In [1]:

```
from selenium import webdriver
```

In [137]:

```
driver = webdriver.Chrome()
url = "https://nid.naver.com/nidlogin.login?url=http%3A%2F%2Fmail.naver.com%2F"
driver.get(url)
```

In [138]:

```
driver.find_element_by_id("pw").clear()
driver.find_element_by_id("id").clear()
driver.find_element_by_name("id").send_keys("")
driver.find_element_by_css_selector("#pw").send_keys("")
```

In [139]:

```
driver.find_element_by_css_selector("input.btn_global").click()
```

In [140]:

```
maillist = []
for _ in driver.find_elements_by_css_selector('strong.mail_title'):
    maillist.append(_.text)
```

In [142]:

```
driver.find_element_by_xpath('//*[@id="5_fol"]/span/a[1]').click()
```

In [143]:

```
spamlist = []

for _ in driver.find_elements_by_css_selector('strong.mail_title'):
    spamlist.append(_.text)
#     with open('spammail.txt', 'wb') as f:
#         pickle.dump(spamlist, f)
# driver.find_element_by_xpath('//*[@id="next_page"]').click()
# driver.implicitly_wait(2)
```

In [81]:

```
# with open('spammail.txt', 'rb') as f:
#     data = pickle.load(f)
```

In [190]:

```
#Naive bayes classifier
trainingData = spamlist + maillist

testingData = "축하드려요 소울아크 공식 카페 카페에 가입 되셨습니다."
#testingData = (5, "Chinese Tokyo Japan")
```

In [191]:

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

N = len(trainingData)

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

In [192]:

```
from collections import defaultdict

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

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

In [193]:

```
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 [194]:

```
countSum, _countSum
```

Out[194]:

```
(97, 83)
```

In [195]:

```
condProbC, _condProbC
```

In [196]:

```
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.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 -34.792759767899184 7.757046995902133e-16
```

In [197]:

```
PriorC, _PriorC
```

Out[197]:

```
(0.5, 0.5)
```

In [198]:

```
driver.close()
```