

Lecture 1b

Web and Twitter Mining

Part I – Web Crawling

- Web crawling is easy in Python.
- We'll use a package called BeautifulSoup
- If you don't have BeautifulSoup use `easy_install beautifulsoup4` (from a DOS prompt)
- If using Linux or Macintosh google how to install BeautifulSoup for those operating systems
- An overview of BeautifulSoup can be found here

<http://www.pythonforbeginners.com/beautifulsoup/web-scraping-with-beautifulsoup>

A Soup Session

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> from bs4 import BeautifulSoup
>>> import requests
>>> r=requests.get("http://rowan.edu")
>>> data = r.text
>>> soup=BeautifulSoup(data,'lxml')
>>> for link in soup.find_all('a'):
>>>     print(link.get('href'))

#main-content
http://rowan.edu/home
http://rowan.edu/home/admissions-aid/undergraduate-admissions/applications
/home/admissions-aid
/home/undergraduate-admissions
/home/financial-aid
http://global.rowanu.com
/home/admissions-aid/medical-education-admissions
/home/admissions-aid/international-admissions
/home/admissions-aid/transfer-students
/home/admissions-aid/cost-value
/home/academics
/home/academics/colleges-schools
/home/academics/degrees-programs
/home/academics/courses-schedules-registration
/home/academics/study-abroad
http://www.rowan.edu/open/summer/home/
/home/academics/honors-program
```

A Couple of Items to Notice

- The session on the previous page didn't actually crawl the web. It listed a bunch of links that could be found on the main page of www.Rowan.edu
- An actual web crawler recursively visits all of the links and accumulates new links while keeping track of links that have already been visited
- A web crawler can accumulate tens or hundreds of thousands of web pages if given the right 'seed' page
- Note also that `link.get('href')` does not give a full path. (e.g. `home/admissions-aid`). We have to keep track of the fact that the first part of the address is `http://rowan.edu/`

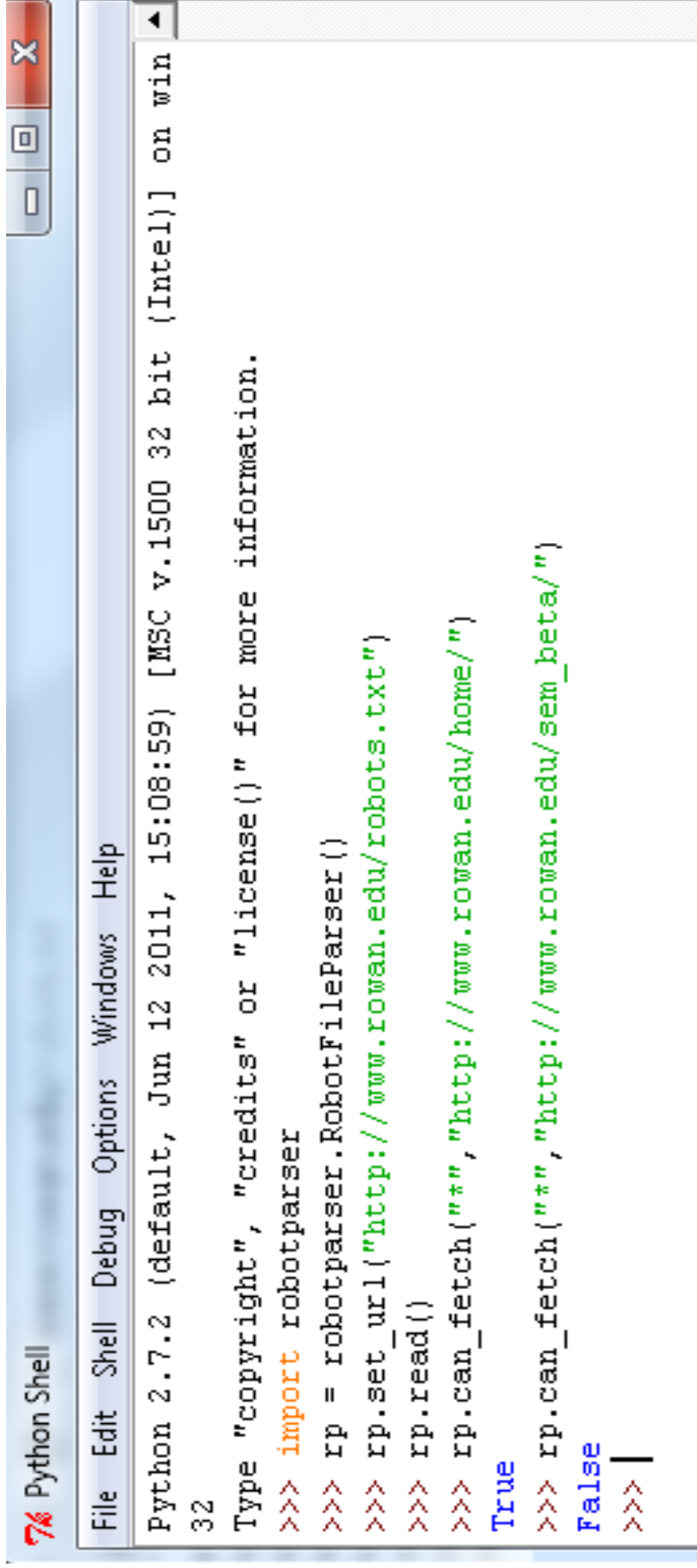
Crawling Etiquette

- When crawling a website, check its robots.txt file for direction
- Here is Rowan's <http://rowan.edu/robots.txt>

```
User-agent: *  
Disallow: /_mm/  
Disallow: /_notes/  
Disallow: /_baks/  
Disallow: /MMWIP/  
  
User-agent: googlebot  
Disallow: *.csi
```

- This suggests that all but 6 subdirectories are open to crawlers and the server would like a 10 second delay between requests
- Python makes it easy to check a robots.txt file and see if you have permission to crawl a particular directory
- See <https://docs.python.org/2/library/robotparser.html>

Using RobotParser



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import robotparser
>>> rp = robotparser.RobotFileParser()
>>> rp.set_url("http://www.rowan.edu/robots.txt")
>>> rp.read()
>>> rp.can_fetch("http://www.rowan.edu/home/")
True
>>> rp.can_fetch("http://www.rowan.edu/sem_beta/")
False
>>> |
```

Putting It All Together

- To build a crawler we need:
 1. A way to get to all links on a page (Beautiful Soup)
 2. A way to check to see if a link is allowed (RobotParser)
 3. A way to recursively visit each link and make a list of all links we need to visit and all the ones we have to visit yet
- I've posted a crawler on blackboard: `crawler.py`

Crawler.py

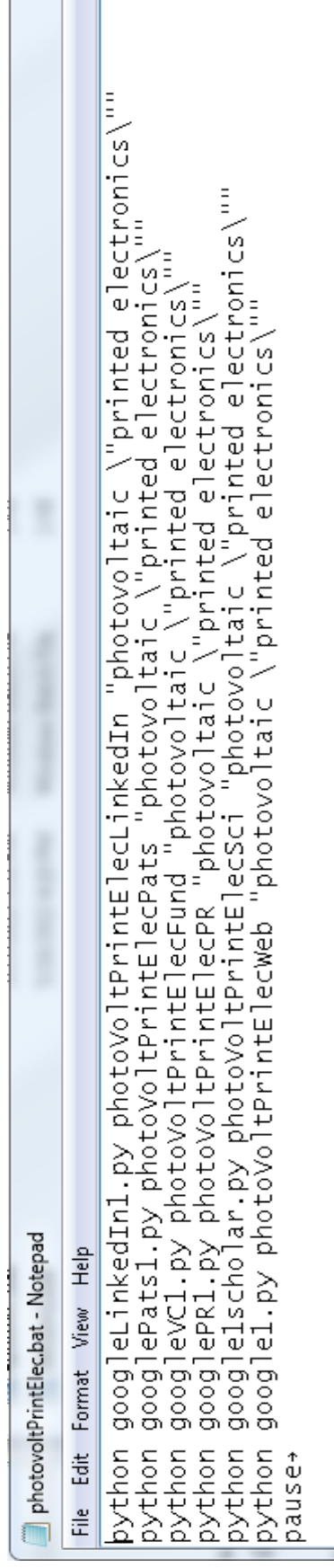
- Uses multiple python lists
 - tobecrawled is a list that acts like a queue
 - crawled is a list of links that have been visited
- The seed site is visited and all of the links are found via soup and added to tobecrawled
- The first site is then visited, removed from tobecrawled and all of the links on that site are added to tobecrawled.
- Before each visit robotParser is used to see if the site is allowed
- A timer is used to sleep for 10 seconds in between site visits so we don't overtax the server and anger the webmaster.

Spider Traps

- You should be aware that many sites do not like crawlers/spiders
- If you see a robots.txt file and it has reasonable restrictions than you are welcome to crawl it provided you stay out of the excluded areas
- If you do not see a robots.txt file and you crawl it anyway, you may hit a spider trap
- The typical spider trap will cause your crawler to repeatedly visit a page that is not visible to human users. (You will then likely get blacklisted by the site) A good way to avoid this is to keep a list of visited pages and compare it with the next page you visit.
- You can google 'spider traps' to find out more information if you intend to use a crawler a lot
- Another way is to avoid crawling altogether and use a google API

Google Custom Searches

- I've stored a bunch of python programs on Blackboard for doing custom google searches
- The batch file below searches for the terms 'photovoltaic' and 'printed electronics' in LinkedIn, GooglePats, Press Releases, GoogleScholar, and regular Google.
- First we'll look at some output and then we'll look at the code



```
photovoltPrintElec.bat - Notepad
File Edit Format View Help
python googleLinkedIn1.py photovoltPrintElecLinkedIn "photovoltaic \printed electronics\'"
python googlePats1.py photovoltPrintElecPats "photovoltaic \printed electronics\'"
python googleVCI1.py photovoltPrintElecFund "photovoltaic \printed electronics\'"
python googlePR1.py photovoltPrintElecPR "photovoltaic \printed electronics\'"
python googleScholar.py photovoltPrintElecSci "photovoltaic \printed electronics\'"
python google1.py photovoltPrintElecWeb "photovoltaic \printed electronics\'"
pause
```

What do you need?

- You need a google+ account (free)
- You need a custom search API key (free)
https://developers.google.com/custom-search/json-api/v1/overview?hl=en_US
- You need to create a custom search engine or two (free)
- You are limited to 100 free queries per day

My custom Search Engine Specs

- Name: googlePat
- Built in keywords: none
- Search Only Included Sites: www.google.com/patents
- Name: linkedIn
- Built in keywords: none
- Search Only Included Sites: linkedIn.com
- Name: PressRelease
- Built in keywords: "press release" or "news release"
- Search Entire Web but emphasize these sites: prnewswire.com; press-news.org
- Name: general
- Built in keywords: none
- Search Entire Web
- Name: Science
- Built in keywords: none
- Search Only Included Sites: <http://citeseerx.ist.psu.edu/>; <http://ieeexplore.ieee.org>; www.ncbi.nlm.nih.gov; <http://scholar.google.com>; www.sciencedirect.com;

Twitter

- A number of twitter archives exist. Most cost \$\$
- There is a twitter API that allows searching of keywords or hashtags
 - Limitation is that it only goes back 8-10 days
 - You also need to set up things to get API keys which is a bit of a pain
- You can also search by user. These go back more than 10 days
- Most searching is rate limited to 180 queries every 15 minutes (which doesn't sound like a limitation, but I've hit it often)

Twitter (2)

- I've uploaded 5 twitter files for you to look at
- Config.py needs to be filled with your API keys
- @IEEESpectrum.txt several year's worth of tweets from IEEE Spectrum Magazine
- fetch@.bat gets tweets from multiple users
- fetch_@tweets.py is the python program used to identify tweets from @IEEESpectrum among others
- nano.txt contains 6 months of tweets from 2013 that mention 'nanoparticle'
- keywprds.bat searches a number of keywords
- fetch_searchtweets.py is the python program used to search keywords
- Rate_limit.py shows how many queries you have left in the current 15 minute period

Twitter (3)

- These programs were originally written for a project I had in 2012-2013
- I modified them in the last 2 days to use Twitter API v1.1
- Big difference is you now need API keys <https://themepacific.com/how-to-generate-api-key-consumer-token-access-key-for-twitter-oauth/994/>
- Important – (this cost me a couple of hours) – before generating the access tokens click settings and change the Application Type to "Read, Write and Access direct messages"

Twitter (4)

- I have not tried this, but Mitch Jablonski says it works and it's easier than dealing with my Twitter programs

GitHub - kennethreitz/twitter-scraper: Scrape the Twitter Frontend API ...

<https://github.com/kennethreitz/twitter-scraper> ▼

README.rst. Twitter Scraper. Twitter's API is annoying to work with, and has lots of limitations — luckily their frontend (JavaScript) has it's own API, which I ...

People also search for

twitter scraper *r* twitter link scraper

tweep github tweet scraper jonbakerfish

scraper github python selenium twitter

×

Homework

- Due July 5 (You may work in teams)
- Choose **Only 1** of the following
 1. Create an API Key for Google+ and get one of my programs working on your system to dump a bunch of web sites for some search term
 2. Write something to do the same for Bing or an alternative search site
- Choose **Only 1** of the following
 1. Download the twitter API keys and use one of my programs to download all of the tweets related to a keyword, hashtag, or screen name.
 2. Use the API to do something completely different like:
 - a) Given a screen name, list all of its followers
 - b) Find the trending topics
 - c) Given a tweet, find all the times it has been retweeted
 - d) Anything else, as long as it uses the API