ANALYZING SOCIAL MEDIA DATA IN PYTHON

Processing Twitter Text

Alex Hanna
Computational Social Scientist

Text in Twitter JSON

```
tweet_json = open('tweet-example.json', 'r').read()
tweet = json.loads(tweet_json)
tweet['text']
```

More than 140 characters

```
tweet['extended_tweet']['full_text']
```

Retweets and quoted tweets

```
tweet['quoted_status']['extended_tweet']['full_text']
```

Textual user information

```
tweet['user']['description']
tweet['user']['location']
```

Flattening Twitter JSON

```
extended_tweet['extended_tweet-full_text'] =
    extended_tweet['extended_tweet']['full_text']
```

Flattening Twitter JSON

```
tweet_list = []
with open('all_tweets.json', 'r') as fh:
    tweets_json = fh.read().split("\n")

    for tweet in tweets_json:
        tweet_obj = json.loads(tweet)

        if 'extended_tweet' in tweet_obj:
            tweet_obj['extended_tweet-full_text'] =
                tweet_obj['extended_tweet']['full_text']
        ...

    tweet_list.append(tweet)

tweets = pd.DataFrame(tweet_list)
```

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Counting words

Alex Hanna
Computational Social Scientist

Why count words?

- Basic step for automation of text analysis
- Can tell us how many times a relevant keyword is mentioned in documents in comparison to others
- In exercises: `#rstats` vs `#python`

Counting with str.contains

- `str.contains`
  - pandas `Series` string method
  - Returns boolean `Series`
  - `case = False` - Case insensitive search

Companies dataset

```
> import pandas as pd

> tweets = pd.DataFrame(flatten_tweets(companies_json))

> apple = tweets['text'].str.contains('apple', case = False)

> print(np.sum(apple) / tweets.shape[0])

0.112
```

Counting in multiple text fields

```
> apple = tweets['text'].str.contains('apple', case = False)

> for column in ['extended_tweet-full_text',
    'retweeted_status-text',
    'retweeted_status-extended_tweet-full_text']:
    apple = apple | tweets[column].str.contains('apple', case = False)

> print(np.sum(apple) / tweets.shape[0])

0.12866666666666668
```

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

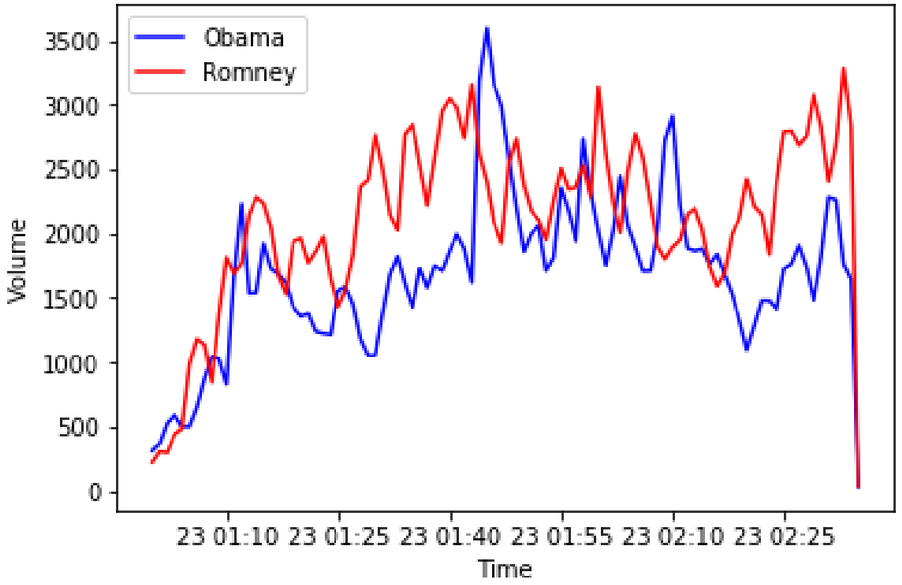
ANALYZING SOCIAL MEDIA DATA IN PYTHON

Time Series

Alex Hanna
Computational Social Scientist

Time series data

```
                     sum person
date
2012-10-23 01:00:00  314  Obama
2012-10-23 01:01:00  369  Obama
2012-10-23 01:02:00  527  Obama
2012-10-23 01:03:00  589  Obama
2012-10-23 01:04:00  501  Obama
...
```



Converting datetimes

```
> print(tweets['created_at'])

0       Sat Jan 27 18:36:21 +0000 2018
1       Sat Jan 27 18:24:02 +0000 2018
2       Sat Jan 27 18:09:14 +0000 2018
...

> tweets['created_at'] = pd.to_datetime(tweets['created_at'])

> print(tweets['created_at'])

0      2018-01-27 18:36:21
1      2018-01-27 18:24:02
2      2018-01-27 18:09:14
...

> tweets = tweets.set_index('created_at')
```

Keywords as time series metrics

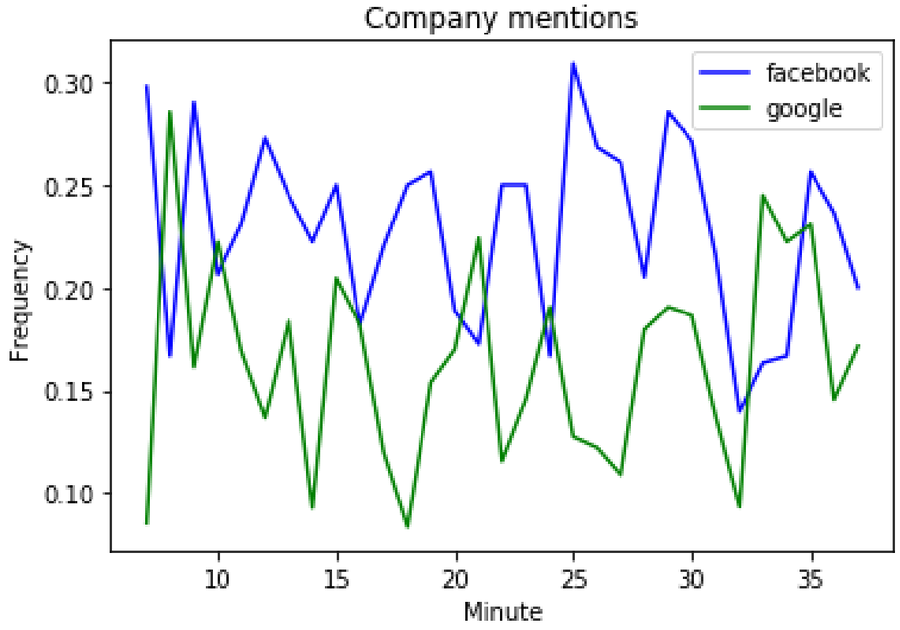
```
> tweets['google'] = check_word_in_tweet('google', tweets)

> print(tweets['google'])

created_at
2018-01-27 18:36:21    False
2018-01-27 18:24:02    False
2018-01-27 18:30:12    False
2018-01-27 18:12:37     True
2018-01-27 18:11:06     True
....

> print(np.sum(tweets['google']))

247
```

Generating keyword means

```
> mean_google = tweets['google'].resample('1 min').mean()

> print(mean_google)

created_at
2018-01-27 18:07:00    0.085106
2018-01-27 18:08:00    0.285714
2018-01-27 18:09:00    0.161290
2018-01-27 18:10:00    0.222222
2018-01-27 18:11:00    0.169231
```

Plotting keyword means

```
import matplotlib.pyplot as plt

plt.plot(means_facebook.index.minute,
    means_facebook, color = 'blue')
plt.plot(means_google.index.minute,
    means_google, color = 'green')
plt.xlabel('Minute')
plt.ylabel('Frequency')
plt.title('Company mentions')
plt.legend(('facebook', 'google'))
plt.show()
```



ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!

ANALYZING SOCIAL MEDIA DATA IN PYTHON

Sentiment Analysis

Alex Hanna
Computational Social Scientist

Understanding sentiment analysis

- Method
  - Counting positive/negative words in the document
  - Assessing positivity/negativity of the whole document
- Uses
  - Analyzing reactions to a company, product, politician, or policy

Sentiment analysis tools

- VADER `SentimentIntensityAnalyzer()`
  - Part of Natural Language Toolkit (`nltk`)
  - Good for short texts like tweets
  - Measures sentiment of particular words (e.g. angry, happy)
  - Also considers sentiment of emoji (☐☐) and capitalization (Nice vs NICE)

Implementing sentiment analysis

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()

sentiment_scores = tweets['text'].apply(sid.polarity_scores)
```

Interpreting sentiment scores

- Reading tweets as part of the process
  - Does it have *face validity*? (i.e. does this match my idea of what it means to be positive or negative?)

Interpreting sentiment scores

```
tweet1 = 'RT @jeffrey_heer: Thanks for inviting me, and thanks for the
lovely visualization of the talk! ...'
print(sid.polarity_scores(tweet1))

{'neg': 0.0, 'neu': 0.496, 'pos': 0.504, 'compound': 0.9041}
```

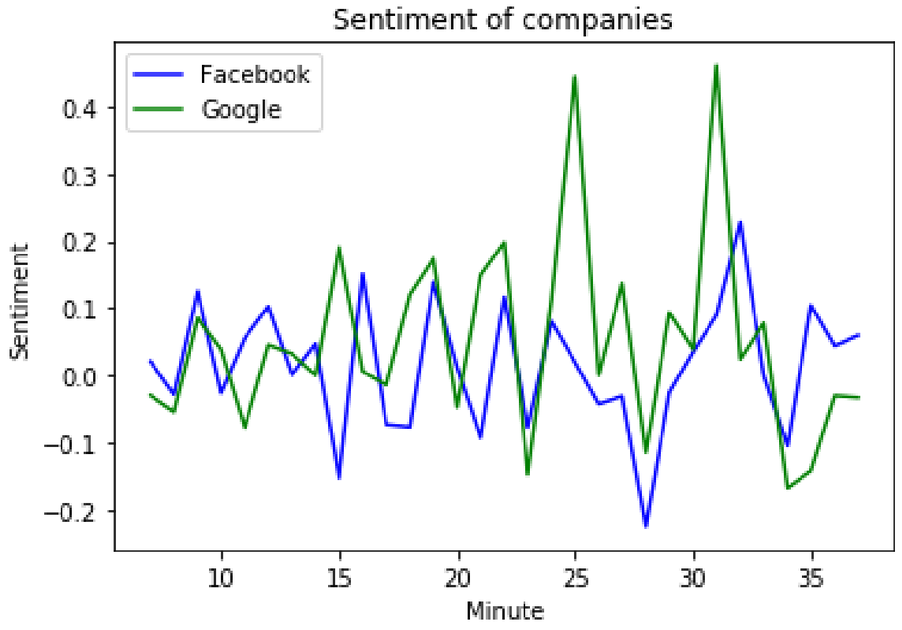
```
tweet2 = 'i am having problems with google play music'
print(sid.polarity_scores(tweet2)

{'neg': 0.267, 'neu': 0.495, 'pos': 0.238, 'compound': -0.0772}
```

Generating sentiment averages

```
sentiment = sentiment_scores.apply(lambda x: x['compound'])

sentiment_fb = sentiment[check_word_in_tweet('facebook', tweets)]
                    .resample('1 min').mean()
sentiment_gg = sentiment[check_word_in_tweet('google', tweets)]
                    .resample('1 min').mean()
```

Plotting sentiment scores

```
plt.plot(sentiment_fb.index.minute,
         sentiment_fb, color = 'blue')
plt.plot(sentiment_g.index.minute,
         sentiment_gg, color = 'green')

plt.xlabel('Minute')
plt.ylabel('Sentiment')
plt.title('Sentiment of companies')
plt.legend(('Facebook', 'Google'))
plt.show()
```



ANALYZING SOCIAL MEDIA DATA IN PYTHON

Let's practice!