# Business Advocate - BA Score

**Tuan Tran**

June 26, 2019

**Giới Thiệu** Trong tài liệu này sẽ giới thiệu định nghĩa, ý tưởng và giải thích các scripts dùng để tính toán điểm Business Advocate (BA) score.

## 1 Giới Thiệu về BA score

**Định Nghĩa** BA score có ý nghĩa tượng trưng biểu diễn cho khả năng làm advocate cho một nhãn hàng nào đó. Hiện tại BA score được tính dựa trên AF score, trong đó có 2 phần.

- BA score của một bài đăng đối với một nhãn hàng được tính bằng AF score, tuy nhiên bài đăng đó phải được đánh mentioned tới nhãn hàng đó.

$$BA\_score\ of\ a\ post = AF\_score\ of\ the\ post\ mentioned\ a\ brand$$

- BA score của một người đối với 1 nhãn hàng nào đó sẽ được tính bằng tổng $log(AF)$ của những bài post tới nhãn hàng đó.

$$A\ user's\ BA\_score = \sum \log(AF_i)$$

## 2 Diễn giải về script

Trước hết cần phải import các library cần thiết đặc biệt là pymongo bởi vì database là MongoDB. Và sau đó là tạo kết nối với hai database, KAPI and KPI-V2. Vì chúng ta cần tính AF cho những bài post đã được đánh "mention to", nên cần phải query database KPI-V2.

```
import pymongo
from pymongo import MongoClient
import datetime
import pandas as pd
import numpy as np
import gc
gc.enable()
```

```
client = MongoClient('45.122.223.198:27017',    #IP address of database
                  username = 'kpi-v2R',          #Username
                  password = 'EecvKJxdTQ1JEK8J2FA', #Password
                  authSource = 'kpi-v2',         #name of database
                  authMechanism = 'SCRAM-SHA-1')
kpitest = client['kpi-v2']

client_1 = MongoClient('45.122.223.198:27017',  #IP address of database
                  username = 'kapiReadOnly',     #Username
                  password = 'pl2oieAt9#tnWV!Yc0', #Password
                  authSource = 'kapi',           #name of database
                  authMechanism = 'SCRAM-SHA-1')
kapi = client_1['kapi']
```

Bước tiếp theo là tạo các function để query data từ database. Sẽ có 3 functions để liên quang nhau để có thể query và aggregate data cần thiết. Function đầu tiên là các id của bài post, query tất cả những người đã tương tác (likes, comments và shares) với bài post đó từ các bảng 'reactions', 'comments', 'posts', sẽ trả ra 3 bảng lần lượt là tb1, tb2 và tb3. Ba bảng này sẽ được dùng để tính distinct sum của các lượt tương tác về sau.

```
def int_fun(post_id):
    cluster1 = database_1['reactions'].find({'fid': {'$in': post_id}},
                                        {'_id': 0, 'fid': 1, 'from_user_id': 1})
    cluster2 = database_1['comments'].find({'post_id': {'$in': post_id}},
                                       {'_id': 0, 'post_id': 1, 'from_raw_user':
                                          1, 'from_user': 1})
    cluster3 = database_1['posts'].find({'parent_id': {'$in': post_id}},
                                    {'_id': 0, 'parent_id': 1, 'from_user': 1,
                                        'comments_count': 1,
                                     'likes_count': 1})

    tb1 = pd.DataFrame()
    for a in cluster1:
        tmp = pd.DataFrame([a])
        tb1 = tb1.append(tmp, ignore_index=True)

    tb2 = pd.DataFrame()
    for b in cluster2:
        tmp = pd.DataFrame([b])
        tb2 = tb2.append(tmp, ignore_index=True)
```

```
    tb3 = pd.DataFrame()
    for c in cluster3:
        tmp = pd.DataFrame([c])
        tb3 = tb3.append(tmp, ignore_index=True)

    for i in range(len(tb2)):
        if tb2.from_user[i] == '':
            tb2.from_user[i] = tb2.from_raw_user[i]

    return tb1, tb2, tb3
```

Function thứ hai được dùng để gọi các thông tin cần thiết khác từ các id của bài posts.

```
def info_fun(fid_list):
    cluster = database_1['posts'].find({'fid': {'$in': fid_list}},
                                       {'_id': 0, 'fid': 1, 'to_user': 1,
                                           'created_date': 1,
                                        'comments_count': 1, 'shares_count': 1,
                                           'likes_count': 1})

    info_tb = pd.DataFrame()
    for i in cluster:
        tmp = pd.DataFrame([i])
        info_tb = info_tb.append(tmp, ignore_index=True)

    return info_tb
```

Function cuối cùng, cũng là function quan trọng nhất. Trong function này, cần phải khai báo danh sách user cần tính AF, thời gian tính AF score (ngày bắt đầu, ngày kết thúc), speed (thường để là 10 để tránh tình trạng timeout). Function này sẽ sử dụng các function ở trên để query data cần thiết sau đó aggregate cũng như tính toán điểm AF score cho từng bài posts của các user id mà chúng ta cần tính, trong khoảng thời gian đã xác định. Trong Function này chỉ query những tất cả bài post đã đánh "mention to" một nhãn hàng hoặc industry.

```
def retrieve_fun(id_list, industry, start_date, end_date, speed):
    tab1 = kpitest['posts'].find({'from_user': {'$in': id_list}, 'brand': {'$ne':
        None},
                                  'created_date': {'$gte': start_date, '$lt':
                                      end_date},
                                  'industry': industry
                                    }, {
```

```
                                '_id': 0, 'fid': 1, 'brand': 1
                            })

ipt = pd.DataFrame()
for i in tab1:
    tmp = pd.DataFrame([i])
    ipt = ipt.append(tmp, ignore_index=True)

print("There are {:} messages for processing .".format(len(ipt)))

int_tmp = pd.DataFrame()
cmt_tmp_tb = pd.DataFrame()
shr_tmp_tb = pd.DataFrame()
len_tmp = (len(ipt) // speed + 1)

try:
    for i in range(len_tmp):
        print('==Processing to==> {:.4}%'.format(((i + 1) * 100) / len_tmp))
        tb1_tmp, tb2_tmp, tb3_tmp = int_fun(ipt['fid'][i * speed:(i + 1) *
            speed].tolist())
        int_tmp = int_tmp.append(tb1_tmp, ignore_index=True)
        cmt_tmp_tb = cmt_tmp_tb.append(tb2_tmp, ignore_index=True)
        shr_tmp_tb = shr_tmp_tb.append(tb3_tmp, ignore_index=True)
except pymongo.errors.CursorNotFound:
    print("Lost cursor. Retry with skip")

cmt_tmp = pd.DataFrame({'fid': cmt_tmp_tb.post_id, 'from_user_id':
    cmt_tmp_tb.from_user})
shr_tmp = pd.DataFrame({'fid': shr_tmp_tb.parent_id, 'from_user_id':
    shr_tmp_tb.from_user})

shr_tmp_tb['sub score'] = shr_tmp_tb['comments_count'] +
    shr_tmp_tb['likes_count']
af_tmp = shr_tmp_tb.groupby('parent_id').agg({'sub score':
    'sum'}).reset_index()
af_tmp.columns = ['fid', 'sub score']

total_int_tmp = int_tmp.append([cmt_tmp,
    shr_tmp]).drop_duplicates().reset_index(drop=True)
total_int_tmp = total_int_tmp.groupby('fid').agg('count')
total_int_tmp = af_tmp.set_index('fid').join(total_int_tmp, how='right')

post_query_tmp = info_fun(total_int_tmp.index.tolist())
post_query_tmp = post_query_tmp.set_index('fid')
```

```
    combine_tb = total_int_tmp.join(post_query_tmp)
    combine_tb = combine_tb.join(ipt.set_index('fid'))
    combine_tb['AF score'] = [np.nansum([x, y]) for x, y in
                          zip(combine_tb['from_user_id'], combine_tb['sub
                              score'])];

    gc.collect()

    return combine_tb
```

Phần cuối là các lệnh cần thiết, gọi các function ở trên để tính toán BA score cho từng posts. Sau đó là lưu bảng kết quả ra file '.csv'

```
start_date = datetime.datetime(2018,7,1) #Year, Month, Day of start date
end_date = datetime.datetime(2018,12,31) #Year, Month, Day of end date

r_tb = retrieve_fun(c_list, start_date, end_date, speed=10)

r_tb.to_csv('c_list.csv')
```

Sau khi tính được BA score cho từng bài posts, ta có thể tính điểm BA score cho các user trong 1 khoảng thời gian dễ dàng bằng aggregate command line như sau.

```
af_df = (r_tb
        .groupby(['to_user'])
        .agg({'AF score': 'mean', 'fid':'count'})
        .reset_index()
        .sort_values(['AF score'], ascending=False)
        .rename({'to_user': 'from_user',
        'fid': 'count', 'AF score': 'BA score'}, axis='columns'))


af_df.to_csv('af_scr.csv') #save AF score to csv file.
```