



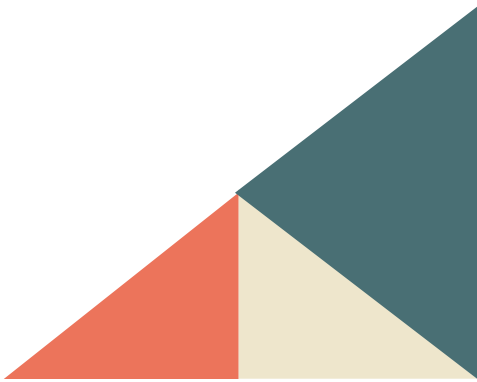
검색 엔진 구현

소프트웨어 공학과
조정근





목차

1. Forward Indexing
 2. Backward Indexing
 3. Inverted file 구성
 4. Keyword 검색
 5. 필터링
 6. 문서 출력
- 

1. Forward Indexing

```
# file read
terms = [] #set of terms #word vector
term_file_cnt = [] # cnt of file which has terms(index) #단어가 나온 파일의 수
term_freq = [] #freq of terms in each files
file_cnt = 0

for txt in f_list:
    try:
        file = open(path + txt, 'r', encoding='cp949')
    except UnicodeDecodeError:
        print(path + txt)
        file_cnt += 1
    s = set()
    file_terms_cnt = 0 #cnt of terms in a file #파일에 있는 단어의 수
    term_cnt = {} #freq of terms in a file #
    tmp = {}
    while(True):
        try:
            line = file.readline()
        except UnicodeDecodeError:
            print(path + txt[:-1])
            continue

        if not line:
            file.close()
            break

        if( len(line[:-1])<=10):
            continue
        try:
            morphs = kkma.morphs(line[:-1])
        except :
            print(line[:-1] + "line end")

        for m in morphs:
            file_terms_cnt += 1
            if( m not in s):
                s.add(m)
            if( m not in terms):
                terms.append(m)
                term_file_cnt.append(0)

            term_cnt[terms.index(m)] = term_cnt.get(terms.index(m),0) + 1

for k,v in term_cnt.items():
    tmp[k] = v/file_terms_cnt
term_freq.append(tmp)

l = list(s)
for m in l:
    term_file_cnt[terms.index(m)] += 1
```

1. 문서를 읽으면서 형태소 분석

2. word vector 에 없는 단어 추가

3. 각 단어별로 tf, df계산

1. Forward Indexing

```
#forward index, backward indexing
backward_index = [{ } for i in range(len(terms))]

# tf * idf
f = open('tf-idf.txt', 'w')
for i in range(len(term_freq)):
    for w_id, v in term_freq[i].items():
        tf_idf = v * math.log10(file_cnt / term_file_cnt[k])
        f.write('{}:{}'.format(w_id, tf_idf) )
        backward_index[w_id][i] = (tf_idf)
    f.write('\n')
f.close()
```

계산해둔 tf와 df값을 가지고 tf-idf계산

계산 하면서 <doc, word vector>

-> <word, doc vector> 변환

2. Backward Indexing

```
print(backward_index[0]) # 단어에 대한 문서-가중치  
# 0번 단어는 0~10번, 17번 21번 문서에 존재
```

```
{0: 0.008128341805277857, 1: 0.012192512707916785, 2: 0.012216005025273079, 3: 0.006215790792271302, 4: 0.006618065352940216, 5: 0.00949117755706097, 6: 0.03336898214798278, 7: 0.012334837758981962, 8: 0.004002592555629248, 9: 0.005513136180971068, 10: 0.004415116022365409, 17: 0.012334837758981962, 21: 0.006143514155151868, 23: 0.004255105106118609, 28: 0.004408975388120117, 30: 0.00540964}
```

0번 단어에 대한 결과

0번 단어는 0~10번 문서, 17, 21번 ... 문서에 존재

3. Inverted File 구성

```
term_table = []
posting_file = []
loc_of_files = 0

#backward_index
f = open('backward_index.txt', 'w')
for word in backward_index:
    n_of_files = 0
    for file, v in word.items():
        f.write('{}:{}'.format(file, v))

        term_table
        posting_file.append([file, v])
        n_of_files += 1

    term_table.append([loc_of_files, n_of_files])
    loc_of_files += n_of_files

    f.write('\n')
f.close()

f = open('posting_file.txt', 'w')
for doc_weight in posting_file:
    f.write(doc_weight)
    f.write('\n')
f.close()
```

term table, posting file로 분해

```
print(term_table[:10])
# 0번 단어는 0부터 322개 # 1번 단어는 322 부터 245개
#상위 10개만 출력
```

```
[[0, 322], [322, 245], [567, 431], [998, 25], [1023, 49], [1072, 1], [1073, 4], [1077, 8], [1085, 991], [2076, 47]]
```

```
print(posting_file[320:330])
# 0번 단어에 대한 문서-가중치 : 133 까지 / 1번 단어에 대한 문서-가중치 : 134 부터
#경계 값을 보기위해 134 근처 출력
```

```
[[982, 0.002691046947417966], [986, 0.0042954651816509], [0, 0.012192512707916785], [22, 0.0034307936191107836], [24, 0.00254827436
01755336], [26, 0.001697030676690773], [28, 0.0022044876940600583], [30, 0.0027048236382750546], [34, 0.009645192608696848], [43,
0.0022675631645624923]]
```

4. Keyword 검색

#keyword 검색

```
keyword = list(map(str, input()[:].split(' ')))  
#keyword = kkma.morphs(str(input()))  
print(keyword)
```

삼성 노트북
['삼성', '노트북']

#keyword에 해당하는 word id

```
keyword_index = [terms.index(k) for k in keyword]  
print(keyword_index)
```

[1059, 2171]

#termtable 검색한 결과

```
result_term_table = [term_table[i] for i in keyword_index]  
print(result_term_table)
```

[[112230, 63], [155784, 34]]

```
result_calc_weight = [[w, file_idx] for file_idx, w in calc_weight.items() if w >= 0.01]  
result_calc_weight.sort(reverse = True)  
print(result_calc_weight[:])
```

```
[[0.04081613696212486, 180], [0.03316946655965403, 836], [0.03177113556405933, 178], [0.028115772098078615, 640], [0.02497617754712  
6506, 179], [0.024015555333775487, 22], [0.02213724374342433, 140], [0.02204828418587734, 216], [0.02189263331532019, 514], [0.0217  
1269386341345, 396], [0.0140267845312317, 482], [0.013319551697724218, 50], [0.012938993077789241, 183], [0.010874968453030409, 33  
5], [0.010800863046195448, 285], [0.01065206083352945, 195], [0.01031470706309663, 215], [0.010095711159421542, 217], [0.0100636612  
search_result = [ f_list[i[i]] for i in result_calc_weight]  
print(search_result[:])
```

['IT-notebookUS.txt', '뜨거운 우정 차가운 승부.txt', 'IT-notebook.txt', 'news-W\$notebook.txt', 'IT-notebookProtect.txt', 'IT-20HPno
tebook.txt', 'IT-LGIBM.txt', 'IT-samsungHDD.txt', 'news-noteb.txt', 'news-eWork.txt', 'news-LCD.txt', 'IT-dualcoreNB.txt', 'IT-oneN
ote.txt', 'news-Cebit.txt', 'IT-windowVista.txt', 'IT-PC.txt', 'IT-samsung.txt', 'IT-samsungLCD.txt', '[음주단속] 소주1병 마시면 8
시간 지나야 안심.txt']

5. 필터링

```
#keyword 헤 해당하는 word id
keyword_index = [terms.index(k) for k in keyword if k[0] != '-']
print(keyword_index)

removal_keyword_index = [terms.index(k[1:]) for k in keyword if k[0]=='-'] # 검색 제외할 문서

#termtable 검색한 결과
result_term_table = [term_table[i] for i in keyword_index]
print(result_term_table)

result_term_table_removal = [term_table[i] for i in removal_keyword_index] # 제외할 문서->termtable 검색

#posting_file 검색 결과
# 제외 할 문서->posting file 에서 검색 |
result_posting_file_removal = [posting_file[idx:idx+cnt] for idx, cnt in result_term_table_removal ]
result_removal_file = [idx for word in result_posting_file_removal for idx,weight in word] # 제외 할 doc id 목록

result_posting_file = [ posting_file[idx:idx+cnt] for idx, cnt in result_term_table ]
print(result_posting_file)
#calc_weight = [ for idx, cnt in result_term_table for file_idx, w in posting_file[idx:idx+cnt] ]
#term_cnt[terms.index(m)] = term_cnt.get(terms.index(m),0) + 1

calc_weight = {}
for idx, cnt in result_term_table:
    for file_idx, w in posting_file[idx:idx+cnt]:
        if(file_idx in result_removal_file): # 제외 keyword 제외
            continue
        calc_weight[file_idx] = calc_weight.get(file_idx, 0) + w
print( calc_weight )

result_calc_weight = [[w, file_idx] for file_idx, w in calc_weight.items() if w >= 0.01 ]
result_calc_weight.sort(reverse = True)
print(result_calc_weight[:])

search_result = [ f_list[i[1]] for i in result_calc_weight]
print(search_result[:])
```


5. 필터링

기존의 코드에서 '삼성'으로 검색한 최종 결과는

```
#keyword 검색
```

```
keyword = list(map(str, input()[:].split(' ')))  
#keyword = kkma.morphs(str(input()))  
print(keyword)
```

```
삼성  
['삼성']
```

```
search_result = [ f_list[i[1]] for i in result_calc_weight]  
print(search_result[:])
```

```
['뜨거운 우정 차가운 승부.txt', 'IT-samsungHDD.txt', 'news-eWork.txt', 'news-LCD.txt', 'IT-samsung.txt', '[음주단속] 소주1병 마시면  
8시간 지나야 안심.txt']
```

이었지만, '삼성 -야구'로 검색해 '야구'에 대해서 필터링 한 결과는 '뜨거운 우정 차가운 승부.txt'가 필터링 되어 보이지 않는다.

```
#keyword 검색
```

```
keyword = list(map(str, input()[:].split(' ')))  
#keyword = kkma.morphs(str(input()))  
print(keyword)
```

```
삼성 -야구  
['삼성', '-야구']
```

```
search_result = [ f_list[i[1]] for i in result_calc_weight]  
print(search_result[:])
```

```
['IT-samsungHDD.txt', 'news-eWork.txt', 'news-LCD.txt', 'IT-samsung.txt', '[음주단속] 소주1병 마시면 8시간 지나야 안심.txt']
```

6. 문서 출력

검색을 마친 후 읽어 올 문서의 번호를 입력 받아 입력한 번호의 문서를 보여준다.

```
while(True) :  
    i=input('i : 읽을 문서 번호, enter : stop')  
    if( i== '' or i == 'enter'):  
        break  
    try :  
        i = int(i) - 1  
    except:  
        print('enter number or enter')  
        continue  
    f = open(path+search_result[i], 'r')  
    while(True):  
        line = f.readline()  
        if not line:  
            f.close()  
            break  
        if( len(line)<10):  
            continue  
        print(line)
```

i : 읽을 문서 번호, enter : stop |

i : 읽을 문서 번호, enter : stop1
뜨거운 우정 차가운 승부

2002 프로야구 한국시리즈 패권을 놓고 1990년에 이어 12년 만에 맞붙은 삼성과 LG의 처지가 알쏭다. 우선 팀만 놓고 보면 LG는 90년, 94년 두 차례 한국시리즈 정상을 차지한 반면, 삼성은 창단 이래 7차례나 한국시리즈에 올랐지만, 번번이 눈물만 뿌렸다.

하지만 양 팀 사령탑의 운명은 판이하다. 삼성 김응용 감독은 9차례나 우승을 차지한 '우승제조기' . 이와반대로 LG 김성근 감독은 감독 데뷔 20년 만에 처음으로 한국시리즈에 올랐을 만큼 한국시리즈와는 인연이 없다.



THANK YOU

