Report_Kenyan_Political_Tweets

December 4, 2018

Note: code is included if it prints an important output or if it is otherwise relevant to the report
Full code can be found at https://github.com/crazyfrogspb/kenya

1 User groups

First, we load the user data and create group mappings based on two TXT files

As it turns out, the followers of two accounts haven't been collected (@mavunochurchorg is suspended, @HouseofGraceHQ is just not in the dataset for some reason, can recollect if necessary)

```
In [7]: for account, group in account_mapping.items():
            if f'follows_{account}' not in users.columns:
                print(f'follows_{account} not in columns')
            else:
                users.loc[users[f'follows_{account}'] == 1, f'group_{group}'] = 1

        for word, group in contain_mapping.items():
            if f'{word}' not in users.columns:
                print(f'{word} not in columns')
            else:
                users.loc[users[f'{word}'] == 1, f'group_{group}'] = 1
```

```
follows_mavunochurchorg not in columns
follows_HouseofGraceHQ not in columns
```

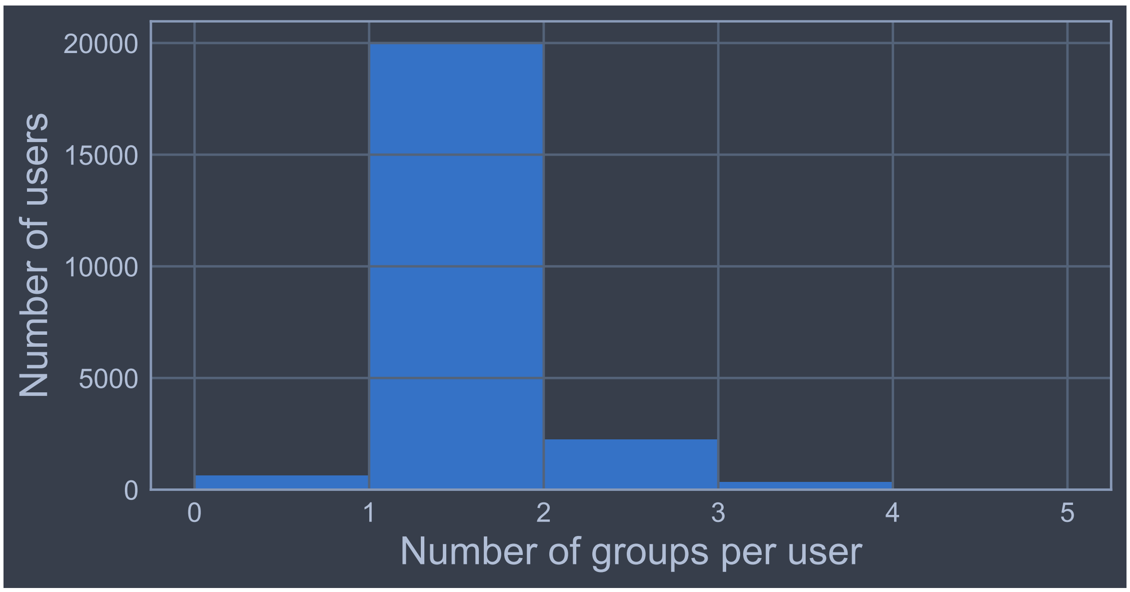
Over 85% of the users belong to single group according to our mapping

```
In [8]: group_cols = list(users.filter(regex='group_', axis=1).columns)
        users[group_cols] = users[group_cols].fillna(0)
        users['number_of_groups'] = users[group_cols].sum(axis=1).astype(int)
        print(users.groupby('number_of_groups').size())
        print(
            pd.DataFrame({
                'Percentage':
                users.groupby(('number_of_groups')).size() / len(users)
            }))
        fig, ax = plt.subplots(figsize=(8, 4))
        ax.set_xlabel('Number of groups per user')
        ax.set_ylabel('Number of users')
        users['number_of_groups'].hist(bins=[0, 1, 2, 3, 4, 5], ax=ax)
```

```
number_of_groups
0      649
1    19972
2     2251
3      355
```

```
4        3
dtype: int64
                  Percentage
number_of_groups
0                   0.027938
1                   0.859750
2                   0.096901
3                   0.015282
4                   0.000129
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f96b125c358>



Among users with multiple groups, the most popular combinations are Mainline Protestant + nan, Mainline Protestant + Pentecostal, Pentecostal + nan and Mainline Protestant + Pentecostal + nan

```
In [9]: users_multiple = users.loc[users.number_of_groups > 1, group_cols].copy()
        users_multiple.groupby(
            users_multiple.columns.tolist(),
            as_index=False).size().to_frame(name='users_num').sort_values(
                by='users_num', ascending=False)
```

Out[9]:

group_Mainline Protestant	group_Pentecostal	group_Catholic	group_nan	users_num
1.0	0.0	0.0	1.0	971
	1.0	0.0	0.0	746
0.0	1.0	0.0	1.0	479
1.0	1.0	0.0	1.0	344
	0.0	1.0	0.0	35
0.0	0.0	1.0	1.0	14
1.0	0.0	1.0	1.0	7
0.0	1.0	1.0	0.0	6

1.0	1.0	1.0	1.0	3
0.0	1.0	1.0	1.0	2
1.0	1.0	1.0	0.0	2

2 Tweets

For now, let's keep only users who belong to a single group. Let's also create user-group mappings and calculate the number of users by group. Note that there are a lot of nan (JCCKenya) users

```
In [11]: users_single = users.loc[users.number_of_groups == 1, :].copy()
         user_mapping = dict(
             zip(users_single['screen_name'], users_single[group_cols].idxmax(axis=1)))
         print(Counter(user_mapping.values()))
```

```
Counter({'group_nan': 8246, 'group_Mainline Protestant': 6262, 'group_Pentecostal':
5265, 'group_Catholic': 678})
```

Next, we load all tweets, remove duplicates and tweets that are out of time range; we also drop tweets for which we couldn't identify a single group

```
In [12]: tweets_file = os.path.join(DATA_DIR, 'processed', 'tweets.csv')
         tweets = pd.read_csv(
             tweets_file,
             lineterminator='\n',
             error_bad_lines=False,
             usecols=['created_at', 'screen_name', 'id', 'retweet', 'text'])

         tweets['created_at'] = pd.to_datetime(
             tweets['created_at'], infer_datetime_format=True, errors='coerce')
         tweets['mnth_yr'] = tweets['created_at'].dt.to_period('M')
         tweets = tweets.loc[(tweets.created_at >= '2013-03-01')
                             & (tweets.created_at < '2017-12-01')]

         tweets['id'] = pd.to_numeric(tweets['id'])
         tweets.drop_duplicates('id', inplace=True)

         tweets['group'] = tweets['screen_name'].map(user_mapping)
         tweets.dropna(subset=['group'], inplace=True)
```

```
/home/crazyfrogspb/.local/share/virtualenvs/kenya/lib/python3.6/site-
packages/IPython/core/interactiveshell.py:3020: DtypeWarning: Columns (1,3) have mixed
types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

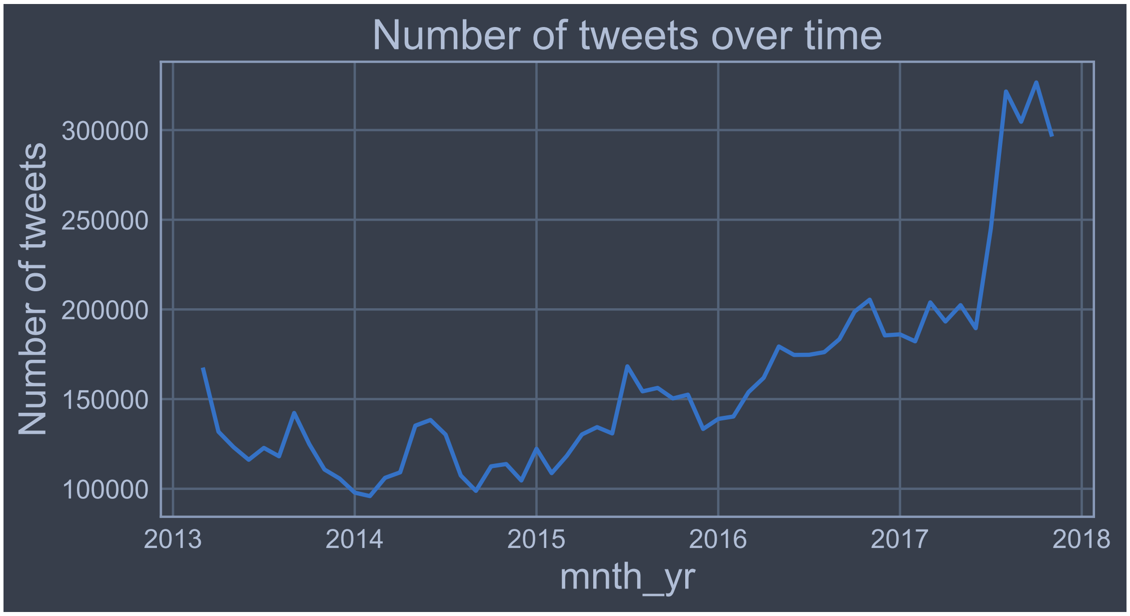
Some descriptive statistics and graphs

```
In [13]: print(f'Dataset size: {tweets.shape}')
         print(f'Unique users: {len(tweets.screen_name.unique())}')
         print(f"Number of tweets per group: {tweets.groupby('group').size()}")
```

```
Dataset size: (8997739, 7)
Unique users: 16190
Number of tweets per group: group
group_Catholic                303839
group_Mainline Protestant    3524303
group_Pentecostal            1923792
group_nan                    3245805
dtype: int64
```

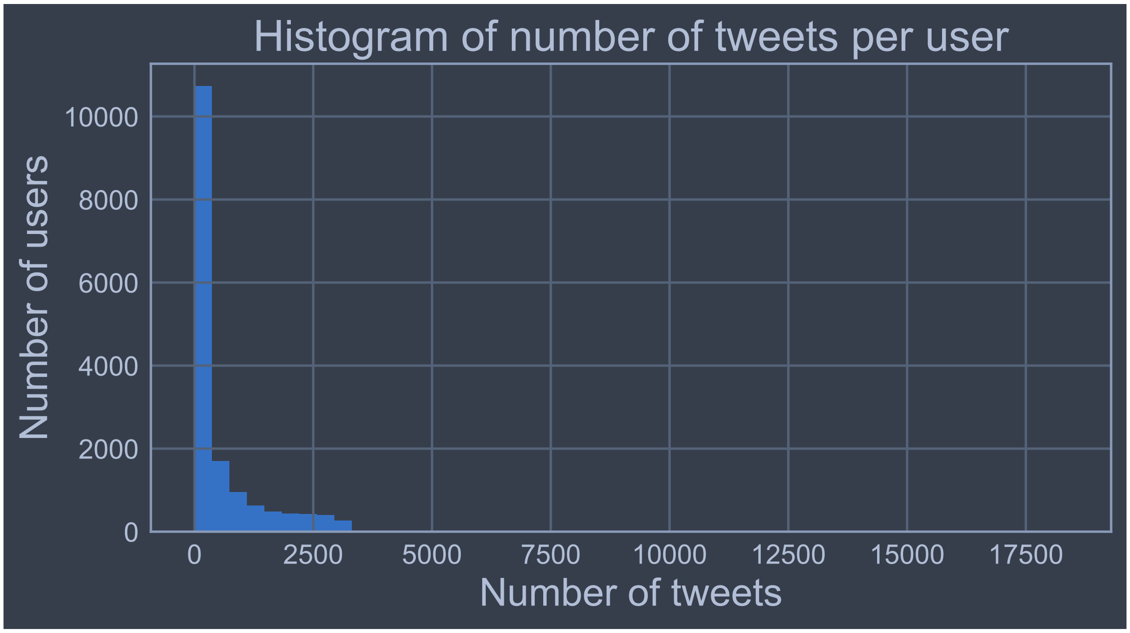
```
In [14]: fig, ax = plt.subplots(figsize=(8, 4))
         ax.set_xlabel('Month')
         ax.set_ylabel('Number of tweets')
         ax.set_title('Number of tweets over time')
         tweets.groupby('mnth_yr').size().plot(ax=ax)
```

```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f96b12f4438>
```




```
In [15]: fig, ax = plt.subplots(figsize=(8, 4))
         ax.set_xlabel('Number of tweets')
         ax.set_ylabel('Number of users')
         ax.set_title('Histogram of number of tweets per user')
         tweets.groupby('screen_name').size().hist(bins=50)
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f96b12a52b0>
```



2.1 Politicalness

Let's load Word2Vec model pretrained on the sample of 500000 Kenyan tweets and define functions that we need for calculating political scores. Here, we use snowballing technique: starting with a small sample of politicall related words, we then gradually add more words that are most similar to the current dictionary

Let's generate expanded list of political words

```
In [18]: political_words = [
             'politics', 'government', 'campaign', 'election', 'vote', 'ideology',
             'political', 'democracy', 'president', 'partisan', 'corruption', 'reform'
         ]
         political_words.extend(['kenyatta', 'uhuru', 'ruto', 'raila', 'odinga'])
         political_words_aug = snowball(w2v_model, political_words)
         print(political_words_aug)
```

```
/home/crazyfrogspb/.local/share/virtualenvs/kenya/lib/python3.6/site-
packages/gensim/matutils.py:737: FutureWarning: Conversion of the second argument of
issubdtype from `int` to `np.signedinteger` is deprecated. In future, it will be
treated as `np.int64 == np.dtype(int).type`.
  if np.issubdtype(vec.dtype, np.int):
```

```
['politics', 'government', 'campaign', 'election', 'vote', 'ideology', 'political',
'democracy', 'president', 'partisan', 'corruption', 'reform', 'kenyatta', 'uhuru',
'ruto', 'raila', 'odinga', 'presidency', 'opposition', 'coalition', 'referendum',
'handshake', 'regime', 'electorate', 'legitimacy', 'rao', 'uhuruto', 'majority',
'2022', 'rigging', 'despots', 'uthamaki', 'dictatorship', 'opposed', 'tribal',
'politicians', 'democratic', 'parties', 'sycophants', 'jubilee', 'odm', 'cord',
'intimidation', "gov't", 'rallies', 'oppose', 'nusu', 'illegitimate',
'unconstitutional', 'elites', 'dismissed', 'judiciary', 'electoral', 'legally',
'politically', 'negotiated', 'minority', 'hypocrisy', 'opposing', 'interference',
```

```
'allegations', 'impasse', 'threaten', 'incompetent', 'bribery', 'complicit',
'constitutional', 'intolerance', 'defended', 'squarely', 'incompetence', 'purported',
'purport', 'anarchy', 'jurisdiction', 'jsc', 'removal', 'suspend', 'outright',
'govt', 'stance', 'tribunal', 'prosecute', 'impeachment', 'parliamentary',
'fraudulent', 'advises', 'baseless', 'stalemate', 'rejecting', 'duplicitous',
'accuse', 'unfit', 'interfering', 'dissent', 'duress', 'irreducible', 'disbanded',
'scork', 'prosecution', 'reiterated', 'irregularities', 'illegalities',
'embarrassment', 'lapdogs', 'incitement', 'blatant', 'castigated', 'sharad',
'restructure', 'candidature', 'pnu', 'perennial', 'abuses', 'scrutinize', 'delegated',
'pseudo', 'undermining', 'jailing', 'aden', 'scotus', 'petitioners', 'guliye',
'abolish', 'subvert', 'dissidents', 'charade', 'malpractices', 'commision',
'outlawed', 'dismissal', 'revoked', 'caretaker', 'onslaught', 'repressive',
'prosecuting', 'politcal', 'disbandment', 'despotism', 'criminality', 'engineered',
'perpetuate', 'unions', 'condone', 'slogans', 'harrassing', 'mischief', 'agitating',
'violate', 'allied']
```

Now we load validation data and generate political scores
Correlation of political scores with coder_1 is 0.65, with coder_2 - 0.73

```
In [21]: validated_tweets.corr()
```

Out[21]:

	coder_2	id	coder_1	W2V_political
coder_2	1.000000	0.177303	0.617595	0.727474
id	0.177303	1.000000	0.210739	0.217393
coder_1	0.617595	0.210739	1.000000	0.659002
W2V_political	0.727474	0.217393	0.659002	1.000000

Let's choose threshold for political score that gives the highest accuracy (according to coder_2). We can achieve 87% accuracy. Precision is 90% (meaning that 90% of tweets identified as political are indeed political), recall is 81% (we can successfully identify 81% of political tweets from the validation dataset)

```
In [22]: best_acc = 0.0
         for t in np.arange(0.0, 1.01, 0.01):
             predictions = np.where(validated_tweets['W2V_political'] >= t, 1, 0)
             acc = metrics.accuracy_score(validated_tweets['coder_2'], predictions)
             if acc > best_acc:
                 best_t = t
                 best_acc = acc
         print(f'Best threshold: {best_t}, best accuracy: {round(best_acc, 3)}')
         predictions = np.where(validated_tweets['W2V_political'] >= best_t, 1, 0)
         precision = metrics.precision_score(validated_tweets['coder_2'], predictions)
         recall = metrics.recall_score(validated_tweets['coder_2'], predictions)
         print(f'Precision: {round(precision, 3)}, recall: {round(recall, 3)}')
```

```
Best threshold: 0.34, best accuracy: 0.868
Precision: 0.895, recall: 0.814
```

2.1.1 Analysis

Now we can apply these rules to all tweets to calculate different statistics for the different group of users

Let's look at some descriptive stats

```
In [27]: print(tweets['W2V_political'].describe().apply(lambda x: format(x, 'f')))
         print(tweets.groupby('is_political').size().to_frame(name='political_tweets'))
```

```
count    8710773.000000
mean           0.226505
std            0.201107
min           -0.466819
25%            0.089854
50%            0.201244
75%            0.339705
max            0.933720
Name: W2V_political, dtype: object
              political_tweets
is_political
0                      6823394
1                      2174345
```

Now we can finally look at the mean and median values of:

1) percentage of political tweets by group
2) political scores

```
In [28]: tweets.groupby('group')[['is_political', 'W2V_political']].agg(['mean', 'median'])
```

Out[28]:

group	is_political mean	is_political median	W2V_political mean	W2V_political median
group_Catholic	0.279237	0	0.247214	0.219846
group_Mainline Protestant	0.280528	0	0.247286	0.223539
group_Pentecostal	0.195158	0	0.203159	0.181588
group_nan	0.223486	0	0.215702	0.187788

Mainline Protesant and Catholic are significantly more political than Pentercostal group (can perform statistical tests if necessary)

We can also calculate mean values, but averaging on the user level first

```
In [29]: tweets.groupby(['screen_name', 'group'])[['is_political',
         'W2V_political']].mean().reset_index().groupby('group').mean()
```

Out[29]:

group	is_political	W2V_political
group_Catholic	0.270294	0.240474
group_Mainline Protestant	0.247475	0.231318
group_Pentecostal	0.193971	0.202099
group_nan	0.208377	0.208057

Same picture - Pentecostal group is the least politically active

What abouts absolute counts of tweets?

```
In [30]: tweets.groupby([
             'screen_name', 'group'
         ]).size().to_frame(name='num_tweets').reset_index().groupby('group').mean()
```

Out[30]:
```
                            num_tweets
group
group_Catholic              595.762745
group_Mainline Protestant   699.544065
group_Pentecostal           474.073928
group_nan                   492.983748
```

In [31]:
```
tweets.loc[tweets.is_political == 1].groupby([
        'screen_name', 'group'
    ]).size().to_frame(name='num_tweets').reset_index().groupby('group').mean()
```

Out[31]:
```
                            num_tweets
group
group_Catholic              184.041215
group_Mainline Protestant   214.740443
group_Pentecostal           111.407715
group_nan                   124.895489
```

Mainline Protestant are much more active in general, and tweet more about politics