1.Indexer Crawler (Focused) => Repository(collection) Document Analyzer => HTML, Tokenizing, Normalizing Stemming(BPE), N-gram, MA, POS Stopwords, RE, Phrases => Preprocessing Features => Lexicon(1) Document(Query) Representation => BoW Document-Term Mat., Term-Document Mat.(핵심) => Inverted Document Indexing(역문헌구조) 2.Relevance Weighting(TF-IDF), Similarity(Euclidean, Cosine:0-1) Sorting 3.Results 끝 (Top k)

TF-IDF

In [4]:

```
from collections import defaultdict
#from konlpy import kobill

collection = [
    ("Document1", "This is a a a a a a a sample."),
    ("Document2", "This is another sample."),
    ("Document3", "This is not a sample.")
]

DTM = defaultdict(lambda:defaultdict(int))
for docName, docContent in collection:
    for term in docContent.lower().split():
        DTM[docName][term] += 1

TDM = defaultdict(lambda:defaultdict(int))
for idx, termList in DTM.items():
    for term, freq in termList.items():
        TDM[term][idx] = freq
```

In [4]:

```
# N = len(DTM), |V| = len(TDM), DTM.keys(), V = TDM.keys()
# maxTF(in Document1) = max(DTM["Document1"].values())
```

Out[4]:

```
(3,
 6,
 dict_keys(['Document1', 'Document2', 'Document3']),
 dict_keys(['this', 'is', 'a', 'sample.', 'another', 'not']))
```

In [7]:

```

```

Out[7]:

```
7
```

In [5]:

```
from math import log2

TWM = defaultdict(lambda:defaultdict(float))
N = len(DTM)
for idx, termList in DTM.items():
    maxTF = max(termList.values()) #단어 빈도
    for term, freq in termList.items():
        TF = freq / maxTF
        IDF = log2(N/len(TDM[term]))
        TWM[term][idx] = TF*IDF
TWM
```

Out[5]:

```
defaultdict(<function __main__.<lambda>()>,
            {'this': defaultdict(float,
                         {'Document1': 0.0,
                          'Document2': 0.0,
                          'Document3': 0.0}),
             'is': defaultdict(float,
                         {'Document1': 0.0,
                          'Document2': 0.0,
                          'Document3': 0.0}),
             'a': defaultdict(float,
                         {'Document1': 0.5849625007211562,
                          'Document3': 0.5849625007211562}),
             'sample.': defaultdict(float,
                         {'Document1': 0.0,
                          'Document2': 0.0,
                          'Document3': 0.0}),
             'another': defaultdict(float, {'Document2': 1.584962500721156}),
             'not': defaultdict(float, {'Document3': 1.584962500721156})})
```

TF <- normalization(max) 해서 세 번째 'a' 결과 보면 document1과 document3의 a 가중치가 동일 (첫 번째는 7개 세 번째는 1개)

흔할수록 결과값이 낮아짐..

In [7]:

```
from konlpy.corpus import kobill
from konlpy.tag import Komoran

ma = Komoran()

DTM = defaultdict(lambda:defaultdict(int))
for idx in kobill.fileids():
#     ma.pos() => (형태소, 품사)
#     ma.morphs() => 형태소, 형태소, ...
#     ma.nouns() => 명사, 명사 <- 보통의 경우 제일 나은 편
    for term in kobill.open(idx).read().split():
        DTM[idx][term] += 1

TDM = defaultdict(lambda:defaultdict(int))
for idx, termList in DTM.items():
    for term, freq in termList.items():
        TDM[term][idx] = freq

TWM = defaultdict(lambda:defaultdict(float))
N = len(DTM)
for idx, termList in DTM.items():
    maxTF = max(termList.values()) #단어 빈도
    for term, freq in termList.items():
        TF = freq / maxTF
        IDF = log2(N/len(TDM[term]))
        TWM[term][idx] = TF*IDF
TWM
```

```
                    '1809899.txt': 0.07511655547371789}),
            '9890': defaultdict(float, {'1809890.txt': 0.17483832078354536}),
            '발의연월일': defaultdict(float,
                        {'1809890.txt': 0.016943583941440122,
                         '1809891.txt': 0.016943583941440122,
                         '1809892.txt': 0.015329909280350587,
                         '1809893.txt': 0.016943583941440122,
                         '1809894.txt': 0.05365468248122705,
                         '1809895.txt': 0.026827341240613527,
                         '1809896.txt': 0.010730936496245411,
                         '1809899.txt': 0.010730936496245411}),
            ':': defaultdict(float,
                        {'1809890.txt': 0.0,
                         '1809891.txt': 0.0,
                         '1809892.txt': 0.0,
                         '1809893.txt': 0.0,
                         '1809894.txt': 0.0,
                         '1809895.txt': 0.0,
                         '1809896.txt': 0.0,
```

In [8]:

```
from konlpy.corpus import kobill
from konlpy.tag import Komoran

ma = Komoran()

DTM = defaultdict(lambda:defaultdict(int))
for idx in kobill.fileids():
#     ma.pos() => (형태소, 품사)
#     ma.morphs() => 형태소, 형태소, ...
#     ma.nouns() => 명사, 명사 <- 보통의 경우 제일 나은 편
    for term in kobill.open(idx).read().split():
        for token in ma.pos(term):
            DTM[idx]["/".join(token)] += 1

TDM = defaultdict(lambda:defaultdict(int))
for idx, termList in DTM.items():
    for term, freq in termList.items():
        TDM[term][idx] = freq

TWM = defaultdict(lambda:defaultdict(float))
N = len(DTM)
for idx, termList in DTM.items():
    maxTF = max(termList.values()) #단어 빈도
    for term, freq in termList.items():
        TF = freq / maxTF
        IDF = log2(N/len(TDM[term]))
        TWM[term][idx] = TF*IDF
TWM
```

Out[8]:

```
defaultdict(<function __main__.<lambda>()>,
            {'지방/NNG': defaultdict(float,
                        {'1809890.txt': 0.0680275933076498,
                         '1809891.txt': 0.02267586443588327,
                         '1809892.txt': 0.017546799861100148,
                         '1809893.txt': 0.023395733148133528,
                         '1809897.txt': 0.011515087408846972,
                         '1809898.txt': 0.010379797100932482}),
             '공무원법/NNP': defaultdict(float,
                        {'1809890.txt': 0.12202413183575654,
                         '1809891.txt': 0.12202413183575654,
                         '1809892.txt': 0.0944234353490973,
                         '1809893.txt': 0.04196597126626547}),
             '일부/NNG': defaultdict(float,
                        {'1809890.txt': 0.019810959685376144,
                         '1809891.txt': 0.019810959685376144,
                         '1809892.txt': 0.015329909280350587,
                         '1809893.txt': 0.020439879040467 45,
```

Similarity

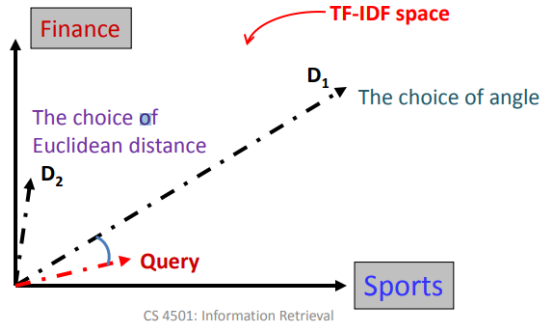
In [9]:

```
Site: http://www.cs.virginia.edu/~hw5x/Course/IR2015/_site/docs/PDFs/Boolean&VS%20model.pdf
```

Out[9]:

From distance to angle

- Angle: how vectors are overlapped
  - Cosine similarity – projection of one vector onto another



In [10]:

```

```

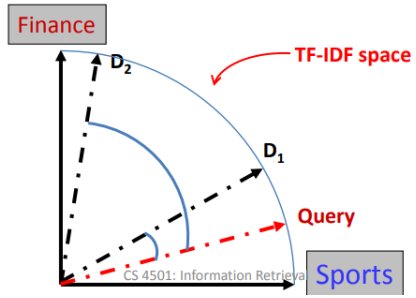
Out[10]:

Cosine similarity

- Angle between two vectors
  - $cosine(V_q, V_d) = \frac{V_q \times V_d}{|V_q|_2 \times |V_d|_2} = \frac{V_q}{|V_q|_2} \times \frac{V_d}{|V_d|_2}$
  - Document length normalized

TF-IDF vector

Unit vector



IDF - 빈도가 낮을수록 높은 가중치 TF - 빈도가 높을수록 높은 가중치 섞어서 사용해 TF-IDF Done

In [12]:

```
from konlpy.corpus import kobill
from konlpy.tag import Komoran
from math import sqrt

ma = Komoran()

DTM = defaultdict(lambda:defaultdict(int))
for idx in kobill.fileids():
#     ma.pos() => (형태소, 품사)
#     ma.morphs() => 형태소, 형태소, ...
#     ma.nouns() => 명사, 명사 <- 보통의 경우 제일 나은 편
    for term in kobill.open(idx).read().split():
        for token in ma.pos(term):
            DTM[idx]["/".join(token)] += 1

TDM = defaultdict(lambda:defaultdict(int))
for idx, termList in DTM.items():
    for term, freq in termList.items():
        TDM[term][idx] = freq

TWM = defaultdict(lambda:defaultdict(float))
DVL = defaultdict(float)
N = len(DTM)
for idx, termList in DTM.items():
    maxTF = max(termList.values()) #단어 빈도
    for term, freq in termList.items():
        TF = freq / maxTF
        IDF = log2(N/len(TDM[term]))
        TWM[term][idx] = TF*IDF
        DVL[idx] += TWM[term][idx]**2

for idx, length in DVL.items():
    DVL[idx] = sqrt(length)

DVL #벡터의 길이
```

Out[12]:

```
defaultdict(float,
            {'1809890.txt': 1.5372595365455324,
             '1809891.txt': 1.5360087893120677,
             '1809892.txt': 1.4064494683553657,
             '1809893.txt': 1.6201854130149445,
             '1809894.txt': 3.601181759066831,
             '1809895.txt': 2.5997540664548255,
             '1809896.txt': 2.03947811568486,
             '1809897.txt': 2.959862672291415,
             '1809898.txt': 2.525695656406215,
             '1809899.txt': 2.074136209536078})
```

In [40]:

```
query = "국방의 의무와 보편적 교육에 대한 법안을 찾아주세요."
#의 와 에 때문에 문제의 소지가 있기는 함 그래도 그 셋이 W->00이 되기는 함

TQM = defaultdict(int)
QWM = defaultdict(float)

for term in query.split():
    for token in ma.pos(term):
        TQM["/".join(token)] += 1

alpha = 0.5
maxTF = max(TQM.values())
for term, freq in TQM.items():
    TF = alpha + (1-alpha)*(freq / maxTF)
    DF = len(TWM[term]) if len(TWM[term]) > 0 else 1
    IDF = log2(N/DF)
    QWM[term] = TF*IDF #IDF = N / 1

#IDF 관건
TQM
#QWM
maxTF
```

Out[40]:

```
1
```

In [41]:

```
candidateList = defaultdict(float)

for term, weight1 in QWM.items():
    for doc, weight2 in TWM[term].items():
        innerProduct = weight1 * weight2
        candidateList[doc] += innerProduct

for doc, sim in candidateList.items():
    candidateList[doc] = sim / DVL[doc]
```

In [42]:

```
candidateList
kobill.open('1809899.txt').read()
TWM["의무/NNG"]
```

Out[42]:

```
defaultdict(float, {'1809899.txt': 0.04711954744521081})
```

In [43]:

```
from nltk.tokenize import sent_tokenize

k = 5
for doc, sim in sorted(candidateList.items(), key=lambda x:x[1], reverse=True)[:k]:
    print("문서이름:{0} / 유사도:{1:.4f}".format(doc, sim))
    print(sent_tokenize(kobill.open(doc).read())[:3])
    print()
```

```
문서이름:1809899.txt / 유사도:0.0778
['결혼중개업의 관리에 관한 법률 일부개정법률안WnWn(한선교의원 대표발의 )WnWn 의 안Wn
번 호WnWn9899WnWn발의연월일 : 2010.', '11.', '15.']

문서이름:1809897.txt / 유사도:0.0377
['국군부대의 아랍에미리트(UAE)군 교육훈련 지원 등에 Wn관한 파견 동의안WnWn의안Wn
제출연월일 :  2010.', '11.', '15.']

문서이름:1809898.txt / 유사도:0.0323
['국군부대의 소말리아 해역 파견연장 동의안WnWn의안Wn
제출연월일 :  2010.', '11.', '15.']

문서이름:1809892.txt / 유사도:0.0295
['교육공무원법 일부개정법률안WnWn(정의화의원 대표발의 )WnWn 의 안Wn 번 호WnWn9892Wn
Wn발의연월일 : 2010.', '11.', '12.']

문서이름:1809891.txt / 유사도:0.0245
['국가공무원법 일부개정법률안WnWn(정의화의원 대표발의 )WnWn 의 안Wn 번 호WnWn9891Wn
Wn발의연월일 : 2010.', '11.', '12.']
```

In [47]:

```
candidateList = defaultdict(float)

for term, docList in TWM.items():
    for doc, weight1 in TWM[term].items():
        weight2 = QWM[term]
        candidateList[doc] += (weight1 - weight2)**2
        #print(term, doc, (weight1 - weight2)**2)

for doc, sim in candidateList.items():
    candidateList[doc] = sqrt(sim)
```

```
지방/NNG 1809890.txt 0.0046277534512310006
지방/NNG 1809891.txt 0.0005141948279145557
지방/NNG 1809892.txt 0.00030789018536550415
지방/NNG 1809893.txt 0.000547360329538674
지방/NNG 1809897.txt 0.00013259723803338608
지방/NNG 1809898.txt 0.00010774018785652636
공무원법/NNP 1809890.txt 0.014889888750270093
공무원법/NNP 1809891.txt 0.014889888750270093
공무원법/NNP 1809892.txt 0.008915785143125158
공무원법/NNP 1809893.txt 0.0017611427443210194
일부/NNG 1809890.txt 0.0003924741236555988 6
일부/NNG 1809891.txt 0.0003924741236555988 6
일부/NNG 1809892.txt 0.00023500611854377907
일부/NNG 1809893.txt 0.0004177886551889406
일부/NNG 1809894.txt 0.0025837653310256488
일부/NNG 1809895.txt 0.0011090835725330074
일부/NNG 1809896.txt 7.673669176939725e-05
일부/NNG 1809899.txt 0.0001876644604396678 5
개정/NNG 1809890.txt 0.0015698964946223954
```

In [45]:

```
#candidateList, W
for doc in DTM:
    print(doc, len(kobill.open(doc).read().split()), len(DTM[doc]), sum(DTM[doc].values()))
candidateList
```

```
1809890.txt 841 484 1947
1809891.txt 834 480 1941
1809892.txt 984 516 2250
1809893.txt 840 490 1942
1809894.txt 242 206 520
1809895.txt 394 272 889
1809896.txt 1939 624 4217
1809897.txt 788 409 1430
1809898.txt 821 405 1466
1809899.txt 1677 532 3899
```

Out[45]:

```
defaultdict(float,
            {'1809890.txt': 2.109480681582283,
             '1809891.txt': 2.1085693880241627,
             '1809892.txt': 2.014253641367023,
             '1809893.txt': 2.17010355889301,
             '1809897.txt': 3.787292042651278,
             '1809898.txt': 3.452282032674107,
             '1809894.txt': 3.614033553900103,
             '1809895.txt': 2.606487478910146,
             '1809896.txt': 2.134677697885167,
             '1809899.txt': 3.8941271751908073})
```

In [46]:

```
k = 5
for doc, sim in sorted(candidateList.items(), key=lambda x:x[1], reverse=True)[:k]:
    print("문서이름:{0} / 거리:{1:.4f}".format(doc, sim))
    print(sent_tokenize(kobill.open(doc).read())[:3])
    print()
```

```
문서이름:1809899.txt / 거리:3.8941
['결혼중개업의 관리에 관한 법률 일부개정법률안WnWn(한선교의원 대표발의 )WnWn 의 안Wn
번 호WnWn9899WnWn발의연월일 : 2010.', '11.', '15.']

문서이름:1809897.txt / 거리:3.7873
['국군부대의 아랍에미리트(UAE)군 교육훈련 지원 등에 Wn관한 파견 동의안WnWn의안Wn
제출연월일 :  2010.', '11.', '15.']

문서이름:1809894.txt / 거리:3.6140
['고등교육법 일부개정법률안WnWn(안상수의원 대표발의 )WnWn 의 안Wn 번 호WnWn9894WnWn
발의연월일 : 2010.', '11.', '15.']

문서이름:1809898.txt / 거리:3.4523
['국군부대의 소말리아 해역 파견연장 동의안WnWn의안Wn
제출연월일 :  2010.', '11.', '15.']

문서이름:1809895.txt / 거리:2.6065
['하도급거래 공정화에 관한 법률 일부개정법률안WnWn(유선호의원 대표발의 )WnWn 의 안Wn
번 호WnWn9895WnWn발의연월일 : 2010.', '11.', '15.']
```

In [ ]: