scikit-learn_Wörter

December 1, 2020

0.0.1 Scikit-learn im Lyrik-Projekt, auf Basis der Wörter, nicht Embeddings.

[1]:
```
import sklearn
import numpy as np
from glob import glob
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.linear_model import SGDClassifier
from sklearn import metrics
from sklearn.pipeline import Pipeline

import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
```

```
[nltk_data] Downloading package stopwords to /home/piah/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

[2]:
```
categories = ['Balladen', 'Lyrik']
```

[3]:
```
docs_to_train = sklearn.datasets.load_files("/home/piah/Dokumente/Uni/
 ↪Projektarbeit/Projektarbeit_LyrikGattungszuweisung/scikit-learn/autoren/",␣
 ↪description=None, categories=categories, load_content=True, shuffle=True,␣
 ↪encoding='utf-8', decode_error='strict', random_state=42)
print(list(docs_to_train.target_names))
```

```
['Balladen', 'Lyrik']
```

[4]:
```
docs_to_train.target_names
```

[4]:
```
['Balladen', 'Lyrik']
```

[5]:
```
len(docs_to_train.data)
```

[5]:
```
409
```

[6]:
```
len(docs_to_train.filenames)
```

[6]:
```
409
```

[7]:
```
print("\n".join(docs_to_train.data[0].split("\n")[:3]))
```

```
winter aus schneegestäub ' und nebelqualm brechen endlich doch eine klar tag
da fliegen alle fenster auf   eine jede spähen   was er vermag   ob jen block
haus sein   eine weiher jen ebnen raum   fürwahr   in dies uniform die
glockenturm erkennen man kaum   und alle leben liegen zerdrücken   wie unter+die
leichentuch ersticken   doch schauen   an horizont rand begegnen ich lebend'ges
land   du starr wächter   laß er los die föhn aus dein kerker schoß   wo
schwärzlich jen riff spalten   da müssen er quarantäne halten   die fremdling
aus die lombardei   o säntis   geben die tauwind frei
```

[8]:
```
print(docs_to_train.target_names[docs_to_train.target[0]])
```

```
Lyrik
```

[9]:
```
docs_to_train.target[:10]
```

[9]:
```
array([1, 1, 0, 0, 0, 0, 1, 1, 1, 1])
```

[10]:
```
for t in docs_to_train.target[:10]:
    print(docs_to_train.target_names[t])
```

```
Lyrik
Lyrik
Balladen
Balladen
Balladen
Balladen
Lyrik
Lyrik
Lyrik
Lyrik
```

0.1 Extracting features from text files

0.1.1 Bags of words

0.1.2 Tokenizing text with scikit-learn

[11]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

[12]:
```
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(docs_to_train.data)
X_train_counts.shape
```

[12]: (409, 9146)

[13]:
```
count_vect.vocabulary_.get(u'algorithm')
```

0.1.3 From occurrences to frequencies

tf tf-idf

[14]:
```
from sklearn.feature_extraction.text import TfidfTransformer
```

[15]:
```
tf_transformer = TfidfTransformer(use_idf=False).fit(X_train_counts)
X_train_tf = tf_transformer.transform(X_train_counts)
X_train_tf.shape
```

[15]: (409, 9146)

[16]:
```
# oder

tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_train_tfidf.shape
```

[16]: (409, 9146)

0.2 Training a classifier

0.2.1 naïve Bayes

[17]:
```
from sklearn.naive_bayes import MultinomialNB
```

[18]:
```
clf = MultinomialNB().fit(X_train_tfidf, docs_to_train.target)
```

Plutchik

[19]:
```
docs_new = ['freude','vertrauen', 'angst','überraschung', 'trauer', 'ekel',␣
 ↪'wut', 'mitgefühl', 'liebe']
X_new_counts = count_vect.transform(docs_new)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)

predicted = clf.predict(X_new_tfidf)

for doc, category in zip(docs_new, predicted):
     print('%r => %s' % (doc, docs_to_train.target_names[category]))
```

```
'freude' => Lyrik
'vertrauen' => Lyrik
'angst' => Lyrik
'überraschung' => Lyrik
```

```
'trauer' => Lyrik
'ekel' => Lyrik
'wut' => Lyrik
'mitgefühl' => Lyrik
'liebe' => Lyrik
```

Sprechmarker

[20]:
```
docs_new = ['sagen', 'sprechen', 'fragen', 'antworten', 'schreien', 'jammern']
X_new_counts = count_vect.transform(docs_new)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)

predicted = clf.predict(X_new_tfidf)

for doc, category in zip(docs_new, predicted):
     print('%r => %s' % (doc, docs_to_train.target_names[category]))
```

```
'sagen' => Lyrik
'sprechen' => Lyrik
'fragen' => Lyrik
'antworten' => Lyrik
'schreien' => Lyrik
'jammern' => Lyrik
```

Sprechmarker_lang

[21]:
```
docs_new =␣
 ↪['abhören','ablehnen','abraten','abschlagen','abschweifen','absprechen','abstreiten',
                     'andeuten','anerkennen',␣
 ↪'anfechten','angeben','ankünden','anraten',
                    ␣
 ↪'antworten','anvertrauen','anweisen','artikulieren','aufmuntern',
                     'aufzeigen','ausdrücken','Ausflüchte␣
 ↪machen','ausforschen','ausfragen','aushorchen',
                     'ausplaudern','ausquetschen','aussagen', 'äußern', 'sich␣
 ↪aussprechen','ausweichen',
                     'bedanken',␣
 ↪'befehlen','befragen','befürchten','begründen','beharren auf','behaupten',
                     'beichten','bejahen', 'bekennen', 'beklagen',␣
 ↪'bekräftigen', 'bekunden', 'beleuchten', 'bemängeln',
                     'bemerken', 'berichten', 'berichtigen', 'beruhigen',␣
 ↪'besagen', 'Bescheid geben', 'beschönigen',
                     'beschreiben','beschwören','bestätigen','bestehen␣
 ↪auf','bestimmen','bestreiten','beten','beteuern',
                    ␣
 ↪'betonen','betteln','beweisen','bezeichnen','bezeugen','bitten','brabbeln','bramarbasieren',
                    ␣
 ↪'breittreten','brüllen','brummen','daherreden','darlegen','dartun','definieren',
```

```
                    'dementieren','demonstrieren', 'den Mund␣
↪vollnehmen','deuteln','dick auftragen','dazwischenfahren',
                    'drohen', 'ein Ohr␣
↪abkauen','einräumen','einreden','einschärfen','einwenden','einwilligen','empfehlen',
                    ␣
↪'entgegenhalten','entgegnen','enthüllen','entkräften','entschuldigen','erdichten','erfinden',
                    'ergänzen','erinnern','erklären','erkundigen',␣
↪'ermahnen','ermuntern','ermutigen','erörtern','erschließen',
                    'erwähnen', 'erwidern','erzählen','evident␣
↪machen','fabulieren','faseln','feststellen',
                    ␣
↪'flehen','flunkern','flüstern','folgern','fordern','fragen', 'geheim halten',
                    'gestehen','herausposaunen','herausreden',␣
↪'herausschreien','herumkritteln','hervorheben','hinweisen',
                    'hinzufügen','höhnen','in Abrede stellen','in Frage␣
↪stellen','ins Gesicht sagen',
                    ␣
↪'jammern','keuchen','klagen','klöhnen','konstatieren','krächzen','kritisieren','kundgeben',
                    ␣
↪'kundtun','labern','lallen','leugnen','lispeln','loben','mahnen','meinen','mitteilen',
                    ␣
↪'munkeln','murmeln','nachfragen','nachweisen','näseln','negieren','nennen',
                    ␣
↪'nuscheln','offenbaren','palavern','petzen','plaudern','plauschen','prahlen','quasseln','quat
                    'raten','raunen','Rede und Antwort stehen','reden wie ein␣
↪Buch','reden wie ein Wasserfall',
                    'reinen Wein␣
↪einschenken','röcheln','rufen','sagen','schildern','schleimen','schließen',
                    ␣
↪'schluchzen','schnacken','schnarren','schnattern','schreien','schwadronieren','schwatzen',
                    ␣
↪'schwätzen','schwören','seiern','skizzieren','sprechen','stammeln','stottern','tadeln',
                    'trösten','tuscheln','überinterpretieren','überreden',
                    'überzeugen','umreißen','umschreiben','unken','Unsinn␣
↪verzapfen','unterhalten','unterstreichen',
                    ␣
↪'unterstützen','veranschaulichen','verdrehen','verfälschen','verhören','verklickern',
                    ␣
↪'verkünden','verlangen','verlautbaren','verleugnen','vernehmen','verneinen','verraten','verso
                    ␣
↪'versichern','versprechen','vertiefen','verwerfen','verzerren','vorbringen','vorenthalten','v
                    'weit␣
↪ausholen','widerlegen','widerrufen','widersprechen','wiederholen','wissen␣
↪wollen','zeigen',
```

```
                    'zitieren','zu bedenken␣
 ↪geben','zugeben','zugestehen','zureden','zurücknehmen',
                    'zurufen','zustimmen', 'hören', 'singen', 'danken',␣
 ↪'grüßen', 'befehlen', 'lügen', 'gestehen', 'seufzen',
                    'stöhnen', 'staunen', 'zuhören', 'jubeln', 'schimpfen',␣
 ↪'wettern', 'krächzen']
X_new_counts = count_vect.transform(docs_new)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)

predicted = clf.predict(X_new_tfidf)

for doc, category in zip(docs_new, predicted):
     print('%r => %s' % (doc, docs_to_train.target_names[category]))
```

```
'abhören' => Lyrik
'ablehnen' => Lyrik
'abraten' => Lyrik
'abschlagen' => Lyrik
'abschweifen' => Lyrik
'absprechen' => Lyrik
'abstreiten' => Lyrik
'andeuten' => Lyrik
'anerkennen' => Lyrik
'anfechten' => Lyrik
'angeben' => Lyrik
'ankünden' => Lyrik
'anraten' => Lyrik
'antworten' => Lyrik
'anvertrauen' => Lyrik
'anweisen' => Lyrik
'artikulieren' => Lyrik
'aufmuntern' => Lyrik
'aufzeigen' => Lyrik
'ausdrücken' => Lyrik
'Ausflüchte machen' => Lyrik
'ausforschen' => Lyrik
'ausfragen' => Lyrik
'aushorchen' => Lyrik
'ausplaudern' => Lyrik
'ausquetschen' => Lyrik
'aussagen' => Lyrik
'äußern' => Lyrik
'sich aussprechen' => Lyrik
'ausweichen' => Lyrik
'bedanken' => Lyrik
'befehlen' => Lyrik
'befragen' => Lyrik
```

```
'befürchten' => Lyrik
'begründen' => Lyrik
'beharren auf' => Lyrik
'behaupten' => Lyrik
'beichten' => Lyrik
'bejahen' => Lyrik
'bekennen' => Lyrik
'beklagen' => Lyrik
'bekräftigen' => Lyrik
'bekunden' => Lyrik
'beleuchten' => Lyrik
'bemängeln' => Lyrik
'bemerken' => Lyrik
'berichten' => Lyrik
'berichtigen' => Lyrik
'beruhigen' => Lyrik
'besagen' => Lyrik
'Bescheid geben' => Lyrik
'beschönigen' => Lyrik
'beschreiben' => Lyrik
'beschwören' => Lyrik
'bestätigen' => Lyrik
'bestehen auf' => Lyrik
'bestimmen' => Lyrik
'bestreiten' => Lyrik
'beten' => Lyrik
'beteuern' => Lyrik
'betonen' => Lyrik
'betteln' => Lyrik
'beweisen' => Lyrik
'bezeichnen' => Lyrik
'bezeugen' => Lyrik
'bitten' => Lyrik
'brabbeln' => Lyrik
'bramarbasieren' => Lyrik
'breittreten' => Lyrik
'brüllen' => Lyrik
'brummen' => Lyrik
'daherreden' => Lyrik
'darlegen' => Lyrik
'dartun' => Lyrik
'definieren' => Lyrik
'dementieren' => Lyrik
'demonstrieren' => Lyrik
'den Mund vollnehmen' => Lyrik
'deuteln' => Lyrik
'dick auftragen' => Lyrik
'dazwischenfahren' => Lyrik
```

```
'drohen' => Lyrik
'ein Ohr abkauen' => Lyrik
'einräumen' => Lyrik
'einreden' => Lyrik
'einschärfen' => Lyrik
'einwenden' => Lyrik
'einwilligen' => Lyrik
'empfehlen' => Lyrik
'entgegenhalten' => Lyrik
'entgegnen' => Lyrik
'enthüllen' => Lyrik
'entkräften' => Lyrik
'entschuldigen' => Lyrik
'erdichten' => Lyrik
'erfinden' => Lyrik
'ergänzen' => Lyrik
'erinnern' => Lyrik
'erklären' => Lyrik
'erkundigen' => Lyrik
'ermahnen' => Lyrik
'ermuntern' => Lyrik
'ermutigen' => Lyrik
'erörtern' => Lyrik
'erschließen' => Lyrik
'erwähnen' => Lyrik
'erwidern' => Lyrik
'erzählen' => Lyrik
'evident machen' => Lyrik
'fabulieren' => Lyrik
'faseln' => Lyrik
'feststellen' => Lyrik
'flehen' => Lyrik
'flunkern' => Lyrik
'flüstern' => Lyrik
'folgern' => Lyrik
'fordern' => Lyrik
'fragen' => Lyrik
'geheim halten' => Lyrik
'gestehen' => Lyrik
'herausposaunen' => Lyrik
'herausreden' => Lyrik
'herausschreien' => Lyrik
'herumkritteln' => Lyrik
'hervorheben' => Lyrik
'hinweisen' => Lyrik
'hinzufügen' => Lyrik
'höhnen' => Lyrik
'in Abrede stellen' => Lyrik
```

```
'in Frage stellen' => Lyrik
'ins Gesicht sagen' => Lyrik
'jammern' => Lyrik
'keuchen' => Lyrik
'klagen' => Lyrik
'klöhnen' => Lyrik
'konstatieren' => Lyrik
'krächzen' => Lyrik
'kritisieren' => Lyrik
'kundgeben' => Lyrik
'kundtun' => Lyrik
'labern' => Lyrik
'lallen' => Lyrik
'leugnen' => Lyrik
'lispeln' => Lyrik
'loben' => Lyrik
'mahnen' => Lyrik
'meinen' => Lyrik
'mitteilen' => Lyrik
'munkeln' => Lyrik
'murmeln' => Lyrik
'nachfragen' => Lyrik
'nachweisen' => Lyrik
'näseln' => Lyrik
'negieren' => Lyrik
'nennen' => Lyrik
'nuscheln' => Lyrik
'offenbaren' => Lyrik
'palavern' => Lyrik
'petzen' => Lyrik
'plaudern' => Lyrik
'plauschen' => Lyrik
'prahlen' => Lyrik
'quasseln' => Lyrik
'quatschen' => Lyrik
'raten' => Lyrik
'raunen' => Lyrik
'Rede und Antwort stehen' => Lyrik
'reden wie ein Buch' => Lyrik
'reden wie ein Wasserfall' => Lyrik
'reinen Wein einschenken' => Lyrik
'röcheln' => Lyrik
'rufen' => Balladen
'sagen' => Lyrik
'schildern' => Lyrik
'schleimen' => Lyrik
'schließen' => Lyrik
'schluchzen' => Lyrik
```

```
'schnacken' => Lyrik
'schnarren' => Lyrik
'schnattern' => Lyrik
'schreien' => Lyrik
'schwadronieren' => Lyrik
'schwatzen' => Lyrik
'schwätzen' => Lyrik
'schwören' => Lyrik
'seiern' => Lyrik
'skizzieren' => Lyrik
'sprechen' => Lyrik
'stammeln' => Lyrik
'stottern' => Lyrik
'tadeln' => Lyrik
'trösten' => Lyrik
'tuscheln' => Lyrik
'überinterpretieren' => Lyrik
'überreden' => Lyrik
'überzeugen' => Lyrik
'umreißen' => Lyrik
'umschreiben' => Lyrik
'unken' => Lyrik
'Unsinn verzapfen' => Lyrik
'unterhalten' => Lyrik
'unterstreichen' => Lyrik
'unterstützen' => Lyrik
'veranschaulichen' => Lyrik
'verdrehen' => Lyrik
'verfälschen' => Lyrik
'verhören' => Lyrik
'verklickern' => Lyrik
'verkünden' => Lyrik
'verlangen' => Lyrik
'verlautbaren' => Lyrik
'verleugnen' => Lyrik
'vernehmen' => Lyrik
'verneinen' => Lyrik
'verraten' => Lyrik
'verschleiern' => Lyrik
'versichern' => Lyrik
'versprechen' => Lyrik
'vertiefen' => Lyrik
'verwerfen' => Lyrik
'verzerren' => Lyrik
'vorbringen' => Lyrik
'vorenthalten' => Lyrik
'vorgeben' => Lyrik
'warnen' => Lyrik
```

```
'weit ausholen' => Lyrik
'widerlegen' => Lyrik
'widerrufen' => Lyrik
'widersprechen' => Lyrik
'wiederholen' => Lyrik
'wissen wollen' => Lyrik
'zeigen' => Lyrik
'zitieren' => Lyrik
'zu bedenken geben' => Lyrik
'zugeben' => Lyrik
'zugestehen' => Lyrik
'zureden' => Lyrik
'zurücknehmen' => Lyrik
'zurufen' => Lyrik
'zustimmen' => Lyrik
'hören' => Lyrik
'singen' => Lyrik
'danken' => Lyrik
'grüßen' => Lyrik
'befehlen' => Lyrik
'lügen' => Lyrik
'gestehen' => Lyrik
'seufzen' => Lyrik
'stöhnen' => Lyrik
'staunen' => Lyrik
'zuhören' => Lyrik
'jubeln' => Lyrik
'schimpfen' => Lyrik
'wettern' => Lyrik
'krächzen' => Lyrik
```

Emotionen

[22]:
```
docs_new = ['liebreich','angst', 'ängstlich', 'trauer', 'traurig', 'zornig',␣
↪'zorn', 'verachtungsvoll', 'verachtung',
            'schuld', 'schuldig', 'liebe',  'geliebt', 'liebevoll', 'stolz',␣
↪'scham', 'schämen', 'überrasschung',
            'überrascht', 'sorge', 'sorgenvoll', 'ekel', 'ekeln', 'angeekelt',␣
↪'neid', 'neidisch', 'neidvoll',
            'glücklich', 'glück', 'freude', 'freudig', 'freuen', 'erleichterung',␣
↪'erleichtert', 'vergnügt', 'vergnügen',
            'zufrieden', 'zufriedenheit', 'verzweiflung', 'verzweifelt',␣
↪'verlegenheit', 'verlegen', 'aufregung',
            'aufgeregt', 'aufregen', 'spannung', 'gespannt', 'erregung',␣
↪'erregt', 'hoffen', 'hoffnung', 'befriedigt',
            'langweilig', 'langeweile', 'mitgefühl', 'mitfühlen', 'enttäuscht',␣
↪'enttäuschung', 'frust', 'frustriert',
```

```
          'eifersucht', 'eifersüchtig', 'wut', 'wütend', 'reue', 'schock',␣
 ↪'schockiert', 'zuneigung', 'verärgert',
          'verärgerung', 'erwartungsvoll', 'erwartung', 'vorfreude', 'scheu',␣
 ↪'gelassen', 'gelassenheit', 'mut',
          'mutig', 'neugierde', 'neugierig', 'depression', 'depressiv',␣
 ↪'niedergeschlagenheit', 'niedergeschlagen',
          'lustvoll', 'lust', 'rausch', 'einfühlend', 'einfühlsam',␣
 ↪'euphorisch', 'euphorie', 'dankbarkeit', 'dankbar',
          'hass', 'entsetzt', 'entsetzen', 'demütigung', 'demütig', 'demut',␣
 ↪'interesse', 'interessiert', 'einsamkeit',
          'einsam', 'empörung', 'empört', 'vertrauen', 'qualvoll', 'qual',␣
 ↪'gleichgültigkeit', 'gleichgültig',
          'fröhlichkeit', 'fröhlich', 'schadenfroh', 'schadenfreude',␣
 ↪'schmerz', 'melancholie', 'melancholisch',
          'panik', 'panisch', 'fühlen', 'herz', 'seele', 'schwermut',␣
 ↪'unglück', 'wanderlust',  'bitterkeit','freuen',
          'weinen',  'erschrecken',  'empfinden', 'vergnügen', 'bekümmern',␣
 ↪'befürchten', 'jauchzen', 'verzweifeln',
          'qual', 'grausne', 'Lust',  'kichern', 'zufrieden',  'leid', 'zorn', ␣
 ↪'heiter', 'lache',  'weinen', 'sehnsucht',
          'hoffnung',  'mutig', 'trost', 'ohnmacht', 'ohnmächtig']
X_new_counts = count_vect.transform(docs_new)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)

predicted = clf.predict(X_new_tfidf)

for doc, category in zip(docs_new, predicted):
     print('%r => %s' % (doc, docs_to_train.target_names[category]))
```

```
'liebreich' => Lyrik
'angst' => Lyrik
'ängstlich' => Lyrik
'trauer' => Lyrik
'traurig' => Lyrik
'zornig' => Lyrik
'zorn' => Lyrik
'verachtungsvoll' => Lyrik
'verachtung' => Lyrik
'schuld' => Lyrik
'schuldig' => Lyrik
'liebe' => Lyrik
'geliebt' => Lyrik
'liebevoll' => Lyrik
'stolz' => Lyrik
'scham' => Lyrik
'schämen' => Lyrik
'überrasschung' => Lyrik
```

```
'überrascht' => Lyrik
'sorge' => Lyrik
'sorgenvoll' => Lyrik
'ekel' => Lyrik
'ekeln' => Lyrik
'angeekelt' => Lyrik
'neid' => Lyrik
'neidisch' => Lyrik
'neidvoll' => Lyrik
'glücklich' => Lyrik
'glück' => Lyrik
'freude' => Lyrik
'freudig' => Lyrik
'freuen' => Lyrik
'erleichterung' => Lyrik
'erleichtert' => Lyrik
'vergnügt' => Lyrik
'vergnügen' => Lyrik
'zufrieden' => Lyrik
'zufriedenheit' => Lyrik
'verzweiflung' => Lyrik
'verzweifelt' => Lyrik
'verlegenheit' => Lyrik
'verlegen' => Lyrik
'aufregung' => Lyrik
'aufgeregt' => Lyrik
'aufregen' => Lyrik
'spannung' => Lyrik
'gespannt' => Lyrik
'erregung' => Lyrik
'erregt' => Lyrik
'hoffen' => Lyrik
'hoffnung' => Lyrik
'befriedigt' => Lyrik
'langweilig' => Lyrik
'langeweile' => Lyrik
'mitgefühl' => Lyrik
'mitfühlen' => Lyrik
'enttäuscht' => Lyrik
'enttäuschung' => Lyrik
'frust' => Lyrik
'frustriert' => Lyrik
'eifersucht' => Lyrik
'eifersüchtig' => Lyrik
'wut' => Lyrik
'wütend' => Lyrik
'reue' => Lyrik
'schock' => Lyrik
```

```
'schockiert' => Lyrik
'zuneigung' => Lyrik
'verärgert' => Lyrik
'verärgerung' => Lyrik
'erwartungsvoll' => Lyrik
'erwartung' => Lyrik
'vorfreude' => Lyrik
'scheu' => Lyrik
'gelassen' => Lyrik
'gelassenheit' => Lyrik
'mut' => Lyrik
'mutig' => Lyrik
'neugierde' => Lyrik
'neugierig' => Lyrik
'depression' => Lyrik
'depressiv' => Lyrik
'niedergeschlagenheit' => Lyrik
'niedergeschlagen' => Lyrik
'lustvoll' => Lyrik
'lust' => Lyrik
'rausch' => Lyrik
'einfühlend' => Lyrik
'einfühlsam' => Lyrik
'euphorisch' => Lyrik
'euphorie' => Lyrik
'dankbarkeit' => Lyrik
'dankbar' => Lyrik
'hass' => Lyrik
'entsetzt' => Lyrik
'entsetzen' => Lyrik
'demütigung' => Lyrik
'demütig' => Lyrik
'demut' => Lyrik
'interesse' => Lyrik
'interessiert' => Lyrik
'einsamkeit' => Lyrik
'einsam' => Lyrik
'empörung' => Lyrik
'empört' => Lyrik
'vertrauen' => Lyrik
'qualvoll' => Lyrik
'qual' => Lyrik
'gleichgültigkeit' => Lyrik
'gleichgültig' => Lyrik
'fröhlichkeit' => Lyrik
'fröhlich' => Lyrik
'schadenfroh' => Lyrik
'schadenfreude' => Lyrik
```

```
'schmerz' => Lyrik
'melancholie' => Lyrik
'melancholisch' => Lyrik
'panik' => Lyrik
'panisch' => Lyrik
'fühlen' => Lyrik
'herz' => Lyrik
'seele' => Lyrik
'schwermut' => Lyrik
'unglück' => Lyrik
'wanderlust' => Lyrik
'bitterkeit' => Lyrik
'freuen' => Lyrik
'weinen' => Lyrik
'erschrecken' => Lyrik
'empfinden' => Lyrik
'vergnügen' => Lyrik
'bekümmern' => Lyrik
'befürchten' => Lyrik
'jauchzen' => Lyrik
'verzweifeln' => Lyrik
'qual' => Lyrik
'grausne' => Lyrik
'Lust' => Lyrik
'kichern' => Lyrik
'zufrieden' => Lyrik
'leid' => Lyrik
'zorn' => Lyrik
'heiter' => Lyrik
'lache' => Lyrik
'weinen' => Lyrik
'sehnsucht' => Lyrik
'hoffnung' => Lyrik
'mutig' => Lyrik
'trost' => Lyrik
'ohnmacht' => Lyrik
'ohnmächtig' => Lyrik
```

0.2.2 mfw_Balladen

[23]:
```
docs_new = ['stehen', 'herz', 'kind', 'könig', 'alt', 'liegen', 'nacht', 'auge',
            'hand', 'rufen', 'arm', 'schön', 'tragen', 'jung', 'still', 'schauen',
            'licht', 'schwer', 'schlagen', 'tief']
X_new_counts = count_vect.transform(docs_new)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)

predicted = clf.predict(X_new_tfidf)
```

```
for doc, category in zip(docs_new, predicted):
     print('%r => %s' % (doc, docs_to_train.target_names[category]))
```

```
'stehen' => Lyrik
'herz' => Lyrik
'kind' => Lyrik
'könig' => Balladen
'alt' => Lyrik
'liegen' => Lyrik
'nacht' => Lyrik
'auge' => Lyrik
'hand' => Lyrik
'rufen' => Balladen
'arm' => Lyrik
'schön' => Lyrik
'tragen' => Lyrik
'jung' => Lyrik
'still' => Lyrik
'schauen' => Lyrik
'licht' => Lyrik
'schwer' => Lyrik
'schlagen' => Lyrik
'tief' => Lyrik
```

mfw_Lyrik

[24]:
```
docs_new = ['herz', 'lied', 'seele', 'nacht', 'dunkel', 'küssen',
            'singen', 'auge', 'voll', 'süß', 'schön', 'tief',
            'glück', 'liegen', 'nieder', 'stehen', 'arm', 'sehnen',
            'meer', 'stern']
X_new_counts = count_vect.transform(docs_new)
X_new_tfidf = tfidf_transformer.transform(X_new_counts)

predicted = clf.predict(X_new_tfidf)

for doc, category in zip(docs_new, predicted):
     print('%r => %s' % (doc, docs_to_train.target_names[category]))
```

```
'herz' => Lyrik
'lied' => Lyrik
'seele' => Lyrik
'nacht' => Lyrik
'dunkel' => Lyrik
'küssen' => Lyrik
'singen' => Lyrik
'auge' => Lyrik
'voll' => Lyrik
```

```
'süß' => Lyrik
'schön' => Lyrik
'tief' => Lyrik
'glück' => Lyrik
'liegen' => Lyrik
'nieder' => Lyrik
'stehen' => Lyrik
'arm' => Lyrik
'sehnen' => Lyrik
'meer' => Lyrik
'stern' => Lyrik
```

0.3 Building a pipeline

[25]:
```
from sklearn.pipeline import Pipeline
```

[26]:
```
text_clf = Pipeline([
('vect', CountVectorizer()),
('tfidf', TfidfTransformer()),
('clf', MultinomialNB()),
])
```

[27]:
```
text_clf.fit(docs_to_train.data, docs_to_train.target)
```

[27]:
```
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf', MultinomialNB())])
```

0.4 Evaluation of the performance on the test set

[28]:
```
import numpy as np
```

[29]:
```
docs_to_test = sklearn.datasets.load_files("/home/piah/Dokumente/Uni/
 ↪Projektarbeit/Projektarbeit_LyrikGattungszuweisung/scikit-learn/angepasst/",␣
 ↪description=None, categories=categories, load_content=True, shuffle=True,␣
 ↪encoding='utf-8', decode_error='strict', random_state=42)
>>> docs_test = docs_to_test.data
>>> predicted = text_clf.predict(docs_test)
>>> np.mean(predicted == docs_to_test.target)
```

[29]:
```
0.7390029325513197
```

0.4.1 SVM (support vector machine)

[30]:
```
from sklearn.linear_model import SGDClassifier
```

[31]:
```
>>> text_clf = Pipeline([
...     ('vect', CountVectorizer()),
...     ('tfidf', TfidfTransformer()),
...     ('clf', SGDClassifier(loss='hinge', penalty='l2',
...                           alpha=1e-3, random_state=42,
...                           max_iter=5, tol=None)),
... ])

>>> text_clf.fit(docs_to_train.data, docs_to_train.target)
```

[31]:
```
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf',
                 SGDClassifier(alpha=0.001, max_iter=5, random_state=42,
                               tol=None))])
```

[32]:
```
>>> predicted = text_clf.predict(docs_test)
>>> np.mean(predicted == docs_to_test.target)
```

[32]:
```
0.9794721407624634
```

0.4.2 metrics

[33]:
```
from sklearn import metrics
```

[34]:
```
 print(metrics.classification_report(docs_to_test.target, predicted,
...     target_names=docs_to_test.target_names))
```

```
              precision    recall  f1-score   support

    Balladen       0.95      1.00      0.97       135
       Lyrik       1.00      0.97      0.98       206

    accuracy                           0.98       341
   macro avg       0.98      0.98      0.98       341
weighted avg       0.98      0.98      0.98       341
```

[35]:
```
metrics.confusion_matrix(docs_to_test.target, predicted)
```

[35]:
```
array([[135,   0],
       [  7, 199]])
```

0.5 Parameter tuning using grid search

[36]:
```
from sklearn.model_selection import GridSearchCV
```

[37]:
```
parameters = {
...     'vect__ngram_range': [(1, 1), (1, 2)],
...     'tfidf__use_idf': (True, False),
...     'clf__alpha': (1e-2, 1e-3),
... }
```

[39]:
```
gs_clf = GridSearchCV(text_clf, parameters, cv=5, n_jobs=-1)
```

[40]:
```
gs_clf = gs_clf.fit(docs_to_train.data[:400], docs_to_train.target[:400])
```

[50]:
```
docs_to_train.target_names[gs_clf.predict([' hören  es klagen die flöte wieder␣
 ↪  hören  es klagen die flöte wieder  und die kühl brunnen rauschen  golden␣
 ↪wehn die ton nieder  stille  still  laß wir lauschen  hold bitte|bitten  ␣
 ↪mild verlangen  wie es süß zu+die herz sprechen  durch die nacht  die ich␣
 ↪umfangen  blicken zu ich die ton licht  '])[0]]
```

[50]:
```
'Lyrik'
```

[36]:
```
 gs_clf.best_score_
```

[36]:
```
0.8825
```

[37]:
```
for param_name in sorted(parameters.keys()):
...     print("%s: %r" % (param_name, gs_clf.best_params_[param_name]))
```

```
clf__alpha: 0.01
tfidf__use_idf: True
vect__ngram_range: (1, 2)
```