Data Analytics Week 6 Assignment

202011431 산업공학과 차승현

Vector Space Model (VSM)

건국대학교

Analysis Procedure

1) 필요한 모듈 pandas, math를 import하고, document1, 2, 3을 입력한다. 추가로, 조사 같이 실효성 없는 stop word를 제외한다.

```
In [96]: import pandas as pd
         import math

In [1]: doc1 = '경찰청 철창살은 외철창살이냐 쌍철창살이냐 경찰청 철창살이 쇠철창살이냐 철철창살이냐 검찰청 쇠철창살은 새쇠철창살이냐 헌쇠철창살이니
         doc2 = '내가 그린 기린 그림은 잘 그린 기린 그림이고 네가 그린 기린 그림은 잘 못 그린 기린 그림이다. 내가 그린 기린 그림은 긴 기린 그림이냐,
         doc3 = '안촉촉한 초코칩 나라에 살던 안촉촉한 초코칩이 촉촉한 초코칩 나라의 촉촉한 초코칩을 보고 촉촉한 초코칩이 되고 싶어서 촉촉한 초코칩 나

In [42]: stop_words_lst = ['은', '이고', '이다', '이니까', '에서', '이냐', '이', '가', '?', '에', '의', '을', '"', ',', '.',]

In [25]: docs = [doc1, doc2, doc3]

In [43]: docs = [doc1, doc2, doc3]

         for i in range(len(docs)):
             for j in range(len(stop_words_lst)):
                 if stop_words_lst[j] in docs[i]:
                     docs[i] = docs[i].replace(stop_words_lst[j], '')

In [40]: docs

Out[40]: ['경찰청 철창살 외철창살 쌍철창살 경찰청 철창살 쇠철창살 철철창살검찰청 쇠철창살 새쇠철창살 헌쇠철창살 경찰청 쇠창살 외철창살 검찰청 쇠창
         살 쌍철창살',
          '내 그린 기린 그림 잘 그린 기린 그림 네 그린 기린 그림 잘 못 그린 기린 그림다 내 그린 기린 그림 긴 기린 그림 그냥 그린 기린 그림 내 그린
         구름그림 새털구름 그린 구름그림 네 그린 구름그림 깃털구름 그린 구름그림다',
          '안촉촉한 초코칩 나라 살던 안촉촉한 초코칩 촉촉한 초코칩 나라 촉촉한 초코칩 보고 촉촉한 초코칩 되고 싶어서 촉촉한 초코칩 나라 갔는데 촉촉
         한 초코칩 나라 촉촉한 초코칩 문지기 넌 촉촉한 초코칩 아니고 안촉촉한 초코칩 안촉촉한 초코칩나라 살아라고해서 안촉촉한 초코칩 촉촉한 초코칩
         되는것 포기하고 안촉촉한 초코칩 나라로 돌아갔다']
```

은, 이고, 이다, 이니까와 같은 stop word들이 성공적으로 제거되었다.

2) 각 document에서 단어를 추출한다. (split 함수 사용)

```
In [51]: docs1_splt = docs[0].split()
         docs2_splt = docs[1].split()
         docs3_splt = docs[2].split()
```

```
In [8]: docs1_splt
```

```
Out[8]: ['경찰청',
 '철창살',
 '외철창살',
 '쌍철창살',
 '경찰청',
 '철창살',
 '쇠철창살',
 '철철창살',
 '검찰청',
 '쇠철창살',
 '새쇠철창살',
 '헌쇠철창살',
 '경찰청',
 '쇠창살',
 '외철창살',
 '검찰청',
 '쇠창살',
 '쌍철창살']
```

성공적으로 추출이 되었다.

3) 각 document들의 추출된 단어를 wrd_lst라는 변수에 합병하여 넣는다.

```
In [72]: wrd_lst = []
         for i in range(len(docs1_splt)):
             wrd_lst.append(docs1_splt[i])

         for j in range(len(docs2_splt)):
             wrd_lst.append(docs2_splt[j])

         for k in range(len(docs3_splt)):
             wrd_lst.append(docs3_splt[k])
```

```
In [75]: word_lst = sorted(list(set(wrd_lst)))
```

```
In [76]: word_lst

Out[76]: ['갔는데',
          '검찰청',
          '경찰청',
          '구름그림',
          '그냥',
          '그린',
          '그림',
          '기린',
          '긴',
          '깃털구름',
          '나라',
          '나라로',
          '내',
          '넌',
          '네',
          '돌아갔다',
          '되고',
          '되는것',
          '못',
          '문지기',
          '보고',
          '살던',
          '살아라고해서',
          '새쇠철창살',
          '새털구름',
          '쇠창살',
          '쇠철창살',
          '싶어서',
          '쌍철창살',
          '아니고',
          '안촉촉한',
          '외철창살',
          '잘',
          '철창살',
          '철철창살검찰청',
          '초코칩',
          '초코칩나라',
          '촉촉한',
          '포기하고',
          '헌쇠철창살']
```

4) tf-based VSM (simple)

tf-based VSM representation (simple)

In [99]:
```
value = []

for rows in range(3):
    value.append([])
    d = docs[rows]
    for columns in range(len(word_lst)):
        t = word_lst[columns]
        value[-1].append(d.count(t))

vsm_simple_df = pd.DataFrame(value, columns=word_lst, index=['doc1', 'doc2', 'doc3'])
vsm_simple_df
```

Out [99]:

	갔는데	검찰청	경찰청	구름그림	그냥	그린	그림	기린	긴	깃털구름	...	안측측한	외철창살	잘	철창살	철철창살검찰청	초코칩	초코칩나라	측측한	포기하고	헌쇠철창살
doc1	0	2	3	0	0	0	0	0	0	0	...	0	2	0	11	1	0	0	0	0	1
doc2	0	0	0	4	1	10	11	7	1	1	...	0	0	2	0	0	0	0	0	0	0
doc3	1	0	0	0	0	0	0	0	0	0	...	6	0	0	0	0	14	1	14	1	0

3 rows × 40 columns

5) tf-based VSM (Boolean)

In [100]:
```
vsm_boolean_df = vsm_simple_df.copy()
for rows in range(len(vsm_boolean_df.index)):
    for columns in range(len(vsm_boolean_df.columns)):
        if vsm_boolean_df.iloc[rows, columns] >= 1:
            vsm_boolean_df.iloc[rows, columns] = 1

vsm_boolean_df
```

Out [100]:

	갔는데	검찰청	경찰청	구름그림	그냥	그린	그림	기린	긴	깃털구름	...	안측측한	외철창살	잘	철창살	철철창살검찰청	초코칩	초코칩나라	측측한	포기하고	헌쇠철창살
doc1	0	1	1	0	0	0	0	0	0	0	...	0	1	0	1	1	0	0	0	0	1
doc2	0	0	0	1	1	1	1	1	1	1	...	0	0	1	0	0	0	0	0	0	0
doc3	1	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0	1	1	1	1	0

3 rows × 40 columns

6) tf-based VSM (Logarithmatically)

tf-based VSM representation (logarithmatically scaled)

In [101]:
```
value_2 = []

for rows in range(3):
    value_2.append([])
    d = docs[rows]
    for columns in range(len(word_lst)):
        t = word_lst[columns]
        value_2[-1].append(math.log(1+d.count(t)))

vsm_log_df = pd.DataFrame(value_2, columns=word_lst, index=['doc1', 'doc2', 'doc3'])
vsm_log_df
```

Out [101]:

	갔는데	검찰청	경찰청	구름그림	그냥	그린	그림	기린	긴	깃털구름	...	안측측한	외철창살	잘	철창살	철철창살검찰
doc1	0.000000	1.098612	1.386294	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.00000	1.098612	0.000000	2.484907	0.69314
doc2	0.000000	0.000000	0.000000	1.609438	0.693147	2.397895	2.484907	2.079442	0.693147	0.693147	...	0.00000	0.000000	1.098612	0.000000	0.00000
doc3	0.693147	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	1.94591	0.000000	0.000000	0.000000	0.00000

3 rows × 40 columns

7) 각 row와 column에 들어간 value는 다음과 같다.

```
In [112]: for i in range(3):
              print(i+1,' 번째 doc의 vsm_simple word value: ')
              print(vsm_simple_df.iloc[i, :])

              print(i+1,' 번째 doc의 vsm_boolean word value: ')
              print(vsm_boolean_df.iloc[i, :])

              print(i+1,' 번째 doc의 vsm_log word value: ')
              print(vsm_log_df.iloc[i, :])
```

```
1  번째 doc의 vsm_simple word value:
갔는데          0
검찰청          2
경찰청          3
구름그림         0
```

(다음과 같은 코드를 사용하여서 전체적인 data frame의 값을 확인하였음.)

(1) tf-based VSM (simple) result

tf-based VSM result (simple)

	갔는데	검찰청	경찰청	구름그림	그냥	그린	그림	기린	긴	깃털구름
doc1	0	2	3	0	0	0	0	0	0	0
doc2	0	0	0	4	1	10	11	7	1	1
doc3	1	0	0	0	0	0	0	0	0	0
	나라	나라로	내	넌	네	돌아갔다	되고	되는것	못	문지기
doc1	0	0	0	0	0	0	0	0	0	0
doc2	0	0	3	0	2	0	0	0	1	0
doc3	6	1	0	1	0	1	1	1	0	1
	보고	살던	살아라고해새	쇠철창살	새털구름	쇠창살	쇠철창살	싶어서	쌍철창살	아니고
doc1	0	0	0	1	0	2	4	0	2	0
doc2	0	0	0	0	1	0	0	0	0	0
doc3	1	1	1	0	0	0	0	1	0	1
	외철창살	안촉촉한	잘	철창살	철철창살	초코칩	초코칩나라	촉촉한	포기하고	헌쇠철창살
doc1	2	0	0	11	1	0	0	0	0	1
doc2	0	0	2	0	0	0	0	0	0	0
doc3	0	6	0	0	0	14	1	14	1	0

(2) tf-based VSM (Boolean) result

tf-based VSM result (boolean)

	갔는데	검찰청	경찰청	구름그림	그냥	그린	그림	기린	긴	깃털구름
doc1	0	1	1	0	0	0	0	0	0	0
doc2	0	0	0	1	1	1	1	1	1	1
doc3	1	0	0	0	0	0	0	0	0	0
	나라	나라로	내	넌	네	돌아갔다	되고	되는것	못	문지기
doc1	0	0	0	0	0	0	0	0	0	0
doc2	0	0	1	0	1	0	0	0	1	0
doc3	1	1	0	1	0	1	1	1	0	1
	보고	살던	살아라고해	새쇠철창살	새털구름	쇠창살	쇠철창살	싶어서	쌍철창살	아니고
doc1	0	0	0	1	0	1	1	0	1	0
doc2	0	0	0	0	1	0	0	0	0	0
doc3	1	1	1	0	0	0	0	1	0	1
	외철창살	안촉촉한	잘	철창살	철철창살	초코칩	초코칩나라	촉촉한	포기하고	헌쇠철창살
doc1	1	0	0	1	1	0	0	0	0	1
doc2	0	0	1	0	0	0	0	0	0	0
doc3	0	1	0	0	0	1	1	1	1	0

(3) tf-based VSM (Logarithmatically) result

tf-based VSM result (logarithmatically)

	갔는데	검찰청	경찰청	구름그림	그냥	그린	그림	기린	긴	깃털구름
doc1	0	1.098612	1.386294	0	0	0	0	0	0	0
doc2	0	0	0	1.609438	0.693147	2.397895	2.484907	2.079442	0.693147	0.693147
doc3	0.693147	0	0	0	0	0	0	0	0	0
	나라	나라로	내	넌	네	돌아갔다	되고	되는것	못	문지기
doc1	0	0	0	0	0	0	0	0	0	0
doc2	0	0	1.386294	0	1.098612	0	0	0	0.693147	0
doc3	1.94591	0.693147	0	0.693147	0	0.693147	0.693147	0.693147	0	0.693147
	보고	살던	살아라고해	새쇠철창살	새털구름	쇠창살	쇠철창살	싶어서	쌍철창살	아니고
doc1	0	0	0	0.693147	0	1.098612	1.609438	0	1.098612	0
doc2	0	0	0	0	0.693147	0	0	0	0	0
doc3	0.693147	0.693147	0.693147	0	0	0	0	0.693147	0	0.693147
	외철창살	안촉촉한	잘	철창살	철철창살	초코칩	초코칩나라	촉촉한	포기하고	헌쇠철창살
doc1	1.098612	0	0	2.484907	0.693147	0	0	0	0	0.693147
doc2	0	0	1.098612	0	0	0	0	0	0	0
doc3	0	1.94591	0	0	0	2.70805	0.693147	2.70805	0.693147	0

8) 결과 해석

(1) tf-based VSM (simple) result의 doc1에서 가장 빈도가 높은 단어는 철창살'로 빈도수 11이였고, doc2에서는 '그림'으로 마찬가지로 빈도수가 11이다. Doc3에서는 '초코칩'과 '촉촉한'이 모두 14번의 빈도수를 나타냈다.

(2) tf-based VSM (Boolean) result를 열 별(column 및 단어 별)로 보았을

때, 각 doc마다 중복되는 단어가 하나도 없음을 확인할 수 있었다.

(3) tf-based VSM (Logarithmatically) result과 (1) tf-based VSM (simple) result를 비교하였을 때, 각 doc마다 빈도수가 높은 것들이 (3) tf-based VSM (Logarithmatically) result에서도 높은 value를 나타냄을 확인할 수 있었다. 해당 공식 $\log(1+f_{t,d})$이 애초에 raw count를 기반으로 하였으므로 정비례하는 관계를 가짐을 알 수 있다. 이는 자연스럽게 (1) 의 결과와 마찬가지로 각 doc마다 value가 가장 높은 단어는 doc1에서 '철창살'이 2.84907, doc2에서 '그림'이 2.484907, doc3에서는 '초코칩'과 '촉촉한'이 2.70805순으로 나타났다.

Python Code Reference: https://wikidocs.net/31698