Data Preparation For Data Science

MINKAI WU

Outline

Data Acumen

- Data Science Process
- Data Quality
- Data Source
- Data File Format
- Data Types

Data Cleaning

- Missing Data
- Invalid Data
- Feature extraction
- Demo

Web Data Preparation

- Understanding the HTML Page Structure

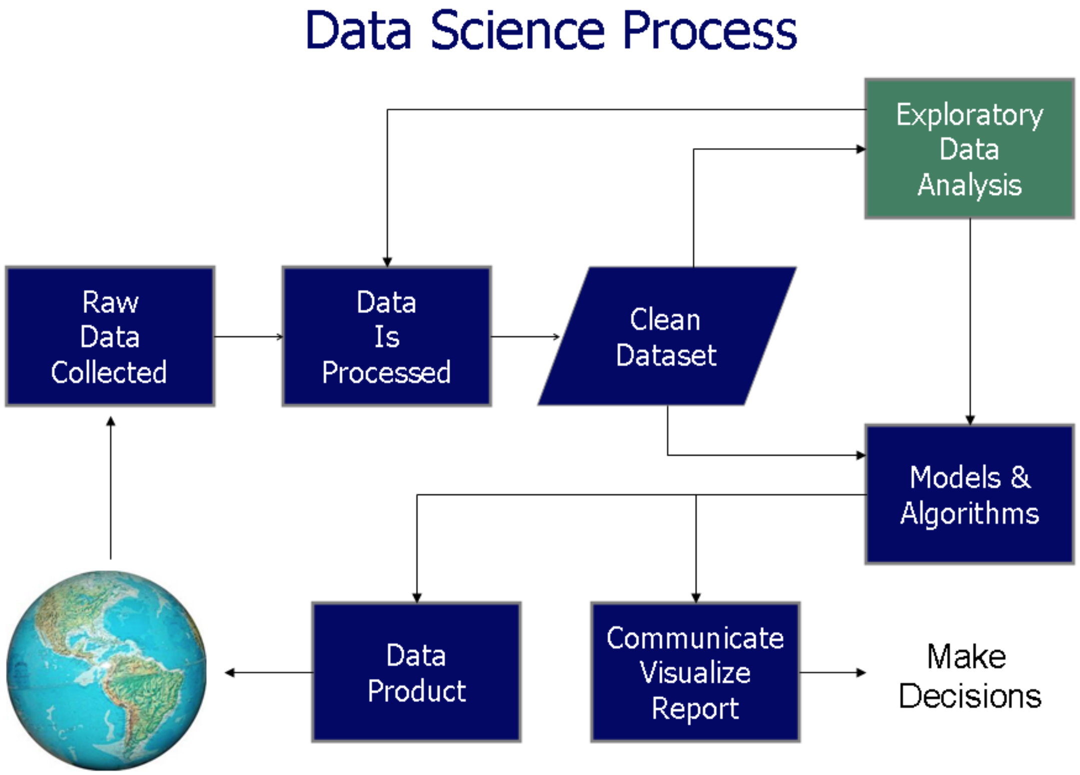
□ Python and Regular Expressions to clean data

□ Python and Beautiful Soup to collect data

□ Demo

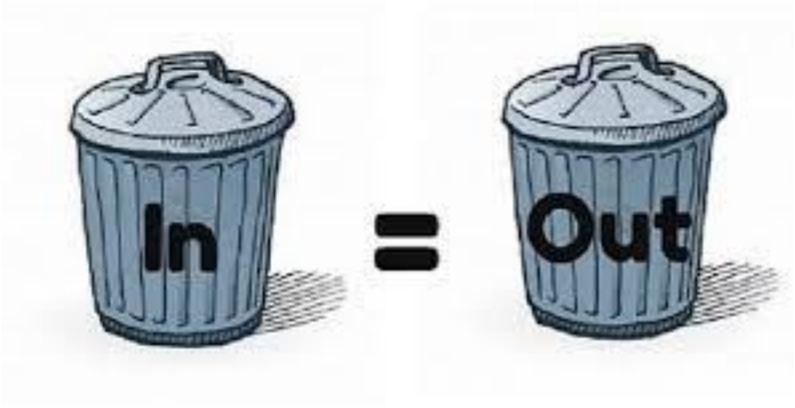
Introduction: Data Science Process

1. Problem Statement
2. Data Collection & Storage
3. **Data Preparation**
   1. **Access Data**
   2. **Clean Data**
   3. **Transform Data**
4. Data Analysis & Visualization
5. Modeling
6. Presentation or Productize



Introduction: Data Quality Issues

- Incorrect/Invalid Entry
  - age = 203; gender = X; price = -100; weekday=8
- Missing Data
  - N/A; Null; ” “; Unknown
- Unstructured Data
  - merged cell; double header; html
- Conflicting Data
  - revenue =1000; unit = 0
- Duplicates
  - double loading; double counting
- Outlier
  - House Price = $1B

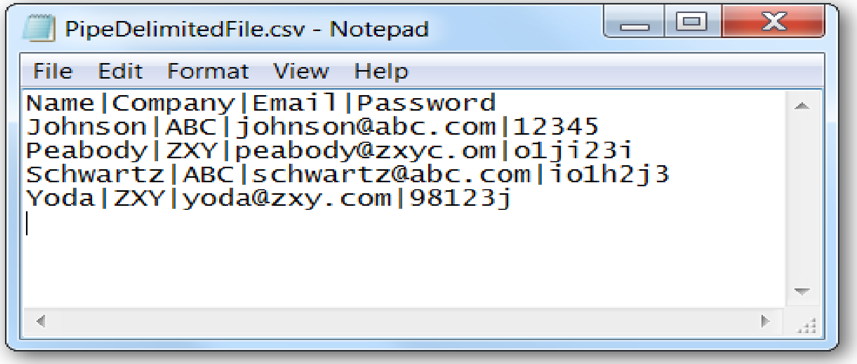


Introduction: Data Source

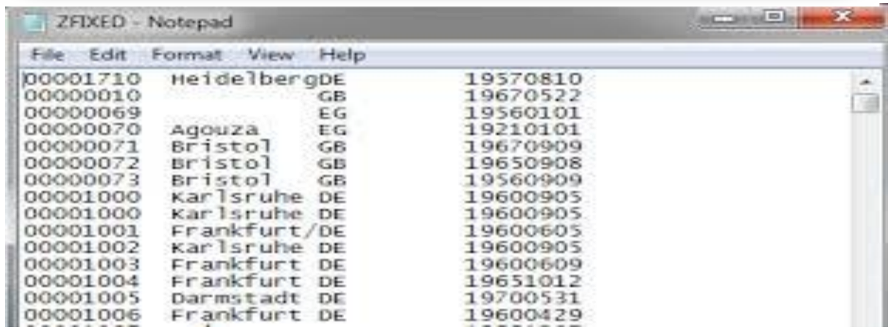
- Data File
- Database/Data Warehouse
- Web Data
- Big Data Platform

Data File: Structured Data

- Excel:
  - Most common; most problematic
- Delimited format
  - Most common; most preferred
  - Common delimited (csv); tab delimited(tsv); “|” delimited
  - Problem: delimiter in data field. E.g. Los Angles, CA
  - Problem: encoding
- Fixed length
  - Every column has fixed length
  - Problem: Oversized column



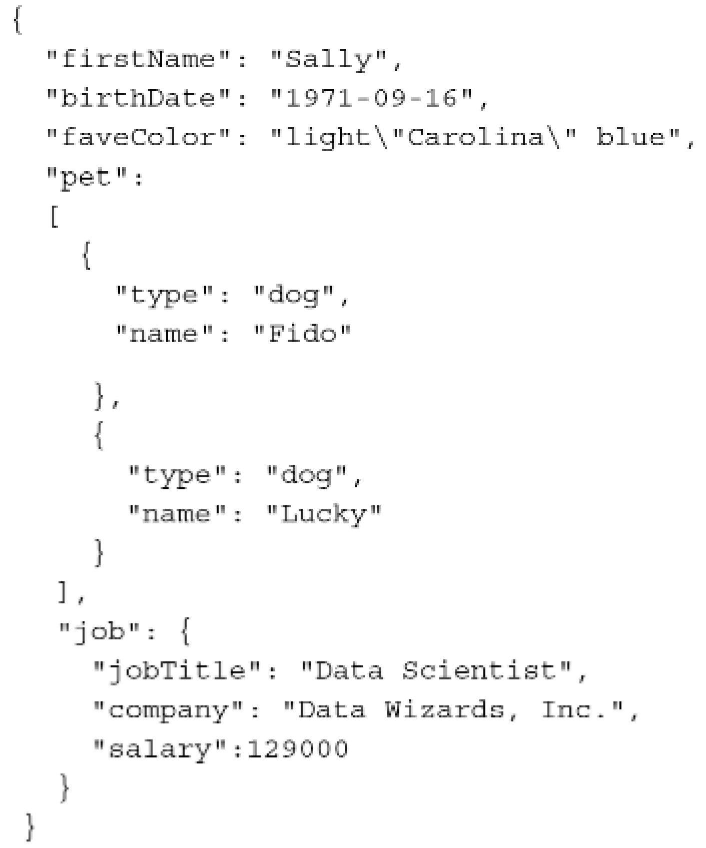
```
Name|Company|Email|Password
Johnson|ABC|johnson@abc.com|12345
Peabody|ZXY|peabody@zxyc.om|o1ji23i
Schwartz|ABC|schwartz@abc.com|io1h2j3
Yoda|ZXY|yoda@zxy.com|98123j
```



```
00001710  HeidelbergDE          19570810
00000010            GB          19670522
00000069            EG          19560101
00000070  Agouza    EG          19210101
00000071  Bristol   GB          19670909
00000072  Bristol   GB          19650908
00000073  Bristol   GB          19560909
00001000  Karlsruhe DE          19600905
00001000  Karlsruhe DE          19600905
00001001  Frankfurt/DE          19600605
00001002  Karlsruhe DE          19600905
00001003  Frankfurt DE          19600609
00001004  Frankfurt DE          19651012
00001005  Darmstadt DE          19700531
00001006  Frankfurt DE          19600429
```

Data File: JSON

- JavaScript Object Notation
- Semi- Structured
- Attributes are on the left-hand side of colon
- Values are on the right-hand side of colon
- Attributes are separated by a comma
- Multi-value attributes are as hierarchical values

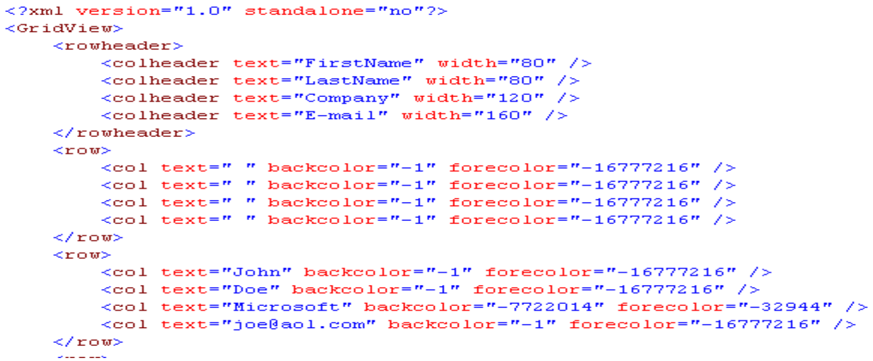


```
{
  "firstName": "Sally",
  "birthDate": "1971-09-16",
  "faveColor": "light\"Carolina\" blue",
  "pet":
  [
    {
      "type": "dog",
      "name": "Fido"

     },
     {
       "type": "dog",
       "name": "Lucky"
     }
   ],
   "job": {
     "jobTitle": "Data Scientist",
     "company": "Data Wizards, Inc.",
     "salary":129000
   }
 }
```

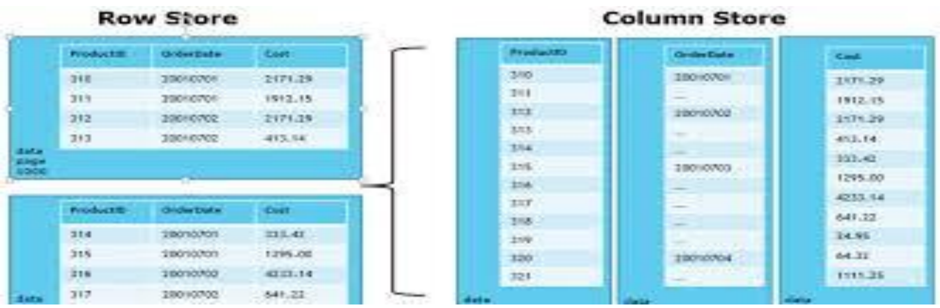
Data File: XML and Parquet

- XML
  - Extensible Markup Language
  - Semi-Structured
  - Most common for data exchange



```xml
<?xml version="1.0" standalone="no"?>
<GridView>
    <rowheader>
        <colheader text="FirstName" width="80" />
        <colheader text="LastName" width="80" />
        <colheader text="Company" width="120" />
        <colheader text="E-mail" width="160" />
    </rowheader>
    <row>
        <col text=" " backcolor="-1" forecolor="-16777216" />
        <col text=" " backcolor="-1" forecolor="-16777216" />
        <col text=" " backcolor="-1" forecolor="-16777216" />
        <col text=" " backcolor="-1" forecolor="-16777216" />
    </row>
    <row>
        <col text="John" backcolor="-1" forecolor="-16777216" />
        <col text="Doe" backcolor="-1" forecolor="-16777216" />
        <col text="Microsoft" backcolor="-7722014" forecolor="-32944" />
        <col text="joe@aol.com" backcolor="-1" forecolor="-16777216" />
    </row>
```

- Parquet
  - Column Store
  - Spark



Row Store

data page 1000

ProductID	OrderDate	Cost
310	20010701	2171.29
311	20010701	1912.15
312	20010702	2171.29
313	20010702	413.14

data

ProductID	OrderDate	Cost
314	20010701	333.42
315	20010701	1295.00
316	20010702	4233.14
317	20010702	641.22

Column Store

data

ProductID
310
311
312
313
314
315
316
317
318
319
320
321

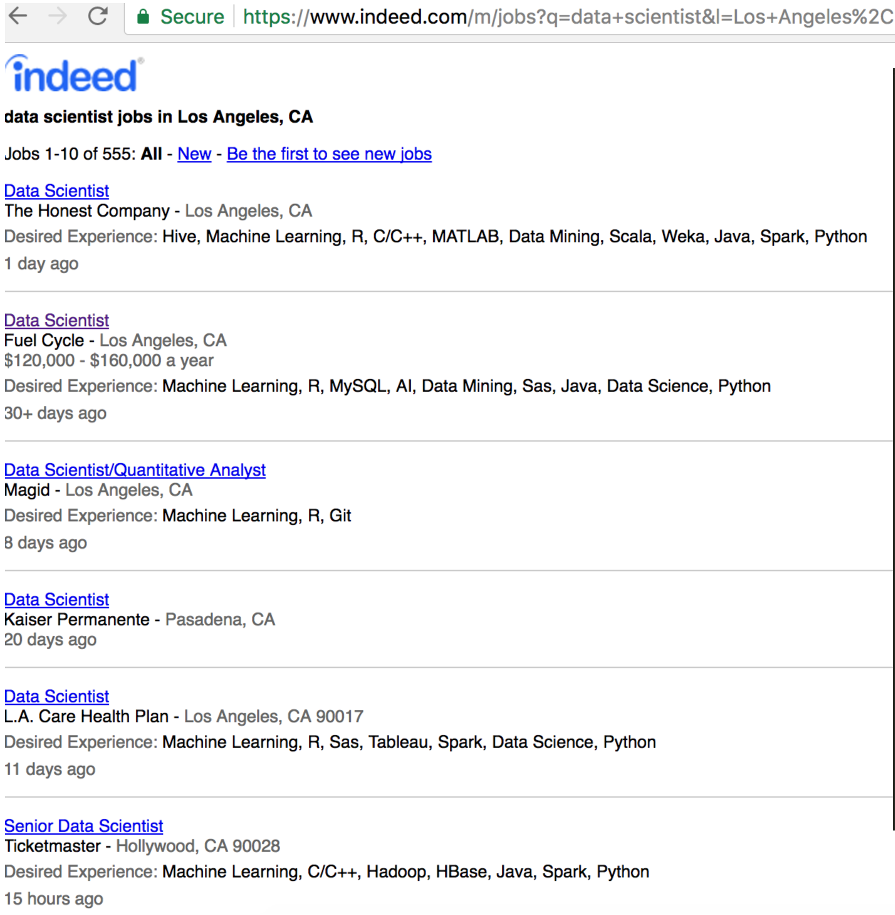
data

OrderDate
20010701
—
20010702
—
—
20010703
—
—
—
—
20010704
—

data

Cost
2171.29
1912.15
2171.29
413.14
333.42
1295.00
4233.14
641.22
24.95
64.32
1111.25

Web Data: HTML - Unstructured



Secure https://www.indeed.com/m/jobs?q=data+scientist&l=Los+Angeles%2C+CA

data scientist jobs in Los Angeles, CA

Jobs 1-10 of 555: **All** - New - Be the first to see new jobs

Data Scientist
The Honest Company - Los Angeles, CA
Desired Experience: Hive, Machine Learning, R, C/C++, MATLAB, Data Mining, Scala, Weka, Java, Spark, Python
1 day ago

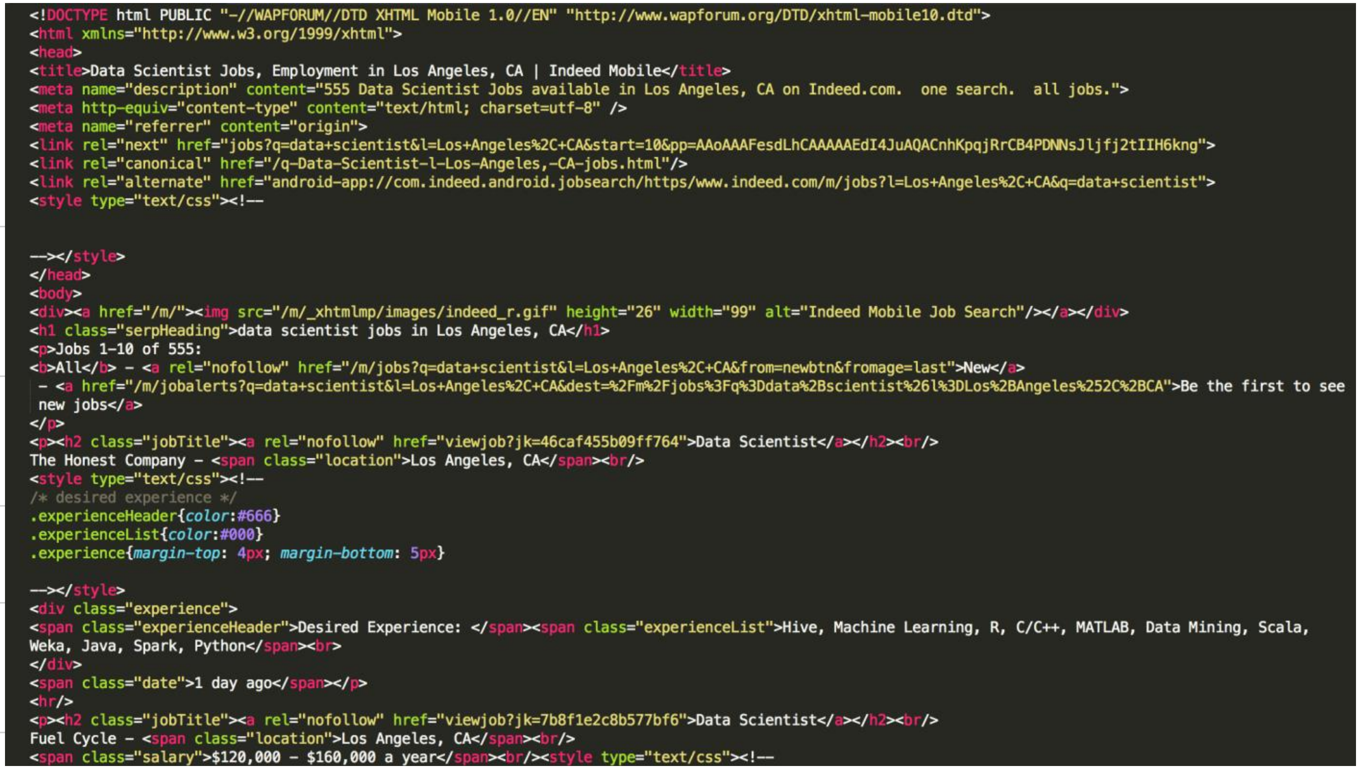
Data Scientist
Fuel Cycle - Los Angeles, CA
$120,000 - $160,000 a year
Desired Experience: Machine Learning, R, MySQL, AI, Data Mining, Sas, Java, Data Science, Python
30+ days ago

Data Scientist/Quantitative Analyst
Magid - Los Angeles, CA
Desired Experience: Machine Learning, R, Git
8 days ago

Data Scientist
Kaiser Permanente - Pasadena, CA
20 days ago

Data Scientist
L.A. Care Health Plan - Los Angeles, CA 90017
Desired Experience: Machine Learning, R, Sas, Tableau, Spark, Data Science, Python
11 days ago

Senior Data Scientist
Ticketmaster - Hollywood, CA 90028
Desired Experience: Machine Learning, C/C++, Hadoop, HBase, Java, Spark, Python
15 hours ago

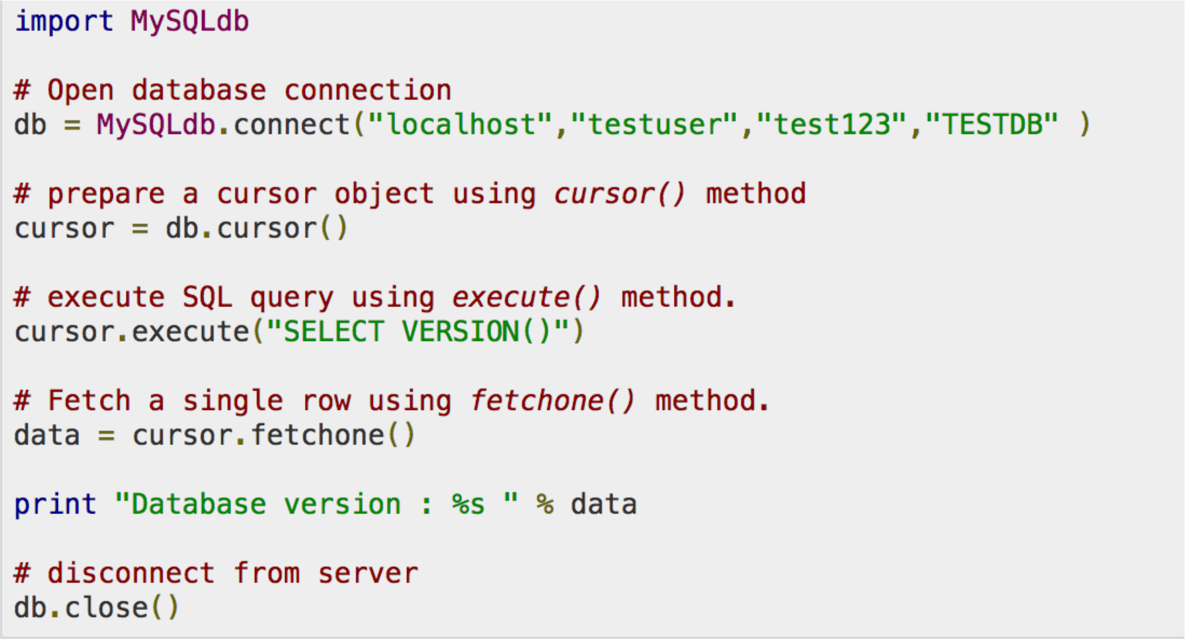


```html
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN" "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Data Scientist Jobs, Employment in Los Angeles, CA | Indeed Mobile</title>
<meta name="description" content="555 Data Scientist Jobs available in Los Angeles, CA on Indeed.com.  one search.  all jobs.">
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<meta name="referrer" content="origin">
<link rel="next" href="jobs?q=data+scientist&l=Los+Angeles%2C+CA&start=10&pp=AAoAAAFesdLhCAAAAAEdI4JuAQACnhKpqjRrCB4PDNNsJljfj2tIIH6kng">
<link rel="canonical" href="/q-Data-Scientist-l-Los-Angeles,-CA-jobs.html"/>
<link rel="alternate" href="android-app://com.indeed.android.jobsearch/https/www.indeed.com/m/jobs?l=Los+Angeles%2C+CA&q=data+scientist">
<style type="text/css"><!--


--></style>
</head>
<body>
<div><a href="/m/"><img src="/m/_xhtmlmp/images/indeed_r.gif" height="26" width="99" alt="Indeed Mobile Job Search"/></a></div>
<h1 class="serpHeading">data scientist jobs in Los Angeles, CA</h1>
<p>Jobs 1-10 of 555:
<b>All</b> - <a rel="nofollow" href="/m/jobs?q=data+scientist&l=Los+Angeles%2C+CA&from=newbtn&fromage=last">New</a>
 - <a href="/m/jobalerts?q=data+scientist&l=Los+Angeles%2C+CA&dest=%2Fm%2Fjobs%3Fq%3Ddata%2Bscientist%26l%3DLos%2BAngeles%252C%2BCA">Be the first to see
 new jobs</a>
</p>
<p><h2 class="jobTitle"><a rel="nofollow" href="viewjob?jk=46caf455b09ff764">Data Scientist</a></h2><br/>
The Honest Company - <span class="location">Los Angeles, CA</span><br/>
<style type="text/css"><!--
/* desired experience */
.experienceHeader{color:#666}
.experienceList{color:#000}
.experience{margin-top: 4px; margin-bottom: 5px}

--></style>
<div class="experience">
<span class="experienceHeader">Desired Experience: </span><span class="experienceList">Hive, Machine Learning, R, C/C++, MATLAB, Data Mining, Scala,
Weka, Java, Spark, Python</span><br>
</div>
<span class="date">1 day ago</span></p>
<hr/>
<p><h2 class="jobTitle"><a rel="nofollow" href="viewjob?jk=7b8f1e2c8b577bf6">Data Scientist</a></h2><br/>
Fuel Cycle - <span class="location">Los Angeles, CA</span><br/>
<span class="salary">$120,000 - $160,000 a year</span><br/><style type="text/css"><!--
```

Data Source: RDBMS



```
import MySQLdb

# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method
cursor = db.cursor()

# execute SQL query using execute() method.
cursor.execute("SELECT VERSION()")

# Fetch a single row using fetchone() method.
data = cursor.fetchone()

print "Database version : %s " % data

# disconnect from server
db.close()
```

Data Source: Big Data Platform

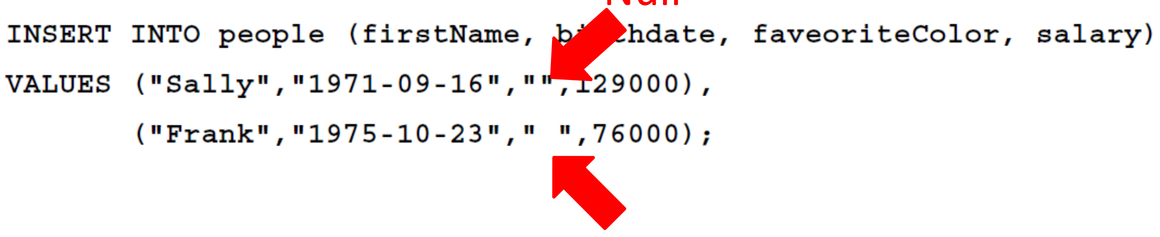
- HDFS – Hive
  - Text file and table
- Spark – RDD
  - Resilient Distributed Datasets
  - RDD is a read-only, partitioned collection of records
- Amazon -- S3
  - Cloud Data storage
  - File can be in any format

Data Types

- Numeric
  - Discrete: Count; Rating; Grade; Fibonacci Series
  - Continuous: Revenue; Distance; Home Value
  - Watch out: data range!
- Binary (Dummy)
  - Special case of numeric
  - E.g.: IsMale; HasCar; Pass
- Categorical
  - Usually contains characters: Gender, Product, Geo, etc.
  - Can be consist of pure numbers: SSN, Zipcode, Phone Number
  - Watch out: Valid Values
- Dates and Time
  - Date, Time, Datetime, Timestamp
  - Watch out: Time Zone! UTC=Coordinated Universal Time = GMT = Greenwich Mean Time
- Missing

Data Types: Missing

- Null
  - Absence of everything; missing; empty
- Blank
  - “ “ or “   “ or any invisible characters
  - can mean missing
  - can mean “N/A”
- N/A
  - Can mean “not available”: e.g. Age
  - Can mean “not applicable”: e.g. Middle Name
  - Can mean “no answer”: e.g. Customer Satisfaction Rating on a Questionnaire



Null

```
INSERT INTO people (firstName, birthdate, faveoriteColor, salary)
VALUES ("Sally","1971-09-16","",129000),
       ("Frank","1975-10-23"," ",76000);
```

Blank

Data Preparation Best Practice

Data Preparation Steps

- **Data Cleansing**
    - **Integrate (mapping)**: integrate various data sources into one dataset. E.g. sales units, sales revenue, price
    - **Conform**: Conform the inconsistent values. E.g. Na, n/a => missing; Los Angeles, L.A. => LA
    - **Filter**: Filter out the columns and rows not needed for modeling
    - **Extract**: Extract new column/feature from existing columns. E.g. month from date
    - **Group**: Group many categorical values into less buckets
    - **Aggregate**: Aggregate/Disaggregate date to the desired granularity
    - **Derived feature**: Calculate new metrics based on existing metrics. E.g. Price =Revenue/Units
- **Handle Missing Data**
- **Identity Outlier**
- **Transform Data**
    - **One hot encoding: categorical to numerical**
    - **Normalization/Standardization**
    - **Log transformation**

Data Cleansing: Regex 101

a single character of: a, b or c	**[abc]**
a character except: a, b or c	**[^abc]**
a character in the range: a-z	**[a-z]**
a character not in the range: a-z	**[^a-z]**
a character in the range: a-z or A-Z	**[a-zA-Z]**
any single character	**.**
any whitespace character	**\s**
any non-whitespace character	**\S**
any digit	**\d**
any non-digit	**\D**
any word character	**\w**
any non-word character	**\W**

capture everything enclosed	**(...)**
match either a or b	**(a|b)**
zero or one of a	**a?**
zero or more of a	**a***
one or more of a	**a+**
exactly 3 of a	**a{3}**
3 or more of a	**a{3,}**
between 3 and 6 of a	**a{3,6}**
start of string	**^**
end of string	**$**

Data Cleansing: Useful Regex

- Replace
  - Reverse last name and first name: San, Zhang => Zhang San
  - Regex=/([a-zA-Z]+),\s∗([a-zA-Z]+)/, Replace = $2 $1
- Extract
  - Extract url from html: <a href="http://www.amghezi.com">amgheziName</a>
  - Regex = /href=/"([^"]∗)/, Replace = $1
- Validation
  - Validate a valid email
  - Regex =/^([a-z0-9_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/i

Missing Data: Types

- **Missing completely at random: MCAR**
  - Roll a dice
  - Lottery number
- **Not missing at random: NMAR**
  - missing values are systematic
  - Income: higher income is less likely to respond
  - Weight: higher weight is less likely to respond
  - Smoking
- **Missing at random: MAR**
  - Most Common
  - Missing values can somewhat be predicted by known info
  - Know height, missing weight
  - Know # of rooms, missing sqrt

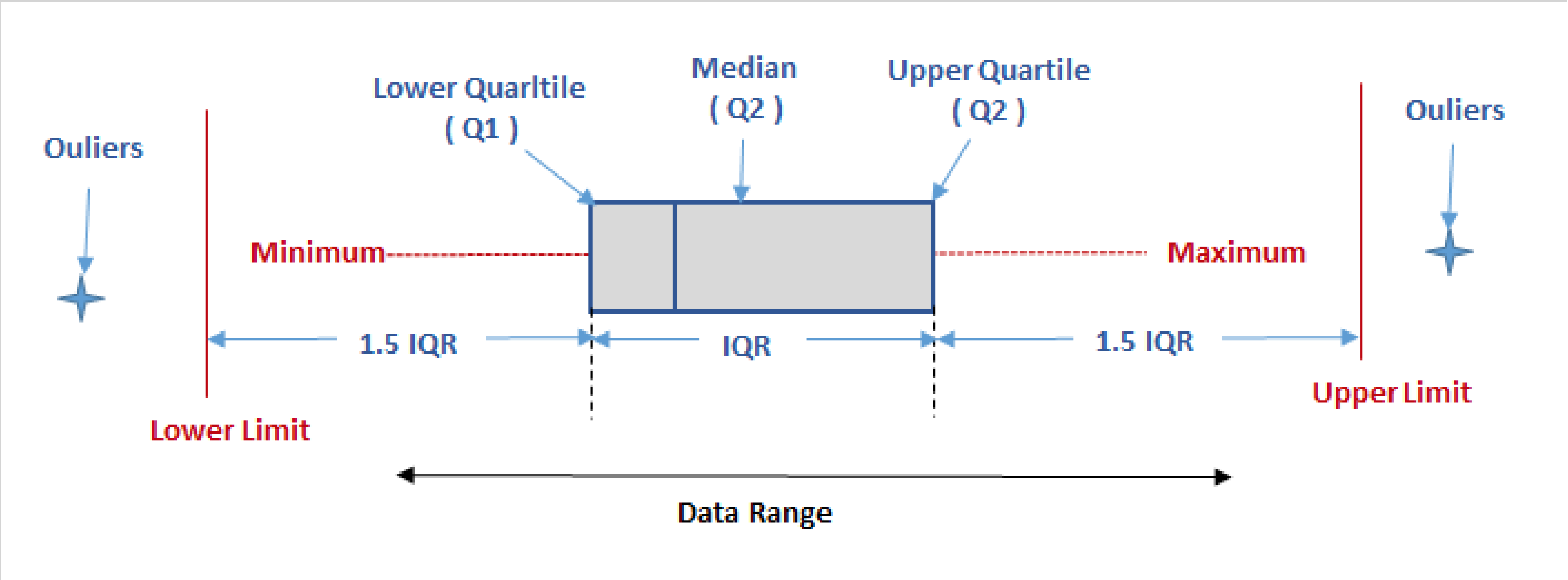
Missing Data: Handling

- Impute from other attributes
- Impute from other observations
  - Majority vote (categorical)
  - Mean of same/similar group (numerical)
  - Carry last value (time series)
  - Linear fill (time series)
  - Carry same trend (time series)
- “Missing” Category (not missing at random)
- Extra indicator
- Logical estimation
- Remove row or column

Outliers: 1.5 IQR

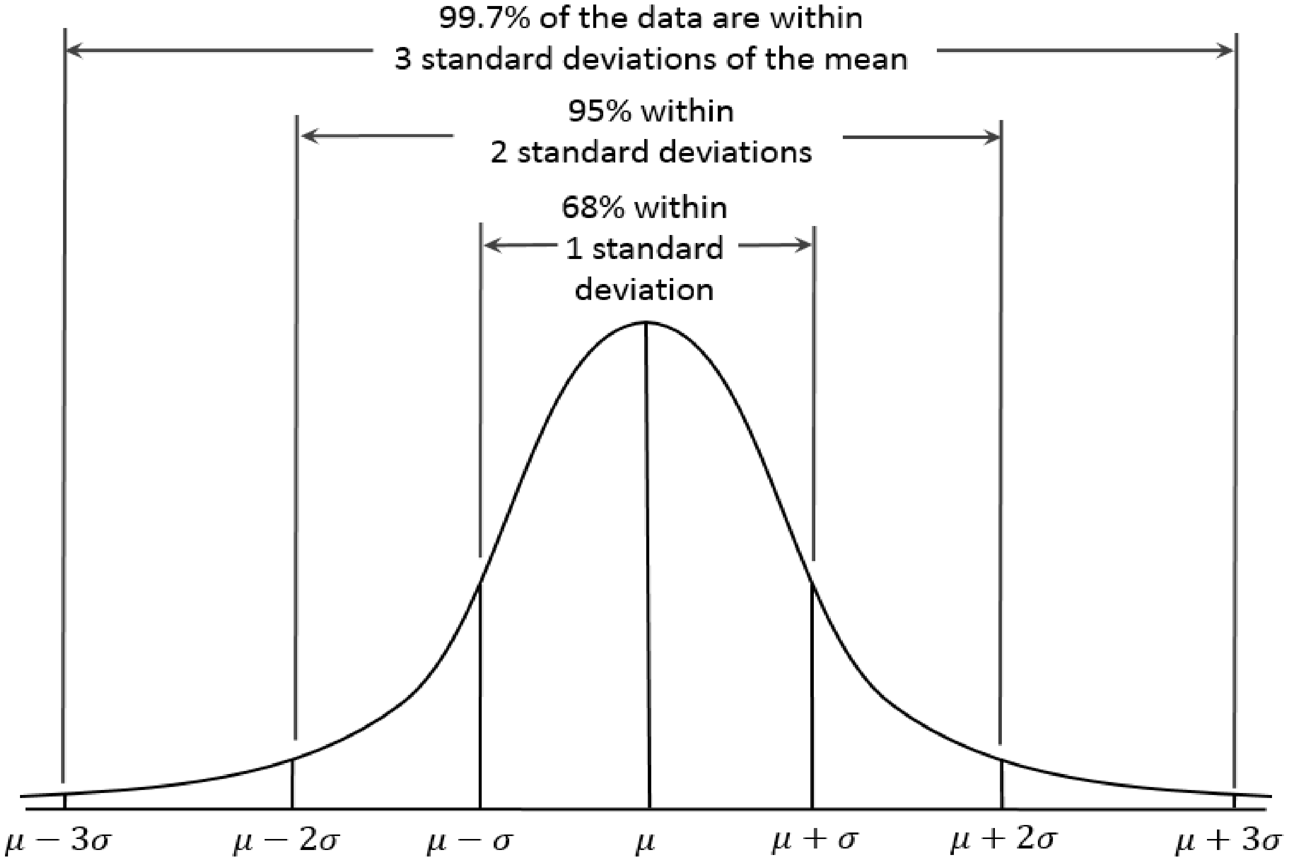
Check the frequency distribution of the data

Box-plot: An outlier is a point of data that lies over 1.5 IQRs below the first quartile (Q1) or above third quartile (Q3) in a given data set.



Outlier: Normal Distribution

Outlier: 2 or 3 STD from mean



Outlier: Other Technics

- Univariable Outlier:
  - Median Absolute Deviation
- Multivariate Outlier
  - Mahalanobis Distance

Data Transformation: Normalization vs Standardization

	Normalization	Standardization
Formula	$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$	$x_{new} = \frac{x - \mu}{\sigma}$
Pro	• Bounded (-1,1) • Apply to all distribution	• Works well for normal distribution
Con	• Make outliers “normal”	• Unbounded • Only works well for normal distribution

Transformation: When to Normalize

- Linear Model
  - Recommended
  - Doesn’t change model accuracy
  - Easier to compare coefficient: larger coefficient, larger impact
  - Intercept well interpreted: the expected value of Yi when the predictors are set to their means
  - Avoid coefficient like 10^-9 when one variable has a very large scale
  - More difficult to interpret the model in terms of on unit change in Xi
- Tree Model
  - Not necessary as the scale is irrelevant
- Logistic Regression
  - Typically not needed
- SVM
  - Recommended
  - Help with faster converge

Transformation: Log

Linear Model; Skewed Data

- Log Predictor

$$y = e^{ax} + \mathrm{b} \xrightarrow{\log \mathrm{x}} y = ax' + \mathrm{b}$$

- Log Outcome

$$y = ln\,(ax + \mathrm{b}) \xrightarrow{\log \mathrm{y}} y' = ax + \mathrm{b}$$

- Log both

$$y = e^{c} * {x_1}^{a} * {x_2}^{b} \xrightarrow{\text{yields}} \ln y = c + ax_1 + \mathrm{b}\mathrm{x}_2$$

Demo

Use Python to clean Airbnb listings data (from file)

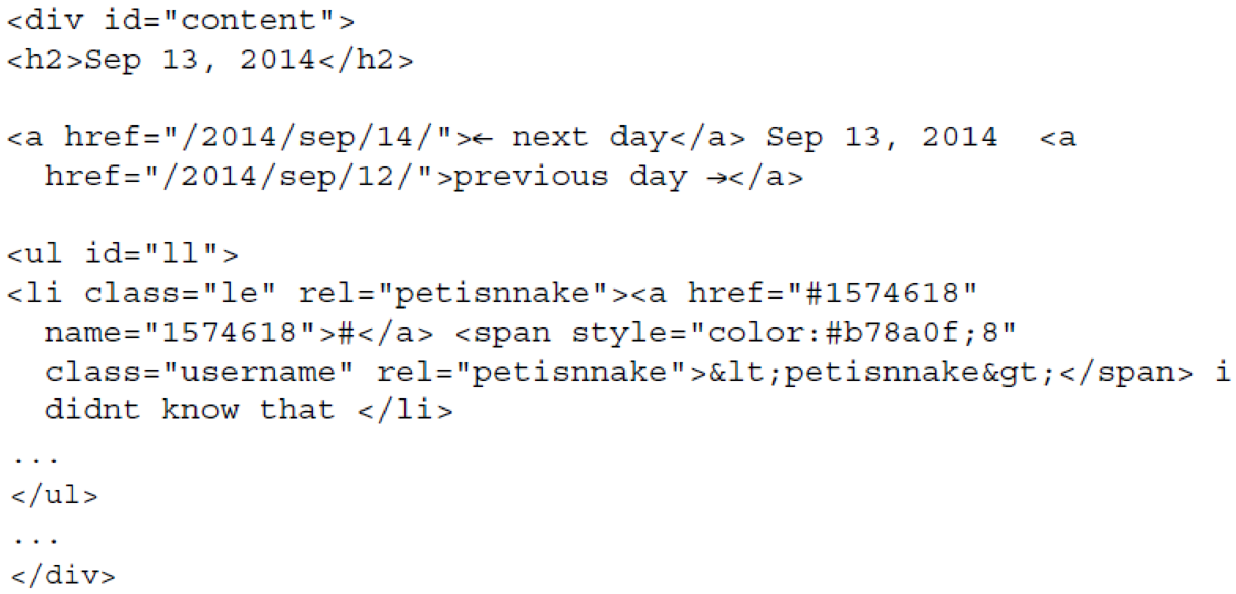
Web Data Preparation

WEB data raw format: HTML

Understanding the HTML Page Structure

HTML can be parsed in two ways:

- The line-by-line delimiter model
- The tree structure model



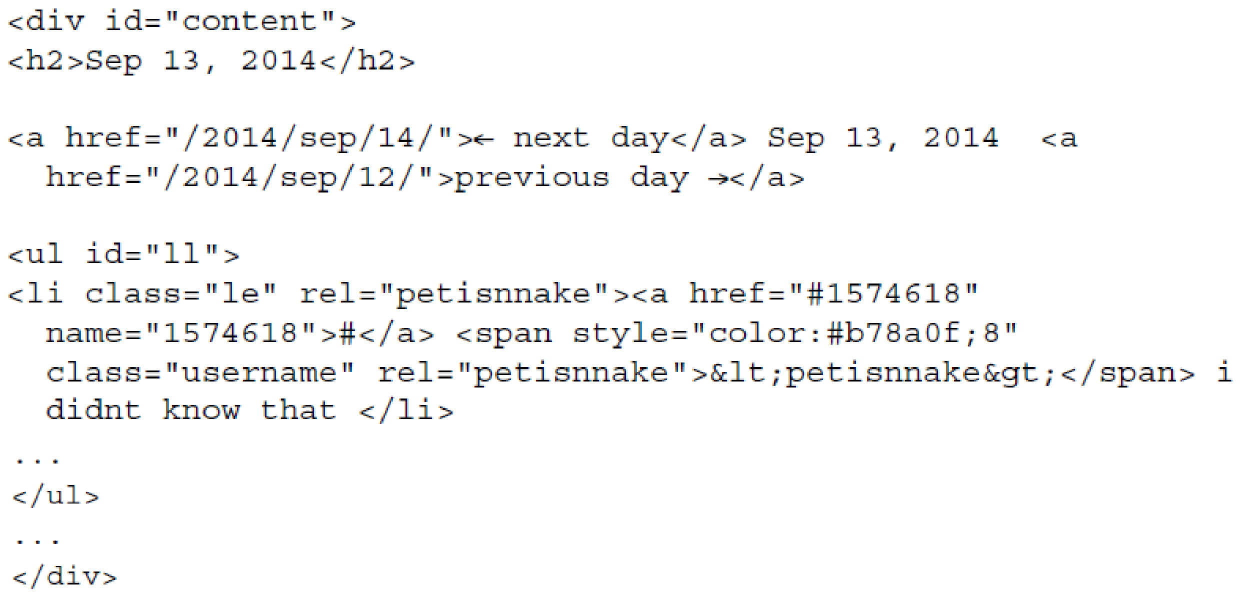
```
<div id="content">
<h2>Sep 13, 2014</h2>

<a href="/2014/sep/14/">← next day</a> Sep 13, 2014  <a
  href="/2014/sep/12/">previous day →</a>

<ul id="ll">
<li class="le" rel="petisnnake"><a href="#1574618"
  name="1574618">#</a> <span style="color:#b78a0f;8"
  class="username" rel="petisnnake">&lt;petisnnake&gt;</span> i
  didnt know that </li>
...
</ul>
...
</div>
```

Web Scraping: Line by Line

The line-by-line delimiter model



```
<div id="content">
<h2>Sep 13, 2014</h2>

<a href="/2014/sep/14/">← next day</a> Sep 13, 2014  <a
  href="/2014/sep/12/">previous day →</a>

<ul id="ll">
<li class="le" rel="petisnnake"><a href="#1574618"
  name="1574618">#</a> <span style="color:#b78a0f;8"
  class="username" rel="petisnnake">&lt;petisnnake&gt;</span> i
  didnt know that </li>
...
</ul>
...
</div>
```

- <h2></h2> tags as delimiters to extract the date
- <li></li> tags as delimiters to extract text
- Rel=“” as delimiters to extract user name
- From the end of </span> to the beginning of </li> is the actual line message

Extract date by Regex: <h2>(.+)<\/h2>

Extract message by Regex : <\/span>(.+)<\/li>

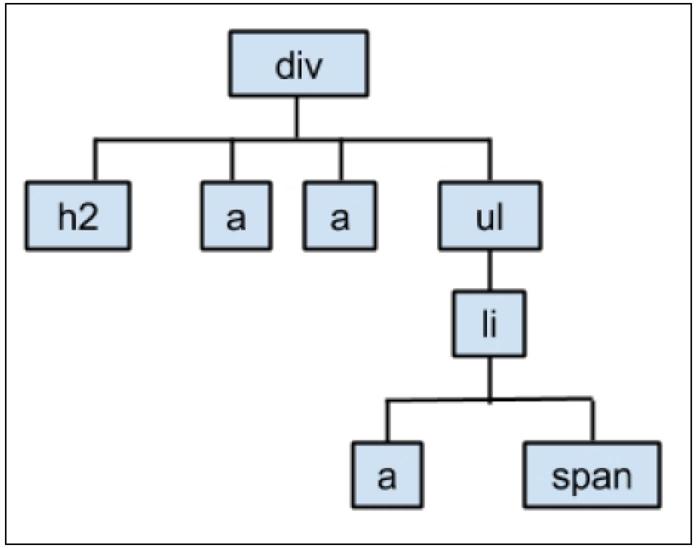
Web Scraping: Tree Model

The tree structure model: we can consider the structure of HTML as a tree structure

```
<div id="content">
<h2>Sep 13, 2014</h2>

<a href="/2014/sep/14/">← next day</a> Sep 13, 2014  <a
  href="/2014/sep/12/">previous day →</a>

<ul id="ll">
<li class="le" rel="petisnnake"><a href="#1574618"
  name="1574618">#</a> <span style="color:#b78a0f;8"
  class="username" rel="petisnnake">&lt;petisnnake&gt;</span> i
  didnt know that </li>
...
</ul>
...
</div>
```



Extract date by beautifulSoup: div.h2.text

Extract message by beautifulSoup: div.ul.li.text

Demo

Use Python Beautifulsoup to collect and clean job listing data from indeed.com