IST-652

Homework2: Semi-Structured Data

Data and source

Twitter data was gathered using the Twitter Developer Account/Twitter API-gateway for the purpose of Homework-2, while working with semi-structured data. Due to API pull restrictions, the number of tweets for this exercise was limited to 5,000.

Further, the data was stored in a local running instance of MongoDB as below:

```
client = MongoClient('localhost', 27017)

# Create a db
db = client.hw2

# Drop a collection
for collection in db.list_collection_names():
    if 'tweet_info' == collection:
        print('Found collection {0}. Removing it!'.format(collection))
        db.tweet_info.drop()

# Create a collection/table
hw2_collection = db.tweet_info

# Use the structure below to put tweets into Mongo-DB
# Time, Tweet, User, Retweeted, Retweet-count
tweet_docs = []
for tweet in tweets:
    tweet_docs.append({'time': tweet['created_at'],
                       'tweet': tweet['text'],
                       'user': tweet['user']['name'],
                       'retweeted': tweet['retweeted'],
                       'retweet_count': tweet['retweet_count']})
```

Data exploration and clean-up

As a first step, the data pulled from Twitter was analyzed and a few key fields were chosen:

1. *created_at* (Type: Date-Time): The time at which the tweet was posted
2. *text* (Type: str): Content of the actual tweet
3. *name* (Type: str): Name of the user posting the tweet
4. *retweeted* (Type: Boolean): If the tweet was retweeted by other users
5. *retweet_count* (Type: Int): Count of the number of times a tweet was retweeted (also seen as *RT* on raw text obtained from Twitter)

A common step with text analysis or Natural Language Processing (NLP) is to remove frequently occurring words (also referred to as *stopwords*) that do not necessarily add weight to the content of sentence or in this instance, tweets and can therefore be deemed redundant.

The following methods were used to reduce the dataset of words:

- NLTK python module – *stopwords* was imported from the nltk.corpus module for popular words in languages English and Spanish (since some tweets in Spanish was also fetched).
- Using visual inspection some common words were further appended to this list

Finally, as part of the clean-up punctuation marks were also discounted.

```
# Initialize punctuations
punc = '''!()-[]{};:'"\, <>./?@#$%^&*_~'''

# Create a stop_words list for English, Spanish etc.
stop_words = stopwords.words('english')
stop_words.append(stopwords.words('spanish'))

# Add custom filters
stop_words.extend(['https', 'rt', 'covid', 'covid19', 'covid-19', '\'ve', '\'nt', 'n\'t',
                   'many', 'de', 'la'])

```

The result of removing the redundant words was efficient resulting in >50% reduction

```
for doc in docs:
    orig_sentence = doc['tweet']
    word_tokens = word_tokenize(orig_sentence)
    fil_sentence_1 = [w for w in word_tokens if w.isascii()]
    # Filter sentence based on stop_words
    fil_sentence_2 = [w for w in fil_sentence_1 if not w.lower() in stop_words]
    # Filter sentence based on punctuations
    fil_sentence_final = [w for w in fil_sentence_2 if not w in punc]
    all_words.extend(word_tokens)
    fil_words.extend(fil_sentence_final)

# Finally, walk and remove any word containing the link '//t.co'
for word in fil_words:
    if 't.co' in word:
        fil_words.remove(word)

print('Orig-words: {0} vs Filtered-words: {1} ({2}% reduction)'.format(len(all_
                                                                       len(fil_
                                                                       (len(all_
```

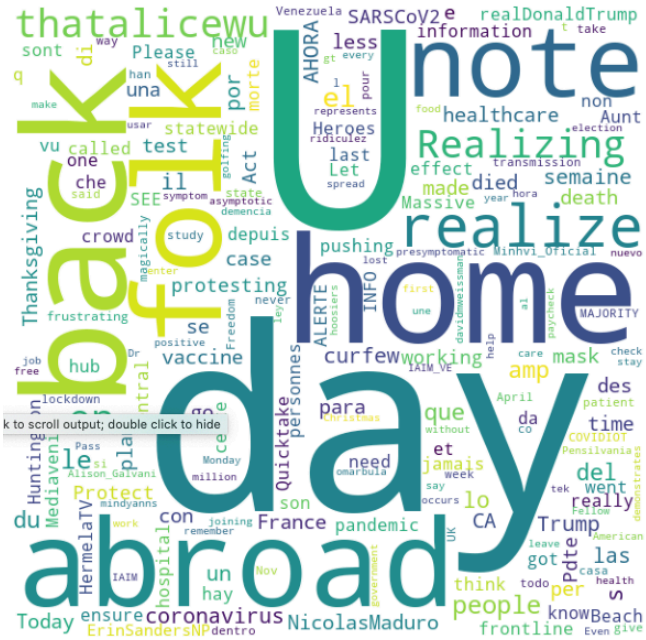
```
Orig-words: 130377 vs Filtered-words: 54920 (57.87600573720825% reduction)
```

Data Analysis

The data analysis was broken down into two parts. First, using a *WordCloud* determine popular topics and sentiment at the time the data was pulled. Second, using the data from Mongo-DB and *Pandas data-frames* draw conclusions using some data-frame operations.

1. Word-Cloud

Using the python WordCloud module along with the matplotlib, the following word-cloud was plotted:



Conclusion

Inferences from word-cloud:

- The trending topic at the time the tweets were pulled are around a topic originated by user '*thatalicewu*':

" A note from abroad: Realizing now that I've been 5 days out of US that many folks back home don't realize how other countr…""

- Correspondingly, we see words: *days, back, abroad, note, U (for USA) etc*. as the high frequency words

2. Data analysis

In this part, three broad questions were chosen:

1. **Determine time range of the tweets for the pulled data**
   Loading the data from Mongo-DB into a data-frame, the time range of tweets was determined as between *Nov-22 22:24:31 to Nov-23 01:41:50* using the head() and tail() functions on the data-frame.

```
# Create a pandas data-frame
df = pd.DataFrame(twitter_dict)

# Determine time-range of tweets
print('Most recent tweets: \n {0}'.format(df['Time'].head(1)[0]))
print('Oldest tweets: \n {0}'.format(df['Time'].tail(1)[max_results - 1]))

# The window of tweets is between Nov-22 22:24:31 to Nov-23 01:41:50

# Find the topic that's most re-tweeted/RT'ed
df[df['Retweet_Count'] == df['Retweet_Count'].max()]
```

```
Most recent tweets:
 Mon Nov 23 01:41:50 +0000 2020
Oldest tweets:
 Sun Nov 22 22:24:31 +0000 2020
```

2. Most tweeted/re-tweeted (RT) topic

Next, the popular topic at the hour (also seen from the word-cloud) based on the re-tweet counter was:

	Time	User	Tweet	Retweeted	Retweet_Count
4957	Sun Nov 22 22:25:39 +0000 2020	AleTheExpat	RT @thatalicewu: A note from abroad: Realizing...	False	11845
4969	Sun Nov 22 22:25:18 +0000 2020	RAIN	RT @thatalicewu: A note from abroad: Realizing...	False	11845
4972	Sun Nov 22 22:25:13 +0000 2020	Elizabeth Feifer	RT @thatalicewu: A note from abroad: Realizing...	False	11845
4985	Sun Nov 22 22:24:47 +0000 2020	NATL SIGNAL BOOST FOR BLUE GA SENATE RACES - #FBR	RT @thatalicewu: A note from abroad: Realizing...	False	11845
4987	Sun Nov 22 22:24:46 +0000 2020	Harriett Robinson	RT @thatalicewu: A note from abroad: Realizing...	False	11845
4988	Sun Nov 22 22:24:45 +0000 2020	Alanna Boudreau	RT @thatalicewu: A note from abroad: Realizing...	False	11845
4993	Sun Nov 22 22:24:39 +0000 2020	Cathipat Persist	RT @thatalicewu: A note from abroad: Realizing...	False	11845
4995	Sun Nov 22 22:24:38 +0000 2020	The CBDiva, from 6' away.	RT @thatalicewu: A note from abroad: Realizing...	False	11845

3. Most active user

Finally, the most active user in the dataset was determined as 'All Express News' at 67 tweets:

```
import operator

# Find most active user in the time window
active_users = {}

# Fetch all Mongo-db records
docs = hw2_collection.find()

for doc in docs:
    if doc['user'] in active_users:
        active_users[doc['user']] += 1
    else:
        active_users[doc['user']] = 1

sorted_active_users = max(active_users.items(), key=operator.itemgetter(1))
print('Most tweets by: {0} (count = {1})'.format(sorted_active_users[0], sorted_active_users[1]))
```

```
Most tweets by: All Express News (count = 67)
```

Conclusion

Using the techniques described the questions sought were answered translating data from the database into a common data-frame.

Final Conclusion

Text analysis or NLP is an interesting topic, and several techniques may be used to derive the overall sentiment to study trends.

In this example, data from a social networking hub (Twitter) was obtained around the topic of 'covid' to serve as semi-structured data. The data was then translated into a JSON-like body or a python dictionary before choosing to store certain parts of it in a non-SQL Mongo database. The data was then restored to python lists, dictionaries or Pandas data-frames to present analysis.

NOTE:
I hit some limitations with the number of entries I could pull from Twitter but if there was more data available one could also plot a comprehensive sentiment analysis at the time.

Description of code

The code was written in entirety in Jupyter Notebook and the corresponding file (.ipynb) is being attached as part of the deliverable.

Libraries or packages: *tweepy* to authenticate and pull data from Twitter, *pymongo* to connect to local database instance of Mongo-db to save and retrieve data, *nltk* for language processing, *Pandas* for data-analysis and *matplotlib* for visualization.

The code is broadly organized into sections that pull data from Twitter, save it to a database and load it back into python data-structures for analysis.