In [26]:

```
#Naive bayes classifier
trainingData = [
    (1, "Chinese Beijing Chinese", True),
    (2, "Chinese Chinese Shanghai", True),
    (3, "Chinese Macao", True),
    (4, "Tokyo Japan Chinese", False)
]

testingData = (5, "Chinese Chinese Chinese Tokyo Japan")
#testingData = (5, "Chinese Tokyo Japan")
```

In [36]:

```
V = list(set([term for _ in trainingData for term in _[1].lower().split()]))

N = len(trainingData)

trueData = [_ for _ in trainingData if _[2]]
falseData = [_ for _ in trainingData if not _[2]]
```

In [37]:

```
len(trueData), len(falseData)
```

Out[37]:

```
(3, 1)
```

In [41]:

```
from collections import defaultdict

# prior = list()
# Tct = list()
# for data in [trueData, falseData]:
#     Nc = len(data)
#     prior.append(Nc/N)
#     termCount = defaultdict(int)

#     for term in data[1].lower().split():
#         termCount[term] += 1

#     Tct.append(termCount)

Tct = defaultdict(int)
for data in trueData:
    Nc = len(trueData)
    PriorC = Nc/N
    for term in data[1].lower().split():
        Tct[term] += 1

_Tct = defaultdict(int)
for data in falseData:
    _Nc = len(falseData)
    _PriorC = _Nc/N
    for term in data[1].lower().split():
        _Tct[term] += 1
```

In [42]:

```
Tct, _Tct
```

Out[42]:

```
(defaultdict(int, {'chinese': 5, 'beijing': 1, 'shanghai': 1, 'macao': 1}),
 defaultdict(int, {'tokyo': 1, 'japan': 1, 'chinese': 1}))
```

In [43]:

```
PriorC, _PriorC
```

Out[43]:

```
(0.75, 0.25)
```

In [44]:

```
condProbC = defaultdict(float)
_condProbC = defaultdict(float)

countSum = sum(Tct.values())
_countSum = sum(_Tct.values())

for term, freq in Tct.items():
    condProbC[term] = (freq+1)/(countSum+len(V))
for term, freq in Tct.items():
    _condProbC[term] = (freq+1)/(_countSum+len(V))
```

In [45]:

```
countSum, _countSum
```

Out[45]:

```
(8, 3)
```

In [46]:

```
condProbC, _condProbC
```

Out[46]:

```
(defaultdict(float,
             {'chinese': 0.42857142857142855,
              'beijing': 0.14285714285714285,
              'shanghai': 0.14285714285714285,
              'macao': 0.14285714285714285}),
 defaultdict(float,
             {'chinese': 0.6666666666666666,
              'beijing': 0.2222222222222222,
              'shanghai': 0.2222222222222222,
              'macao': 0.2222222222222222}))
```

In [47]:

```
from math import log, exp

#prior probability
result = log(PriorC)
_result = log(_PriorC)

#P(C)Multi(P(TCT|C)) -> log(P(C)) + Sum(P(Tct|C))
#Joint prob => conditional independence
for term in testingData[1].lower().split():
    result += log((Tct[term]+1)/(countSum+len(V)))
    _result += log((_Tct[term]+1)/(_countSum+len(V)))

if result > _result:
    print("True", result, exp(result))#exp(_result))
else:
    print("False", _result, exp(_result))
```

```
True -8.107690312843910 0.00030121377997263
```

In [ ]:

```

```