# HUDK 4051: LEARNING ANALYTICS PROCESS & THEORY

2/4/20 2:22 PM

# Today

- Finish up with AWS
- Matching
- Cosine similarity
- Recommender

# Finish Up SQL

- SoloLearn
- LeetCode.com
- Test
- Projects
- Delete your AWS database

# Git/Github

**RStudio** Git

Working Directory

What you see

Add

Staging Area

Commit

Local Repository

Clone

Push

Pull

Friend's Remote Repository

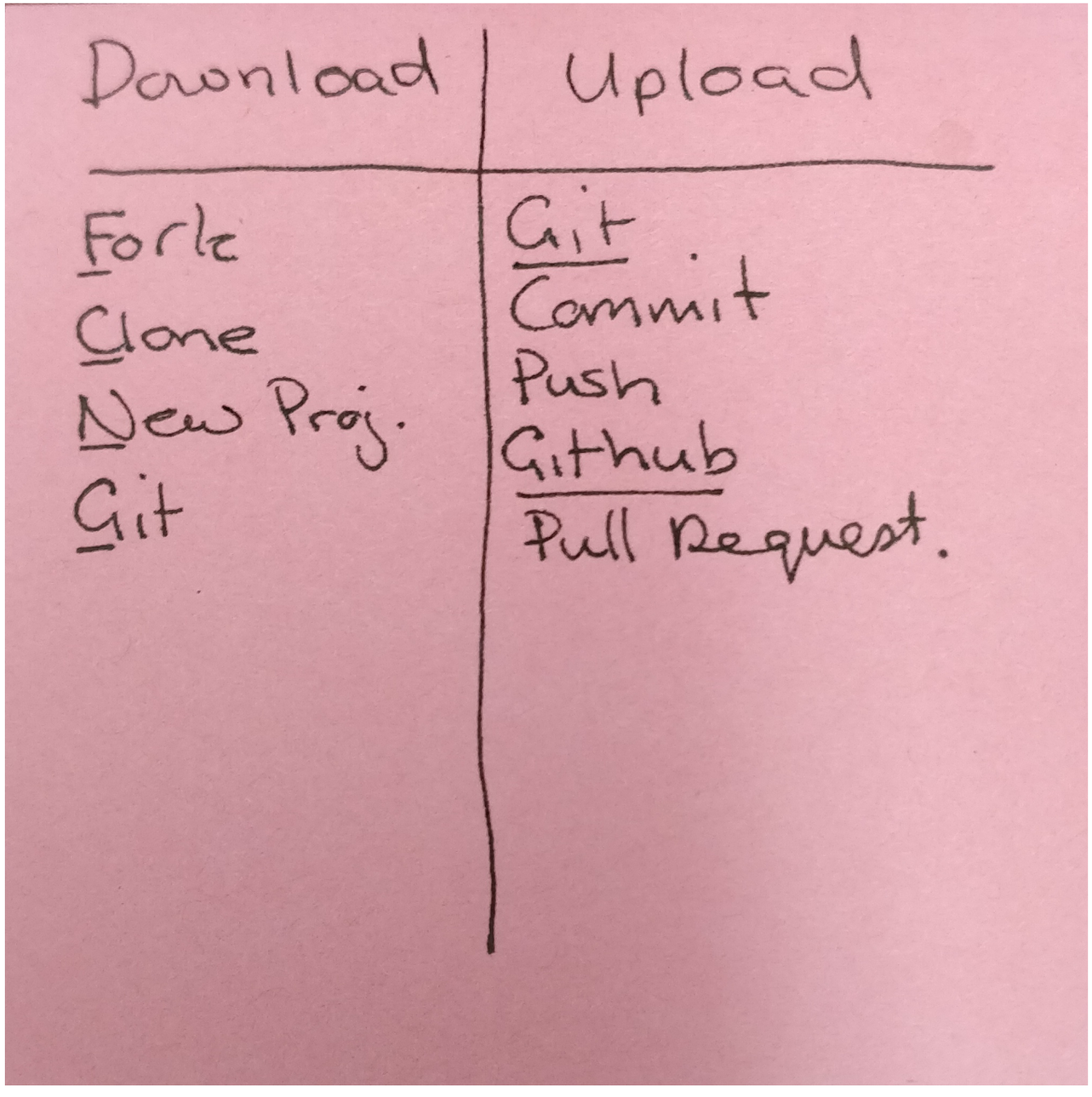
Pull Request

Fork

Github

Remote Repository

# Git/Github


Download
Upload
Fork
Clone
New Proj.
Git
Git
Commit
Push
Github
Pull Request.

❤️ Matching ❤️

# Matching

- Common problem
- Assigning medical students to hospitals
- Assigning organ donors to recipients
- Dating websites
- Assigning students to dorms

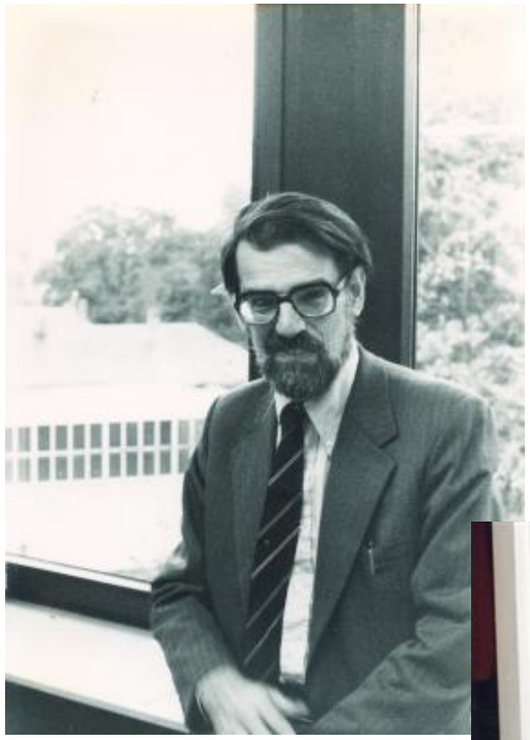
# Characteristics

- Preference measure
- Match two groups together
  - Both groups generate rating

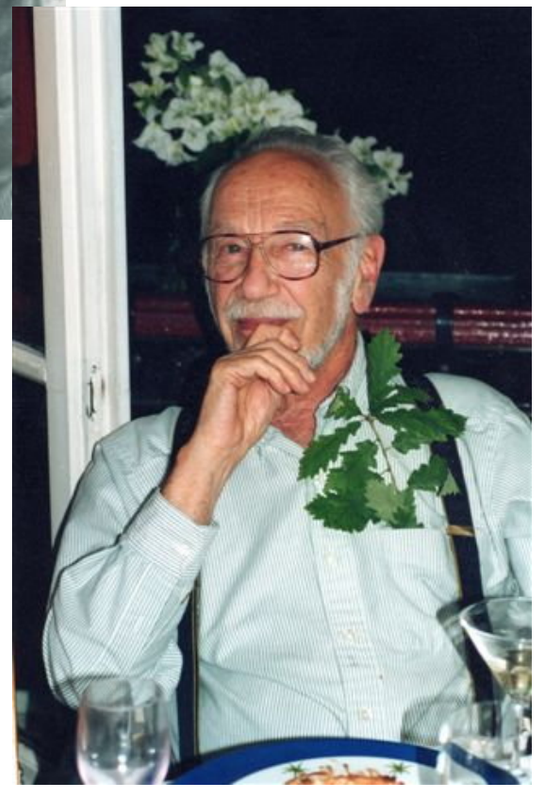
# Solution

- 2012 Nobel Prize in Economics

- Gale-Shapley Algorithm (1962)



Lloyd Stowell Shapley



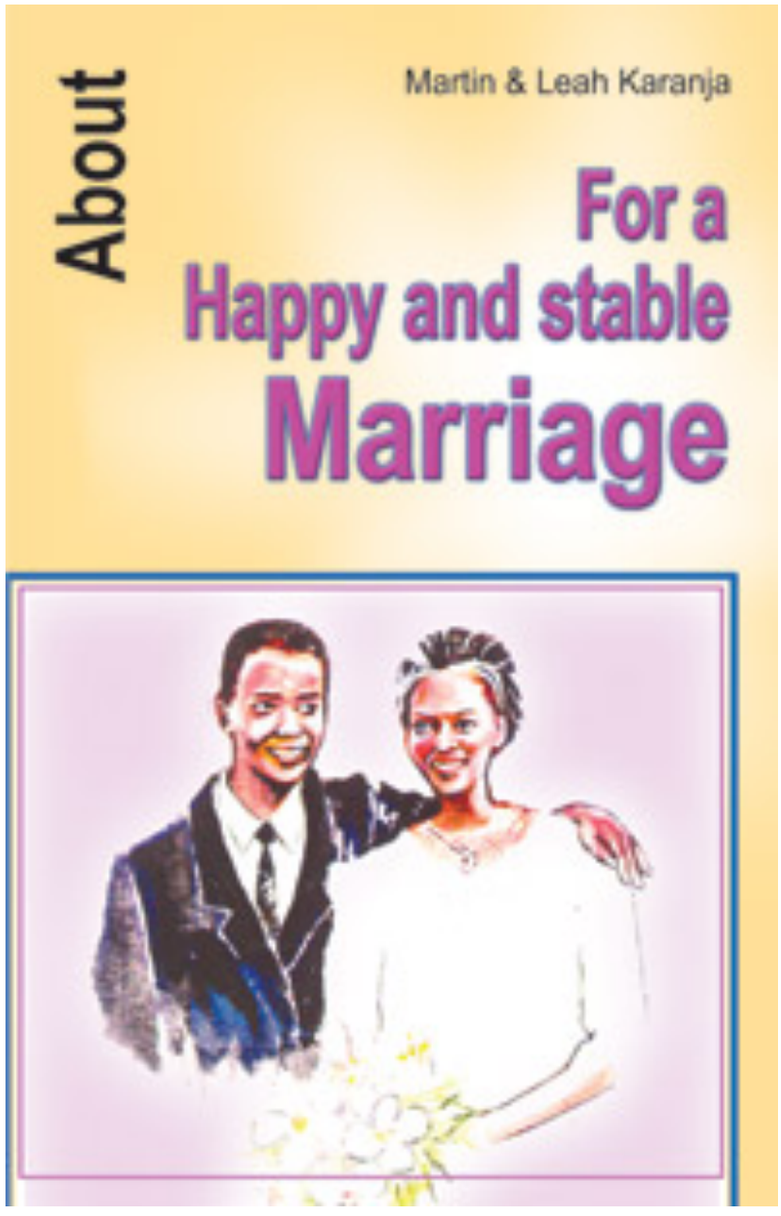
David Gale

# Stable Marriage Problem

Finding a stable matching between two equally sized sets of elements given an ordering of preferences for each element.

Mapping from the elements of one set to the elements of the other set.

Stable: No element of set *A* prefers a different match when *B* also prefers *A* over the element to which *B* is already matched



# Stability

*The Heirs* (왕관을 쓰려는 자, 그 무게를 견뎌라 – 상속자들)

**Kim Tan**

**Choi Young-do**

**Yoon Chae-young**

**Rachael Yoo**

**Cha Eun-sang**

**Lee Bo-na**

| | | | | | |
|---|---|---|---|---|---|
| Rachel | Cha | Lee | Kim | Kim | Yoon |
| Lee | Rachel | Rachel | Yoon | Yoon | Kim |
| Cha | Lee | Cha | Choi | Choi | Choi |

# Gale-Shapley Algorithm

- Solves for stability
- Two steps:
  - 1. A “proposes” to B and B accepts their preference to create provisional “engagements”
  - 2. Each rejected A proposes to their second preference B and B can “trade up” or not
  - Repeat until all matched

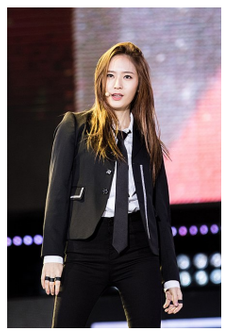
Kim Tan

Yoon Chan-young

Choi Young-do

Rachael Yoo

Cha Eun-sang



Lee Bo-na

| | | | | | |
|---|---|---|---|---|---|
| Rachel | Cha | Lee | Kim | Kim | Yoon |
| Lee | Rachel | Rachel | Yoon | Yoon | Kim |
| Cha | Lee | Cha | Choi | Choi | Choi |

# Gale-Shapley Algorithm

- Two libraries in R: matchingMarkets & matchingR
- Runs the algorithm
- Checks for stability

# What happens when you only have one set of preferences?

- How do we solve this?
  - Provide random preferences?
    - Randomize only those that double up?
    - Distance strategy?
- How do we judge fairness?

# Adaptive Systems

NETFLIX

amazon.com

PANDORA

last.fm

hulu

LinkedIn

# Recommender Systems

Collaborative filter: build a model from a user's past behavior + similar decisions made by other users

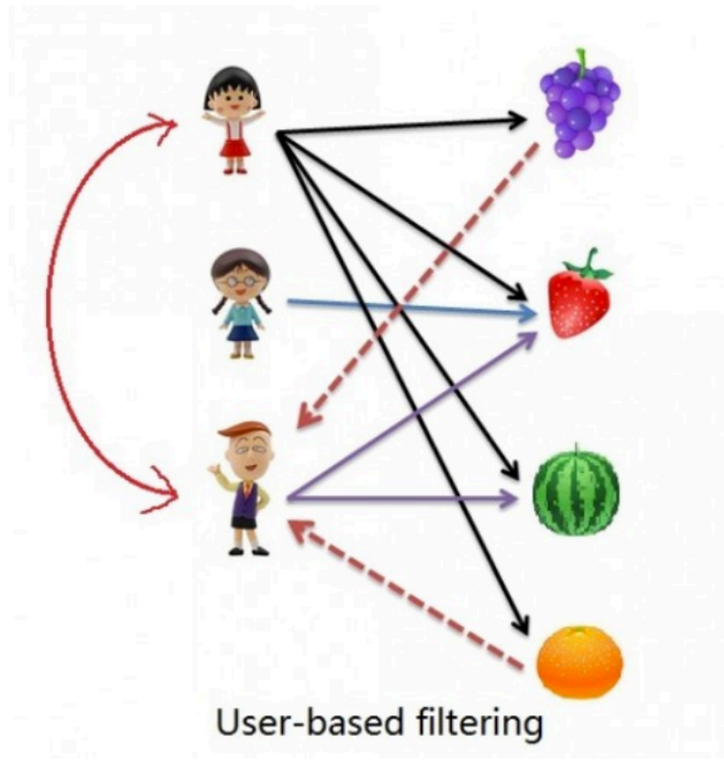

Content filter: utilize a series of discrete characteristics of an item in order to recommend additional items with similar properties

# Cold Start Problem

The system cannot draw any inferences for users or items about which it has not yet gathered sufficient information.

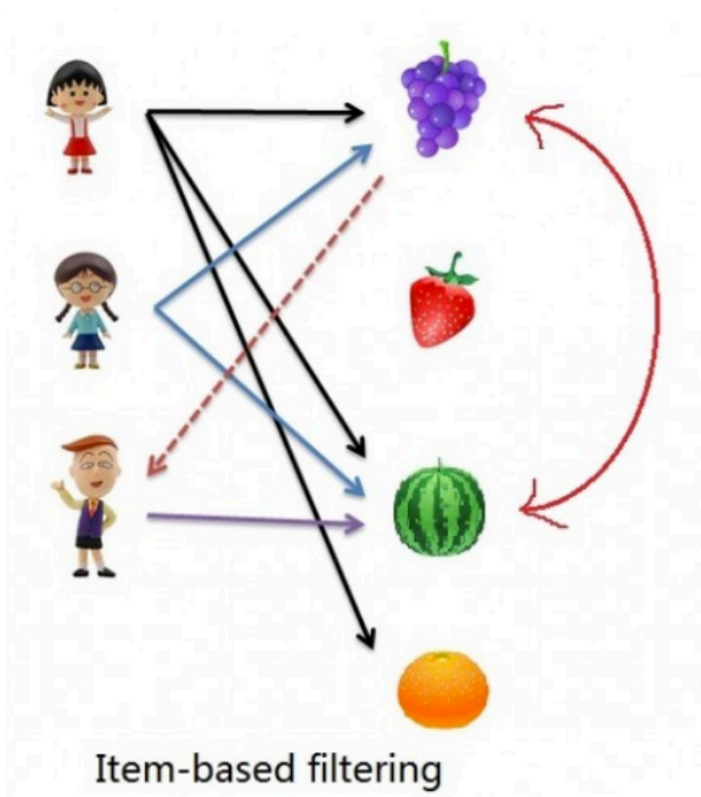
# User Based Collaborative Filter

| | **student A** | **student B** | **student C** |
|---|---|---|---|
| **podcast** | score improved = yes | yes | no |
| **game** | yes | no | no |
| **quiz** | yes | yes | no |



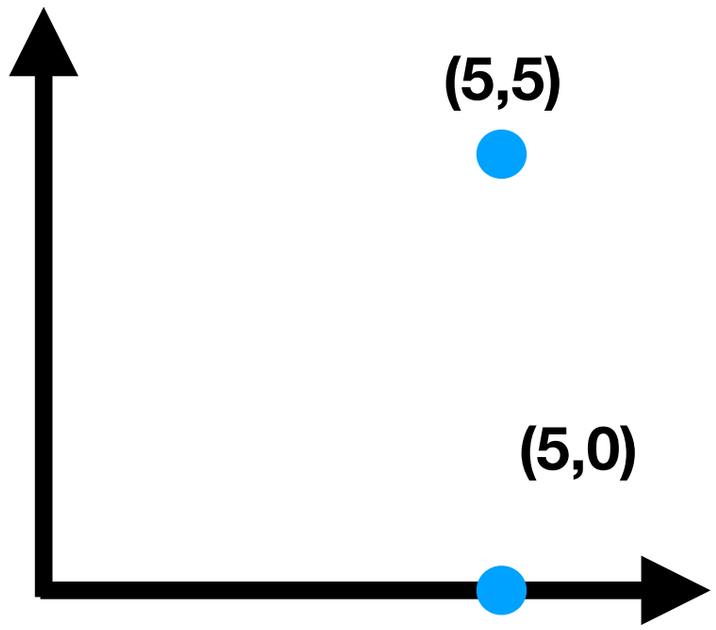
# Item Based Collaborative Filter

| | **student A** | **student B** | **student C** |
|---|---|---|---|
| **podcast** | score improved = yes | yes | no |
| **game** | yes | no | no |
| **quiz** | yes | yes | no |



# Similarity

- Many different ways to calculate
- Euclidean Distance (dissimilarity)

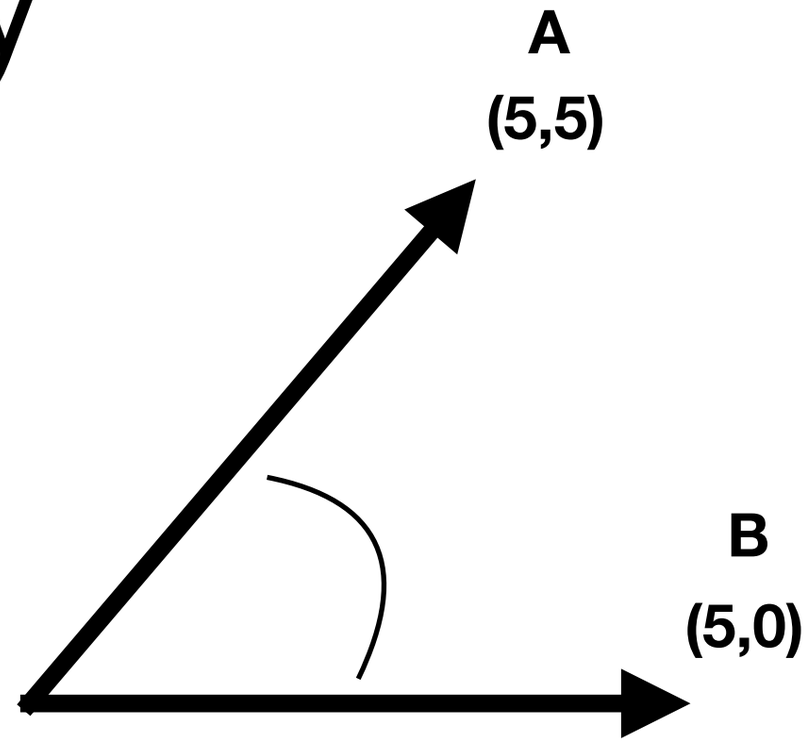



$$d_{L2}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

```
sqrt((5+5)^2 + (5+0)^2)
```

# Similarity

- Cosine similarity:
  - Calculate the angle between two vectors
  - Same direction = 1
  - Opposite direction = -1




**A = 5 5**
**B = 5 0**

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

$$\frac{\text{5*1 + 5*0}}{\text{sqrt(5*5 + 1*1) * sqrt(5*5 + 0*0)}}$$

# Item Based Collaborative Filter

| | **student A** | **student B** | **student C** |
|---|---|---|---|
| **podcast** | score improved = yes | yes | no |
| **game** | yes | no | no |
| **quiz** | yes | yes | yes |

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

**A = c(1,1,1)**

**B = c(1,0,1)**

$$\mathbf{sim_{AB}} = \frac{\mathbf{(1 \times 1 + 1 \times 0 + 1 \times 1)}}{\mathbf{sqrt((1x1 + 1x1 + 1x1)) \times sqrt((1x1 + 0x0 + 1x1))}}$$

**$sim_{AB}$ = 0.816**

# Similarity Matrix

| | **student A** | **student B** | **student C** |
|---|---|---|---|
| **student A** | 1 | 0.82 | 0.58 |
| **student B** | 0.82 | 1 | 0.71 |
| **student C** | 0.58 | 0.71 | 1 |

| | **podcast** | **game** | **quiz** |
|---|---|---|---|
| **podcast** | 1 | 0.71 | 0.82 |
| **game** | 0.71 | 1 | 0.58 |
| **quiz** | 0.82 | 0.58 | 1 |

# Which to use?

- Depends what you are trying to do?
- There are usually more users than items, therefore more variation
  - Scaling issues (bigger matrix)
  - Items more likely to converge (once converged don’t have to calculate)
- Often an extra step in user-based
  - Find neighborhood of similar individuals
  - Then recommend