Compsci 101
Stable Sorting

Susan Rodger

November 17, 2022

U is for …

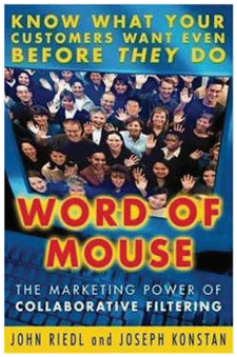
- **URL**
  - https://duke.edu
- **Usenet**
  - Original source of FAQ, Flame, Spam, more
- **UI and UX**
  - User is front and center

John Riedl

- **Co-Inventor of Recommender systems**
- **PhD at Purdue University**
- **Professor at Univ. of Minnesota**
- **ACM Software System Award – GroupLens System**
- **Died of cancer in 2013**

- **Quote from his son about John:**

"He once looked into how likely people are to follow your book recommendations based on how many books you recommend. We went to his talk at the AH Conference in which he described the answer. It turns out that if you recommend too many books to people, they get overwhelmed and are less likely to follow your suggestions. As he told us in his talk, the optimal number of books to recommend turns out to be about two. Then he proceeded to recommend eight books during the talk."

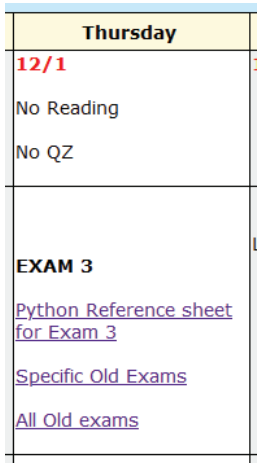


Announcements

- **APT-6 due Tuesday, Nov 29**
- **Assignment 5 Clever GuessWord due tonight**
- **Lab 9 Friday**
  - There is a prelab!
- **No reading or QZ until last week of classes**
- **Exam 3 is December 1**

Exam 3 – in person – Thurs, Dec 1

- **Exam is in class on paper – 10:15am**
  - Need pen or pencil
- **See materials under 12/1 date**
  - Exam 3 Reference sheet - part of exam
- **Covers**
  - topics /reading through today
  - APTs through APT6
  - Labs through Lab 9
  - Assignments through Assignment 5, parts of Assign 6 helpful



Exam 3 topics include …

- **List, tuples, list comprehensions**
- **Loops – for loop, while loop, indexing with a loop**
- **Reading from a file**
  - Converting data into a list of things
- **Parallel lists**
- **Sets – solving problems**
- **Dictionaries – solving problems**
- **Sorting – lists, tuples**
- **No turtles, no images - but note we are practicing other concepts with images**

Exam 3

- **Exam 3 is your own work!**
- **No looking at other people's exam**
- **You cannot use any notes, books, computing devices, calculators, or any extra paper**
- **Bring only a pen or pencil**
- **The exam has extra white space and has the Exam 3 reference sheet as part of the exam.**

- **Do not discuss any problems on the exam with others until it is handed back**

Exam 3 – How to Study

- **Practice writing code on paper!**
- **Rewrite an APT**
- **Try to write code from lecture from scratch**
- **Try to write code from lab from scratch**
- **Practice from old exams**
- **Put up old Sakai quizzes, but better to practice writing code**
- **Look at Exam 3 reference sheet when writing code!**

PFTD

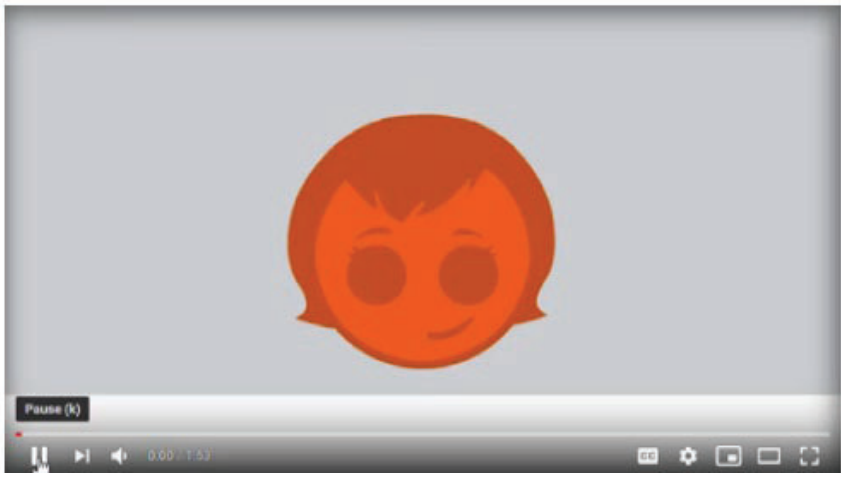
- **Recommender**
- **An APT**

Recommendation Systems: Yelp

- **Are all users created equal?**
- **Weighting reviews**
- **What is a recommendation?**

Recommendation Systems: Yelp

https://www.youtube.com/watch?v=PniMEnM89iY



Recommender Systems: Amazon

- **How does Amazon create recommendations?**

Recommendation Systems: Netflix

- **Netflix offered a prize in 2009**
  - Beat their system? Win a million $$
  - http://nyti.ms/sPvR

Compsci 101 Recommender

- **Doesn't work at the scale of these systems, uses publicly accessible data, but ...**
  - Movie data, food data, book data
- **Make recommendations**
  - Based on ratings, how many stars there are
  - Based on weighting ratings by users like you!
- **Collaborative Filtering: math, stats, compsci**

Where to eat? Simple Example

Tandoor	IlForno	McDon	Loop	Panda	Twin
0	3	5	0	-3	5
1	1	0	3	0	-3
-3	3	3	5	1	-1

- Rate restaurants on a scale of (low) -5 to 5 (high)
  - Each row is one user's ratings
  - But a zero/0 means no rating, not ambivalent
- What restaurant should I choose to go to?
  - What do the ratings say? Let's take the average!

Calculating Averages

- **What is average rating of eateries?**

Tandoor	IlForno	McDon	Loop	Panda	Twin
0	3	5	0	-3	5
1	1	0	3	0	-3
-3	3	3	5	1	-1

- **Tandoor:**

Python Specification

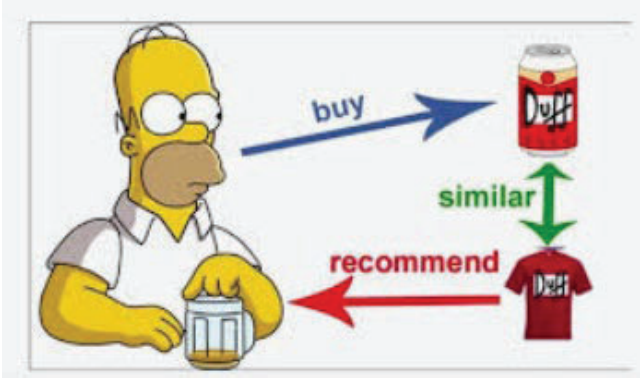
- **Items: list of strings (header in table shown)**

```
items = ["DivinityCafe", "FarmStead", "IlForno", "LoopPizzaGrill",
         "McDonalds", "PandaExpress", "Tandoor", "TheCommons",
         "TheSkillet"]
```

- **Values in dictionary are ratings: int list**
  - **`len(ratings[i]) == len(items)`**

```
ratings = \
          {"Sarah Lee"  : [3, 3, 3, 3,  0,-3, 5, 0,-3],
           "Melanie"    : [5, 0, 3, 0,  1, 3, 3, 3, 1],
           "J J"        : [0, 1, 0,-1,  1, 1, 3, 0, 1],
           "Sly one"    : [5, 0, 1, 3,  0, 0, 3, 3, 3],
           "Sung-Hoon"  : [0,-1,-1, 5,  1, 3,-3, 1,-3],
           "Nana Grace": [5, 0, 3,-5, -1, 0, 1, 3, 0],
           "Harry"      : [5, 3, 0,-1, -3,-5, 0, 5, 1],
           "Wei"        : [1, 1, 0, 3, -1, 0, 5, 3, 0]
          }
```


WOTO-2 Averages
http://bit.ly/101f22-1117-1



Recommender averages

- **`def averages(items,ratings):`**

- ***Input: items -- list of restaurants/strings***
- ***Input: dictionary of rater-name to ratings***
  - ratings: list of ints, **`[1,0,-1, ... 1]`** -- parallel list to list of restaurants
    - k^{th} rating maps to k^{th} restaurant
- ***Output: recommendations***
  - List of tuples (name, avg rating) or (str, float)
    - Sort by rating from high to low

Drawbacks of Averaging, Instead …

- **Are all user's ratings the same to me?**
  - Weight/value ratings of people most similar to me

- **Collaborative Filtering**
  - https://en.wikipedia.org/wiki/Collaborative_filtering
  - How do we determine who is similar to/"near" me?

- **Mathematically: treat ratings as vectors in an N-dimensional space, N = # of items that are rated**
  - a.k.a. weight has higher value → closer to me

Determining "closeness"

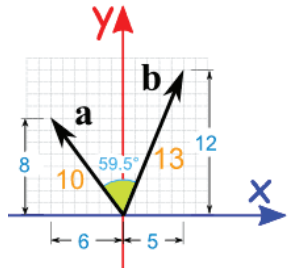
- **Calculate a number measuring closeness to me**
  - The higher the number, the closer to me
  - I'm also a rater, "me" is parameter to function

- **Function:**
  - `similarities("rodger", ratings)`

What's close? Dot Product

- https://en.wikipedia.org/wiki/Dot_product
  - For [3,4,2] and [2,1,7]
    - `3*2 + 4*1 + 2*7 = 6+4+14 = 24`

- **How close am I to each rater?**
- **What happens if the ratings are**
  - Same sign? Me: 3, -2 Other: 2, -5
  - Different signs? Me: -4 Other: 5
  - One is zero? Me: 0 Other: 4
- **What does it mean when # is…**
  - Big? Small? Negative?



Writing similarities

- **Given a name and a dictionary, return list of tuples**

```
def similarities(name, ratings):
    return [('name0', #), ...('nameN', #)]
```

- **What is the # here?**
  - Dot product of two lists
  - One list is fixed (name)
  - Other list varies (loop)

- **Think: How many tuples are returned?**

food.json

```
{"Sarah Lee":
   [3, 3, 3, 3, 0, -3, 5, 0, -3],
 "Melanie":
   [5, 0, 3, 0, 1, 3, 3, 3, 1],
  "J J":
   [0, 1, 0, -1, 1, 1, 3, 0, 1],
  "Sly one":
   [5, 0, 1, 3, 0, 0, 3, 3, 3],
  "Sung-Hoon":
   [0, -1, -1, 5, 1, 3, -3, 1, -3],
  "Nana Grace":
   [5, 0, 3, -5, -1, 0, 1, 3, 0],
  "Harry":
   [5, 3, 0, -1, -3, -5, 0, 5, 1],
  "Wei":
   [1, 1, 0, 3, -1, 0, 5, 3, 0]
}
```


Collaborative Filtering

- **Once we know raters "near" me? Weight them!**
  - How many raters to consider? 1? 10?
  - Suppose Fran is `[2, 4, 0, 1, 3, 2]`
- **What is Sam's similarity to Fran?**

	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

What is Chris's similarity and weights?

- **Suppose Fran is `[2, 4, 0, 1, 3, 2]`**
- **Chris's similarity is:**

	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

Recommendations

- **Get new weighted averages for each eatery**
  - Then find the best eatery I've never been to

```
def recommendations(name,items,ratings,numUsers)
   return [('eatery0', #), ...('eateryN', #)]
```

Fran gets a recommendation (considering numUsers raters)


```
rc = recommendations("Fran",items,ratings,3)
#use this to provide evals to Fran
```


Steps for Recommendations

- **Start with you, a rater/user and all the ratings**
  - Get similarity "weights" for users: dot product
- **Calculate new weighted ratings for all users**
  - **`[weight * r for r in ratings]`**
- **Based on these new ratings, find average**
  - Don't use zero-ratings
- **Check recommendations by … (not required)**
  - Things I like are recommended? If so, look at things I haven't tried!

Similarities Summarized

- **How do we get weighted ratings?**

	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1
Fran	2	4	0	1	3	2

```
def similarities(name,ratings):
   return [('name', #), ...('name', #)]

weights = similarities("Fran", ratings)
```

Making Recommendations

- **How do we get weighted ratings? Call average?**

	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1
Fran	2	4	0	1	3	2

```
weights = similarities("Fran", ratings)
weights = #slice based on numUsers
weightedRatings = {}. # new dictionary
for person, weight in weights:
    weightedRatings[?] = ?
```


Calculating Weighted Average

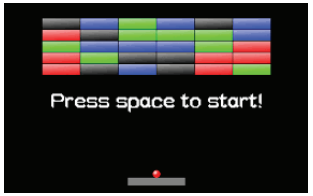
	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	39	65	0	-39	65
~~Chris~~	~~3~~	~~3~~	~~0~~	~~9~~	~~0~~	~~-9~~
Nat	-36	36	36	60	12	-12
Total	-36	75	101	60	-27	53
Avg	-36	37.5	50.5	60	-13.5	26.5

```
recommendations("Fran",items,ratings,2)
```

- Make recommendation for Fran? Best? Worst?
- Fran should eat at Loop! Even though only using Nat's rating
- But? Fran has been to Loop! Gave it a 1, … McDonalds!!!! ??

WOTO-2 Sims to Recs
http://bit.ly/101f22-1117-2

- **From Similarities to Recommendations**



Assignment Modules

RecommenderEngine
1.averages(…)
2.similaries(…)
3.recommendations(…)

RecommenderMaker
1.makerecs(…)

TestRecommender

MovieReader
1.getdata(…)

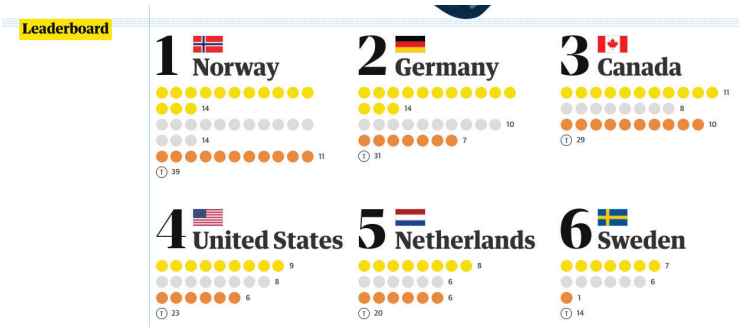
BookReader
1.getdata(…)

Function Call Ordering

- **Some_Reader_Module.getdata(…)**
- **RecommenderMaker.makerecs(…)**
  - RecommenderEngine.recommendations(…)
    - RecommenderEngine.similarities(…)
    - RecommenderEngine.averages(…)

APT: MedalTable

- **http://bit.ly/apt-medal-table**



APT: MedalTable

Problem Statement

The Olympic Games will be held, and have been held (and might be being held). Given the results of the olympic disciplines, generate and return the medal table.

The results of the disciplines are given as a `String list` results, where each element is in the format "GGG SSS BBB". GGG, SSS and BBB are the 3-letter country codes (three capital letters from 'A' to 'Z') of the countries winning the gold, silver and bronze medal, respectively.

Specification

```
filename: MedalTable.py

def generate(results):
    """
    return list of strings
    based on data in results, a list of strings
    """

    # you write code here
    return []
```

The medal table is a `String list` with an element for each country appearing in results. Each element has to be in the format "CCO G S B" (quotes for clarity), where G, S and B are the number of gold, silver and bronze medals won by country CCO, e.g. "AUT 1 4 1". The numbers should not have any extra leading zeros.

Sort the elements by the number of gold medals won in decreasing order. If several countries are tied, sort the tied countries by the number of silver medals won in decreasing order. If some countries are still tied, sort the tied countries by the number of bronze medals won in decreasing order. If a tie still remains, sort the tied countries by their 3-letter code in ascending alphabetical order.

1. `["ITA JPN AUS", "KOR TPE UKR", "KOR KOR GBR", "KOR CHN TPE"]`

   Returns:
   `[ "KOR 3 1 0", "ITA 1 0 0", "TPE 0 1 1", "CHN 0 1 0", "JPN 0 1 0", "AUS 0 0 1", "GBR 0 0 1", "UKR 0 0 1" ]`

Tracking the Data

- **What do we need to obtain for each country?**
  - What's the data, how do we store it?
  - What's the data, how do we calculate it?
- **Method and code to transform input**
  - What will we store, how do we initialize/update
  - Verifying we've done this properly

Sorting the Data

- **Write a helper function to build the dictionary**
  - d = builddict(results)
  - Where results is string of countries for each event
- **Use dictionary to get list of tuples**

```
[('JPN', [0, 1, 1]), ('KOR', [3, 1, 0]),
('TPE', [0, 1, 1]), ('UKR', [0, 0, 1])]
```

- **Then do passes to sort the data**
  - Will discuss sorting the data in lab

WOTO-3 Building Dictionary
http://bit.ly/101f22-1117-3