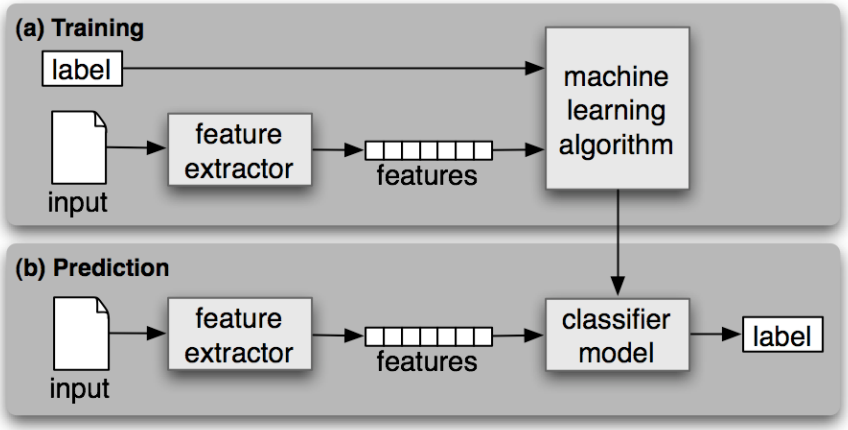
Text Classification

In [1]:
```
import nltk
```

- Supervised Classification



Gender Name Identification

In [2]:
```
def gender_features(word):
    return {'last_letter': word[-1]}
```

In [3]:
```
gender_features('Alice')
```

Out[3]:
```
{'last_letter': 'e'}
```

In [4]:
```
from nltk.corpus import names
```

In [5]:
```
names.words('male.txt')[:5]
```

Out[5]:
```
['Aamir', 'Aaron', 'Abbey', 'Abbie', 'Abbot']
```

In [6]:
```
names.words('female.txt')[-5:]
```

Out[6]:
```
['Zorine', 'Zsa Zsa', 'Zsazsa', 'Zulema', 'Zuzana']
```

In [7]:
```
labeled_names = (
    [(name, 'male') for name in names.words('male.txt')] +
    [(name, 'female') for name in names.words('female.txt')]
)
```

In [8]:
```
labeled_names[:5]
```

Out[8]:
```
[('Aamir', 'male'),
 ('Aaron', 'male'),
 ('Abbey', 'male'),
 ('Abbie', 'male'),
 ('Abbot', 'male')]
```

In [9]:
```
labeled_names[-5:]
```

Out[9]:
```
[('Zorine', 'female'),
 ('Zsa Zsa', 'female'),
 ('Zsazsa', 'female'),
 ('Zulema', 'female'),
 ('Zuzana', 'female')]
```

In [10]:
```
import random
```

In [11]:
```
random.seed(321)
labeled_names = (
    [(name, 'male') for name in names.words('male.txt')] +
    [(name, 'female') for name in names.words('female.txt')]
)
random.shuffle(labeled_names)
```

In [12]:
```
labeled_names[:5]
```

Out[12]:
```
[('Ebony', 'female'),
 ('Thaddius', 'male'),
 ('Theodora', 'female'),
 ('Helli', 'female'),
 ('Ugo', 'male')]
```

In [13]:
```
featuresets = [(gender_features(n), gender)
               for (n, gender) in labeled_names]
```

In [14]:
```
featuresets[:5]
```

Out[14]:
```
[({'last_letter': 'y'}, 'female'),
 ({'last_letter': 's'}, 'male'),
 ({'last_letter': 'a'}, 'female'),
 ({'last_letter': 'i'}, 'female'),
 ({'last_letter': 'o'}, 'male')]
```

In [15]:
```
train_set, test_set = featuresets[500:], featuresets[:500]
```

In [16]:
```
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

In [17]:
```
classifier.classify(gender_features('Neo'))
```

Out[17]:
```
'male'
```

In [18]:
```
classifier.classify(gender_features('Trinity'))
```

Out[18]:
```
'female'
```

In [19]:
```
print(nltk.classify.accuracy(classifier, test_set))
```

```
0.77
```

In [20]:
```
classifier.show_most_informative_features(10)
```

```
Most Informative Features
             last_letter = 'a'            female : male   =     37.1 : 1.0
             last_letter = 'k'              male : female =     30.6 : 1.0
             last_letter = 'v'              male : female =     17.4 : 1.0
             last_letter = 'f'              male : female =     16.5 : 1.0
             last_letter = 'p'              male : female =     11.1 : 1.0
             last_letter = 'd'              male : female =      9.6 : 1.0
             last_letter = 'm'              male : female =      8.3 : 1.0
             last_letter = 'o'              male : female =      7.6 : 1.0
             last_letter = 'r'              male : female =      6.5 : 1.0
             last_letter = 'g'              male : female =      5.1 : 1.0
```

In [ ]:

In [21]:
```
actual_labels = []
```

In [22]:
```
observed_labels = []
```

In [23]:
```
test_set[:3]
```

Out[23]:
```
[({'last_letter': 'y'}, 'female'),
 ({'last_letter': 's'}, 'male'),
 ({'last_letter': 'a'}, 'female')]
```

In [24]:
```
for i, (feats, label) in enumerate(test_set):
    actual_labels.append(label)
    observed = classifier.classify(feats)
    observed_labels.append(observed)
```

In [25]:
```
actual_labels[:10]
```

Out[25]:
```
['female',
 'male',
 'female',
 'female',
 'male',
 'female',
 'female',
 'female',
 'male',
 'female']
```

In [26]:
```
observed_labels[:10]
```

Out[26]:
```
['female',
 'male',
 'female',
 'female',
 'male',
 'male',
 'female',
 'female',
 'male',
 'female']
```

In [27]:
```
cm = nltk.ConfusionMatrix(actual_labels, observed_labels)
```

In [28]:
```
print(cm)
```

```
        |   f      |
        |   e      |
        |   m   m  |
        |   a   a  |
        |   l   l  |
        |   e   e  |
-------+---------+
female |<267> 58  |
  male |  57<118>|
-------+---------+
(row = reference; col = test)
```

In [ ]:

In [ ]:

In [29]:
```
# Overfitting features

def gender_features2(name):
    features = {}
    features["first_letter"] = name[0].lower()
    features["last_letter"] = name[-1].lower()
    for letter in 'abcdefghijklmnopqrstuvwxyz':
        features["count({})".format(letter)] = name.lower().count(letter)
        features["has({})".format(letter)] = (letter in name.lower())
    return features
```

In [30]:
```
gender_features2('Alice')
```

Out[30]:
```
{'first_letter': 'a',
 'last_letter': 'e',
 'count(a)': 1,
 'has(a)': True,
 'count(b)': 0,
 'has(b)': False,
 'count(c)': 1,
 'has(c)': True,
 'count(d)': 0,
 'has(d)': False,
 'count(e)': 1,
 'has(e)': True,
 'count(f)': 0,
 'has(f)': False,
 'count(g)': 0,
 'has(g)': False,
 'count(h)': 0,
 'has(h)': False,
 'count(i)': 1,
 'has(i)': True,
 'count(j)': 0,
 'has(j)': False,
 'count(k)': 0,
 'has(k)': False,
 'count(l)': 1,
 'has(l)': True,
 'count(m)': 0,
 'has(m)': False,
 'count(n)': 0,
 'has(n)': False,
 'count(o)': 0,
 'has(o)': False,
 'count(p)': 0,
 'has(p)': False,
 'count(q)': 0,
 'has(q)': False,
 'count(r)': 0,
 'has(r)': False,
 'count(s)': 0,
 'has(s)': False,
 'count(t)': 0,
 'has(t)': False,
 'count(u)': 0,
 'has(u)': False,
```

In [31]:
```
featuresets = [(gender_features2(n), gender)
               for (n, gender) in labeled_names]
```

In [32]:
```
train_set, test_set = featuresets[500:], featuresets[:500]
```

In [33]:
```
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

In [34]:
```
print(nltk.classify.accuracy(classifier, test_set))
```

```
0.784
```

In [35]:
```
test_set[0]
```

Out[35]:
```
({'first_letter': 'e',
  'last_letter': 'y',
  'count(a)': 0,
  'has(a)': False,
  'count(b)': 1,
  'has(b)': True,
  'count(c)': 0,
  'has(c)': False,
  'count(d)': 0,
  'has(d)': False,
  'count(e)': 1,
  'has(e)': True,
  'count(f)': 0,
  'has(f)': False,
  'count(g)': 0,
  'has(g)': False,
  'count(h)': 0,
  'has(h)': False,
  'count(i)': 0,
  'has(i)': False,
  'count(j)': 0,
  'has(j)': False,
  'count(k)': 0,
  'has(k)': False,
  'count(l)': 0,
  'has(l)': False,
  'count(m)': 0,
  'has(m)': False,
  'count(n)': 1,
  'has(n)': True,
  'count(o)': 1,
  'has(o)': True,
  'count(p)': 0,
  'has(p)': False,
  'count(q)': 0,
  'has(q)': False,
  'count(r)': 0,
  'has(r)': False,
  'count(s)': 0,
  'has(s)': False,
  'count(t)': 0,
  'has(t)': False,
  'count(u)': 0,
  'has(u)': False,
```

In [36]:
```
train_names = labeled_names[500:]
test_names = labeled_names[:500]

errors = []
for (name, tag) in test_names:
     guess = classifier.classify(gender_features2(name))
     if guess != tag:
          errors.append( (tag, guess, name) )
```

In [37]:
```
for (tag, guess, name) in sorted(errors):
    print('correct={:<8} guess={:<8s} name={:<30}'.format(tag, guess, name))
```

```
correct=female   guess=male     name=Ag
correct=female   guess=male     name=Beau
correct=female   guess=male     name=Berget
correct=female   guess=male     name=Berry
correct=female   guess=male     name=Beth
correct=female   guess=male     name=Bev
correct=female   guess=male     name=Birgit
correct=female   guess=male     name=Brynn
correct=female   guess=male     name=Charlot
correct=female   guess=male     name=Christen
correct=female   guess=male     name=Chrystal
correct=female   guess=male     name=Daffy
correct=female   guess=male     name=Diamond
correct=female   guess=male     name=Dion
correct=female   guess=male     name=Dorit
correct=female   guess=male     name=Dorry
correct=female   guess=male     name=Ebony
correct=female   guess=male     name=Ermentrude
correct=female   guess=male     name=Fawn
```

In [38]:
```
def gender_features(word):
     return {'suffix1': word[-1:],
             'suffix2': word[-2:]}
```

In [39]:
```
featuresets = [(gender_features(n), gender)
               for (n, gender) in labeled_names]
```

In [40]:
```
train_set, test_set = featuresets[500:], featuresets[:500]
```

In [41]:
```
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

In [42]:
```
print(nltk.classify.accuracy(classifier, test_set))
```

```
0.786
```

In [43]:
```
errors = []
for (name, tag) in test_names:
     guess = classifier.classify(gender_features(name))
     if guess != tag:
         errors.append( (tag, guess, name) )
```

In [44]:
```
for (tag, guess, name) in sorted(errors):
    print('correct={:<8} guess={:<8s} name={:<30}'.format(tag, guess, name))
```

```
correct=female   guess=male     name=Adel
correct=female   guess=male     name=Adriaens
correct=female   guess=male     name=Ag
correct=female   guess=male     name=Ainsley
correct=female   guess=male     name=Allis
correct=female   guess=male     name=April
correct=female   guess=male     name=Beau
correct=female   guess=male     name=Berget
correct=female   guess=male     name=Berry
correct=female   guess=male     name=Bev
correct=female   guess=male     name=Birgit
correct=female   guess=male     name=Cass
correct=female   guess=male     name=Cathleen
correct=female   guess=male     name=Charin
correct=female   guess=male     name=Charlot
correct=female   guess=male     name=Christen
correct=female   guess=male     name=Chrystal
correct=female   guess=male     name=Cynthy
correct=female   guess=male     name=Daffy
```

In [ ]:
```
```

Document Classification

In [45]:
```
from nltk.corpus import movie_reviews
```

In [46]:
```
movie_reviews.categories()
```

Out[46]:
```
['neg', 'pos']
```

In [47]:
```
len(movie_reviews.fileids('neg')), len(movie_reviews.fileids('pos'))
```

Out[47]:
```
(1000, 1000)
```

In [48]:
```
movie_reviews.fileids('neg')[:5]
```

Out[48]:
```
['neg/cv000_29416.txt',
 'neg/cv001_19502.txt',
 'neg/cv002_17424.txt',
 'neg/cv003_12683.txt',
 'neg/cv004_12641.txt']
```

In [49]:
```
print(movie_reviews.words('neg/cv000_29416.txt')[:25])
```

```
['plot', ':', 'two', 'teen', 'couples', 'go', 'to', 'a', 'church', 'party', ',', 'drink', 'and', 'then', 'drive
', '.', 'they', 'get', 'into', 'an', 'accident', '.', 'one', 'of', 'the']
```

In [50]:
```
movie_reviews.fileids('pos')[:5]
```

Out[50]:
```
['pos/cv000_29590.txt',
 'pos/cv001_18431.txt',
 'pos/cv002_15918.txt',
 'pos/cv003_11664.txt',
 'pos/cv004_11636.txt']
```

In [51]:
```
print(movie_reviews.words('pos/cv000_29590.txt')[:25])
```

```
['films', 'adapted', 'from', 'comic', 'books', 'have', 'had', 'plenty', 'of', 'success', ',', 'whether', 'they
', "'", 're', 'about', 'superheroes', '(', 'batman', ',', 'superman', ',', 'spawn', ')', ',']
```

In [52]:
```
documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]
```

In [53]:
```
random.shuffle(documents)
```

- FreqDist

In [54]:
```
x = ['hello', 'how', 'hello', 'are', 'hello']
y = nltk.FreqDist(w.lower() for w in x)
y
```

Out[54]:
```
FreqDist({'hello': 3, 'how': 1, 'are': 1})
```

In [55]:
```
list(y)
```

Out[55]:
```
['hello', 'how', 'are']
```

In [ ]:
```

```

In [56]:
```
all_words = nltk.FreqDist(w.lower() for w in movie_reviews.words())
```

In [57]:
```
# list of the 2000 most frequent words in the overall corpus
word_features = list(all_words)[:2000]
```

In [58]:
```
def document_features(document):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    return features
```

In [59]:
```
print(document_features(movie_reviews.words('pos/cv000_29590.txt')))
```

```
{'contains(plot)': False, 'contains(:)': True, 'contains(two)': False, 'contains(teen)': False, 'contains(couples)': False, 'contains(go)': True, 'contains(to)': True, 'contains(a)': True, 'contains(church)': False, 'contains(party)': False, 'contains(,)': True, 'contains(drink)': False, 'contains(and)': True, 'contains(then)': False, 'contains(drive)': False, 'contains(.)': True, 'contains(they)': True, 'contains(get)': True, 'contains(into)': True, 'contains(an)': True, 'contains(accident)': False, 'contains(one)': True, 'contains(of)': True, 'contains(the)': True, 'contains(guys)': False, 'contains(dies)': False, 'contains(but)': True, 'contains(his)': False, 'contains(girlfriend)': False, 'contains(continues)': False, 'contains(see)': True, 'contains(him)': True, 'contains(in)': True, 'contains(her)': True, 'contains(life)': False, 'contains(has)': True, 'contains(nightmares)': False, 'contains(what)': False, "contains(')": True, 'contains(s)': True, 'contains(deal)': False, 'contains(?)': True, 'contains(watch)': True, 'contains(movie)': False, 'contains(")': True, 'contains(sorta)': False, 'contains(find)': True, 'contains(out)': False, 'contains(critique)': False, 'contains(mind)': False, 'contains(-)': True, 'contains(fuck)': False, 'contains(for)': True, 'contains(generation)': False, 'contains(that)': True, 'contains(touches)': False, 'contains(on)': True, 'contains(very)': True, 'contains(cool)': False, 'contains(idea)': False, 'contains(presents)': False, 'contains(it)': True, 'contains(bad)': True, 'contains(package)': False, 'contains(which)': False, 'contains(is)': True, 'contains(makes)': False, 'contains(this)': True, 'contains(review)': False, 'contains(even)': True, 'contains(harder)': False, 'contains(write)': False, 'contains(since)': False, 'contains(i)': True, 'contains(generally)': False, 'contains(applaud)': False, 'contains(films)': True, 'contains(attempt)': True, 'contains(break)': False, 'contains(mold)': False, 'contains(mess)': False, 'contains(with)': True, 'contains(your)': False, 'contains(head)': False, 'contains(such)': True, 'contains
```

In [60]:
```
featuresets = [(document_features(d), c) for (d,c) in documents]
```

In [61]:
```
train_set, test_set = featuresets[100:], featuresets[:100]
```

In [62]:
```
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

In [63]:
```
print(nltk.classify.accuracy(classifier, test_set))
```

```
0.81
```

In [64]:
```
classifier.show_most_informative_features(50)
```

```
Most Informative Features
    contains(schumacher) = True              neg : pos    =     12.5 : 1.0
     contains(atrocious) = True              neg : pos    =      7.1 : 1.0
        contains(shoddy) = True              neg : pos    =      7.1 : 1.0
        contains(turkey) = True              neg : pos    =      6.4 : 1.0
        contains(suvari) = True              neg : pos    =      6.4 : 1.0
          contains(mena) = True              neg : pos    =      6.4 : 1.0
       contains(singers) = True              pos : neg    =      6.3 : 1.0
  contains(surveillance) = True              neg : pos    =      5.7 : 1.0
        contains(sordid) = True              neg : pos    =      5.7 : 1.0
        contains(poorly) = True              neg : pos    =      5.7 : 1.0
     contains(everyday) = True               pos : neg    =      5.6 : 1.0
      contains(unravel) = True               pos : neg    =      5.6 : 1.0
          contains(ugh) = True               neg : pos    =      5.5 : 1.0
       contains(awful) = True                neg : pos    =      5.4 : 1.0
        contains(justin) = True              neg : pos    =      5.1 : 1.0
        contains(canyon) = True              neg : pos    =      5.1 : 1.0
    contains(uninspired) = True              neg : pos    =      5.1 : 1.0
       contains(bronson) = True              neg : pos    =      5.1 : 1.0
     contains(underwood) = True              neg : pos    =      5.1 : 1.0
        contains(waste) = True               neg : pos    =      5.0 : 1.0
       contains(wasted) = True               neg : pos    =      5.0 : 1.0
      contains(miscast) = True               neg : pos    =      4.8 : 1.0
    contains(ridiculous) = True              neg : pos    =      4.8 : 1.0
        contains(welles) = True              neg : pos    =      4.6 : 1.0
      contains(martian) = True               neg : pos    =      4.6 : 1.0
       contains(sexist) = True               neg : pos    =      4.6 : 1.0
       contains(bland) = True                neg : pos    =      4.5 : 1.0
       contains(kudos) = True                pos : neg    =      4.4 : 1.0
      contains(savages) = True               neg : pos    =      4.4 : 1.0
          contains(h20) = True               neg : pos    =      4.4 : 1.0
     contains(runtime) = True                neg : pos    =      4.4 : 1.0
  contains(unimpressive) = True              neg : pos    =      4.4 : 1.0
      contains(banality) = True              neg : pos    =      4.4 : 1.0
       contains(seymour) = True              pos : neg    =      4.3 : 1.0
     contains(explores) = True               pos : neg    =      4.3 : 1.0
        contains(groan) = True               neg : pos    =      4.2 : 1.0
    contains(stretched) = True               neg : pos    =      4.2 : 1.0
        contains(tones) = True               pos : neg    =      4.1 : 1.0
    contains(painfully) = True               neg : pos    =      4.1 : 1.0
      contains(unfunny) = True               neg : pos    =      4.0 : 1.0
         contains(dull) = True               neg : pos    =      3.9 : 1.0
         contains(skip) = True               neg : pos    =      3.9 : 1.0
         contains(mess) = True               neg : pos    =      3.8 : 1.0
```

In [ ]:

In [ ]:

In [ ]: