse 'TIP' dunyasinda kullanilan terimlerle 'BILISIM' dunyasinda kullanilan kelimeler yazilis olarak ayni olsa da anlam olarak farkli seyleri ifade edebilirler. Dolayisiyle toplayacagimiz verileri ilgili alandan, ilgili konudan toplamamiz gerekir.

Twitter ise her insanin her konudan bahsedebildigi bir platformdur. Icerik olarak cok zengindir. Ancak filtreleme yapmak biraz zordur.

Veri setini olusturan kisi bu konu su sekilde bir yol bulmus:

Tweetler https://hatebase.org adresinde bulunan kelimeleri icerip icermeme durumlarina gore twitter uzerinde 'Scrape' edilmistir.

Insan dilinin ne kadar karmasik bir yapi oldugunu sadece yukaridaki histograma bakarak da anlayabiliriz. 'Tweetler nefret soylemi/kufur barindiran tweetler arasindan seciliyor ve ona ragmen "Ne ofansif ne nefret soylemi" barindiran tweetler sadece nefret soylemi barindiran tweetlerden daha fazla cikiyor. 😊 🌪 ⛈'

Tum bunlara ragmen cikan sonuclar bizim gibi baslangic seviyesindeki iki ogrenci icin tatmin edici durumda.

In [ ]:
```
tweets = df.tweet
```

Metin On Isleme

In [ ]:
```
# Stop wordsler belirlenir
stopwords = nltk.corpus.stopwords.words('english')

# Twitter'a ozel terimler belirlenir.
exclusions_for_twitter = ['#ff', 'ff', 'rt']

# Belirlenen terimler stopwords listesine eklenir.
stopwords.extend(exclusions_for_twitter)

stemmer = PorterStemmer()

def pre_process(txt_string):
    """
    RegEXP
        - URL
        - Strip
        - Mentions

        !hastaghler nefret soylemi barindirabileceginden dolayi hashtagler dahil ediln
    """

    space_pattern = '\s+'
    url_pattern = 'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]| [!*\(\),]|(?:%[0-9a-fA-F][0
```

```
    mention_pattern = '@[\w\-]+'

    parsed_text = re.sub(space_pattern, ' ', txt_string) # birden fazla bosluk varsa b
    parsed_text = re.sub(url_pattern, '', parsed_text) # URLleri sil
    parsed_text = re.sub(mention_pattern, '', parsed_text) # Mentionleri sil

    return parsed_text

def tokenize(tweet):
    # noktalama isaretleri, bosluklar silinir, kelimeler kokune indirilir (stemming) v
    tweet = " ".join(re.split("[^a-zA-Z]*", tweet.lower())).strip()
    tokens = [stemmer.stem(t) for t in tweet.split()]
    return tokens


def basic_tokenize(tweet):
    # Stemming yapmadan tokenlestirme # etiketler icin kullan
    tweet = " ".join(re.split("[^a-zA-Z.,!?]*", tweet.lower())).strip()
    return tweet.split()
```

Terim Dokuman Matrisinin Olusturulmasi

In [ ]:

```
vectorizer = TfidfVectorizer(
    tokenizer=tokenize,
    preprocessor=pre_process,
    ngram_range=(1, 3), # unigrams, bigrams, trigramslaolusturulur.
    stop_words=stopwords,
    use_idf=True,
    smooth_idf=False,
    norm=None,
    decode_error='replace',
    max_features=10000,
    min_df=5,
    max_df=0.75
)

# Terim dokuman matrisi
tfidf = vectorizer.fit_transform(tweets).toarray()
vocab = {v: i for i, v in enumerate(vectorizer.get_feature_names_out())} # featurelarl
idf_vals = vectorizer.idf_

# Terim dokuman puanlari
idf_dict = {i: idf_vals[i] for i in vocab.values()}
```

Dil Bilgisel Etiketleme

In [ ]:

```
tweet_tags = [] # tweetlerin dil bilgisel etiketleri
for t in tweets:
    tokens = basic_tokenize(pre_process(t))
    tags = nltk.pos_tag(tokens)
    tag_list = [x[1] for x in tags]
    tag_str = " ".join(tag_list)
    tweet_tags.append(tag_str)

pos_vectorizer = TfidfVectorizer(
```

```
    tokenizer=None,
    lowercase=False,
    preprocessor=None,
    ngram_range=(1, 3),
    stop_words=None,
    use_idf=False,
    smooth_idf=False,
    norm=None,
    decode_error='replace',
    max_features=5000,
    min_df=5,
    max_df=0.75,
)

pos = pos_vectorizer.fit_transform(pd.Series(tweet_tags)).toarray()

pos_vocab = {v: i for i, v in enumerate(
    pos_vectorizer.get_feature_names_out())}
```

Ozellik Cikarimi

In [ ]:

```
sentiment_analyzer = VS()


def count_twitter_objs(text_string):
    """
    Bu kisimda twittera ait ozelliklerin sayisi bulunur.
    1) urller URLHERE
    2) mentionlar MENTIONHERE
    3) hashtagler HASHTAGHERE
        ile degistirilir.

    Boylelikle tweet icersinde bu ozelliklerden kacar adet oldugu hesaplanir.
    Hesaplanan degerler tuple olarak return edilir.
    """
    space_pattern = '\s+'
    giant_url_regex = ('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|'
                       '[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')
    mention_regex = '@[\w\-]+'
    hashtag_regex = '#[\w\-]+'
    parsed_text = re.sub(space_pattern, ' ', text_string)
    parsed_text = re.sub(giant_url_regex, 'URLHERE', parsed_text)
    parsed_text = re.sub(mention_regex, 'MENTIONHERE', parsed_text)
    parsed_text = re.sub(hashtag_regex, 'HASHTAGHERE', parsed_text)
    return(parsed_text.count('URLHERE'), parsed_text.count('MENTIONHERE'), parsed_text


def other_features(tweet):
    """
        Tweete ait ekstra ozellikleri bu kisimda Hesaplariz.
        1) Tweet'in duygu analizi sonuclari: VADER (Valence Aware Dictionary and sEnti
            - Pos
            - Neg
            - Neut
            - Compound: pozitif, negatif, ve neutrik puanlari toplaminda ortalama puar
        2) Kelime sayisi
        3) Hece sayisi
```

```
        3) Harf/Karakter sayisi
        4) Tweetin uzunlugu
        5) Ortalama Hece sayisi
        6) Essiz terim sayisi
        7) Terim sayisi
        8) Okunabilirlik puanlari
    """
    sentiment = sentiment_analyzer.polarity_scores(tweet)

    words = pre_process(tweet)  # Get text only

    syllables = textstat.syllable_count(words)
    num_chars = sum(len(w) for w in words)
    num_chars_total = len(tweet)
    num_terms = len(tweet.split())
    num_words = len(words.split())
    avg_syl = round(float((syllables+0.001))/float(num_words+0.001), 4)
    num_unique_terms = len(set(words.split()))

    # Okunabilirlik puanlari
    # https://readable.com/readability/flesch-reading-ease-flesch-kincaid-grade-level/
    # FKRA: Okuma Seviyesi
    # FRE: Kolaylik Seviyesi
    FKRA = round(float(0.39 * float(num_words)/1.0) +
                 float(11.8 * avg_syl) - 15.59, 1)
    FRE = round(206.835 - 1.015*(float(num_words)/1.0) -
                (84.6*float(avg_syl)), 2)


    twitter_objs = count_twitter_objs(tweet)
    retweet = 0
    if "rt" in words:
        retweet = 1

    features = [FKRA, FRE, syllables, avg_syl, num_chars, num_chars_total, num_terms, 
                num_unique_terms, sentiment['neg'], sentiment['pos'], sentiment['neu']
                twitter_objs[2], twitter_objs[1],
                twitter_objs[0], retweet]

    return features

# tweetlere ait ozellikleri cikaririz/hesaplariz.
def get_feature_array(tweets):
    feats = []
    for t in tweets:
        feats.append(other_features(t))
    return np.array(feats)

# olusturdugumuz ozelliklerin isimleri
other_features_names = ["FKRA", "FRE", "num_syllables", "avg_syl_per_word", "num_chars
                        "num_terms", "num_words", "num_unique_words", "vader neg", "va
                        "vader compound", "num_hashtags", "num_mentions", "num_urls", 

# elde edilen tum featurelar
feats = get_feature_array(tweets)


M = np.concatenate([tfidf, pos, feats], axis=1)
```

In [ ]:
```
M.shape
```
Out[ ]:
```
(24783, 4023)
```
In [ ]:
```
# Tum ozellik isimlerinin bir listesini olusturuyoruz.
variables = ['']*len(vocab)
for k, v in vocab.items():
    variables[v] = k

pos_variables = ['']*len(pos_vocab)
for k, v in pos_vocab.items():
    pos_variables[v] = k

feature_names = variables+pos_variables+other_features_names
```

Modelin Calisitirilmasi

In [ ]:
```
X = pd.DataFrame(M)
y = df['class'].astype(int)
```
In [ ]:
```
# Bu kisimda feature onemlerini Logistic Regression ile hesapliyoruz.
# l1: katsayıların büyüklüğünün mutlak değerine eşit
select = SelectFromModel(LogisticRegression(
    class_weight='balanced', penalty="l1", C=0.01, solver='liblinear'))
X_ = select.fit_transform(X, y)
```

Linear SVC

In [ ]:
```
# l2: katsayıların büyüklüğünün karesine eşit
model = LinearSVC(class_weight='balanced', C=0.01, penalty='l2', max_iter=3000).fit(X_
```
In [ ]:
```
y_preds = model.predict(X_)
```

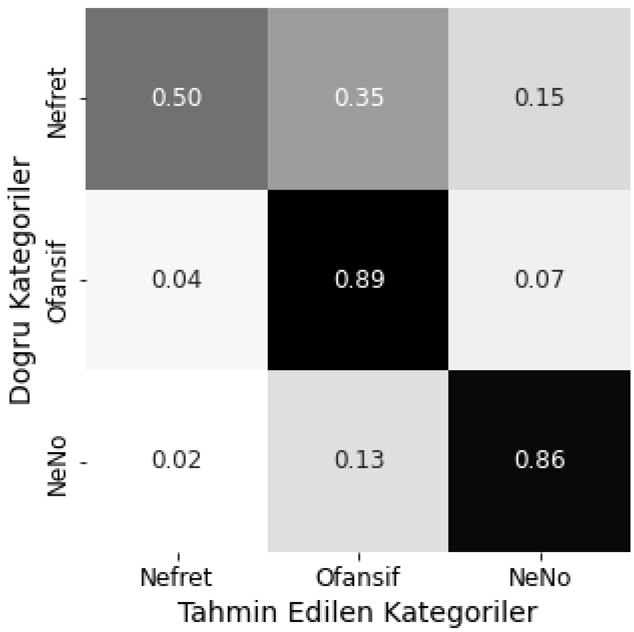
Degerlendirme ve Yorumlama

In [ ]:
```
report = classification_report(y, y_preds)
print(report)
```
```
              precision    recall  f1-score   support

           0       0.44      0.50      0.47      1430
           1       0.94      0.89      0.92     19190
           2       0.70      0.86      0.77      4163

    accuracy                           0.86     24783
   macro avg       0.70      0.75      0.72     24783
weighted avg       0.87      0.86      0.87     24783
```
In [ ]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y, y_preds)
```

```
matrix_proportions = np.zeros((3, 3))
for i in range(0, 3):
    matrix_proportions[i, :] = confusion_matrix[i, :] / \
        float(confusion_matrix[i, :].sum())
# NeNo(Neutral): Ne Nefret Ne Ofansif
names = ['Nefret', 'Ofansif', 'NeNo']
confusion_df = pd.DataFrame(matrix_proportions, index=names, columns=names)
plt.figure(figsize=(5, 5))
seaborn.heatmap(confusion_df, annot=True, annot_kws={
                "size": 12}, cmap='gist_gray_r', cbar=False, square=True, fmt='.2f')
plt.ylabel(r'Dogru Kategoriler', fontsize=14)
plt.xlabel(r'Tahmin Edilen Kategoriler', fontsize=14)
plt.tick_params(labelsize=12)
```



Logistic Regression

In [ ]:
```
model = LogisticRegression(class_weight='balanced',
                           penalty='l2', C=0.01, max_iter=3000).fit(X_, y)
```

In [ ]:
```
y_preds = model.predict(X_)
```

Degerlendirme ve Yorumlama

In [ ]:
```
report = classification_report(y, y_preds)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.27      0.77      0.40      1430
           1       0.97      0.75      0.85     19190
           2       0.63      0.89      0.74      4163

    accuracy                           0.78     24783
   macro avg       0.62      0.80      0.66     24783
weighted avg       0.87      0.78      0.80     24783
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y, y_preds)
matrix_proportions = np.zeros((3, 3))
for i in range(0, 3):
    matrix_proportions[i, :] = confusion_matrix[i, :] / \
        float(confusion_matrix[i, :].sum())
# NeNo(Neutral): Ne Nefret Ne Ofansif
names = ['Nefret', 'Ofansif', 'NeNo']
confusion_df = pd.DataFrame(matrix_proportions, index=names, columns=names)
plt.figure(figsize=(5, 5))
seaborn.heatmap(confusion_df, annot=True, annot_kws={
                "size": 12}, cmap='gist_gray_r', cbar=False, square=True, fmt='.2f')
plt.ylabel(r'Dogru Kategoriler', fontsize=14)
plt.xlabel(r'Tahmin Edilen Kategoriler', fontsize=14)
plt.tick_params(labelsize=12)
```