In [1]:

```
documents = [
    ["Hadoop", "Big Data", "HBase", "Java", "Spark", "Storm", "Cassandra"],
    ["NoSQL", "MongoDB", "Cassandra", "HBase", "Postgres"],
    ["Python", "scikit-learn", "scipy", "numpy", "statsmodels", "pandas"],
    ["R", "Python", "statistics", "regression", "probability"],
    ["machine learning", "regression", "decision trees", "libsvm"],
    ["Python", "R", "Java", "C++", "Haskell", "programming languages"],
    ["statistics", "probability", "mathematics", "theory"],
    ["machine learning", "scikit-learn", "Mahout", "neural networks"],
    ["neural networks", "deep learning", "Big Data", "artificial intelligence"],
    ["Hadoop", "Java", "MapReduce", "Big Data"],
    ["statistics", "R", "statsmodels"],
    ["C++", "deep learning", "artificial intelligence", "probability"],
    ["pandas", "R", "Python"],
    ["databases", "HBase", "Postgres", "MySQL", "MongoDB"],
    ["libsvm", "regression", "support vector machines"]
]
```

In [14]:

```
type(documents)
```

Out[14]:

```
list
```

In [2]:

```
import random

a = 0.1
b = 0.1
K = 3 #군집으로 만들..

docTermTopicMat = list()
Vocabulary = list()
random.seed(0)

for d in documents:
    termTopic = list()
    for t in d:
        termTopic.append([t.lower(), random.randrange(K)])
        Vocabulary.append(t.lower())
    docTermTopicMat.append(termTopic)
Vocabulary = list(set(Vocabulary))

M = len(docTermTopicMat)
N = len(Vocabulary)
```

In [3]:

```
print(docTermTopicMat[0])
len(Vocabulary)
```

```
[['hadoop', 1], ['big data', 1], ['hbase', 0], ['java', 1], ['spark', 2], ['stor
m', 1], ['cassandra', 1]]
```

Out[3]:

```
36
```

In [4]:

```
from collections import defaultdict

topicTermMatrix = defaultdict(lambda:defaultdict(int))
docTopicMatrix = defaultdict(lambda:defaultdict(int))
for i, termTopic in enumerate(docTermTopicMat):
    for row in termTopic:
        #분자

        #분모 k번째 토픽에서, r번째 고유어가 몇 번
        topicTermMatrix[row[1]][row[0]] += 1
        #docTopicMatrix[m번째문서][k번째토픽] = 몇 개의 단어
        #(2)i번째 문서에서, k번재 토픽에 몇 개의 단어
        #[r번째 단어][k번째 토픽] = 몇 번
        docTopicMatrix[i][row[1]] += 1

        '''
        [r번째 단어][k번째 토픽] = 몇 번
        row[0] => 단어
        row[1] => 토픽
       theta = 문서의 토픽 분포
        '''
'''
1. 문서 X , 단어 <- topic에 몇 번?
2. 특정 단어 K번째 <- 몇 번?
3. 문서 m, k번째 토픽...
'''
```

Out[4]:

```
'₩n1. 문서 X , 단어 <- topic에 몇 번?₩n2. 특정 단어 K번째 <- 몇 번?₩n3. 문서 m, k
번째 토픽...₩n'
```

In [5]:

```
def topicLikelihood(k,l):
    return (topicTermMatrix[k][l] + b) / ₩
          (sum(topicTermMatrix[k].values()) + (b*N))
```

In [6]:

```
def docLikelihood(m, k):
    return (docTopicMatrix[m][k] + a)
```

In [7]:

```
def topicAssign(m, l):
    totalCount = sum([f for k in range(K) for t,f in topicTermMatrix[k].items() if t == 1])
    probList = list()
    for k in range(K):
        probList.append(topicLikelihood(k, l) * docLikelihood(m, k))

    _sum = sum(probList) * random.random()
    for i,p in enumerate(probList):
        _sum -= p
        if _sum <= 0:
            k = i
            break
    #(1) X (2)
    return k
```

In [8]:

```
_iter = 100

for _ in range(_iter):
    for i, termTopic in enumerate(docTermTopicMat):
        for row in termTopic:
            topicTermMatrix[row[1]][row[0]] -= 1
            docTopicMatrix[i][row[1]] -= 1

            k = topicAssign(i, row[0])

            row[1] = k
            topicTermMatrix[row[1]][row[0]] += 1
            docTopicMatrix[i][row[1]] += 1
#             row[1] <= Topic Assign
#             row[0], row[1]
#     break
```

In [9]:

```
print(sum(docTopicMatrix[0]))
docTopicMatrix[0]
```

```
3
```

Out[9]:

```
defaultdict(int, {1: 1, 0: 6, 2: 0})
```

In [10]:

```
for k,termList in  topicTermMatrix.items():
    print(k, "번째 토픽")
    print(sorted(termList.items(), key=lambda x:x[1], reverse=True)[:4])
```

```
1 번째 토픽
[('machine learning', 2), ('deep learning', 2), ('artificial intelligence', 2),
('regression', 2)]
0 번째 토픽
[('hbase', 3), ('postgres', 2), ('big data', 2), ('hadoop', 2)]
2 번째 토픽
[('python', 4), ('r', 4), ('statistics', 3), ('pandas', 2)]
```

In [11]:

```
docTopicMatrix
```

Out[11]:

```
defaultdict(<function __main__.<lambda>()>,
            {0: defaultdict(int, {1: 1, 0: 6, 2: 0}),
             1: defaultdict(int, {1: 0, 2: 0, 0: 5}),
             2: defaultdict(int, {2: 6, 0: 0, 1: 0}),
             3: defaultdict(int, {1: 0, 2: 5, 0: 0}),
             4: defaultdict(int, {1: 4, 0: 0, 2: 0}),
             5: defaultdict(int, {2: 5, 1: 0, 0: 1}),
             6: defaultdict(int, {1: 0, 2: 2, 0: 2}),
             7: defaultdict(int, {0: 0, 2: 2, 1: 2}),
             8: defaultdict(int, {2: 0, 1: 4, 0: 0}),
             9: defaultdict(int, {0: 3, 2: 1, 1: 0}),
             10: defaultdict(int, {2: 3, 0: 0, 1: 0}),
             11: defaultdict(int, {0: 0, 2: 0, 1: 4}),
             12: defaultdict(int, {0: 0, 2: 3, 1: 0}),
             13: defaultdict(int, {2: 0, 0: 5, 1: 0}),
             14: defaultdict(int, {0: 0, 2: 0, 1: 3})})
```

In [12]:

```
topicTermMatrix
```