05_sentiment_analysis_twitter

September 29, 2021

1 Text classification and sentiment analysis: Twitter

Once text data has been converted into numerical features using the natural language processing techniques discussed in the previous sections, text classification works just like any other classification task.

In this notebook, we will apply these preprocessing technique to news articles, product reviews, and Twitter data and teach various classifiers to predict discrete news categories, review scores, and sentiment polarity.

1.1 Imports

[1]:
```
%matplotlib inline
import warnings
from collections import Counter, OrderedDict
from pathlib import Path

import numpy as np
import pandas as pd
from pandas.io.json import json_normalize
import pyarrow as pa
import pyarrow.parquet as pq
from fastparquet import ParquetFile
from scipy import sparse
from scipy.spatial.distance import pdist, squareform

# Visualization
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter, ScalarFormatter
import seaborn as sns

# spacy, textblob and nltk for language processing
from textblob import TextBlob, Word

# sklearn for feature extraction & modeling
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score, roc_curve, accuracy_score,␣
↪confusion_matrix
from sklearn.externals import joblib

import lightgbm as lgb

import json
from time import clock, time
```

[2]:
```
plt.style.use('fivethirtyeight')
warnings.filterwarnings('ignore')
```

1.2 Twitter Sentiment

We use a dataset that contains 1.6 million training and 350 test tweets from 2009 with algorithmically assigned binary positive and negative sentiment scores that are fairly evenly split.

Download the data from here.

Extract the content of the compressed file, move to 'data/sentiment140/' and rename the files: - `training.1600000.processed.noemoticon.csv` to `train.csv`, and - `testdata.manual.2009.06.14.csv` to `test.csv`

- 0 - the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive); training data has no neutral tweets
- 1 - the id of the tweet (2087)
- 2 - the date of the tweet (Sat May 16 23:58:44 UTC 2009)
- 3 - the query (lyx). If there is no query, then this value is NO_QUERY. (only test data uses query)
- 4 - the user that tweeted (robotickilldozr)
- 5 - the text of the tweet (Lyx is cool)

1.2.1 Read train/test data

We move the data to the faster parqu3et

[3]:
```
names = ['polarity', 'id', 'date', 'query', 'user', 'text']
train = (pd.read_csv('data/sentiment140/train.csv',
                     low_memory=False,
                     encoding='latin1',
                     header=None,
                     names=names,
                     parse_dates=['date'])
         .drop(['id', 'query'], axis=1)
         .drop_duplicates(subset=['polarity', 'text']))

train = train[train.text.str.len()<=140]
train.polarity = (train.polarity>0).astype(int)
```

[4]:
```
train.info(null_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1566668 entries, 0 to 1599999
Data columns (total 4 columns):
polarity    1566668 non-null int64
date        1566668 non-null datetime64[ns]
user        1566668 non-null object
text        1566668 non-null object
dtypes: datetime64[ns](1), int64(1), object(2)
memory usage: 59.8+ MB
```

[5]:
```
train.to_parquet('data/sentiment140/train.parquet')
```

[7]:
```
test = (pd.read_csv('data/sentiment140/test.csv',
                    low_memory=False,
                    encoding='latin1',
                    header=None,
                    names=names,
                    parse_dates=['date'])
        .drop(['id', 'query'], axis=1)
        .drop_duplicates(subset=['polarity', 'text']))
test = test[(test.text.str.len()<=140) & (test.polarity.isin([0,4]))]
```

[8]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 354 entries, 0 to 497
Data columns (total 4 columns):
polarity    354 non-null int64
date        354 non-null datetime64[ns, UTC]
user        354 non-null object
text        354 non-null object
dtypes: datetime64[ns, UTC](1), int64(1), object(2)
memory usage: 13.8+ KB
```

[9]:
```
test.to_parquet('data/sentiment140/test.parquet')
```

[10]:
```
train = pd.read_parquet('data/sentiment140/train.parquet')
test = pd.read_parquet('data/sentiment140/test.parquet')
```

1.2.2 Explore data

[11]:
```
train.head()
```

[11]:

	polarity	date	user	\
0	0	2009-04-06 22:19:45	_TheSpecialOne_	

```
1         0 2009-04-06 22:19:49    scotthamilton
2         0 2009-04-06 22:19:53         mattycus
3         0 2009-04-06 22:19:57          ElleCTF
4         0 2009-04-06 22:19:57            Karoli

                                                  text
0  @switchfoot http://twitpic.com/2y1zl - Awww, t…
1  is upset that he can't update his Facebook by …
2  @Kenichan I dived many times for the ball. Man…
3    my whole body feels itchy and like its on fire
4  @nationwideclass no, it's not behaving at all…
```

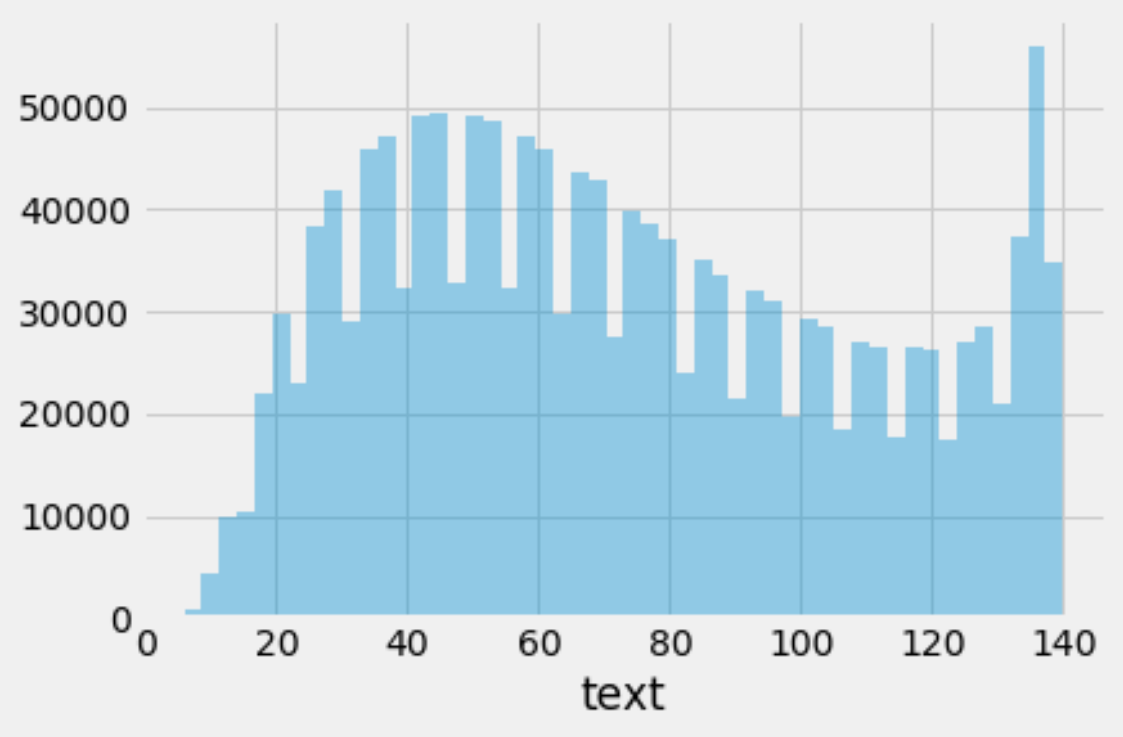
[12]:
```
train.polarity = (train.polarity>0).astype(int)
train.polarity.value_counts()
```

[12]:
```
1    784335
0    782333
Name: polarity, dtype: int64
```

[13]:
```
test.polarity = (test.polarity>0).astype(int)
test.polarity.value_counts()
```

[13]:
```
1    180
0    174
Name: polarity, dtype: int64
```

[14]:
```
sns.distplot(train.text.str.len(), kde=False);
```



[15]:
```
train.date.describe()
```

[15]:
```
count                 1566668
unique                 765666
top       2009-06-15 12:53:14
freq                       20
first     2009-04-06 22:19:45
last      2009-06-25 10:28:31
Name: date, dtype: object
```

[16]:
```
train.user.nunique()
```

[16]:
```
650606
```

[17]:
```
train.user.value_counts().head(10)
```

[17]:
```
lost_dog           549
webwoke            341
SallytheShizzle    276
VioletsCRUK        275
mcraddictal        274
tsarnick           247
what_bugs_u        246
Karen230683        237
```

```
DarkPiano          232
SongoftheOss       226
Name: user, dtype: int64
```

1.2.3 Create text vectorizer

We create a document-term matrix with 934 tokens as follows:

[18]:
```
vectorizer = CountVectorizer(min_df=.001, max_df=.8, stop_words='english')
train_dtm = vectorizer.fit_transform(train.text)
```

[19]:
```
train_dtm
```

[19]:
```
<1566668x934 sparse matrix of type '<class 'numpy.int64'>'
        with 6332930 stored elements in Compressed Sparse Row format>
```

[20]:
```
test_dtm = vectorizer.transform(test.text)
```

1.2.4 Train Naive Bayes Classifier

[21]:
```
nb = MultinomialNB()
nb.fit(train_dtm, train.polarity)
```

[21]:
```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

1.2.5 Predict Test Polarity

[22]:
```
predicted_polarity = nb.predict(test_dtm)
```

1.2.6 Evaluate Results

[24]:
```
accuracy_score(test.polarity, predicted_polarity)
```

[24]:
```
0.7768361581920904
```

1.2.7 TextBlob for Sentiment Analysis

[25]:
```
sample_positive = train.text.loc[256332]
print(sample_positive)
parsed_positive = TextBlob(sample_positive)
parsed_positive.polarity
```

```
Ok its cake and ice cream time! Ha! See what I'm talking about! The temptation
is there!
```

[25]:
```
1.0
```

[26]:
```
sample_negative = train.text.loc[636079]
print(sample_negative)
parsed_negative = TextBlob(sample_negative)
parsed_negative.polarity
```

```
i hate this place
```

[26]:
```
-0.8
```

[27]:
```
def estimate_polarity(text):
    return TextBlob(text).sentiment.polarity
```

[28]:
```
train[['text']].sample(10).assign(sentiment=lambda x: x.text.
↪apply(estimate_polarity)).sort_values('sentiment')
```

[28]:

	text	sentiment
286878	Waiting to board the plane. Frustrated that Sa…	-0.141667
608505	I FEEL SO LOST RIGHT NOW. wtf..	-0.107143
1049211	Chocolate cookies with cocoa nibs and lime rec…	0.000000
1054879	@GrahamNelson You're a cubs fan?! Oh boy. http…	0.000000
261809	work tomorrow. goodbye weekend	0.000000
776057	damn mood off..leavimg m dear home..hyderabad…	0.000000
1296985	is packing her bag and wants to dive into the …	0.066667
1180514	yay new moon is a trending topic on twitter	0.136364
350779	Forget same page… I think I'm reading a whol…	0.175000
425694	@tommcfly you would get on really well with my…	0.200000

1.2.8 Compare with TextBlob Polarity Score

We also obtain TextBlob sentiment scores for the tweets and note (see left panel in below figure) that positive test tweets receive a significantly higher sentiment estimate. We then use the MultinomialNB 's model .predict_proba() method to compute predicted probabilities and compare both models using the respective Area Under the Curve (see right panel below).

[29]:
```
test['sentiment'] = test.text.apply(estimate_polarity)
```

[30]:
```
accuracy_score(test.polarity, (test.sentiment>0).astype(int))
```

[30]:
```
0.7429378531073446
```

ROC AUC Scores

[31]:
```
roc_auc_score(y_true=test.polarity, y_score=test.sentiment)
```

[31]:
```
0.8254948914431672
```

[32]:
```
roc_auc_score(y_true=test.polarity, y_score=nb.predict_proba(test_dtm)[:, 1])
```

[32]:
```
0.848595146871009
```

[33]:
```
fpr_tb, tpr_tb, _ = roc_curve(y_true=test.polarity, y_score=test.sentiment)
roc_tb = pd.Series(tpr_tb, index=fpr_tb)
fpr_nb, tpr_nb, _ = roc_curve(y_true=test.polarity, y_score=nb.
 ↪predict_proba(test_dtm)[:, 1])
roc_nb = pd.Series(tpr_nb, index=fpr_nb)
```

The Naive Bayes model outperforms TextBlob in this case.

[34]:
```
fig, axes = plt.subplots(ncols=2, figsize=(14, 6))
sns.boxplot(x='polarity', y='sentiment', data=test, ax=axes[0])
axes[0].set_title('TextBlob Sentiment Scores')
roc_nb.plot(ax=axes[1], label='Naive Bayes', legend=True, lw=1, title='ROC␣
 ↪Curves')
roc_tb.plot(ax=axes[1], label='TextBlob', legend=True, lw=1)
fig.tight_layout();
```