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]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
%matplotlib inline

from pathlib import Path
import numpy as np
import pandas as pd

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns

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

# sklearn for feature extraction & modeling
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score, roc_curve, accuracy_score
```

[3]:
```
sns.set_style('white')
```

1.2 Twitter Sentiment

1.2.1 Download the data

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.

Follow the instructions to create the dataset.

- 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)

1.2.2 Read and preprocess train/test data

```
[4]: data_path = Path('..', 'data', 'sentiment140')
     if not data_path.exists():
         data_path.mkdir(parents=True)
```

```
[5]: names = ['polarity', 'id', 'date', 'query', 'user', 'text']
```

Take a few preprocessing steps: - remove tweets above the legal (at the time) length of 140 characters, - binarize polarity, and - move the data to the faster parquet format.

```
[6]: def load_train_data():
         parquet_file = data_path / 'train.parquet'
         if not parquet_file.exists():
             df = (pd.read_csv(data_path / 'train.csv',
                               low_memory=False,
                               encoding='latin1',
                               header=None,
                               names=names,
                               parse_dates=['date'])
                   .drop(['id', 'query'], axis=1)
                   .drop_duplicates(subset=['polarity', 'text']))
             df = df[df.text.str.len() <= 140]
             df.polarity = (df.polarity > 0).astype(int)
             df.to_parquet(parquet_file)
             return df
         else:
             return pd.read_parquet(parquet_file)
```

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

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

[8]:
```
def load_test_data():
    parquet_file = data_path / 'test.parquet'
    if not parquet_file.exists():
        df = (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']))
        df = df[(df.text.str.len() <= 140) &
                (df.polarity.isin([0, 4]))]
        df.to_parquet(parquet_file)
        return df
    else:
        return pd.read_parquet(parquet_file)
```

[9]:
```
test = load_test_data()
test.info(null_counts=True)
```

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

1.2.3 Explore data

[10]:
```
train.head()
```

[10]:
```
   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…
```

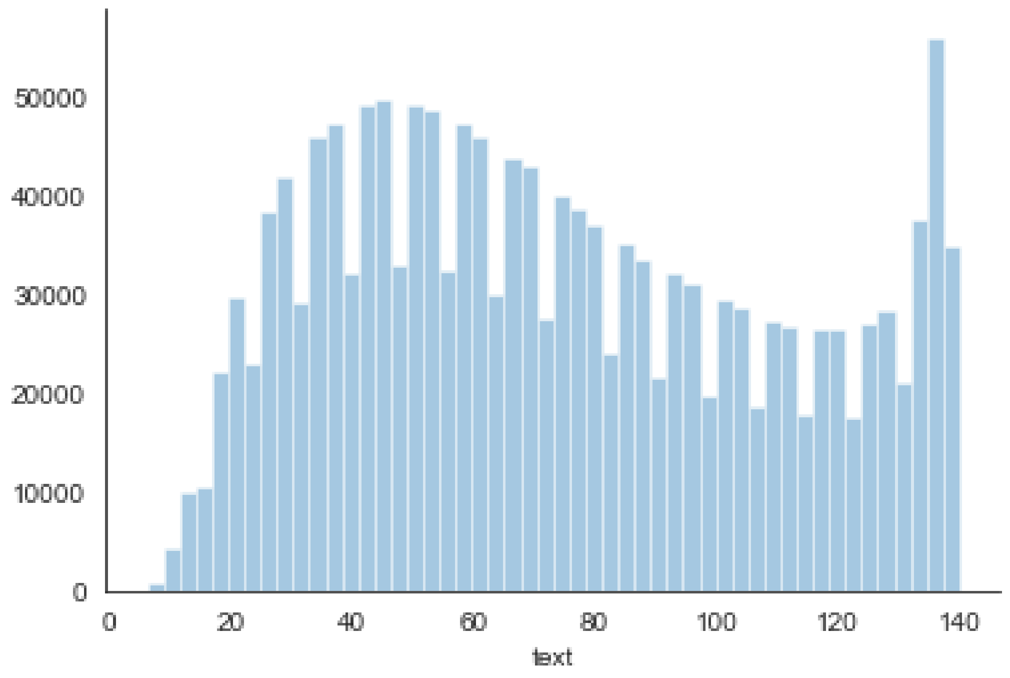
[11]:
```
train.polarity = (train.polarity>0).astype(int)
train.polarity.value_counts()
```

[11]:
```
1    784335
0    782333
Name: polarity, dtype: int64
```

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

[12]:
```
1    180
0    174
Name: polarity, dtype: int64
```

[13]:
```
sns.distplot(train.text.str.len(), kde=False)
sns.despine();
```



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

```
[14]: 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
```

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

```
[15]: 650606
```

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

```
[16]: 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
```

1.2.4 Create text vectorizer

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

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

```
[18]: train_dtm
```

```
[18]: <1566668x934 sparse matrix of type '<class 'numpy.int64'>'
              with 6332930 stored elements in Compressed Sparse Row format>
```

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

1.2.5 Train Naive Bayes Classifier

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

```
[20]: MultinomialNB()
```

1.2.6 Predict Test Polarity

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

1.2.7 Evaluate Results

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

```
[22]: 0.7768361581920904
```

1.2.8 TextBlob for Sentiment Analysis

```
[23]: 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!
```

```
[23]: 1.0
```

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

```
       i hate this place
```

```
[24]: -0.8
```

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

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

```
[26]:                                                        text  sentiment
      392473   No one will speak to me on this  Seems useless…    -0.5000
      492394   Fuck pacsun for not having any smalls in anyth…    -0.4000
      613144   I'm absolutely JOYFUL that Shahid Afridi made …    -0.1500
      1189887  I tell you something, I think you'll understan…     0.0000
      1001446  OWWW! Hurt myself. Keno ftw! Taking her up on …     0.0000
      92768    @AlexaNDYE Yup, didn't manage to dodge a 12 ho…     0.0000
      902549   finished watching the movie 'mirrors'. I liked…     0.2875
      1578586  The BBC (R4) will 'Keep in touch with Demotix …     0.4000
      1441092  Well everyone, I'm going to bed, mighty night …     0.4000
      409953   @2kutekreations Nope… no chocolate.    i rea…     0.4750
```

1.2.9 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).

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

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

```
[28]: 0.7429378531073446
```

ROC AUC Scores

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

```
[29]: 0.8254948914431672
```

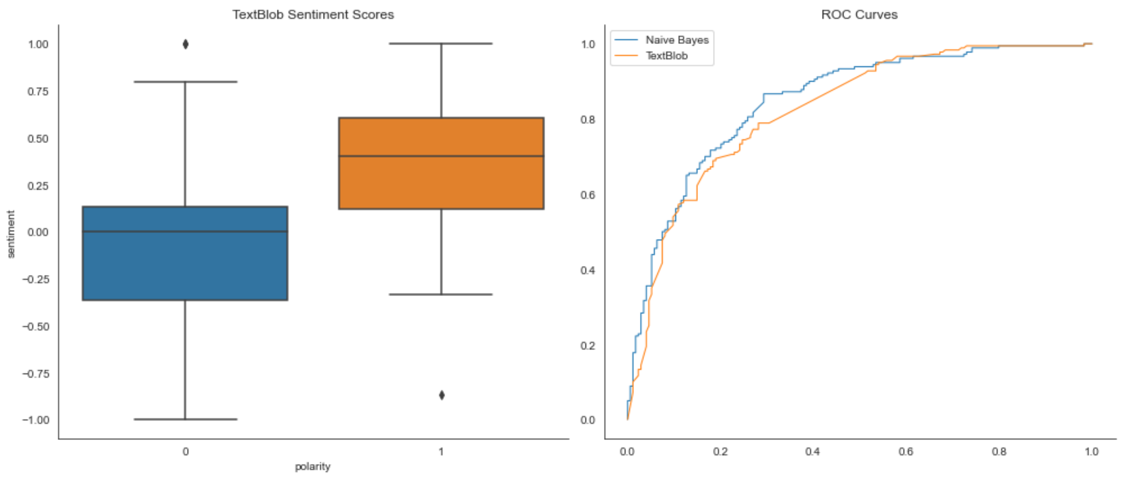
```
[30]: roc_auc_score(y_true=test.polarity, y_score=nb.predict_proba(test_dtm)[:, 1])
```

```
[30]: 0.848595146871009
```

```
[31]: 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.

```
[32]: 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)
      sns.despine()
      fig.tight_layout();
```


TextBlob Sentiment Scores
sentiment
polarity
ROC Curves
Naive Bayes
TextBlob