Web scraping and social media scraping – APIs

Jacek Lewkowicz, Dorota Celińska-Kopczyńska

University of Warsaw

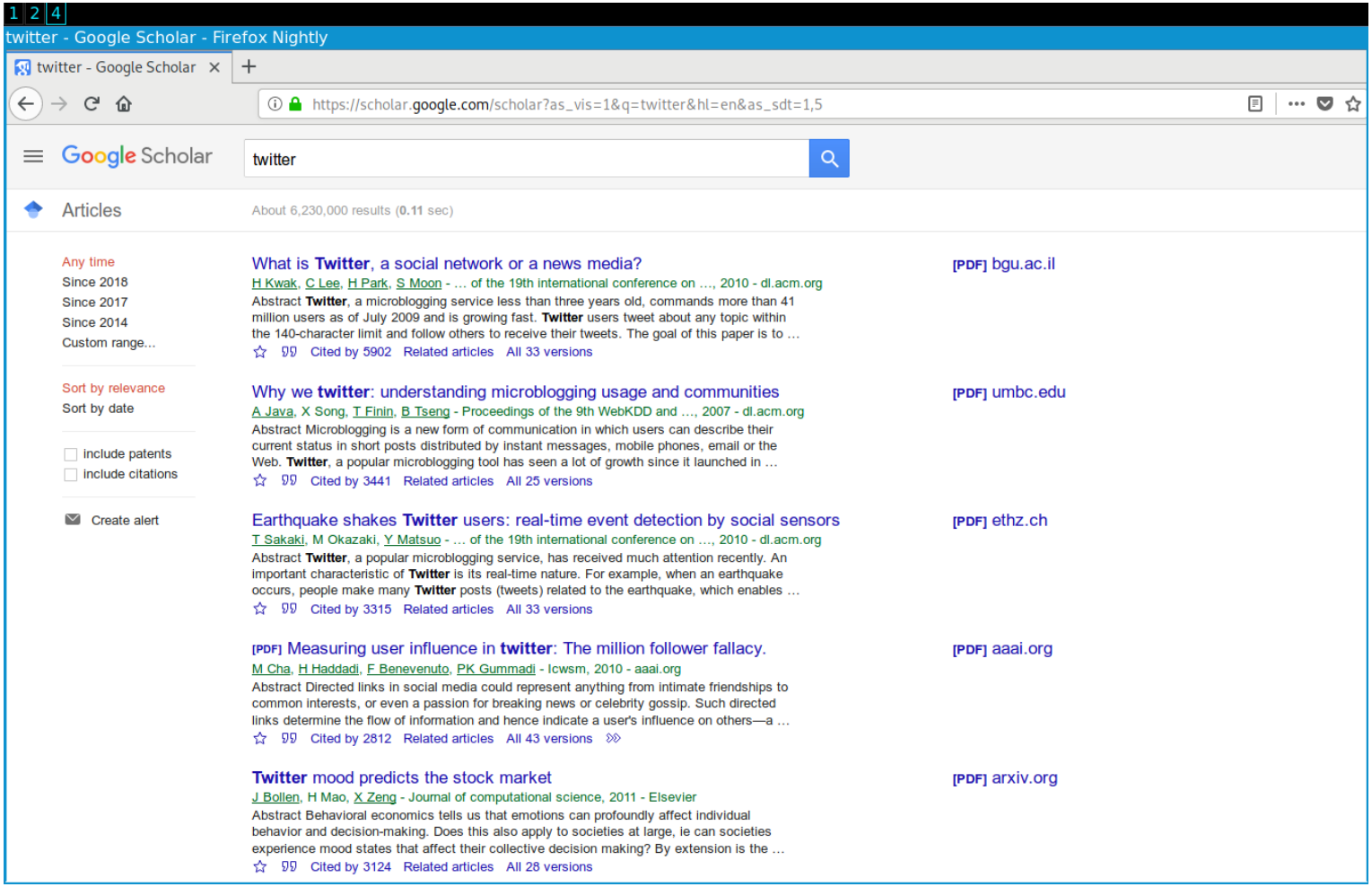
April 14, 2019

Internet as a source of data for research

- Imagine a highly popular website, e.g., Twitter, Reddit, etc
- There are millions of users, myriads of activities, tons of text messages...
- Which seems not only to be a great source of data for the needs of your master or doctoral thesis, but if utilized correctly might even earn you a tenure!
- The sea of immeasurable gains lies at the top of your fingers... just scrape the data!

Problem #1

- The problem is... it is not only you who had the same idea



Problem #2

- ... and sometimes "publish or perish" is really innocent
- Cambridge Analytica issue – the company is accused of buying millions of Americans' data from a researcher who told Facebook he was collecting it strictly for academic purposes
- Which generally made accessing social media data real pain...

Solution

- In theory, the site admins should be prepared not only for the typical activity of the users but also for the activity of third parties that want to get access to the data
- In such cases providing and using APIs may save a lot of time (and trouble)

What is API?

- An **Application Programming Interface** is a set of subroutine definitions, protocols and tools for building application software
- It should make communication between various software components easier
- In web scraping usage of APIs usually boils down to sending requests messages and getting response messages usually in XML or JSON format
- Example:
  `https://console.developers.google.com/apis/dashboard`

Data from APIs

- APIs encompass tools which enable programmers to connect their software with "something else"
- They are useful in programming software that relies on external soft- or hardware because the developers do not have to go into the details of external soft- or hardware mechanics

Scheme of usage

- The API provider sets up a service that grants access to data from the application or to the application itself
- The API user accesses the API to gather data or communicate with the application
- It may be necessary to write wrapper software for convenient data exchange with the web service

Common commands

- `GET` – visiting a website through the address bar in the browser
- `POST` – filling out a form/submitting information to a backend script on the server
- `PUT` – updating an object or information
- `DELETE` – deleting an object

Authentication

- Services may impose some restrictions in accessing data if the user is not authenticated
- The methods of authentication discussed earlier still hold
- If you (for some reason) provide sensitive data, remember to use, e.g., environment variables (or any other "relatively safe" technique)

```
token = "<your api key>"
webRequest = urllib.request.Request("http://myapi.com", headers={"token":token})
html = urlopen(webRequest)
```

Why is it interesting?

- Lots of active users
- A lot of topics: from public debate and political issues to breakups...
- Fast and easy access

Terms of Use and available information

- `http://developer.twitter.com/en/docs/`
- Account Activity API: manage account settings and interactions with users (e.g., follow, mute, search users)
- Post, retrieve and engage with tweets
- Direct Message API, Media and Trends

Authentication

- Twitter refers to an open standard of authentication that permits connections without sharing username/password
- All queries require a valid OAuth token
  - New application on dev.twitter.com or apps.twitter.com/app/new
  - Sign in and save your consumer key and consumer secret for future sessions
  - Python twitter tools: `http://mike.verdone.ca/twitter/#downloads`

Twitter: an example

- We will search for tweets containing hashtag `python`
- Start with navigating to the directory in which you store twitter tools
- `$ python setup.py install`

```
# import the library
from twitter import Twitter

# authenticate
t = Twitter(auth=OAuth(<Access Token>,<Access Token Secret>, <Consumer Key>,<Consumer Secret>))

pythonTweets = t.search.tweets(q = "#python")
print(pythonTweets)
```

Twitter: an example

- We will retrieve the list of tweets by a given user
- For more examples see also
  `http://github.com/sixohsix/twitter/tree/master`

```
# import the library
from twitter import Twitter

# authenticate
t = Twitter(auth=OAuth(<Access Token>,<Access Token Secret>, <Consumer Key>,<Consumer Secret>))

pythonStatuses = t.statuses.user_timeline(screen_name="POTUS", count=10)
print(pythonStatuses)
```

Why is it interesting?

- Lots of active users
- Differently themed boards, popular and less well-known ones
- Availability of searching for a particular user's activity in many boards

Terms of Use and available information

- `https://www.reddit.com/dev/api/`
- Everything a user can do (for your account)
- Getting comments, titles, overviews, upvotes and downvotes...

Authentication

- `https://github.com/reddit-archive/reddit/wiki/API`
- Clients must authenticate with a valid OAuth2 token
  - Details described here: `https://github.com/reddit-archive/reddit/wiki/OAuth2`

Reddit: an example

- We will search for posts and comments in subreddit `boardgames`

```
# import necessary libraries
import urllib3
import praw

urllib3.disable_warnings()

# provide authentication
r = praw.Reddit(user_agent='insert-here', client_id='your-id', client_secret='your-secret', redirect_uri='your-site',
username='your-login')

submissions = r.subreddit('boardgames').hot(limit=None)

for x in submissions:
  print("SUBMISSION BY "+str(x.author))
  print("ID "+str(x.id))
  print("TITLE "+x.title.encode('utf-8'))
  print("SCORE "+str(x.score))
  print("DATE "+str(x.created_utc))
  print(x.selftext.encode('utf-8'))
  # print(vars(x))
  print("")
  x.comments.replace_more(limit=None, threshold=0)
  comment_queue = x.comments[:]  # Seed with top-level
  while comment_queue:
    comment = comment_queue.pop(0)
    print("COMMENT BY "+str(comment.author))
    print("ID "+str(comment.id))
    print("PARENTID "+str(comment.parent_id))
    print("DATE "+str(comment.created_utc))
    print(comment.body.encode('utf-8'))
    # print(vars(comment))
    print("")
    comment_queue.extend(comment.replies)
```