### Web Mining and Recommender Systems

Tools: Collecting and parsing Web data with urllib and BeautifulSoup



Show how to crawl and parse web data

### Collecting our own datasets

Suppose that we wanted to collect data from a website, but didn't yet have CSV or JSON formatted data

- How could we collect new datasets in machinereadable format?
  - What Python libraries could we use to collect data from webpages?
  - Once we'd collected (e.g.) raw html data, how could we extract structured information from it?

### Collecting our own datasets

## E.g. suppose we wanted to collect reviews of "The Great Gatsby" from goodreads.com:

(https://www.goodreads.com/book/show/4671.The\_Great\_Gatsby)



#### The Great Gatsby

by F. Scott Fitzgerald, Jake Gyllenhaal (Narrator)

★★★★ 3.9 · 〒 Rating details · 3,090,834 Ratings · 56,204 Reviews

A true classic of twentieth-century literature, this edition has been updated by Fitzgerald scholar James L.W. West III to include the author's final revisions and features a note on the composition and text, a personal foreword by Fitzgerald's granddaughter, Eleanor Lanahan—and a new introduction by two-time National Book Award winner Jesmyn Ward.

Want to Read	
Rate this book	
****	

The Great Gatsby, F. Scot ...more



Published September 2004 by Scribner (first published April 1925) More Details...

edit details

### Collecting our own datasets



Oh Gatsby, you old sport, you poor semi-delusionally hopeful dreamer with 'some heightened sensitivity to the promises of life', focusing your whole self and soul on that elusive money-colored green light – a dream that shatters just when you are \*this\* close to it.



Jay Gatsby, who dreamed a dream with the passion and courage few possess - and the tragedy was that it was a wrong dream colliding with reality that was even more wrong - and deadly.

Just like the Great Houdini - the association the ...more

713 likes · Like · see review

Alex rated it \*\*\*\*\*

#### Dec 24, 2007

The Great Gatsby is your neighbor you're best friends with until you find out he's a drug dealer. It charms you with some of the most elegant English prose ever published, making it difficult to discuss the novel without the urge to stammer awestruck about its beauty. It would be evidence enough to How could we extract fields including

- The *ID* of the user,
- The *date* of the review
- The star rating
- The *text* of the review itself?
- The shelves the book belongs to

### Code: urllib

## Our first step is to extract the html code of the webpage into a python string. This can be done using **urllib**:

In [1]: from urllib.request import urlopen In [2]: f = urlopen("https://www.goodreads.com/book/show/4671.The Great Gatsby") Note: url of "The Great In [3]: html = str(f.read()) Gatsby" reviews **Note:** acts like a file object once opened In [4]: html Out[4]: 'b\'<!DOCTYPE html>\\n<html class="desktop\\n">\\n\\n<head prefix="og: http://ogp.me/ns# fb: http://ogp.me/ns/f</pre> b# good reads: http://ogp.me/ns/fb/good reads#">\\n <title>\\nThe Great Gatsby by F. Scott Fitzgerald\\n</title> <script type="text/javascript"> var ue t0=window.ue t0||+new Date();\\n </script>\\n <script type="t</pre>  $\lambda n n n n$ (function(e){var c=e;var a=c.ue||{};a.main scope="mainscopecsm";a.g=[];a.t0=c.ue t0||+new D ext/javascript">\\n ate();a.d=g;function g(h){return +new Date()-(h?0:a.t0)}function d(h){return function(){a.g.push({n:h,a:arguments, t:a.d()})}function b(m,l,h,j,i){var k={m:m,f:l,l:h,c:""+j,err:i,fromOnError:1,args:arguments};c.ueLogError(k);ret urn false}b.skipTrace=1;e.onerror=b;function f(){c.uex("ld")}if(e.addEventListener){e.addEventListener("load",f,fa lse)}else{if(e.attachEvent){e.attachEvent("onload",f)}}a.tag=d("tag");a.log=d("log");a.reset=d("rst");c.ue csm=c; c.ue=a;c.ueLogError=d("err");c.ues=d("ues");c.uet=d("uet");c.uex=d("uex");c.uet("ue")})(window);(function(e,d){var}) a=e.ue||{};function c(q){if(!q){return}var f=d.head||d.getElementsByTagName("head")[0]||d.documentElement,h=d.cre ateElement("script");h.async="async";h.src=q;f.insertBefore(h,f.firstChild)}function b(){var k=e.ue cdn||"z-ecx.im ages-amazon.com",g=e.ue cdns||"images-na.ssl-images-amazon.com",j="/images/G/01/csminstrumentation/",h=e.ue file|

### Reading the html data

## This isn't very nice to look at, it can be easier to read in a browser or a text editor (which preserves formatting):

```
<!DOCTYPE html>
  <html class="desktop
  ">
3
  <head prefix="oq: http://oqp.me/ns# fb: http://oqp.me/ns/fb# good reads: http://oqp.me/ns/fb/good reads#">
    <title>
  The Great Gatsby by F. Scott Fitzgerald
  </title>
10
11
      <script type="text/javascript"> var ue t0=window.ue t0||+new Date();
12
   </script>
13
    <script type="text/javascript">
14
      (function(e){var c=e;var a=c.ue||{};a.main scope="mainscopecsm";a.q=[];a.t0=c.ue t0||+new Date();a.d=q;fu
  {m:m,f:l,l:h,c:""+j,err:i,fromOnError:1,args:arguments};c.ueLogError(k);return false}b.skipTrace=1;e.onerror=
  {e.attachEvent("onload",f)}}a.tag=d("tag");a.log=d("log");a.reset=d("rst");c.ue csm=c;c.ue=a;c.ueLogError=d("
  f=d.head||d.getElementsByTagName("head")[0]||d.documentElement,h=d.createElement("script");h.async="async";h.
  amazon.com", j="/images/G/01/csminstrumentation/", h=e.ue file||"ue-full-11e51f253e8ad9d145f4ed644b40f692. V1 .
  1:0}i=f?"https://":"http://";i+=f?g:k;i+=j;i+=h;c(i)}if(!e.ue inline){if(a.loadUEFull){a.loadUEFull()}else{b(
```

### Reading the html data

## To extract review data, we'll need to look for the part of the html code which contains the reviews:



### Reading the html data

To extract review data, we'll need to look for the part of the html code which contains the reviews:



- Note that each individual review starts with a block containing the text "<div id="review\_..."</li>
- We can collect all reviews by looking for instances of this text

### Code: string.split()

# To split the page into individual reviews, we can use the string.split() operator. Recall that we saw this earlier when reading csv files:

<pre>In [5]: reviews = html.split('<div href="/user/show/3672777-nataliya" id="review_')[1:]&lt;/pre&gt;&lt;/th&gt;&lt;th&gt;&lt;/th&gt;&lt;/tr&gt;&lt;tr&gt;&lt;th&gt;&lt;pre&gt;l"><img alt="Nataliy 77.jpg"/>\\n\\n <div class="left bodycol">\\n ="reviewDate createdAt right" href="/review/show/101057 itemprop="author" itemscope itemtype="http://schema.org ="url" name="Nataliya" href="/user/show/3672777-nataliy</div></div></pre>	<pre>now/101057684" /&gt;\\n <a class="left imgco&lt;br&gt;ya" reviewheader="" src="https://images.gr-assets.com/users/1395089173p2/36727&lt;br&gt;&lt;div class=" stacked"="" title="Nataliya" uitext="">\\n <a class<br="">'684?book_show_action=true"&gt;May 02, 2010</a>\\n\\n <span y/Person"&gt;\\n <a class="user" itemprop<br="" title="Nataliya">ya"&gt;Nataliya</a>\\n \\n\\n rated it\\n pan size="15x15" class="staticStar p10"&gt;it was amazing size="15x15" class="staticStar p10"&gt;<span <="" pre="" size="15x15"></span></span </a></pre>
--	--

#### Next we have to write a method to parse individual reviews (i.e., given the text of one review, extract formatted fields into a dictionary)

```
In [8]: def parseReview(review):
             d = \{\}
             d['stars'] = review.split('<span class=" staticStars" title="')[1].split('"')[0]</pre>
             d['date'] = review.split('<a class="reviewDate')[1].split('>')[1].split('<')[0]</pre>
             d['user'] = review.split('<a title="')[1].split('"')[0]</pre>
             shelves = []
             try:
                 shelfBlock = review.split('<div class="uitext greyText bookshelves">')[1].split('</div')[0]</pre>
                 for s in shelfBlock.split('shelf=')[1:]:
                     shelves.append(s.split('"')[0])
                 d['shelves'] = shelves
             except Exception as e:
                 pass
             reviewBlock = review.split('<div class="reviewText stacked">')[1].split('</div')[0]</pre>
             d['reviewBlock'] = reviewBlock
             return d
```

Let's look at it line-by-line:

#### In [8]: def parseReview(review): d = {}

- We start by building an empty dictionary
- We'll use this to build a *structured* version of the review

#### Let's look at it line-by-line:

**Note:** Two splits: everything *after* the first quote, and *before* the second quote

• The next line is more complex:

d['stars'] = review.split('<span class=" staticStars" title="')[1].split('"')[0]</pre>

• We made this line by noticing that the stars appear in the html inside a span with class " staticStars":



• Our "split" command then extracts everything inside the "title" quotes

1240

#### Let's look at it line-by-line:

• The following two lines operate in the same way: Note: Everything between the two brackets of this "<a" element

```
d['date'] = review.split('<a class="reviewDate')[1].split('>')[1].split('<')[0]
d['user'] = review.split('<a title="')[1].split('"')[0]</pre>
```

• Again we did this by noting that the "date" and "user" fields appear inside certain html elements:

```
1241 
1242 
1243 
1244 
1244 
1244 
1244 
1245 
1246 
1246 
1246 
1247 
1247 
1248 
1248 
1249 
1249 
1240 
1240 
1241 
1242 
1243 
1244 
1244 
1245 
1245 
1246 
1245 
1246 
1247 
1246 
1247 
1248 
1247 
1248 
1248 
1249 
1249 
1249 
1240 
1240 
1240 
1241 
1241 
1242 
1243 
1244 
1245 
1245 
1245 
1245 
1245 
1246 
1245 
1246 
1247 
1248 
1248
```

#### Let's look at it line-by-line:

- Next we extract the "shelves" the book belongs to
- This follows the same idea, but in a "for" loop since there can be many shelves per book:



 Here we use a try/except block since this text will be missing for users who didn't add the book to any shelves

#### Next let's extract the review contents:

```
In [8]: def parseReview(review):
             d = \{\}
             d['stars'] = review.split('<span class=" staticStars" title="')[1].split('"')[0]</pre>
             d['date'] = review.split('<a class="reviewDate')[1].split('>')[1].split('<')[0]</pre>
             d['user'] = review.split('<a title="')[1].split('"')[0]</pre>
             shelves = []
             try:
                 shelfBlock = review.split('<div class="uitext greyText bookshelves">')[1].split('</div')[0]</pre>
                 for s in shelfBlock.split('shelf=')[1:]:
                     shelves.append(s.split('"')[0])
                 d['shelves'] = shelves
             except Exception as e:
                 pass
          reviewBlock = review.split('<div class="reviewText stacked">')[1].split('</div')[0]</pre>
             d['reviewBlock'] = reviewBlock
             return d
```

#### Now let's look at the results:

In [9]: reviewDict = [parseReview(r) for r in reviews]

#### In [10]: reviewDict[0]

Out[10]: {'date': 'May 02, 2010', 'reviewBlock': '\\n

> \\n<span 1d="freeT opeful dreamer with \\\'< soul on that elusive mone ><img src="https://i.gr-a pg" width="400" class="gr possess - and the tragedy y.</b> <br><br>Just like tyle="display:none"><br>0 ened sensitivity to the p een light - a dream that ages/S/compressed.photo.g mg"><br><br><br>>b>Jay Gatsby t was a wrong dream colli udini - the association t ven the power of most cou

ewBlock': '\\n <span id="reviewTextContainer101057684" class="readable"\\n >\\n \\n<span id="freeT<u>extContainer7513160808421149349"><br>0h Gatsby, you old</u> sport, you poor semi-delusionally h

- Looks okay, but the review block itself still contains embedded html (e.g. images etc.)
- How can we extract just the text part of the review?

sport, you poor semi-delusionally n i>\\\', focusing your whole self and you are \*this\* close to it. <br>dimages/1380334543i/693798.\_SX540\_.j eam with the passion and courage few hat was even more wrong - and deadl n id="freeText7513160808421149349" s eful dreamer with \\\'<i>some height oul on that elusive money-colored gr <img src="https://i.gr-assets.com/im g" width="400" class="gr-hostedUserI bossess - and the tragedy was that i .</b> <br>dossess - and the tragedy was that i .</b> <br>dossess - and escape. Except e be the world, our past, and ourselve

s, giving rise to one of the most famous closing lines of a novel.<br/>
blockquote>\\n <i>\\\'Gatsby believed in the gre en light, the orgastic future that year by year recedes before us. It eluded us then, but that\\xe2\\x80\\x99s no ma

### The BeautifulSoup library

Extracting the text contents from the html review block would be extremely difficult, as we'd essentially have to write a html parser to capture all of the edge cases

Instead, we can use an existing library to parse the html contents: **BeautifulSoup** 

### Code: parsing with BeautifulSoup

#### BeautifulSoup will build an element tree from the html passed to it. For the moment, we'll just use it to extract the text from a html block

In [11]: from bs4 import BeautifulSoup

In [12]: soup = BeautifulSoup(reviewDict[0]['reviewBlock'])

In [13]: soup.text

Out[13]: "\\n \\n \\n0h Gatsby, you old sport, you poor semi-delusionally hopeful dreamer with \\'som e heightened sensitivity to the promises of life\\', focusing your whole self and soul on that elusive money-colored green light - a dream that shatters just when you are \*this\* close to it. Jay Gatsby, who dreamed a dream with the p assion and courage few possess - and the tragedy was that it was a wrong dream colliding with reality that was even more wrong - and deadly. Just like the Great Houdini - the association the\\n 0h Gatsby, you old sport, you poor se mi-delusionally hopeful dreamer with \\'some heightened sensitivity to the promises of life\\', focusing your whole self and soul on that elusive money-colored green light - a dream that shatters just when you are \*this\* close to i t. Jay Gatsby, who dreamed a dream with the passion and courage few possess - and the tragedy was that it was a wron g dream colliding with reality that was even more wrong - and deadly. Just like the Great Houdini - the association the title of this book so easily invokes - you specialized in illusions and escape. Except even the power of most co urageous dreamers can be quite helpless to allow us escape the world, our past, and ourselves, giving rise to one of

### The BeautifulSoup library

In principle we could have used BeautifulSoup to extract *all* of the elements from the webpage

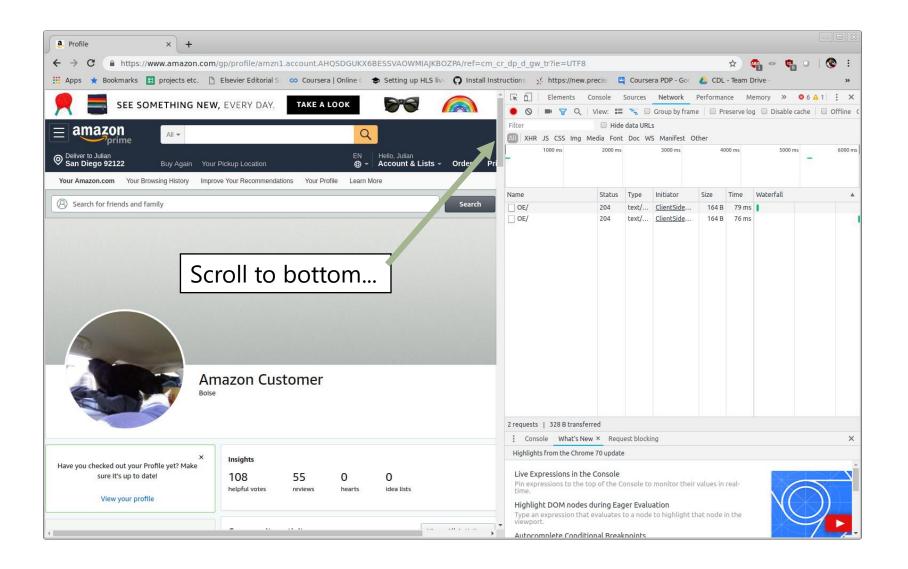
However, for simple page structures, navigating the html elements is not (necessarily) easier than using primitive string operations

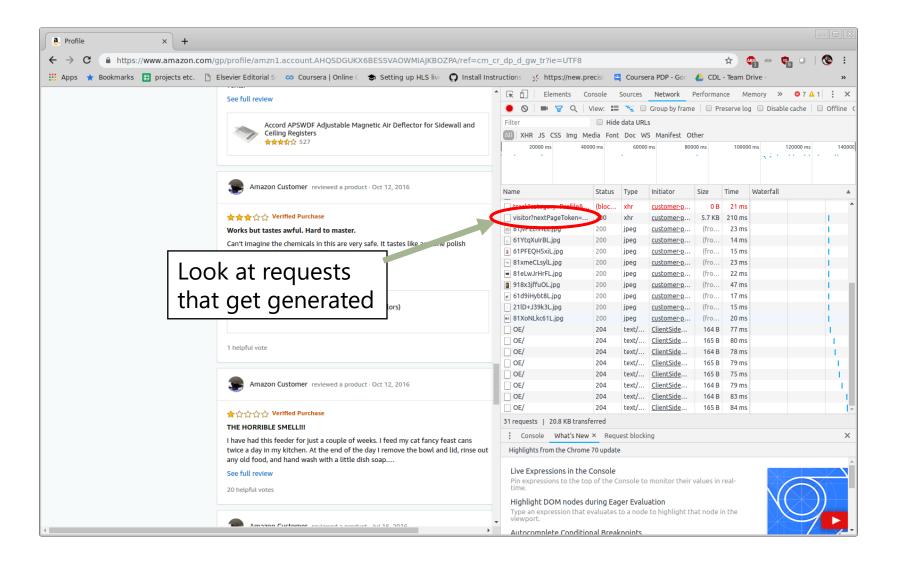
### Advanced concepts...

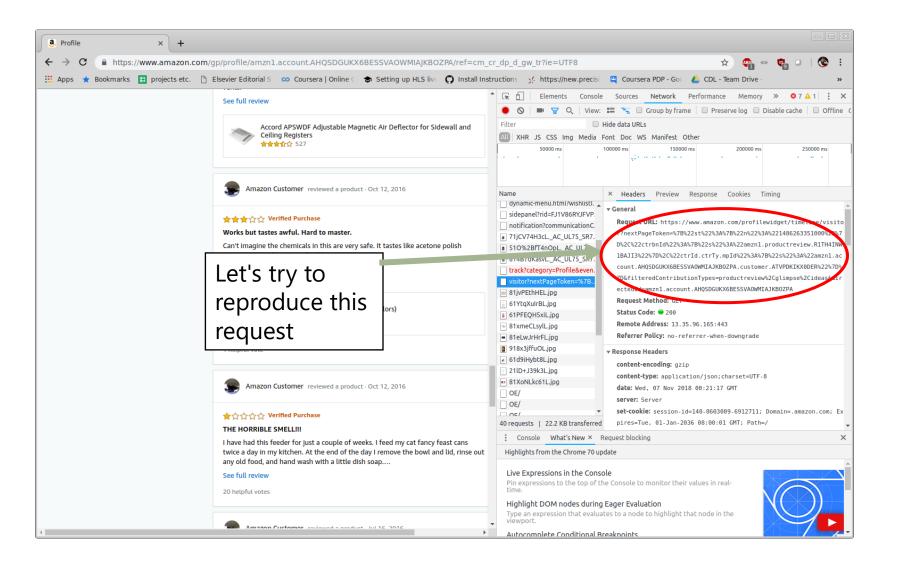
## 1. What if we have a webpage that loads content **dynamically?**

(e.g. <u>https://www.amazon.com/gp/profile/amzn1.account.AHQSDGUKX6</u> <u>BESSVAOWMIAJKBOZPA/ref=cm\_cr\_dp\_d\_gw\_tr?ie=UTF8</u>)

- The page (probably) uses javascript to generate requests for new content
- By monitoring network traffic, perhaps we can view and reproduce those requests
- This can be done (e.g.) by using the Developer Tools in chrome







a, Profile	× 🧕 https://v	www.amazon.com/prox	+					
- > C 🔒 https://w	ww.amazon.com/p	profilewidget/timeline/vi	sitor?nextPageToken=%7B'	'st"%3A%7B"n"%3A"1486263	351000"%7D%2C"ctrbn	11d"%3A%7B"s"%3A"am	nzn1 🛧 💩 👁 🛡	⊘
Apps 🛧 Bookmarks 🖽	projects etc. 🗋 🛾	Elsevier Editorial S 🛛 🚥 Co	ursera   Online 🤇 🐟 Setting u	p HLS live 🛛 🗛 Install Instruction	s 🔀 https://new.precisi	🚆 Coursera PDP - Goo	🝐 CDL - Team Drive -	;
"s\":\"amznl.account.AH "id":"amznl.productrevi uldn't believe it!!!"," ailable. I tried everyt uldn't believe it!!!"," ailable. I tried everyt eks on this food I was at did the trick after en off of insulin for a uly going to Keep her o ncy Feast Medleys","lin zon.com/images/I/81jvP attachments":[{"asin":" zon.com/images/I/81jvP id":"amznl.productrevie IGHT scent?", "text":"Lo uid/dp/B0e01VA80C","ima id":"amznl.productrevie ms.", "text":"Only one szon.com/images/I/61PFE ttachments":[{"asin":" azon.com/images/I/61PFE tid:":"amznl.productrevie ms.", "text":"Only one szon.com/images/I/61PFE ttachments":[{"asin":"B azon.com/images/I/61PFE id":"amznl.productrevie ms.", "text":"Text":"N	<pre>U0SDGUKX6BESSYAOW eww,RIUMY62M00J20' text":" spent th hing even whole g iskeptical when i e only diet. draw and the second all to take her all this time. Th lmost a year and uut of diabetic tr hk":"/Purina-Fancy B001STPJ00","teit B001STPJ00","teit Heil Ego", "revi ww.R36X5Y014H4KAX: HVIIFBL.j0g","revi B003SY014H4KAX: Wu Fb natural i ile so I don't u se Skin and Strei HVIIFBL.j0g","revi B003SY019,","evi B003SY019,","revi B003SY019,"</br></pre>	<pre>MIAJKB02PA.customer.AT. "externalId":"RIUMYE hree and a half years ground raw rabbit. Grc I saw all the reviews /&gt; I was amazed t off insulin all toget he classic pate ones a has not needed freque rouble and I am so so y-Feast-Classic-Collee iewCount":1774, "averag le":"Purina Fancy Feas iewCount":1774, "averag le":"Purina Fancy Feas iewCount":1774, "averag e":"ReternalId": "R36X5Y ngredients! My husband sei ton my chest or a tch Marks, 4-Fluid Our ges-na.ssl-images-amaz ","externalId": "R3E603 ic in these but they h s-758399753228-OralBic iewCount":320, "average e":"NOW OralBiotic,60 e":"NOW OralBiotic,60 ef Adjustable Magnetic</br></pre>	<pre>VPDKIKX0DER\"]}", "contri 2M00J20", "sortTimestamp" (and a small fortune) try ss. I wanted to get my d saying it stopped type 2 hat in the first few weel her. My vet follows my o' re the only ones that di ent vet check ups or blad grateful to have finally tion/dp/B001STJJO", "ima eRating":4.5, "attachment" '0I4H4KAX", "sortTimestamp I thinks it smells too st its avoid smelling it icce", "link":"/Farth-Mama-1 kating":4.2, "attachment '1 Help Ease Skin and '2UM2M00H", "sortTimestamp lelp my gums and my breat '2UM2M00H", "sortTimestamp lelp my gums and my breat '2UM2M00H", "sortTimestamp lelp my gums and my breat '2UM2M00H", "sortTimestamp 'storog enough to keep ti strong enough to keep ti 'strong enough to keep ti 'strong</pre>	5529474000\"},\"ctrbnId\": butions": 1476307421000, "rating":5.( jing the most expensive gra- labetic cat in remission ar diabetes in cats from proy- ks of being on this food hu- dic at closely and was amaz- dn't spike her blood sugar. Jer infection treatments oi found this!,",writtenInLar ge":"https://images-na.s6L Type":"PRODUCT", "prime":fal a-Fancy-Feast-Classic-Coll Type":"PRODUCT","prime":fal sll day.", writtenInLangur 3elly-Stretch-4-Fluid/dp/BK pe":"PRODUCT","prime":fal Stretch Marks, 4-Fluid/dp/BK Stretch Marks,	<pre>0, "visibility":"public ainfree, raw, organic, d off of insulin so b pressing and even reve er blood sugar was too zed that of all things . c/sbr /&gt;Sbe is 17 r insulin once. scr /&gt;scr //scr /</pre>	<pre>","helpfulVotes":24,' and prescription dia vad.  <b< td=""><td><pre>title":"Reversed diabet abetic cat foods wet and an RN so I checked her b and it took my cat a go several times a week an counter grocery store br eighs 9 lbs instead of 1 long to be convinced th ("asin":"B001STPJJO","ti "zebra":false,"type":"pro ages-na.ssl-images- "title":"How is that a L y Butter uses. It smells ages-na.ssl-images- "title":"Fouly. So do ity.ebr/&gt;sebrey:"pro DUCT","prime":false]], "title":"Fruity. So do ity.ebr/&gt;chr/stle":"NOW zebra":false,"type":"pro //images-na.ssl-images- "title":Little flimsy b "product":</pre></td><td>es!!! I just dry lood sugar a od 3 months d after 4 and food 4. She has at it was tle":"Purina oductreview" IGHT or strong even ":"Earth ductreview", r breath and ith multiple ductreview",</td></b<></pre>	<pre>title":"Reversed diabet abetic cat foods wet and an RN so I checked her b and it took my cat a go several times a week an counter grocery store br eighs 9 lbs instead of 1 long to be convinced th ("asin":"B001STPJJO","ti "zebra":false,"type":"pro ages-na.ssl-images- "title":"How is that a L y Butter uses. It smells ages-na.ssl-images- "title":"Fouly. So do ity.ebr/&gt;sebrey:"pro DUCT","prime":false]], "title":"Fruity. So do ity.ebr/&gt;chr/stle":"NOW zebra":false,"type":"pro //images-na.ssl-images- "title":Little flimsy b "product":</pre>	es!!! I just dry lood sugar a od 3 months d after 4 and food 4. She has at it was tle":"Purina oductreview" IGHT or strong even ":"Earth ductreview", r breath and ith multiple ductreview",
<pre>ttachments":[("asin":"B gisters/dp/B0052E68R0", id":"amarl.productrevie rd to master.","text";" lor all day on may pale r /&gt;I do not have much rt of the lip is or the ve leared how to get it me.","writtenInLanguage Lors/dp/B01AK00QUU","im azon.com/images/I/81eLw</pre>	<pre>CLSy[L.jpg,"rev; 10052EE8R0", "title "image": "https:// w.R1D6D2DL4RW915' Can't imagine the lips and the dau natural color to pigment is notit looking good I v Code": "en US", ii nage": "https://ima J7HFL.jpg", "retty B01AK00QUU", "titl</pre>	<pre>iewCount":527, "aveTage e":"Accord APSWDF Adju /images-na.ssl-images- ","externalId":"R13G02 e chemicals in this ar rkest colors used in t my lips and this just ceably uneven in darkr will probably just ree mages":[],"product":{" ages-na.ssl-images- iewCount":1014, "averag lew":[Berrisom] Oops M</pre>	Rating":3.7, "attachment"; istable Magnetic Air Defli amazon.com/images/I/81xm DL4RW915", "sortTimestamp e very safe. It tastes l the evening leave my lips give a nice bit of pink tess. Like I have been dr: order sexy red and bubble asin": "B01AK0DQUU", "title peRating":3.4, "attachment" y Lip Tint Pack (6 Color:	<pre>ype":"PRODUCT","prime":fals ector for Sidewall and Cei ecLsyL.jpg","reviewCount" ":1476305637000,"rating":3. ike acetone polish remover slightly pink all day the to them. It's hard to put inking wine all night. I du pink instead of the who pa ":"[Berrisom] Oops My Lip Type":"PRODUCT","prime":fal</pre>	<pre>Ling Registers", "link" :527, "averageRating":3 o, "visibility": "publi even after you pull o next day. I am super on at first and I hav on't leave it on more ack. Unless I find a b Tint Pack (6 Colors)" lse}, "verifiedPurchase</pre>	:"/Accord-APSWDF-Adj .7, "attachmentType": .7, "attachmentType": .7, "helpfulVotes":1, iff but it works! but it works! or pale skinned and do r e to be careful not f than 10 min or it ge! etter lip staining pu 	ustable-Deflector- "PRODUCT","prime":false} "title":"Works but taste /> I just wanted a not wear much makeup if to place it where the we 	]}, s awful. hint of any.  t and dry Now that I

### Advanced concepts...

2. What if we require **passwords**, **captchas**, or **cookies**?

- You'll probably need to load an actual browser
- This can be done using a headless browser, i.e., a browser that is controlled via Python
   I usually use

splinter (https://splinter.readthedocs.io/en/latest/)

 Note that once you've entered the password, solved the captcha, or obtained the cookies, you can normally continue crawling using the *requests* library



- The urllib library can be used to request data from the web as if it is a file, whereas
   BeautifulSoup can be used to convert the data to structured objects
  - Parsing can also be achieved using primitive string processing routines
- Make sure to check the page's terms of service first!

### Learning Outcomes

### Introduced programmatic approaches to collect datasets from the web

### Web Mining and Recommender Systems

Parsing time and date data



# Show how to parse time and date data

### Time and date data

Dealing with time and date data can be difficult as string-formatted data doesn't admit easy comparison or feature representation:

- Which date occurs first, 4/7/2003 or 3/8/2003?
- How many days between 4/5/2003 7/15/2018?
- e.g. how many hours between 2/6/2013 23:02:38 - 2/7/2013 08:32:35?

### Time and date data

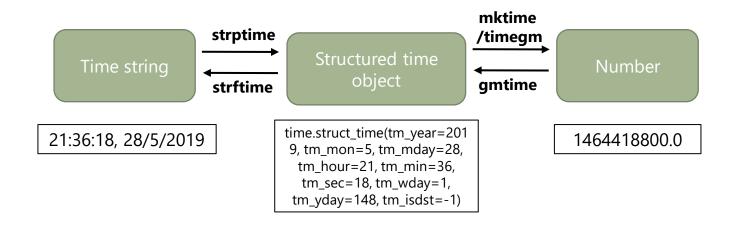
## Most of the data we've seen so far include plain-text time data, that we need to carefully manipulate:

{'business\_id': 'FYWN1wneV18bWNgQjJ2GNg', 'attributes': {'BusinessAcceptsCreditCards': True, 'AcceptsInsurance': True, 'ByAppointmentOnly': True}, 'longitude': -111.9785992, 'state': 'AZ', 'address': '4855 E Warner Rd, Ste B9', 'neighborhood': '' 'loity'. 'Nhuatukee', 'hours': {'Tuesday': '7:30-17:00', 'Wednesday': '7:30-17:00', 'Thursday': '7:30-17:00', 'Friday': '7:30-17:00', 'Monday': '7:30-17:00'}, 'postal\_code': '85044', 'review\_count': 22, 'stars': 4.0, 'categories': ['Dentists', 'General Dentistry', 'Health & Medical', 'Oral Surgeons', 'Cosmetic Dentists', 'Orthodontists'], 'is\_open': 1, 'name': 'Dental by Design', 'latitude': 33.3306902} Here we'll cover a few functions:

- Time.strptime: convert a time **string** to a structured time **object**
- Time.strftime: convert a time **object** to a **string**
- Time.mktime / calendar.timegm: convert a time **object** to a **number**
- Time.gmtime: convert a number to a time object

### Time and date data

#### Here we'll cover a few functions:



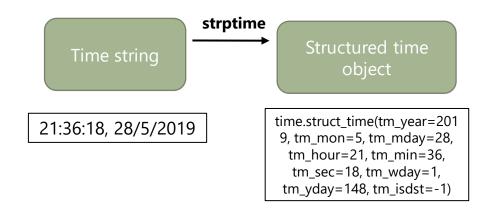
### Concept: Unix time

Internally, time is often represented as a number, which allows for easy manipulation and arithmetic

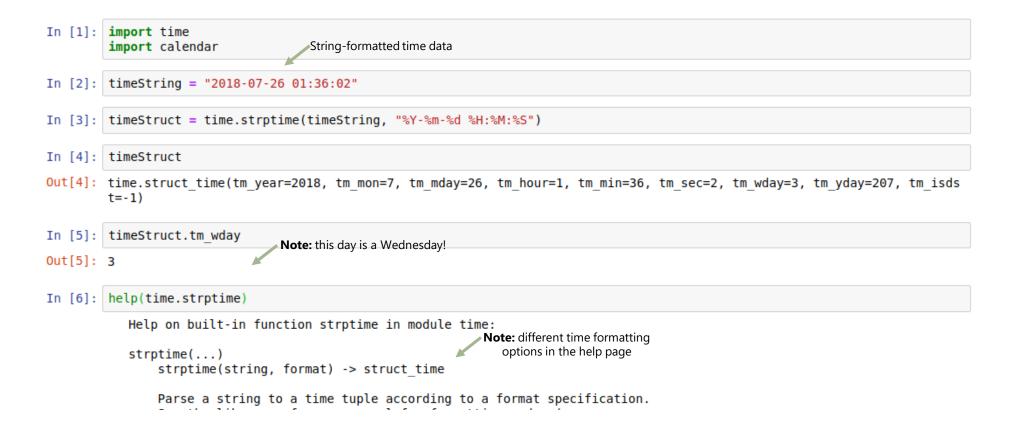
- The value (Unix time) is the number of seconds since Jan 1, 1970 in the UTC timezone
- so I made this slide at 1532568962 = 2018-07-26 01:36:02 UTC (or 18:36:02 in my timezone)
- But real datasets generally have time as a "human readable" string
- Our goal here is to convert between these two formats

### strptime

## First, let's look at converting a string to a structured object (strptime)



# Code: time.strptime()



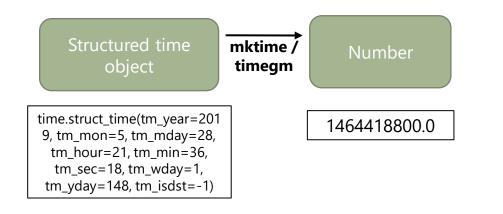
#### strptime

# Strptime is convenient when we want to extract **features** from data

- E.g. does a date correspond to a weekday or a weekend?
- Converting month names or abbreviations (e.g. "Jan") to month numbers
- Dealing with mixed-format data by converting it to a common format
- But if we want to perform arithmetic on timestamps, converting to a number may be easier

# time.mktime and calendar.timegm

# For this we'll use mktime to convert our structured time object to a number:



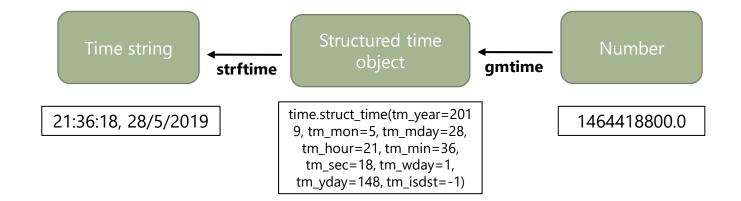
# Code: time.mktime() and calendar.timegm()

	Structured time data from previous slide
In [7]:	<pre>t1 = calendar.timegm(timeStruct)</pre>
In [8]:	<pre>t2 = time.mktime(timeStruct)</pre>
In [9]:	t1, t2
Out[9]:	(1532568962, 1532594162.0)
In [10]:	t1 + 60*60*24*5
Out[10]:	1533000962 Five days later

- time.mktime() allows us to convert our structured time object to a number
- **NOTE:** mktime assumes the structure is a *local* time whereas timegm assumes the structure is a *UTC* time
- This allows for easy manipulation, arithmetic, and comparison (e.g. sorting) of time data

#### time.strftime and time.gmtime

Finally, both of these operations can be *reversed*, should we wish to format time data as a string or structure



#### Code: time.strftime() and time.gmtime()

In [11]:	time.gmtime(t1 +	⊦ 60*60*24*5)	Five days later than the previous time
----------	------------------	---------------	--

- In [12]: time.strftime("%Y-%m-%d %H:%M:%S", time.gmtime(t1 + 60\*60\*24\*5))
- Out[12]: '2018-07-31 01:36:02'

 These methods can be used to put adjusted times back into string format

# Learning Outcomes

#### Introduced various methods to parse time and date data

Web Mining and Recommender Systems

Introduction to Matplotlib

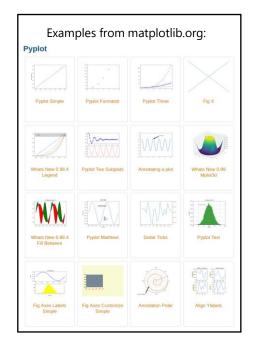


Introduce Matplotlib for plotting and visualizing data

# Matplotlib

Matplotlib is a powerful library that can be used to generate both quick visualizations, as well as publication-quality graphics

- We'll introduce some of its most basic functionality (via pyplot), such as bar and line plots
- Examples (with code) of the types of plots that can be generated are available on <u>https://matplotlib.org/</u>



#### Code: generating some simple statistics

# First, let's quickly compile some statistics from (e.g.) Yelp's review data

In [1]: import json import time path = "datasets/yelp\_data/review.json" f = open(path, 'r')

- In [3]: datasetWithTimeValues = []

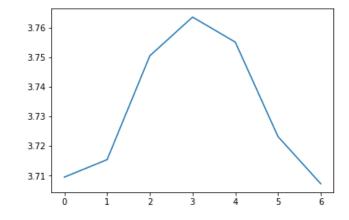
In [4]: for d in dataset: d['date'] d['timeStruct'] = time.strptime(d['date'], "%Y-%m-%d") d['timeInt'] = time.mktime(d['timeStruct']) datasetWithTimeValues.append(d)

#### Code: generating some simple statistics

- In [5]: from collections import defaultdict
- In [6]: weekRatings = defaultdict(list)
- In [7]: for d in datasetWithTimeValues: day = d['timeStruct'].tm\_wday weekRatings[day].append(d['stars'])
- In [8]: weekAverages = {}
- In [9]: for d in weekRatings: weekAverages[d] = sum(weekRatings[d]) \* 1.0 / len(weekRatings[d])
- In [10]: weekAverages
- Out[10]: {0: 3.7094594594594597,
  - 1: 3.715375187253166,
    - 2: 3.750551876379691,
    - 3: 3.763665361751486,
    - 4: 3.7551891653172382,
    - 5: 3.7231843981953134, Average ratings per day of week
    - 6: 3.7072147651006713}

# Code: drawing a simple plot

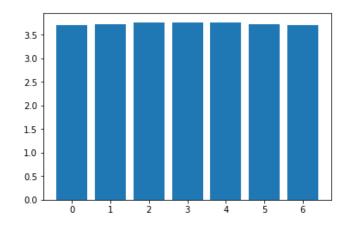
- In [12]: Y = [weekAverages[x] for x in X]
- In [13]: import matplotlib.pyplot as plt
- In [14]: plt.plot(X, Y)
- Out[14]: [<matplotlib.lines.Line2D at 0x7fc15a615a20>]



# Code: bar plots

In [15]: plt.bar(X, Y)

Out[15]: <Container object of 7 artists>

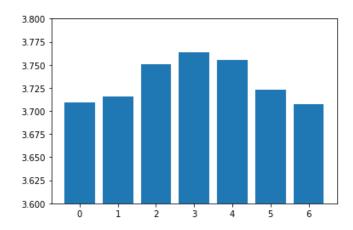


• Looks right, but need to zoom in more to see the detail

# Code: bar plots

In [16]: plt.ylim(3.6, 3.8)
 plt.bar(X, Y)

Out[16]: <Container object of 7 artists>

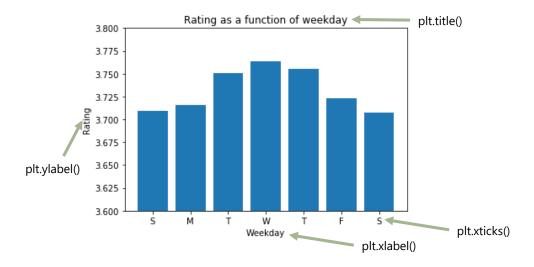


• Next let's add some details

# Code: bar plots

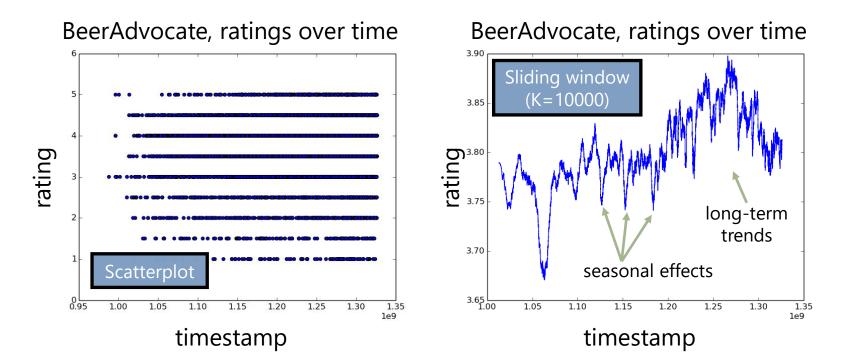
In [17]: plt.ylim(3.6, 3.8)
 plt.xlabel("Weekday")
 plt.ylabel("Rating")
 plt.xticks([0,1,2,3,4,5,6],['S', 'M', 'T', 'W', 'T', 'F', 'S'])
 plt.title("Rating as a function of weekday")
 plt.bar(X, Y)

Out[17]: <Container object of 7 artists>



# Example: sliding windows

# Also useful to plot data:



Code on course webpage

# Learning Outcomes

• Briefly introduced Matplotlib

# Web Mining and Recommender Systems

Gradient descent in tensorflow



#### • Introduce Tensorflow

 Show how high-level libraries can help to automate gradient-based optimization

## Tensorflow

**Tensorflow**, though often associated with deep learning, is really just a library that simplifies gradient descent and optimization problems, like those we've already implemented

Most critically, it computes gradients **symbolically**, so that you can just specify the objective, and Tensorflow can run gradient descent

Here we'll reimplement some of our previous gradient descent code in tensorflow

Reading the data is much the same as before (except that we first import the tensorflow library)

```
In [1]: import tensorflow as tf
```

```
In [2]: path = "datasets/PRSA_data_2010.1.1-2014.12.31.csv"
f = open(path, 'r')
```

```
In [3]: dataset = []
header = f.readline().strip().split(',')
for line in f:
    line = line.split(',')
    dataset.append(line)
```

```
In [4]: header.index('pm2.5')
```

```
Out[4]: 5
```

In [5]: dataset = [d for d in dataset if d[5] != 'NA']

#### Next we extract features from the data

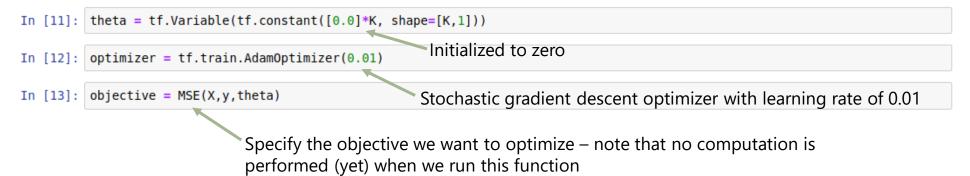
```
In [6]: def feature(datum):
    feat = [1, float(datum[7]), float(datum[8]), float(datum[10])] # Temperature, pressure, and wind speed
    return feat
In [7]: X = [feature(d) for d in dataset]
y = [float(d[5]) for d in dataset]
In [8]: y = tf.constant(y, shape=[len(y),1])
In [9]: K = len(X[0])
```

Note that we convert *y* to a native tensorflow vector. In particular we convert it to **column** vector. We have to be careful about getting our matrix dimensions correct or we may (accidentally) apply the wrong matrix operations.

Next we write down the objective – note that we use native tensorflow operations to do so

In [10]: def MSE(X, y, theta):
 return tf.reduce\_mean((tf.matmul(X,theta) - y)\*\*2)

Next we setup the variables we want to optimize – note that we explicitly indicate that these are **variables** to be optimized (rather than constants)



Boilerplate for initializing the optimizer...

- In [15]: init = tf.global\_variables\_initializer()
- In [16]: sess = tf.Session()
   sess.run(init)

#### Run 1,000 iterations of gradient descent:

	<pre>= sess.run([train, objective]) objective = " + str(cvalues[1]))</pre>
object	ve = 7836.5107
object	ve = 7836.5103
object	ve = 7836.5107
object	ve = 7836.5107
	ve = 7836.5107
object	ve = 7836.5103
object	ve = 7836.5103
object	ve = 7836.5093
	ve = 7836.5093
object	ve = 7836.5093

#### Print out the results:

In [18]: with sess.as\_default():
 print(MSE(X, y, theta).eval())
 print(theta.eval())

7836.5093 [[ 0.23223479] [-0.89481604] [ 0.11925128] [-0.4959688 ]]



Note that in contrast to our "manual" implementation of gradient descent, many of the most difficult issues were taken care of for us:

- No need to compute the gradients tensorflow does this for us!
- Easy to experiment with different models
- Very fast to run 1,000 iterations, especially with GPU acceleration!

# Other libraries

Tensorflow is just one example of a library that can be used for this type of optimization. Alternatives include:

- Theano <u>http://deeplearning.net/software/theano/</u>
  - Keras <u>https://keras.io/</u>
  - Torch <u>http://torch.ch/</u>

• Etc.

Each has fairly similar functionality, but some differences in interface

# Learning Outcomes

#### • Introduced Tensorflow