Business Analytics Programming

Lab 3 (Twitter API & TextBlob - Analyze Tweets)

Dr. Wajahat Gilani

Rutgers Business School

March 14, 2019

User's Tweets

```
tjson=api.statuses.user_timeline(screen_name="realDonaldTrump
    ",tweet_mode='extended',count = 200)
dftrump=json_normalize(tjson)
dftrump.shape #(200, 303)
```

Even though the Twitter API allows us to get up 3200 tweets of a public user, the status call only allows us to receive up to 200 tweets at a time. The more recent 200 tweets will be given, but to get older tweets, we have to make use of another parameter, named *max_id*.

Each tweet has a unique id:

```
dftrump['id']
```

The id's are in chronological order, the higher the id, the newer the tweet.

```
mid = dftrump['id'].min()
```

The id value in mid is the oldest tweet out of those 200 tweets.

Get Previous Tweets

```
mid = dftrump['id'].min()
mid=mid-1
tjson2=api.statuses.user_timeline(screen_name="
    realDonaldTrump",tweet_mode='extended',count = 200,max_id
    = mid)
dftrump2=json_normalize(tjson2)
```

The *max_id* parameter gets the tweets with id's less than or EQUAL to the twitter id you provide in the parameter. That is why we decrease the mid variable by one value.

```
mid_l=dftrump2['id'].max()
```

The *mid_l* is less than *mid*.

Get Previous Tweets - Loop

```
df = pd.DataFrame()
mid=0
for i in range(34):
    if i==0:
        tjson=api.statuses.user_timeline(screen_name="
        realDonaldTrump",tweet_mode='extended',count = 200)
    else:
        tjson=api.statuses.user_timeline(screen_name="
        realDonaldTrump",tweet_mode='extended',count = 200,max_id
        = mid)
    if len(tjson)>0:
        dftrump=json_normalize(tjson)
        mid=dftrump['id'].min()
        mid=mid-1
        df = pd.concat([df, dftrump], ignore_index=True)
```

The first time we run the function without the max_id parameter, after that we use it. We also make use of the pandas concat() function.

Pandas Concat

```
a=pd.Series([1,2,3,4,5])
b=pd.Series([10,20,30,40,50])
c=pd.Series([6,7,8,9,10])
d=pd.Series([60,70,80,90,100])
```

	a		b		c		d
0	1	0	10	0	6	0	60
1	2	1	20	1	7	1	70
2	3	2	30	2	8	2	80
3	4	3	40	3	9	3	90
4	5	4	50	4	10	4	100

```
dfa=pd.DataFrame({'a':a,'b':b})
dfb=pd.DataFrame({'a':c,'b':d})
dfc=pd.concat([dfa,dfb])
dfd=pd.concat([dfa,dfb], ignore_index=True)
```

Pandas Concat - Continued

```
1 dfa=pd.DataFrame({'a':a,'b':b})
2 dfb=pd.DataFrame({'a':c,'b':d})
3 dfc=pd.concat([dfa,dfb])
4 dfd=pd.concat([dfa,dfb], ignore_index=True)
```

	a	b
0	1	10
1	2	20
2	3	30
3	4	40
4	5	50

	a	b
0	6	60
1	7	70
2	8	80
3	9	90
4	10	100

	a	b
0	1	10
1	2	20
2	3	30
3	4	40
4	5	50
0	6	60
1	7	70
2	8	80
3	9	90
4	10	100

	a	b
0	1	10
1	2	20
2	3	30
3	4	40
4	5	50
5	6	60
6	7	70
7	8	80
8	9	90
9	10	100

Get Previous Tweets - Loop

```
df = pd.DataFrame()
mid=0
for i in range(34):
  if i==0:
    tjson=api.statuses.user_timeline(screen_name="
    realDonaldTrump",tweet_mode='extended',count = 200)
  else:
    tjson=api.statuses.user_timeline(screen_name="
    realDonaldTrump",tweet_mode='extended',count = 200,max_id
    = mid)
  if len(tjson)>0:
    dftrump=json_normalize(tjson)
    mid=dftrump['id'].min()
    mid=mid-1
    df = pd.concat([df, dftrump], ignore_index=True)
```

We append dftrump to df every loop, dftrump takes on 200 rows at a time.

```
df.shape #(3194, 328)
```

Natural Language Processing (NLP)

- Natural Language Understanding
  - Taking some spoken/typed sentence and working out what it means
- Natural Language Generation
  - Taking some formal representation of what you want to say and working out a way to express it in a natural (human) language (e.g., English)

Applications of Natural Language Processing

- Machine Translation
- Database Access
- Information Retrieval
- Selecting from a set of documents the ones that are relevant to a query
- Text Categorization
- Sorting text into fixed topic categories
- Extracting data from text
- Converting unstructured text into structure data
- Spoken language control systems
- Spelling and grammar checkers

TextBlob

TextBlob is a Python (2 and 3) library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more. TextBlob objects can be treated as if they were Python strings that learned how to do Natural Language Processing. TextBlob heavily depends on Python NLTK and pattern module by CLIPS. Corpora used by NLTK is the default corpora for TextBlob as well.

Open python code file, textblob_code.py