정보검색과
데이터마이닝

hw2 - 검색엔진구현 1차

20142772 최승호

* 기본세팅 - 디렉토리 세팅이랑 fnames 리스트 형태로 가지고 있기

```
In [1]: cd ₩Users₩User₩Desktop₩승호₩3학년 2학기(2018)₩정보검색과 데이터마이닝₩과제₩hw_2
```

```
C:₩Users₩User₩Desktop₩승호₩3학년 2학기(2018)₩정보검색과 데이터마이닝₩과제₩hw_2
```

fnames 파일 읽어서 list 형태로 가지고 있자

```
In [2]: f = open("fnames.txt", 'r')
f_list = []
while True:
    line = f.readline()
    if not line: break
    f_list.append(line[:-1])
f.close()
# f_list
```

```
In [3]: cd klt2010/EXE
```

```
C:₩Users₩User₩Desktop₩승호₩3학년 2학기(2018)₩정보검색과 데이터마이닝₩과제₩hw_2₩klt2010₩EXE
```

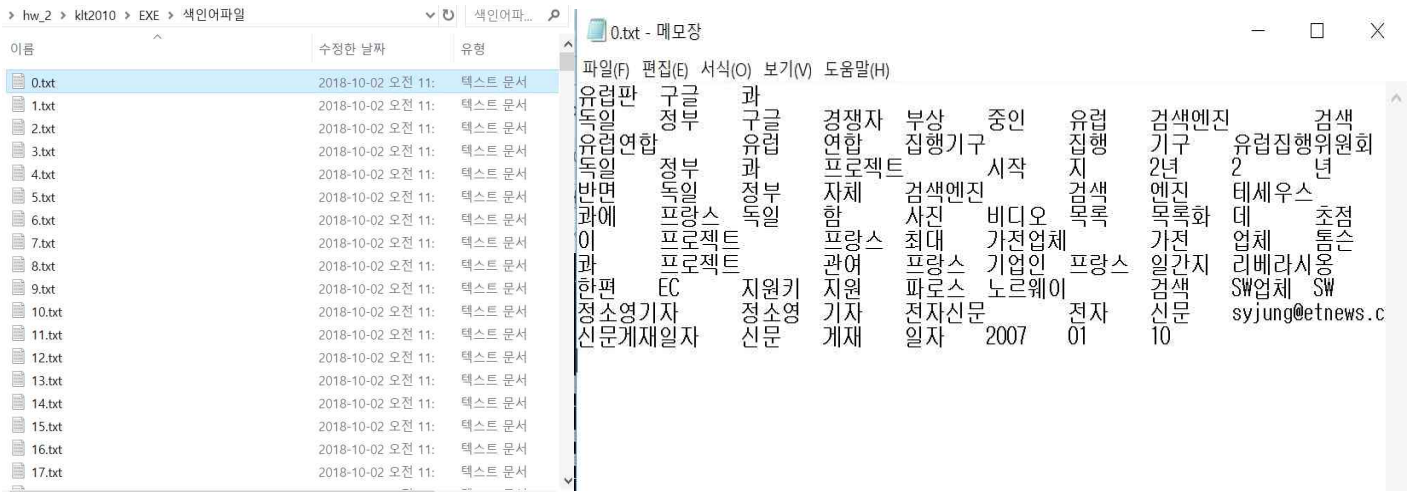
1-1 index2018.exe를 이용하여 각 텍스트 파일에 대한 인덱싱 파일을 만들자(색인어 추출)
+ 나중에 df계산을 위해 all_unique_indexing파일도 작성한다.

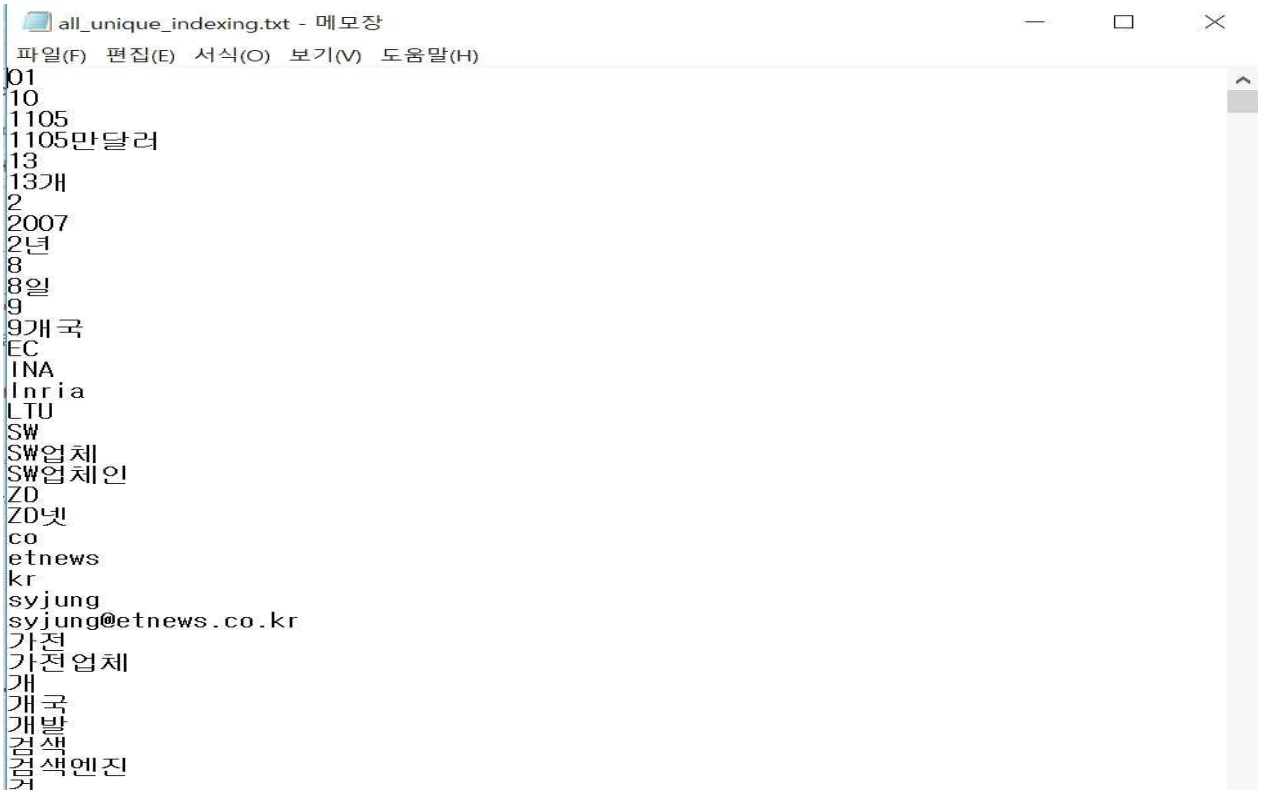
1-1) index2018.exe를 이용하여 각 텍스트 파일에 대한 인덱싱 파일을 만들자(색인어 추출)

- 추가로 각 파일당 unique한 색인어만 들어간 all_unique_indexing.txt파일 만들기

```
import subprocess
import os

# all_unique_indexing파일 존재하면 삭제
if os.path.exists("all_unique_indexing.txt"):
    os.remove("all_unique_indexing.txt")
f = open("all_unique_indexing.txt", 'a')

# 색인어파일 디렉토리 있으면 삭제
if os.path.exists("색인어파일"):
    subprocess.call("rm -rf 색인어파일")
subprocess.call("mkdir 색인어파일")

for i in range(len(f_list)):
    input_f_name = "../../textfiles/" + f_list[i]
    output_f_name = "./색인어파일/" + str(i) + ".txt"

    # 각 파일 indexing file 만듬
    cmd_exe = " ".join(["index2018.exe", input_f_name, output_f_name])
    subprocess.call(cmd_exe)

    # 각 파일에 unique한 색인어 목록 all_unique_indexing.txt파일에 써주기
    cmd_exe = " ".join(["wordcount.exe -new -uniq -i", output_f_name, "temp_unique_indexing.txt"])
    subprocess.call(cmd_exe)
    f_read = open("temp_unique_indexing.txt", 'r', encoding = "ANSI")
    lines = f_read.readlines()
    for line in lines:
        try:
            f.write(line)
        except:
            print("encoding error")
    f_read.close()

f.close()
```

결과는





1-2) 색인어 추출 결과를 아래 형태로 저장

DID: <term, freq> <term, freq> ... <term, freq>

1-2) 만든 인덱싱 파일과 wordcount.exe를 이용해서 모든 파일에 docid : (빈도수, 단어), (빈도수, 단어) 형태로 저장하자

```
forward_indexing_table = dict()

for i in range(len(f_list)):
    # word_count실행
    f_name = "./색인어파일/" + str(i) + ".txt"
    cmd_exe = " ".join(["wordcount.exe -new -i", f_name, f_name])
    subprocess.call(cmd_exe)

    #파일 열어서 forward_indexing_table만들기
    f = open(f_name, "r", encoding = "ANSI")
    word_list = []
    while True:
        try:
            line = f.readline()
            if not line:
                break
            word_list.append(line.split())
        except:
            print("encoding - error")
    # key는 docid = i 고 value는 해당파일 word_list를 가진 dictionary로 만든다.
    forward_indexing_table[i] = word_list.copy()
print(forward_indexing_table.items())
f.close()
```

```
dict_items([(0, [['1', '01'], ['1', '10'], ['1', '1105'], ['1', '1105만달러'], ['1', '13'], ['1', '13개'], ['1', '2'], ['1', '2007'],
['1', '2년'], ['1', '8'], ['1', '8일'], ['1', '9'], ['1', '9개국'], ['2', 'EC'], ['1', 'INA'], ['1', 'Inria'], ['1', 'LTU'], ['2', 'S
W'], ['2', 'SW업체'], ['1', 'SW업체인'], ['1', 'ZD'], ['1', 'ZD넷'], ['1', 'co'], ['1', 'etnews'], ['1', 'kr'], ['1', 'syjung'], ['1',
'syjung@etnews.co.kr'], ['1', '가전'], ['1', '가전업체'], ['1', '개'], ['1', '개국'], ['1', '개발'], ['11', '검색'], ['5', '검색엔진'],
```

파이썬의 dictionary를 활용하여 docid가 key이고 <frequency, term>의 리스트가 value인 forward_indexing_table을 만들었다. 밑에 출력코드에서 key가 0, 즉 docid가 0인 <frequency, term>으로 이루어진 리스트를 출력하고 있다.

1-3) df계산하기

1-3) wordcount.exe를 all_unique_indexing.txt(각 파일당 unique indexing list)에 이용하여 DF계산하자

In [5]:
```
output_f_name = "word_df.txt"
cmd_exe = " ".join(["wordcount.exe -new -i","all_unique_indexing.txt", output_f_name])
subprocess.call(cmd_exe)

# word_df 파일을 wordcount하여서 각 색인어당 df계산 [색인어, df]
f = open("word_df.txt", 'r')
lines = f.readlines()
df_table = dict()
for line in lines:
    try:
        a = line.split()
        df_table[a[1]] = a[0]
    except:
        pass
print(df_table)
```

```
{'%(': '2', '++': '1', '//': '1', '0': '1', '0.05%': '1', '0.07': '1', '0.07센트': '1', '0.10': '1', '0.10미크론': '1', '0.11': '1',
'0.11포인트': '1', '0.12': '1', '0.12미크론급': '1', '0.13': '1', '0.13미크론': '1', '0.14': '1', '0.14%': '1', '0.14센트': '1', '0.
2%': '1', '0.3%': '1', '0.63': '1', '0.63주': '1', '0.96': '1', '0.96달러': '1', '00': '17', '0000': '1', '000N': '1', '00년': '1', '0
1': '91', '010': '1', '011': '3', '016': '1', '017': '2', '018': '1', '019': '1', '02': '107', '026': '1', '03': '103', '030311': '1',
'031': '1', '03월': '1', '04': '90', '05': '78', '06': '69', '060': '1', '07': '80', '070': '1', '08': '87', '09': '85', '1': '302',
'1%': '7', '1.0': '13', '1.0버전': '2', '1.1': '2', '1.2': '3', '1.22%': '1', '1.25': '2', '1.2메가와트': '1', '1.3': '1', '1.33': '1',
'1.3배': '1', '1.4': '1', '1.40': '1', '1.43%': '1', '1.47': '1', '1.4M': '1', '1.5': '6', '1.5%': '2', '1.51': '1', '1.51%': '1', '1.5
1달러대': '1', '1.52': '1', '1.52달러': '1', '1.53': '1', '1.5M': '1', '1.5TB': '1', '1.5TB급': '1', '1.5kg': '1', '1.5배': '1', '1.6':
'3', '1.65%': '1', '1.6달러': '1', '1.7': '5', '1.7%': '1', '1.7퐁': '1', '1.8': '1', '1.8%': '1', '1.9': '1', '1.9%': '2', '1.92%':
'1', '1.95%': '1', '10': '256', '10%': '32', '10.13%': '1', '10.4': '1', '10.4인치': '1', '10.7': '1', '100': '86', '100%': '12', '100
0': '54', '10000': '1', '1000G': '1', '1000G플롭스': '1', '1000개': '5', '1000건': '2', '1000고지': '1', '1000기가플롭스': '1', '1000
대': '3', '1000대가': '1', '1000만': '3', '1000만달러': '4', '1000만대': '1', '1000만명': '2', '1000만원': '2', '1000만파운드': '1', '1
000명': '3', '1000명당': '1', '1000배': '1', '1000분': '1', '1000억': '1', '1000억엔': '1', '1000억원': '9', '1000억원의': '1', '1000여
가지의': '1', '1000여개': '3', '1000여곡': '1', '1000여명': '1', '1000원': '2', '1000퐁': '3', '1000퐁이상': '1', '1000포인트': '1', '1
000회': '1', '100G': '1', '100Gb': '1', '100G플롭스': '1', '100MB': '2', '100MB/s': '1', '100MB급': '1', '100MB벽': '1', '100N': '1',
```

1-1에서 만든 all_unique_indexing파일을 이용하여 전체 색인어에 대해 df계산하면 된다. 출력에서 보면 %(단어가 2번, ++ 단어가 1번 나온 것을 확인할 수 있다.

1-4) <tid,df> table 구성하기

1-4) <TID, DF> table 구성 (그냥 색인어 대신 TID를 넣어주면 된다.)

```
In [6]: tid_df_table = []
        index = 0
        for df in df_table.values():
            tid_df_table.append((index, df))
            index = index + 1
        print(tid_df_table)
```

```
[(0, '2'), (1, '1'), (2, '1'), (3, '1'), (4, '1'), (5, '1'), (6, '1'), (7, '1'), (8, '1'), (9, '1'), (10, '1'), (11, '1'), (12, '1'),
(13, '1'), (14, '1'), (15, '1'), (16, '1'), (17, '1'), (18, '1'), (19, '1'), (20, '1'), (21, '1'), (22, '1'), (23, '1'), (24, '17'), (2
5, '1'), (26, '1'), (27, '1'), (28, '91'), (29, '1'), (30, '3'), (31, '1'), (32, '2'), (33, '1'), (34, '1'), (35, '107'), (36, '1'), (3
7, '103'), (38, '1'), (39, '1'), (40, '1'), (41, '90'), (42, '78'), (43, '69'), (44, '1'), (45, '80'), (46, '1'), (47, '87'), (48, '8
5'), (49, '302'), (50, '7'), (51, '13'), (52, '2'), (53, '2'), (54, '3'), (55, '1'), (56, '2'), (57, '1'), (58, '1'), (59, '1'), (60,
'1'), (61, '1'), (62, '1'), (63, '1'), (64, '1'), (65, '1'), (66, '6'), (67, '2'), (68, '1'), (69, '1'), (70, '1'), (71, '1'), (72,
'1'), (73, '1'), (74, '1'), (75, '1'), (76, '1'), (77, '1'), (78, '1'), (79, '3'), (80, '1'), (81, '1'), (82, '5'), (83, '1'), (84,
'1'), (85, '1'), (86, '1'), (87, '1'), (88, '2'), (89, '1'), (90, '1'), (91, '256'), (92, '32'), (93, '1'), (94, '1'), (95, '1'), (96,
'1'), (97, '86'), (98, '12'), (99, '54'), (100, '1'), (101, '1'), (102, '1'), (103, '5'), (104, '2'), (105, '1'), (106, '1'), (107,
'3'), (108, '1'), (109, '3'), (110, '4'), (111, '1'), (112, '2'), (113, '2'), (114, '1'), (115, '3'), (116, '1'), (117, '1'), (118,
'1'), (119, '1'), (120, '1'), (121, '9'), (122, '1'), (123, '1'), (124, '3'), (125, '1'), (126, '1'), (127, '2'), (128, '3'), (129,
'1'), (130, '1'), (131, '1'), (132, '1'), (133, '1'), (134, '1'), (135, '2'), (136, '1'), (137, '1'), (138, '1'), (139, '1'), (140,
'2'), (141, '1'), (142, '1'), (143, '1'), (144, '1'), (145, '1'), (146, '1'), (147, '3'), (148, '1'), (149, '3'), (150, '1'), (151,
'1'), (152, '3'), (153, '6'), (154, '2'), (155, '1'), (156, '2'), (157, '3'), (158, '1'), (159, '1'), (160, '1'), (161, '2'), (162,
'1'), (163, '2'), (164, '1'), (165, '5'), (166, '12'), (167, '1'), (168, '1'), (169, '6'), (170, '3'), (171, '2'), (172, '1'), (173,
'1'), (174, '1'), (175, '1'), (176, '2'), (177, '1'), (178, '1'), (179, '1'), (180, '1'), (181, '3'), (182, '2'), (183, '3'), (184,
```

1-3에서 만든 df계산한 df_table을 이용해서 indexing만 해주면 된다.