Sentiment Analysis

In [1]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.svm import LinearSVC

from sklearn.pipeline import Pipeline

from sklearn.datasets import load_files

from sklearn.model_selection import train_test_split

from sklearn import metrics
```

In [2]:
```
movie_reviews_data_folder = 'data/txt_sentoken'
dataset = load_files(movie_reviews_data_folder, shuffle=False)

print("n_samples: {}".format(len(dataset.data)))
```

```
n_samples: 2000
```

In [3]:
```
print(dataset.data[-1][:1000])
```

```
b'truman ( " true-man " ) burbank is the perfect name for jim carrey\'s character in this film . \npresident
truman was an unassuming man who became known worldwide , in spite of ( or was it because of ) his stature .
\n " truman " also recalls an era of plenty following a grim war , an era when planned communities built by g
overnment scientists promised an idyllic life for americans . \nand burbank , california , brings to mind the
tonight show and the home of nbc . \nif hollywood is the center of the film world , burbank is , or was , the
center of tv\'s world , the world where our protagonist lives . \ncombine all these names and concepts into "
truman burbank , " and you get something that well describes him and his artificial world . \ntruman leads th
e perfect life . \nhis town , his car , and his wife are picture perfect . \nhis idea of reality comes under
attack one day when a studio light falls from the sky . \nthe radio explains that an overflying airplane star
ted coming apart . \n . \n . \nb'
```

In [4]:
```
dataset.target[:5]
```

Out[4]:
```
array([0, 0, 0, 0, 0])
```

In [5]:
```
dataset.target[-5:]
```

Out[5]:
```
array([1, 1, 1, 1, 1])
```

In [6]:
```
dataset.target_names[:5]
```

Out[6]:
```
['neg', 'pos']
```

In [7]:
```
# split the dataset in training and test set:
docs_train, docs_test, y_train, y_test = train_test_split(
    dataset.data, dataset.target, test_size=0.25, random_state=1)
```

- Build a vectorizer / classifier pipeline that filters out tokens that are too rare or too frequent

In [8]:
```
pipeline = Pipeline([
    ('vect', TfidfVectorizer(ngram_range=(1, 2), min_df=3, max_df=0.95)),
    ('clf', LinearSVC(C=10000))
    ])
```

- Build a grid search to find out whether unigrams or bigrams are more useful.
- Fit the pipeline on the training set using grid search for the parameters

In [9]:
```
pipeline.fit(docs_train, y_train);
```

- Predict the test set

In [10]:
```
y_predicted = pipeline.predict(docs_test)
```

In [11]:
```
    # Print the classification report
print(metrics.classification_report(y_test, y_predicted,
                                    target_names=dataset.target_names))
```

```
              precision    recall  f1-score   support

         neg       0.88      0.84      0.86       255
         pos       0.84      0.88      0.86       245

    accuracy                           0.86       500
   macro avg       0.86      0.86      0.86       500
weighted avg       0.86      0.86      0.86       500
```

In [12]:
```
# Print and plot the confusion matrix
cm = metrics.confusion_matrix(y_test, y_predicted)
print(cm)
```

```
[[214  41]
 [ 29 216]]
```

In [13]:
```
sentences = [
    'The movie has an abrupt ending.',
    'The movie is awesome',
    'The movie is boring',
    'The movie will be a blockbuster'
]
predicted = pipeline.predict(sentences)
```

In [14]:
```
for s, p in zip(sentences, predicted):
    print("The language of {} is '{}'".format(s, dataset.target_names[p]))
```

```
The language of The movie has an abrupt ending. is 'neg'
The language of The movie is awesome is 'pos'
The language of The movie is boring is 'neg'
The language of The movie will be a blockbuster is 'pos'
```

In [ ]:
```

```