In [1]:

```
%load_ext watermark
%watermark -a 'Sebastian Raschka' -u -d -v -p numpy,pandas,matplotlib,sklearn,nltk
```

```
UsageError: unrecognized arguments: Raschka'
```

In [2]:

```
# Added version check for recent scikit-learn 0.18 checks
from distutils.version import LooseVersion as Version
from sklearn import __version__ as sklearn_version
```

In [3]:

```
import pyprind
import pandas as pd
import os

# change the `basepath` to the directory of the
# unzipped movie dataset

#basepath = '/Users/Sebastian/Desktop/aclImdb/'
basepath = './aclImdb'

labels = {'pos': 1, 'neg': 0}
pbar = pyprind.ProgBar(50000)
df = pd.DataFrame()
for s in ('test', 'train'):
    for l in ('pos', 'neg'):
        path = os.path.join(basepath, s, l)
        for file in os.listdir(path):
            with open(os.path.join(path, file), 'r', encoding='utf-8') as infile:
                txt = infile.read()
            df = df.append([[txt, labels[l]]], ignore_index=True)
            pbar.update()
df.columns = ['review', 'sentiment']
```

```
0% [##############################] 100% | ETA: 00:00:00
Total time elapsed: 00:06:16
```

In [4]:

```
import numpy as np

np.random.seed(0)
df = df.reindex(np.random.permutation(df.index))
```

In [9]:

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer

count = CountVectorizer()
docs = np.array([
        'The sun is shining',
        'The weather is sweet',
        'The sun is shining, the weather is sweet, and one and one is two'])
bag = count.fit_transform(docs)
```

In [10]:

```
print(count.vocabulary_)
```

```
{'the': 6, 'sun': 4, 'is': 1, 'shining': 3, 'weather': 8, 'sweet': 5, 'and': 0, 'o
ne': 2, 'two': 7}
```

In [11]:

```
print(bag.toarray())
```

```
[[0 1 0 1 1 0 1 0 0]
 [0 1 0 0 0 1 1 0 1]
 [2 3 2 1 1 1 2 1 1]]
```

In [12]:

```
np.set_printoptions(precision=2)
```

In [13]:

```
from sklearn.feature_extraction.text import TfidfTransformer

tfidf = TfidfTransformer(use_idf=True, norm='l2', smooth_idf=True)
print(tfidf.fit_transform(count.fit_transform(docs)).toarray())
```

```
[[ 0.    0.43  0.    0.56  0.56  0.    0.43  0.    0.  ]
 [ 0.    0.43  0.    0.    0.    0.56  0.43  0.    0.56]
 [ 0.5   0.45  0.5   0.19  0.19  0.19  0.3   0.25  0.19]]
```

In [14]:

```
tf_is = 3
n_docs = 3
idf_is = np.log((n_docs+1) / (3+1))
tfidf_is = tf_is * (idf_is + 1)
print('tf-idf of term "is" = %.2f' % tfidf_is)
```

```
tf-idf of term "is" = 3.00
```

In [15]:

```
tfidf = TfidfTransformer(use_idf=True, norm=None, smooth_idf=True)
raw_tfidf = tfidf.fit_transform(count.fit_transform(docs)).toarray()[-1]
raw_tfidf
```

Out[15]:

```
array([ 3.39,  3.  ,  3.39,  1.29,  1.29,  1.29,  2.  ,  1.69,  1.29])
```

In [16]:

```
l2_tfidf = raw_tfidf / np.sqrt(np.sum(raw_tfidf**2))
l2_tfidf
```

Out[16]:

```
array([ 0.5 ,  0.45,  0.5 ,  0.19,  0.19,  0.19,  0.3 ,  0.25,  0.19])
```

In [17]:

```
df.loc[0, 'review'][-50:]
```

Out[17]:

```
'and I suggest that you go see it before you judge.'
```

In [18]:

```
import re
def preprocessor(text):
    text = re.sub('<[^>]*>', '', text)
    emoticons = re.findall('(?::|;|=)(?:-)?(?:¥)|¥(|D|P)', text)
    text = re.sub('[¥W]+', ' ', text.lower()) +¥
        ' '.join(emoticons).replace('-', '')
    return text
```

In [19]:

```
preprocessor(df.loc[0, 'review'][-50:])
```

Out[19]:

```
'and i suggest that you go see it before you judge '
```

In [20]:

```
preprocessor("</a>This :) is :( a test :-)!")
```

Out[20]:

```
'this is a test :) :( :)'
```

In [21]:

```
df['review'] = df['review'].apply(preprocessor)
```

In [22]:

```
from nltk.stem.porter import PorterStemmer

porter = PorterStemmer()

def tokenizer(text):
    return text.split()


def tokenizer_porter(text):
    return [porter.stem(word) for word in text.split()]
```

In [23]:

```
tokenizer('runners like running and thus they run')
```

Out[23]:

```
['runners', 'like', 'running', 'and', 'thus', 'they', 'run']
```

In [24]:

```
tokenizer_porter('runners like running and thus they run')
```

Out[24]:

```
['runner', 'like', 'run', 'and', 'thu', 'they', 'run']
```

In [25]:

```
import nltk

nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to C:¥Users¥watanabe
[nltk_data]     shinichi¥AppData¥Roaming¥nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[25]:

```
True
```

In [26]:

```
from nltk.corpus import stopwords

stop = stopwords.words('english')
[w for w in tokenizer_porter('a runner likes running and runs a lot')[-10:]
if w not in stop]
```

Out[26]:

```
['runner', 'like', 'run', 'run', 'lot']
```

In [27]:

```
X_train = df.loc[:25000, 'review'].values
y_train = df.loc[:25000, 'sentiment'].values
X_test = df.loc[25000:, 'review'].values
y_test = df.loc[25000:, 'sentiment'].values
```