In [ ]:
```
TextMining - Information Retrieval(IR)
Key Concepts: Information needs(Query), Document/Query Repr.(Embedding)
              Relevance?, Rank, (User relevance feedback => 개인화)
Components: Crawler(Text Handling), Indexer(색인기), Search algorithm
            -> Focused, Preprocessing, DOM
                                    -> doc1 = {token1, token2, ...}
                                    -> Space, Time Complexity
                                     (|D|,|V|) => Sparse Metrix
                                     (|Q|,|D|,|V|) => 병목
            : ? |D|*|V| => Metrix(메모리에 X) => Inverted Index(Linked)
=> Vocabulary => Independent Assum. => Bag of Words(BoW)
                 Dependency => N-gram => Improved BoW, O(V**n)
=> |D|*|V| => |V|*|D| (Transpose)
```

In [1]:
```
from konlpy.corpus import kobill
data = list()
for file in kobill.fileids():
    data.append(kobill.open(file).read().split())
```

In [3]:
```
len(data), len(set(data[0]))
```

Out[3]: (10, 797)

In [4]:
```
V = list()
for d in data:
    for t in d:
        if t not in V:
            V.append(t)
```

In [6]:
```
len(data), len(V) # 행:문서, 열:유니크한 토큰 => DTM(Document/Term Metrix)
```

Out[6]: (10, 2638)

In [14]:
```
D = dict()
V = dict()
DTM = dict()
TDM = dict()

for file in kobill.fileids():
    # 처음, len() = 0
    D[len(D)] = file # {0:'파일1'}
    i = len(D) - 1
    DTM[i] = list()

    for t in kobill.open(file).read().split():
        if t not in V.values():
            V[len(V)] = t # {0:'단어1'}
            TDM[len(V) - 1] = list()
        for k,v in V.items():
            if v == t:
                j = k

        DTM[i].append(j) # {0번째 문서:['단어1의 인덱스']}
        TDM[j].append(i) # {0번째 단어:['문서1', '...']}
```

In [15]:
```
# TDM(Term-Doc. Metrix)
# DTM: [d1:[t1, t2, ...],
#       d2:[t3, t2, ...]]
```

```
# TDM: [t1:[d1, d3, ...],
#       t2:[d2, d4, ...]]
D = list()
V = list()
DTM = list()
TDM = list()

for file in kobill.fileids():
    D.append(file)
#     i = D.index(file)
    # Preprocessing
    DTM.append(list())
    i = len(D)-1

    for t in kobill.open(file).read().split():
        if t not in V:
            V.append(t)
            TDM.append(list())

        j = V.index(t)

        DTM[i].append(j)
        TDM[j].append(i)


# {k:v}, index:pointer(주소를 가리키는 것)
# D = [[t1,t2,...],[t2,t3,...]]
#      0:[t1,t2,...]
#      1:[t2,t3,...]
# DTM = {0:..., 1:....}
#       [[0,4,2,..], [2,3,5...]]
```

In [16]:
```
D[0], len(DTM[0]), len([V[j] for j in DTM[0]])
```

Out[16]:
```
('1809896.txt', 1939, 1939)
```

In [21]:
```
Q = '의 안 번 호'

# q:'의', '안', '번', '호'
# V=['...', '의'] 이때의 index
result = list()
for q in [V.index(q) for q in Q.split()]: # |Q|
    for termList in DTM: # |D|
        i = DTM.index(termList)
        for j in termList: # |V|
            if q == j: # else: 인 겨우 대다수(sparse)
                result.append(i)
[D[i] for i in list(set(result))]
```

Out[21]:
```
['1809896.txt',
 '1809897.txt',
 '1809895.txt',
 '1809894.txt',
 '1809890.txt',
 '1809891.txt',
 '1809893.txt',
 '1809892.txt',
 '1809899.txt',
 '1809898.txt']
```

In [22]:
```
result = list()
for q in [V.index(q) for q in Q.split()]: # |Q|
    for i in TDM[q]:
        result.append(i)
list(set(result))
```

Out[22]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [ ]:
```
# 빈도
# TDM => list(index:termIndex -> list((docIndex, freq), ...))
Dictionary       Posting
t:0pointer      fp:0번째위치:(문서, 빈도)
```

In [ ]:
```
# C, C++(객체지향), Java(GC), Python()?
header => Stdlib
typedef struct {      Class
    (public private protected) int node;          property
    void* next;
    int (*init)(this;self);
    int (*append)(int); # 함수형 포인터   method
} list;

list LinkedList;
LinkedList.node ->
LinkedList->node

LinkedList.append = 이름
LinkedList.init = init
function init(void* self, int) {
    self->nex
}
function 이름(int) {

}
```

In [37]:
```
D = list() # [0:'doc1', 1:'doc2', 2:'doc3', ...]
V = list() # [0:'term1', 1:'term2', 2:'term3', ...]
TDM = dict() # {0:0, 1:10, ...}
Posting = list() # [0:(0,5,다음주소:1), 1:(1,10,다음주소:-1)]
# TDM[0] = 0
# Posting[0] = (0,5,1->pointer)
# Posting[1] = (1,10,-1)

for file in kobill.fileids():
    D.append(file)
    DTM.append(list())
    i = len(D)-1

    for t in kobill.open(file).read().split():
        if t not in V:
            V.append(t)
            TDM[len(V)-1] = -1 # {TDM[0] = -1}

        j = V.index(t) # j=0

        if TDM[j] == -1: # TDM[0] == -1
            Posting.append([i,1,-1]) # Posting[0] = [i,1,-1]
            TDM[j] = len(Posting)-1 # {TDM[0] = 0}
        else:
            info = Posting[TDM[j]]
```

```
            if i == info[0]: # 현재문서i, Posting 마지막정보의 문서i
                # Posting[0] = [i,Posting[0][1]+1,-1]
                Posting[TDM[j]][1] = info[1]+1
            else: #
                Posting.append([i,1,TDM[j]])
                TDM[j] = len(Posting)-1
```

In [48]:
```
i = 1
V[i], TDM[i]
```

Out[48]: ('일부개정법률안', 3864)

In [52]:
```
p = Posting[TDM[i]]
while True:
    print(p)

    if p[-1] == -1:
        break

    p = Posting[p[-1]]
```

```
[8, 2, 3250]
[7, 3, 2706]
[6, 3, 2163]
[5, 3, 1622]
[4, 3, 1451]
[3, 2, 1198]
[2, 2, 1]
[0, 2, -1]
```

In [53]:
```
D = list()
V = list()
TDM = dict()

for file in kobill.fileids():
    D.append(file)
    i = len(D)-1

    # Local
    Posting = dict()

    for t in kobill.open(file).read().split():
        if t not in V:
            V.append(t)
            TDM[len(V)-1] = list()

        j = V.index(t)

        # Posting[단어의index] = (문서의index, 빈도)
        if j not in Posting:
            Posting[j] = (i,1)
        else:
            Posting[j] = (i,Posting[j][-1]+1)

    # Global Update
    for k,v in Posting.items():
        TDM[k].append(v)
        # TDM[단어의index] = [(문서의index, 빈도), (문서의index, 빈도), ..]
```

In [57]:
```
V[1], len(TDM[1]), TDM[1]
```

Out[57]:
```
('일부개정법률안',
 8,
 [(0, 2), (2, 2), (3, 2), (4, 3), (5, 3), (6, 3), (7, 3), (8, 2)])
```

In [68]:
```
result = dict()
for q in [V.index(q) for q in Q.split()]: # |Q|
    for info in TDM[q]:
        if info[0] not in result:
            result[info[0]] = info[1]
        else:
            result[info[0]] += info[1]
sorted(result, key=result.get, reverse=True)
result
```

Out[68]:
```
{0: 15, 1: 1, 2: 9, 3: 7, 4: 9, 5: 8, 6: 9, 7: 8, 8: 18, 9: 1}
```

In [70]:
```
from struct import pack, unpack
```

In [71]:
```
pack('ii', 1,2)
```

Out[71]:
```
b'\x01\x00\x00\x00\x02\x00\x00\x00'
```

In [72]:
```
unpack('ii', pack('ii', 1,2))
```

Out[72]:
```
(1, 2)
```

In [ ]:
```
TDM[obama] = Positing[위치] ...
AND(intersection)
[healthcare] = Posting[위치] ...
All=D   , 부분집합={D|obama∈d}
```

In [75]:
```
for D, for T => M*N, T

DTM[1][0], V[797]
Ti=2,j=1 = w797∈V
```

Out[75]:
```
(797, '국군부대의')
```