In [1]:
```
import nltk
from nltk.corpus import names
from pylab import *
import random as pyrandom
```

Sentence Segmentation

In [34]:
```
sents = nltk.corpus.treebank_raw.sents()
sents = [s for s in sents if len(s)>3]
sents = [s for s in sents if "START" not in s]
```

In [35]:
```
tokens = []
boundaries = []
for s in sents:
    tokens += s
    boundaries.append(len(tokens)-1)
```

In [36]:
```
print tokens[:200]
```

```
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the',
'board', 'as', 'a', 'nonexecutive', 'director', 'Nov', '.', '29', '.', 'Mr',
'.', 'Vinken', 'is', 'chairman', 'of', 'Elsevier', 'N', '.', 'V', '.,', 'the',
'Dutch', 'publishing', 'group', '.', 'Rudolph', 'Agnew', ',', '55', 'years',
'old', 'and', 'former', 'chairman', 'of', 'Consolidated', 'Gold', 'Fields',
'PLC', ',', 'was', 'named', 'a', 'nonexecutive', 'director', 'of', 'this',
'British', 'industrial', 'conglomerate', '.', 'A', 'form', 'of', 'asbestos',
'once', 'used', 'to', 'make', 'Kent', 'cigarette', 'filters', 'has', 'caused',
'a', 'high', 'percentage', 'of', 'cancer', 'deaths', 'among', 'a', 'group',
'of', 'workers', 'exposed', 'to', 'it', 'more', 'than', '30', 'years', 'ago',
',', 'researchers', 'reported', '.', 'The', 'asbestos', 'fiber', ',',
'crocidolite', ',', 'is', 'unusually', 'resilient', 'once', 'it', 'enters',
'the', 'lungs', ',', 'with', 'even', 'brief', 'exposures', 'to', 'it',
'causing', 'symptoms', 'that', 'show', 'up', 'decades', 'later', ',',
'researchers', 'said', '.', 'Lorillard', 'Inc', '.,', 'the', 'unit', 'of',
'New', 'York', '-', 'based', 'Loews', 'Corp', '.', 'that', 'makes', 'Kent',
'cigarettes', ',', 'stopped', 'using', 'crocidolite', 'in', 'its', 'Micronite',
'cigarette', 'filters', 'in', '1956', '.', 'Although', 'preliminary',
'findings', 'were', 'reported', 'more', 'than', 'a', 'year', 'ago', ',', 'the',
'latest', 'results', 'appear', 'in', 'today', "'", 's', 'New', 'England',
'Journal', 'of', 'Medicine', ',', 'a', 'forum', 'likely', 'to', 'bring', 'new',
'attention', 'to', 'the', 'problem', '.', 'A', 'Lorillard', 'spokewoman',
'said', ',', '"']
```

In [42]:
```
def features(s,i):
    return dict(current=tokens[i],
                prev=tokens[i-1],
                next=tokens[i+1],
                upper=tokens[i+1][0].isupper(),
                plen=len(tokens[i-1]),
                nlen=len(tokens[i+1]))
```

In [46]:
```
data = []
for i in range(1,len(tokens)-1):
    if tokens[i] not in [".","?","!"]: continue
    c = (i in boundaries)
    f = features(tokens,i)
    data.append((f,c))
pyrandom.shuffle(data)
n = len(data)
print n
training_set = data[n//10:]
test_set = data[:n//10]
```

```
5951
```

In [47]:
```
classifier = nltk.NaiveBayesClassifier.train(training_set)
nltk.classify.accuracy(classifier,test_set)
```

Out[47]:
```
0.9798319327731092
```

In [48]:
```
classifier.classify(features("The quick . brown".split(),2))
```

Out[48]:
```
True
```

In [59]:
```
def segment_sentences(words):
    sentences = [[words[0]]]
    for i in range(1,len(words)):
        sentences[-1].append(words[i])
        c = words[i] in [".","?","!"] and classifier.classify(features(words,i))
        if c: sentences.append([])
    if sentences[-1]==[]: sentences = sentences[:-1]
    return sentences
```

In [61]:
```
segment_sentences("""Smith ran . J . Smith really ran . """.split())
```

Out[61]:
```
[['Smith', 'ran', '.'], ['J', '.', 'Smith', 'really', 'ran', '.']]
```