Summary Statistics

CS109a: Fall 2018

Authors: Gordon Hew, Wenqin Hu, Blair Leduc

TF: Ken Arnold

Set up

Load spaCy, a natural language processing library

In [1]:
```
!pip install spacy
!python -m spacy download en_core_web_sm
```

Import common libraries that we will be using

In [2]:
```
import numpy as np
import pandas as pd
import spacy
import os
from pathlib import Path

pd.set_option('max_seq_items', 4000)
pd.set_option('max_rows',20)
```

File locations for input/output of this notebook

In [3]:
```
final_users_df_gz_file = os.path.join('tmp',
                                      'users_final_df.pkl.gz')
final_tweets_df_gz_file = os.path.join('tmp',
                                       'tweets_final_df.pkl.gz')
clean_tweets_df_gz_file = os.path.join('tmp',
                                       'tweets_clean_df.pkl.gz')
users_summary_df_gz_file = os.path.join('data',
                                        'users_final_agg_df.pkl.gz')
```

Load User and Tweet Dataframes

In [4]:
```
# Reload from binary file
users_pkl_df = pd.read_pickle(final_users_df_gz_file,
                              compression='gzip')
tweets_pkl_df = pd.read_pickle(final_tweets_df_gz_file,
                               compression='gzip')
```

In [5]:
```
# Make a copy here just incase we corrupt the datasets, we can
# start over quickly
users_df = users_pkl_df.copy(deep=True)
tweets_df = tweets_pkl_df.copy(deep=True)
```

Smoke test to make sure we loaded things correctly

In [6]:
```
print(f'Total number of users: {len(users_df)}')
unique_users = tweets_df.user_id.unique()
print(f'Number of unique users in tweets: {len(unique_users)}')
print(f"User {unique_users[0]} has " \
      + "{len(tweets_df[tweets_df['user_id'] == unique_users[0]])}")
```

```
Total number of users: 1000
Number of unique users in tweets: 1000
User 934576158305345536 has {len(tweets_df[tweets_df['user_id'] == uniq
ue_users[0]])}
```

Change the index of users_df to the unique id of the user

In [7]:
```
users_df = users_df.set_index('id')
```

Add columns to tweets_df that will speed up processing

In [8]:
```
if not Path(clean_tweets_df_gz_file).exists():
    tweets_df['clean_text'] = tweets_df['text']\
                                .apply(lambda x: ' '.join(\
                                   [t for t in x.split()
                                    if (t[0].isalpha() or t[0]=='(')
                                        and not t.lower()\
                                                    .startswith('http')
                                        and not t == 'RT']))

    tweets_df['created_at_hour'] = tweets_df['created_at']\
                                    .apply(lambda x: x[11:13])\
                                    .astype(int)
```

Natural Language Processing

Preprocess named entities once, here, for use in the rules to follow.

In [9]:
```
if not Path(clean_tweets_df_gz_file).exists():
    nlp = spacy.load('en_core_web_sm')
    # This column will have an list of named entities
    # We can filter on this later
    tweets_df['named_entities'] = tweets_df['clean_text']\
        .apply(lambda x: [e.label_ for e in nlp(x).ents])
```

Save clean and digested tweets_df to save time

Or load the previously saved DataFrame to save time

In [10]:
```
if not Path(clean_tweets_df_gz_file).exists():
    tweets_df.to_pickle(clean_tweets_df_gz_file,
                        compression='gzip')
else:
    tweets_df = pd.read_pickle(clean_tweets_df_gz_file,
                               compression='gzip')
```

Code to process tweets to and add metrics to users

In [11]:
```
class CollectSummaryMetrics:

    def __init__(self):
        self.columns = []

    def add_column(self, name, processor):
        self.columns.append({'name': name, 'func': processor})

    def run_processor(self, processor, s, df):
        for i in s.index:
            s[i] = processor(df[df['user_id'] == i])
        return s

    def run(self, users_df, tweets_df):
        df = users_df.copy()
        print('Processing:')
        unique_users = tweets_df.user_id.unique();
        for column in self.columns:
            print(f"Adding column {column['name']}...")

            df[column['name']] = self.run_processor(column['func'],
                        pd.Series(index = unique_users), tweets_df)
        print('Done.')
        return df
```

Create Summary Metrics

- tweets per hour ✓
- histogram array for tweets per hour ✓
- average number of links per Tweet ✓
- average number of contributors per Tweet ✗ (no contribs in any tweet)
- average tweet status word length per Tweet ✓
- average number of hashtags per Tweet ✓
- average user mentions per Tweet ✓
- average favorite count per Tweet ✓
- average media per Tweet ✓
- average symbols per Tweet ✓
- average retweet count per Tweet ✓
- average number of truncated tweets ✓
- Total links for each account per Pew reserach Category ✓
- Retweet ratio ✓
- Natural Language Processing columns (PERSON, NORP, ORG, GPE, PRODUCT, EVENT, LAW, MONEY) ✓

In [12]:
```
new_metrics = CollectSummaryMetrics()
```

Tweets per hour

In [13]:
```
def count_on_hour(df, hour):
    s = df['created_at_hour'].apply(lambda x: x[11:13]).astype(int)
    return s[s == hour].count()

for hour in range(0,24):
    new_metrics.add_column(f'tweets_per_hour_{hour:02}',
                           lambda df, hour=hour:
                               df['created_at_hour'][\
                                df['created_at_hour'] == hour].count())
```

Tweets per hour histogram array

In [14]:
```
new_metrics.add_column('tweets_per_hour',
                       lambda df:
                           np.histogram(df['created_at_hour'],
                                        range(0,24)))
```

Mean links per tweet

In [15]:
```
new_metrics.add_column('mean_links_per_tweet',
                       lambda df:
                           df['entities.urls'].apply(lambda x:
                                                     len(x)).mean())
```

Mean number of words per tweet

In [16]:
```
new_metrics.add_column('mean_words_per_tweet',
                       lambda df:
                           df['clean_text'].apply(lambda x:
                                                  len(x.split())).mean())
```

Mean number of hashtags per Tweet

In [17]:
```
new_metrics.add_column('mean_hashtags_per_tweet',
                       lambda df:
                           df['entities.hashtags'].apply(lambda x:
                                                         len(x)).mean())
```

Mean user mentions per Tweet

In [18]:
```
new_metrics.add_column('mean_user_mentions_per_tweet',
                       lambda df:
                           df['entities.user_mentions']\
                               .apply(lambda x:
                                      len(x)).mean())
```

Mean favorite count per Tweet

In [19]:
```
new_metrics.add_column('mean_favourites_per_tweet',
                       lambda df:
                           df['favorite_count'].mean())
```

Mean media per Tweet

In [20]:
```
new_metrics.add_column('mean_media_per_tweet',
                       lambda df:
                           df['entities.media'].apply(lambda x:
                                                      0 if x != x
                                                      else len(x)).mean())
```

Mean symbols per Tweet

In [21]:
```
new_metrics.add_column('mean_user_symbols_per_tweet',
                       lambda df:
                           df['entities.symbols'].apply(lambda x:
                                                        0 if x != x
                                                        else len(x)).mean())
```

Mean retweet count per tweet

In [22]:
```
new_metrics.add_column('mean_retweets_per_tweet',
                       lambda df: df['retweet_count'].mean())
```

Mean truncated text per tweets

In [23]:
```
new_metrics.add_column('mean_truncations_per_tweet',
                       lambda df:
                           df['truncated'].apply(lambda x:
                                                 1 if x else 0).mean())
```

Mean number of links per tweet source

In [24]:
```
sources = [c[9:] for c in tweets_df.columns if c.startswith('links_to')]

for source in sources:
    new_metrics.add_column(f'mean_links_to_{source}',
                           lambda df, source=source:
                               df[f'links_to_{source}'].mean())
```

Retweet ratio

In [25]:
```
new_metrics.add_column('retweet_ratio',
                       lambda df: df['text']\
                           .apply(lambda x:
                                  (1 if x.strip().startswith('RT @')
                                     else 0)).mean())
```

Natural Language Processing

We will collect statistics on these named entities:

- **PERSON**: People, including fictional.
- **NORP**: Nationalities or religious or political groups.
- **ORG**: Companies, agencies, institutions, etc.
- **GPE**: Countries, cities, states.
- **PRODUCT**: Objects, vehicles, foods, etc. (Not services.)
- **LAW**: Named documents made into laws.
- **MONEY**: Monetary values, including unit.

In [26]:
```
named_entities = ['PERSON', 'NORP', 'ORG', 'GPE',
                  'PRODUCT', 'LAW', 'MONEY']

for entity in named_entities:
    new_metrics.add_column(f'mean_ref_to_{entity.lower()}',
                           lambda df,entity=entity: df['named_entities']\
                               .apply(lambda x: x.count(entity)).mean())
```

Add Summary Metrics Columns

In [27]:
```
users_summary_df = new_metrics.run(users_df, tweets_df)
```

```
Processing:
Adding column tweets_per_hour_00...
Adding column tweets_per_hour_01...
Adding column tweets_per_hour_02...
Adding column tweets_per_hour_03...
Adding column tweets_per_hour_04...
Adding column tweets_per_hour_05...
Adding column tweets_per_hour_06...
Adding column tweets_per_hour_07...
Adding column tweets_per_hour_08...
Adding column tweets_per_hour_09...
Adding column tweets_per_hour_10...
Adding column tweets_per_hour_11...
Adding column tweets_per_hour_12...
Adding column tweets_per_hour_13...
Adding column tweets_per_hour_14...
Adding column tweets_per_hour_15...
Adding column tweets_per_hour_16...
Adding column tweets_per_hour_17...
Adding column tweets_per_hour_18...
Adding column tweets_per_hour_19...
Adding column tweets_per_hour_20...
Adding column tweets_per_hour_21...
Adding column tweets_per_hour_22...
Adding column tweets_per_hour_23...
Adding column tweets_per_hour...
Adding column mean_links_per_tweet...
Adding column mean_words_per_tweet...
Adding column mean_hashtags_per_tweet...
Adding column mean_user_mentions_per_tweet...
Adding column mean_favourites_per_tweet...
Adding column mean_media_per_tweet...
Adding column mean_user_symbols_per_tweet...
Adding column mean_retweets_per_tweet...
Adding column mean_truncations_per_tweet...
Adding column mean_links_to_twitter...
Adding column mean_links_to_top_social_media...
Adding column mean_links_to_top_digital_media...
Adding column mean_links_to_top_news...
Adding column mean_links_to_top_products_services...
Adding column mean_links_to_top_celebrities...
Adding column mean_links_to_top_organizations...
Adding column mean_links_to_top_sports...
Adding column mean_links_to_top_adult...
Adding column retweet_ratio...
Adding column mean_ref_to_person...
Adding column mean_ref_to_norp...
Adding column mean_ref_to_org...
Adding column mean_ref_to_gpe...
Adding column mean_ref_to_product...
Adding column mean_ref_to_law...
Adding column mean_ref_to_money...
Done.
```

Smoke test to check to see if columns added with correct content

In [28]:
```
users_summary_df.head()
```

Out[28]:

id	contributors_enabled	created_at	default_profile	default_profile_image	desc
934576158305345536	False	Sun Nov 26 00:14:54 +0000 2017	True	False	Unap adv comm cor
3133965632	False	Fri Apr 03 02:54:56 +0000 2015	True	False	def real wor n
3167730160	False	Wed Apr 15 00:56:50 +0000 2015	True	False	whe you'
893957155	False	Sat Oct 20 20:11:16 +0000 2012	False	False	i doo • Ni it's n
540553113	False	Fri Mar 30 02:47:09 +0000 2012	False	False	

5 rows × 95 columns

Save updated users' dataframe for the next step

In [29]:
```
users_summary_df.to_pickle(users_summary_df_gz_file, compression='gzip')
```

In [ ]:
```

```