In [23]:
```
import nltk
from nltk.corpus import names
from pylab import *
import random as pyrandom
```

Parts of Speech Tagging

In [24]:
```
from nltk.corpus import brown
```

In [25]:
```
tagged_words = brown.tagged_words(categories='news')
```

In [44]:
```
def features(s,i,y):
    f = dict(ltag=y[i-1] if i>0 else "^", # previous tag
             lword=s[i-1] if i>0 else "^", # previous word
             s1 = s[i][-1:], # current word features
             s2 = s[i][-2:],
             s3 = s[i][-3:])
    return f
```

Training

In [35]:
```
data = []
for sy in brown.tagged_sents(categories='news'):
    s,y = zip(*sy)
    for i in range(len(s)):
        data.append((features(s,i,y),y[i]))
n = len(data)
training_set = data[n//10:]
test_set = data[:n//10]
```

In [36]:
```
classifier = nltk.NaiveBayesClassifier.train(training_set)
```

In [37]:
```
nltk.classify.accuracy(classifier,test_set)
```

Out[37]: 0.8176031824962705

Greedy Decoding

In [46]:
```
class MyTagger:
    def __init__(self,classifier):
        self.classifier = classifier
    def tag(self,s):
        y = []
        for i in range(len(s)):
            f = features(s,i,y)
            y.append(classifier.classify(features(s,i,y)))
        return zip(s,y)
```

In [40]:
```
tagger = MyTagger(classifier)
```

In [45]:
```
tagger.tag("The quick brown fox jumped over the lazy dogs.".split())
```

Out[45]:
```
[('The', 'AT'),
 ('quick', 'NN'),
 ('brown', 'NN'),
 ('fox', 'NPS-TL'),
 ('jumped', 'VBD'),
 ('over', 'RP'),
 ('the', 'AT'),
 ('lazy', 'JJ'),
 ('dogs.', 'NP')]
```

More Advanced Models

- Viterbi Decoding
- MEMM
- Conditional Random Fields

In [ ]:
```

```