DB Homework

개발환경

Window 10 Education

Intel Core i3-4150 CPU 3.50GHz

RAM 8.00GM 64 bit

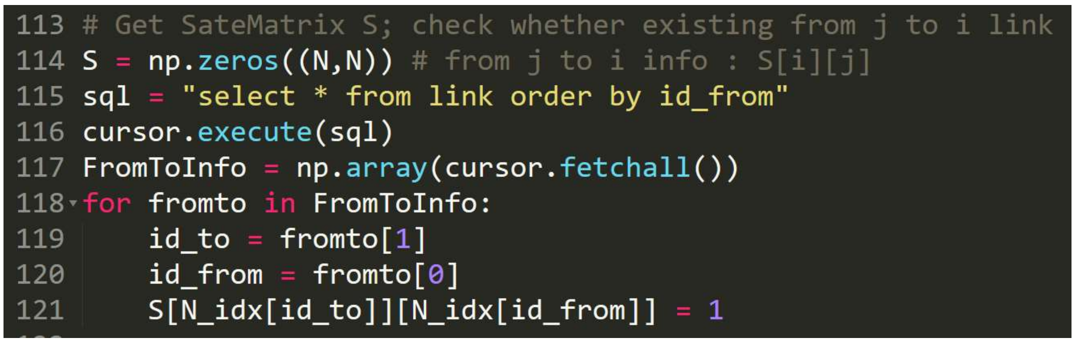
Ubuntu 16.04 (using vmware)

구현 기능 소개(간략하게)

1. Connect to mysql
2. Create inverted index table
3. PageRank algorithm에 의한 모든 document id에 대해 PageRank score 계산
4. 입력 받은 query를 여러 term으로 나누고, 각 term 을 포함하는 document id에 대해서 TFIDF score를 계산한다. 그 이후, 각 term을 적어도 한 개라도 포함하는 document id 리스트에 대해서 union 된 TFIDF score 를 구한다.
5. TFIDF score와 PageRank score의 곱에 대해 document id list 를 sorting 하여 원하는 결과(id, title, TFIDF score, PageRank score)를 출력한다.

PageRank 구현 코드 설명

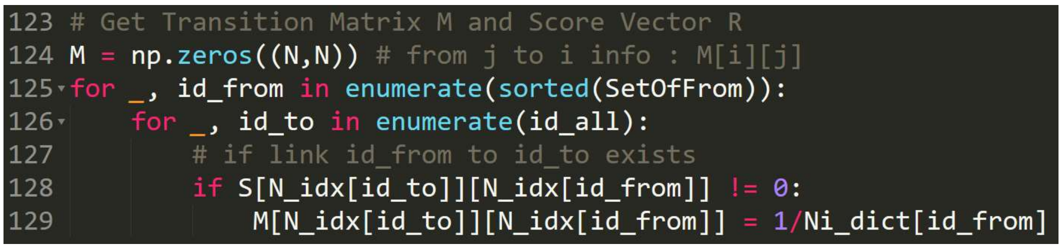
1. PageRank 계산방식
   A. PageRank iteration 공식 이용
      i. Notation
         1. N: The number of all Document ids
         2. $\mathbb{R}$: PageRank score vector $N \times 1$
         3. $\mathbb{M}$: Transition Matrix $N \times N$
         4. δ: Damping factor
         5. $\mathbb{I}$: One vector $N \times 1$
         6. N_i: The number of outgoing hyperlink from document_i
      ii. Formula
         1. $\mathbb{R}_{t+1} = \delta \mathbb{M} \mathbb{R}_t + \frac{1-\delta}{N} \mathbb{I}$
         2. $\mathbb{M} \ni m_{ij} = \begin{cases} \frac{1}{N_j}, & if\ j\ links\ to\ i \\ 0, & ow \end{cases}$
2. 코드 설명
   A. sql을 사용하여 N_i & N 을 구한다.
   B. link 여부를 확인하기 위한 State Matrix $\mathbb{S}$를 구축한다.
      i. $\mathbb{S} \ni s_{ij} = \begin{cases} 1, & if\ document_j\ links\ to\ document_i \\ 0, & ow \end{cases}$



```python
# Get SateMatrix S; check whether existing from j to i link
S = np.zeros((N,N)) # from j to i info : S[i][j]
sql = "select * from link order by id_from"
cursor.execute(sql)
FromToInfo = np.array(cursor.fetchall())
for fromto in FromToInfo:
    id_to = fromto[1]
    id_from = fromto[0]
    S[N_idx[id_to]][N_idx[id_from]] = 1
```

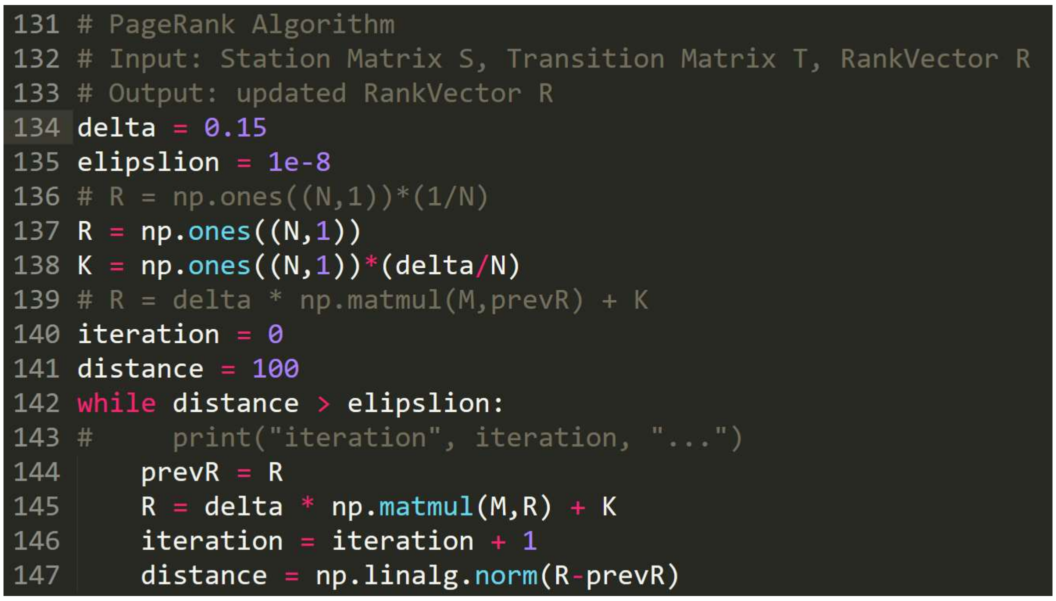
C. Transition Matrix $\mathbb{M}$를 구축한다.

i. $$\mathbb{M} \ni \mathrm{m}_{\mathrm{ij}} = \begin{cases} \frac{1}{N_j}, & if\ \ j\ \ links\ \ to\ \ i \\ 0, & ow \end{cases}$$



```
# Get Transition Matrix M and Score Vector R
M = np.zeros((N,N)) # from j to i info : M[i][j]
for _, id_from in enumerate(sorted(SetOfFrom)):
    for _, id_to in enumerate(id_all):
        # if link id_from to id_to exists
        if S[N_idx[id_to]][N_idx[id_from]] != 0:
            M[N_idx[id_to]][N_idx[id_from]] = 1/Ni_dict[id_from]
```

D. PageRank Algorithm iteration반복

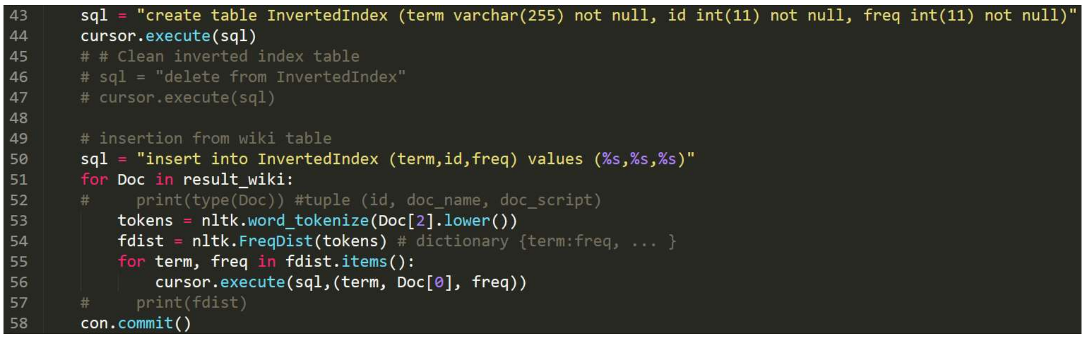


```
# PageRank Algorithm
# Input: Station Matrix S, Transition Matrix T, RankVector R
# Output: updated RankVector R
delta = 0.15
elipslion = 1e-8
# R = np.ones((N,1))*(1/N)
R = np.ones((N,1))
K = np.ones((N,1))*(delta/N)
# R = delta * np.matmul(M,prevR) + K
iteration = 0
distance = 100
while distance > elipslion:
#     print("iteration", iteration, "...")
    prevR = R
    R = delta * np.matmul(M,R) + K
    iteration = iteration + 1
    distance = np.linalg.norm(R-prevR)
```

```
iteration 0 ...
distance =  74.10893977199761
iteration 1 ...
distance =  19.654939936357895
iteration 2 ...
distance =  1.317147357409797
iteration 3 ...
distance =  0.16673183038197417
iteration 4 ...
distance =  0.023230458923275472
iteration 5 ...
distance =  0.00363864354638029
iteration 6 ...
distance =  0.0005172302450978651
iteration 7 ...
distance =  8.159564757324994e-05
iteration 8 ...
distance =  1.1621562302864015e-05
iteration 9 ...
distance =  1.8351128469182678e-06
iteration 10 ...
distance =  2.615057805869916e-07
iteration 11 ...
distance =  4.129248301639776e-08
iteration 12 ...
distance =  5.885459257455751e-09
```

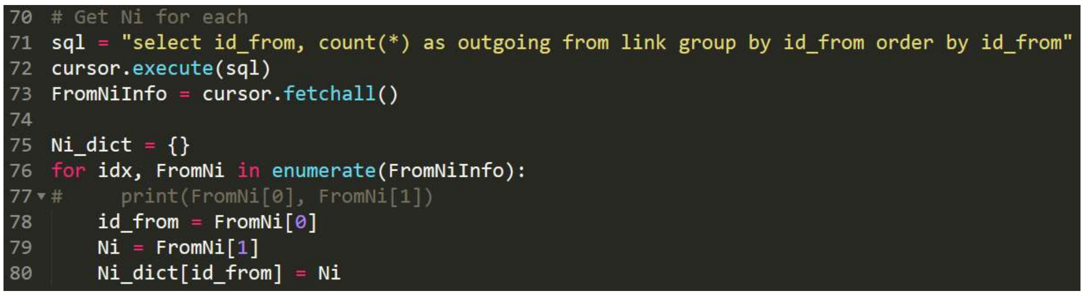
주요 SQL문 설명

1. Inverted index table 생성



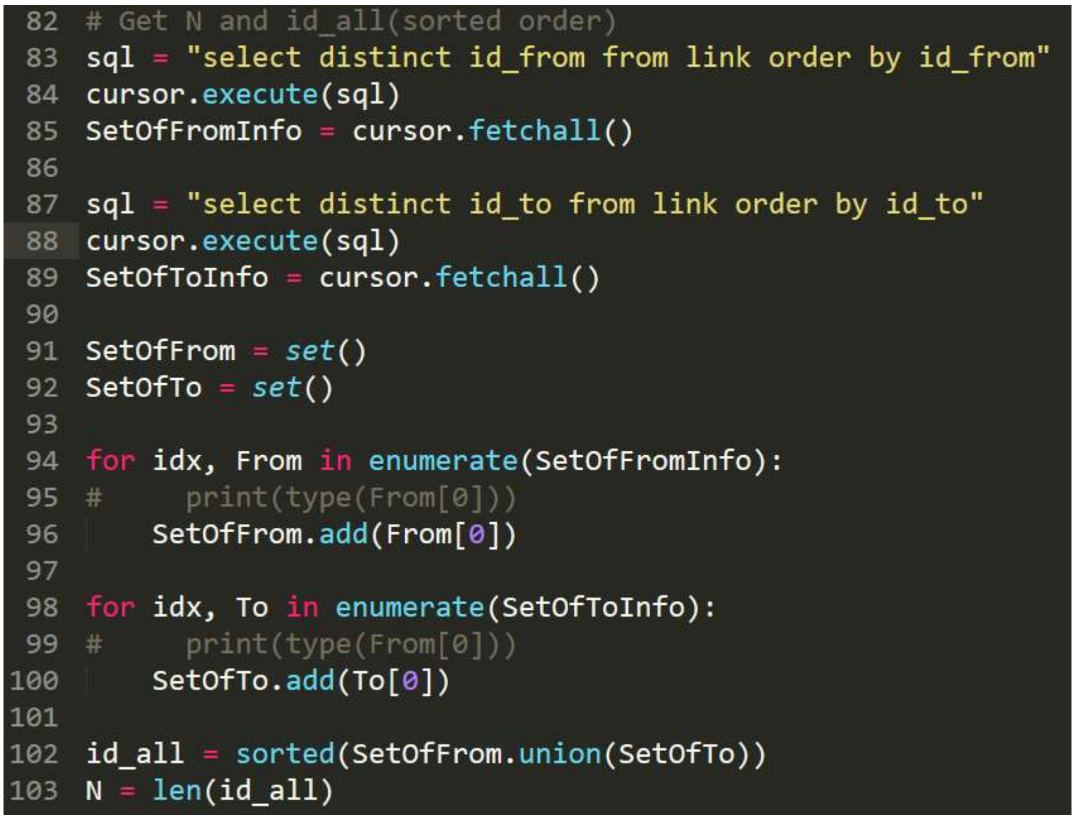
```
    sql = "create table InvertedIndex (term varchar(255) not null, id int(11) not null, freq int(11) not null)"
    cursor.execute(sql)
    # # Clean inverted index table
    # sql = "delete from InvertedIndex"
    # cursor.execute(sql)

    # insertion from wiki table
    sql = "insert into InvertedIndex (term,id,freq) values (%s,%s,%s)"
    for Doc in result_wiki:
    #     print(type(Doc)) #tuple (id, doc_name, doc_script)
        tokens = nltk.word_tokenize(Doc[2].lower())
        fdist = nltk.FreqDist(tokens) # dictionary {term:freq, ... }
        for term, freq in fdist.items():
            cursor.execute(sql,(term, Doc[0], freq))
    #     print(fdist)
    con.commit()
```

2. Document id 별로 N_i값을 구함



```
# Get Ni for each
sql = "select id_from, count(*) as outgoing from link group by id_from order by id_from"
cursor.execute(sql)
FromNiInfo = cursor.fetchall()

Ni_dict = {}
for idx, FromNi in enumerate(FromNiInfo):
#     print(FromNi[0], FromNi[1])
    id_from = FromNi[0]
    Ni = FromNi[1]
    Ni_dict[id_from] = Ni
```

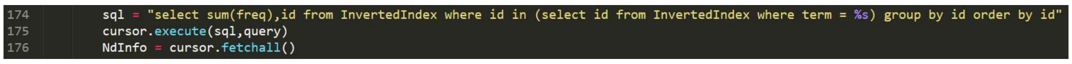
3. link table에 존재하는 distinct한 id set 인 id_all과 그 숫자 N을 구함



```
# Get N and id_all(sorted order)
sql = "select distinct id_from from link order by id_from"
cursor.execute(sql)
SetOfFromInfo = cursor.fetchall()

sql = "select distinct id_to from link order by id_to"
cursor.execute(sql)
SetOfToInfo = cursor.fetchall()

SetOfFrom = set()
SetOfTo = set()

for idx, From in enumerate(SetOfFromInfo):
#     print(type(From[0]))
    SetOfFrom.add(From[0])

for idx, To in enumerate(SetOfToInfo):
#     print(type(From[0]))
    SetOfTo.add(To[0])

id_all = sorted(SetOfFrom.union(SetOfTo))
N = len(id_all)
```

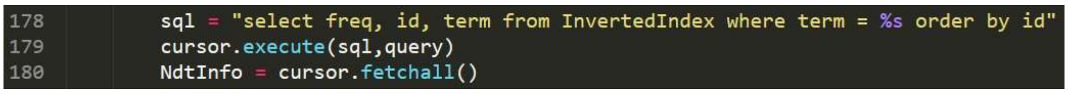
4. 주어진 term이 존재하는 문서 id들 중에서, id별 모든 term frequency값의 합을 출력함

   (N_d for each id 를 구함)



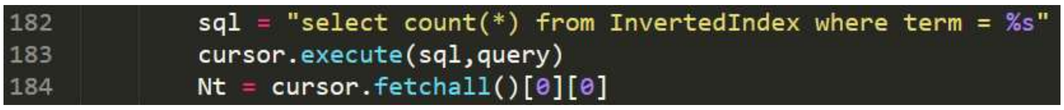
```
        sql = "select sum(freq),id from InvertedIndex where id in (select id from InvertedIndex where term = %s) group by id order by id"
        cursor.execute(sql,query)
        NdInfo = cursor.fetchall()
```

5. 각 문서 id별 주어진 term의 frequency값을 출력함($N_{d,t}$ for each id 를 구함)



```
        sql = "select freq, id, term from InvertedIndex where term = %s order by id"
        cursor.execute(sql,query)
        NdtInfo = cursor.fetchall()
```

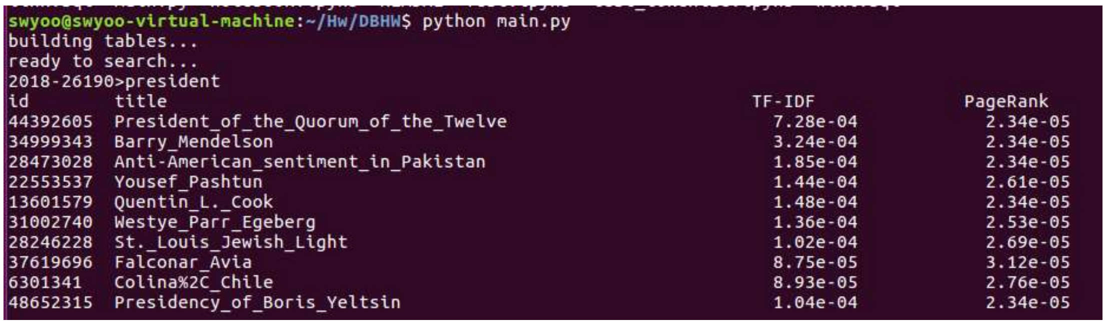
6. 주어진 term을 가진 document id들의 수를 구함(N_t 구함)



```
        sql = "select count(*) from InvertedIndex where term = %s"
        cursor.execute(sql,query)
        Nt = cursor.fetchall()[0][0]
```

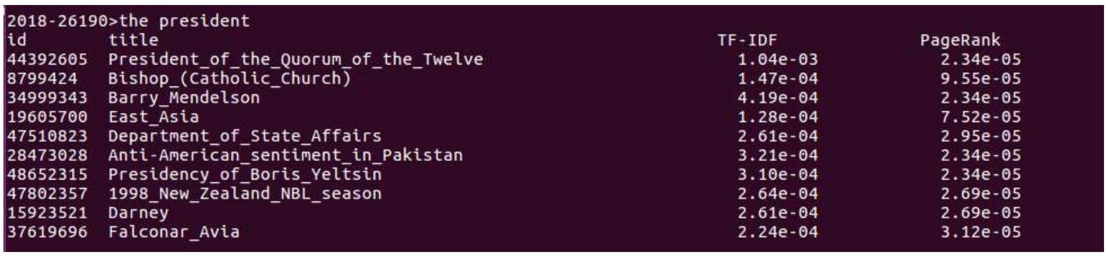
프로그램 실행 예시

1. 단일 단어 검색



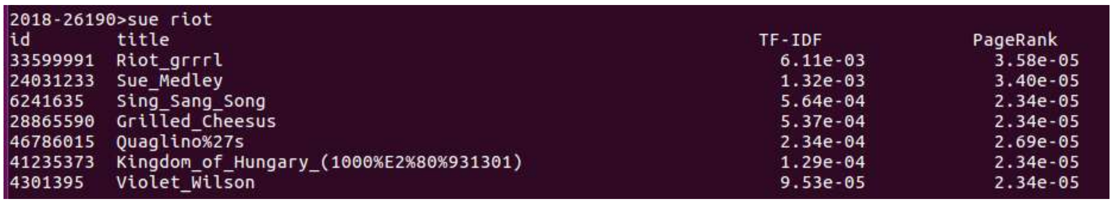
```
swyoo@swyoo-virtual-machine:~/Hw/DBHW$ python main.py
building tables...
ready to search...
2018-26190>president
id        title                                                        TF-IDF           PageRank
44392605  President_of_the_Quorum_of_the_Twelve                          7.28e-04          2.34e-05
34999343  Barry_Mendelson                                                3.24e-04          2.34e-05
28473028  Anti-American_sentiment_in_Pakistan                            1.85e-04          2.34e-05
22553537  Yousef_Pashtun                                                 1.44e-04          2.61e-05
13601579  Quentin_L._Cook                                                1.48e-04          2.34e-05
31002740  Westye_Parr_Egeberg                                            1.36e-04          2.53e-05
28246228  St._Louis_Jewish_Light                                         1.02e-04          2.69e-05
37619696  Falconar_Avia                                                  8.75e-05          3.12e-05
6301341   Colina%2C_Chile                                                8.93e-05          2.76e-05
48652315  Presidency_of_Boris_Yeltsin                                    1.04e-04          2.34e-05
```

2. 복수 단어 검색



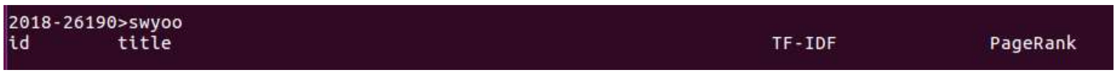
```
2018-26190>the president
id        title                                                        TF-IDF           PageRank
44392605  President_of_the_Quorum_of_the_Twelve                          1.04e-03          2.34e-05
8799424   Bishop_(Catholic_Church)                                       1.47e-04          9.55e-05
34999343  Barry_Mendelson                                                4.19e-04          2.34e-05
19605700  East_Asia                                                      1.28e-04          7.52e-05
47510823  Department_of_State_Affairs                                    2.61e-04          2.95e-05
28473028  Anti-American_sentiment_in_Pakistan                            3.21e-04          2.34e-05
48652315  Presidency_of_Boris_Yeltsin                                    3.10e-04          2.34e-05
47802357  1998_New_Zealand_NBL_season                                    2.64e-04          2.69e-05
15923521  Darney                                                         2.61e-04          2.69e-05
37619696  Falconar_Avia                                                  2.24e-04          3.12e-05
```

3. 검색 단어에 대해 10개 미만의 검색결과



```
2018-26190>sue riot
id        title                                                   TF-IDF          PageRank
33599991  Riot_grrrl                                                6.11e-03        3.58e-05
24031233  Sue_Medley                                                1.32e-03        3.40e-05
6241635   Sing_Sang_Song                                            5.64e-04        2.34e-05
28865590  Grilled_Cheesus                                           5.37e-04        2.34e-05
46786015  Quaglino%27s                                              2.34e-04        2.69e-05
41235373  Kingdom_of_Hungary_(1000%E2%80%931301)                    1.29e-04        2.34e-05
4301395   Violet_Wilson                                             9.53e-05        2.34e-05
```

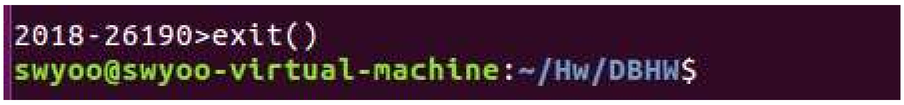
4. 검색 단어가 없는 경우



```
2018-26190>swyoo
id        title                                                   TF-IDF          PageRank
```

5. 끝내는 명령어(mysql server와 연결 해제 및 프로그램 종료) 추가

   A. exit()



```
2018-26190>exit()
swyoo@swyoo-virtual-machine:~/Hw/DBHW$
```