

# { 5주차 과제 발표 }

1. 신문 기사 단어 별 구분

2. 단어 빈도 count

3. Unigram 빈도 계산

4. Unigram wordcount 구현

# 1. 단어(word) 별 구분

```
1 import re
2 import string
3 import os
4 import sys
5
6 from collections import Counter
7
8
9 n = 1
10 file = open("input.txt", "r")
11 fout = open('finword.txt', 'w')
12
13 word = file.read()
14 tokens = word.split()
15
16 if (len(tokens) < n) :
17     print("wrong")
18 else :
19     for i in range(len(tokens) - (n-1)) :
20         for j in range(n) :
21             fout.write(tokens[i+j].decode('cp949').encode('utf-8'))
22             fout.write("\n")
23
24
```

색상	단어
Red	웹
Yellow	검색시장
Green	'태깅'
Red	뜬다
Yellow	검색엔진
Green	대신
Red	고객이
Yellow	웹사이트에
Green	주제어를
Red	붙이는
Yellow	'태깅(tagging)'이
Green	웹
Red	검색시장의
Yellow	최대
Green	화두로
Red	떠올랐다고
Yellow	월스트리트저널(WSJ)이
Green	25일
Red	보도했다.
Yellow	이는
Green	네이트이
Red	직접
Yellow	입력한
Green	태깅정보를
Red	공유하는
Yellow	정보사이트가
Green	우후죽순으로
Red	생겨나면서
Yellow	구글,
Green	야후등

## 2. 단어(word) 빈도 count

```
1 import re
2 import string
3 import os
4 import sys
5
6 from collections import Counter
7
8 n = 1
9 file = open("input.txt", "r")
10 fout = open('finword.txt', 'w')
11
12 word = file.read()
13 tokens = word.split()
14
15
16
17
18 count = Counter(tokens)
19
20 tag_count = []
21
22 tags = []
23
24 for n, c in count.most_common(100):
25
26     dics = {'tag': n, 'count': c}
27
28     if len(dics['tag']) >= 2 and len(tags) <= 49:
29
30         tag_count.append(dics)
31
32         tags.append(dics['tag'])
33
34 for tag in tag_count:
35
36     fout.write(" {:<14}{}".format(tag['tag']).decode('cp949').encode('utf-8'))
37
38     fout.write(" {}".format(tag['count']).decode('cp949').encode('utf-8'))
39
```

검색	23 정보	16 태그	15 사이트	10	
엔진	9 검색엔진	9 입력	7 태깅	6 사진	
5 웹	5 공유	5 네티즌	4 것	4	
com	4 구글	4 등	3 닷컴	3 정	
보검색	3 정확	3 인수	3 가능	3 야후	
3 주목	3 지도	3 서비스	3 딜리셔스	3 플리	
커	3 시장	3 전자	2 사용자	2 25	
2 대형	2 파리	2 검색업체	2 신문	2 온	
라인	2 정보사이트	2 우후	2 즐겨찾기	2 기술	
2 표식	2 업체	2 과정	2 검색시장	2 사	
람	2 공유사이트	2 커뮤니티	2 태그정보	2 주제어	
2 죽순	2				

### 3. Unigram 빈도 계산

```
1 #include <stdio.h>
2 #include <string.h>
3
4 float count = 0;
5 float freq;
6 float prob = 1.0;
7
8 void sum(FILE* fp)
9 {
10    int ii;
11    char c1[100];
12    fp = fopen("finword.txt", "r");
13
14    while(!feof(fp))
15    {
16        fscanf(fp,"%s %d", &c1, &ii);
17        if(feof(fp)) break;
18        count += ii;
19    }
20}
21
22 int main()
23 {
24
25    FILE *fp = NULL;
26
27    fp = fopen( "finword.txt", "r" );
28
29    sum(fp);
30    printf("count : %f\n", count);
31
32    while(!feof(fp))
33    {
34        char c[100];
35        int i;
36
37        fscanf(fp,"%s %d", &c, &i);
38        if(feof(fp)) break;
39
40        printf("%s %d \t", c, i);
41        freq = i/count;
42        printf("freq = %f\n", freq);
43        prob *= freq;
44
45    }
46    printf("probability : %f\n", prob);
47
48    fclose(fp);
49
50    return 0;
51}
52
```

```
gwonbogyeong-ui-MacBook-Pro:hw4 bokyeong$ ./a.out
count : 207.000000
검색 23 freq = 0.111111
정보 16 freq = 0.077295
태그 15 freq = 0.072464
사이트 10 freq = 0.048309
엔진 9 freq = 0.043478
검색 엔진 9 freq = 0.043478
입력 7 freq = 0.033816
태깅 6 freq = 0.028986
사진 5 freq = 0.024155
웹 5 freq = 0.024155
공유 5 freq = 0.024155
네티즌 4 freq = 0.019324
것 4 freq = 0.019324
com 4 freq = 0.019324
구글 4 freq = 0.019324
등 3 freq = 0.014493
닷컴 3 freq = 0.014493
정보검색 3 freq = 0.014493
정확 3 freq = 0.014493
인수 3 freq = 0.014493
가능 3 freq = 0.014493
야후 3 freq = 0.014493
주목 3 freq = 0.014493
지도 3 freq = 0.014493
서비스 3 freq = 0.014493
딜리셔스 3 freq = 0.014493
플리커 3 freq = 0.014493
시장 3 freq = 0.014493
```

## 4. Unigram wordcloud 구현

```
1 from collections import Counter
2 import urllib
3 import random
4 import webbrowser
5
6 from konlpy.tag import Twitter
7 from lxml import html
8 import pytagcloud
9 import sys
10
11 r = lambda: random.randint(0,255)
12 color = lambda: (r(), r(), r())
13
14 def get_tags(text, ntags=50, multiplier=3):
15     spliter = Twitter()
16     nouns = spliter.nouns(text)
17     count = Counter(nouns)
18     return [{ 'color': color(), 'tag': n, 'size': (c*multiplier)/2 }\
19             for n, c in count.most_common(ntags)]
20
21 def draw_cloud(tags, filename, fontname='Noto Sans CJK', size=(800, 600)):
22     pytagcloud.create_tag_image(tags, filename, fontname=fontname, size=size)
23     webbrowser.open(filename)
24
25 def main():
26     text_file=open("finword.txt",'r')
27     text=text_file.read()
28     tags = get_tags(text)
29     draw_cloud(tags, 'wordcloud.png')
30
31     text_file.close()
32
33     if __name__=="__main__":
34         main()
```



# 감사합니다.