Exercise Text Classification

June 29, 2019

1 Text Classification

In dieser Aufgabe werden Sie eine Text Classification Pipeline bauen, die Partei gegeben einen Text vorhersagt. Statt der Parlamentsdebatten koennen Sie auch gerne einen anderen Text Datensatz nehmen, wenn Sie einen guten finden. Stellen Sie aber sicher, dass Ihre Kollegen Zugriff auf die Daten haben fuer die Korrektur.

```
In [1]: #!pip install pandas
```

```
In [2]: import os, gzip
        import pandas as pd
        import numpy as np
        import urllib.request

        import warnings
        warnings.filterwarnings('ignore')

        DATADIR = "data"

        if not os.path.exists(DATADIR):
            os.mkdir(DATADIR)

        file_name = os.path.join(DATADIR, 'bundestags_parlamentsprotokolle.csv.gzip')
        if not os.path.exists(file_name):
            url_data = 'https://www.dropbox.com/s/1nlbfehnrwwa2zj/bundestags_parlamentsprotoko
            urllib.request.urlretrieve(url_data, file_name)

        df = pd.read_csv(gzip.open(file_name), index_col=0).sample(frac=1)
```

Ein Auszug der Parlamentsdebatten

```
In [13]: df[:4]
```

```
Out[13]:        sitzung  wahlperiode        sprecher  \
         42752      234           18  Dr.ãAndré Hahn
         13898      159           17    Jimmy Schulz
         24828       17           18    Dennis Rohde
         10093      120           17  Kerstin Griese
```

```
                                                    text partei
42752  Frau Präsidentin! Meine Damen und Herren! Durc...  linke
13898  Natürlich gibt es Probleme bei der Ausbildung ...    fdp
24828  Wir alle wissen doch: Meistens trifft es die S...    spd
10093  Insofern ist meine Entscheidung auch frauenpol...    spd
```

Splitten Sie die Daten in Train (80%) und Test (20%), dafuer koennen sie die sklearn train_test_split function benutzen.

Dann trainieren Sie eine Pipeline mit einem geeigneten Vectorizer und einem sklearn Modell Ihrer Wahl.

Vergleichen Sie die Precision/Recall/F1 und Accuracy auf dem Train und Test set.

```
In [3]: from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.model_selection import train_test_split, GridSearchCV
        from sklearn.neighbors import NearestCentroid
        from sklearn.linear_model import SGDClassifier
        from sklearn.pipeline import Pipeline
        from sklearn.metrics import confusion_matrix, classification_report

In [4]: # split data and labels into train and test data, train and test labels
        train_data, test_data, train_labels, test_labels = train_test_split(df['text'], df['pa
        vect    = TfidfVectorizer(max_features = int(1e8))                      # construct fe

In [14]: ncc     = NearestCentroid()                                             # construct N
         ncc_clf = Pipeline([('vect', vect),('clf', ncc)])                       # construct P
         ncc_clf.fit(train_data, train_labels)                                   # train pipel

         ncc_predictions = ncc_clf.predict(test_data)                            # predict lab
         ncc_clf_report  = classification_report(ncc_predictions, test_labels)   # create repo
         print(ncc_clf_report)

              precision    recall  f1-score   support

      cducsu       0.48      0.63      0.54      2479
         fdp       0.38      0.19      0.25      1402
      gruene       0.39      0.32      0.35      1541
       linke       0.56      0.36      0.44      1829
         spd       0.27      0.44      0.34      1485

    accuracy                           0.41      8736
   macro avg       0.42      0.39      0.38      8736
weighted avg       0.43      0.41      0.41      8736



In [15]: sgd        = SGDClassifier()
         logreg_clf = Pipeline([('vect', vect),('clf', sgd)])
         logreg_clf.fit(train_data, train_labels)
```

```
        logreg_predictions = logreg_clf.predict(test_data)
        logreg_clf_report  = classification_report(logreg_predictions, test_labels)
        print(logreg_clf_report)
```

```
              precision    recall  f1-score   support

      cducsu       0.90      0.56      0.69      5197
         fdp       0.06      0.65      0.10        60
      gruene       0.30      0.63      0.41       614
       linke       0.66      0.59      0.62      1291
         spd       0.39      0.59      0.47      1574

    accuracy                           0.58      8736
   macro avg       0.46      0.60      0.46      8736
weighted avg       0.73      0.58      0.62      8736
```

```
In [5]: from sklearn.neighbors import KNeighborsClassifier
        knc     = KNeighborsClassifier(3)
        knc_clf = Pipeline([('vect', vect), ('clf', knc)])
        knc_clf.fit(train_data, train_labels)

        knc_predictions = knc_clf.predict(test_data)
        knc_clf_report  = classification_report(knc_predictions, test_labels)
        print(knc_clf_report)
```

```
              precision    recall  f1-score   support

      cducsu       0.79      0.44      0.57      5792
         fdp       0.16      0.21      0.18       521
      gruene       0.14      0.40      0.21       444
       linke       0.15      0.56      0.24       335
         spd       0.32      0.45      0.37      1644

    accuracy                           0.43      8736
   macro avg       0.31      0.41      0.32      8736
weighted avg       0.61      0.43      0.48      8736
```

```
In [6]: from sklearn.tree import DecisionTreeClassifier
        dtc     = DecisionTreeClassifier(max_depth=5)
        dtc_clf = Pipeline([('vect', vect), ('clf', dtc)])
        dtc_clf.fit(train_data, train_labels)

        dtc_predictions = dtc_clf.predict(test_data)
```

```
        dtc_clf_report  = classification_report(dtc_predictions, test_labels)
        print(dtc_clf_report)
```

```
              precision    recall  f1-score   support

      cducsu       0.83      0.46      0.59      5858
         fdp       0.05      0.40      0.09        83
      gruene       0.11      0.42      0.17       325
       linke       0.20      0.66      0.30       369
         spd       0.31      0.34      0.33      2101

    accuracy                           0.44      8736
   macro avg       0.30      0.46      0.30      8736
weighted avg       0.64      0.44      0.49      8736
```

In [14]:
```
from sklearn.ensemble import AdaBoostClassifier

        adc     = AdaBoostClassifier()
        adc_clf = Pipeline([('vect', vect), ('clf', adc)])
        adc_clf.fit(train_data, train_labels)

        adc_predictions = adc_clf.predict(test_data)
        adc_clf_report  = classification_report(adc_predictions, test_labels)
        print(adc_clf_report)
```

```
              precision    recall  f1-score   support

      cducsu       0.77      0.52      0.62      4808
         fdp       0.13      0.60      0.22       146
      gruene       0.25      0.38      0.30       819
       linke       0.46      0.45      0.45      1278
         spd       0.30      0.42      0.35      1685

    accuracy                           0.48      8736
   macro avg       0.38      0.47      0.39      8736
weighted avg       0.58      0.48      0.51      8736
```