چالش دوم: Tweet Sentiment Extraction

فراهم‌آورنده: سینا حیدری

زمستان ۹۸

برای مشاهده‌ی جزئیات این چالش در کگل به لینک زیر مراجعه کنید.

https://www.kaggle.com/c/tweet-sentiment-extraction/overview

۱ در مورد این چالش

”My ridiculous dog is amazing.” [sentiment: positive]

به‌دلیل تعداد بالای توییت‌هایی که در هر لحظه ردوبدل می‌شود، تایین اینکه تمایل یک توییت مشخص بر روی شرکت‌ها، اشخاص یا برندها چه تاثیری (مثبت یا منفی) می‌گذارد سخت است. امروزه، پی بردن به تمایل متن‌ها از آن جهت اهمیت یافته که تصمیم‌ها و واکنش‌ها در طی چند لحظه اتفاق افتاده و به‌روزرسانی می‌شود. اما چه کلماتی باعث تعیین تمایل متن‌ها می‌شود؟ در این چالش، باید بخشی از یک توییت (کلمه یا عبارت) را جدا کنیم که تمایل را تحت تاثیر قرار می‌دهد.

۲ راه حل

در مورد این گزارش

در این نوتبوک، ساختار مجموعه‌داده را به طور خلاصه توضیح داده، ویژگی‌های جدید را تولید و آنالیز می‌کنیم؛ سپس، مجموعه‌داده را با استفاده از Matplotlib، Seaborn و Plotly مصورسازی خواهیم کردم. ما به این مسئله، به عنوان یک مسئله‌ی NER [۱] خواهیم پرداخت.
بررسی اکتشافی داده، اطلاعاتی به ما می‌دهد که بر اساس آن می‌توانیم نوع مدل خود را انتخاب کرده و به‌درستی شخصی‌سازی کنیم.

Named Entity Recognition

NER شاید اولین قدم برای استخراج اطلاعات باشد که تلاش می‌کند موجودیت‌های اسم‌دار را در متن شناسایی کرده و آن‌ها را به‌دسته‌های از پیش تعیین شده (همچون اسم افراد، سازمان‌ها، موقعیت‌ها، عبارت‌های زمان، کمیت‌ها، واحدهای پول، درصدها و غیره) نسبت دهد. NER در زمینه‌های مختلف NLP [۲] کاربرد دارد و می‌تواند به‌سوال‌های دنیای واقعی جواب دهد. نمونه‌هایی از این سوال‌ها را در زیر می‌بینید:

[۱] Named-entity recognition
[۲] Natural Language Processing

- کدام شرکت‌ها در مقاله‌ی خبری مشخصی ذکر شده بودند؟
- آیا محصول مشخصی در شکایت‌ها یا نقدها ظاهر شده است؟
- آیا یک توییت شامل اسم یک شخص می‌شود؟ آیا این توییت موقعیت شخص را هم شامل می‌شود؟

ما از Spacy برای ساختن مدل‌های شخصی‌سازی شده‌مان استفاده خواهیم برد. نمونه‌یی از کاربرد NER بر اساس مدل اصلی SpaCy که با مجموعه‌داده‌ی OntoNotes 5 آموزش دیده را در زیر مشاهده می‌کنید. این مدل از موجودیت‌های زیر پشتیبانی می‌کند.

TYPE	DESCRIPTION
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including "%".
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	"first", "second", etc.
CARDINAL	Numerals that do not fall under another type.

طبقه‌بندی در سطح موجودیت، روی متن نمونه:

```
doc = nlp('European authorities fined Google a record $5.1 billion on
    Wednesday for abusing its power in the mobile phone market and
    ordered the company to alter its practices')
pprint([(X.text, X.label_) for X in doc.ents])

```

```
[('European', 'NORP'),
 ('Google', 'ORG'),
 ('$5.1 billion', 'MONEY'),
 ('Wednesday', 'DATE')]
```

در بخش بررسی اکتشافی داده خواهیم دید که برای بیشتر توییت‌های بی‌تمایل (neutral) جواب (selected_text) دقیقا برابر با متن توییت می‌باشد؛ از این رو مدلی برای این دسته از توییت‌ها تعلیم نخواهیم داد. اما دو دسته توییت‌های منفی و مثبت وجود دارد که هر کدام نیازمند یک مدل هستند. به‌طور خلاصه، دو موجودیت اسم‌دار (مثبت و منفی) خواهیم داشت که برای هر کدام یک مدل آموزش می‌دهیم. ورودی مدل توییت‌های مثبت، آن سطرهایی از مجموعه‌داده است که تمایل مثبت دارد و ورودی مدل توییت‌های منفی آن سطرهایی از مجموعه‌داده است که تمایل منفی دارند. عبارت یا کلماتی که توسط NER شناسایی می‌شود نیز همان selected_text می‌باشد.

۳ پیاده‌سازی

بارگذاری پکیج‌های مورد نیاز

```
import re
import string
import numpy as np
import random
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from plotly import graph_objs as go
import plotly.express as px
import plotly.figure_factory as ff
from collections import Counter

from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator


import nltk
from nltk.corpus import stopwords

from tqdm import tqdm
import os
import nltk
import spacy
import random
from spacy.util import compounding
from spacy.util import minibatch

import warnings
warnings.filterwarnings("ignore")

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
  for filename in filenames:
    print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

خواندن داده

```
    print(train.shape)
    print(test.shape)

```

ما ۲۷۴۸۶ توییت در مجموعه‌ی آموزشی و ۳۵۳۵ توییت در داده‌ی تست داریم.

```
(27481, 4)
(3534, 3)
```

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27481 entries, 0 to 27480
Data columns (total 4 columns):
textID           27481 non-null object
text             27480 non-null object
selected_text    27480 non-null object
sentiment        27481 non-null object
dtypes: object(4)
memory usage: 858.9+ KB
```

ما تنها یک مقدار null در داده‌ی آموزشی داریم. سطر شامل این مقدار را از داده‌ی آموزشی حذف می‌کنیم.

```
train.dropna(inplace=True)
```

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3534 entries, 0 to 3533
Data columns (total 3 columns):
textID       3534 non-null object
text         3534 non-null object
sentiment    3534 non-null object
dtypes: object(3)
memory usage: 83.0+ KB
```

هیچ مقدار null در داده‌ی تست وجود ندارد.

بررسی اکتشافی داده

```
train.head()
```

Out[8]:

	textID	text	selected_text	sentiment
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
2	088c60f138	my boss is bullying me...	bullying me	negative
3	9642c003ef	what interview! leave me alone	leave me alone	negative
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative

```
train.describe()
```

Out[9]:

	textID	text	selected_text	sentiment
count	27480	27480	27480	27480
unique	27480	27480	22463	3
top	609f4a0832	can`t wait to crack it open and no doubt will...	good	neutral
freq	1	1	199	11117

حال، بیاید نگاهی به توزیع توییت‌ها در داده‌ی تست بی‌اندازیم.

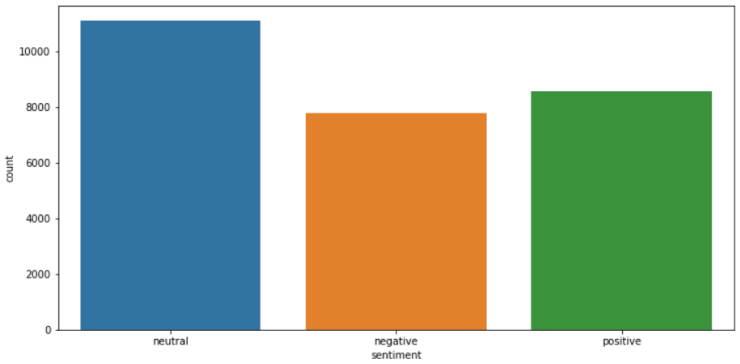
```
temp = train.groupby('sentiment').count()['text'].reset_index().
sort_values(by='text',ascending=False)
temp.style.background_gradient(cmap='Purples')
```

Out[10]:

	sentiment	text
1	neutral	11117
2	positive	8582
0	negative	7781

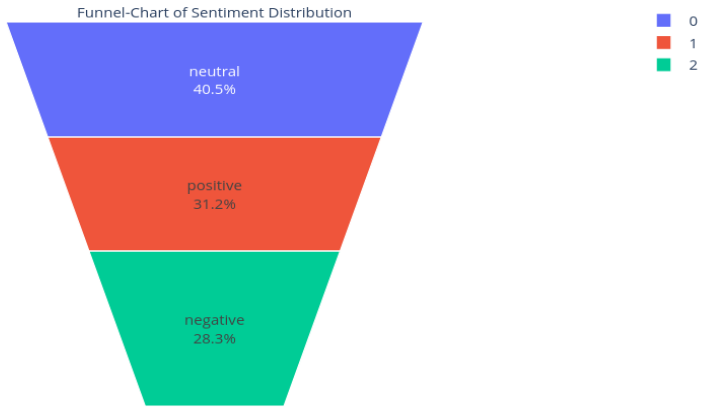
```
plt.figure(figsize=(12,6))
sns.countplot(x='sentiment',data=train)
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f99b80710b8>
```



برای مصورسازی بهتر، یک Funnel-chart از داده ترسیم می‌کنیم:

```
fig = go.Figure(go.Funnelarea(
text =temp.sentiment,
values = temp.text,
title = {"position": "top center", "text": "Funnel-Chart of Sentiment Distribution"}
))
fig.show()
```



مواردی که تا کنون از داده می‌دانیم:

- می‌دانیم selected_text زیرمجموعه‌یی از text است.
- می‌دانیم selected_text تنها یک بخش از text را شامل می‌شود. یعنی، از بخشی از یک عبارت و بخشی از یک عبارت دیگر را شامل نمی‌شود. برای مثال؛ اگر text برابر با:

«Spent the entire morning in a meeting w/ a vendor, and my boss was not happy w/ them. Lots of fun. I had other plans for my morning»

باشد؛ selected_text می‌تواند «my boss was not happy w/ them. Lots of fun» یا «my boss was not happy w/ them. Lots of fun» باشد، اما نمی‌تواند برابر با «Morning vendor, and my boss» باشد.

- با وجود گفتگوی کگلی که از طریق این لینک قابل دسترسی است، می‌دانیم که توییت‌های بی‌طرف، دارای ۹۷ درصد شباهت ژاکارد بین text و selected_text می‌باشد.
- همچنین با استفاده از گفتگویی که از طریق این لینک در اختیارمان قرار دارد، می‌دانیم selected_text در بعضی سطرها از میان یک کلمه شروع می‌شود، در نتیجه selected_text همیشه معنادار نیست. از آنجایی که ما نمی‌دانیم خروجی مجموعه تست هم این ناسازگاری را دارد یا نه، مطمئن نیستیم حذف علائم نگارشی رویکرد خوبی باشد.

تولید متافیچرها

از آنجایی که می‌خواهیم selected_text-که زیرمجموعه‌یی از text است-را پیش‌بینی کنیم، ویژگی‌هایی که تولیدشان سودمند خواهد بود عبارت است از:

- اختلاف بین تعداد کلمات در text و selected_text
- ضریب شباهت ژاکارد برای text و selected_text

تابع برای ضریب ژاکارد:

```
def jaccard(str1, str2):
  a = set(str1.lower().split())
  b = set(str2.lower().split())
  c = a.intersection(b)
  return float(len(c)) / (len(a) + len(b) - len(c))
```

به‌دست آوردن ضریب ژاکارد برای تمام سطرهای داده‌ی آموزشی:

```
results_jaccard=[]

for ind,row in train.iterrows():
  sentence1 = row.text
  sentence2 = row.selected_text

  jaccard_score = jaccard(sentence1,sentence2)
  results_jaccard.append([sentence1,sentence2,jaccard_score])
```

تبدیل لیست ضریب ژاکارد-که در سلول قبلی محاسبه شد-به دیتافریم و ادغام آن با داده‌ی آموزشی:

```
jaccard = pd.DataFrame(results_jaccard,columns=["text","selected_text",
    "jaccard_score"])
train = train.merge(jaccard,how='outer')
```

اضافه کردن ویژگی‌های موردنظر- Num_words_ST، Num_word_text و difference_in_words -به‌داده‌ی آموزشی

```
train['Num_words_ST'] = train['selected_text'].apply(lambda x:len(str(x
    ).split())) #Number Of words in Selected Text
train['Num_word_text'] = train['text'].apply(lambda x:len(str(x).split
    ())) #Number Of words in main text
train['difference_in_words'] = train['Num_word_text'] - train['
    Num_words_ST'] #Difference in Number of words text and Selected
    Text
```

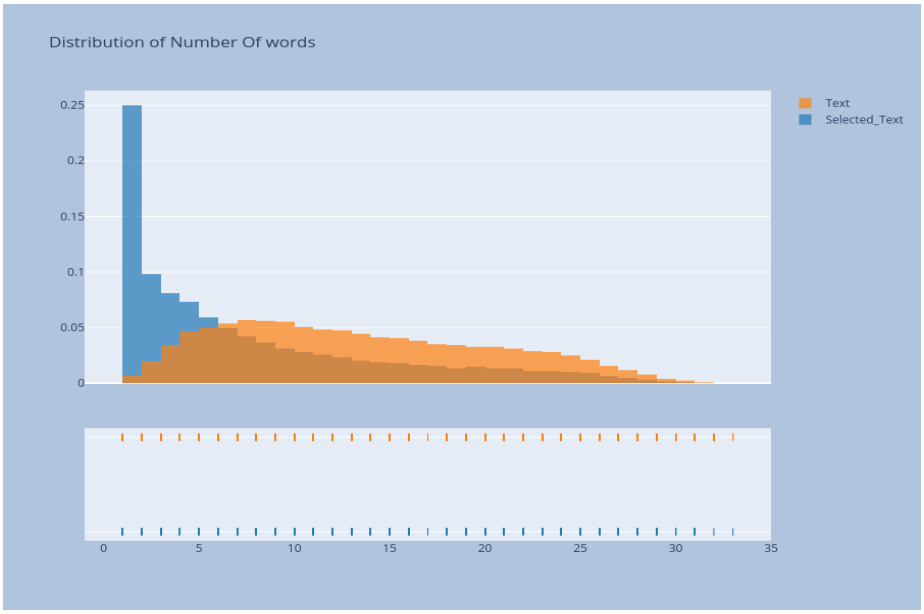
نمونه‌یی از داده‌ی آموزشی بعد از اضافه کردن متافیچرها:

Out[17]:

	textID	text	selected_text	sentiment	jaccard_score	Num_words_ST	Num_word_text	difference_in_words
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral	1.000000	7	7	0
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative	0.200000	2	10	8
2	088c60f138	my boss is bullying me...	bullying me	negative	0.166667	2	5	3
3	9642c003ef	what interview! leave me alone	leave me alone	negative	0.600000	3	5	2
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative	0.214286	3	14	11

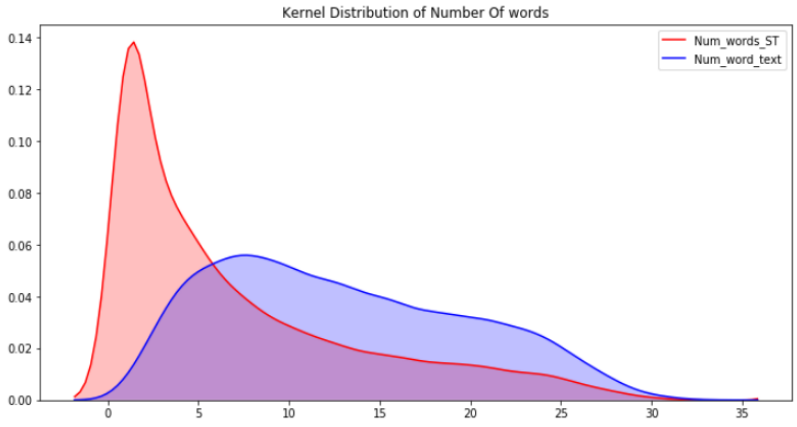
حال بیایید نگاهی به توزیع متافیچرها بی‌اندازیم.

```
hist_data = [train['Num_words_ST'],train['Num_word_text']]

group_labels = ['Selected_Text', 'Text']

# Create distplot with custom bin_size
fig = ff.create_distplot(hist_data, group_labels,show_curve=False)
fig.update_layout(title_text='Distribution of Number Of words')
fig.update_layout(
  autosize=False,
  width=900,
  height=700,
  paper_bgcolor="LightSteelBlue",
)
fig.show()

```



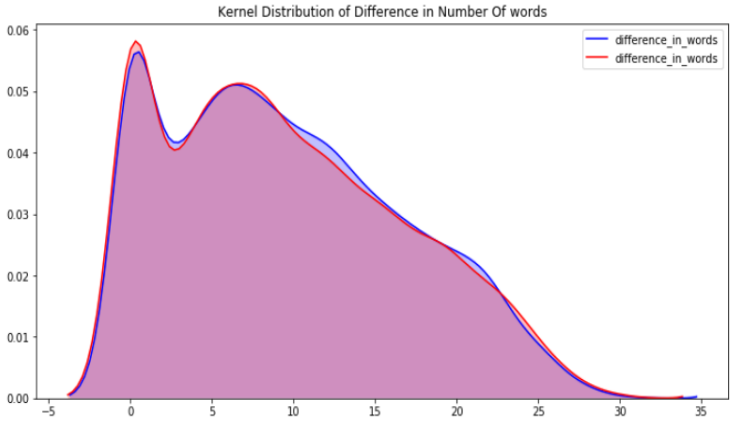
نمودار تعداد کلمات از این نظر حائز اهمیت است که توییت‌هایی که تعداد کلمات‌شان بیشتر از ۲۵ تا می‌باشد بسیار کم‌تر هستند، از این رو منحنی توزیع چوله به راست است.

```
p1=sns.kdeplot(train['Num_words_ST'], shade=True, color="r").set_title(
    'Kernel Distribution of Number Of words')
p1=sns.kdeplot(train['Num_word_text'], shade=True, color="b")
```



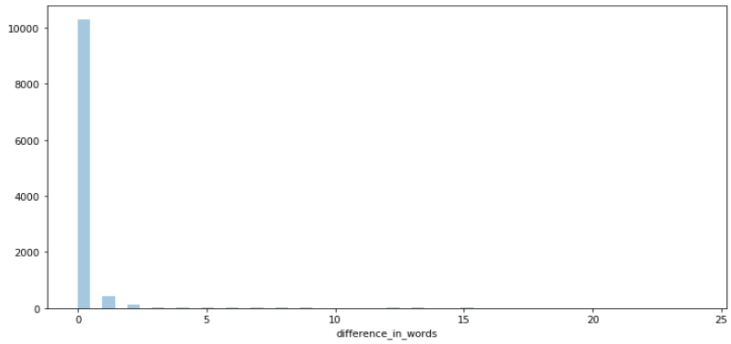
حالا مفید است نگاهی هم به توزیع اختلاف تعداد کلمات و ضریب ژاکارد در تمایل‌ها بی‌اندازیم.

```
plt.figure(figsize=(12,6))
p1=sns.kdeplot(train[train['sentiment']=='positive']['
    difference_in_words'], shade=True, color="b").set_title('Kernel
    Distribution of Difference in Number Of words')
p2=sns.kdeplot(train[train['sentiment']=='negative']['
    difference_in_words'], shade=True, color="r")

```




```
plt.figure(figsize=(12,6))
sns.distplot(train[train['sentiment']=='neutral'][['difference_in_words'
    ],kde=False)

```

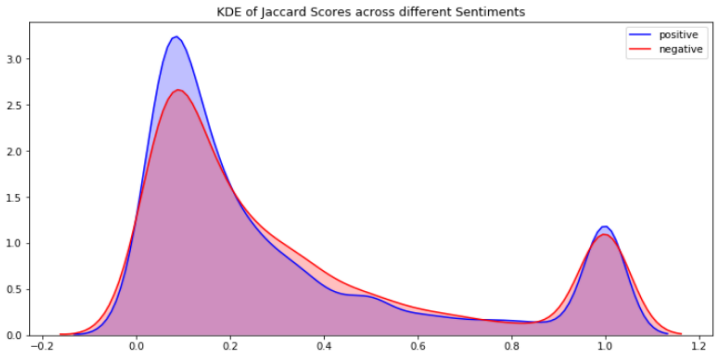
```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9968f22ac8>
```



قادر به ترسیم KDE Plot برای توییت‌های بی‌تمایل (Neutral) نیستیم، چراکه برای اکثر توییت‌های خنثی اختلاف میان تعداد کلمات در text و selected_text برابر صفر است.

```
plt.figure(figsize=(12,6))
p1=sns.kdeplot(train[train['sentiment']=='positive']['jaccard_score'],
    shade=True, color="b").set_title('KDE of Jaccard Scores across
    different Sentiments')
p2=sns.kdeplot(train[train['sentiment']=='negative']['jaccard_score'],
    shade=True, color="r")
plt.legend(labels=['positive','negative'])

```

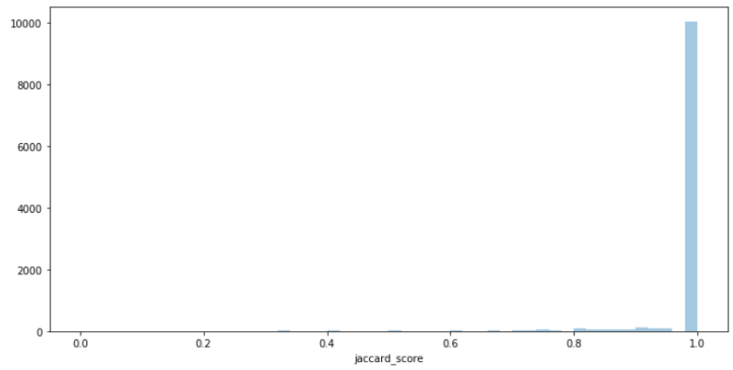
Out[22]:
```
<matplotlib.legend.Legend at 0x7f99682eac50>
```



به همان دلیل ذکر شده در مورد ترسیم KDE Plot برای اختلاف کلمات در توییت‌های خنثی، قادر به ترسیم KDE Plot برای ضریب ژاکارد در توییت‌های خنثی نیز نیستیم.

```
plt.figure(figsize=(12,6))
sns.distplot(train[train['sentiment']=='neutral']['jaccard_score'],kde=
    False)

```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f99682fce80>
```



ما می‌توانیم چند گرایش جالب را در اینجا ببینیم:

- توییت‌های مثبت و منفی کوتوز بالایی دارند، از این رو، توییت‌ها در دو ناحیه‌ی کم‌تراکم و پرتراکم متمرکز شده‌اند.
- توییت‌های خنثی مقدار کورتوز پایینی دارند و یک برآمدگی تراکم در نزدیکی مقادیر ۱ وجود دارد.

نتیجه‌گیری

- با مشاهده‌ی پلات ضریب ژاکارد، می‌توان دید برای توییت‌های مثبت و منفی، در نزدیکی ۱ برآمدگی وجود دارد. این یعنی، یک خوشه از توییت‌ها وجود دارد که در آن‌ها text و selected_text شباهت بالایی دارند. اگر بتوانیم آن خوشه‌ها را پیدا کنیم، با صرف نظر از کلاس‌های sentiment، می‌توانیم selected_text را پیش‌بینی کنیم.

بیایید بررسی کنیم که آیا می‌توان این خوشه‌ها را پیدا کرد:

```
k = train[train['Num_word_text']<=2]
k.groupby('sentiment').mean()['jaccard_score']
```

Out[25]:

```
sentiment
negative    0.788580
neutral     0.977805
positive    0.765700
Name: jaccard_score, dtype: float64
```

می‌توان دید شباهتی میان text و selected_text وجود دارد. بیایید با دقت بیشتر بررسی کنیم:

```
k[k['sentiment']=='positive']

```

واضح است که بیشتر مواقع، text و selected_text برابر هستند.

پاک‌سازی داده

حالا استخراج اطلاعات از کلمات توییت‌ها را آغاز خواهیم کرد. بیایید اول داده‌ی آموزشی را پاکسازی کنیم:

```
def clean_text(text):
  '''Make text lowercase, remove text in square brackets,remove links,
    remove punctuation
  and remove words containing numbers.'''
  text = str(text).lower()
  text = re.sub('\[.*?\]', '', text)
  text = re.sub('https?://\S+|www\.\S+', '', text)
  text = re.sub('<.*?>+', '', text)
  text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
  text = re.sub('\n', '', text)
```

Out[26]:

	textID	text	selected_text	sentiment	jaccard_score	Nu
68	fa2654e730	Chilliin	Chilliin	positive	1.0	1
80	bbbc46889b	THANK YYYYYYYYYOOOOOOOOOOUUUUU!	THANK YYYYYYYYYOOOOOOOOOOUUUUU!	positive	1.0	2
170	f3d95b57b1	good morning	good morning	positive	1.0	2
278	89d5b3f0b5	Thanks	Thanks	positive	1.0	1
429	a78ef3e0d0	Goodmorning	Goodmorning	positive	1.0	1
...	...	...	...	...	...	...
26689	e80c242d6a	Goodnight;	Goodnight;	positive	1.0	1
26725	aad244f37d	*hug*	*hug*	positive	1.0	1
26842	a46571fe12	congrats!	congrats!	positive	1.0	1
26959	49a942e9b1	Happy birthday.	Happy birthday.	positive	1.0	2
27292	47c474aaf1	Good choice	Good	positive	0.5	1

```
  text = re.sub('\w*\d\w*', '', text)
  return text

```

```
train['text'] = train['text'].apply(lambda x:clean_text(x))
train['selected_text'] = train['selected_text'].apply(lambda x:
    clean_text(x))

```

```
train.head()

```

Out[29]:

	textID	text	selected_text	sentiment	jaccard_score	Num_words_ST	Num_word_text	difference_in_words
0	cb774db0d1	id have responded if i were going	id have responded if i were going	neutral	1.000000	7	7	0
1	549e992a42	sooo sad i will miss you here in san diego	sooo sad	negative	0.200000	2	10	8
2	088c60f138	my boss is bullying me	bullying me	negative	0.166667	2	5	3
3	9642c003ef	what interview leave me alone	leave me alone	negative	0.600000	3	5	2
4	358bd9e861	sons of why couldnt they put them on the rel...	sons of	negative	0.214286	3	14	11

رایج‌ترین کلمات در selected_text

```
train['temp_list'] = train['selected_text'].apply(lambda x:str(x).split
    ())
top = Counter([item for sublist in train['temp_list'] for item in
    sublist])
temp = pd.DataFrame(top.most_common(20))
temp.columns = ['Common_words','count']
temp.style.background_gradient(cmap='Blues')

```

Out[30]:

	Common_words	count
0	i	7200
1	to	5305
2	the	4590
3	a	3538
4	my	2783
5	you	2624
6	and	2321
7	it	2158
8	is	2115
9	in	1986
10	for	1854
11	im	1676
12	of	1638
13	me	1540
14	on	1488
15	so	1410
16	have	1345
17	that	1297
18	but	1267
19	good	1251

مشاهده می‌کنید که stopword ها را حذف نکرده‌ایم (برای نمونه: کلمه‌ی to).

```
def remove_stopword(x):
  return [y for y in x if y not in stopwords.words('english')]
train['temp_list'] = train['temp_list'].apply(lambda x:remove_stopword(
    x))
```

Out[33]:

	Common_words	count
1	good	1251
2	day	1058
3	love	909
4	happy	852
5	like	774
6	get	772
7	dont	765
8	go	700
9	cant	613
10	work	612
11	going	592
12	today	564
13	got	558
14	one	538
15	time	534
16	thanks	532
17	lol	528
18	really	520
19	u	519

رایج‌ترین کلمات در **text**

```
train['temp_list1'] = train['text'].apply(lambda x:str(x).split()) #
    List of words in every row for text
train['temp_list1'] = train['temp_list1'].apply(lambda x:
    remove_stopword(x)) #Removing Stopwords
```

```
top = Counter([item for sublist in train['temp_list1'] for item in
    sublist])
temp = pd.DataFrame(top.most_common(25))
temp = temp.iloc[1:,:]
temp.columns = ['Common_words','count']
temp.style.background_gradient(cmap='Blues')
```

Out[36]:

	Common_words	count
1	day	2044
2	good	1549
3	get	1426
4	like	1346
5	go	1267
6	dont	1200
7	love	1122
8	work	1112
9	going	1096
10	today	1096
11	got	1072
12	cant	1020
13	happy	976
14	one	971
15	lol	948
16	time	942
17	know	930
18	u	923
19	really	908
20	back	891
21	see	797
22	well	744
23	new	740
24	night	737

رایج‌ترین کلمات در تمایل‌ها

```
Positive_sent = train[train['sentiment']=='positive']
Negative_sent = train[train['sentiment']=='negative']
Neutral_sent = train[train['sentiment']=='neutral']
```

رایج‌ترین کلمات در توییت‌های مثبت

```
#MosT common positive words
top = Counter([item for sublist in Positive_sent['temp_list'] for item
    in sublist])
temp_positive = pd.DataFrame(top.most_common(20))
temp_positive.columns = ['Common_words','count']
temp_positive.style.background_gradient(cmap='Greens')
```

Out[39]:

	Common_words	count
0	good	826
1	happy	730
2	love	697
3	day	456
4	thanks	439
5	great	364
6	fun	287
7	nice	267
8	mothers	259
9	hope	245
10	awesome	232
11	im	185
12	thank	180
13	like	167
14	best	154
15	wish	152
16	amazing	135
17	really	128
18	better	125
19	cool	119

رایج‌ترین کلمات در توییت‌های منفی

```
#MosT common negative words
top = Counter([item for sublist in Negative_sent['temp_list'] for item
    in sublist])
temp_negative = pd.DataFrame(top.most_common(20))
temp_negative = temp_negative.iloc[1:,:]
temp_negative.columns = ['Common_words','count']
temp_negative.style.background_gradient(cmap='Reds')

```

Out[41]:

	Common_words	count
1	miss	358
2	sad	343
3	sorry	300
4	bad	246
5	hate	230
6	dont	221
7	cant	201
8	sick	166
9	like	162
10	sucks	159
11	feel	158
12	tired	144
13	really	137
14	good	127
15	bored	115
16	day	110
17	hurts	108
18	work	99
19	get	97

رایج‌ترین کلمات در توییت‌های خنثی

```
#MosT common Neutral words
top = Counter([item for sublist in Neutral_sent['temp_list'] for item
    in sublist])
temp_neutral = pd.DataFrame(top.most_common(20))
temp_neutral = temp_neutral.loc[1:,:]
temp_neutral.columns = ['Common_words','count']
temp_neutral.style.background_gradient(cmap='Reds')
```

Out[41]:

	Common_words	count
1	miss	358
2	sad	343
3	sorry	300
4	bad	246
5	hate	230
6	dont	221
7	cant	201
8	sick	166
9	like	162
10	sucks	159
11	feel	158
12	tired	144
13	really	137
14	good	127
15	bored	115
16	day	110
17	hurts	108
18	work	99
19	get	97

- مشاهده می‌شود کلماتی همچون get، go، dont، got، u، cant، lol و like در هر سه کلاس تمایل مشترک هستند، نکته‌ی جالب اینجاست که کلماتی همچون: cant و dont طبیعتا منفی هستند و کلمه‌یی همچون: lol، طبیعتا مثبت است. آیا داده‌ی ما اشتباه برچسب خورده؟-بعد از آنالیز N-gram دید بهتری به این موضوع خواهیم داشت.
- بررسی کلمات یکتا در تمایل‌های مختلف جالب است.

کلمات یکتا در هر تمایل

ما کلمات متمایز را برای هر تمایل-با استفاده از تابع words_unique-به ترتیب زیر بررسی خواهیم کرد:

- مثبت
- منفی
- خنثی

```
raw_text = [word for word_list in train['temp_list1'] for word in
    word_list]

```

توییت‌های مثبت

```
Unique_Positive= words_unique('positive', 20, raw_text)
print("The top 20 unique words in Positive Tweets are:")
Unique_Positive.style.background_gradient(cmap='Greens')

```

```
The top 20 unique words in Positive Tweets are:
```

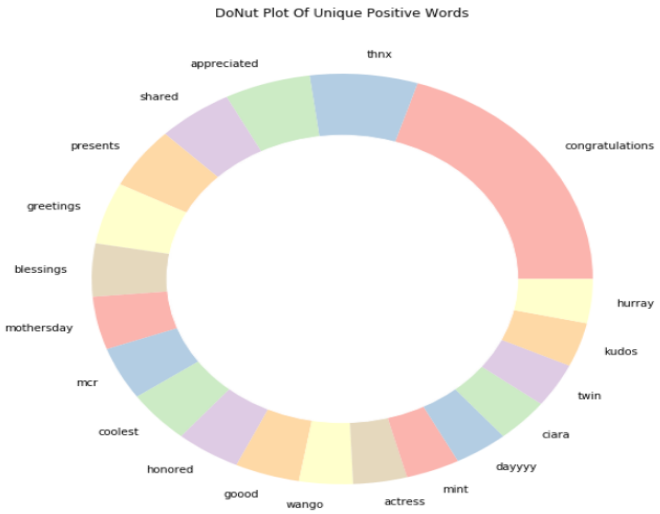
Out[48]:

	words	count
0	congratulations	29
1	thnx	10
2	appreciated	8
3	shared	7
4	presents	7
5	greetings	7
6	blessings	6
7	mothersday	6
8	mcr	6
9	coolest	6
10	honored	6
11	goood	6
12	wango	5
13	actress	5
14	mint	5
15	dayyyy	5
16	ciara	5
17	twin	5
18	kudos	5
19	hurray	5

```
fig = px.treemap(Unique_Positive, path=['words'], values='count',title=
    'Tree Of Unique Positive Words')
fig.show()
```


```
from palettable.colorbrewer.qualitative import Pastel1_7
plt.figure(figsize=(16,10))
my_circle=plt.Circle((0,0), 0.7, color='white')
plt.pie(Unique_Positive['count'], labels=Unique_Positive.words, colors=
    Pastel1_7.hex_colors)
p=plt.gcf()
p.gca().add_artist(my_circle)
```

```
plt.title('DoNut Plot Of Unique Positive Words')
plt.show()

```



توییت‌های منفی

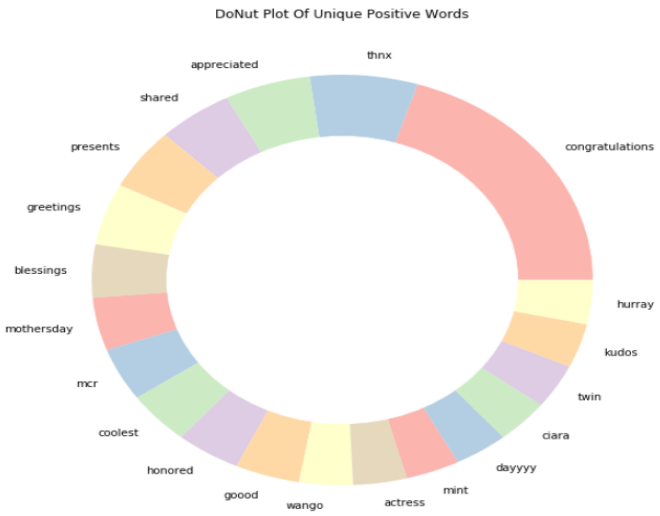
```
Unique_Negative= words_unique('negative', 10, raw_text)
print("The top 10 unique words in Negative Tweets are:")
Unique_Negative.style.background_gradient(cmap='Reds')

```

```
The top 20 unique words in Positive Tweets are:
```

Out[48]:

	words	count
0	congratulations	29
1	thnx	10
2	appreciated	8
3	shared	7
4	presents	7
5	greetings	7
6	blessings	6
7	mothersday	6
8	mcr	6
9	coolest	6
10	honored	6
11	goood	6
12	wango	5
13	actress	5
14	mint	5
15	dayyyy	5
16	ciara	5
17	twin	5
18	kudos	5
19	hurray	5

```
from palettable.colorbrewer.qualitative import Pastel1_7
plt.figure(figsize=(16,10))
my_circle=plt.Circle((0,0), 0.7, color='white')
plt.rcParams['text.color'] = 'black'
plt.pie(Unique_Negative['count'], labels=Unique_Negative.words, colors=
    Pastel1_7.hex_colors)
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.title('DoNut Plot Of Unique Negative Words')
plt.show()

```



توییت‌های خنثی

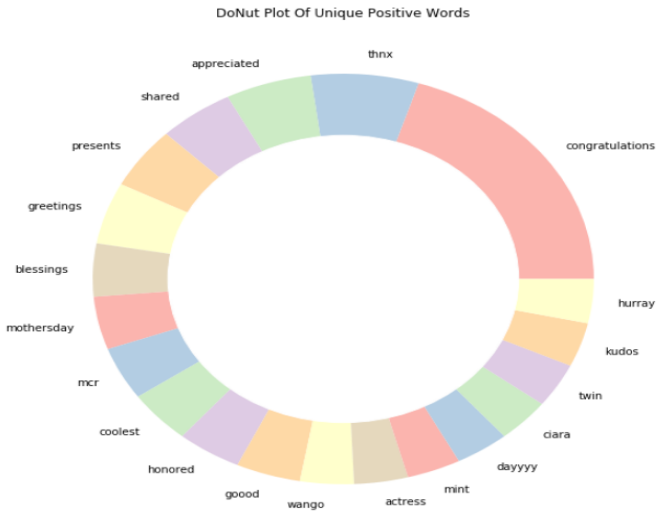
```
Unique_Neutral= words_unique('neutral', 10, raw_text)
print("The top 10 unique words in Neutral Tweets are:")
Unique_Neutral.style.background_gradient(cmap='Oranges')

```

```
The top 20 unique words in Positive Tweets are:
```

Out[48]:

	words	count
0	congratulations	29
1	thnx	10
2	appreciated	8
3	shared	7
4	presents	7
5	greetings	7
6	blessings	6
7	mothersday	6
8	mcr	6
9	coolest	6
10	honored	6
11	goood	6
12	wango	5
13	actress	5
14	mint	5
15	dayyyy	5
16	ciara	5
17	twin	5
18	kudos	5
19	hurray	5

```
from palettable.colorbrewer.qualitative import Pastel1_7
plt.figure(figsize=(16,10))
my_circle=plt.Circle((0,0), 0.7, color='white')
plt.pie(Unique_Neutral['count'], labels=Unique_Neutral.words, colors=
    Pastel1_7.hex_colors)
p=plt.gcf()
p.gca().add_artist(my_circle)
plt.title('DoNut Plot Of Unique Neutral Words')
plt.show()

```



حال که کلمات متمایز را بررسی کردیم، دید به مراتب واضح‌تری از داده داریم. این کلمات یکتا از تعیین کنندگان قوی sentiment برای توییت‌ها می‌باشند.

مدل‌سازی

این مسئله می‌تواند به شکل‌های زیر مدل شود:

- Named Entity Recognition
- Question-Answering
- من یک رویکرد دیگر-که Nick در کرنل‌اش به اشتراک گذاشته-یافتم که از مفهوم Gini Impurity استفاده کرده تا به کلمات درون توییت‌ها وزن بدهد، سپس پیش‌بینی بر اساس آن وزن‌ها اتفاق می‌افتد. می‌توانید از طریق این لینک به کرنل رجوع کنید.
- یک ایده‌ی متفاوت‌تر از طریق این لینک قابل دسترسی است.
- یک رویکرد مفید دیگر: لینک

منابع

- برای مدلینگ مسئله به‌عنوان مسئله‌ی NER: لینک
- برای مدلینگ مسئله به‌عنوان مسئله‌ی Question-Answering: لینک

مدل کردن مسئله به‌عنوان مسئله‌ی NER

NER یک استاندارد برای مسئله‌های **پردازش زبان طبیعی** است، که شامل استخراج موجودیت‌های اسم‌دار-مردم، مکان‌ها، سازمان‌ها و غیره-از داده‌ی متنی و طبقه‌بندی آن‌ها در مجموعه‌ای از دسته‌ها می‌باشد. برای درک بهتر NER به این لینک مراجعه کنید. مواردی در مورد مدل‌مان:

- ما برای توییت‌های بی‌تمایل، از text به جای selected_text استفاده خواهیم کرد-به‌دلیل ضریب بالای ژاکارد در آن‌ها.
- همچنین در توییت‌هایی که کم‌تر از سه کلمه دارند هم text را به جای selected_text به‌کار خواهیم برد.
- دو مدل را برای توییت‌های مثبت و توییت‌های منفی آموزش داده می‌شوند.
- داده پیش‌پردازش نخواهد شد، چرا که selected_text تنها شامل متن خام می‌شود.

خواندن داده‌های آموزشی و آزمایشی

```
df_train = pd.read_csv('/kaggle/input/tweet-sentiment-extraction/train.csv')
df_test = pd.read_csv('/kaggle/input/tweet-sentiment-extraction/test.csv')
df_submission = pd.read_csv('/kaggle/input/tweet-sentiment-extraction/sample_submission.csv')
```

رایج‌ترین کلمات در توییت‌های منفی

```
df_train['Num_words_text'] = df_train['text'].apply(lambda x:len(str(x).split())) #Number Of words in main Text in train set
```

انتخاب توییت‌های که بیشتر از دو کلمه را شامل می‌شوند

```
df_train = df_train[df_train['Num_words_text']>=3]
```

تابعی برای ذخیره کردن مدل در مسیر داده شده

```
def save_model(output_dir, nlp, new_model_name):
  ''' This Function Saves model to
  given output directory'''

  output_dir = f'../working/{output_dir}'
  if output_dir is not None:
  if not os.path.exists(output_dir):
  os.makedirs(output_dir)
  nlp.meta["name"] = new_model_name
  nlp.to_disk(output_dir)
  print("Saved model to", output_dir)

```

تابعی برای دریافت کردن مسیر مدل‌ها

```
def get_model_out_path(sentiment):
  '''
  Returns Model output path
```

```
  '''
  model_out_path = None
  if sentiment == 'positive':
  model_out_path = 'models/model_pos'
  elif sentiment == 'negative':
  model_out_path = 'models/model_neg'
  return model_out_path

```

تابعی برای دریافت کردن داده با فرمتی که برای Spacy NER مناسب است

```
def get_training_data(sentiment):
  '''
  Returns Trainong data in the format needed to train spacy NER
  '''
  train_data = []
  for index, row in df_train.iterrows():
  if row.sentiment == sentiment:
  selected_text = row.selected_text
  text = row.text
  start = text.find(selected_text)
  end = start + len(selected_text)
  train_data.append((text, {"entities": [[start, end, 'selected_text'
    ]]}))
  return train_data

```

آموزش مدل‌ها

```
sentiment = 'positive'

train_data = get_training_data(sentiment)
model_path = get_model_out_path(sentiment)
# For DEmo Purposes I have taken 3 iterations you can train the model
    as you want
train(train_data, model_path, n_iter=3, model=None)

```

```
sentiment = 'negative'

train_data = get_training_data(sentiment)
model_path = get_model_out_path(sentiment)

train(train_data, model_path, n_iter=3, model=None)

```

پیشبینی با مدل‌های آموزش داده شده

```
def predict_entities(text, model):
  doc = model(text)
  ent_array = []
```

```
  for ent in doc.ents:
  start = text.find(ent.text)
  end = start + len(ent.text)
  new_int = [start, end, ent.label_]
  if new_int not in ent_array:
  ent_array.append([start, end, ent.label_])
  selected_text = text[ent_array[0][0]: ent_array[0][1]] if len(
    ent_array) > 0 else text
  return selected_text

```

```
MODELS_BASE_PATH = '../input/tse-spacy-model/models/'

if MODELS_BASE_PATH is not None:
  print("Loading Models  from ", MODELS_BASE_PATH)
  model_pos = spacy.load(MODELS_BASE_PATH + 'model_pos')
  model_neg = spacy.load(MODELS_BASE_PATH + 'model_neg')

  for index, row in df_test.iterrows():
    text = row.text
    output_str = ""
    if row.sentiment == 'neutral' or len(text.split()) <= 2:
    selected_texts.append(text)
    elif row.sentiment == 'positive':
    selected_texts.append(predict_entities(text, model_pos))
    else:
    selected_texts.append(predict_entities(text, model_neg))
df_test['selected_text'] = selected_texts
```

```
Loading Models  from  ../input/tse-spacy-model/models/
```

```
df_submission['selected_text'] = df_test['selected_text']
df_submission.to_csv("submission.csv", index=False)
display(df_submission.head(10))

```

	textID	selected_text
0	f87dea47db	Last session of the day http://twitpic.com/67ezh
1	96d74cb729	exciting
2	eee518ae67	Recession
3	01082688c6	happy bday!
4	33987a8ee5	I like it!!
5	726e501993	visitors!
6	261932614e	HATES
7	afa11da83f	blocked
8	e64208b4ef	and within a short time of the last clue all ...
9	37bcad24ca	What did you get? My day is alright.. haven`...

سخن پایانی

کگل همیشه فرصت زیادی برای رقابت‌ها در نظر می‌گیرد، اینگونه شرکت‌کنندگان می‌توانند یاد بگیرند و رشد کنند. همانطور که عهد کرده بودیم یک مدل همراه با توضیحات‌اش را ارائه کردیم؛ برای مطالعه بیشتر، می‌توانید مستندات spacy را در این کرنل مطالعه کنید.

References

[1] A simple solution using only word counts

[2] Twitter sentiment Extaction-Analysis, EDA and Model

[3] NER - training using spacy (Ensemble)

[4] Question-Answering Starter pack