MMA/MMAI 869
Machine Learning and AI

Recommender Systems

Stephen Thomas

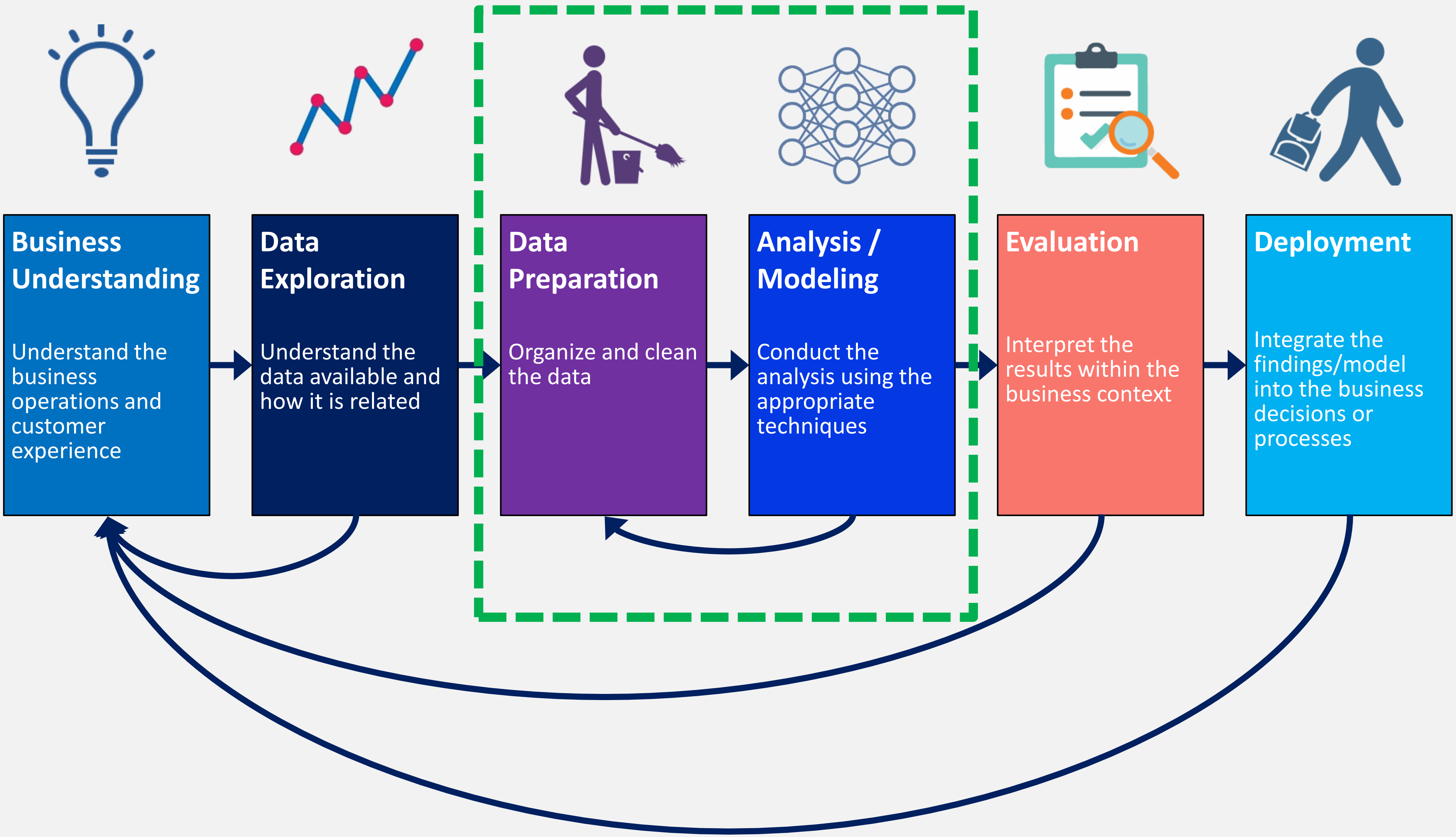
Updated: November 30, 2022

Smith School of Business
Queen's University

Outline

- What is a recommender system?
- What are the common algorithms?
- How to evaluate?
- Practical issues?
- Case studies

The Analytics Process: CRISP-DM



More Detail

Data Preparation

Organize and clean the data

Feature Engineering

- Normalization
- Discretization
- Coding
- Temporal, text, image

Feature Selection

- Filter
- Wrapper

Analysis / Modeling

Conduct the analysis using the appropriate techniques

Model Training

- Algorithms
- Ensembles
- Parameter tuning

Model Selection/ Evaluation

- Cross validation
- A/B Testing

OVERVIEW

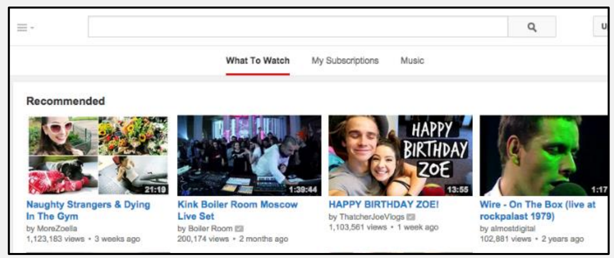
Overview

Recommender System

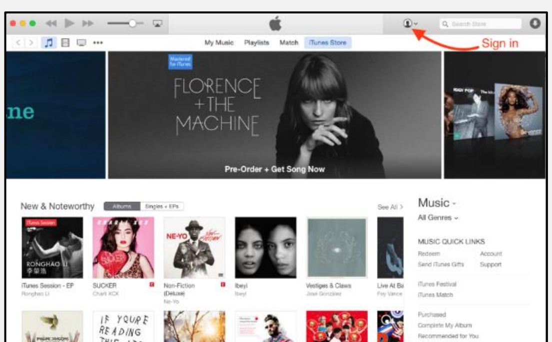
noun

- A system that selects a small set of ***items*** for each ***user***.

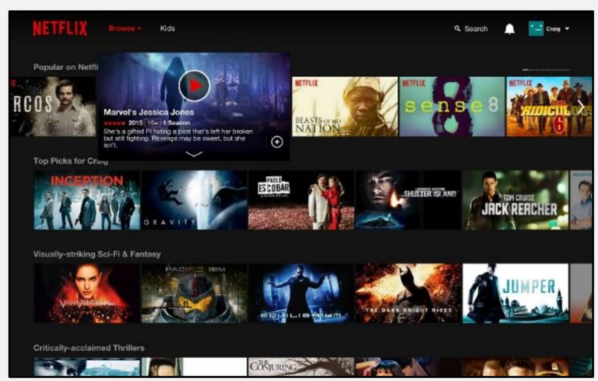
Amazon



YouTube



iTunes



Netflix

- Google News
- Google Search
- Google Play
- IMDb
- TripAdvisor
- YouTube
- Pandora
- Facebook
- Jester
- StumpleUpon
- LinkedIn
- Delicious
- …

Why Recommender Systems?

Too many options

Search and browsing not good enough

- Netflix: 15 thousand shows
- iTunes: 60 million songs
- Amazon: 480 million products
- YouTube: 14 billion videos

Recommender Systems

Discover new items | Personalization | Increase loyalty | Increase profits | Reduce churn

94% of top ecommerce companies say that RS are "important to business success"

The Long Tail

Business Value

amazon **$35B/year (40%)** come from recommendations

YouTube **60%** watch time from recommendations

Google Play **40%** app installs from recommendations

NETFLIX **75%** watch time from recommendations

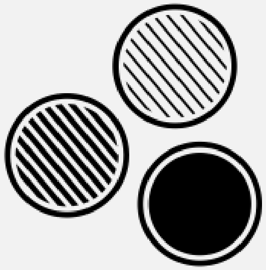
Goals of a RS

- Increase product sales
- Increase click through rates
- Increase conversions
- Increase user engagement
- Improve customer experience:

Relevant
User needs/wants it

Novel
User has not seen item in the past

Serendipitous
Unexpected; lucky



Diverse
Increase chance that user might like at least one

ASIDE: GENERIC APPROACH

Generic Approaches

Most Popular

- Show the most popular items

Staff Picks

- Humans manually curate a list of their favorite items

Newly Added

- Show new items

Trending

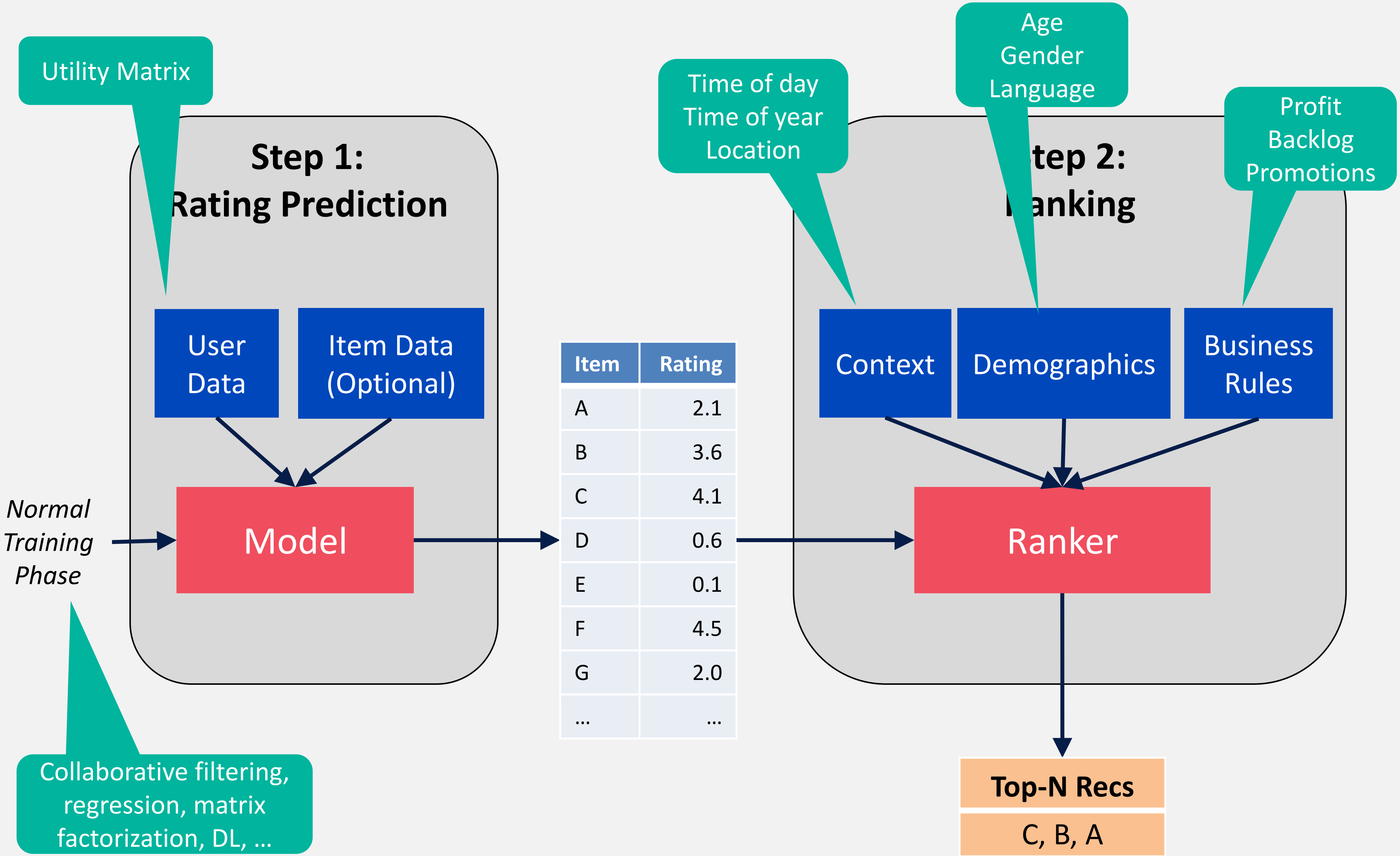
- Show items that have increased the most in popularity in some time period

Seasonal

- Show items that are most popular in the current time of year

RECOMMENDER SYSTEM ARCHITECTURE

Recommender System Architecture



Item	Rating
A	2.1
B	3.6
C	4.1
D	0.6
E	0.1
F	4.5
G	2.0
…	…

Existing Tools and Packages

- **R libraries**
  - recommenderlab
  - recosystem
  - slopeOne
  - SVDApproximation
  - reco
- **Python packages**
  - surprise
  - lightfm
  - logistic-mf
  - python-recsys
  - crab
  - tensorrec
- **Open Source Tools**
  - SuggestGrid
  - Apache Mahout
  - Apache Spark
  - Apache PredictionIO
  - Amazon's DSSTNE
- **Cloud**
  - Google Recommendation AI
  - Amazon Personalize
  - Microsoft ML

Utility Matrix

Smith
SCHOOL OF BUSINESS
Queen's University

Conceptual View

Items

(e.g., movies, stores, products, …)

Users

4	3			5	
5		4		4	
4		5	3	4	
	3				5
	4				4
		2	4		5

Ratings

DataFrame View

UserId	ItemId	Rating	Timestamp
1	201	4	2021-01-01
1	202	3	2021-01-01
1	205	5	2021-01-02
2	201	5	2021-01-02
2	203	4	2021-01-02
2	205	4	2021-01-01
3	201	4	2021-01-01
3	203	5	2021-01-03
3	204	3	2021-01-03
3	205	4	2021-01-03
4	202	3	2021-01-01
4	206	5	2021-01-03
5	202	4	2021-01-03
5	206	4	2021-01-03
6	203	2	2021-01-04
6	204	4	2021-01-01
6	206	5	2021-01-02

What Goes in the Utility Matrix?

- ***Explicit Ratings:*** Users intentionally give rating
  – Rating a movie 1-5
  – Liking an Insta post
  – Disliking a song on Google Play Music
- ***Interval***: Values come from a discrete set of ordered numbers
  – E.g., {1, 2, 3, 4, 5}
  – E.g., {-2, -1, 0, 1, 2}
- ***Binary***: user specifies like or dislike
  – E.g., {0, 1}
- ***Unary ratings***: user specifies like, but no way to specify dislike

What Goes in the Utility Matrix?

- ***Implicit Ratings***: Ratings are inferred from user actions
  - Customer buys an item → customer likes item
  - Customer watches video → customer likes video

UserId	ItemId	Action	Timestamp
1	201	click	2021-01-01
1	202	watch	2021-01-01
1	205	watch	2021-01-02
2	201	click	2021-01-02
2	203	long_watch	2021-01-02
2	205	click	2021-01-01
3	201	subscribe	2021-01-01
3	203	watch	2021-01-03
3	204	watch_later	2021-01-03
3	205	watch	2021-01-03
4	202	long_watch	2021-01-01
4	206	long_watch	2021-01-03
5	202	click	2021-01-03
5	206	rent	2021-01-03
6	203	long_watch	2021-01-04
6	204	watch_later	2021-01-01
6	206	watch	2021-01-02

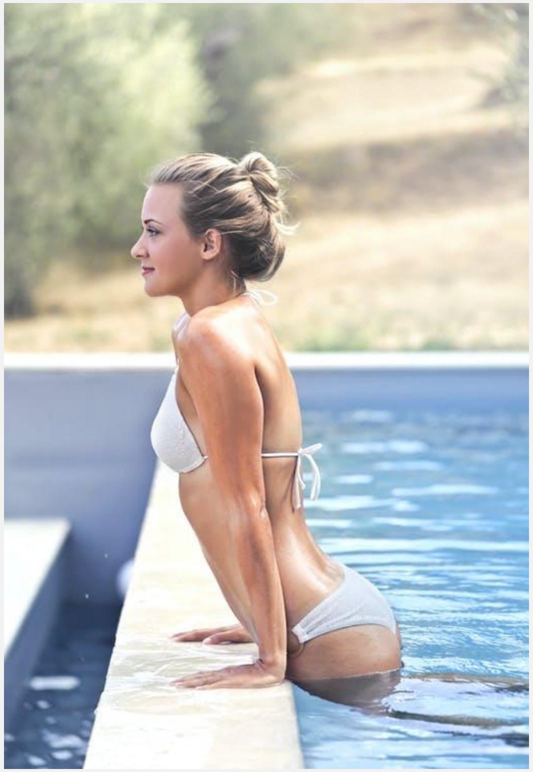
Action	Value
click	1
watch_later	2
watch	3
long_watch	4
rent	5
subscribe	6

Weighted Count

UserId	ItemId	Implicit
1	201	1
1	202	3
1	205	3
2	201	1
2	203	4
2	205	1
3	201	6
3	203	3
3	204	2
3	205	3
4	202	4
4	206	4
5	202	1
5	206	5
6	203	4
6	204	2
6	206	3

Is Click Data Good for Implicit Ratings?

- Careful!
- Click can be misleading
  - Human behaviour
  - Easy to game/bot by malicious users
- Better to let people “vote with their wallet” or with their time

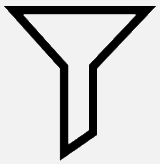


Comparison

	Pros	Cons
Explicit		
Implicit		

STEP 1: RATING PREDICTION

Rating Prediction Algorithms - Intuition

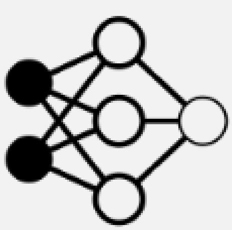


Collaborative Filtering

If item A and item B are ***liked by the same people***, then they might have something in common

Content-based

If item A and item B have ***similar characteristics***, then they might have something in common



Deep Learning

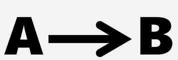
.... ***pure magic***.

COLLABORATIVE FILTERING

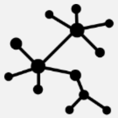
Collaborative Filtering

- ***Collaborative filtering*** techniques use *utility matrix* to predict ratings of un-rated items
  - ***User-based***: If person A and person B share an opinion on one item, then they might share an opinion on a different item
  - ***Item-based***: If item A and item B are liked by the same people, then they might have something in common
- CF uses utility matrix to build a ML model
  - Regression, especially KNN
  - Association rules
  - Graph analytics
  - Matrix decomposition/Factor Analysis

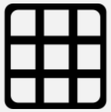
Regression



Association Rules



Graph



Matrix

Item Characteristics NOT Used!

Quick Exercise

- Before I tell you how to do it, see if you can answer this.
- Would Mr. Purple like Beyoncé or Smashing Pumpkins more?

	ADELE	RED HOT CHILI PEPPERS	BEYONCÉ	The Smashing Pumpkins
	5	2	5	1
	4	1	3	1
	2	5	1	4
	1	5	?	?
	1	4	1	5

1 = lowest, 5 = highest

Example of CF with KNN (User-Based)

3	3		4	5	0	0.93
5	5	4	$\boxed{3}$	4	1	
4		5	3	4		0.98
	3		1		5	0.67
0	4		2	4	4	0.70
1	0	2	4	5	5	0.56

- What would Blue rate LOTR?
- Find K=2 users whose ratings are most similar to Blue's
  - Use similarity metrics
- Result: Red, Green
- Estimate Blue's rating of LOTR based on Mr. Red's and Mr. Green’s rating
  - Use weighted average

$\text{sim}_{\cos}$ (Mr. Blue, Mr. Green) = 0.93

…

r(Mr. Blue, LOTR) = ((.93)* 4 + (0.98) * 3) / (.93+.98) = 3.48 ~= 3

Example of CF with KNN (Item-Based)

3	3		4	5	0
5	5	4	4	4	1
4		5	3	4	
	3		1		5
0	4		2	4	4
1	0	2	4	5	5

0.81 0.64 0.85 0.98 0.66

- What would Blue rate LOTR?
- Find two items whose ratings are most similar to LOTR's
  - Compute similarity metrics
- Result: SPR, Matrix
- Estimate Mr. Blue's rating of LOTR based on Mr. Blue's rating of SPR and Matrix
  - Use weighted average

r(Mr. Blue, LOTR) = ((.85)* 4 + (.98) * 4) / (.85+.98) = 4

Pros and Cons

- Pros of Item-based:
  - Usually gives more relevant predictions
    - Because it uses the user's own ratings
  - Easier to explain
    - *"Because you liked SPR, you might like LOTR."*
  - Scales better
    - Usually, fewer items than users
  - Items don’t change; people do
- Cons of Item-based
  - Tend to give "obvious" recommendations
    - Not novel or serendipitous

Exercise

- What will Steve rate item *e*?
  - Use user-based KNN (K=1) with Euclidean Distance

$$dist_{euc}(A, B) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2}$$

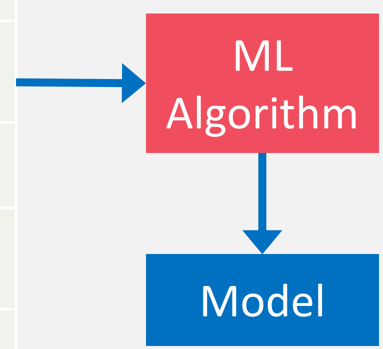
	a	b	c	d	e
Steve	5	2	1	3	**?**
Peyton	4	1	1	5	5
Tom	1	5	1	1	2

STEP 1:
CONTENT BASED

Content-Based

- ***Content-based*** techniques use utility matrix ***and item profiles*** to predict ratings of un-rated items
  - Create ***item profiles***
  - For each user:
    - Label the item profiles with user ratings
    - Use ML to build model (regression, KNN, NN, …)
    - Use model to predict ratings of un-rated items

Genre	Year	Director	Stars	Length	…	Mr. Blue's Rating
Adventure	2000	Ridley Scott	Russell Crow, Joaquin Phoenix	171m		3.5
Comedy	2006	David Frankel	Anne Hathaway, Meryl Streep, Adrian Grenier	110m		5.0
Sci-Fi	1999	Lana Wachowski	Keanu Reeves, Laurence Fishburne	150m		
Drama	1998	Steven Spielberg	Tom Hanks, Matt Damon	170m		
Adventure	2001	Peter Jackson	Elijah Wood, Ian McKellen, Orlando Bloom	228m		1.0
Comedy	2004	Mark Waters	Lindsay Lohan, Jonathan Bennett, Rachel McAdams	97m		



Mr. Blue


MAKE A SEPARATE MODEL
FOR ALL THE USERS
imgflip.com

Content-Based: Creating Item Profiles

- Item profiles depend on the application domain
- Movies
  - Genre, actors, directors, …
- News articles
  - ???
- Televisions
  - ???
- Friends
  - ???

Uncle Steve's Comparison Guide

Method	Summary	Algorithms	Pros	Cons
Collaborative Filtering	*Give Bob recommendations based on the ratings and actions of Bob's peers*	• Regression • Associating rules • Graph analytics • Matrix decomposition • …	✓ Works for any item ✓ No item profiles ✓ Easy to implement	× Need lots of data × Computationally expensive × Performs badly with sparse data
Content-based	*Give Bob recommendations based on the content that Bob have favored in the past*	• Regression	✓ More personalized ✓ Faster ✓ No new item problem ✓ Higher accuracy	× Must create profiles × Overspecialization

STEP 2: RANKING

(Re-)Ranking the Predictions

- Given predicted ratings, can use context to rank
- Example: Steve's 6-year-old logs into Netflix on December 9
  - Demographics (age, gender, language, location)
  - Time of day/year
  - Business rules

Model

Ranker

Item	Predicted Rating	Age Score	Lang Score	Season Score	Steve Score	Final Score
The Grinch	3.9	1.5	1.0	1.5	1.0	8.8
Home Alone	4.0	1.5	1.0	1.5	1.0	9.0
Frozen	4.1	1.5	1.0	1.0	0.4	2.5
Epic Movie	1.2	0.5	1.0	1.0	1.0	0.6
Bambi 2	2.0	1.5	1.0	1.0	1.0	3.0
Zipi Y Zape	4.5	1.5	0.4	1.0	1.0	2.7
Elf	3.5	0.5	1.0	1.5	1.0	2.6

x x x x =

Top-3 Recs

Home Alone, The Grinch, Bambi 2

EVALUATION

Evaluation of RS

- How to determine the quality of recommendations?
- **Offline testing**
  - Cross validation (as per usual)
  - Evaluate ratings (MSE, RMSE)
  - Evaluate rankings (precision, recall, hit-rate)
  - Evaluate diversity (coverage, novelty)
- **Online testing**
  - A/B testing

Evaluate Predicted Ratings

- Compare predicted ratings with truth

Truth Rating	Predicted Rating	Error	Abs Error
5	3	2	2
4	1	3	3
4	5	-1	1
1	1	0	0

RMSE = $(2^2+3^2+-1^2+0^2)/4 = 3.5$

MAE = $(2+3+1+0)/4 = 1.5$

Evaluate Rankings

- Compare recommendations with "liked" items in truth
  - ***Precision @ K:*** percentage of recommendations that are correct
  - ***Recall @ K:*** percentage of liked items that are recommended
  - ***Hit:*** Is at least one correct?

	User 1	User 2	User 3	
Truth	A, B, C, D	A, C, E	E	
Top-3 recs — **Recs**	A, D, E	A, C, E	A, B, C	
				Mean
Precision@3:	2/3 = 0.67	3/3 = 1.00	0/3 = 0.00	**0.56**
Recall@3:	2/4 = 0.50	3/3 = 1.00	0/1 = 0.00	**0.50**
Hit:	1	1	0	**0.67**

Evaluate Diversity

- Evaluate overall diversity of recommendations
  - ***Novelty*** – Mean popularity rank of recommended items
  - ***Diversity*** – Similarity of recommended items
  - ***Coverage*** – Number of items that are recommended

Recommender A

User 1	User 2	User 3
Recs	**Recs**	**Recs**
The Godfather The Dark Knight The Matrix	Lord of the Rings The Departed Memento	Avatar Casino Royal The Godfather
Recs	**Recs**	**Recs**
Dark Knight Rises The Dark Knight Batman Begins	Harry Potter Harry Potter 2 Harry Potter 3	Toy Story Toy Story 2 Toy Story 3
Recs	**Recs**	**Recs**
Coco The Dark Knight Adventureland	Coco The Dark Knight Adventureland	Coco Harry Potter Adventureland

Recommender B

User 1	User 2	User 3
Recs	**Recs**	**Recs**
The Godfather Adventureland Donnie Darko	Amelie American Psycho Memento	Spirited Away Mitchells vs Machines Elf

PRACTICAL ISSUES

Mean Centering

- Potential problem with utility matrix
  - Some users usually rate everything high
  - Other users usually rate everything low
- Solution: ***mean centering***
  - Subtract mean for each user

5	3	5	4
1	2	1	1
1		5	
5	5	5	5

Original Utility Matrix

4.25
1.25
3
5

Mean

0.75	-1.25	0.75	-0.25
-0.25	0.75	-0.25	-0.25
-2		2	
0	0	0	0

Mean Centered Utility Matrix

Cold Start Problems

- **New user problem:** no ratings yet for new users
  - Possible solutions:
    - Use generic (top 10, new, etc.) ...
    - Ask user to rate a few key products
- **New item problem**: no ratings yet for new items
  - Possible solutions:
    - Just don’t worry about it
    - Use Content-based attributes

Stop Lists

- Sometimes good idea to never recommend certain items
  - Offensive, illegal, PR disaster, etc.
- Examples:
  - Adult content
  - Vulgarity
  - Illegal
  - Medical/Drugs
  - Religion

Filter Bubbles

- Recommender systems will, by default, continue to recommend things like the things users already liked
- Is this good or bad? Why?

“Fake” Users

- Malicious bots
- Web-crawlers
- Industrial buyers
- Professional reviewers

Fraud and Attacks

- Sellers have huge incentive to manipulate recs
  - Increase recs of own products
  - Decrease recs of competitors products
- Types of attacks:
  - ***push attack:*** submit high ratings for "my" items
  - ***nuke attack:*** submit low ratings for "opponent's" items
- CF is susceptible to such attacks – Why?
- Possible solutions (nuclear arms race):
  - Verified purchases
  - Captcha
  - Limited number of accounts for single IP address
  - Detect *suspect users*

Shared Accounts

- Users often share an account with loved ones, friends, etc.
- Is this a problem? Why?

Use Ensemble Methods

Implement two or more separate recommenders and combine predictions using, e.g., Voting

Recommender 1	
Item1	0.5
Item3	0.3
Item4	0.1
Item2	0
Item5	0

Recommender 2	
Item2	0.8
Item1	0.9
Item3	0.4
Item4	0
Item5	0

Combined	
Item1	0.65
Item2	0.45
Item3	0.35
Item4	0.05
Item5	0.00

RESOURCES

Resources

recSys.Rmd

slides_recsys.ipynb

Introduction

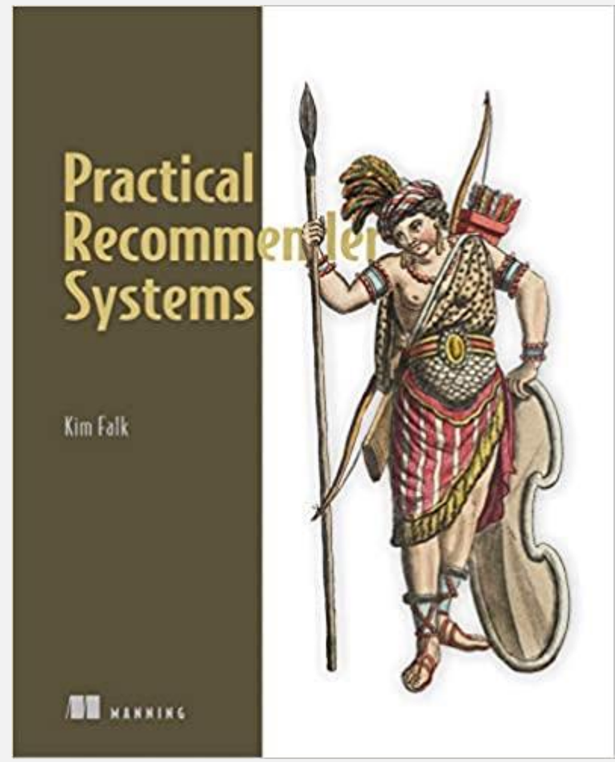
This repository contains examples and best practices for building recommendation systems, provided as Jupyter notebooks. The examples detail our learnings on five key tasks:

- Prepare Data: Preparing and loading data for each recommender algorithm
- Model: Building models using various classical and deep learning recommender algorithms such as Alternating Least Squares (ALS) or eXtreme Deep Factorization Machines (xDeepFM).
- Evaluate: Evaluating algorithms with offline metrics
- Model Select and Optimize: Tuning and optimizing hyperparameters for recommender models
- Operationalize: Operationalizing models in a production environment on Azure

Several utilities are provided in recommenders to support common tasks such as loading datasets in the format expected by different algorithms, evaluating model outputs, and splitting training/test data. Implementations of several state-of-the-art algorithms are included for self-study and customization in your own applications. See the recommenders documentation.

For a more detailed overview of the repository, please see the documents on the wiki page.

Microsoft's Best Practices for Recommender Systems GitHub



SUMMARY

Summary

- ***Recommender systems*** recommend items to users
  - Keeps users happy, engaged
  - Hugely important big data analytics system
- ***Prediction Algorithms:***
  - ***Collaborative Filtering***: use utility matrix of user/item ratings
  - ***Content-based***: use ratings and item profiles
- Other topics (not discussed today)
  - Collaborative filtering with AR, Graph, Latent Factor Analysis
  - Knowledge-based recommenders
  - Case studies (YouTube, SCENE)

APPENDIX

DISTANCE METRICS

Reminder: Distance Metrics

Goal: Measure how "far apart" two vectors are from each other

- Equivalently: how "close" they are

A = [0.3, 1.5, 7.6, 0.0, 9.3]
B = [1.3, 2.9, 0.8, 0.4, 5.1]
C = [4.5, 1.9, 8.0, 0.4, 8.7]
…

$$dist_{cos}(A,B) = 1 - \frac{\sum_{i=0}^{n} A_i B_i}{\sqrt{\sum_{i=0}^{n} A_i^2}\sqrt{\sum_{i=0}^{n} B_i^2}}$$

$$dist_{euc}(A,B) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2}$$

$$dist_{man}(A,B) = \sum_{i=1}^{n} |A_i - B_i|$$

Jaccard Distance

- A distance metric for ***sets***
- Size of their intersection divided by the size of their union

$$dist_{jac}(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}$$

A = {dog, cat, fish}

B = {dog, cat, horse, pig}

2 in intersection (dog, cat)

5 in union (dog, cat, fish, horse, pig)

$dist_{jac}$ (A,B) = 3/5

Exercise

Find the Jaccard distance between the following two sets:

A = {a, b, c, x, y, z}

B = {x, w, b, q, z, p}

Simple Matching Coefficient

- Jaccard distance is cool, but what about items that are not in either set?

	i1	i2	i3	i4	i5
Tom	1	1	0	1	0
Bob	0	1	1	1	0

$dist_{jac}$ (Tom,Bob) = 1-(2/4)=1/2

Not considered by Jaccard

- Alternative: Simple Matching Coefficient (SMC)
  - Unlike Jaccard, it counts both mutual presences and ***mutual absences***

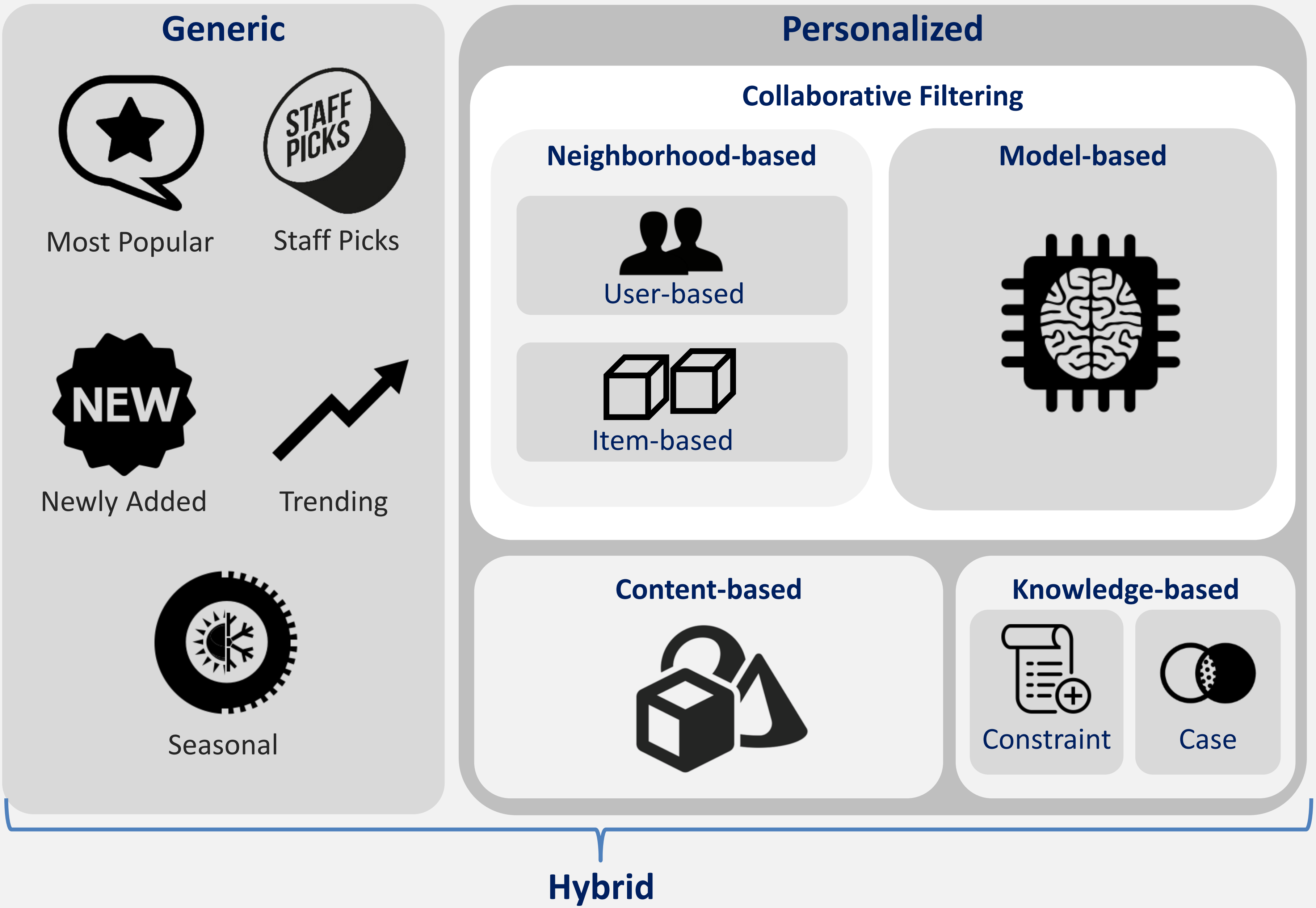
		A	
		0	**1**
B	**0**	M_{00}	M_{10}
	1	M_{01}	M_{11}

$$dist_{smc}(A,B) = 1 - \frac{M_{00} + M_{11}}{M_{00} + M_{10} + M_{01} + M_{11}}$$

		Tom	
		0	**1**
Bob	**0**	1	1
	1	1	2

$dist_{smc}$ (Tom, Bob) = 1-(1+2)/(1+1+1+2)=2/5

Algorithms for RS



Personalized Algorithms Overview

	Main Idea	Data Used
Collaborative Filtering	• Give me recommendations based on the ratings and actions of my peers	• User ratings • Community ratings
Content-based	• Give me recommendations based on the content that I have favored in my past ratings and actions	• User ratings • Item profiles
Knowledge-based	• Let me explicitly specify the kind of content I want	• User ratings • Item profiles • Domain knowledge

Knowledge-based RS

- When to use Knowledge-based RS?
  - Products with low number of ratings


  - Customer wants to define their requirements exactly
    - *"The color of the car should be black."*
- Two types
  - Constraint-based
    - User explicitly defines set of rules/constraints
    - Recommendations fulfill rules
  - Case-based
    - User explicitly defines sets of rules/constraints
    - Recommendations are "as close as possible" to rules

Knowledge-based RS

- Select items from this catalog that match the user's requirements

Item	price	mpix	opt-zoom	LCD-size	movies	sound	waterproof
1	148	8.0	4×	2.5	no	no	yes
2	182	8.0	5×	2.7	yes	yes	no
3	189	8.0	10×	2.5	yes	yes	no
4	196	10.0	12×	2.7	yes	no	yes
5	151	7.1	3×	3.0	yes	yes	no
6	199	9.0	3×	3.0	yes	yes	no
7	259	10.0	3×	3.0	yes	yes	no
8	278	9.1	10×	3.0	yes	yes	yes

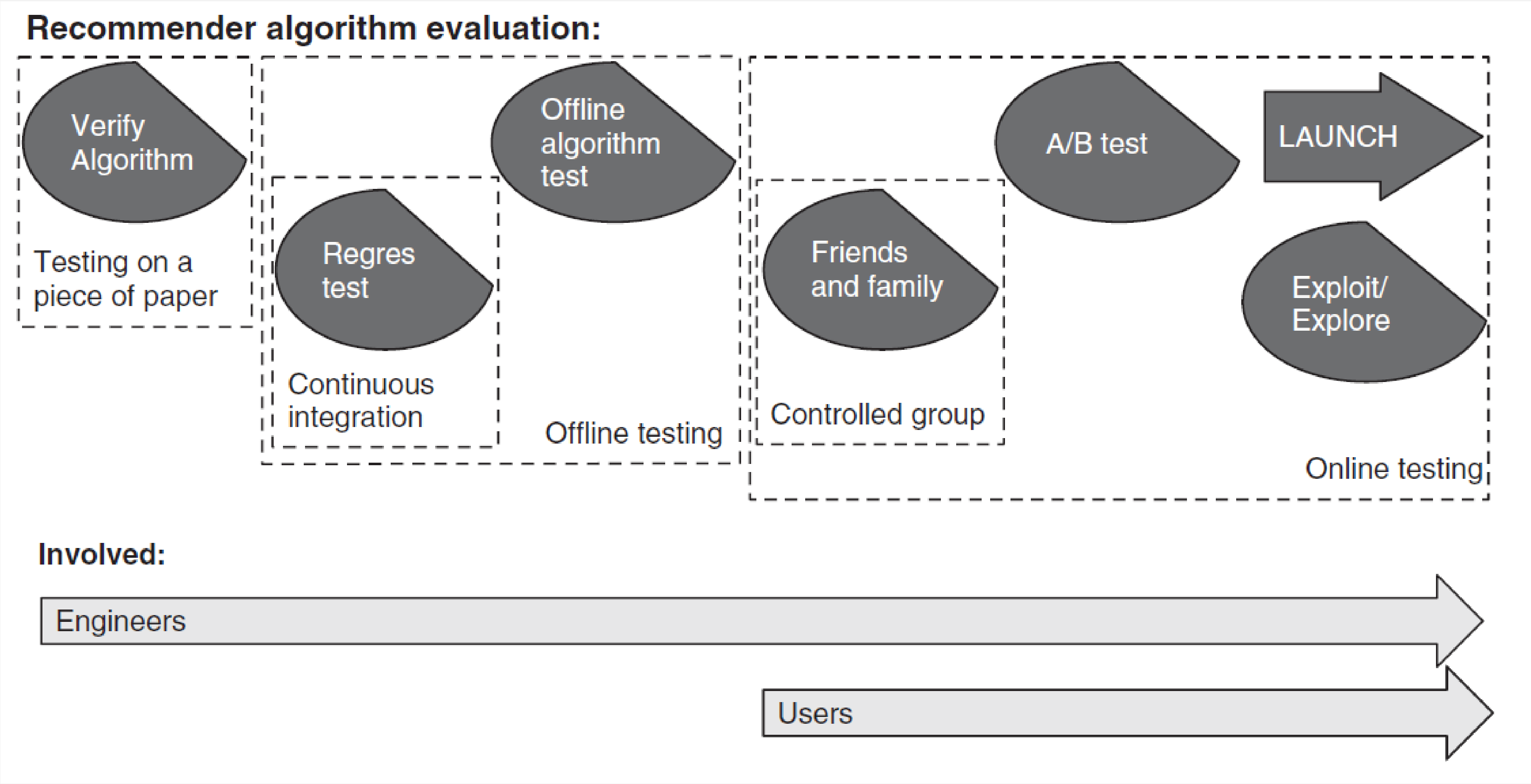
- User's requirements can, for example, be
  - *"The price should be lower than 300"*
  - *"The camera should be suited for sports photography"*

Exercise

- Which item is most similar to item "a"?
- Use SMC

	a	b	c	d
Steve	1	1	0	1
Peyton	0	1	1	1
Tom	0	0	0	1

Evaluation of RS



- Offline: similar to cross validation
  - Common metrics: MAE, RMSE, MSE

CF

Example of CF with Association Rules

- **Idea**: use association rules to predict items
- Association rules learning: {Diapers} -> {Beer}

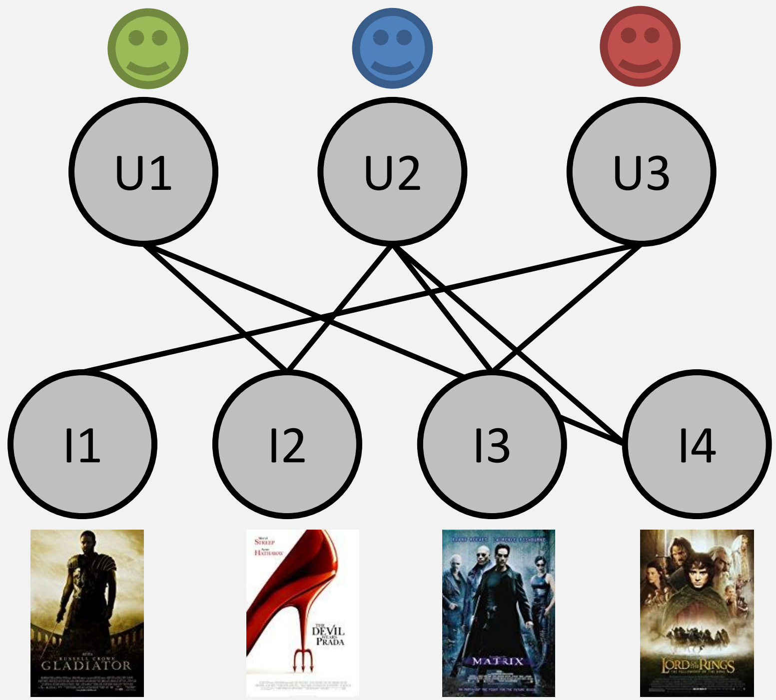
Example:

- Transform ratings to binary
- Find association rules, e.g.,
  - {Gladiator} → {Saving Private Ryan}
  - {The Matrix} → {Saving Private Ryan}
  - {Gladiator} → {The Matrix}
  - …
- Mr. Green has watched Gladiator, so predict The Matrix and Saving Private Ryan
  - Sort by confidence

1	0	0	0	?
1	0	1	0	1
1	0	1	0	1
0	0	0	1	1
0	1	1	0	0

Example of CF with Graph Analytics

- **Idea**: Build graph of users and items
- Traverse graph to find items to recommend
  - Common: Path length 3



- Example: what to predict for Mr. Green?
  - Predict The Matrix because Green→DWP→Blue→The Matrix

Exercise

- Estimate Steve's rating of item e
  - Use Graph Analytics with a path of 3

	a	b	c	d	e
Steve	5	2	1	3	?
Peyton	4	1	1	5	5
Tom	1	5	1	1	4

Example of CF with Latent Factor Analysis

- **Idea**: decompose utility matrix into sub matrices
  - Helps to reduce dimensionality
- E.g.: Singular Value Decomposition $M_k = U_k \times \Sigma_k \times V_k^{\,T}$

	Dim1	Dim2
	0.47	-0.30
	-0.44	0.23
	0.70	-0.06
	0.31	0.93

U_k

					
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

V_k^T

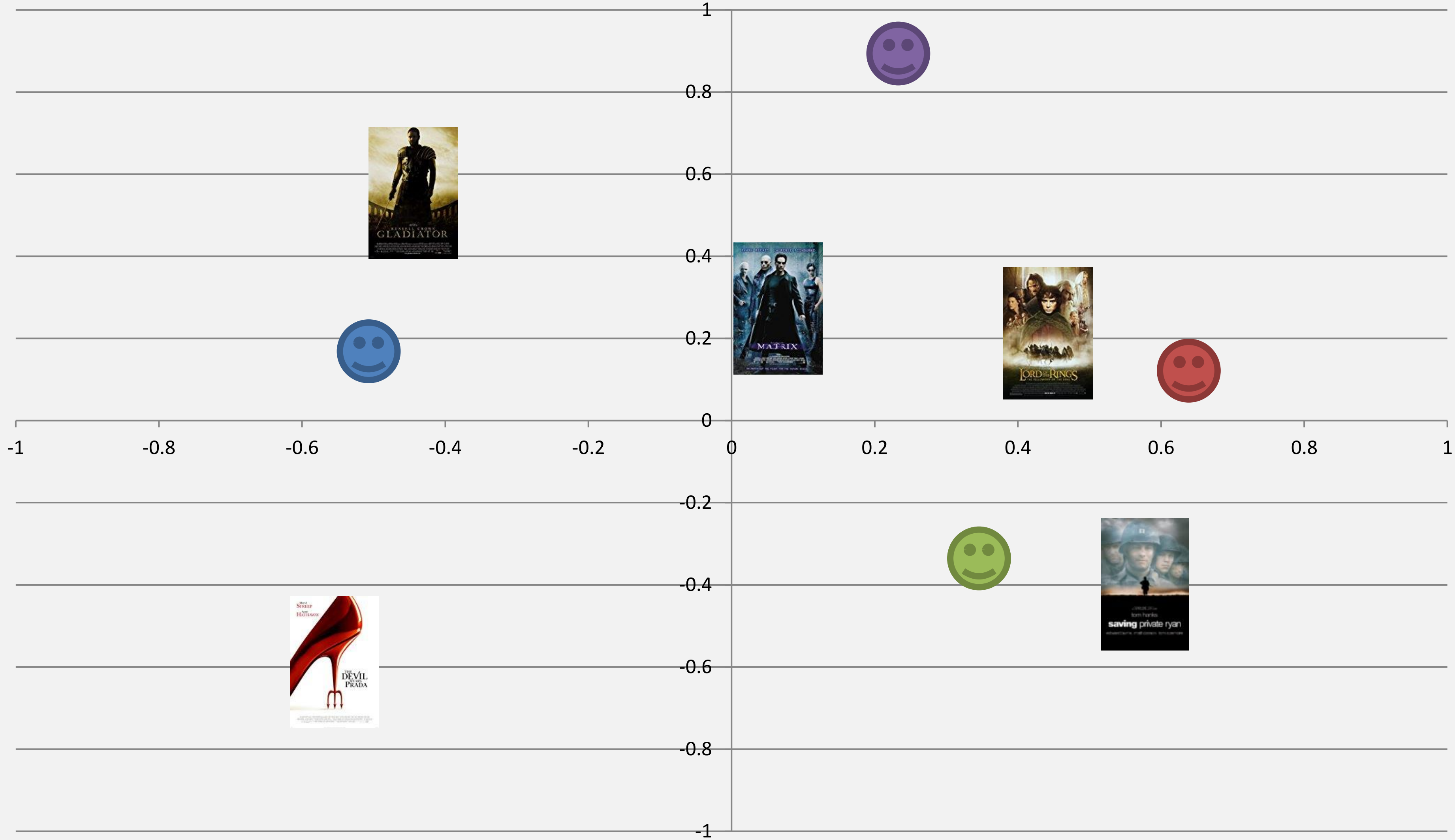
	Dim1	Dim2
Dim1	5.63	0
Dim2	0	3.23

Σ_k

- Prediction: $\hat{r}_{ui} = \bar{r}_u + U_k(Mr.Green) \times \Sigma_k \times V_k^T(LOTR)$

  $= 3 + 0.84 = 3.84$

Latent Factor Analysis



Projecting both users and items onto 2-D plot

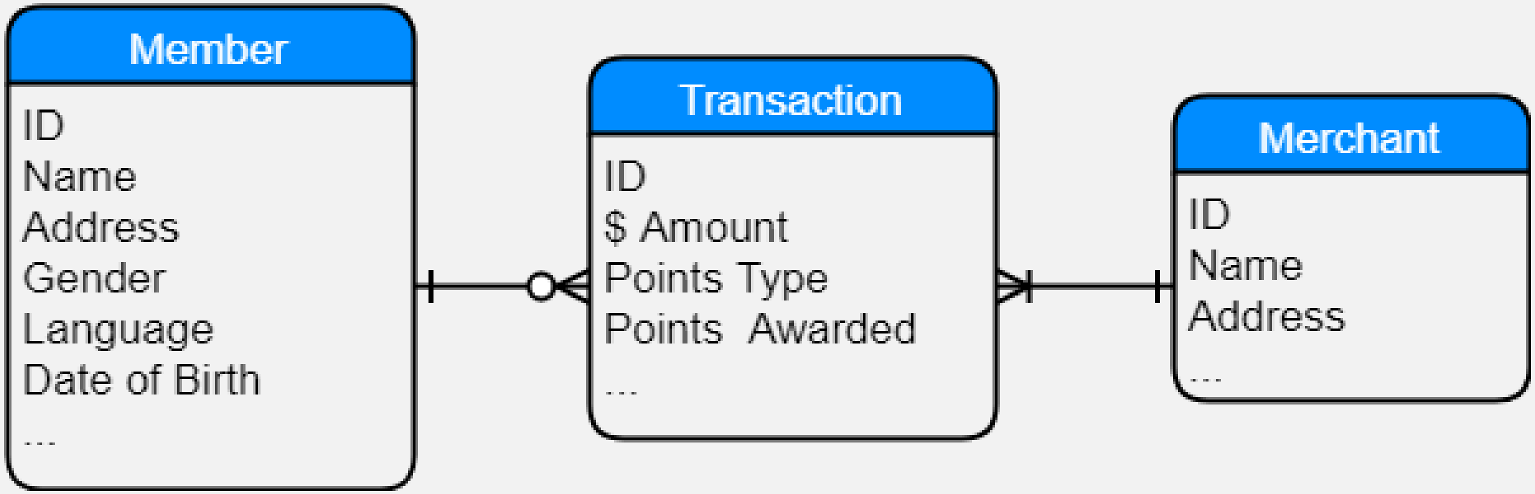
CASE STUDY: SCENE

SCENE

Smith
SCHOOL OF BUSINESS
Queen's University

- Rewards program in Canada
- SCENE wants to increase customer engagement

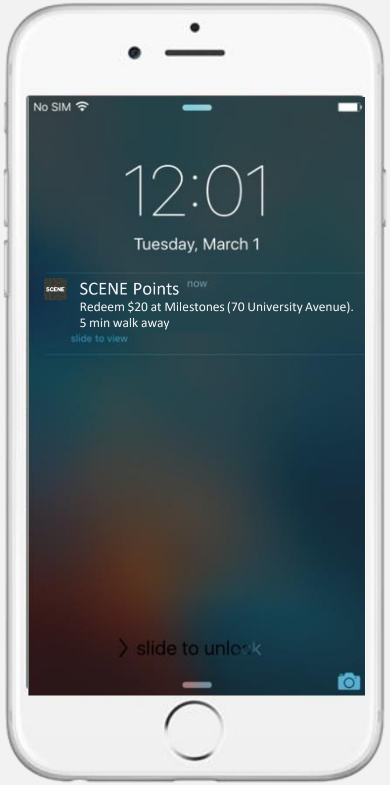
Dataset Outline (Simplified)



- 125,000,000+ transactions
- 1,800,000+ members
- 19 tables in all
- Data stored on the CAC

MMA Student Project

- **Idea**: Increase engagement by giving customers recommendations as to where to use their card next
- **Method:** Build a RS that pushes real-time recommendations to customers' mobile devices
  - Uses *model-based collaborative filtering*
  - Is *context-aware* (location, time)
- **Prototype**:



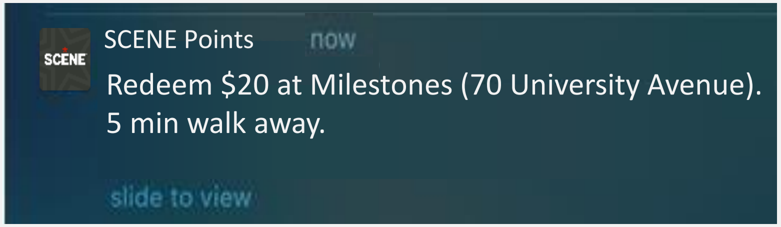


Image source: Team Botterell, MMA class of 2018S

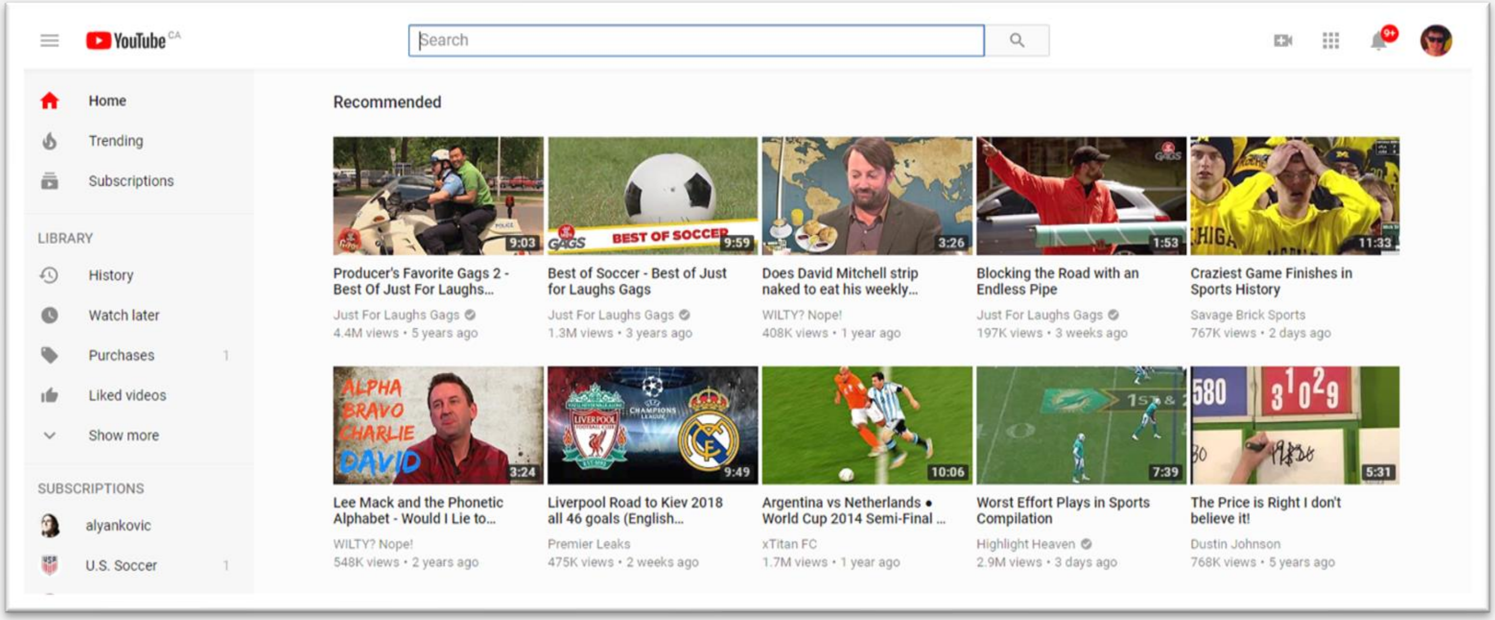
Benefits and Challenges

- Benefits
  - Increase customer engagement
    - Increase profits
    - Increase customer satisfaction
    - Decrease customer churn
    - Increase partner satisfaction
- Challenges
  - Shared accounts
  - Sparsity of utility matrix
    - Most users haven't rated most items

CASE STUDY: YOUTUBE

YouTube

Smith
SCHOOL OF BUSINESS
Queen's University



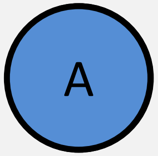
- Huge need for recommendations
- But compared to movies (Netflix) or books (Amazon):
  - Poor meta-data
  - Many items, relatively short
  - Short life cycle
  - Short and noisy interactions

Input Data

- Input Data (in call cases, quite noisy)
  - Content data
    - Raw video streams
    - Metadata (title, description, ...)
  - User activity data
    - Explicit: rating, liking, subscribing, ...
    - Implicit: watch, long watch
- Algorithm
  - Neighborhood-based collaborative filtering
    - Graph traversal

Example

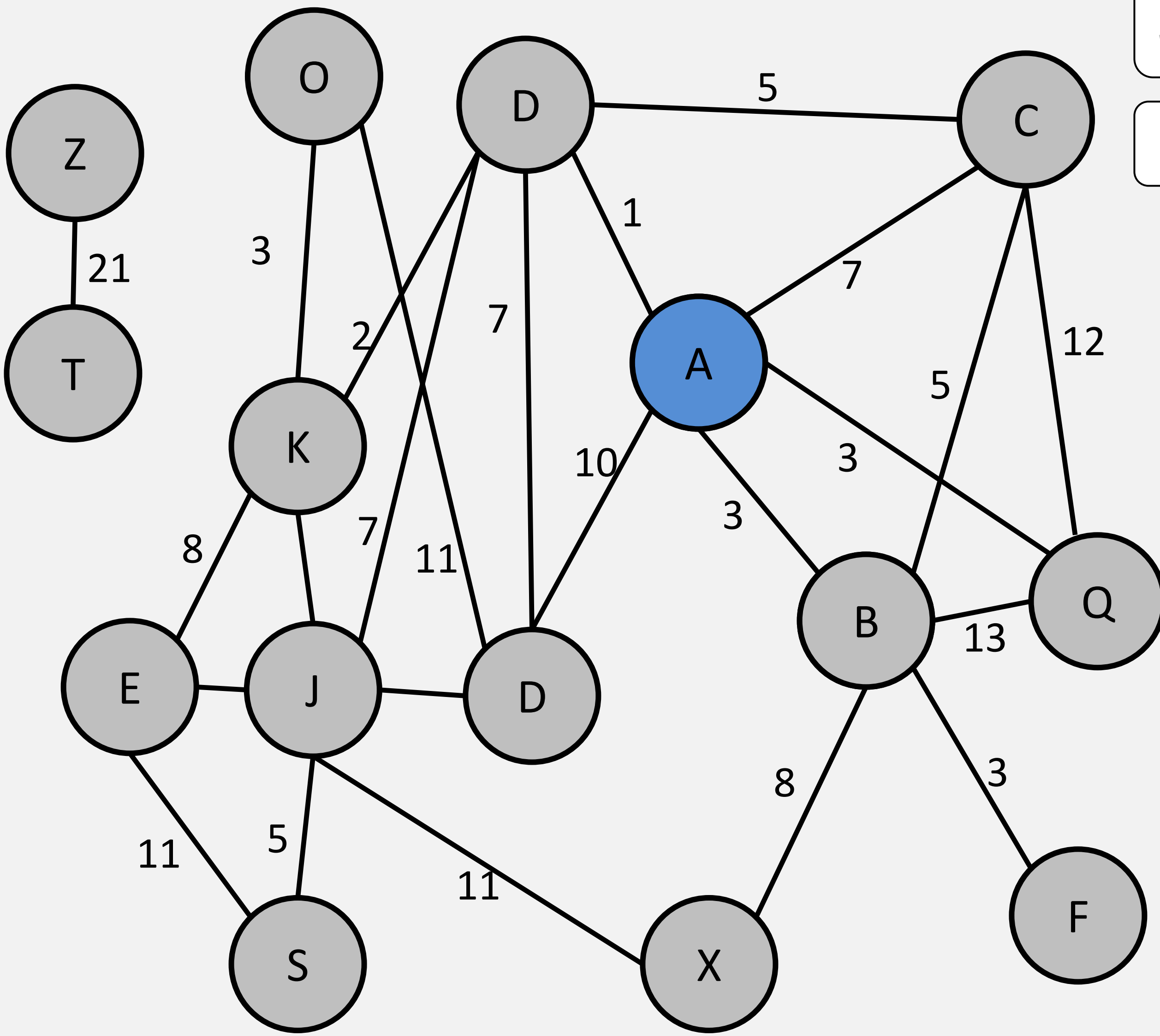
We know user likes video A.
What else to recommend?



Example

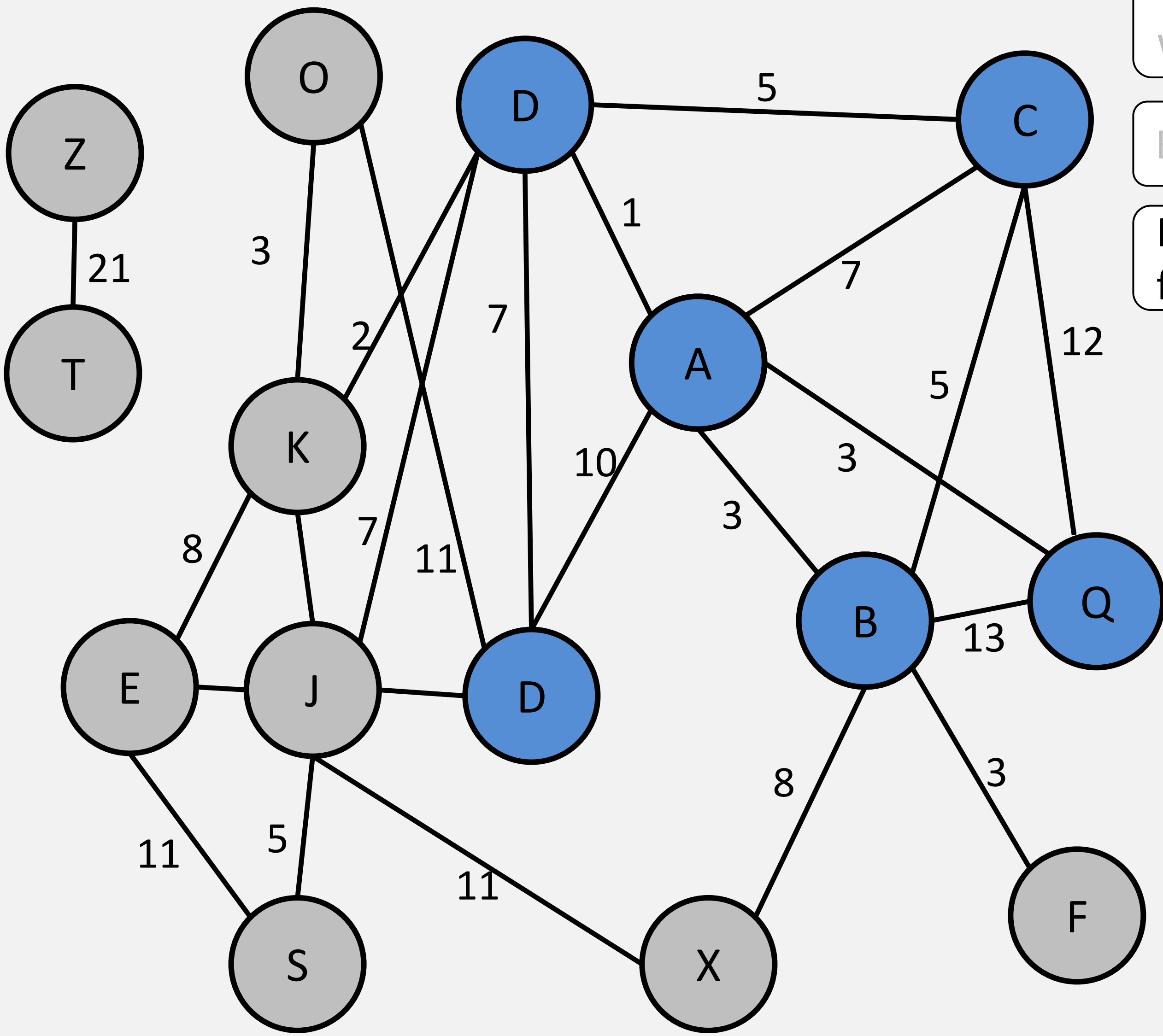
We know user likes video A.
What else to recommend?

Build co-visitation graph.



Nodes = videos
Edges = videos were viewed N times in the last 24 hours

Example



We know user likes video A. What else to recommend?

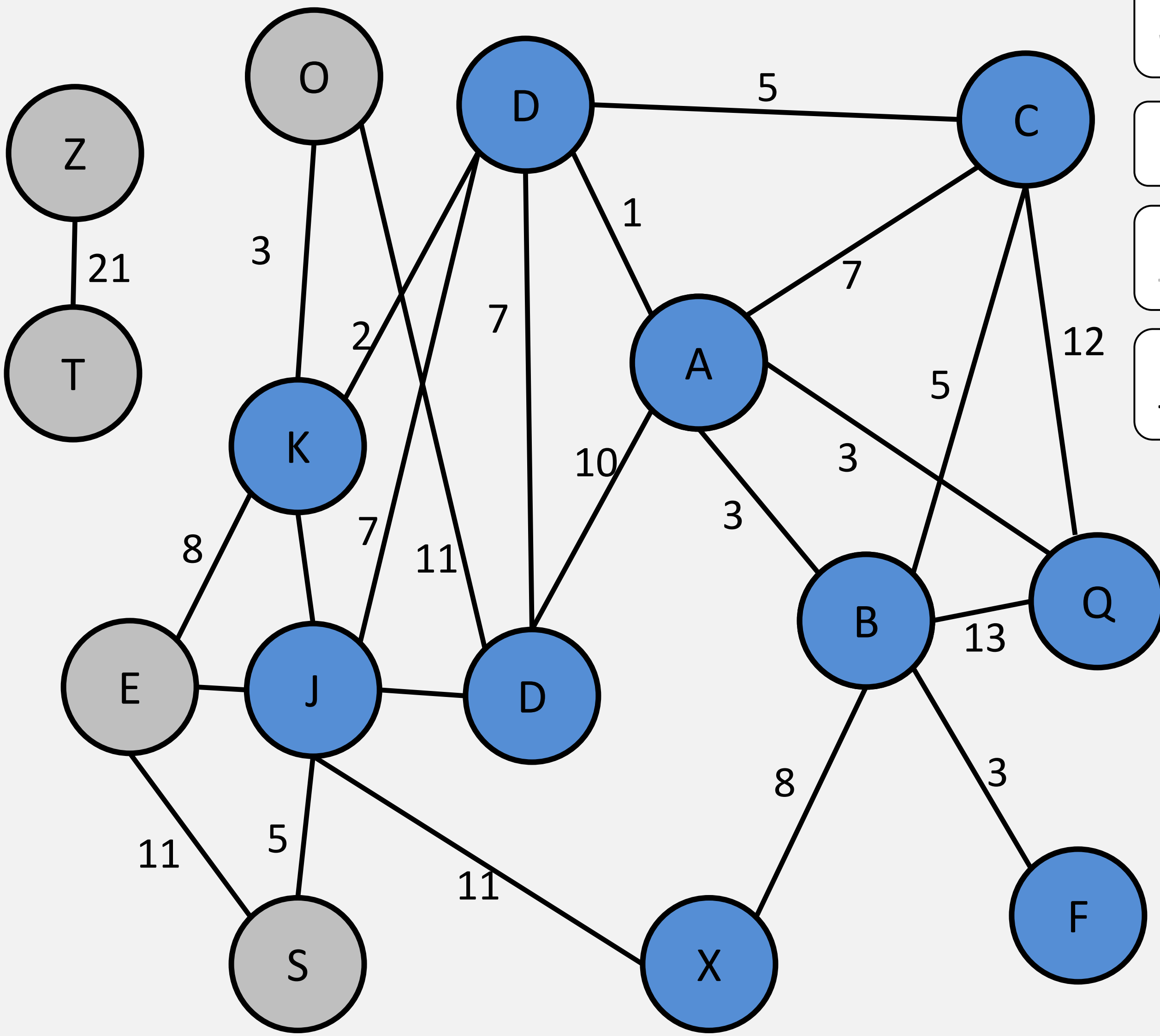
Build co-visitation graph.

Find all videos one hop away from A.

Nodes = videos

Edges = videos were viewed N times in the last 24 hours

Example



We know user likes video A. What else to recommend?

Build co-visitation graph.

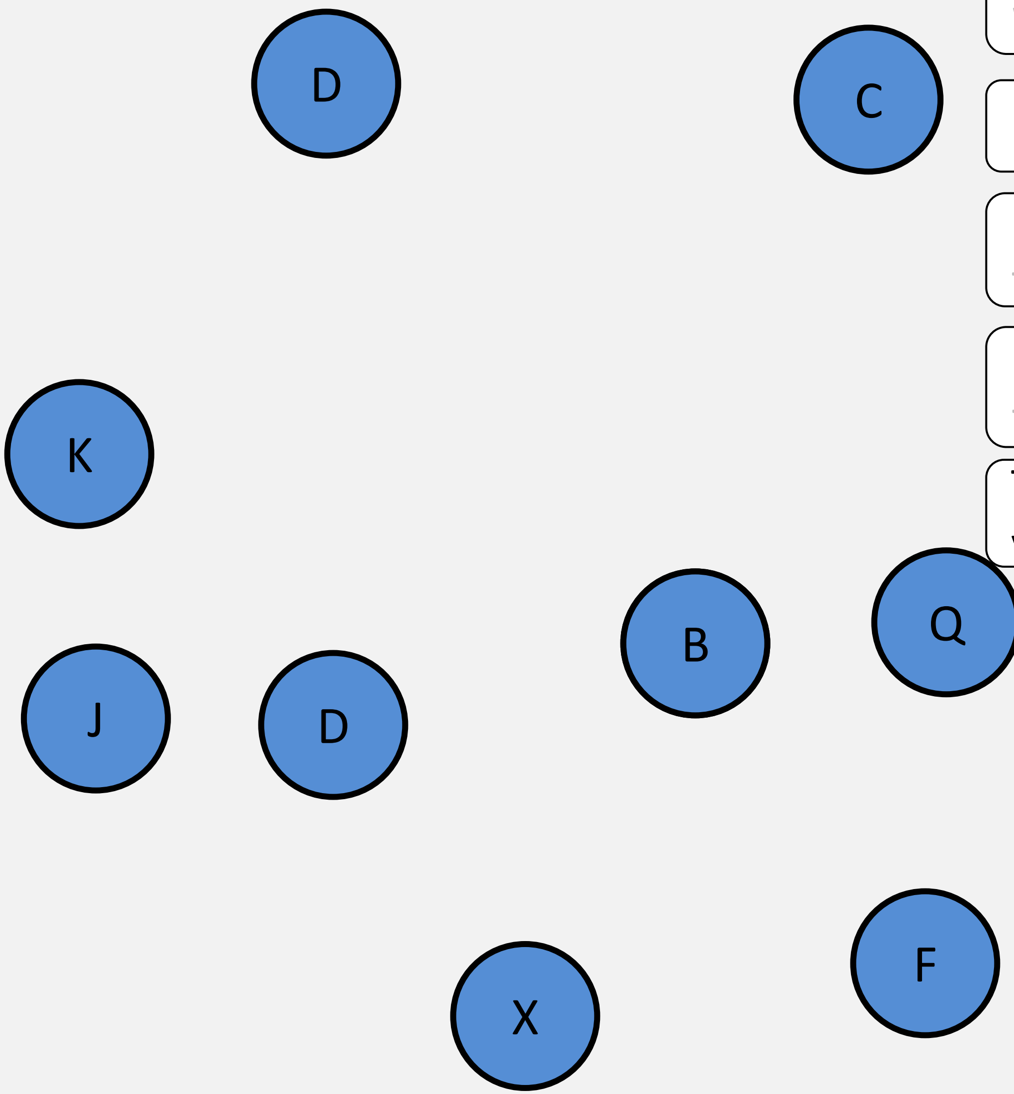
Find all videos one hop away from A.

Find all videos one hop away from those.

Nodes = videos

Edges = videos were viewed N times in the last 24 hours

Example



We know user likes video A. What else to recommend?

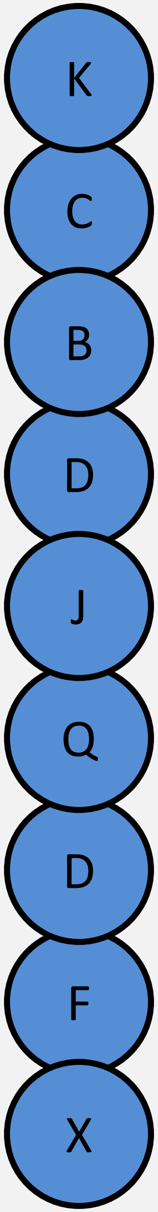
Build co-visitation graph.

Find all videos one hop away from A.

Find all videos one hop away from those.

These are the candidate videos.

Example



Ranking Criteria

- Video quality
  - Global stats
  - Total views, ratings, commenting, sharing, ...
- User specificity
  - Properties of the seed video
  - User watch history
- Diversification
  - Balance between relevancy and diversity
  - Limit on number of videos from the same author, same seed video

We know user likes video A. What else to recommend?

Build co-visitation graph.

Find all videos one hop away from A.

Find all videos one hop away from those.

These are the candidate videos.

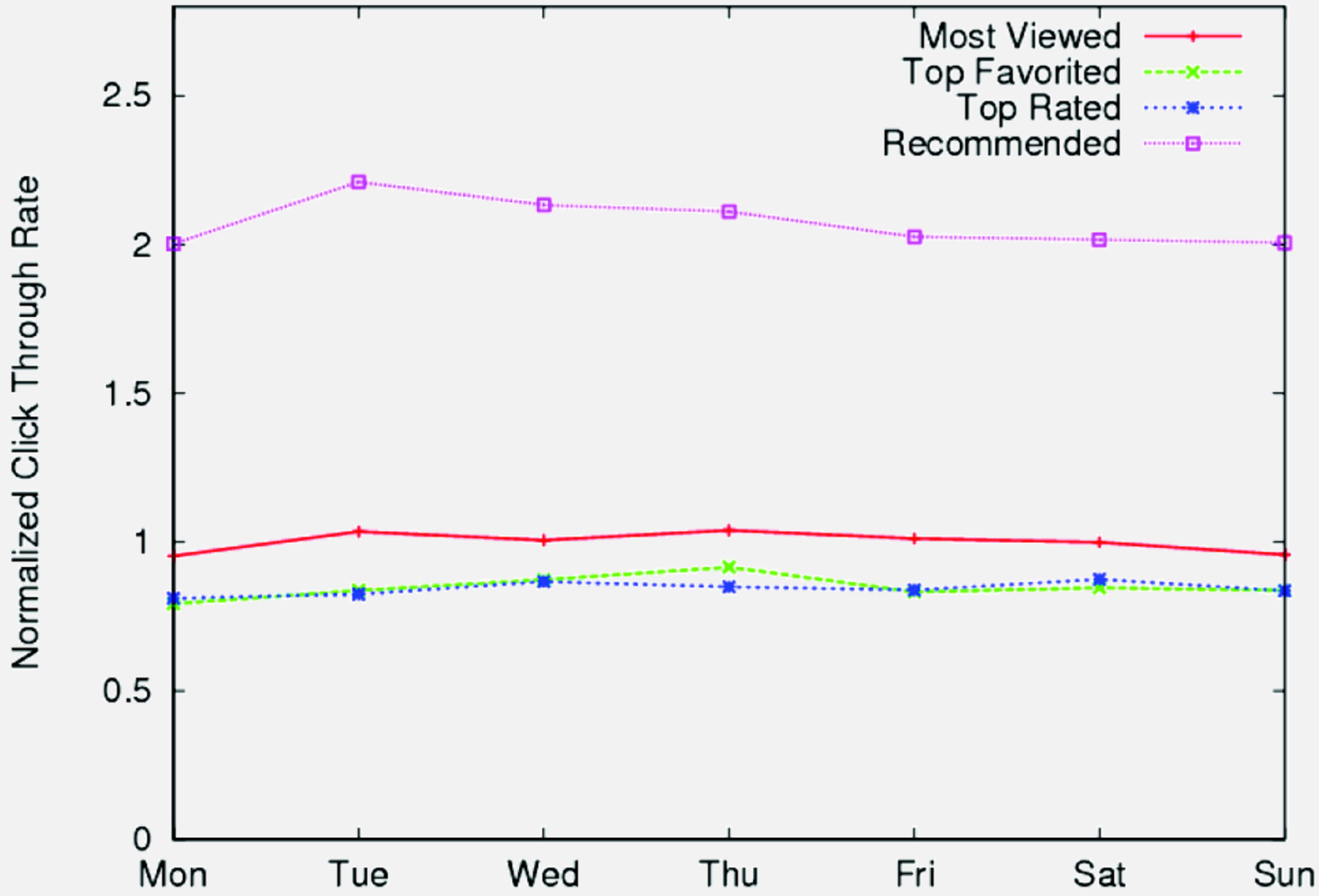
Rank the recommendations.

Implementation

Batch-oriented pre-computation approach

- Data collection
  - User data processed, stored in BigTable
- Recommendation generation
  - MapReduce implementation
- Recommendation serving
  - Pre-generated results quickly served to user

Evaluation



Most Viewed
Top Favorited
Top Rated
Recommended
Normalized Click Through Rate
0
0.5
1
1.5
2
2.5
Mon
Tue
Wed
Thu
Fri
Sat
Sun

OTHER CONSIDERATIONS

Evaluation of RS

- How to determine the quality of recommendations?
- **Offline testing**
  - Gather historical data
  - Split into training, testing
  - Measure metrics like MAE, RMSE, etc.
- **Online testing**
  - Friends and family
  - A/B testing

Context-Aware RS

Often times, a user's *context* should be taken into account

- Physical: location, time
- Environmental: weather, light, sound
- Personal: Health, mood, schedule, activity
- Social: who's in the room, group activity

Attacks

- Sellers of products and services have huge incentive to manipulate the output of a RS
  - Make a competitor's RS system worse
  - Influence rating (recommendations) of a particular item
- Collaborative filtering is susceptible to such attacks
- To make a competitor's RS worse
  - ***random attack***: generate profiles with random values
  - ***average attack:*** generate profiles with average values
  - ***bandwagon attack:*** high rating for "blockbusters", random values for others
- To influence rating of specific items
  - ***push attack:*** submit high ratings for "my" items
  - ***nuke attack:*** submit low ratings for "opponent's" items

Attacks: Countermeasures

- Increase cost of ratings
  - Captcha
  - Limited number of accounts for single IP address
- Automated attack detection
  - Detect *suspect users*
    - E.g., rate a high number of items, or rate very low or very high always

Mean Centering

- Potential problem with utility matrix
  - Some users usually rate everything high
  - Other users usually rate everything low
- Solution: ***mean centering***
  - Subtract mean for each user

5	3	5	4
1	2	1	1
1		5	
5	5	5	5

Original Utility Matrix

4.25
1.25
3
5

Mean

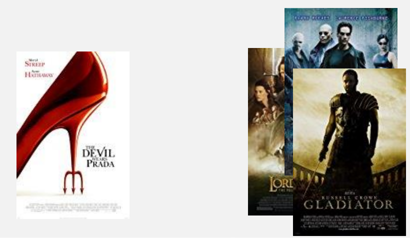
0.75	-1.25	0.75	-0.25
-0.25	0.75	-0.25	-0.25
-2		2	
0	0	0	0

Mean Centered Utility Matrix

Cluster Items in the Utility Matrix

- If utility matrix is sparse, probably just a few users that watched some movies
- Can run a clustering algorithm on the items to reduce the number of "items"
  - Matrix entry is now average of the user's rating of all the movies in that cluster

1.0		0.2	
	0.5		0.6
0.3		1.0	
	0.6		0.4



Cluster 1	Cluster 2	...
0.5	0.8	
0.4	0.7	
0.3	0.6	
0.2	0.6	

Cluster Users in the Utility Matrix

- Some users may have similar tastes but not have watched the exact same movies
- Can run a clustering algorithm on the users to reduce the number of "users"
  - Matrix entry is now average of rating of movie by all users in cluster

1.0		0.2	
	0.5		0.6
0.3		1.0	
	0.6		0.4

0.6		0.5	
	0.5		0.5

CASE STUDY: NETFLIX

Netflix

- Internet TV space is highly competitive
- Netflix's RS is a key pillar of their product
  - Not one algorithm, but a collection of different algorithms to serve different use cases

Motivation

- Internet TV is about choice
  - What to watch, when to watch, and where to watch
- But, humans are bad at choosing between many options
  - Members lose interest after 60-90 seconds, having reviewed 10-20 titles (3 in detail)
- Customers *think* they want:
  - As much choice as possible
  - Comprehensive search and navigation tools
- But, what customers *actually need* is a few compelling choices simply presented
- Netflix's RS serves 100% of videos watched
  - 80% from Homepage recommendations, 20% from Search
- The RS needs to make sure that each member will find something compelling to view, and will understand why it might be of interest

The Historical Approach

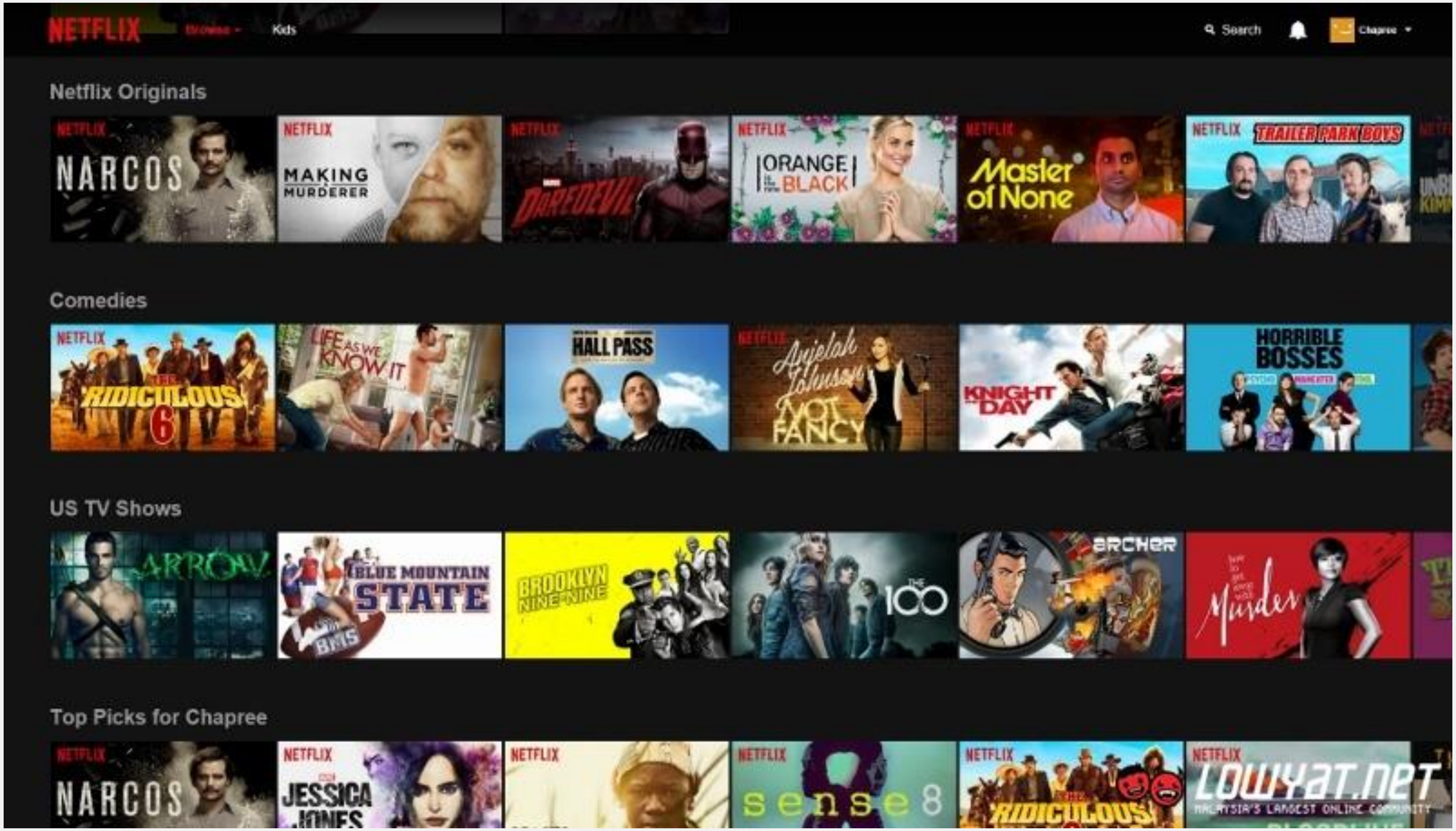
The Netflix RS problem used to be thought of as the problem of predicting the number stars that a user would rate a video.

- Used when Netflix shipped DVDs
- Main approach for the 2009 Netflix Challenge

The Modern Approach

- Much more data available now
  - What each member watches
  - How long it takes each member to finish watching
  - How/when each member discovered the video
  - The recommendations that were shown but were not played
- RS now consists of many algorithms that collectively define the Netflix experience
  - Personalized Ranker (CF)
  - Because You Watched (Context-based)
  - Top-N Ranker (Generic)
  - Trending Now (Generic)

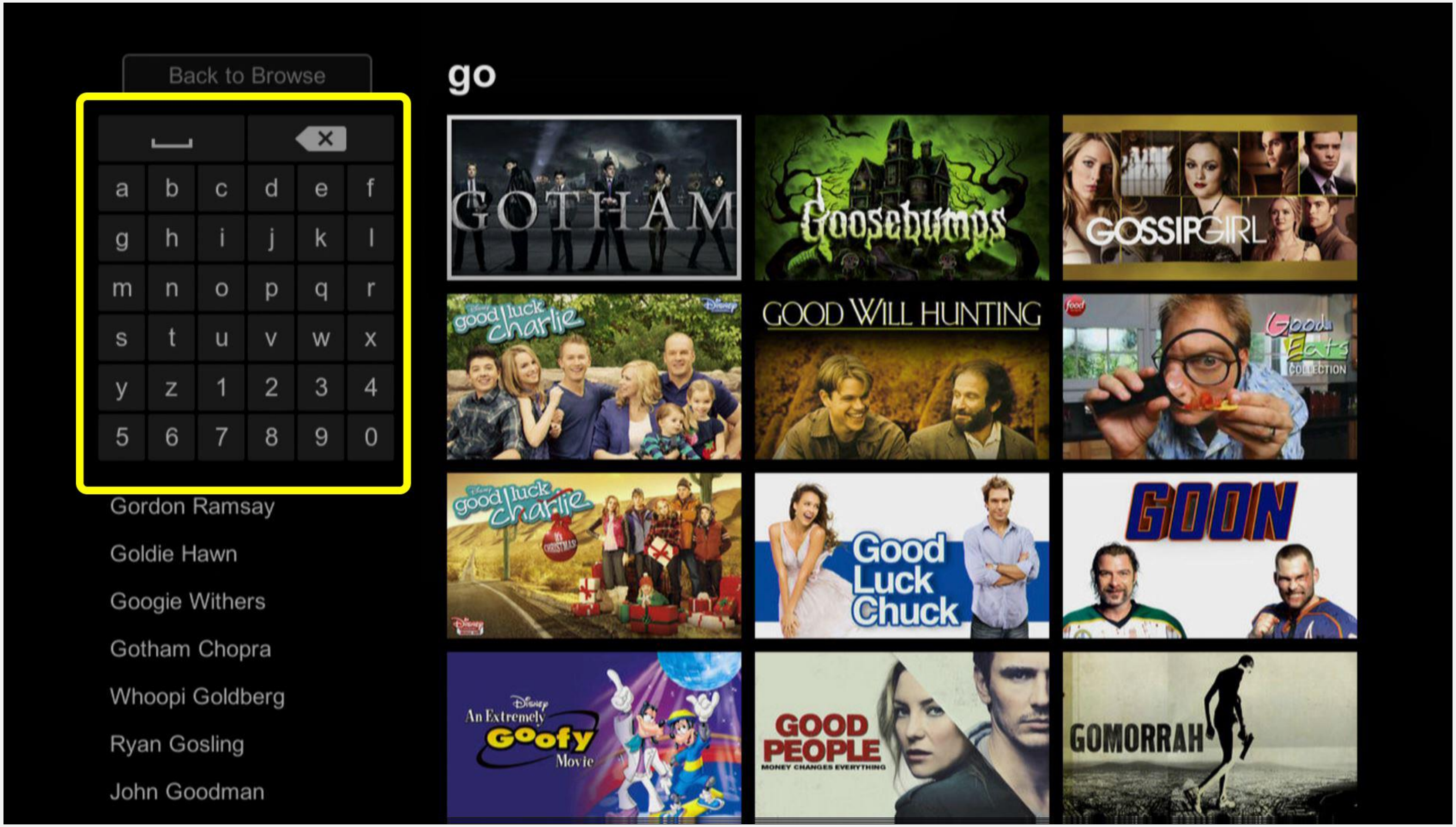
The Netflix Homepage



- Matrix-like layout
  - Typically 40 rows, 75 videos per row
- Each entry is a recommended video
- Rows of videos contain recommendations with a certain theme
  - From a single RS algorithm

Other Recommender Systems

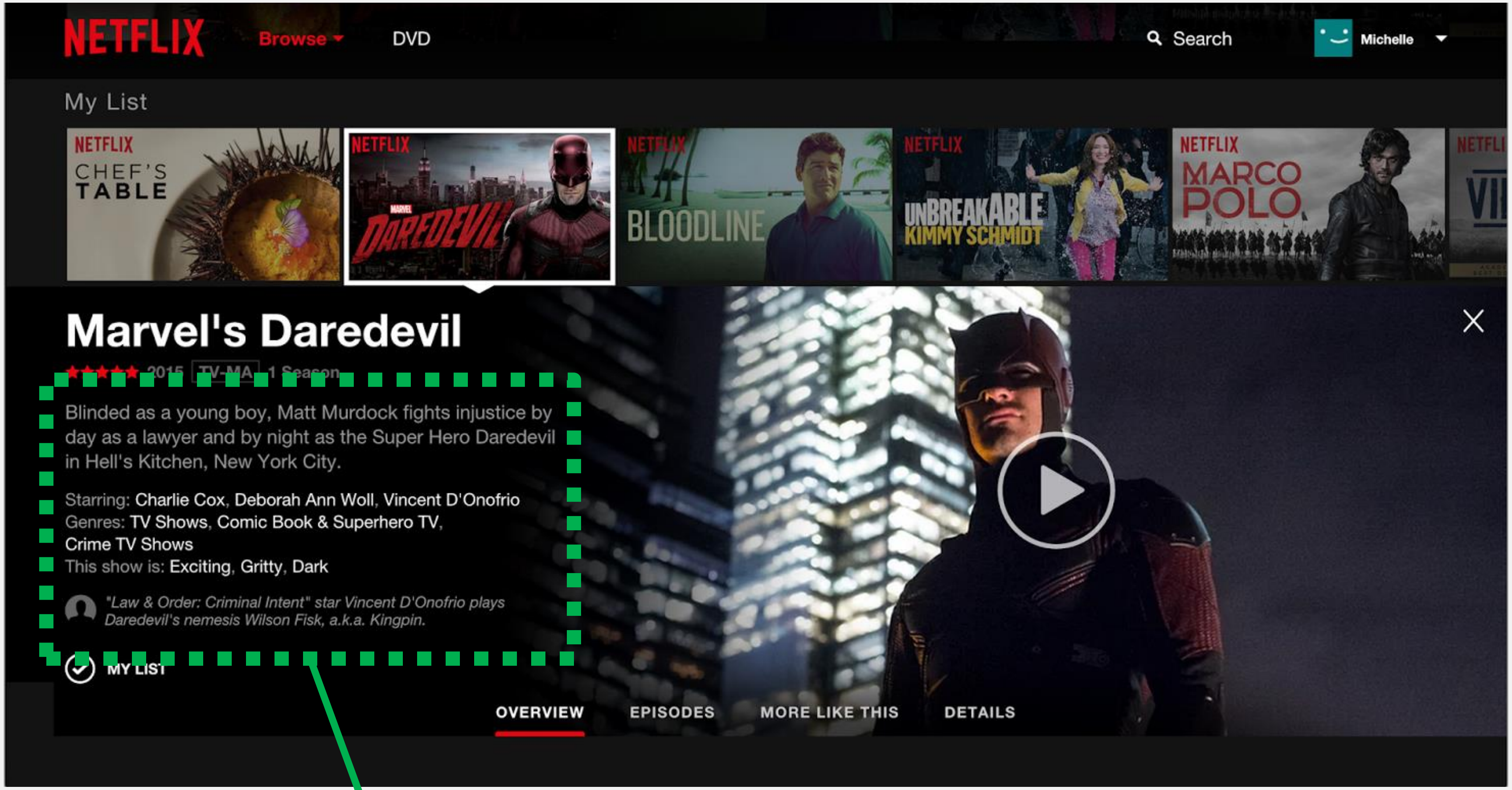
The *Search* RS



Which videos, actors, and concepts should be displayed for the partial query "go"?

Other Recommender Systems

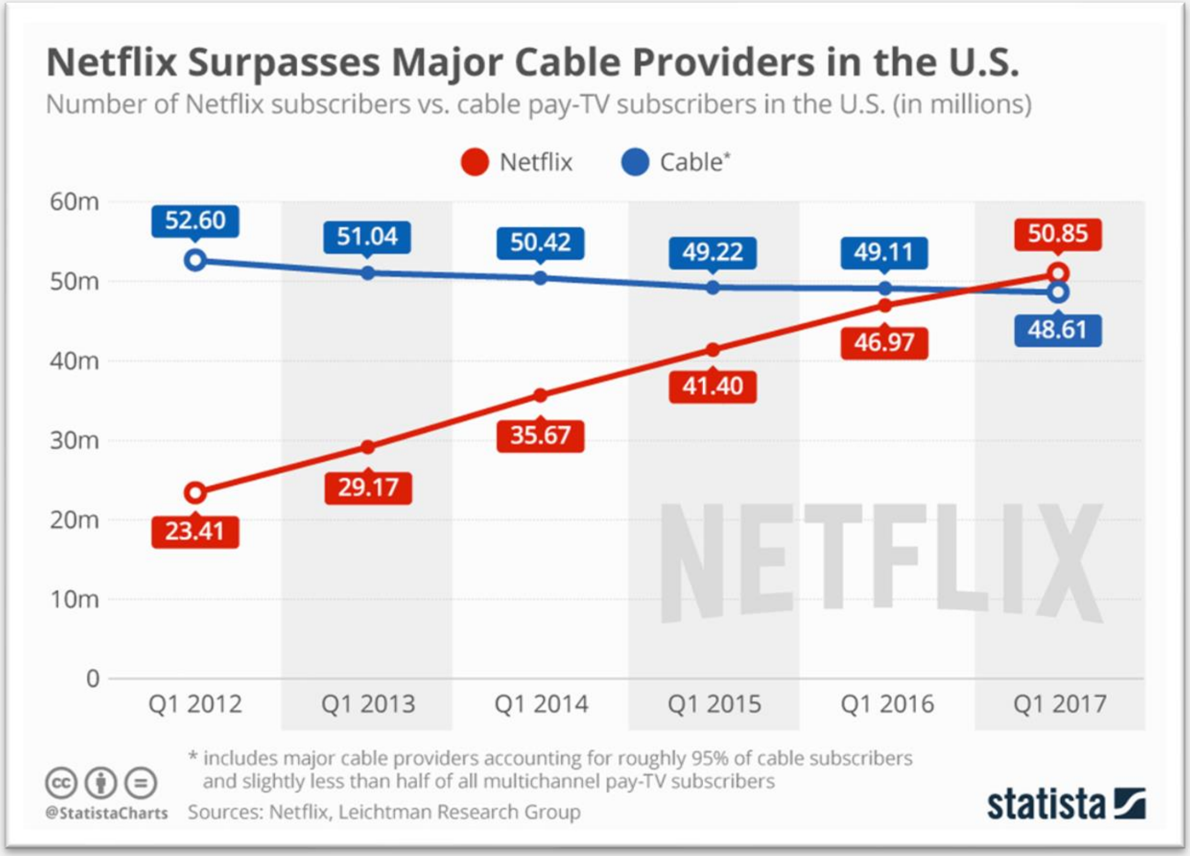
The *Evidence* RS



Which metadata should be included to best inform the user?

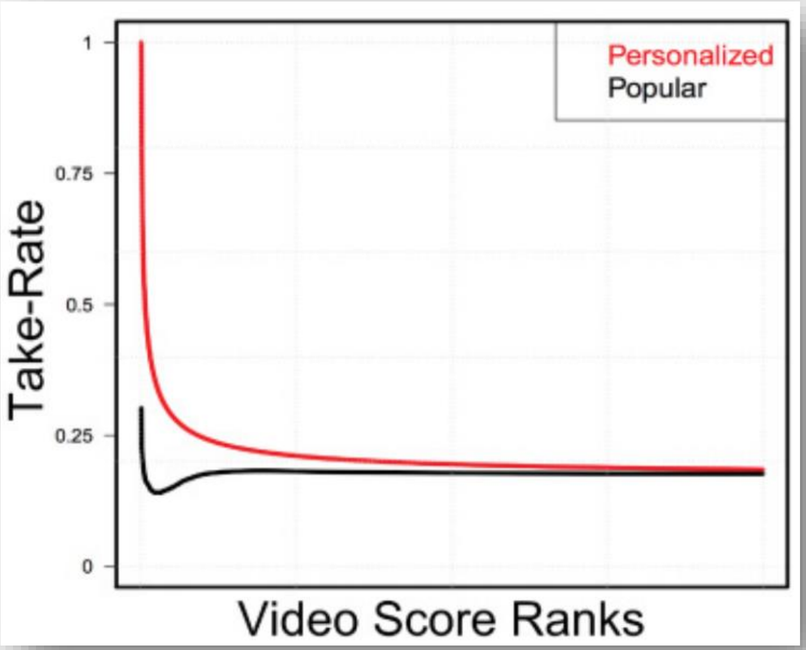
Business Value to Netflix

- Netflix wants to grow business on an enormous scale
- Allowing members to find engaging content prevents churn
- Netflix has a 100 person team for their RS
- Makes/saves $1B/year



Assessing the RS

- How well is the RS working?
  - ***Retention rate:*** fraction of customers that don't leave
  - ***Take-rate***: fraction of recommendations that get "taken"



- Ultimately, Netflix uses A/B testing
  - Mostly on new members
  - Same experience for entire test duration (not each session)

Key Open Problems

- Global populations and language
  - E.g., don't recommend House of Cards to someone who only speaks Thai, if House of Cards is not available in Thai
- Member coldstarting
  - RS does a good job helping long-time Members with lots of history
  - But not very well for new members
  - Current approach is to give a survey during the sign-up process

Key Open Problems

- Account Sharing
  - Owner, spouse, children all share account
  - How to provide personalized recommendations for multiple groups at the same time?
- Children
  - Younger children often like to watch the same content many times; not so for adults
  - Children's tastes change more rapidly

Why Recommender Systems?

Recommender systems are the perfect big data ML system!

- Do something humans can't do well at scale
- Easy for end-users to interact with
- $ Billions in economic lift
- Used by thousands of businesses