In [1]:
```
# Text classification, positive and negative words
import nltk
import random
import pickle
from nltk.corpus import movie_reviews

documents = []

# for each sub-folder in movie reviews, pos and neg
for category in movie_reviews.categories():
    # for each file *movie review, in each sub-folder
    for fileid in movie_reviews.fileids(category):
        # create a tuple containing every word in the current file, and the ca
tegory (the name of the sub-folder pos and neg)
        item = (movie_reviews.words(fileid), category)
        # add the tuple to the documents list
        documents.append(item)
# print("before shuffle", documents)
# shuffle the documents list so that pos/neg reviews are randomly distributed
random.shuffle(documents)
# print("after shuffle", documents)

all_words = []
# for every word in all movie reviews
for w in movie_reviews.words():
    # convert the current word to lower case
    all_words.append(w.lower())

# obtain a frequency distribution object for all words (this is a dictionary c
ontaining word:numberOfOccurences)
all_words = nltk.FreqDist(all_words)

# step 4 obtain a list of the top 3000 most common words, so we can iterate th
rough it in a for loop
common_words = []
for i in list(all_words.most_common(3000)):
    # save just the word, stripping away the number of occurences
    common_words.append(i[0])

# step 5 this function takes in list of words from a document
def find_features(document):
    # get a set of words from the document - ie. remove all duplicates
    unique_words_in_document = set(document)
    features = {}
    # for each of the top 3000 most common words
    for w in common_words:
        features[w] = (w in unique_words_in_document)
    return features
# the output of this method will be a dict containing each common word, and wh
ether the word is present or not in current doc
# print(find_features(movie_reviews.words("neg/cv070_13249.txt"))) # <- exampl
ary on only ONE movie review

# step 6 now we calculate features for all files and record whether the review
was neg or pos
featuresets = []
```

```
for(review_words, review_category) in documents:
    featuresets.append((find_features(review_words), review_category))

# step 7 split the labelled dataset (The categories) into a test set (500 rema
ining) and 1500 randomised as training
training_set = featuresets[:1500]
test_set = featuresets[1500:]

classifier = nltk.NaiveBayesClassifier.train(training_set)
# NEW save the classifier
save_classifier = open("naivebayes.pickle", "wb")
pickle.dump(classifier, save_classifier)
save_classifier.close()


accuracy = nltk.classify.accuracy(classifier, test_set)
print(accuracy)
print("The classifier could classify 500 test movies as either pos or neg with
following ")
# how correctly the classifier could the 500 test movie reviews classify as ei
ther pos or neg
print("accuracy:", accuracy*100, "%")

# step 8 results will vary, due to randomisation of test vs training set
# we can see which features were most useful for differentiating between pos a
nd neg reviews>
print("These informative features were most useful for differentiating between
pos and neg:")
classifier.show_most_informative_features(15) # show this many

# 66 seconds on Surface 2-core
# 43 seconds on 6-core
```

```
0.82
The classifier could classify 500 test movies as either pos or neg with follo
wing
accuracy: 82.0 %
These informative features were most useful for differentiating between pos a
nd neg:
Most Informative Features
                   inept = True              neg : pos    =     19.9 : 1.0
              whatsoever = True              neg : pos    =     11.6 : 1.0
                 idiotic = True              neg : pos    =     10.7 : 1.0
                religion = True              pos : neg    =      8.1 : 1.0
                  finest = True              pos : neg    =      7.3 : 1.0
             outstanding = True              pos : neg    =      7.3 : 1.0
                  prinze = True              neg : pos    =      6.7 : 1.0
             wonderfully = True              pos : neg    =      6.5 : 1.0
                   jedi = True              pos : neg    =      6.5 : 1.0
                 mature = True              pos : neg    =      6.5 : 1.0
                 seagal = True              neg : pos    =      6.3 : 1.0
                  mulan = True              pos : neg    =      6.2 : 1.0
                 spacey = True              pos : neg    =      6.1 : 1.0
                   anna = True              pos : neg    =      6.1 : 1.0
             beautifully = True              pos : neg    =      6.0 : 1.0
```

In [3]:
```
# Load the classifier
classifier_f = open("naivebayes.pickle", "rb")
classifier = pickle.load(classifier_f)
classifier_f.close();

# followed by these lines from previous solution, calculate accuracy
# THIS SHOULD GO MUCH FASTER btw
accuracy = nltk.classify.accuracy(classifier, test_set)
print(accuracy)
print("The classifier could classify 500 test movies as either pos or neg with
following ")
print("accuracy:", accuracy*100, "%")
classifier.show_most_informative_features(15) # show this many

# correct, much faster: 12 seconds on Surface 2-core
# 9 seconds on 6-core
```

```
0.82
The classifier could classify 500 test movies as either pos or neg with follo
wing
accuracy: 82.0 %
Most Informative Features
                    inept = True              neg : pos    =     19.9 : 1.0
               whatsoever = True              neg : pos    =     11.6 : 1.0
                  idiotic = True              neg : pos    =     10.7 : 1.0
                 religion = True              pos : neg    =      8.1 : 1.0
                   finest = True              pos : neg    =      7.3 : 1.0
              outstanding = True              pos : neg    =      7.3 : 1.0
                   prinze = True              neg : pos    =      6.7 : 1.0
              wonderfully = True              pos : neg    =      6.5 : 1.0
                     jedi = True              pos : neg    =      6.5 : 1.0
                   mature = True              pos : neg    =      6.5 : 1.0
                   seagal = True              neg : pos    =      6.3 : 1.0
                    mulan = True              pos : neg    =      6.2 : 1.0
                   spacey = True              pos : neg    =      6.1 : 1.0
                     anna = True              pos : neg    =      6.1 : 1.0
              beautifully = True              pos : neg    =      6.0 : 1.0
```