Text classification

March 7, 2018

1 Introduction to NLP - Text Classification

2 Katrin Strasser - Catalysts

email: katrin.strasser@catalysts.cc
xing: https://www.xing.com/profile/Katrin_Strasser5
linkedin: https://www.linkedin.com/in/katrin-strasser-9582755b
www.catalysts.cc

2.0.1 Some additional information in case you want to try text classification on your own:

The 20 newsgroups text dataset, a public dataset: http://scikit-learn.org/stable/datasets/twenty_newsgroups.html

Code for the confusion matrix plot can be found here: http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py

What you can do to improve the accuracy:

Hyperparamter grid search: http://scikit-learn.org/stable/modules/grid_search.html

Different encodings: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEn
http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
Nice blog entry about encoding: https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/

Different classifiers: http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparis
http://scikit-learn.org/stable/supervised_learning.html

```
In [1]: %matplotlib inline
        import pandas as pd
        import numpy as np

        import spacy
        #from spacy import displacy
```

```
import itertools

import matplotlib.pyplot as plt

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder

from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier

from sklearn.metrics import confusion_matrix, accuracy_score

from plot_cm import plot_confusion_matrix
```

2.1 Dataset

Read the .csv file into the variable "data"

```
In [2]: data = pd.read_csv('articles.csv')
```

Print the first 10 rows

```
In [27]: data.shape
```

```
Out[27]: (18550, 7)
```

```
In [3]: data[:10]
```

```
Out[3]:        id  char_count                                            content  \
        0  337240        4894  'Wieder einmal will die Türkei Abgeordnete nic...
        1  337259         469  'Stuttgart (dpa/lsw) - Ministerpräsident Winfr...
        2  337262         372  'Cottbus (dpa/bb) - Der Braunkohlenausschuss d...
        3  337267        3404  'Frank Plasberg lässt die Erfolgsaussichten de...
        4  337271         492  'Washington (dpa) - US-Präsident Donald Trump ...
        5  337278         502  'New York (dpa) - UN-Generalsekretär António G...
        6  337284        2638  'Düsseldorf  Die SPD will nach ihrer schweren...
        7  337294        2317  'Nordkorea könnte Experten zufolge hinter der ...
        8  337297         153  'Der Bundesgerichtshof (BGH) hat der Unterlass...
        9  337298        2317  'Nordkorea könnte Experten zufolge hinter der ...

                                                   title  \
        0  'Erneuter Incirlik-Eklat Pokern um den "Tornad...
        1  'Kretschmann empfängt Adelsfamilien zum gemein...
        2  'Braunkohlenausschuss kommt nach Revierkonzept...
        3  '"Hart aber fair" mit Frank Plasberg Es muss n...
        4  'McMaster: Bericht über Trumps Geheimnis-Weite...
        5  'Vereinte Nationen verurteilen Nordkoreas Rake...
```

```
6  'Nach NRW-Schlappe | SPD schliet groe&lt;br ...
7  'Experten: Nordkorea könnte hinter weltweiter ...
8  'BGH-Urteil - Darlehenskontogebühren bei Bausp...
9  'Experten: Nordkorea könnte hinter weltweiter ...

                                                 url    category  \
0  'http://www.spiegel.de//politik/deutschland/in...     politik
1  'http://www.sueddeutsche.de/news/politik/regie...     politik
2  'http://www.sueddeutsche.de/news/wirtschaft/en...  wirtschaft
3  'http://www.sueddeutsche.de/medien/hart-aber-f...      medien
4  'http://www.sueddeutsche.de/news/politik/gehei...     politik
5  'http://www.sueddeutsche.de/news/politik/konfl...     politik
6  'http://www.bild.de/politik/inland/politik/nrw...     politik
7  'http://www.donaukurier.de/nachrichten/wirtsch...  wirtschaft
8  'http://www.donaukurier.de/nachrichten/wirtsch...  wirtschaft
9  'http://www.donaukurier.de/nachrichten/medien/...      medien

                                content_preprocessed
0  'Türkei Abgeordnete Incirlik lassen. Bundesreg...
1  'Stuttgart (dpa/lsw) - Ministerpräsident Winfr...
2  'Cottbus (dpa/bb) - Braunkohlenausschuss Land ...
3  'Frank Plasberg lässt Erfolgsaussichten SPD-Ka...
4  'Washington (dpa) - US-Präsident Donald Trump ...
5  'New York (dpa) - UN-Generalsekretär António G...
6  'Düsseldorf  SPD schwer Niederlage Landtagswa...
7  'Nordkorea Experte zufolge jung weltweit Cyber...
8  'Bundesgerichtshof (BGH) Unterlassungsklage Ve...
9  'Nordkorea Experte zufolge jung weltweit Cyber...
```

Investigate how many instances of each class are present in the data

```
In [4]: data['category'].value_counts()
```

```
Out[4]: sport         6366
        politik       5110
        wirtschaft    3341
        kultur        1366
        leben         1264
        medien        1103
        Name: category, dtype: int64
```

2.2 Introduction to Spacy

Load english spacy model into nlp_en

```
In [5]: nlp_en = spacy.load('en')
```

Have a look at the pipeline

```
In [6]: nlp_en.pipeline

Out[6]: [<spacy.tagger.Tagger at 0x1a08f652558>,
         <spacy.pipeline.DependencyParser at 0x1a0990ce7c8>,
         <spacy.matcher.Matcher at 0x1a0a4779ac8>,
         <spacy.pipeline.EntityRecognizer at 0x1a0a428c818>]
```

Feed an english sentence to the pipeline

```
In [7]: doc_en = nlp_en('These apples cost 10 dollars and are harvested in San Francisco')
```

Investigate the lemmas and POS

```
In [8]: for toc in doc_en:
            print(toc.text + toc.whitespace_ + toc.lemma_ + toc.whitespace_ + toc.pos_)

These these DET
apples apple NOUN
cost cost VERB
10 10 NUM
dollars dollar NOUN
and and CCONJ
are be VERB
harvested harvest VERB
in in ADP
San san PROPN
FranciscofranciscoPROPN
```

Look at entities

```
In [9]: doc_en.ents

Out[9]: (10 dollars, San Francisco)

In [10]: for ent in doc_en.ents:
             print(ent.text + " " + ent.label_)

10 dollars MONEY
San Francisco GPE
```

2.3 Classification Time

Load the german spacy word model

```
In [11]: nlp = spacy.load('de')
```

Apply spacy pre-processing to content

```
In [12]: content = nlp.pipe(data['content'])
```

Read the stopwords file "stopwords.txt" line by line

```
In [13]: with open('stopwords.txt', 'r') as file:
             stopwords = file.readlines()
```

Remove whitespaces and newlines from the stopwords

```
In [15]: stopwords = [s.strip() for s in stopwords]

In [16]: stopwords[:10]

Out[16]: ['a', 'ab', 'aber', 'ach', 'acht', 'achte', 'achten', 'achter', 'achtes', 'ag']

In [17]: def set_stop_word(sw):
             nlp.vocab[sw].is_stop = True
             sw = sw[0].upper() + sw[1:]
             nlp.vocab[sw].is_stop = True

In [18]: for w in stopwords:
             set_stop_word(w)
```

Remove stopwords from content

```
In [ ]: content_preprocessed=[]
        for t in content:
            content_preprocessed.append("".join(list("".join(tok.lemma_ + tok.whitespace_) for

In [19]: data['content_preprocessed'][1]

Out[19]: '\'Stuttgart (dpa/lsw) - Ministerpräsident Winfried Kretschmann (grün) empfangen Stut
```

Split the data into two sets: one for training the classifier and the other one for testing the performance

```
In [20]: split = int((len(data)/100)*80)

         train = data[:split]
         test = data[split:]
```

Encode the texts with the TfidfVectorizer(max_features)

```
In [21]: tfidf = TfidfVectorizer(max_features=10000)

         train_x = tfidf.fit_transform(train['content_preprocessed']).toarray()
         test_x = tfidf.transform(test['content_preprocessed']).toarray()
```

Encode the labels with the LabelEncoder

```
In [22]: le = LabelEncoder()

         train_y = le.fit_transform(train['category'])
         test_y = le.transform(test['category'])
```

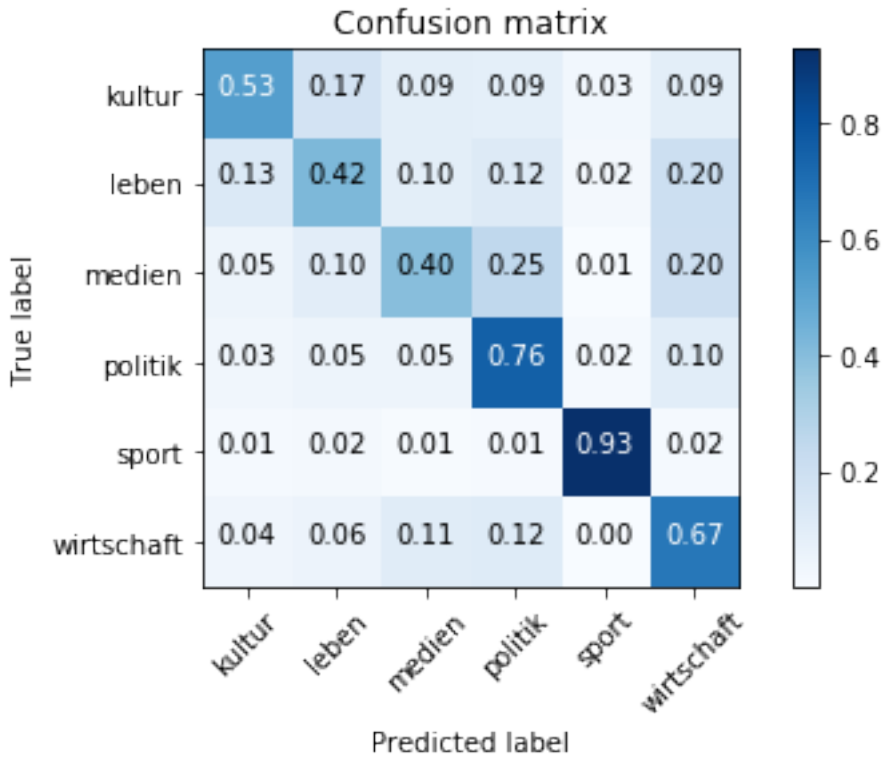
Naive Bayer Classifier (GaussianNB)

```
In [24]: nb = GaussianNB()
         nb.fit(train_x, train_y)

         nb_pred = nb.predict(test_x)
         nb_cnf = confusion_matrix(nb_pred, test_y)

         print(accuracy_score(nb_pred, test_y))
         plot_confusion_matrix(nb_cnf, le.classes_)
```

```
0.7474393530997304
```



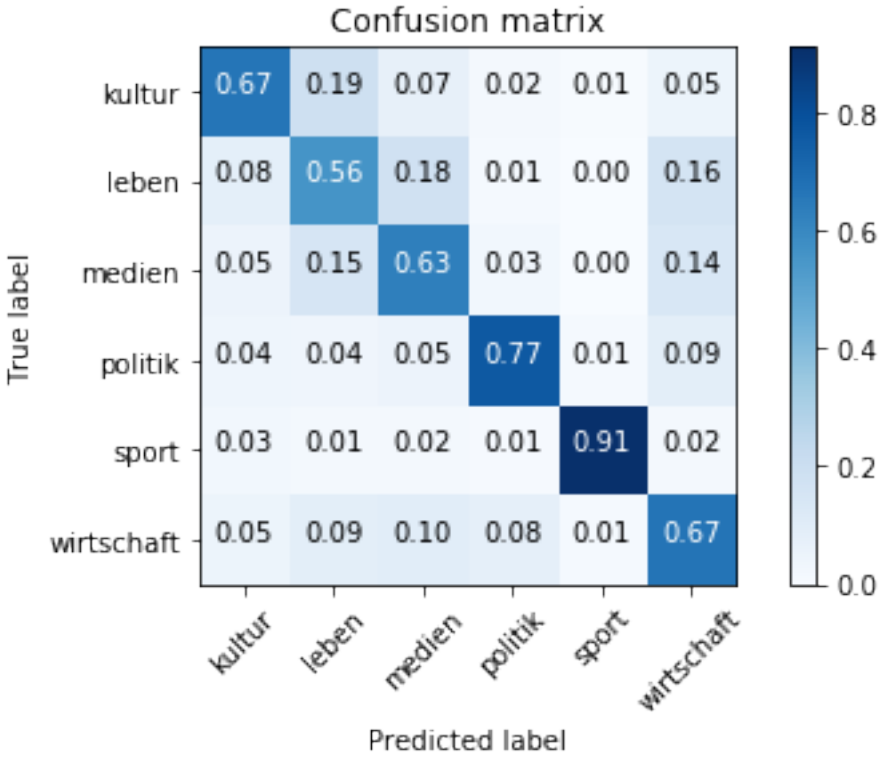
Random Forest Classifier (RandomForestClassifier(max_depth))

```
In [25]: rf = RandomForestClassifier(max_depth=100)
         rf.fit(train_x, train_y)

         rf_pred = rf.predict(test_x)
         rf_cnf = confusion_matrix(rf_pred, test_y)

         print(accuracy_score(rf_pred, test_y))
         plot_confusion_matrix(rf_cnf, le.classes_)
```

```
0.7819407008086253
```



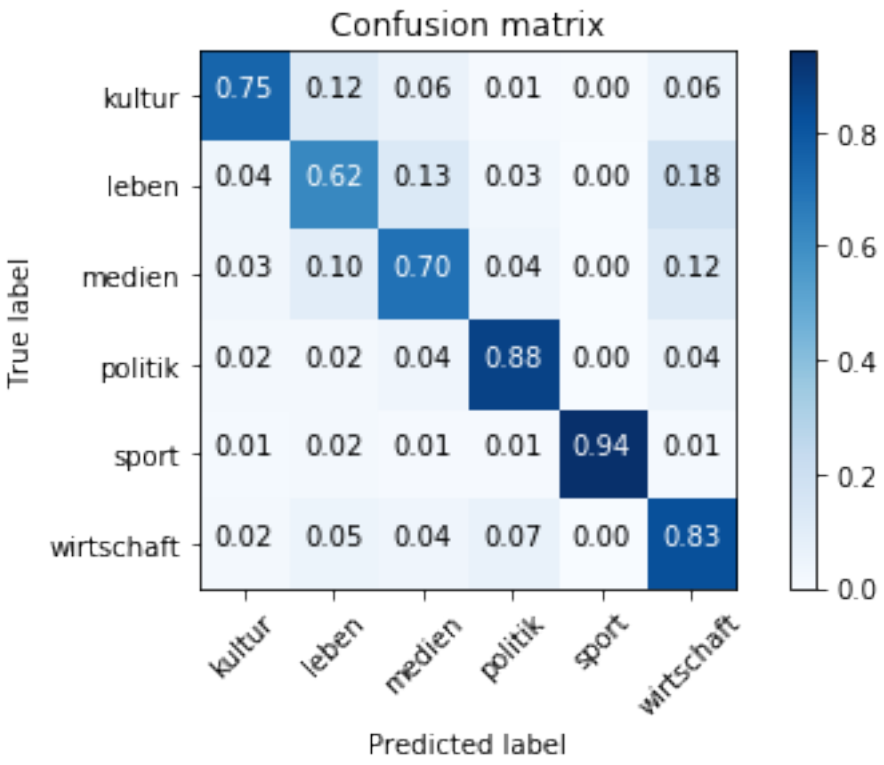
Support Vector Machine (SGDClassifier(max_iter))

```
In [26]: svm = SGDClassifier(max_iter=10)
         svm.fit(train_x, train_y)

         svm_pred = svm.predict(test_x)
         svm_cnf = confusion_matrix(svm_pred, test_y)

         print(accuracy_score(svm_pred, test_y))
         plot_confusion_matrix(svm_cnf, le.classes_)
```

```
0.8563342318059299
```


Confusion matrix
kultur
leben
medien
politik
sport
wirtschaft
True label
0.75 0.12 0.06 0.01 0.00 0.06
0.04 0.62 0.13 0.03 0.00 0.18
0.03 0.10 0.70 0.04 0.00 0.12
0.02 0.02 0.04 0.88 0.00 0.04
0.01 0.02 0.01 0.01 0.94 0.01
0.02 0.05 0.04 0.07 0.00 0.83
kultur
leben
medien
politik
sport
wirtschaft
Predicted label
0.8
0.6
0.4
0.2
0.0

In [ ]: