

Compsci 101

Recommender Assignment

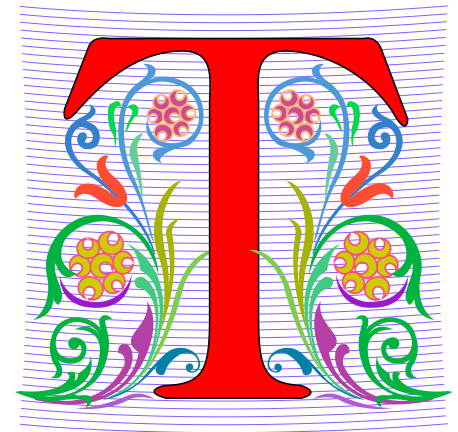
Live Lecture

Susan Rodger

April 6, 2021

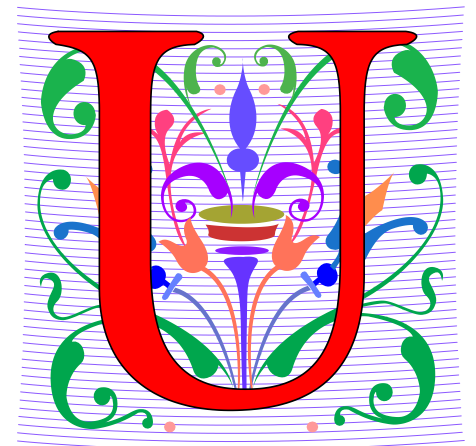
| Tandoor | IlForno | McDon | Loop | Panda | Twin |
|---------|---------|-------|------|-------|------|
| 0 | 3 | 5 | 0 | -3 | 5 |
| 1 | 1 | 0 | 3 | 0 | -3 |
| -3 | 3 | 3 | 5 | 1 | -1 |

T is for ...



- **Type**
 - From int to float to string to list to ...
- **Text**
 - From .txt to editors to ...
- **Turing Award**
 - Nobel, Fields, Turing
 - Turing Duke Alums:
 - Ed Clarke (MS)
 - John Cocke (BS, PhD)
 - Fred Brooks (BS)

U is for ...



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

Interested in being a UTA?

- Enjoy Compsci101?
- Would like to help others learn it?
- Consider applying to join the team!
- <https://www.cs.duke.edu/undergrad/uta>
- Apply soon

Announcements

- Assign 5 – Due today! April 6
- APT-7 due Thursday, April 8
- Assign 6 – Recommender out, due Thur. April 22
- APT Quiz 2 – April 8-11
- Exam 3 – in one week, April 13
- April 12-13 – No Consulting/office hours
 - Wellness day/Exam 3

PFTD

- APT Quiz 2
- Exam 3
- Recommender
 - Recommendations big picture
 - Assignment big picture
 - Simple recommendation example
 - Actual recommendation assignment

APT Quiz 2

- APT Quiz 2 is 4/8 8AM -4/11 11PM – finish by 11pm
- There are two parts – each part is 1.5 hours
- Pick a start time for each part,
 - Once you start a part, You have 1.5 hours
 - If you get accommodations, you get those
- 4 APTs to solve (2 in each part)
 - Take parts 1 and 2 on same day or different days
- **Start APT Quiz on Sakai!**
- See old APT Quiz 2 problems so you can practice
 - On APT page – NOT FOR CREDIT

APT Quiz 2

- **Is your own work!**
 - No collaboration with others!
 - Use your notes, lecture notes, your code, textbook
 - DO NOT search for answers!
 - Do not talk to others about the quiz until grades are posted
- **Post private questions on Piazza**
 - We are not on between 10pm and 8am EDT!
 - We are not on all the time
 - Will try to answer questions between 8am – 10pm

Protect your APT Quiz 2!

- Be defensive in both directions!
- Reduce risk others will see your code
 - Complete it in your dorm room, alone
 - Lock the door!
 - Do not do it in a public space
- Reduce risk you will see/know other's answers
 - Do not ask others about the quiz
 - Do the quiz alone
 - Can't ask for help if there's no one around
- If your code is suspiciously similar to another's, both of you are in trouble

Exam 3 Topics

- Everything from Exam 1 and 2
- Sets
- Dictionaries
- Sorting
 - `sort()` vs `sorted()`
 - Stable sorting
 - Lambda functions
 - Sorting on multiple criteria
- Problem solving
 - Given a problem, what do you use?
 - Greedy algorithm
 - Modules

Exam 3 Rules

- This is your own work, no collaboration
- Open book, Open notes
- Do not search for answers on the internet
- Do not type in code where it can be compiled and run
 - Do not use Pycharm, textbook code boxes, Python tutor or any other place the code can be run
- Do not talk to anyone about the exam during the exam, and until it is handed back!

Exam 3 Logistics

- Take on Tues. April 13 between 7am and 11pm
- You pick the start time
 - Must start by 9:15pm
- You get 1 hour 45 min
 - Longer if you have accommodations
- Once you start, your timer starts and you must finish in 1 hour, 45 minutes
- You cannot pause the timer
- See Python Reference Sheet for Exam 3
 - Added set and dictionary functions

Exam 3 Logistics (2)

- Go to Gradescope to start
 - login with your netid
 - select the CompSci 101 **Exam** site
 - Different site than where you turn in assignments
- Click on **Exam 3** to start
- Gradescope saves answers as you type them in
 - Type 4 spaces to indent code
- Disconnected? Just log back in to Gradescope
- Question? Post a private post on Piazza

Don't go to Gradescope site until
you are ready to start!

You click it, you have started!

We do not restart it!

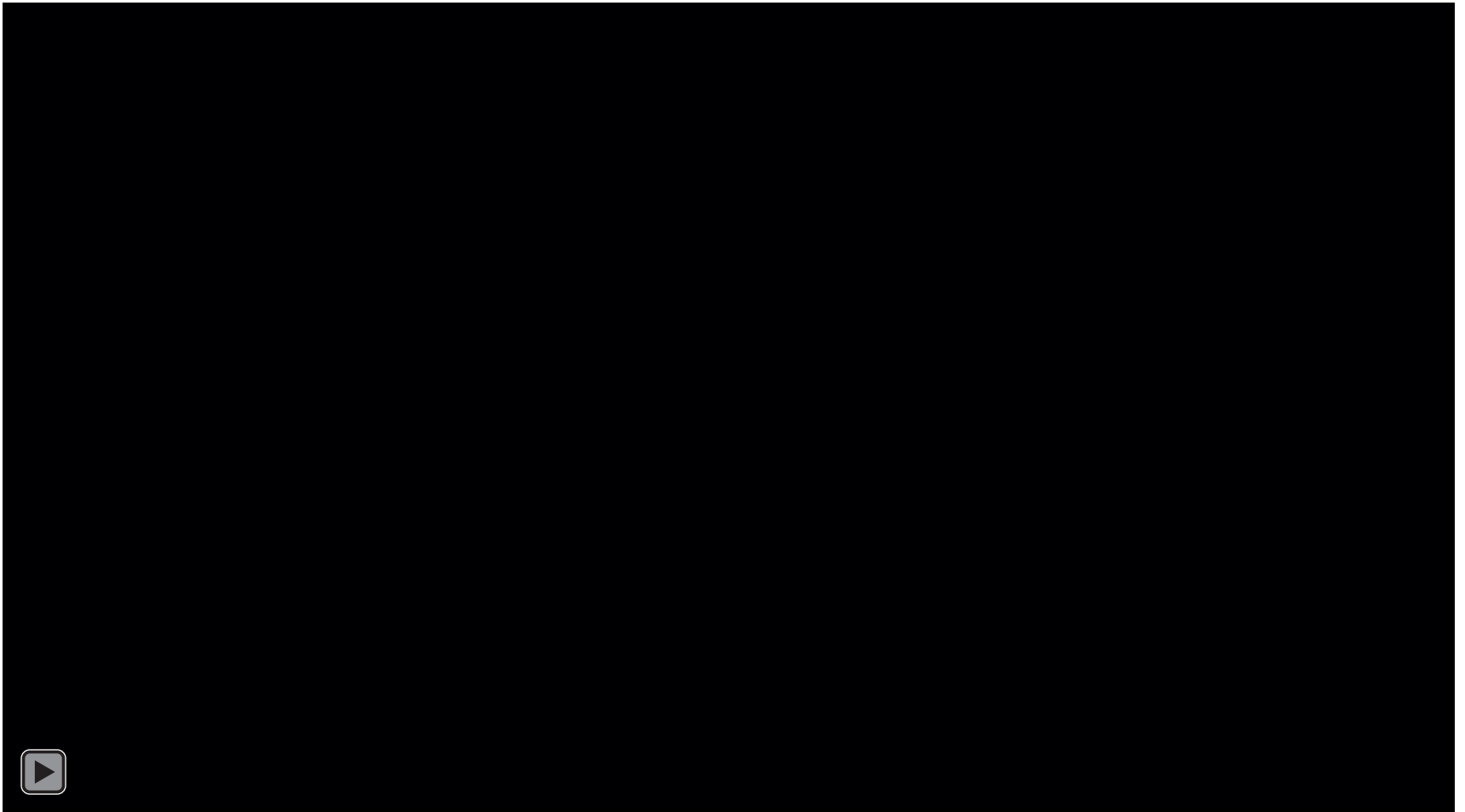
Recommendation Systems: Yelp

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



Recommendation Systems: Yelp

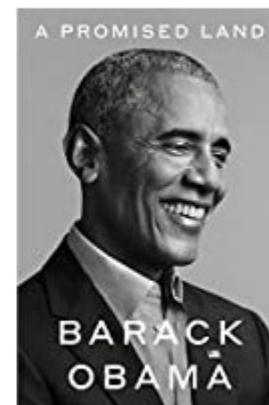
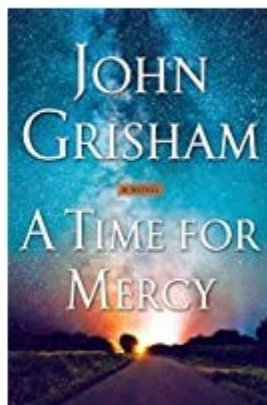
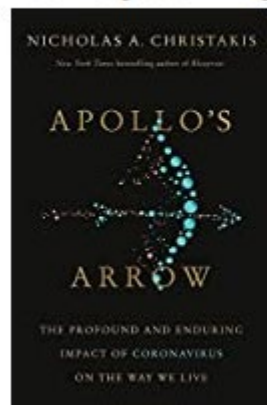
<https://www.youtube.com/watch?v=PniMEnM89iY>



Recommender Systems: Amazon

- How does Amazon create recommendations?

Books you may like



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
- Machine learning!

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)
 - 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: $(1 + -3) / 2 = -1.00$
 - Don't count rating if not rated

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: $(1 + -3) / 2 = -1.00$
 - Don't count rating if not rated
- Il Forno: $(3 + 1 + 3) / 3 = 2.33$
- Where should we eat? What's the best average
- Highest: $8/2 = 4$ for McDonalds and The Loop

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]  
}
```

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]  
}
```

index = 4

index = 4

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

WOTO-1 Averages

<http://bit.ly/101s21-0406-1>



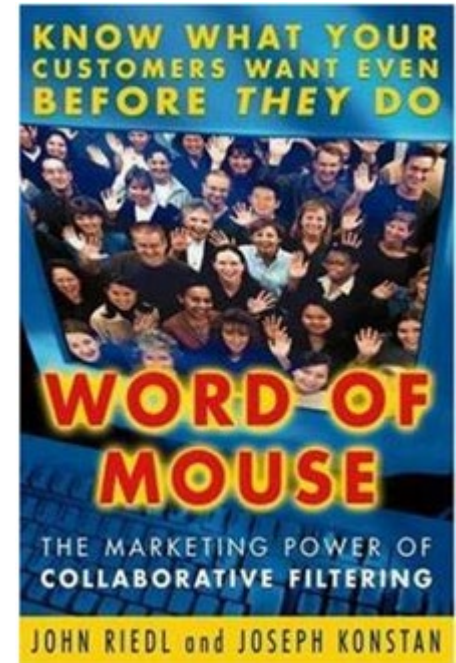
John Riedl

- 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.”



Drawbacks of Averaging

- 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 \rightarrow closer to me

Determining "closeness"

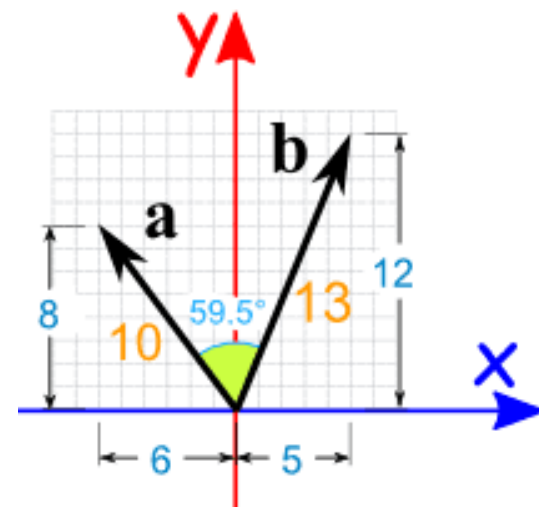
- Calculate a number measuring closeness to me
 - I'm also a rater, "me" is parameter to function

Same as before, dictionary
of rater to ratings

- Function:
 - **`similarities("rodger", ratings)`**
- Return **`[("rater1", #), ("rater2", #), ...]`**
 - List of tuples based on closeness to me
 - sorted high-to-low by similarity

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 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  
1  {"Sarah Lee":  
2    [3, 3, 3, 3, 0, -3, 5, 0, -3],  
3    "Melanie":  
4    [5, 0, 3, 0, 1, 3, 3, 3, 1],  
5    "J J":  
6    [0, 1, 0, -1, 1, 1, 3, 0, 1],  
7    "Sly one":  
8    [5, 0, 1, 3, 0, 0, 3, 3, 3],  
9    "Sung-Hoon":  
10   [0, -1, -1, 5, 1, 3, -3, 1, -3],  
11   "Nana Grace":  
12   [5, 0, 3, -5, -1, 0, 1, 3, 0],  
13   "Harry":  
14   [5, 3, 0, -1, -3, -5, 0, 5, 1],  
15   "Wei":  
16   [1, 1, 0, 3, -1, 0, 5, 3, 0]  
17 }
```

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 |

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?
 - $2*0 + 4*3 + 0*5 + 1*0 + 3*-3 + 2*5 = 13$
 - Sam's ratings **[0, 3, 5, 0, -3, 5]** * 13
 - Sam's weighted: **[0, 39, 65, 0, -39, 65]**

| | 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 |

Think:
What is
Chris's
similarity?

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 |

What is Chris's similarity and weights?

- Suppose Fran is **[2, 4, 0, 1, 3, 2]**
- Chris's similarity is:
 - $2*1 + 4*1 + 0*0 + 1*3 + 3*0 + 2*(-3) = 3$
- Chris' weighted ratings:
 - $3 * [1, 1, 0, 3, 0, -3]$
 - **[3, 3, 0, 9, 0, -9]**

| | 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 |

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!

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
```

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 |

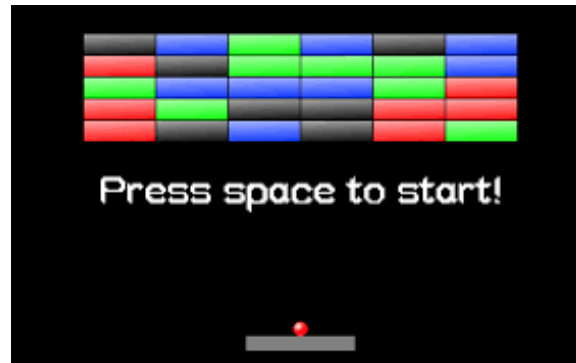
recommendations ("Fran", items, ratings, 2)

- 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/101s21-0406-2>

- From Similarities to Recommendations



Assignment Modules

Implement functions
in this order

RecommenderEngine

- 1. averages(...)
- 2. similaries(...)
- 3. recommendations(...)

RecommenderMaker

- 1. makerecs(...)

TestRecommender

Can be implemented before
Recommender stuff or after

MovieReader

- 1. getdata(...)

BookReader

- 1. getdata(...)

RecommenderEngine before
RecommenderMaker and use
TestRecommender

Function Call Ordering

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

Start with inner most call
and work outwards

Test on your computer
and on Gradescope as
you go!