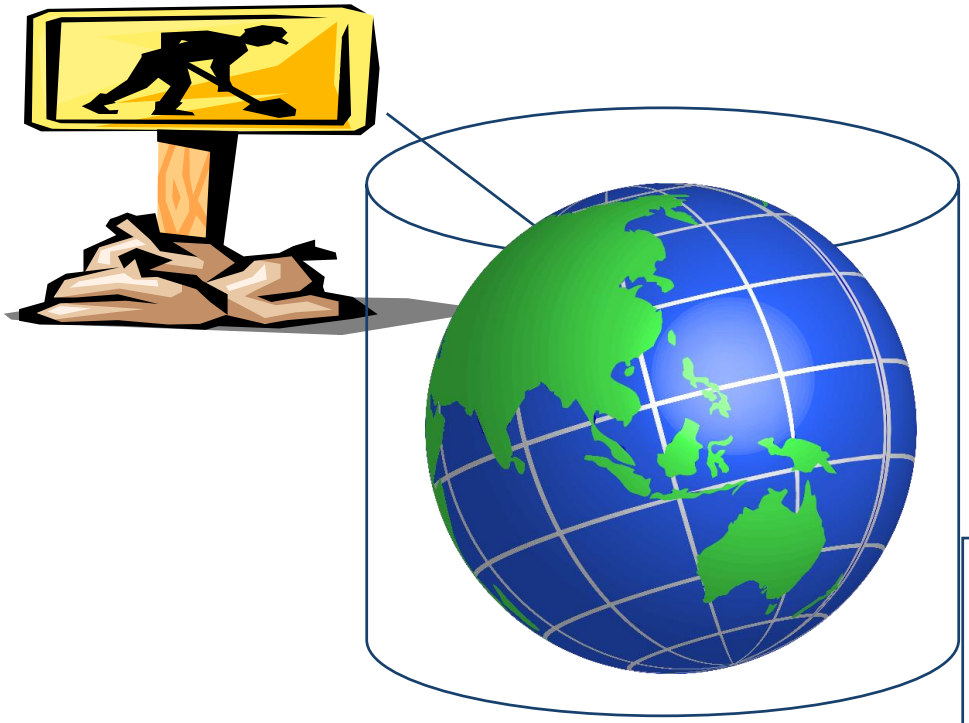
Comp 8920: Selected Topics on Web Data Extraction and Recommendation Systems Techniques



Dr. C.I. Ezeife
School of Computer Science,
University of Windsor, Canada.
Email: cezeife@uwindsor.ca

Course Content and Parts

- 1. Overview of Data Mining Techniques for Analazing Extracted Web Data (Part I)
- 2. Web Data Extraction with Python (Part II)
- 3. Web Recommendation Systems and Algorithms (Part III)
- 4. Student assigned seminar and project topics on Parts I to III and on existing data extraction and web recommendation systems

PART 1 - DATA MININING

- 1. Data Mining Techniques Used for Data Analysis
  - 1. 1 Association Rule Mining
  - 1.2 Classification and Regression Rules
    - Decision Trees
  - 1.3 Clustering
  - 1.4 Genetic Algorithms
- 2. Data Visualization

1. Data Mining

- **Data mining helps end users extract useful business information from large databases.**
- **Data mining is used for finding interesting trends or patterns in large datasets to guide decisions about future activities.**
- **Data mining helps answer questions like**
  - **1. What has been going on in my data?**
  - **2. What is going to happen next?**
  - **3. How can I profit?**
- **Data mining tools provide solutions which are built on such computer algorithms as association rule mining, decision tree classifications, neural networks, and clustering techniques like nearest neighbor.**

1. Data Mining

- **Data mining is closely related to the area of statistics called exploratory data analysis and also related to the subareas of artificial intelligence called knowledge discovery, but handles much larger data in an automated fashion.**
- **Data mining tools are built to be embedded into the business data warehouse and to be understandable and usable by marketing professionals, while classic statistical tools can not fulfil these objectives.**
- **Data mining is used for information discovery (looking for gold in mountain of data) and prediction (using the found information gold to predict what will happen next)**
- **The knowledge discovery and data mining (KDD) process consists of data selection, data cleaning, data mining and result evaluation.**

1. Data Mining

- **The importance of the discovered information can be measured by (1) how strong the association is and (2) how unexpected**
- **Two ways for building predictive models are called supervised learning and unsupervised learning.**
- **With supervised learning, the aim is to predict some target field in the historical database, unsupervised learning has no well-defined goal or target to predict.**
- **Techniques such as clustering and detection of association rules fall into the category of unsupervised learning. Mining techniques are discussed next.**

1.1 Association Rule Mining

- **Association Rule mining is the most common form of knowledge discovery in unsupervised learning systems.**
- **Mining for gold in this method is equivalent to looking for an interesting rule, formed from database attributes occurring strongly together in the same transactions.**
- **Association Rule mining in a database aims at pulling out all possible patterns (combinations of attributes) from the database, calculating their support (or coverage) and confidence (accuracy).**
- **Rules are generally simple, e.g.,**
- **"If bagels are purchased, then cream cheese is purchased 90% of the time and this pattern occurs in 3% of all shopping baskets."**

1.1 Association Rule Mining

- **Where is Association Rule Mining Used?**
- **Examples of rules are**
  - **if paper plates, then plastic forks (paper plates => plastic forks)**
  - **if dip, then potato chips(dips => potato chips)**
  - **if salsa, then tortilla chips (salsa => tortilla chips)**
- **For the computed rules to be useful, two pieces of information must be supplied by the mining result as well:**
  - **1. Confidence (accuracy): how often is the rule correct?**
  - **2. Support (or Coverage) : how often does the rule apply?**
- **The rules themselves consist of two halves, the left side called *antecedent* and right side called *consequent*.**

1.1 Association Rule Mining

- **Example of Rule confidence and support**

Rule	Confidence	Support
if breakfast cereal, then milk is purchased	85	20
if bread purchased, then Swiss cheese purchased	15	6
if 42 years old and purchased pretzels and dry roasted peanuts, then bear is purchased	95	0.01

- **The first rule is a pattern that occurs quite often and is right more often than not, while the last rule is almost never wrong but also is almost never applicable.**
- **The antecedent can consist of one or multiple conditions which must all be true in order for the consequent to be true at the given confidence (accuracy).**
- **Generally, the consequent is a single condition.**

1.1 Association Rule Mining

- **What can business do with a rule?**
- **1. Target the antecedent: E.g., collect all rules that show a certain value (e.g., nail) on the antecedent to determine its relationship with other items.**
- **2. Target the consequent: E.g., place rules showing items in consequent close to items as sales promo.**
- **3. Target based on confidence: target highly accurate rules of 80 to 90% for higher benefit.**
- **4. Target based on support: increased support means most readily applicable.**
- **5. Target based on interestingness: Interesting rules have high confidence and high support and deviate from the norm.**

1.1 Association Rule Mining

- **General idea**
- **Rules are created to show events in the database and are used to find interesting patterns in the database. The two important features of a rule are:**
  - **1. Confidence: the probability that if the antecedent is true, then the consequent will be true. High confidence means that this is a rule that is highly dependable. That is,**
    confidence = |Rule|/|antecedent|
  - **2. Support: the number of records in the database that the rule applies to. High support means that the rule can be used very often. That is,**
    support = |Rule|/|total|
  - **Coverage: |antecedent|/|total| is also used in place of support sometimes.**

1.1 Association Rule Mining

- **In statistics, support is apriori probability of both the antecedent and consequent occurring and the confidence is the probability of the consequent conditional on the antecedent.**
- **Thus, to calculate for a simple rule like " if milk, then egg"**
  milk ➔ egg
  Total = 100 = total number of shopping baskets in the database.
  Egg = 30 = number of baskets with eggs in them
  Milk = 40 = number of baskets with milk in them
  Both = 20 = number of baskets with both eggs and milk in them
  Confidence = $\frac{|\text{Both}|}{|\text{Milk}|} = \frac{20}{40} = 50\%$
- **Coverage = $\frac{|\text{Milk}|}{|\text{Total}|} = \frac{40}{100} = 40\%$**
- **Support = $\frac{|\text{Both}|}{|\text{Total}|} = \frac{20}{100} = 20\%$**

1.1 Association Rule Mining

- **Other measures of usefulness include:**
- **1. Support of a Rule as either percentage or number of records: percentage of times the entire rule occurs**

$$\text{Support} = \frac{|\text{Both}|}{|\text{Total}|} = \text{confidence} * \text{coverage}$$

$$= \frac{|\text{Both}|}{|\text{antecedent}|} * \frac{|\text{antecedent}|}{|\text{Total}|}$$

or Support= |Both|

- **2. Support as number: used to capture support not as a probability or percentage but as the total number of records that match the given antecedent. Support number = |antecedent U consequent|.**
- **3. Significance: compares given pattern to random chance. To compute significance, we find the probability that all events are independent and compare it with the support for the rule.**

1.1 Association Rule Mining

- **Association Rule Mining Algorithms include the Apriori Algorithm and the Frequent Pattern Tree Algorithm.**
- **Given a market basket database with a list of 5 items and 4 transactions, with minimum support of 2 trans:**
- **C1= {milk, sugar, bread, salt, tea}**

TID	items Purchased
1	milk, sugar
2	milk, sugar, tea
3	milk, sugar, bread, tea
4	sugar, bread, salt, tea

1.1 Association Rule Mining

- **The Apriori algorithm finds the set of frequent patterns (large itemsets, L_i) iteratively by computing the support of each itemset in the candidate set C_i . During each ith iteration, it finds the ith large itemsets (L_i) from C_i, before computing the next (i+1) candidate set C_{i+1} as L_i apriori-gen join L_i. It then, prunes itemsets from C_{i+1} which have any subset that are not already large. The process terminates when either a C_i or L_i is an empty set.**
- **The apriori-gen join of L_i with L_i joins every itemset k of first L_i with every itemset n of second L_i where n > k and first (i-1) members of itemsets k and n are the same.**
- **Example mining of the above table with the Apriori algorithm by executing the steps of the algorithm on a sample database proceeds as:**

1.1 Association Rule Mining

- **L1(large itemsets 1) = {milk, sugar, bread, tea}**
- **C2 = L1 apriori gen L1 = {milk:sugar, milk:bread, milk:tea, sugar:bread, sugar:tea, bread:tea} after join and prune stages.**
- **L2= {milk:sugar, milk:tea, sugar:bread, sugar:tea, bread:tea}**
- **C3={milk:sugar:tea, sugar:bread:tea}**
- **L3 ={milk:sugar:tea, sugar:bread:tea}**
- **C4={empty set} and Apriori ends.**
- **The discovered large itemsets L = L_1 U L_2 U L_3.**
- **Rules are formed from these large itemsets and only strong rules with confidence greater than or equal to minimum confidence are kept.**
- **To form rules, from each each frequent pattern L_i, obtain subsets and from each subset s, form a rule as s → L_i – s, where s is the antecedent and L_i – s is the consequent.**

1.1 Association Rule Mining

- **Versions of the Apriori algorithm have been developed for other kinds of mining including sequential and web mining (check article references for details).**
- **The Frequent Pattern (FP-tree) algorithm stores the database information in an FP-tree which is used for mining.**

1.1 Association Rule Mining

- **Details of the FP-tree and FP-growth algorithms can be found in references. Summary of the algorithm is:**
- **1. Get the frequent 1-items F_1 in descending order of support.**
- **2. For each transaction t in DB, get frequent transaction ft as only items in F_1 order.**
- **3. Using the ft set, construct the FP tree with F_1 header linkages.**
- **4. Recursively mine (in suffix order) the FP tree by starting from the lowest item down the leaf on the tree and obtaining the conditional pattern bases of the suffix frequent pattern first. And the process continues by reconstructing an FP tree with conditional pattern bases that are frequent with count greater than or equal to minimum support as long as there is more than one node on the tree to continuously find prefix frequent patterns of the already found suffix patterns.**

1.2 Classification and Regression Rules

- **Classification algorithms have two phases of (1) training/learning phase used to define class labels from historical data; and (2) testing phase to predict the classes of new incoming datasets. Decision tree is an example classification algorithm.**
- **E.g. Info from a mailing campaign by an insurance company is:**
- **InsuranceInfo(age:integer, cartype:string, highrisk:boolean)**
- **We want to use this table information to identify rules that predict the insurance risk of new insurance applicants whose ages and cartypes are known.**
- **E.g., if age is between 16 and 25, and cartype is either Sports or Truck, then, the risk is high.**
- **The attribute we want to predict (highrisk) is called the dependent attribute. The other attrs are called predictor attributes (age and cartype).**
- **The general form of the types of rules we want to discover is:**
  $P_1(X_1) \wedge P_2(X_2) \wedge \ldots \wedge P_k(X_k) \rightarrow Y = C$

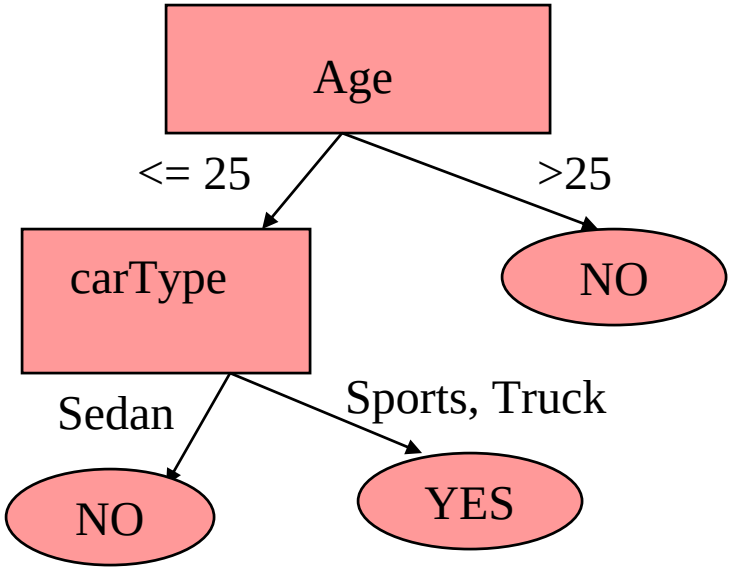
1.2 Classification and Regression Rules

- **The $P_i(X_i)$ are predicates that involve predictor attrs X_i.**
- **Attributes can be either numerical (e.g., age) or categorical (e.g., cartype and highrisk).**
- **If the dependent attr is categorical, the rule is called classification rule, but it is regression rule if the dependent attr is numerical.**
- **Thus, since highrisk is categorical, we have classification rule in our example as:**
  (age <= 25) ^ (cartype ε{Sports, Truck}) ➔ highrisk = true
- **Support and confidence for classification and regression rules can be defined as:**
- **Support: for a rule C1 ➔ C2 is the %age of tuples that satisfy C1 ^ C2**
- **Confidence: for a rule C1➔C2 is the %age of C1 tuples satisfying C2 too.**

1.2 Classification and Regression Rules

- **A decision tree is a graphical or tree representation of a collection of classification rules. Each internal node denotes a test on an attribute, each branch represents an outcome of the test and leaf nodes represent classes or class distributions. Decision trees are built or learnt from training data. A basic decision tree induction algm is ID3 (Fig. 7.3 of Han book) and another is C4.5.**
- **In order to classify an unknown sample, the attribute values of the sample are tested against the decision tree. A path from root to a leaf node holding the class prediction for that sample is traced. Given a data record, the tree directs the record from the root to a leaf. Each internal node of the tree is labeled with a predictor attribute called splitting attribute.**
- **The outgoing edges of an internal node are labeled with predicate that involve splitting attributes of the node.**
- **The combined information about the splitting attr and the predicates on the outgoing edges is called the splitting criterion of the node.**
- **Each leaf node of the tree is labeled with a value of the dependent attribute. E.g.,**

1.2 Classification and Regression Rules



A DECISION TREE EXAMPLE

Insurance Risk Decision Tree

- A decision tree represents a collection of classification rules, one for each leaf node. The tree is usually constructed in 2 phases – the growth and the prune phases. Several algorithms exist for making both stages efficient (ID3, C4.5). Other classification techniques are Naïve Bayesian Classification, which is based on Bayes theorem "The effect of an attr value on a given class is independent of the values of the other attrs" (see section 7.4 for more).

1.2 Classification and Regression Rules

- **Where are decision trees used?**
- **Decision trees automate the process of hypothesis generation and testing and handle data with little or no preprocessing. They have been used in application areas ranging from medicine to game theory and business.**
- **It can be used for exploration of data set by looking at predictors and values chosen for each split of the tree.**
- **It can be used to preprocess data for other prediction algorithms by creating a subset of possibly useful predictors in the first run of data.**
- **It is a supervised learning that can be built from historical data.**

1.2 Classification and Regression Rules

- **Growing the Tree**
- **The first step in the process is to grow the tree.**
- **In growing the tree, the objective is to find the best possible question to ask at each branch point of the tree to come up with homogeneous leaves**
- **In building the tree, decision tree algorithms look at all possible distinct questions that could break up the data set into segments that are homogeneous with respect to the prediction values. They pick the questions through heuristics or at random. An entropy-based measure may be used for selecting the test or decision attribute.**

1.2 Classification and Regression Rules

- **Classification and Regression trees (CART) pick the questions by trying them all before picking the best one.**
- **The decision tree stops growing when either of the following 3 conditions occurs:**
  - **1. Segment contains only one record or algorithmically defined minimum number of records.**
  - **2. Segment is completely organized into one prediction value.**
  - **3. The improvement in organization is not sufficient to warrant making the split (e.g. splitting 90% of churners to get 89.99% does not represent any progress).**

1.3. Nearest Neighbor and Clustering

- **Other less developed classification methods include K-nearaest neighbor classification, genetic algorithms, rough set and fuzzy set approaches. K-Nearest neighbor is a prediction technique that is based on learning by analogy and is quite similar to clustering**
- **The idea here is that in order to determine what a prediction value is in a new record, the user should look for records with similar predictor values in the historical or training database and use the prediction value from the record that is "nearest" to the unknown record. "Nearness" is defined in terms of Euclidean distance $d(X,Y) = sqrt(\sum^{n}_{i=1}(x_i - y_i)^2)$ between two points $X=(x_1, x_2, .., x_n)$ and $Y=(y_1,y_2, ..y_n)$**
- **For example, in order to predict a person's income, use the income of his nearest neighbor.**
- **Nearness in a database may consist of other factors like school attended by person, degrees obtained etc.**

1.3. Nearest Neighbor and Clustering

- **Clustering is used to partition a set of records into groups such that records within a group are similar to each other. However, unlike classification, the class label of each objet is unknown. Clustering groups the data into classes or clusters where objects within a cluster have high similarity in comparison to one another, but are very dissimilar to objects in other clusters.**
- **Each such group is called a cluster and similarity or dissimilarity between records may be measured by a distance function.**
- **Clustering techniques include partition-based (e.g. K-Means), hierarchical (e.g. BIRCH), desnsity-based (e.g DBSCAN, LSC), grid-based methods (e.g. STING).**
- **E.g., Customerinfo(caddress: string, salary: real)**

1.3. Nearest Neighbor and Clustering

- **Clusters are used more for exploration and summarization than for prediction and have been used in pattern recognition, image processing and market research.**
- **Clustering can be used to identify outliers or records that stick out**
- **The K-means algm is on Fig. 10.2 and is summarized as:**
  Input: number of clusters k and a database of n objects
  Output: a set of k clusters that minimizes the squared erorr criterion.
  Method:
  1. arbirtarily choose k objects as the initial cluster centers
  2. repeat
  3. (re)assign each object to the cluster to which the object is most similar, based on the mean value of the objects in the cluster;
  4. update the cluster means by calculating the mean of objects in each cluster.
  5. until no change;

1.4. Genetic Algorithms

- **Genetic algorithms are algorithms that dictate how population of organisms should be formed, evaluated and modified.**
- **E.g., there is a genetic algorithm that determines how to select organisms for sexual reproduction and another that will determine which organisms to be deleted from the population.**
- **Genetic algorithm can be used to optimize a variety of data mining techniques like neural networks and nearest neighbor.**

OTHER Mining Approaches to Possibly Explore

- **Sequential Mining and Web Mining algorithms (e.g., PLWAP) using Web access pattern tree**
- **AI approaches to Mining using Neural Networks.**
- **Statistical approaches to Mining using point estimation, Bayes theorem, hypothesis testing, regression and correlation, similarity measures etc.**
- **Data Stream Mining**
- **Incremental Mining and incremental sequential mining**
- **Presentation of Mining Results through data visualization and other means.**

2. Data Visualization

- **Data visualization displays patterns in a way that can easily be understood by humans.**
- **Data visualization has to do with finding a mapping between very high-dimensional spaces that exist in the database or the predictive model and the two-dimensional space that exists on the computer screen.**
- **Three dimensional and higher dimensional spaces are simulated through the 2-dimensional space making the task of data visualization even harder.**
- **Good data visualization allows the user the following key abilities:**

2. Data Visualization

- **1. Ability to compare data**
- **2. Ability to control scale (look from a high level or drill down to detail).**
- **3. Ability to map the visualization back to the detailed data that created it.**
- **4. Ability to filter the data to look only at subsets or subregions of it at a given time.**

2. Data Visualization

- **Visualization is used in a number of places within data mining as follows:**
- **As a first-pass at the data mountain which is used to provide the user some idea of where to begin mining.**
- **As a way to display the data mining results and predictive model in a way that is understandable to the end user.**
- **As a way of providing confirmation that the data mining was performed the correct way.**
- **As a way to perform data mining directly through exploration analysis, allowing the end user to look for and find patterns so efficiently that it can be done in real time by the end users without using automated data mining techniques.**

Part II: WEB DATA EXTRACTION

- 1. Web Scraping, Web Scraping Tasks
- 2. Web Data
- 3. Web Scraper Project
- 4. BeautifulSoup Library
- 5. Advanced HTML Parsing
- 6. Starting to Crawl, Using APIs, Storing Data, Reading Documents
- 7. Etc.

- Reference: Web Scraping with Python: Collecting Data from Modern Web by Ryan Mitchell, Oreilly books.

Web Scraping, Web Scraping Tasks

- Web scraping is the ability to write a simple program that collects data (not through an application programming interface API) from the world wide web page and streams it down a terminal or stores it in a database.
- It is also called web harvesting. Web scraping programs are called bots.
- Modern web has html codes, Java Script, multimedia cookies and other heterogeneous data.

Web Scraping, Web Scraping Tasks

- Web scraping is most commonly accomplished by writing a program that queries a web server, requests data (usually in the form of the HTML and other files that comprise web pages), and then parses that data to extract needed information.
- Web scraping encompasses a wide variety of programming techniques and technologies, such as data analysis/mining and information security
- Supplemental material (code examples, exercises, etc.) is available for download at *https://github.com/REMitchell/python-scraping*.

Web Scraping, Web Scraping Tasks

Why Web Scraping?

1. Web Scraper Versus Search Engine: Web content data can be found by search engines like Google. However, doing specific comparative querying on things like product pricing on such content may require sifting through several results.

   - For example, “Get the cheapest flights from Toronto to New York” can not easily be answered with search engine. However, web scrapers can chart the cost of a flight from Toronto to New York across a number of web sites and tell you the best time to buy tickets.

Web Scraping, Web Scraping Tasks

- 2. Web Scraper versus Web browsers: While web browsers are good for presenting images and multimedia data of only a web page at a time (e.g., an E-Commerce store product page), web scrapers can present a database of thousands or even millions of product pages at once so they be queried.
- 3. Web Scraper versus API: API's are used for data gathering as well. Example API's are Twitter posts API, APIs for Wikipedia pages. However, APIs may not exist for the type of data you are gathering. Also, the format of data provided by the API may not be adequate for your purpose. With Web scrapers, Python scripts can be used to extract data to be stored in a database for analysis.

Example Web Scraper Application

- The 2006 project "We Feel Fine" by Jonathan Harris and Sep Kamvar, scraped a variety of English-language blog sites for phrases starting with "I feel" or "I am feeling."
- The data collected for the 2006 Art project was visualized and used to describe how the world was feeling day by day and minute by minute.
- Web scraping can guide business practices more effectively and productively.

Basic Web Scraping Tasks

- 1. Retrieving HTML data from a domain name
- 2. Parsing that data for target information
- 3. Storing the target information
- 4.Optionally, moving to another page to repeat the process

Web Data

- The web data rendered to us through web browsers like Firefox, Chrome include
  - 1. HTML formatted text content
  - 2. CSS (cascading style sheet) styling
  - 3. Javascript executions
  - 4. image rendering
  - 5. and so on.

Web Scraper Project

- Task 1: Send a GET request to a web server for a specific web page (e.g., http://pythonscraping.com/pages/page1.html), read the HTML output from that page, and do some simple data extraction to isolate the content that we are looking for.
- Solution 1: Using Python with its library functions, such as urllib and urlopen, task 1 can be done with the following code:

```
from urllib.request import urlopen
html =  urlopen("http://pythonscraping.com/pages/page1.html")
print(html.read())
```

Web Scraper Project

- Save this code as scrapetest.py and run it in your terminal using the command:
- $python scrapetest.py
- Note that if you also have Python 2.x installed on your machine, you may need to explicitly call Python 3.x by running the command this way:
- $python3 scrapetest.py
- This will output the complete HTML code (file) for the page at http://pythonscraping.com/pages/page1.html.

Web Scraper Project

- The Python code scrapetest.py first calls the Python module called request located in the urllib library and imports only the function urlopen.
- Urllib is a standard Python library that contains functions for requesting data across the web, handling cookies, changing metadata such as headers and your user agent.
- Read the Python documentation for the library (*https://docs.python.org/3/library/urllib.html*).

Web Scraper Project

- The urlopen is used to open a remote object across a network and read it. It can read HTML files, image files, or any other file stream with ease).
- BeautifulSoup library helps format and organize the messy web by fixing bad HTML and presenting us with easily-traversible Python objects representing XML structures.
- BeautifulSoup 4 (BS4) library is not a default Python library, it must be installed and instructions on its installation can be found at Crummy.com or *http://www.crummy.com/software/BeautifulSoup/#Download*.

Web Scraper Project

- Once BeautifulSoup 4 is installed and recognized as a Python library on your machine, you can import it in a Python terminal using the following commands:

```
$python
> from bs4 import BeautifulSoup
```

- Running BeautifulSoup
- The most commonly used object in the BeautifulSoup library is, the BeautifulSoup object.
- We can update the code1 in the scrapetest.py to the following code2 by calling the BS object to format the HTML of the page obtained through call to html.read as:

Web Scraper Project

- ```
  from urllib.request import urlopen
  from bs4 import BeautifulSoup
  html =
  urlopen("http://www.pythonscraping.com/pages/page1.
  html")
  bsObj = BeautifulSoup(html.read())
  print(bsObj.h1)
  ```

- The output is:
- <h1>An Interesting Title</h1>
  ```

Web Scraper Project

- Code2 is importing the urlopen library and calling html.read() in order to get the HTML content of the page.
- This HTML content is then transformed into a BeautifulSoup object, with the following structure:
- html → <html><head>...</head><body>...</body></html>
- head → <head><title>A Useful Page</title></head>
- title → <title>A Useful Page</title>
- body → <body><h1>An Int...</h1> Lorem ip... </body>
- h1 → <h1>An Interesting Title</h1>
- div → Lorem Ipsum dolor...

Web Scraper Project

- The <h1> tag that we extracted from the page was nested two layers deep into our BeautifulSoup object structure (html → body → h1).
- However, when we actually fetched it from the object, we called the h1 tag directly with the bsObj.h1:

Connecting Reliably to Retrieve Web Data

- Web data can be messy due to poor formatting, missing tags and web sites going down.
- To handle such exception as “page not found” (e.g., HTTP error 404) or error 500 for “internal server error”, the urlopen function will throw the generic exception “HTTPError” which we need to handle with the following code3.

Connecting Reliably to Retrieve Web Data

```
from urllib.request import urlopen
from urllib.error import HTTPError

try:
        html =
    urlopen("http://www.pythonscraping.com/pages/page1.html")
except HTTPError as e:
        print(e)
        #return null, break, or do some other "Plan B"
else:
        #program continues. Note: If you return or break in the
        #exception catch, you do not need to use the "else" statement
```

Connecting Reliably to Retrieve Web Data

- If an HTTP error code is returned, the program now prints the error, and does not execute the rest of the program under the else statement.
- If the server is not found at all (if, say, http://www.pythonscraping.com was down, or the URL was mistyped), urlopen will throw an URLError. This indicates that no server could be reached at all, and, because the remote server is responsible for returning HTTP status codes, an HTTPError cannot be thrown, and the more serious URLError must be caught. We can add a check to see if this is the case using the following code4:

Connecting Reliably to Retrieve Web Data

```
from urllib.request import urlopen
from urllib.error import HTTPError
from urllib.error import URLError

try:
        html =
        urlopen("https://pythonscrapingthisurldoesnotexist.com")
except HTTPError as e:
        print(e)
except URLError as e:
        print("The server could not be found!")
else:
        print("It Worked!")
```

Connecting Reliably to Retrieve Web Data

- If the server is not found at all, then urlopen will throw an URLError and another check is needed to catch this condition as in code4.
- Other exceptions include accessing a non-existent tag which will cause BeautifulSoup to return a Noneobject.
- An attempt to access a tag on a Noneobject will result in AttributeError with for example (print(bsObj.nonExistentTag)

Connecting Reliably to Retrieve Web Data

- An attempt to call some other function on the Noneobject will result in another AttributeError.
- The call and the error are as follows:

  print(bsObj.nonExistentTag.someTag)
- which returns the exception:

  AttributeError: 'NoneType' object has no attribute 'someTag'

Connecting Reliably to Retrieve Web Data

- The code to handle the above exception is defined as a function getTitle which returns either the title of the page or a Noneobject if there was some problem with retrieving the page.
- This is shown as code5 below.

Connecting Reliably to Retrieve Web Data

```
from urllib.request import urlopen
from urllib.error import HTTPError
from bs4 import BeautifulSoup
def getTitle(url):
   try:
                html = urlopen(url)
   except HTTPError as e:
                return None
   try:
                bsObj = BeautifulSoup(html.read())
                title = bsObj.body.h1
   except AttributeError as e:
               return None
           return title
title = getTitle("http://www.pythonscraping.com/pages/page1.html")
if title == None:
                print("Title could not be found")
else:
               print(title)
```

Advanced HTML Parsing

- Given complex web HTML source, we can search for tags by attributes, work with lists of tags and parse tree navigation.
- Nearly every website you encounter contains stylesheets (CSS).
- CSS relies on the differentiation of HTML elements that might otherwise have the exact same markup in order to style them differently. That is, some tags might look like this:
-
- while others look like this:
-

Advanced HTML Parsing

- Web scrapers can easily separate these two different tags based on their class; for example, they might use BeautifulSoup to grab all of the red text but none of the green text.
- Because CSS relies on these identifying attributes to style sites appropriately, you are almost guaranteed that these class and ID attributes will be plentiful on most modern websites.

Advanced HTML Parsing

- Task 2: Create an example web scraper that scrapes the page located at http://www.pythonscraping.com/pages/warandpeace.html.
- Solution 2: We can grab the entire page and create a BeautifulSoup object with it using a program similar to the one used in code1 of scrapetest.py as code6a below.

Advanced HTML Parsing

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html = urlopen("http://www.pythonscraping.com/pages/warandpeace.html")
bsObj = BeautifulSoup(html)
```

- Using this BeautifulSoup object, we can use the findAll function to extract a Python list of proper nouns found by selecting only the text within tags using the following code6b:

```
nameList = bsObj.findAll("span", {"class":"green"})
for name in nameList:
print(name.get_text())
```

Advanced HTML Parsing

- If we combine the code6a and code6b as code6 and run, the program would list all the proper nouns in the text at the Ware and Peace web page in the order they appear.
- It gets the list of all tags on the page rather than just the first one. Then, the program iterates through all names in the list and prints name.get_text() in order to separate the content from the tags.
- The code6 is given below.

Advanced HTML Parsing

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html =
urlopen("http://www.pythonscraping.com/pages/warandpeace.
html")
bsObj = BeautifulSoup(html)
nameList = bsObj.findAll("span", {"class":"green"})
for name in nameList:
    print(name.get_text())
```

Advanced HTML Parsing

- BeautifulSoup's find() and findAll() are the two functions you can easily use to filter HTML pages to find lists of desired tags, or a single tag, based on their various attributes.
- Example use of these functions are:
  findAll(tag, attributes, recursive, text, limit, keywords)
  find(tag, attributes, recursive, text, keywords)
  findAll({"h1","h2","h3","h4","h5","h6"})
  findAll("span", {"class":{"green", "red"}})

Advanced HTML Parsing

- While the first two commands show the general use of the functions, the third call shows its use with the first two parameters and the 4th call shows its use with only the first parameter. The 4th call takes a Python dictionary of attributes and match tags containing any of the attributes in the list “h1”, “h2”, ..
- The third call returns both the green and red span tags in the HTML document.

Advanced HTML Parsing

- The first two arguments are most important and for the rest, check reference for their use.
- Other sections for more assigned reading and learning as needed are summarized below:
- 1. Working with other BeautifulSoup's objects (Tag objects, Navigatable string objects, The comment object).
- 2. Navigating trees: See the HTML for the web page on page 19.

Advanced HTML Parsing

- 3. Dealing with children and other descendants.
- 4. Dealing with siblings.
- 5. Dealing with parents.
- Other topics as use of regular expression, crawling to other pages, using APIs, storing data and reading documents can be assigned as extra reading and presentation topics.

Part III: Web Recommendation Systems

- 1. Recommender Systems, Terms and Principle
- 2. Recommender Systems Problems & Applications
- 3. Recommender Systems Models
  - i. Collaborative Filtering Models
    - Predicting Ratings with Neighborhood-based Methods
  - ii. Content-based Recommender Systems
  - iii. Knowledge-based Recommender Systems
- 4. Other topics as assigned reading
- References: Recommender Systems: The Textbook by Charu C. Aggarwal, 2016, Springer Publishers.

1. Recommender Systems, Terms and Principle

- What are Recommender Systems?
- The web is an important medium for business and E-Commerce transactions and it has made access to data for recommendation easy.
- Recommender systems enable using various types of user-preferences and requirements data to make recommendations.
- This topic is important and is the focus of the ACM Conference on Recommender Systems.

1. Recommender Systems, Terms and Principle

- The most well-known techniques in Recommender Systems include collaborative filtering methods, content-based methods and knowledge-based methods.
- Specialized methods for various data domains (e.g., music, restaurants) and contexts (e.g., time, location and social information) have been designed too.
- Application domains for recommendation systems include query log mining, news recommendation and computational advertising.

1. Recommender Systems, Terms and Principle

- We shall look at some topics in:
  - 1. Recommender systems algorithms and evaluation. E.g., collaborative filtering, content-based and knowledge-based methods.
  - 2. Recommendations in specific domains and contexts. For example, user looking for a restaurant would want to use their location as additional conext.
  - 3. Advanced topics and applications: E.g., group recommendations, multi-criteria recommendations. Example applications are news recommendations, query recommendations.

1. Recommender Systems, Terms and Principle

- What are Rec Systems User Feedback Input?
- Rec systems use various sources of data to infer customer interests.
- Web enables users to provide feedback about their likes and dislikes. For example, for TV programming with a content provider such as Netflix, users can provide feedback with a simple click of a mouse.
- A typical method for providing explicit feedback is through ratings where users select numerical values from a specific evaluation system (e.g., five-star system) to specify their likes and dislikes of various items.

1. Recommender Systems, Terms and Principle

- Other forms of feedback not explicitly expressed but collectible from the web are:
- (i) likes of an E-Commerce product or item can be inferred if the user buys or browses the item. Amazon.com uses this type of feedback to infer customers' interest.

1. Recommender Systems, Terms and Principle

- What are Recommendation Systems Terms?
- Recommendation analysis is based on the previous interaction between *users* and *items*.
- This is because past user interests are good indicators of future choices with collaborative and content-based approaches while for knowledge-based methods, user-specified requirements guide future recommendations.
- The *user* is the entity to which recommendation is provided and an *item* is the product or service being recommended.

1. Recommender Systems, Terms and Principle

- What is the Basic Principle of Recommendation Algorithms?
- The basic principle of recommendations is that dependencies exist between user and item-centered activities.
- The goal of recommendation analysis algorithms is to represent users and items interactions in a ratings matrix and use learning algorithms (e.g., classification, clustering, association rule and sequential pattern mining) in a data-driven manner on ratings matrix to define user-item dependencies model which are used to make predictions for new or targeted users.

1. Recommender Systems, Terms and Principle

- The ratings matrix can be defined at a coarser item level (such as category of books: documentary, science, romance, games) or more detailed (fine granularity) item level (such as topic or author level: rain, fractions, dating, lego).
- An example dependency and recommendation that can be discovered by these algorithms is: user 1 is interested in historical documentary books or movies but we do not know what their interests in educational program and action movies would be.
- We can use their interest or rating on historical documentary movies to recommend to user 1 an educational program rather than an action movie.

1. Recommender Systems, Terms and Principle

- The larger the number of rated items that are available for a user, the easier it is to make robust predictions about the future behavior of the user.
- The collection of buying or rating behavior of various users can be leveraged to create cohorts of similar users interested in similar products.
- The interests and actions of these cohorts can be combined to make recommendations to individual members of these groups.

1. Recommender Systems, Terms and Principle

- The family of recommendation algorithms using the approach described above is called neighborhood models which are collaborative filtering algorithms.
- Collaborative filtering algorithms use ratings from multiple users in a collaborative way to predict missing ratings.
- In content-based recommendation systems, the content (the attributes' descriptions (e.g., price of the items) play a primary role in the recommendation process. Thus, both user ratings and attribute descriptions of items are used to make predictions.

1. Recommender Systems, Terms and Principle

- Here, the idea is that user interests can be modeled on the basis of properties (or attributes) of the items they have rated or accessed in the past.
- With knowledge-based systems, users interactively specify their interests and the user specification is combined with domain knowledge to provide recommendations.
- In advanced models, contextual data such as temporal, location, social or network information may be used.

2. Recommender Systems Problems & Applications

- Different Formulations of Recommendation Problem
- A recommendation problem can be formulated as one of the two versions:
- 1. Prediction Version: Prediction of an item to recommend to a user (that is, predicting ratings of items or users): This problem consists of predicting the rating value for user-item combination.
- This problem is called the m x n matrix R completion problem for m users and n items, where the entry (i,j) of matrix R (typically incomplete), indicates the rating (or preference) of user i for item j and is denoted as r_{ij} when it is actually observed and it is denoted as $\hat{r}_{ij}$ (with a hat) when the matrix entry is predicted by the rec algorithm.

2. Recommender Systems Problems & Applications

- Note that the matrix R is incomplete because some of its entries are observed data used for training while the other missing entries are to be predicted by the rec algorithms to complete the matrix and this is why it is called the R matrix completion problem.
- 2. Ranking Version of the Problem: Recommend the top-k items for a particular user or determine the top-k users to target for a particular item. This is called the top-k recommendation problem.
- Although the solution from version 1 can be ranked to solve version 2, direct methods for solving the ranking problem exist and are discussed in Chapter 13 of book.

2. Recommender Systems Problems & Applications

- Goals of Recommendation Systems
- The primary goal of recommendation systems is increasing product sales and revenue for merchants.
- In order to achieve the revenue goal, the operational and technical goals of rec system are:
- 1. Relevance: The most important goal of a rec system is to recommend items that are relevant to users.

2. Recommender Systems Problems & Applications

- 2. Novelty: Rec systems are helpful if they can recommend new relevant items that users have not seen in the past.
- 3. Serendipity: Recommended items are unexpected and relevant and seen as lucky discovery. It is a surprising discovery rather than an unknown discovery as in novelty.
- 4. Increasing Recommendation Diversity: Recommending a list of top-k diverse items has the benefit of ensuring that the user does not get bored of similar items.

2. Recommender Systems Problems & Applications

- 5. Other Soft Goals include:
- i) Increasing user satisfaction with recommendation web site that does good and reliable recommendations.
- ii) Providing explanation for why an item is recommended will increase user satisfaction and loyalty.
- iii) Recommending social connections to the items so as to increase usability of the recommendation site and its profit.

2. Recommender Systems Problems & Applications

- Example Applications of Recommendation Systems are:
- 1. GroupLens Recommender System for Usenet News which collected ratings from Usenet readers and predicted whether other readers would like an article before they read it.
- Extensions to other products are BookLens, MovieLens, Benchmark datasets for some of these applications were also released.

2. Recommender Systems Problems & Applications

- 2. Amazon.com Recommender system.
- 3. Netflix movie recommender system.
- 4. Google news personalization systems.
- 5. Facebook friend recommendation system.

- Students can explore any of these systems in detail and develop applications based on their methodologies for project.

2. Recommender Systems Problems & Applications

- Spectrum of Recommendation Applications
- Many of the rec systems are focused on traditional E-commerce applications for various products such as books, movies, videos, travel, and other goods and services.
- Recommender systems have expanded from traditional product recommendation to computational advertising which advertises other products along with specific product recommendation (e.g., Google Search).

3. Recommender Systems Models

- Basic Models of Recommender Systems
- 1. Collaborative Filtering (CF) Models
- CF are based on observed ratings of users for items placed in a matrix of m user ratings of n items.
- Some of the matrix entries are observed training data while the others are missing values to be predicted by recommendation algorithms using techniques similar to, for example, data mining classification algorithms.

3. Recommender Systems Models

- Two types of methods used in collaborative filtering are:
- 1. Memory-based methods: These are also called neighborhood-based CF algms. With this method, the user-item ratings are predicted on the basis of their neighborhoods which can be defined as:
- i) User-based CF: Here, ratings of like-minded users of a target user A are used to make recommendations for A. The weighted ratings of this group of neighbors can be used as the predicted rating of the targeted user A. Similarly, functions can be computed between the rows of the ratings matrix to discover similar users.

3. Recommender Systems Models

- ii) Item-based CF: To determine the rating predictions for target item B by user A, first find a set of S items that are most similar to target item B. The ratings in item set S are used to predict whether the user A will like item B.
- Memory-based algms are easy to implement and explain but do not work well with sparse ratings matrices.
- For example, we might not be able to find full coverage of rating predictions for similar users to target user Bob, who have rated the movie Gladiator so we can use their rating to predict Bob's Gladiator rating.

3. Recommender Systems Models

- When only the top-k items are required, lack of full coverage of rating predictions is not an issue.
- 2. Model-based methods: Here, machine learning and data mining techniques are used as predictive model where parameters can be learned .
- Example of such model-based methods are decision trees, rule-based models, Bayesian methods, latent factor models some of which have a high level of coverage even for sparse matrices.

3. Recommender Systems Models

- Types of Ratings
- Ratings are specified on a scale indicating the specific level of like or dislike of item at hand. Ratings can be:
- 1. Continuous valued ratings: E.g., ratings take any value between -10 and 10.
- 2. Interval-based ratings: These are discrete values as a 5-point rating scale from the set
  {-2, -1, 0, 1, 2} or {1, 2, 3, 4, 5} representing from left to right, extreme dislike to extreme like.
- The use of 5-point, 7-point and 10-point ratings are common.

3. Recommender Systems Models

- 3. Ordinal Rating: Ordered categorical values such as {strongly disagree, disagree, neutral, agree, strongly agree}.
- 4. Binary Rating: The user may represent only a like with 1 and a dislike with 0.
- 5. Unary Rating: Allows specifying like but not dislike, e.g., implicit facebook data sets.
- The implicit feedback are derived from cutomer activities rather than from explicitly specified ratings.
- A ratings matrix is called utility matrix and some examples are given next.

3. Recommender Systems Models

- Examples of utility matrices for 6 movies and 6 users.
- The movies denoted as: Gl for Gladiator, Go for Godfather, Be for Ben-Hur, Gf for Goodfellas, Sc for Scarface and Sp for Spartacus. (a) Ordered ratings

	Gl	Go	Be	Gf	Sc	Sp
U1	1			5		2
U2		5			4	
U3	5	3		1		
U4			3			4
U5				3	5	
U6	5		4			

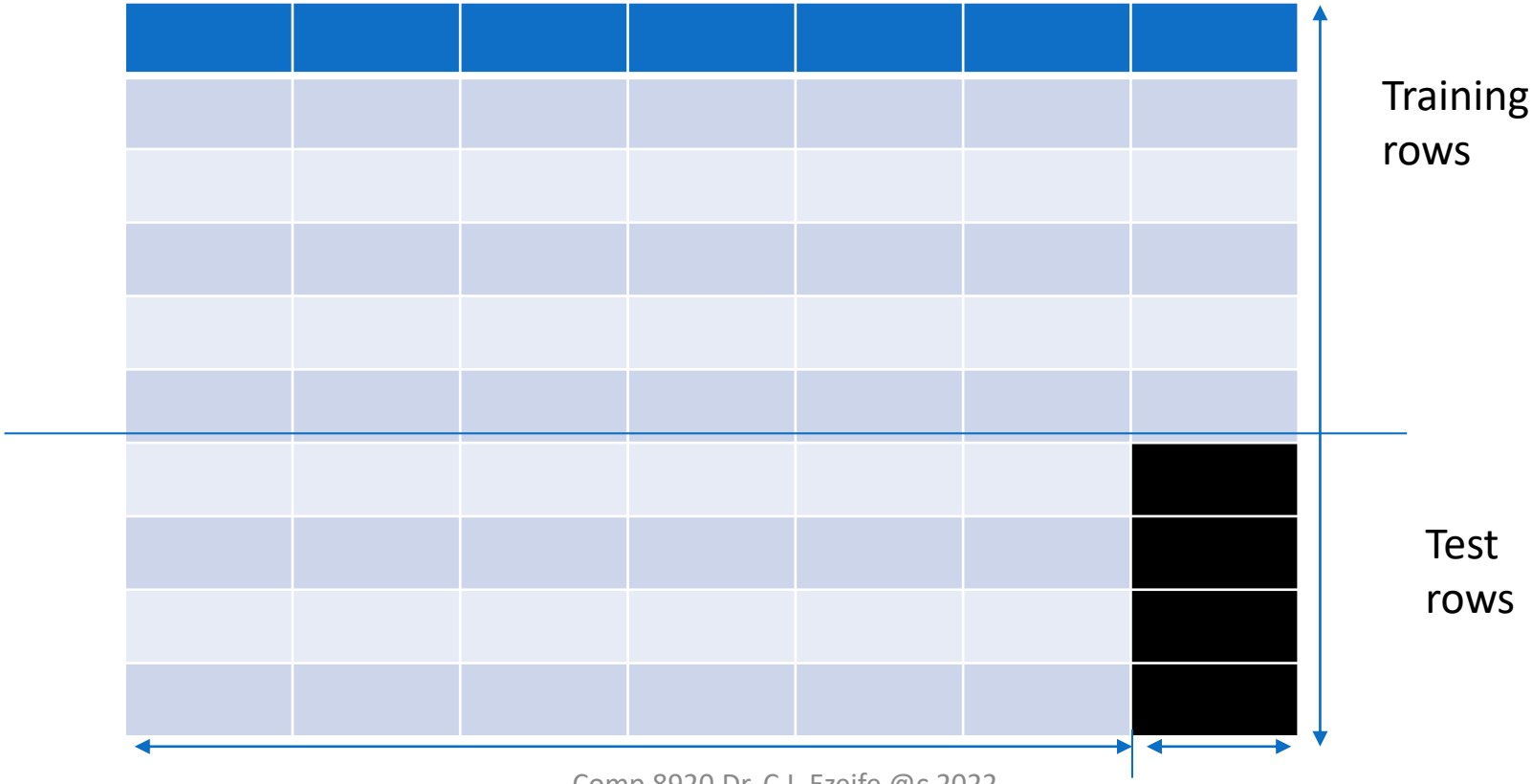
3. Recommender Systems Models

- Examples of utility matrices for 6 movies and 6 users.
- The movies denoted as: Gl for Gladiator, Go for Godfather, Be for Ben-Hur, Gf for Goodfellas, Sc for Scarface and Sp for Spartacus. (b) Unary ratings

	Gl	Go	Be	Gf	Sc	Sp
U1	1			1		1
U2		1			1	
U3	1	1		1		
U4			1			1
U5				1	1	
U6	1		1			

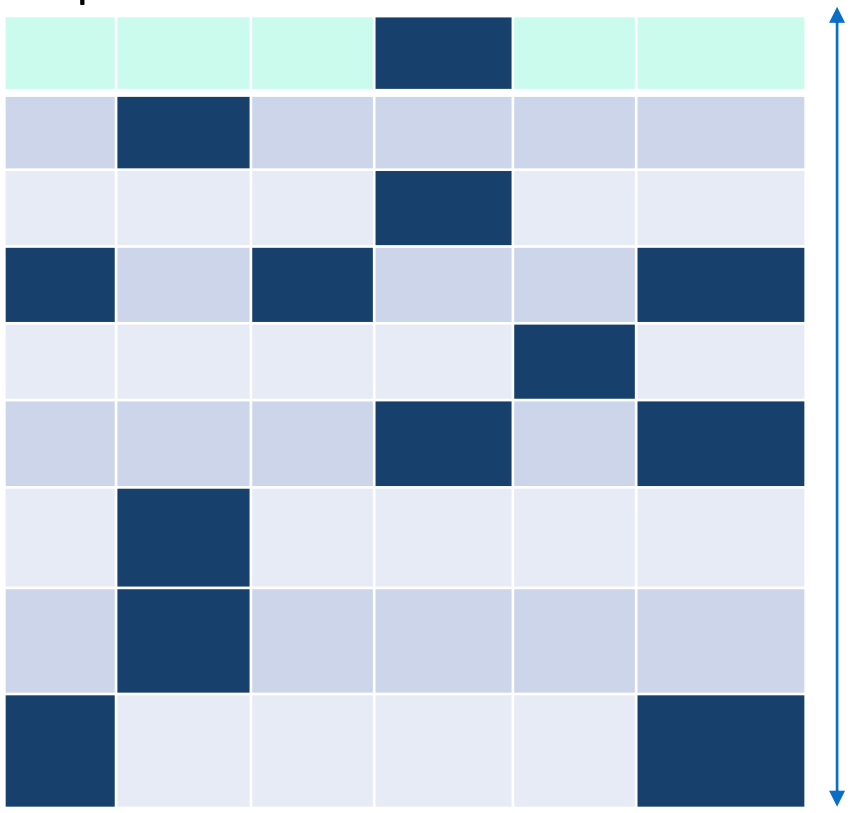
3. Recommender Systems Models

- Comparing Classification with CF entries
- (a) Classification: left attrs are independent var while the rightmost is dependent variable



3. Recommender Systems Models

- (b) is collaborative filtering example which has no demarcation of matrix entries into training and test data or column variables into independent and dependent variables as in classification.



No demarcaatiions

3. Recommender Systems Models

- Neighborhood-Based Collaborative Filtering
- Predicting Ratings with Neighborhood-Based Methods
- The basic idea in neighborhood-based methods is to use either user-user similarity or item-item similarity to make recommendations from a ratings matrix.
- Neighborhood consists of similar users or similar items.
- The two basic principles in neighborhood-based models are:

3. Recommender Systems Models

- 1. User-based models where similar users have ratings on the same item. For example, if Alice and Bob have rated movies in a similar way in the past, then, we can use Alice's observed ratings on the movie 'Terminator' to predict Bob's unobserved ratings on this movie.
- 2. Item-based models: Similar items are rated in a similar way by the same user. For example, Bob's ratings on similar Science fiction movies like Alien and Predator can be used to predict his rating on Terminator.

3. Recommender Systems Models

- Collaborative filtering problem can be viewed as a generalization of the classification/regression modeling problem.
- Neighborhood-Based Methods is a generalization of nearest neighbor classifiers.
- Unlike classification, where the nearest neighbors are determined only on the basis of row similarity, in collaborative filtering, the nearest neighbors are determined on the basis of either rows or columns.
- This is because all missing entries are concentrated in a single column in classification but they are spread over different rows and columns in Collaborative filtering.

3. Recommender Systems Models: User-Based Neighborhood Models

- User-Based Neighborhood Models
- Here user-based neighborhoods are defined in order to identify similar users to the target user i.
- In order to find the neighborhood of target user I, her similarity to all the other users is computed.
- Thus, a similarity function needs to be defined between the ratings specified by users.
- Some difficulties with user ratings include:
- a) Lack of uniform scale for rating by users. For example, one user may be biased toward liking most of the items while another user may be biased toward not liking most of the items.

3. Recommender Systems Models: User-Based Neighborhood Models

- b) Different users may have rated different (not the same) collection of items. For example, user u rates items 1, 3, 5, while user v rates items 1, 2, 3, 4.
- Then, the set of rated items by both users u and v is given by $I_u \cap I_v = \{1, 3, 5\} \cap \{1, 2, 3, 4\} = \{1, 3\}$.
- The set $I_u \cap I_v$ stands for the mutually observed ratings, which are used to compute the similarity between the uth and the vth users of neighborhood computation.
- One measure that captures the similarity Sim(u, v) between the rating vectors of two users u and v is the Pearson correlation coefficient.

3. Recommender Systems Models: User-Based Neighborhood Models

- The Pearson coefficient is computed only on the set of item indices for which both user u and user v have specified ratings and the coefficient is computed only on this set of items, that is on $I_u \cap I_v = = \{1, 3\}$.
- Given an incomplete ratings matrix R of m users and n items, and the missing ratings ($\hat{r}_{uj}$) of some item j for a user u, the user-based neighborhood model algorithms would go through the following sequence of steps to predict the unknown ratings ($\hat{r}_{uj}$) .

3. Recommender Systems Models: User-Based Neighborhood Models

- User-based Neighborhood Models Algorithm
- Step 1: Compute the mean rating μ_u for each user u using the specified ratings as:
- $\mu_u = \frac{\sum_{k \varepsilon I_u} r_{uk}}{|I_u|}$ for all u ε {1 …m} ……. Eqn 2.1
- Where r_{uk} is the observed rating of user u for item k.
- Step 2: Compute the similarity between the target user v and all other users u as Define the Pearson correlation coefficient between the rows (users) u and v as:

3. Recommender Systems Models: User-Based Neighborhood Models

- Sim(u, v) = Pearson(u, v) =

$$\frac{\sum_{k \in (Iu \cap Iv)} (r_{uk} - \mu_u) \cdot (r_{vk} - \mu_v)}{\sqrt{\sum_{k \in (Iu \cap Iv)} (r_{uk} - \mu_u)^2} \cdot \sqrt{\sum_{k \in (Iu \cap Iv)} (r_{vk} - \mu_v)^2}} \quad \text{.... Eqn 2.2}$$

* Note that traditional definition of Pearson(u, v) mandates using only the values of items rated by both users ($I_u \cap I_v$) to compute the μ_u and μ_v in eqn 2.1 although here we are using only one mean rating for each user.

- With the traditional approach, the μ_u and μ_v and the Pearson coefficient are computed in pairwise fashion between the target user and all other users.
- Also, other similarity functions such as the Cosine function (Cosine(u, v)) may be used.

$$\text{Cosine}(u,v) = \frac{\sum_{k \in (Iu \cap Iv)} (r_{uk} \cdot r_{vk})}{\sqrt{\sum_{k \in (Iu \cap Iv)} (r_{uk})^2} \cdot \sqrt{\sum_{k \in (Iu \cap Iv)} (r_{vk})^2}} \quad \text{.... Eqn 2.2a}$$

3. Recommender Systems Models: User-Based Neighborhood Models

- Step 3: Compute user u's peer group $P_u(j)$ for item j as follows:
- (a) Compute the peer group of the target user as the set of k users having the highest Pearson coefficient with the target user.
- Or
- (a) Find the closest k users for the target user separately for each predicted item such that each of these k users have specified ratings for that item.
- (b) Find the predicted rating for each item of target user I as the weighted average of these k closest ratings for the item. Here, rating is weighted with the Pearson correlation coefficient of its owner user to the target user.

3. Recommender Systems Models: User-Based Neighborhood Models

- To minimize the effect of observed ratings from users on varying scales, the raw ratings need to be mean-centered in row-wise fashion before finding the weighted average rating of the peer group.
- The mean-centered rating S_{uj} of a user u for item j is defined by subtracting her mean rating from the raw rating r_{uj}.
- $S_{uj} = r_{uj} - \mu_u$, for all u ε {1 …m} ……. Eqn 2.3
- Then, the mean rating of the target user is added back to this prediction to provide a row rating prediction $\hat{r}_{uj}$ for target user u for item j.
- $P_u(j)$ = the set of k closest users to target user u, who have specified ratings for item j.

3. Recommender Systems Models: User-Based Neighborhood Models

- Step 4: Compute the predicted ratings ($\hat{r}_{uj}$) for target user u for item j
- The overall neighborhood-based prediction function for computing rating can be done using
- (a) Pearson-weighted average as in the eqn 2.4 below is:

$$\hat{r}_{uj} = \frac{\sum_{v \in Pu(j)} r_{uj} * Sim(u,v)}{\sum_{v \in Pu(j)} Sim(u,v)} \quad \text{.... Eqn 2.4}$$

Note that when a different measure as Cosine(u,v) is used, then, that replaces Sim(u,v)

3. Recommender Systems Models: User-Based Neighborhood Models

- (b) A more relative prediction can be made using mean-centered equation 2.4a given below:

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \,\varepsilon\, Pu(j)} Sim(u,v).\, S_{vk}}{\sum_{v \,\varepsilon\, Pu(j)} |Sim(u,v)|}$$

$$= \mu_u + \frac{\sum_{v \,\varepsilon\, Pu(j)} Sim(u,v).\, (r_{vk} - \mu_v)}{\sum_{v \,\varepsilon\, Pu(j)} |Sim(u,v)|} \quad \text{.... Eqn 2.4a}$$

3. Recommender Systems Models: User-Based Neighborhood Models

- Example Application of the User-Based Algorithm
- Given the incomplete ratings matrix of Figure 2.1 on page 34 of book for 5 users and 6 items, predict the unknown ratings for user 3 on items 1 and 6 or compute the ratings $\hat{r}_{31}$ and $\hat{r}_{36}$. The ratings are drawn from the range of values {1…7}.

Item id > User id v	**1**	**2**	**3**	**4**	**5**	**6**
1	7	6	7	4	5	4
2	6	7	?	4	3	4
3	?	3	3	1	1	?
4	1	2	2	3	3	4
5	1	?	1	2	3	3

3. Recommender Systems Models: User-Based Neighborhood Models

- Step 1: Compute the Mean rating of each user u with eqn 2.1. The mean ratings for users 1 to 6 are respectively: 5.5, 4.8, 2, 2.5 an 2.
- Step 2: Compute the similarity between user 3 and all other users using either the Pearson correlation coefficient, e.g., Pearson(1,3) between user u and target user v=3 as given in eqn 2.2. You may also use a different similarity measure as Cosine(1,3) using eqn 2.2a.
- For our example, Pearson(1,3) = 0.894
- Cosine(1,3) = 0.956

3. Recommender Systems Models: User-Based Neighborhood Models

- Following table shows the Table 2.1 with computed Mean rating, Cosine and Pearson similarity measures.

Item id > User id v	1	2	3	4	5	6	Mean Rating	Cosine(I,3) (user-user)	Pearson(I,3) (user-user)
1	7	6	7	4	5	4	5.5	0.956	0.894
2	6	7	?	4	3	4	4.8	0.981	0.939
3	?	3	3	1	1	?	2	1.0	1.0
4	1	2	2	3	3	4	2.5	0.789	-1.0
5	1	?	1	2	3	3	2	0.645	-0.817

3. Recommender Systems Models: User-Based Neighborhood Models

- The Pearson correlation coefficient is more discriminative and the sign of the coefficient provides information about similarity and dissimilarity.
- The top two closest users to user 3 are users 1 and 2 according to both measures.
- Step 4: Compute the predicted ratings ($\hat{r}_{uj}$) or $\hat{r}_{31}$ and $\hat{r}_{36}$ for target user u for item j using eqn 2.4 or 2.4a.

3. Recommender Systems Models: User-Based Neighborhood Models

- The Pearson-weighted average (eqn 2.4) of the raw ratings of users 1 and 2 for predicting user 3's unrated items 1 and 6 are:
- $\hat{r}_{31} = (7 * 0.894 + 6 * 0.939)/(0.894 + 0.939) = 6.49$
- $\hat{r}_{36} = (4 * 0.894 + 4 * 0.939)/(0.894 + 0.939) = 4.$
- This means that item 1 should be given higher priority over item 6 as a recommendation to user 3.
- Also, the prediction suggests that user 3 is likely to be interested in both movies 1 and 6 to a greater degree than any of the movies she has already rated.

3. Recommender Systems Models: User-Based Neighborhood Models

- This is a result of the bias caused by the fact that peer group {1, 2} of user indices is a far more optimistic group with positive ratings, when compared to the target user 3.
- A more relative prediction that takes into account the bias or scaling difference between the target user and their peer group is mean-centered computation (eqn 2.4a) with the following predictions:
- $\hat{r}_{31} = 2 + (1.5 * 0.894 + 1.2 * 0.939)/(0.894 + 0.939) = 3.35$
- $\hat{r}_{36} = (-1.5 * 0.894 + 4\text{-}0.8 * 0.939)/(0.894 + 0.939) = 0.86$

3. Recommender Systems Models: User-Based Neighborhood Models

- The mean-centered computation also provides the prediction that item 1 should be prioritized over item 6 as a recommendation to user 3.
- Note that with mean-centered equation, the predicted rating of item 6 is only 0.86 which is less than in the previous case where the predicted rating of item 6 for user 3 is higher than all the other users ratings for that item.
- On further inspection of Table 2.1, it can be seen that item 6 ought to be rated very low by user 3 (compared to her other items), since her closest peers (users 1 and 2) have also rated it lower than their other items.

3. Recommender Systems Models: User-Based Neighborhood Models

- Thus, the mean-centered process enables a better relative prediction with respect to the ratings that have been observed.
- Sometimes, it can affect also the relative order of the predicted items.
- The only weakness in this result is that the predicted rating of item 6 is 0.85, which is outside the range of allowed ratings.
- Such ratings can always be used for ranking and the predicted value can be corrected to the closest value in the allowed range.

3. Recommender Systems Models and Other Topics

- Other topics on recommender system models are left as assigned reading and they include the following:

- ii. Content-based Recommender Systems
- iii. Knowledge-based Recommender Systems

- 4. Other topics as assigned reading