Introduction to Social Media Analytics (Lec 2)

Hao PENG
Department of Data Science
City University of Hong Kong
https://haoopeng.github.io/

Agenda for Week 2

- Python refreshments
- Data collection, APIs
- Requests, BS4, Json
- Data preprocessing
- Paper presentation

- Homework 1 will be released in Week 4 (Sep 26)!
  - Due in three weeks (end of week 7; after midterm)
  - Hw 2 & 3 will be due in two weeks after release.

If you need python material

Check out Chuck Severence’s **Python for Informatics**:

https://do1.dr-chuck.net/py4inf/EN-us/book.pdf

Basic python data structures

The key to fast, flexible manipulation of data

List: `l = [1,2,3,"a","b"]`

Tuple: `t = (1,2,3,"a","b")`

Dict: `d = {"a":1, "b":2, "c":"hello"}`

Set `s = {1, 2, "hello"}`

List vs Tuple vs Dict vs Set

List: **ordered sequence** of items, is mutable

tuple is like list, but immutable

dict and set are **unordered**

Dict: associates each unique **key** with a **value**

list and set just contain **values**

Set: **unordered set** of unique items; is mutable

list/tuple allow duplicates, are ordered

Compound data structures

We can create combined data structures!

- **list** where each element is a **dictionary** or **set**
- **dictionary** where each key is a **tuple** or **set**
- Etc.

Makes more efficient data structures and algorithms

List, Set, Dict comprehensions

Comprehension syntax parallels definition syntax

Definition syntax		**Comprehension syntax**
List:	`[0, 1, 4, 9]`	`[x**2     for x in range(4)]`
Dict:	`{0: 0, 1: 1, 2: 4, 3: 9}`	`{x: x**2 for x in range(4)}`
Set:	`{0, 1, 4, 9}`	`{x**2     for x in range(4)}`

Sorting lists

`sorted(l)`

- Takes a list `l`
- Returns a new, sorted list
- Does not change input list

- http://docs.python.org/3/library/functions.html#sorted
- http://docs.python.org/3/howto/sorting.html

Custom sorting with key=...

```
>>> sorted("This is a test string from Andrew".split())
```

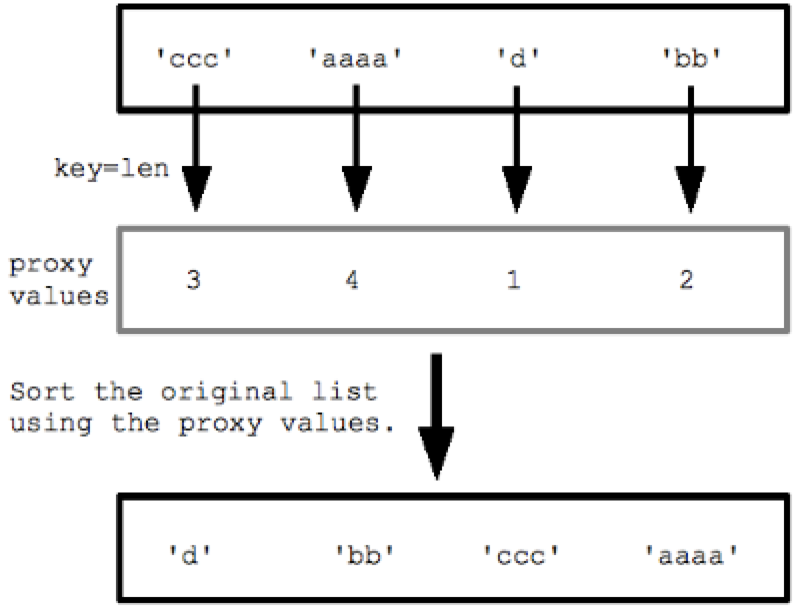
value of the key parameter is a **function:**

- key **function** must take a single argument, which is applied to each of the elements being sorted (e.g. each word above)
- key **function** could be a *lambda* (customized) function

  (e.g. key = lambda x: x[-1])
- key **function** output is used as the sort key
  (e.g. str.lower("Andrew") returns "andrew", ...)

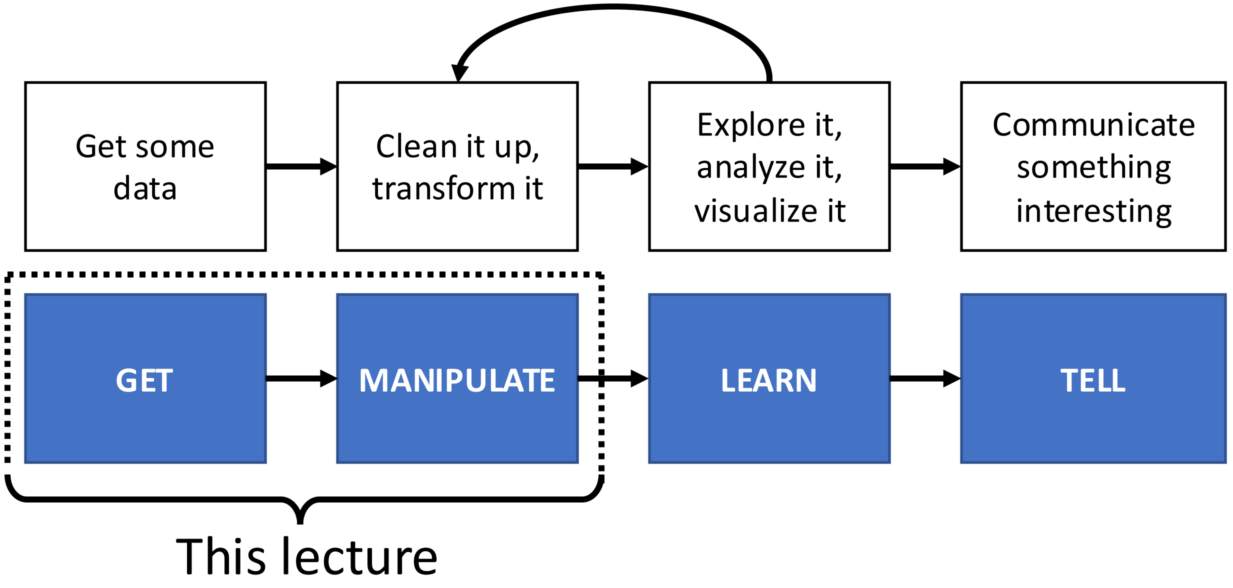
Custom sorting with key=...

```
strs = ['ccc', 'aaaa', 'd', 'bb']
print sorted(strs, key=len)  ## ['d', 'bb', 'ccc', 'aaaa']
```



Before you can analyze data or make fancy visualizations….

You need some data to analyze / visualize!


Get some data
Clean it up, transform it
Explore it, analyze it, visualize it
Communicate something interesting
GET
MANIPULATE
LEARN
TELL
This lecture

Data processing is crucial step

- Roughly 50-80% of initial work of a data science project involves data processing:
  - accessing, collecting, cleaning, converting, transforming, filtering, aggregating, grouping, summarizing data
- SMA is tightly coupled with data processing:
  - Not just in the initial stages, but when iterating with exploration, model building, evaluation…

Social media data is ubiquitous

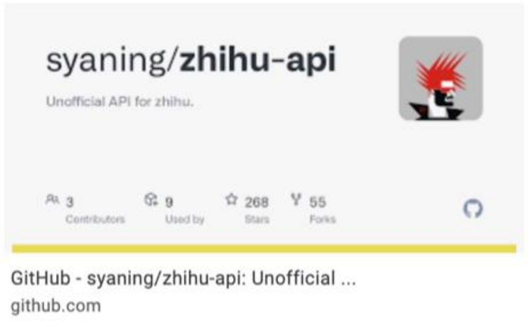
- Data is central to organizations and decision-makings.
- There is a huge volume of data around us! (http://www.internetlivestats.com/)
- Thousands of APIs and web services
  - Twitter, Facebook, Google search, LLMs (Grok, etc.)
  - > 16,000 APIs (programmableweb.com)

How to collect social media data?

- API is the easiest (first and third-party APIs)
- Build customized web crawler (parse data in html)
- Browser-like crawler (e.g., Selenium, LLM agent)

Twitter API

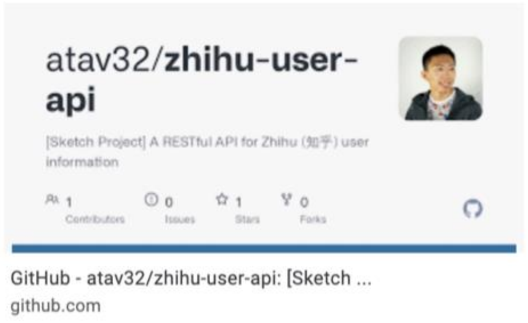
facebook DEVELOPER


syaning/**zhihu-api**

Unofficial API for zhihu.

3 Contributors · 9 Used by · 268 Stars · 55 Forks

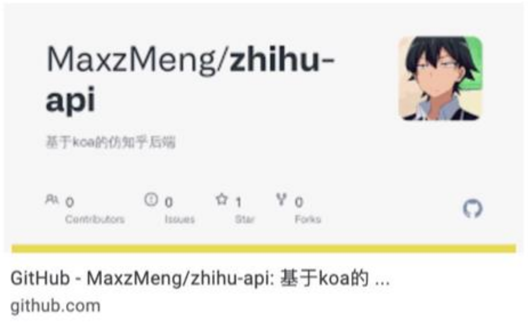
GitHub - syaning/zhihu-api: Unofficial ...
github.com


atav32/**zhihu-user-api**

[Sketch Project] A RESTful API for Zhihu (知乎) user information

1 Contributors · 0 Issues · 1 Stars · 0 Forks

GitHub - atav32/zhihu-user-api: [Sketch ...
github.com


MaxzMeng/**zhihu-api**

基于koa的仿知乎后端

0 Contributors · 0 Issues · 1 Star · 0 Forks

GitHub - MaxzMeng/zhihu-api: 基于koa的 ...
github.com

What’s an API

An Application Programming Interface (API) is a set of rules and protocols that allows different software applications to communicate with each other. APIs define the methods & data formats that applications can use to request & exchange information.

Two basic things that APIs do

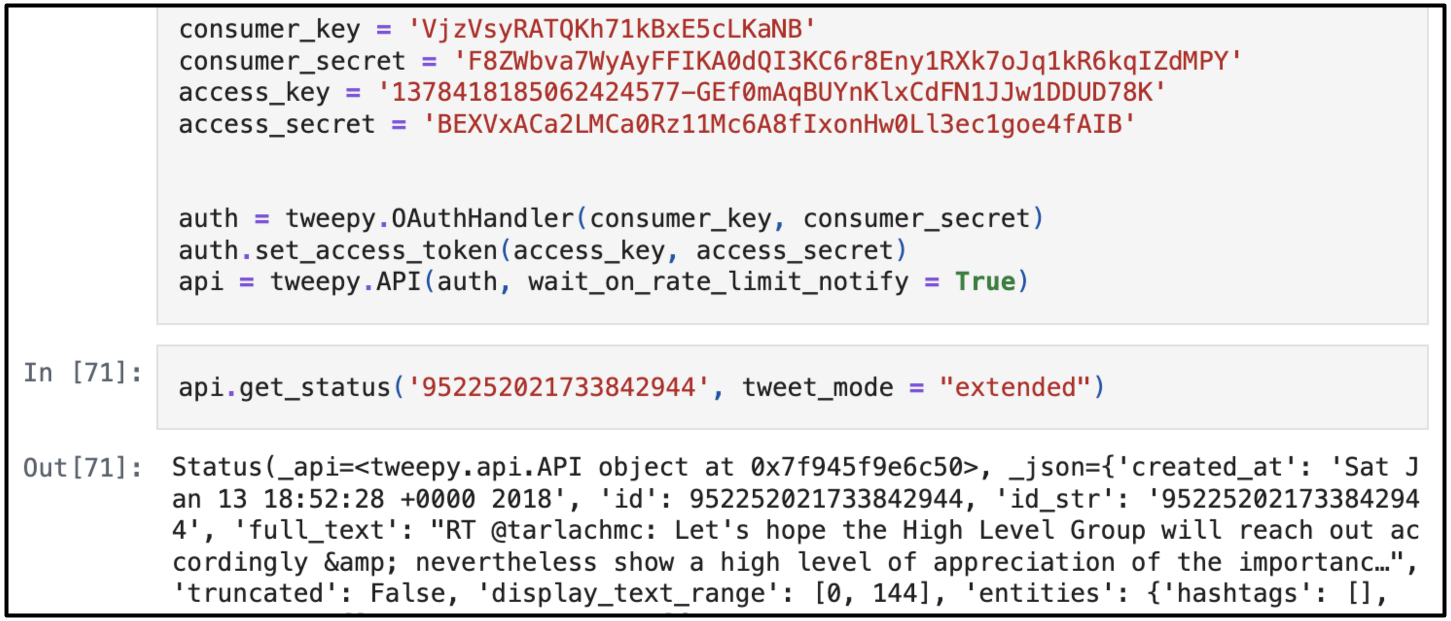
Retrieve data according to some query (like how you would get data from a database via SQL query)

Perform some sophisticated computation on data that you send to the API

Tweets via 3rd party API wrapper

Complex Twitter API: https://github.com/xdevplatform/Twitter-API-v2-sample-code

- Tweepy: Python library for easy interaction with the Twitter API: https://github.com/tweepy/tweepy/

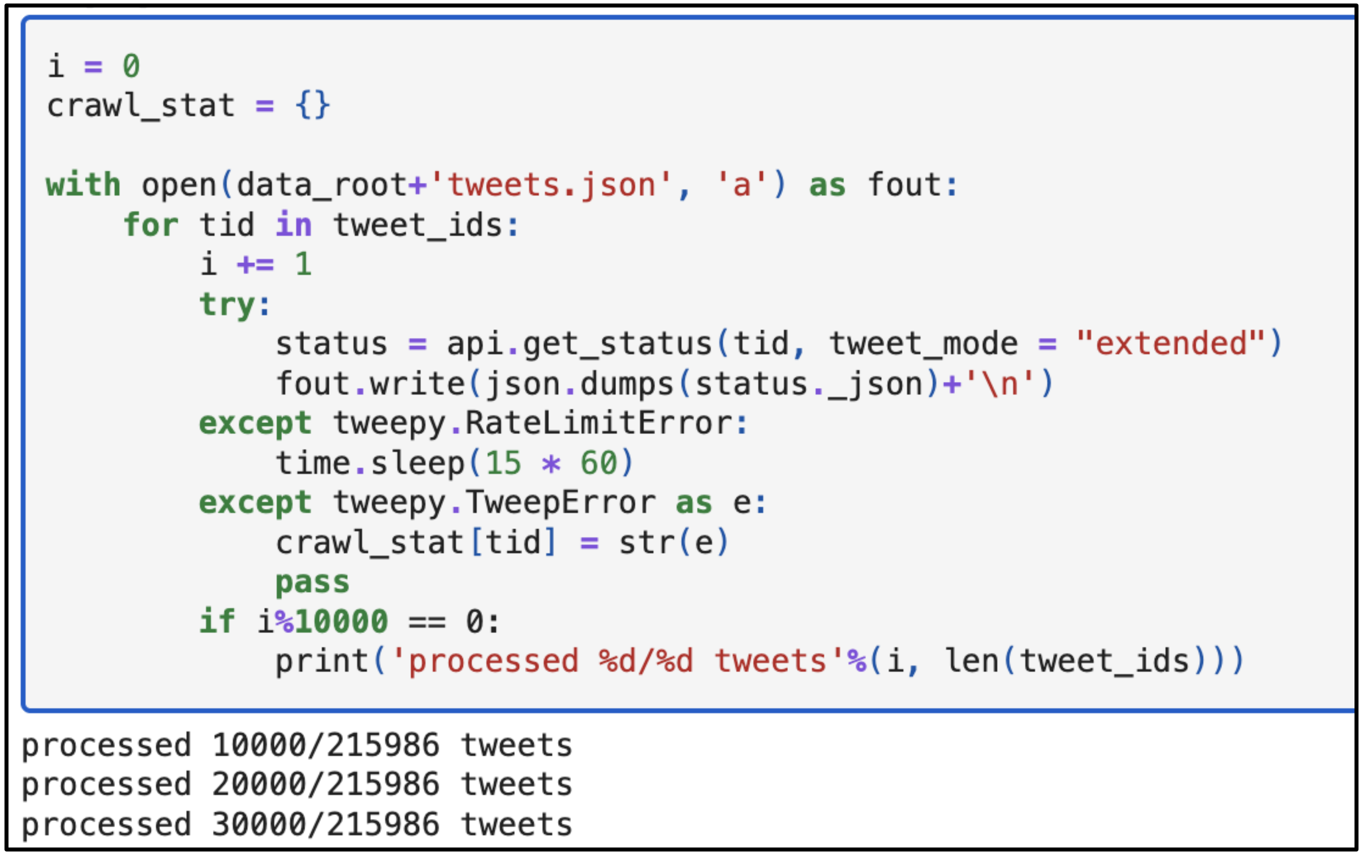


```
consumer_key = 'VjzVsyRATQKh71kBxE5cLKaNB'
consumer_secret = 'F8ZWbva7WyAyFFIKA0dQI3KC6r8Eny1RXk7oJq1kR6kqIZdMPY'
access_key = '1378418185062424577-GEf0mAqBUYnKlxCdFN1JJw1DDUD78K'
access_secret = 'BEXVxACa2LMCa0Rz11Mc6A8fIxonHw0Ll3ec1goe4fAIB'


auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
api = tweepy.API(auth, wait_on_rate_limit_notify = True)
```

```
In [71]:  api.get_status('952252021733842944', tweet_mode = "extended")

Out[71]:  Status(_api=<tweepy.api.API object at 0x7f945f9e6c50>, _json={'created_at': 'Sat J
          an 13 18:52:28 +0000 2018', 'id': 952252021733842944, 'id_str': '95225202173384294
          4', 'full_text': "RT @tarlachmc: Let's hope the High Level Group will reach out ac
          cordingly &amp; nevertheless show a high level of appreciation of the importanc…",
          'truncated': False, 'display_text_range': [0, 144], 'entities': {'hashtags': [],
```

Collecting Tweets using Tweepy

- Print out progress (when collecting large datasets)
- Sleep for a while to avoid hitting rate limits

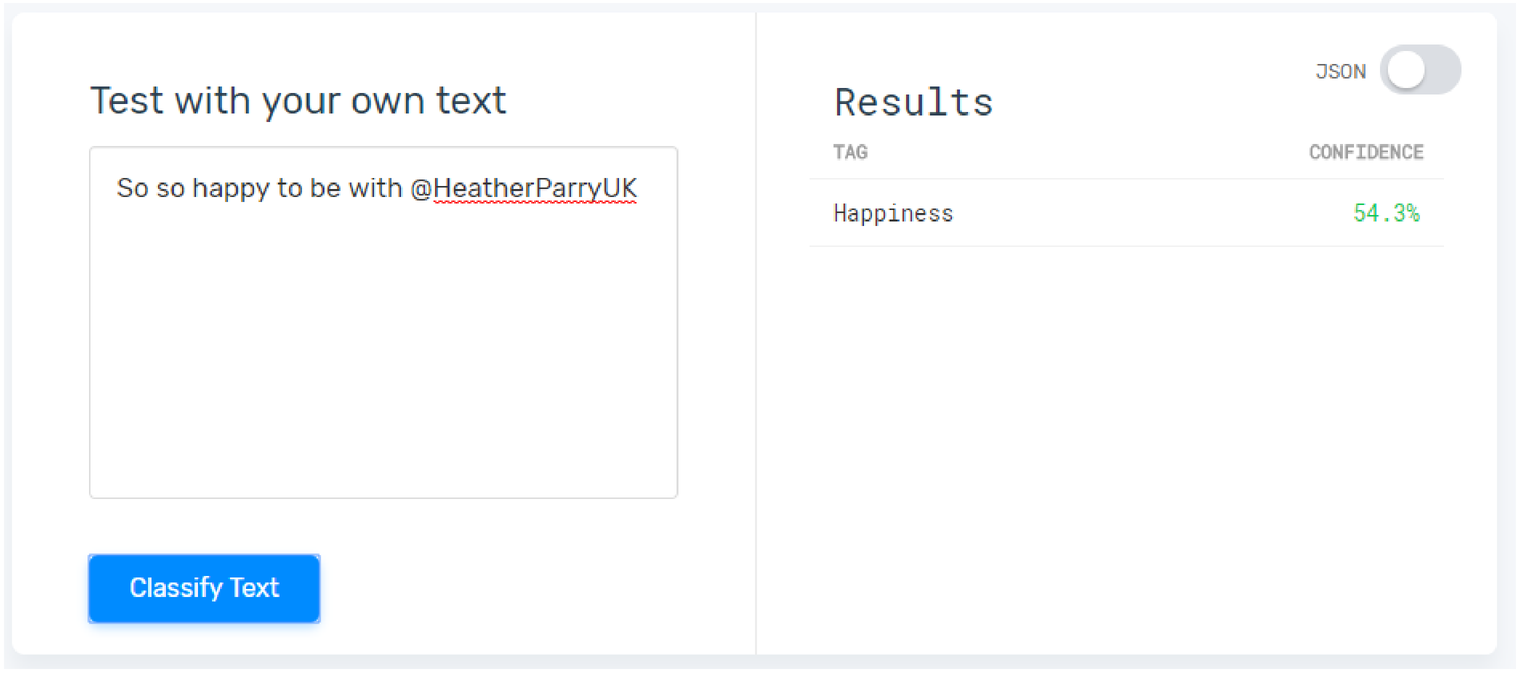


```python
i = 0
crawl_stat = {}

with open(data_root+'tweets.json', 'a') as fout:
    for tid in tweet_ids:
        i += 1
        try:
            status = api.get_status(tid, tweet_mode = "extended")
            fout.write(json.dumps(status._json)+'\n')
        except tweepy.RateLimitError:
            time.sleep(15 * 60)
        except tweepy.TweepError as e:
            crawl_stat[tid] = str(e)
            pass
        if i%10000 == 0:
            print('processed %d/%d tweets'%(i, len(tweet_ids)))
```

```
processed 10000/215986 tweets
processed 20000/215986 tweets
processed 30000/215986 tweets
```

Computationally-oriented APIs

E.g., MonkeyLearn API:
https://github.com/monkeylearn/monkeylearn-python



Results

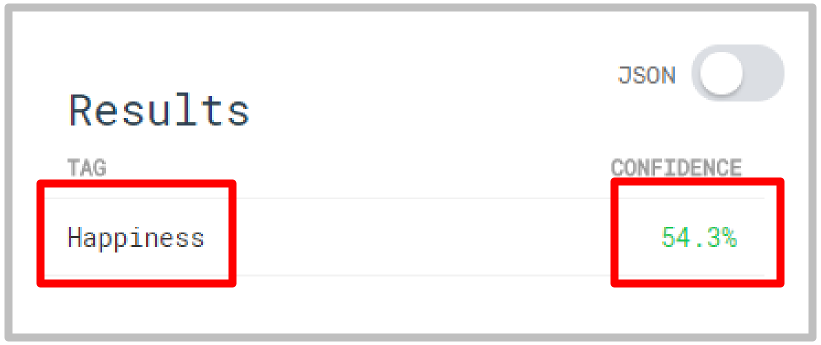
TAG	CONFIDENCE
Happiness	54.3%

Computationally-oriented APIs

```
from monkeylearn import MonkeyLearn
ml = MonkeyLearn('bd321e7772444c114885de7ad3371b69ac6286f8')
module_id = 'cl_mEcCuEcG’

text_list = ["So so happy to be with @HeatherParryUK"]
res = ml.classifiers.classify(module_id, text_list)
```

Computationally-oriented APIs

```
res.body
[{'text': 'So so happy to be with
  'external_id': None,
  'error': False,
  'classifications': [{'tag_name': 'Happiness',
    'tag_id': 55168807,
    'confidence': 0.543,
    'parents': [{'tag_name': 'Positive’,
      'tag_id': 55168804,
      'confidence': 0.692}]}]}]
```


Results

TAG	CONFIDENCE
Happiness	54.3%

What if there is no API wrapper?

- Work with APIs using the requests library
- Most APIs utilize the URL technology
  - URLs (Uniform Resource Locators) serve as the core mechanism for identifying and locating resources on the internet. This approach leverages the existing infrastructure and principles of the WWW.
  - Unique identifier for each web resource
  - Structured format (e.g., http://, https://)
  - Standard protocol for communication and exchange information between the client and the server

Web-based API Basics

Input: URL with params: http://something/foo/bar?x=y...

⬇

HTTP GET (or POST) the URL

⬇

Output: JSON or HTML or XML result

Note: Not all URLs are for APIs:

- URLs for static website: designed for humans to access web pages. They are used for content display in a web browser.
- URLs for APIs: designed for access by software applications. They serve as endpoints for apps. to request & exchange data.

Interacting with APIs in Python

In the old days: `sockets`, `HTTPClient`, `urllib2`

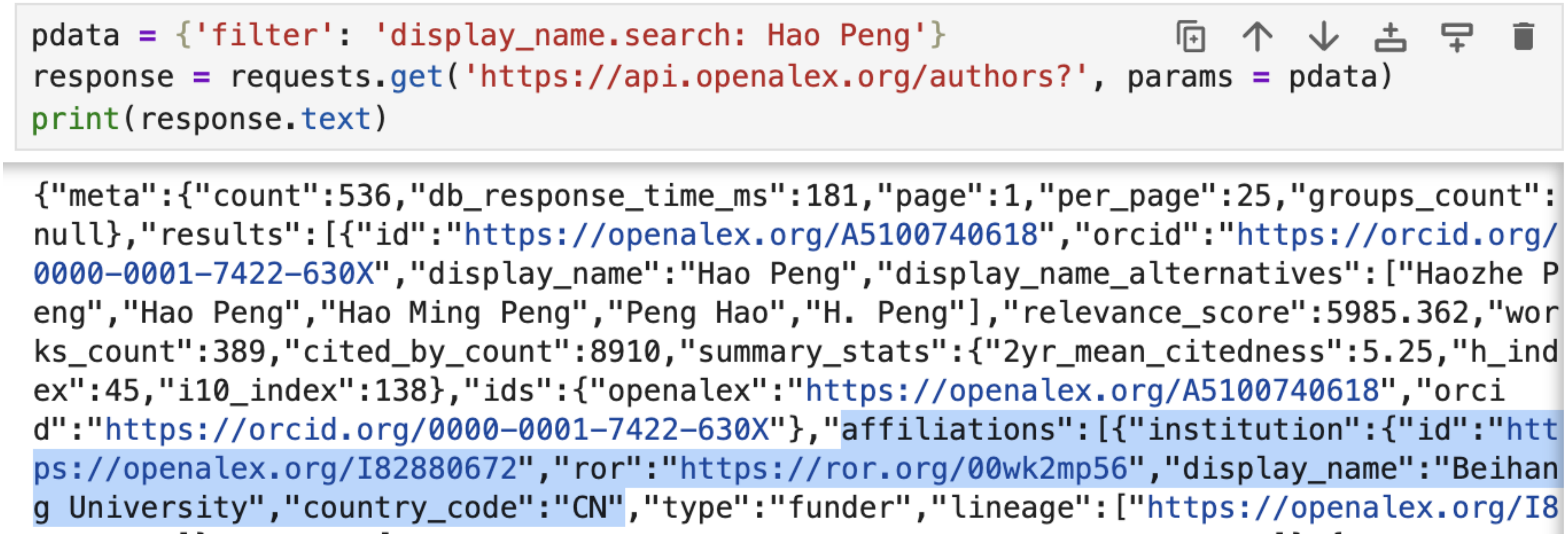
Now: **`requests`**

Note: It can be used to access any webpage with a URL

- API-based URLs (https://x.com/haoopeng)
- URLs for static pages (e.g., https://haoopeng.github.io/)
- URLs for dynamic/complex websites (not our focus!)

Accessing APIs with `requests`

- Search for authors by name in **OpenAlex** database:



```python
pdata = {'filter': 'display_name.search: Hao Peng'}
response = requests.get('https://api.openalex.org/authors?', params = pdata)
print(response.text)
```

```
{"meta":{"count":536,"db_response_time_ms":181,"page":1,"per_page":25,"groups_count":
null},"results":[{"id":"https://openalex.org/A5100740618","orcid":"https://orcid.org/
0000-0001-7422-630X","display_name":"Hao Peng","display_name_alternatives":["Haozhe P
eng","Hao Peng","Hao Ming Peng","Peng Hao","H. Peng"],"relevance_score":5985.362,"wor
ks_count":389,"cited_by_count":8910,"summary_stats":{"2yr_mean_citedness":5.25,"h_ind
ex":45,"i10_index":138},"ids":{"openalex":"https://openalex.org/A5100740618","orci
d":"https://orcid.org/0000-0001-7422-630X"},"affiliations":[{"institution":{"id":"htt
ps://openalex.org/I82880672","ror":"https://ror.org/00wk2mp56","display_name":"Beihan
g University","country_code":"CN","type":"funder","lineage":["https://openalex.org/I8
```

Check out results in your web browser:
https://api.openalex.org/authors?filter=display_name.search%3AHao+Peng

What if there is no API at all?

- Work with html directly using the `requests` library

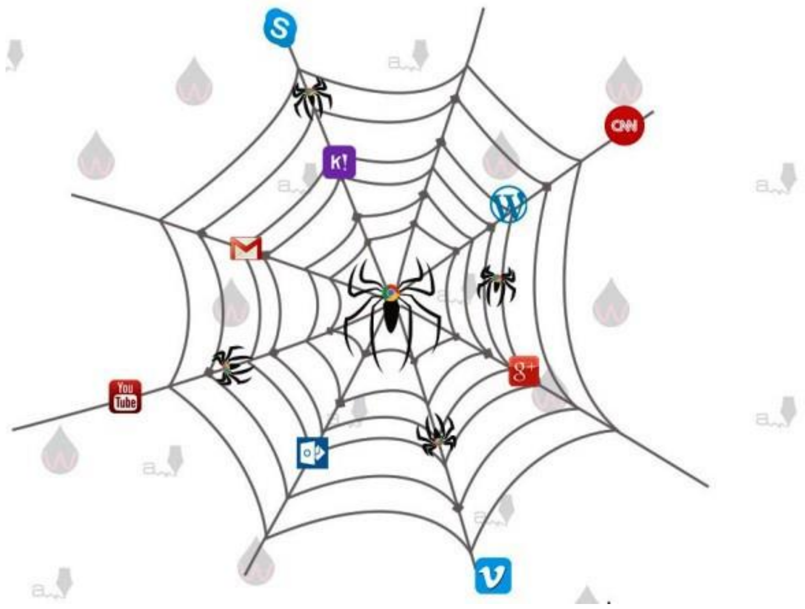
```
import requests
response = requests.get('http://data.pr4e.org/romeo.txt')
print(response.text)
But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief
```

What about more complicated websites?

- Nested websites / urls
- Target content buried in html/css markup language

Welcome to Web Crawling

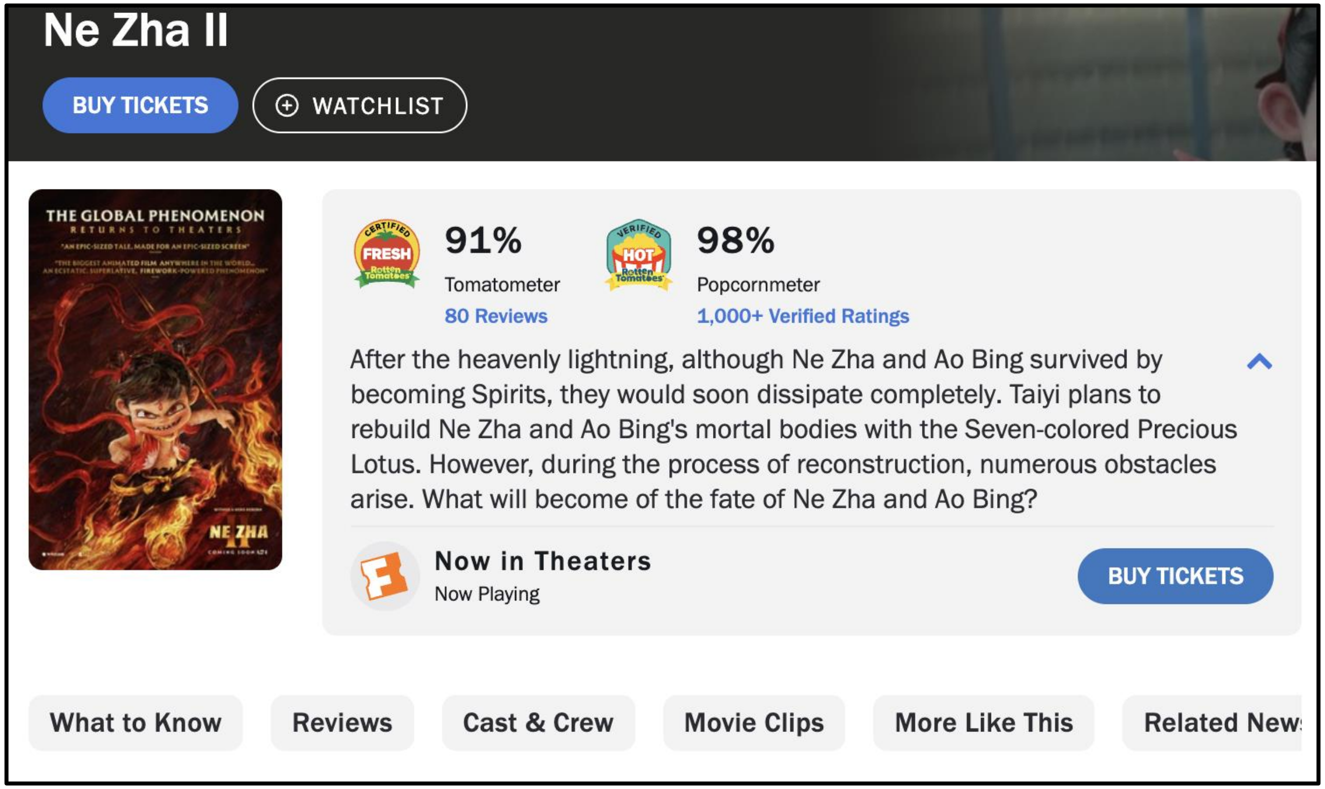
- A script to collect your target information from the Internet – a ***spider*** on the web:
- Automatic (machine strength)
- Save you time
- The spider would follow all your commands and would not make any mistakes (hopefully).



https://medium.com/@csgsajeewa/sample-web-crawler-using-crawler4j-8033bfcd8e60

Case study: RottenTomatoes

Collect the titles, ratings, No. of reviews, and detailed comments of Top 100 movies of the **21ST CENTURY**



Ne Zha II

BUY TICKETS

WATCHLIST

91%
Tomatometer
80 Reviews

98%
Popcornmeter
1,000+ Verified Ratings

After the heavenly lightning, although Ne Zha and Ao Bing survived by becoming Spirits, they would soon dissipate completely. Taiyi plans to rebuild Ne Zha and Ao Bing's mortal bodies with the Seven-colored Precious Lotus. However, during the process of reconstruction, numerous obstacles arise. What will become of the fate of Ne Zha and Ao Bing?

Now in Theaters
Now Playing

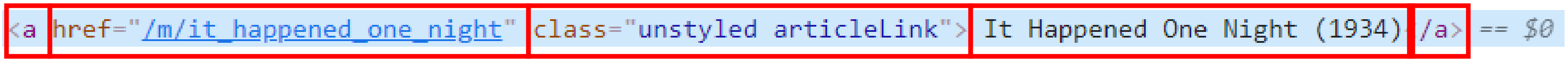
BUY TICKETS

Step 1: Inspect your data source

- Decipher URLs for the content you want
  - The base URL:
    https://editorial.rottentomatoes.com/guide/best-movies-21st-century/
  - The specific site location for each movie:
    https://www.rottentomatoes.com/m/ parasite_2019
    https://www.rottentomatoes.com/m/modern_times
    https://www.rottentomatoes.com/m/avengers_endgame
  - More specific site location for reviews:
    https://www.rottentomatoes.com/m/avengers_endgame/reviews
    https://www.rottentomatoes.com/m/avengers_endgame/reviews?type=top_critics
    https://www.rottentomatoes.com/m/avengers_endgame/reviews?type=user
- The URLs are more like tree structure
  - Clicked on the website and see how the URL changes.

Step 1: Inspect your data source

- Browser's developer tools.
  - **Mac:** ⌘ Cmd + Alt + I
  - **Windows/Linux:** ^ Ctrl + ⇧ Shift + I
- Right-click on the page and select the "Inspect" option.
- Example structure of the website source code (HTML).



```
<a href="/m/it_happened_one_night" class="unstyled articleLink"> It Happened One Night (1934)</a> == $0
```

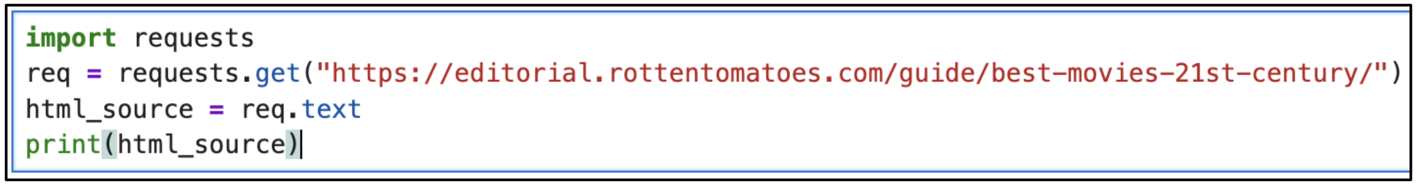
- Right-click the target element on the page and select "Inspect" to identify their location in the source code.

Step 2: Divide your tasks

- Get the description link of each movie from: https://editorial.rottentomatoes.com/guide/best-movies-21st-century/
- Collect title, ratings, reviews from each movie’s URL. E.g. https://www.rottentomatoes.com/m/parasite_2019
- Collect detailed review comments from: E.g. https://www.rottentomatoes.com/m/parasite_2019/reviews

Step 3: Get the html content

- Use *requests* to get the source code of each webpage.

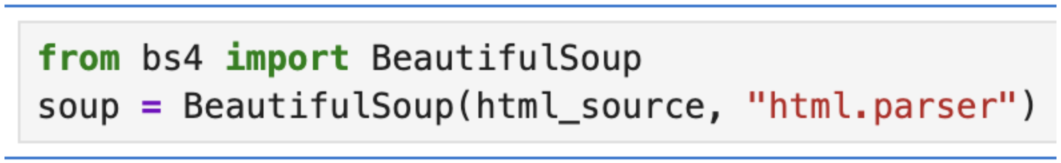


```python
import requests
req = requests.get("https://editorial.rottentomatoes.com/guide/best-movies-21st-century/")
html_source = req.text
print(html_source)
```

  - Retrieve HTML code from the server
  - Can only store content of static sites!
- Static websites (we will focus on this type)
  - The data fetched by *requests* contain all the information as shown on the browser, including your target data.
- Dynamic websites (not covered in this course)
  - Not all information can be fetched by *requests.*
  - Data shown in browser can be stored in a database on server.
  - Need to use browser-like crawler (e.g., Selenium).

Step 4: Process HTML source code

- Can build your own regular expression to match the data you want (time consuming but more powerful)
- Beautiful Soup (available as a Python package)
  - https://www.crummy.com/software/BeautifulSoup/
  - Interact with HTML similarly to how you would interact with a webpage using the developer tool.
- Just create a BS object with the whole html string



```python
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_source, "html.parser")
```

  - Can use different parsers depending on type of source data
  - https://www.crummy.com/software/BeautifulSoup/bs4/doc/#differences-between-parsers

Step 4: Process HTML source code

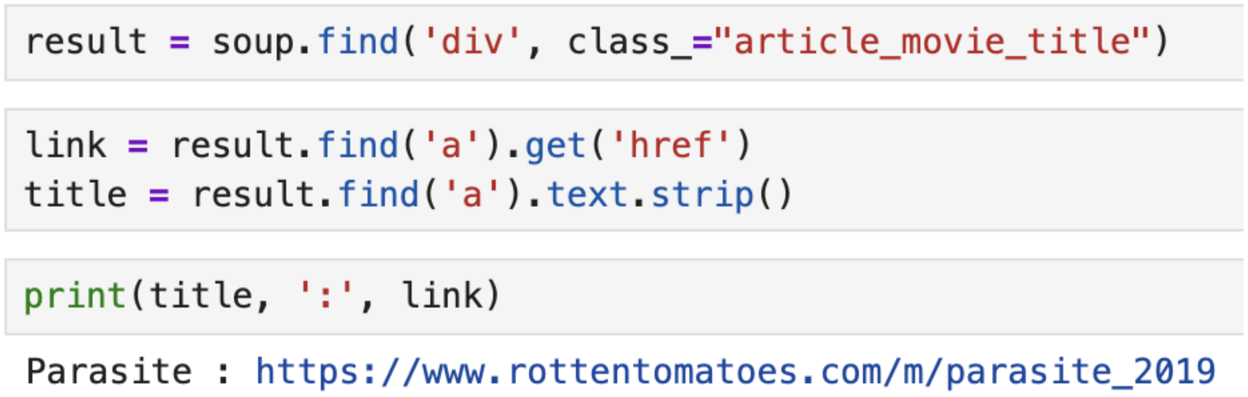
Find elements via the BS object

- Use the tag and attribute to find elements you want.
- *find(tag, attrs, recursive, text, **kwargs)*
  - https://beautiful-soup-4.readthedocs.io/en/latest/#find
  - just find one matched result (the first one)
  - *tag*: the tag where your data is contained
  - *attrs*: more precise finding with additional attributes filters
  - *recursive*: True or False, the function will find all the child nodes if True, otherwise only the parent
- *find_all(tag, attrs, recursive, text, limit, **kwargs)*
  - https://beautiful-soup-4.readthedocs.io/en/latest/#find-all
  - return a list containing all the matched results
  - *limit:* the number of results returned. Equals to find when limit=1

Step 4: Process HTML source code

Process matched elements:

- Find (nested) elements by tags and attributes
- Extract link from HTML element via *[.get('href')]*
- Extract text from HTML element via *[.text]*



```
result = soup.find('div', class_="article_movie_title")
```

```
link = result.find('a').get('href')
title = result.find('a').text.strip()
```

```
print(title, ':', link)
```

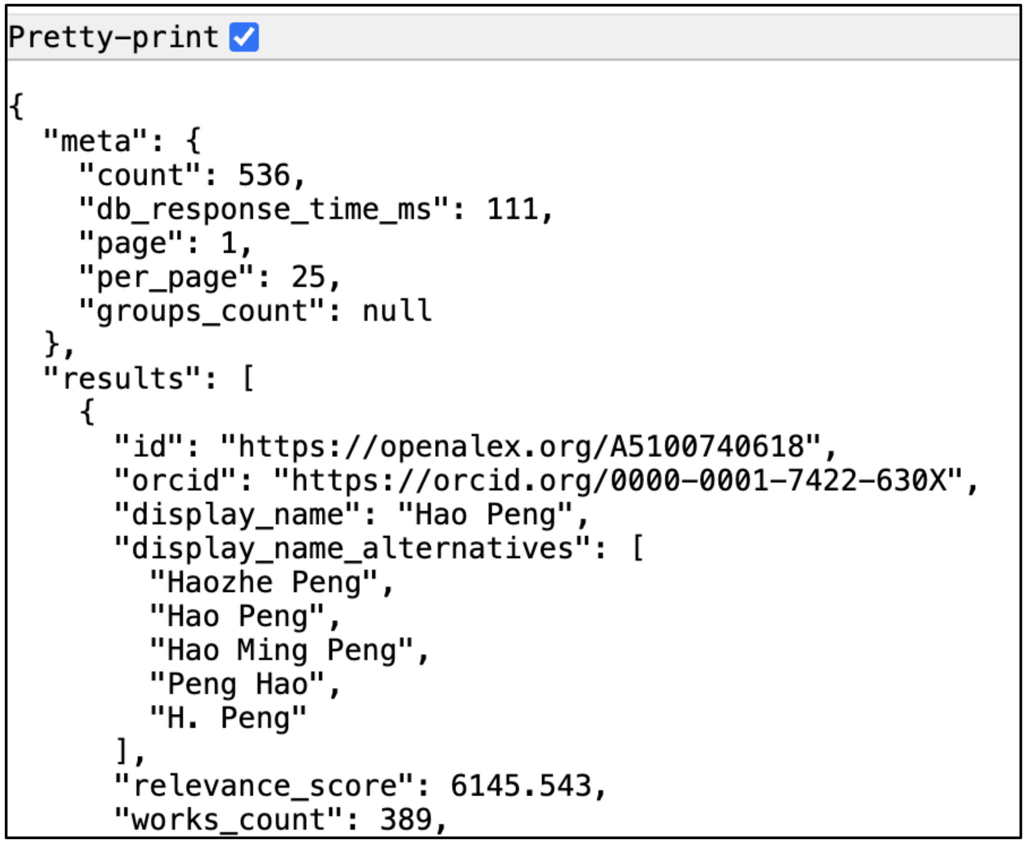
```
Parasite : https://www.rottentomatoes.com/m/parasite_2019
```

- Iterate through all URLs and search reviews for each movie

Why do most APIs return JSON?

- Search authors in OpenAlex:

https://api.openalex.org/authors?filter=display_name.search%3AHao+Peng


Pretty-print ✓

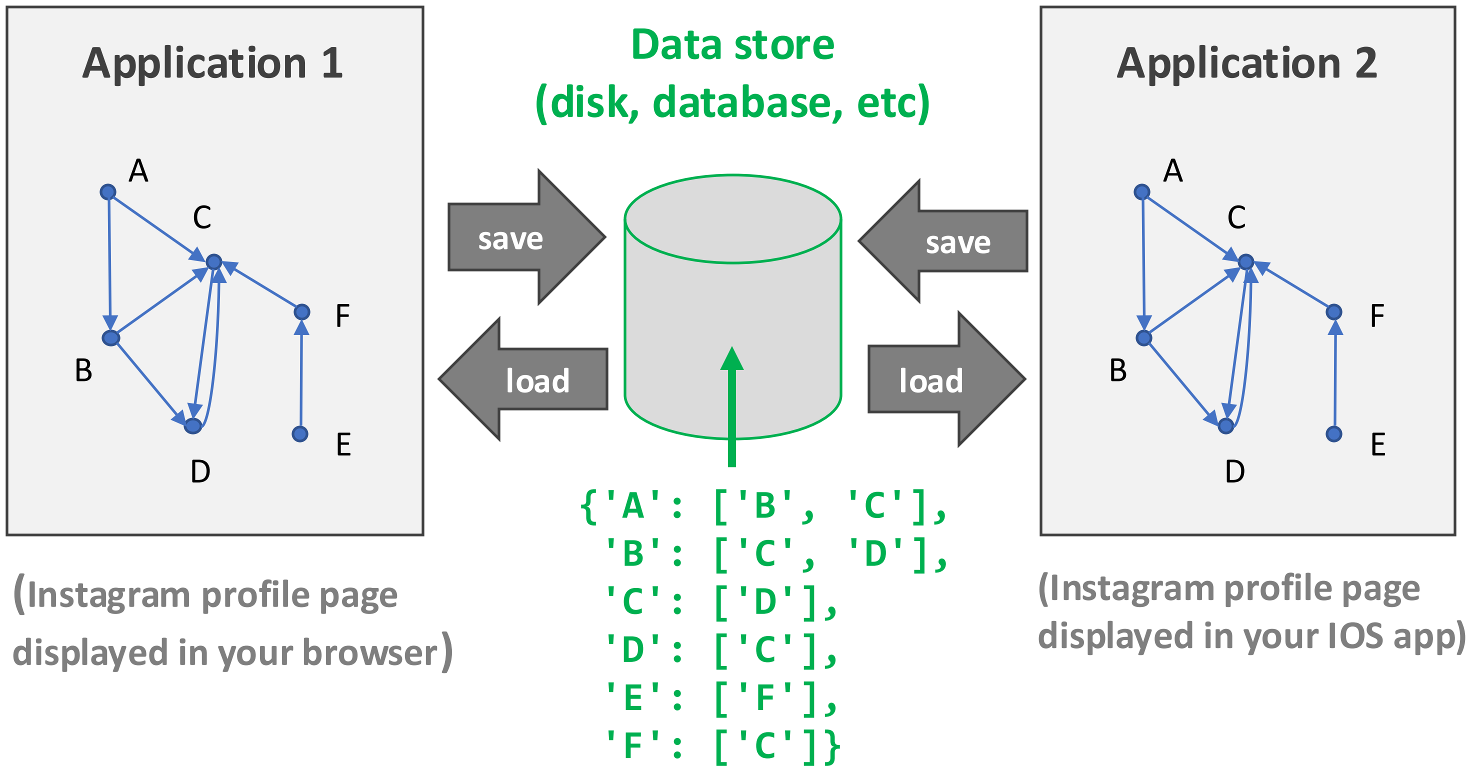
```
{
  "meta": {
    "count": 536,
    "db_response_time_ms": 111,
    "page": 1,
    "per_page": 25,
    "groups_count": null
  },
  "results": [
    {
      "id": "https://openalex.org/A5100740618",
      "orcid": "https://orcid.org/0000-0001-7422-630X",
      "display_name": "Hao Peng",
      "display_name_alternatives": [
        "Haozhe Peng",
        "Hao Peng",
        "Hao Ming Peng",
        "Peng Hao",
        "H. Peng"
      ],
      "relevance_score": 6145.543,
      "works_count": 389,
```

JSON: JavaScript Object Notation

```
{"employees": [
        { "firstName": "John",  "lastName": "Doe"   },
        { "firstName": "Anna",  "lastName": "Smith" },
        { "firstName": "Peter", "lastName": "Jones" }
    ]
}
```

- JSON syntax is a subset of JavaScript
- **Two basic structures**:
    - **Curly braces** hold **objects** (aka **dictionaries**)
    - **Square brackets** hold **arrays** (aka **lists**)
- Filename extension `".json"`
- MIME type (e.g. HTTP header) `application/json`

Why do most APIs return JSON?



```
{'A': ['B', 'C'],
 'B': ['C', 'D'],
 'C': ['D'],
 'D': ['C'],
 'E': ['F'],
 'F': ['C']}
```

(Instagram profile page displayed in your browser)

(Instagram profile page displayed in your IOS app)

Why JSON? **Persistence**

Common format, works in many languages

Simple, fast to parse

Human-readable

(Learned some lessons from **XML…)**

JSON vs XML/HTML

Similarities

- JSON is plain text
- JSON human readable
- JSON is hierarchical

Differences

- JSON is shorter, simpler
- JSON is faster to read/write (less complex to parse)

JSON

```
{"employees": [
    { "firstName":"John" , "lastName":"Doe" },
    { "firstName":"Anna" , "lastName":"Smith" },
    { "firstName":"Peter" , "lastName":"Jones" }
  ]
}
```

XML

```
<employees>
    <name firstName = "John", lastName = "Doe"/>
    <name firstName = "Anna", lastName = "Smith"/>
    <name firstName = "Peter", lastName = "Jones"/>
</employees>
```

JSON and Python

Default support:

```
import json
```

Decoding JSON strings into objects:

```
json.loads(string)
```

loads = "**load string**"

string -> Python object

Encoding Python objects into strings:

```
json.dumps(object)
```

dumps = "**dump string**"

Python object -> string

http://docs.python.org/3/library/json.html

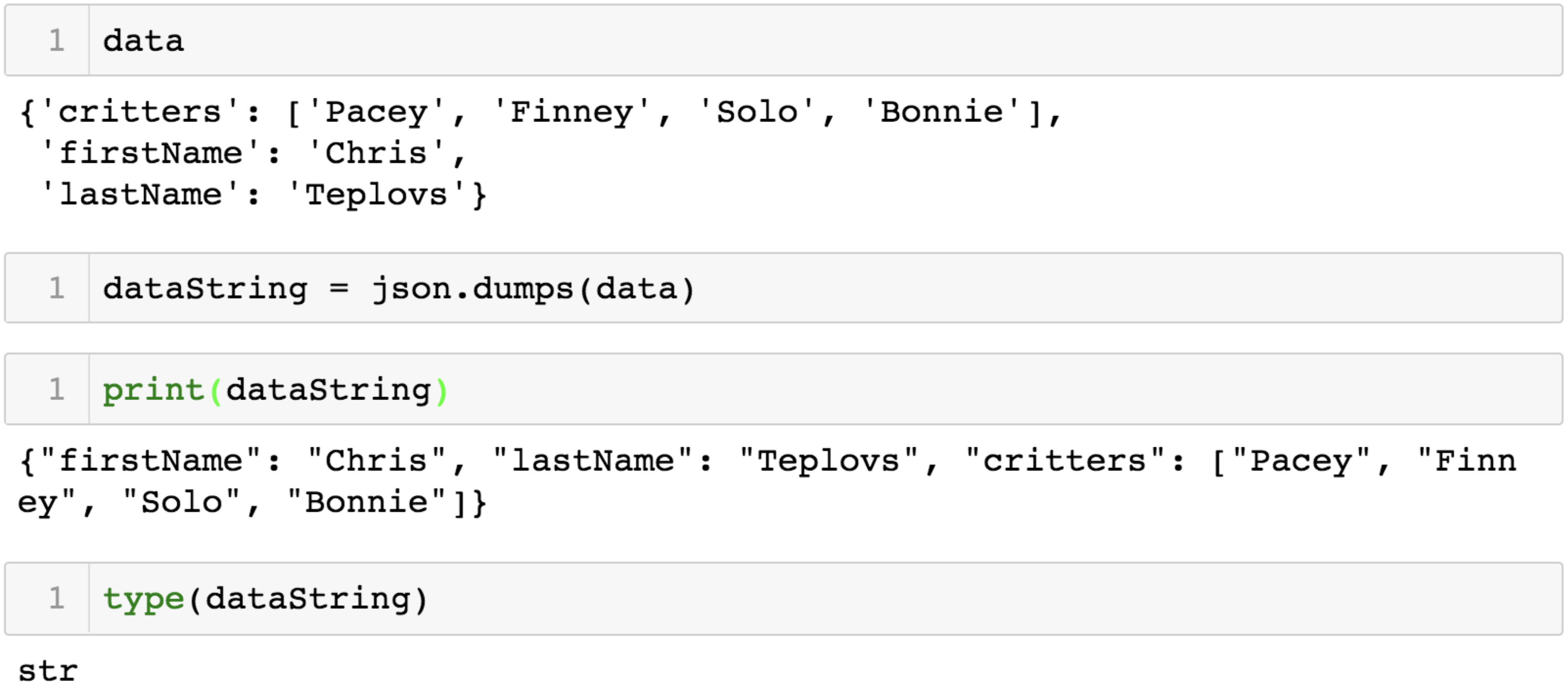
Decoding string to Json (loads)

```python
import json

data = json.loads('''
{"firstName": "Chris",
 "lastName": "Teplovs",
 "critters": ["Pacey","Finney", "Solo", "Bonnie"]}'''
                  )
```

```python
data
```

```
{'critters': ['Pacey', 'Finney', 'Solo', 'Bonnie'],
 'firstName': 'Chris',
 'lastName': 'Teplovs'}
```

Encoding Json to string (dumps)



```
data
```

```
{'critters': ['Pacey', 'Finney', 'Solo', 'Bonnie'],
 'firstName': 'Chris',
 'lastName': 'Teplovs'}
```

```
dataString = json.dumps(data)
```

```
print(dataString)
```

```
{"firstName": "Chris", "lastName": "Teplovs", "critters": ["Pacey", "Finn
ey", "Solo", "Bonnie"]}
```

```
type(dataString)
```

```
str
```

Round-trips may not yield identical results

Encoding then decoding may not give exactly the same type

```
data = {'a': 'A', 'b': (2, 4)}


data_string = json.dumps(data)
data_string
#     '{"a": "A", "b": [2, 4]}'
```

Round-trips may not yield identical results

Encoding then decoding may not give exactly the same type

```
data = {'a': 'A', 'b': (2, 4)}


data_string = json.dumps(data)
data_string
#      '{"a": "A", "b": [2, 4]}'


decoded = json.loads(data_string)
decoded
#       {'a': 'A', 'b': [2, 4]}
```

Python tuple becomes JSON array

Round-trips may not yield identical results

Encoding then decoding may not give exactly the same type

```
data = {'a': 'A', 'b': (2, 4)}



data_string = json.dumps(data)
data_string
#     '{"a": "A", "b": [2, 4]}'



decoded = json.loads(data_string)
decoded
#      {'a': 'A', 'b': [2, 4]}
```

Python <u>tuple</u> becomes JSON <u>array</u>

JSON <u>array</u> becomes Python <u>list</u>

Sorting keys in output

```
data = [{'a': 'A', 'b': (2, 4), 'c': 3.0}]

json.dumps(data)
#      '[{"a": "A", "c": 3.0, "b": [2, 4]}]'  # random key order

json.dumps(data, sort_keys=True)
#      '[{"a": "A", "b": [2, 4], "c": 3.0}]'  # sorted key order
```

Indentation in output :

```
data = [{'a': 'A', 'b': (2, 4), 'c': 3.0}]
```

```
print(json.dumps(data, sort_keys=True, indent=2))
```

```
[
  {
    "a": "A",
    "b": [
      2,
      4
    ],
    "c": 3.0
  }
]
```

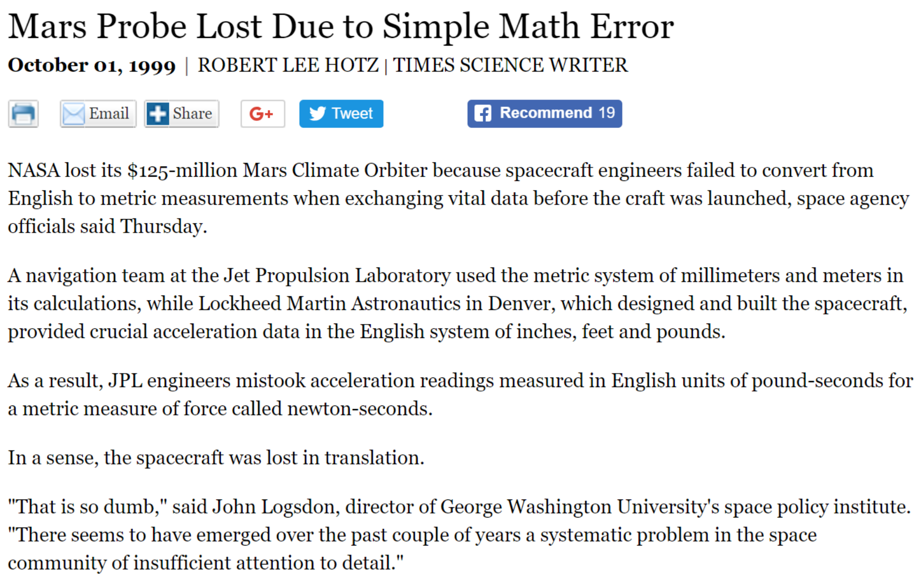
Data preprocessing pipelines

- Why preprocess the data?
- How to clean the data?
- Data integration and transformation
- Data reduction
- Data discretization

Why

- Data in the real world is dirty
    - incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data;
        - Very often you don’t know why certain values are missing.
    - noisy: containing errors or outliers
    - inconsistent: discrepancies in codes or names
- No quality data, no quality results!
    - Quality decisions must be based on quality data
    - Data warehouse needs consistent integration of quality data – this is often the most difficult part!

Inconsistency can be disastrous



Mars Probe Lost Due to Simple Math Error

October 01, 1999 | ROBERT LEE HOTZ | TIMES SCIENCE WRITER

NASA lost its $125-million Mars Climate Orbiter because spacecraft engineers failed to convert from English to metric measurements when exchanging vital data before the craft was launched, space agency officials said Thursday.

A navigation team at the Jet Propulsion Laboratory used the metric system of millimeters and meters in its calculations, while Lockheed Martin Astronautics in Denver, which designed and built the spacecraft, provided crucial acceleration data in the English system of inches, feet and pounds.

As a result, JPL engineers mistook acceleration readings measured in English units of pound-seconds for a metric measure of force called newton-seconds.

In a sense, the spacecraft was lost in translation.

"That is so dumb," said John Logsdon, director of George Washington University's space policy institute. "There seems to have emerged over the past couple of years a systematic problem in the space community of insufficient attention to detail."

Remember the Mars Climate Orbiter incident from 1999?

Data cleaning

- Data cleaning tasks
    - Fill in missing values
    - Identify outliers and smooth out noisy data
    - Correct inconsistent data

Missing data

- Data is not always available
  - E.g., many observations have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data
- Missing data may need to be inferred.

How to handle missing data?

- Fill in the missing value manually: tedious + infeasible?
- Use a global constant to fill in the missing value: e.g., “unknown”, a new class?!
- Use the attribute mean to fill in the missing value
- Use the most probable value to fill in the missing value (e.g., impute using the mean of *n* similar data points; KNN)

Noise data

- Noise: random error or variance in a measured variable
- Incorrect values may be due to
  - faulty data collection instruments
  - data entry problems
  - data transmission problems
  - inconsistency in naming convention
- Other data noise problems which require data cleaning
  - duplicate records
  - incomplete data
  - inconsistent data

How to handle noisy data?

- Binning method:
  - first sort data and partition into (equal-size) bins
  - then smooth by bin mean, median, boundaries, etc.
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human
- Regression
  - smooth by fitting the data into regression functions

Data binning

- Equal-width (distance) partitioning:
  - It divides the range into *N* intervals of equal size: uniform grid
  - if *A* and *B* are the lowest and highest values of the variable, the width of intervals will be: W = (B-A)/N.
  - The most straightforward
  - But outliers may dominate presentation
  - Skewed data is not handled well.
- Equal-size (frequency) partitioning:
  - It divides the range into *N* intervals, each containing approximately same number of samples
  - Good data scaling
  - Managing categorical attributes can be tricky.

Data binning

* Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
* Partition into (equal-size) bins:
    - Bin 1: 4, 8, 9, 15
    - Bin 2: 21, 21, 24, 25
    - Bin 3: 26, 28, 29, 34
* Smoothing by bin means:
    - Bin 1: 9, 9, 9, 9
    - Bin 2: 23, 23, 23, 23
    - Bin 3: 29, 29, 29, 29
* Smoothing by bin boundaries:
    - Bin 1: 4, 4, 4, 15
    - Bin 2: 21, 21, 21, 25
    - Bin 3: 26, 26, 26, 34

Data integration

- Data integration:
  - combines data from multiple sources into a coherent store
- Schema integration
  - integrate metadata from different sources
  - entity identification: identify same entities from multiple data sources
- Detecting and resolving data value conflicts
  - for the same real-world entity, its attribute values from different sources may be different
  - possible reasons: representations, different scales/metrics/units

Data transformation

- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
    - min-max normalization
    - z-score normalization
    - normalization by decimal scaling

Normalization

- min-max normalization

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

- z-score normalization

$$v' = \frac{v - mean_A}{stand\_dev_A}$$

- normalization by decimal scaling (or log scale)

$$v' = \frac{v}{10^j}$$ Where j is the smallest integer such that Max(| v' |)<1

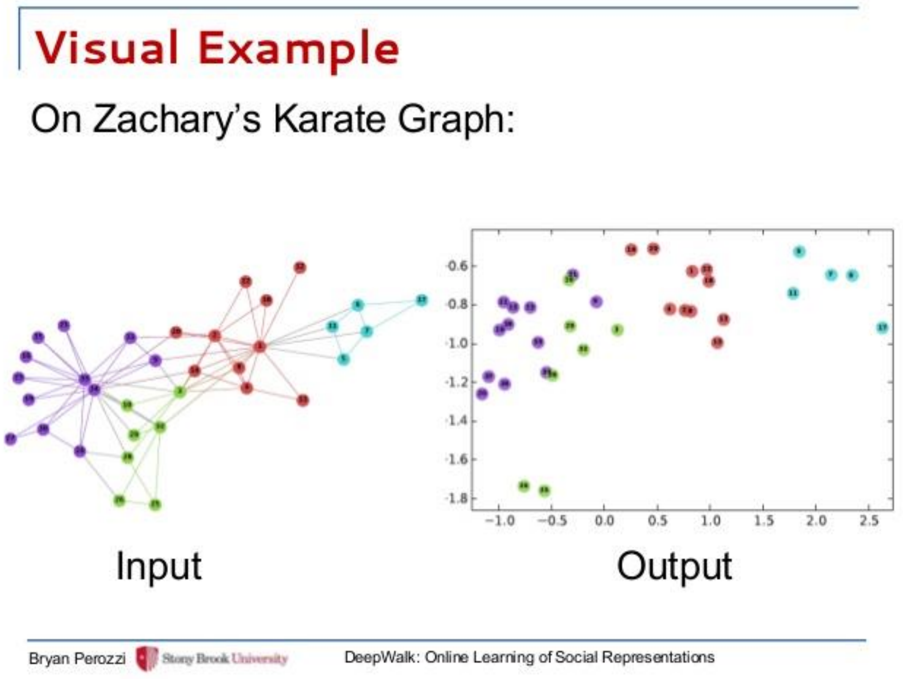
Data reduction

- Data warehouse may store terabytes of data:
    - analysis can take a long time to run on the full dataset.
- Data reduction
    - Obtains a reduced representation of the dat that is much smaller in volume but produces the same (or almost the same) results.
- Data reduction strategies
    - Dimensionality reduction
    - Discretization and concept hierarchy generation

Data reduction

- Feature selection (i.e., selecting subset of attributes):
  - Select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features.
  - Reduce # of patterns in the data, easier to understand.

Data reduction



Visual Example

On Zachary's Karate Graph:

Input

Output

Bryan Perozzi Stony Brook University DeepWalk: Online Learning of Social Representations

Discretization

- Discretization
  - reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values.
- Concept hierarchies
  - reduce the data by collecting and replacing low level concepts (such as numeric values for the attribute age) by higher level concepts (such as young, middle-aged, or senior).

Discretization

- Common methods
  - Binning
  - Histogram analysis
  - Clustering analysis

Paper presentation