Recommendation Systems

Data Science Immersive

FLATIRON SCHOOL

By the end of the lesson students will be able to

- Explain the advantages/disadvantages of each type of recommendation system
- Implement code to calculate similarity metrics and determine predicted ratings in a collaborative filtering context
- Explain latent factor recommendation models and how they are created

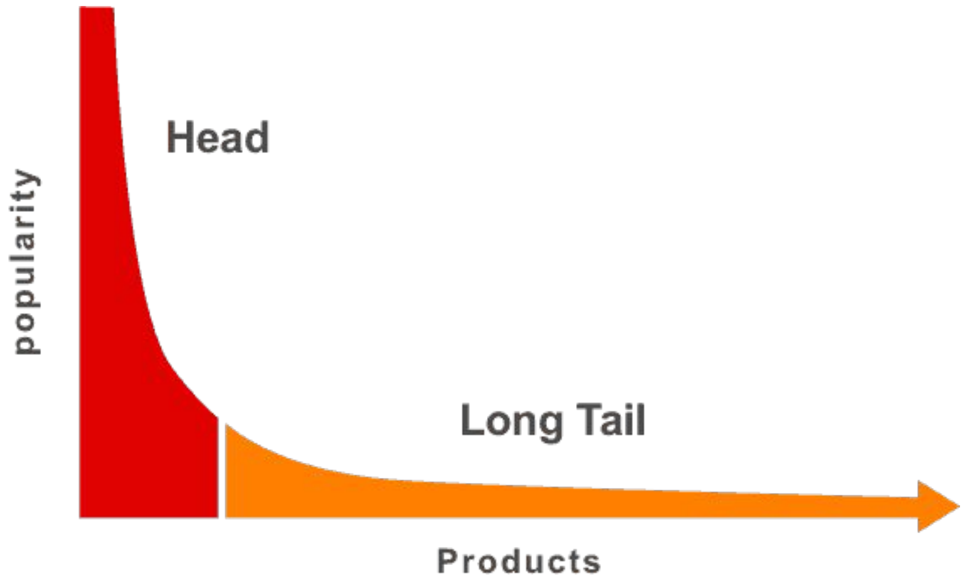
Outline

- Why Recommendation Systems?
- Content-Based
- Collaborative Filtering
    - Memory-based techniques (Neighborhood Based)
        - Similarity Metrics
        - Making Recommendations
    - Model-Based techniques
- Evaluating Recommendation Engines
- Issues with Recommendation Engines

Why Recommendation Systems??

amazon

Why Recommendation Systems??



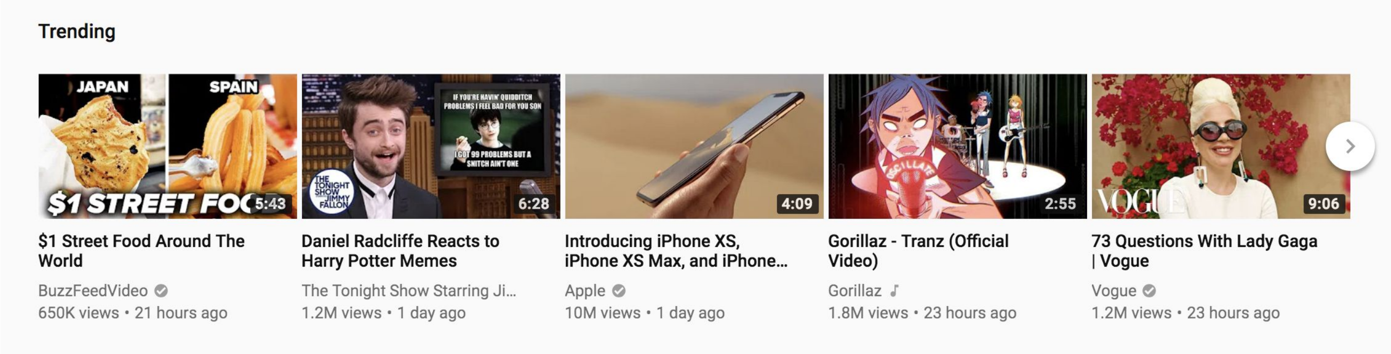
Benoit Mandelbrot: “Father of the Long Tail”

Types of Recommendation Engines

- **Non-personalized**
  - **Suggest the most popular items to users**
  - **Make the same suggestion to users regardless of characteristics**
- Content-Based
  - Recommend based on the properties of the items
  - Other user behavior is not considered
- Collaborative Filtering
  - Make use of user data to make recommendations
  - User - User
  - Item - Item
  - Memory Based
  - Model Based

Non-Personalized Recommendation Engines

- Non-personalized
  - Suggest the most popular items to users
  - Top 10, Top 5% etc...
  - Make the same suggestion to users regardless of characteristics



Non-Personalized Recommendation Summary

Advantages

- Super Easy (computationally and for the user to understand)
- Items are usually popular for a reason

Disadvantages

- Not personalized
- New items won’t gain traction

Types of Recommendation Engines

- Non-personalized
  - Suggest the most popular items to users
  - Make the same suggestion to users regardless of characteristics

- **Content-Based**
  - **Recommend based on the properties of the items**
  - **Other user behavior is not considered**

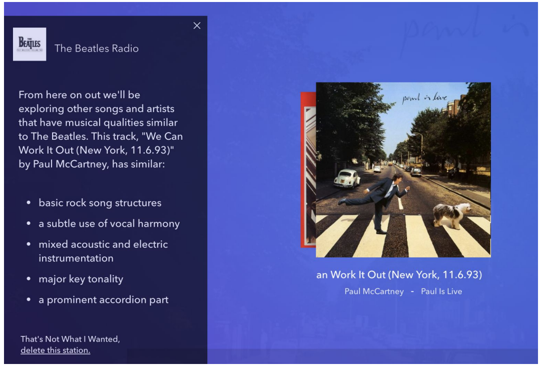
- Collaborative Filtering
  - Make use of user data to make recommendations
  - User - User
  - Item - Item
  - Memory Based
  - Model Based

Types of Recommendation Engines

Content-Based

- Recommend based on the properties of items
- User Profile and Item Profile
- Every item has the same set of attributes
- User profile is based off of an average of the item profiles that they’ve rated positively
- Similarity metrics are frequently based off of cosine similarity (angle)
- Often used when recommending similar articles (take the cosine similarity of TF-IDF vectors for each article)

pandora

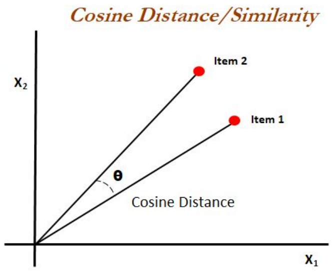


Content-Based Recommendation

Items						
Features		**A**	**B**	**C**	**D**	**...**
	Genre	2	3	5	1	**...**
	Actor	5	4	2	1	**...**
	Director	1	1	1	3	**...**
	Year	3	4	5	2	**...**
	IMDB Ratings	5	4	1	2	**...**

Content-Based Recommendation Engines

Genre	Actor	Director	Year	IMDB rating



item	score
Item 1	0.99
Item 2	0.97
Item 3	0.95

Content-Based Recommendation Summary

Advantages:

- More transparent
- No cold start issue (new items can be recommended immediately)
- Easy to do!
- Recommend items to user with unique tastes

Disadvantages:

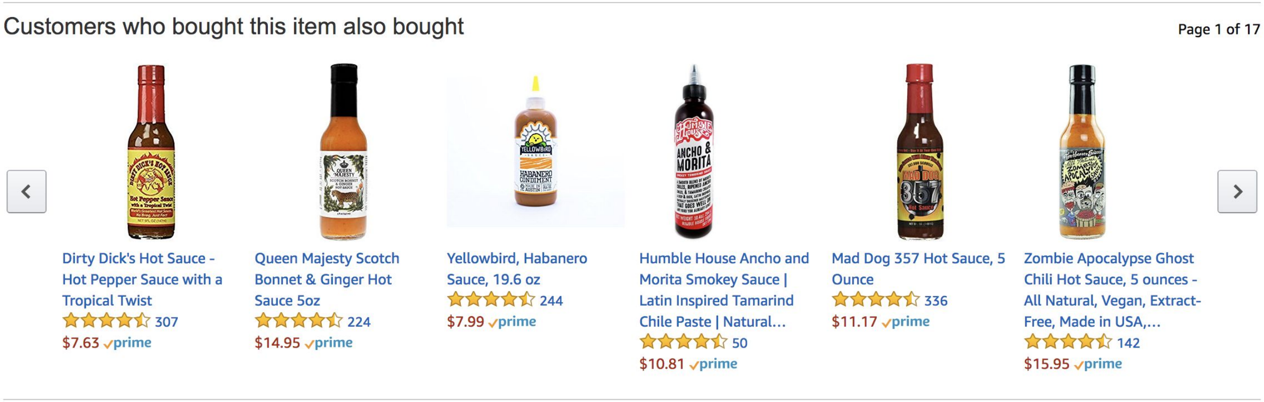
- Requires some type of tagging of items
- Overspecialization to certain types of items

Types of Recommendation Engines

- Non-personalized
  - Suggest the most popular items to users
  - Make the same suggestion to users regardless of characteristics
- Content-Based
  - Recommend based on the properties of the items
  - Other user behavior is not considered
- **Collaborative Filtering**
  - **Make use of user data to make recommendations**
  - **User - User**
  - **Item - Item**
  - **Memory Based (Neighborhood Techniques)**
  - Model Based

Types of Recommendation Engines

- Collaborative Filtering
    - Make use of other user data to make recommendations
    - **User - User**
    - **Item - Item**

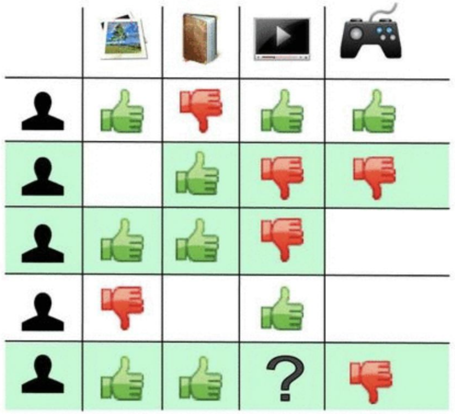


Utility Matrix

Utility Matrix: A matrix that contains users' ratings of different items

Utility Matrix

	King Kong	LOTR	Matrix	National Treasure
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4



Utility Matrix

Our goal is to make predictions about all of the blank values and then return the new items that have the highest predicted rating

	Movie 1	Movie 2	Movie ...	Movie N
User 1	1	BLANK	BLANK	3
User 2	BLANK	5	BLANK	3
User 3	BLANK	BLANK	1	BLANK
User 4	2	3	BLANK	BLANK
User 5	BLANK	BLANK	1	BLANK
User 6	4	BLANK	5	BLANK
User 7	BLANK	4	BLANK	BLANK
User ...	BLANK	3	BLANK	BLANK
User m	BLANK	BLANK	BLANK	4

ALS

	Movie 1	Movie 2	Movie ...	Movie N
User 1	1	4	2	3
User 2	1	5	3	3
User 3	2.5	2.8	1	3.5
User 4	2	3	2	3.5
User 5	2.5	2.8	1	3.1
User 6	4	1.2	5	1.4
User 7	1	4	2.5	3
User ...	2	3	2	3
User m	1	4	2	4

Utility Matrix

Items					
Users		SW	HP	LTR	Avengers
	Cersei	2	?	5	1
	Daenerys	5	4	?	1
	Joffrey	?	1	1	3
	Jon	3	?	5	2
	Tyrion	?	4	?	2

Explicit Ratings

Typically matrices will be extremely sparse!

Utility Matrix

Items					
Users		SW	HP	LTR	Avengers
	Cersei	1	0	1	1
	Daenerys	1	1	0	1
	Joffrey	0	1	1	1
	Jon	1	0	1	1
	Tyrion	1	1	0	1

Implicit Ratings

If we don't have numerical ratings, we can make *assumptions* based on users' behavior.

What are some examples?

User-User Similarity

Items					
Users		SW	HP	LTR	Avengers
	Cersei	2	?	5	1
	Daenerys	5	4	?	1
	Joffrey	?	1	1	3
	Jon	3	?	5	2
	Tyrion	?	4	?	2

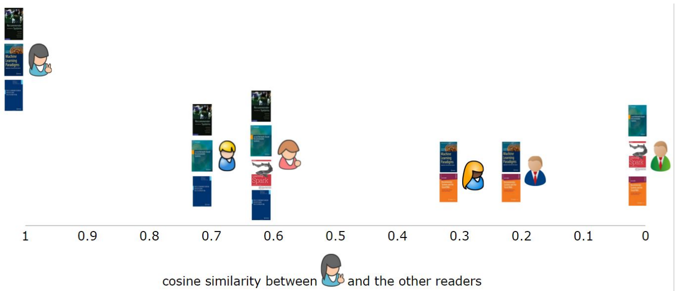
1. Choose similarity metric
2. Compare similarity of one user to other users
3. Take a weighted average of N similiar users' rating of the movies

Collaborative Filtering (User-User)

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

Find user-user similarity

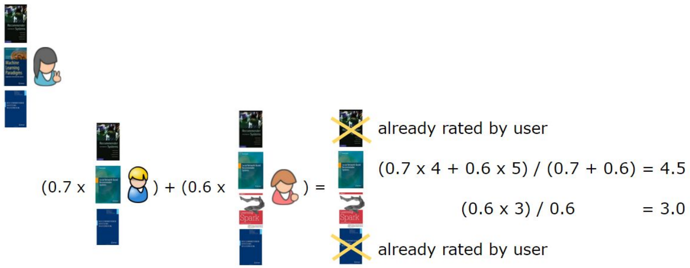
1.00	0.75	0.63	0.22	0.30	0.00
0.75	1.00	0.91	0.00	0.00	0.16
0.63	0.91	1.00	0.00	0.00	0.40
0.22	0.00	0.00	1.00	0.97	0.64
0.30	0.00	0.00	0.97	1.00	0.53
0.00	0.16	0.40	0.64	0.53	1.00



Collaborative Filtering (User-User)

1.00	0.75	0.63	0.22	0.30	0.00
0.75	1.00	0.91	0.00	0.00	0.16
0.63	0.91	1.00	0.00	0.00	0.40
0.22	0.00	0.00	1.00	0.97	0.64
0.30	0.00	0.00	0.97	1.00	0.53
0.00	0.16	0.40	0.64	0.53	1.00

Make rating estimations



Item-Item Similarity

Items					
Users		SW	HP	LTR	Avengers
	Cersei	2	?	5	1
	Daenerys	5	4	?	1
	Joffrey	?	1	1	3
	Jon	3	?	5	2
	Tyrion	?	4	?	2

1. Choose similarity metric
2. Compare similarity of items to the item that you are trying to determine the rating of
3. Multiply the similarity of each item by the rating of other items a given user has rated

Collaborative Filtering (Item-Item)

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

Find item-item similarity

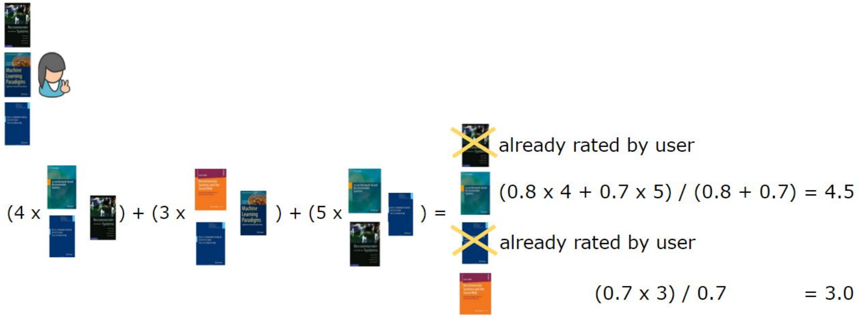
1.00	0.27	0.79	0.32	0.98	0.00
0.27	1.00	0.00	0.00	0.34	0.65
0.79	0.00	1.00	0.69	0.71	0.18
0.32	0.00	0.69	1.00	0.32	0.49
0.98	0.34	0.71	0.32	1.00	0.00
0.00	0.65	0.18	0.49	0.00	1.00

Made by Amazon:
https://dl.acm.org/citation.cfm?id=642471

Collaborative Filtering (Item-Item)

1.00	0.27	0.79	0.32	0.98	0.00
0.27	1.00	0.00	0.00	0.34	0.65
0.79	0.00	1.00	0.69	0.71	0.18
0.32	0.00	0.69	1.00	0.32	0.49
0.98	0.34	0.71	0.32	1.00	0.00
0.00	0.65	0.18	0.49	0.00	1.00

Calculate Predicted Utility for each person



User-User or Item-Item?

	Items			
Users	A	B	C	D
Cersei	2	?	5	1
Daenerys	5	4	?	1
Joffrey	?	1	1	3
Jon	3	?	5	2
Tyrion	?	4	?	2

	Items			
Users	A	B	C	D
Cersei	2	?	5	1
Daenerys	5	4	?	1
Joffrey	?	1	1	3
Jon	3	?	5	2
Tyrion	?	4	?	2

Which one do you think will perform better?

User-User or Item-Item, which wins??

- Item-Item or User-User Collaborative Filtering????
  - In general, item-item has proven to be more effective than user-user
  - Users have unique tastes that are difficult to predict
- Depends on whether you have a higher number of items or users

Given *m* : users, *n* : items

Time Complexity

User-user ≈ $O(m^2n)$
Item-Item ≈ $O(mn^2)$

Memory Based Collaborative Filtering

Also known as **neighborhood-based**:

These are models that are based off of how similar users/items are for items that have already been rated

We want users with more similar tastes to have higher similarity than those with different tastes. We have multiple ways of achieving this:

1) Jaccard Similarity
2) Euclidean Distance
3) Cosine Similarity
4) Pearson Correlation

Similarity Metrics

- Jaccard Index
  - Typically used for implicit data
  - The intersection of items rated by users divided by the union of items rated by both users
  - Ignores rating values!

$$sim(A, B) = \frac{|r_A \cap r_B|}{|r_A \cup r_B|}$$

Similarity Metrics

- Euclidean Distance
  - A simple measure of distance between the vectors

Euclidean Distance

$$r_2(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + \cdots + (x_{1d} - x_{2d})^2} = \sqrt{\sum_{j=1}^{d} (x_{1j} - x_{2j})^2}.$$

Manhattan Distance

$$r_1(\mathbf{x}_1, \mathbf{x}_2) = |x_{11} - x_{21}| + |x_{12} - x_{22}| + \cdots + |x_{1d} - x_{2d}| = \sum_{j=1}^{d} |x_{1j} - x_{2j}|.$$

To make this measurement more useful, we can normalize this measurement between 0 and 1.

$$\mathrm{sim}(\mathbf{x}, \mathbf{y}) = \frac{1}{1 + r_2(\mathbf{x}, \mathbf{y})}.$$

Similarity Metrics

- Cosine Similarity
  - We assume that the unrated values are 0
  - Scale between [-1,1]
  - Issue: It treats the missing ratings as if the person has rated them poorly
    - This can lead to misleading information for someone who has not rated many movies

$$sim(A, B) = \frac{r_A \cdot r_B}{||r_A|| \, ||r_B||}$$

Similarity Metrics

- Adjusted Cosine Similarity
  - Insensitive to the magnitude of users' ratings and trends of users
    - For example, if one critic rated everything as 5 and another rated everything as 1
    - Positive Ratings mean that a user liked an item more than average, negative ratings means they liked a movie less than average

$$AC(i,j) = \frac{\sum_{U \in u_{ij}} (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_{U \in u_{ij}} (r_{ui} - \bar{r}_u)^2 \sum_{U \in u_{ij}} (r_{uj} - \bar{r}_u)^2}}$$

Similarity Metrics

- Pearson Correlation/
  - Find the correlation between users' ratings
  - Scaled from -1 to +1
  - Only includes set *I* of items that both users have rated

$$PCC\_Sim(u,v) = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_{u,I})(r_{v,i} - \bar{r}_{v,I})}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_{u,I})^2 + \sum_{i \in I} (r_{v,i} - \bar{r}_{v,I})^2}}$$

Where $\bar{r}_{u,I}$ **and** $\bar{r}_{v,I}$ **represents** Average rating of user u and user v , respectively ,for co-rated items represented by set I

Similarity Metrics

Which metric is best???

It Depends

although pearson correlation has been demonstrated experimentally to be the best

User-User Similarity Recommendation

Items					
Users		**SW**	**HP**	**LOTR**	**Avengers**
	Cersei	2	?	5	1
	Daenerys	5	4	?	1
	Joffrey	?	1	1	3
	Jon	3	**?**	5	2
	Tyrion	?	4	?	2

Avg rating for each **user**
2.66
3.33
1.66
3.33
3

Find Similarity (Using adjusted cosine similarity)

First we are going to mean normalize to account for individuals' average rating

User-User Similarity Rec. Part 2

Items					
Users		**SW**	**HP**	**LOTR**	**Avengers**
	Cersei	-0.66	0	2.33	-1.66
	Daenerys	1.66	0.66	0	-2.33
	Joffrey	0	-0.66	-0.66	1.33
	Jon	-0.33	?	1.66	-1.33
	Tyrion	0	1	0	-1

How similar Jon is to other users (using cosine similarity)

0.995
0.414
-0.896
1
0.6178

Compare similarity of one user to other users using adjusted cosine similarity

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

User-User Similarity Rec. Part 3

Take a weighted average of the users' rating of the movies

	Items		
Users		**SW**	**HP**
	Cersei	2	?
	Daenerys	5	4
	Joffrey	?	1
	Jon	3	**?**
	Tyrion	?	4

0.995
0.414
-0.896
1
0.6178

$$r_{xi} = \frac{\sum_{y \in N} s_{xy} r_{yi}}{\sum_{y \in N} s_{xy}}$$

((0.414 * 4) +
(0.6178*4)) /

(0.414 +0.6178)

= **4**

Local v. Global Effects (Advanced)

Local
"Naive"
Approach

$$\boxed{r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}}$$

s_{ij}… similarity of items i and j
r_{xj}…rating of user x on item j
$N(i;x)$… set items rated by x similar to i

Global

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

baseline estimate for r_{xi}

$$b_{xi} = \mu + b_x + b_i$$

- μ = overall mean movie rating
- b_x = rating deviation of user x
  = *(avg. rating of user x)* – μ
- b_i = rating deviation of movie i

Collaborative Filtering (Memory Based) Summary

Advantages:

- Personalized. You're special!

Disadvantages:

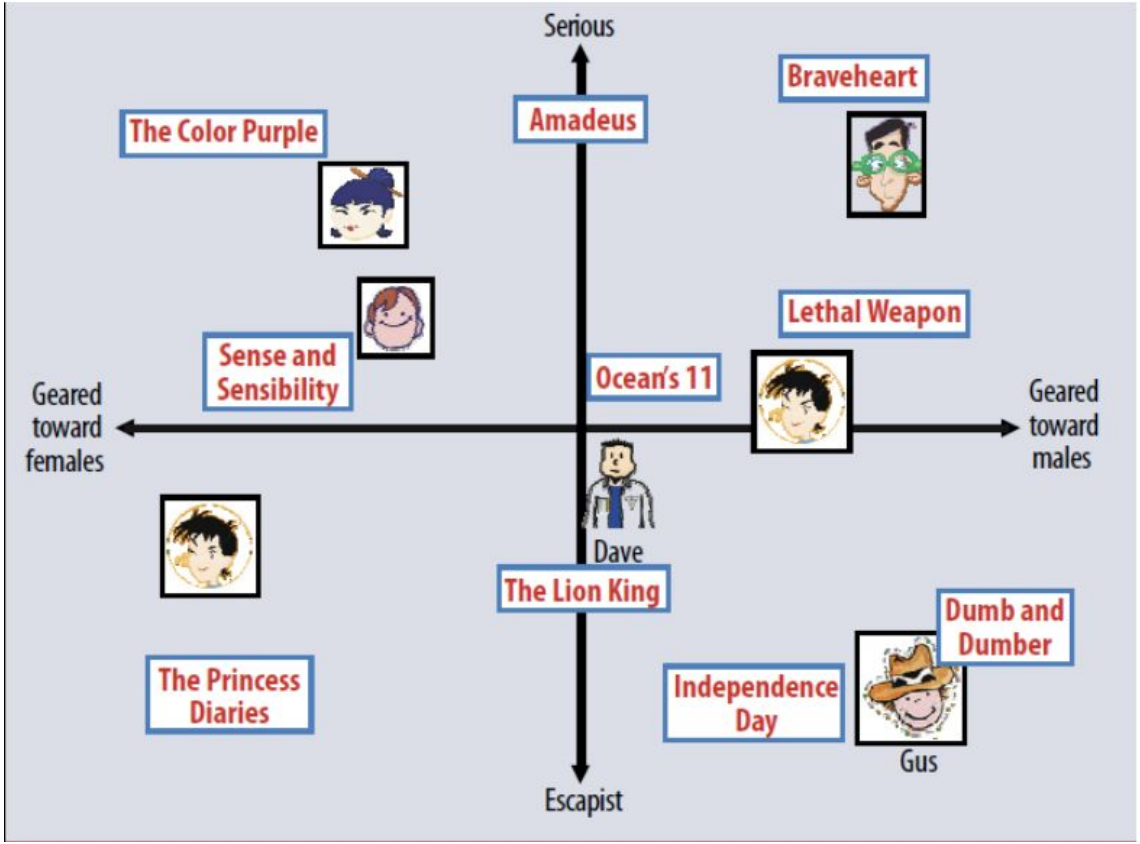
- Can require **a lot** of computation (most often the computation is done **offline** and the points are calculated easily)
- Cold start: need to have a lot of ratings to be worthwhile
- Popularity Bias: biased towards items that are popular. May not capture people's unique tastes.

Model Based Collaborative Filtering

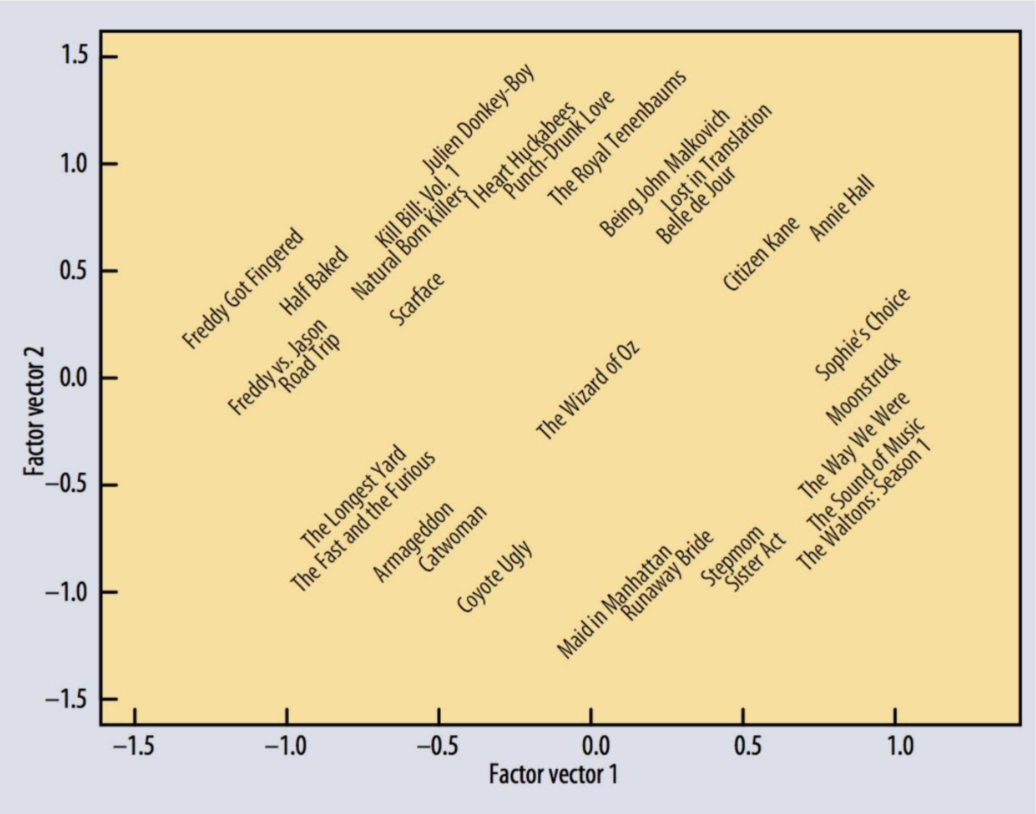
Types of Recommendation Engines

- Non-personalized
  - Suggest the most popular items to users
  - Make the same suggestion to users regardless of characteristics
- Content-Based
  - Recommend based on the properties of the items
  - Other user behavior is not considered
- **Collaborative Filtering**
  - Make use of user data to make recommendations
  - User - User
  - Item - Item
  - Memory Based
  - **Model Based (Matrix Factorization)**

Latent Features

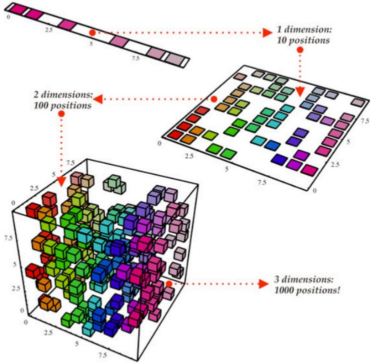


Latent Features

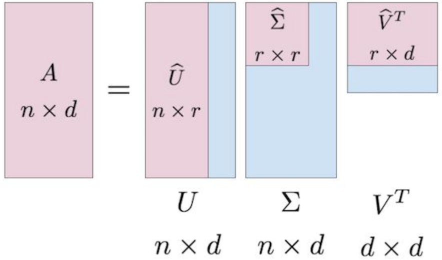


Dimensionality Reduction

What is a more compact way for us to represent our data?



We choose the value for *d*, the number of features



Through matrix factorization, we can discover "latent features" that are present in our data.

- U, V orthogonal matrices
- S diagonal matrix, diagonal entries $\sim$ singular values

Also known as "topic-modelling"

SVD Example

$$A = U \Sigma V^T$$

U is “user-to-concept” similarity matrix

S or Σ : “Strength” of each concept

Hidden Features that are discovered through matrix factorization.

Also known as “topic-modelling”

V : “movie-to- concept” similarity matrix

$$
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
3 & 3 & 3 & 0 & 0 \\
4 & 4 & 4 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 2 & 0 & 4 & 4 \\
0 & 0 & 0 & 5 & 5 \\
0 & 1 & 0 & 2 & 2
\end{bmatrix}
\end{array}
=
\begin{array}{c}
\begin{array}{cc} \text{SciFi-concept} & \text{Romance-concept} \end{array} \\
\begin{bmatrix}
0.13 & 0.02 & -0.01 \\
0.41 & 0.07 & -0.03 \\
0.55 & 0.09 & -0.04 \\
0.68 & 0.11 & -0.05 \\
0.15 & -0.59 & 0.65 \\
0.07 & -0.73 & -0.67 \\
0.07 & -0.29 & 0.32
\end{bmatrix}
\end{array}
\times
\begin{bmatrix}
12.4 & 0 & 0 \\
0 & 9.5 & 0 \\
0 & 0 & 1.3
\end{bmatrix}
\times
\begin{bmatrix}
0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
0.40 & -0.80 & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

- U, V orthogonal matrices
- S diagonal matrix, diagonal entries $\sim$ singular values

Modified SVD

- The issue with SVD:
  - SVD only works with non-sparse matrices
- There is a way to factor our matrices into two components
  - Each item is modeled by a vector $q_i \in \mathbb{R}^k$ $\quad$ $Q = U,\ P^T = \Sigma V^T$
  - Each user is modeled by a vector $p_u \in \mathbb{R}^k$
- Such that a value close to the actual rating r_{ui} can be computed by the dot product

$$r_{ui} \approx \hat{r}_{ui} = q_i^T p_u$$

Calculating Rating in Model Based Approach

- How to estimate the missing rating of user *x* for item *i*?

users

items

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

$\approx$

$$\hat{r}_{xi} = q_i \cdot p_x$$

$$= \sum_f q_{if} \cdot p_{xf}$$

q_i = row $\boldsymbol{i}$ of $\boldsymbol{Q}$

p_x = column $\boldsymbol{x}$ of $\boldsymbol{P}^T$

items

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

factors

$\boldsymbol{Q}$

$\bullet$

users

factors

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

$\boldsymbol{P}^T$

Alternating Least Squares

- Matrices are determined by minimizing the regular square error on only known ratings. Usually done through Alternating Least Squares:
  - Usually slower to calculate than Stochastic Gradient Descent, but it is more parallelizable
- Its training routine is different: ALS minimizes two loss functions alternatively; It first holds user matrix fixed and runs gradient descent with item matrix; then it holds item matrix fixed and runs gradient descent with user matrix

$$\min_{q\cdot,p\cdot} \sum_{(u,i)\in R_o} (r_{ui} - \langle p_u, q_i \rangle)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

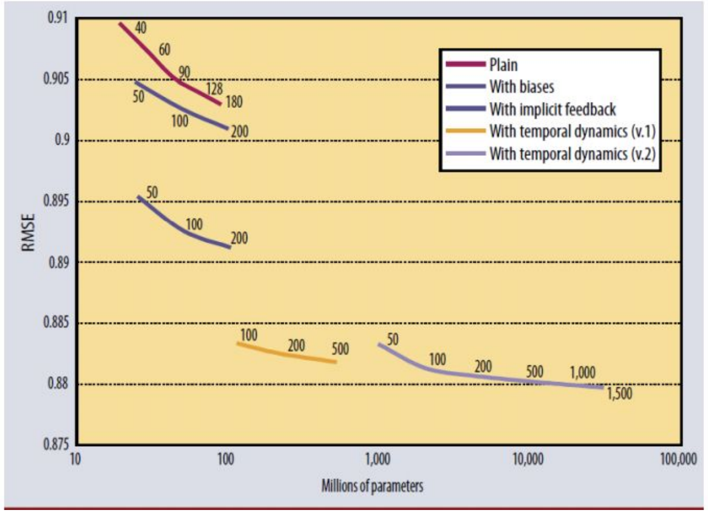
Collaborative Filtering (Model Based) Summary

Advantages:

- The best in class models use latent features

Disadvantages:

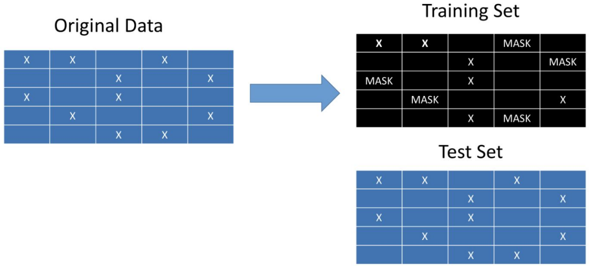
- Not interpretable to users
- Cold start problem
- Computational Complexity



Evaluating Recommenders

- We can quantify goodness of fit for recommendation engines using RMSE if we are estimating explicit ratings
- If we are estimating implicit ratings, we would use classification metrics such as ROC curve, AUC, Accuracy, Precision

$$\mathrm{RMSE} = \sqrt{\frac{1}{|R_o|} \sum_{(u,i)\in R_o} (r_{ui} - \hat{r}_{ui})^2}$$



Problems with Evaluating Systems

- Accurate models tend to predict items that are extremely similar
  - Not great for displaying a wide diversity of items
- Prediction Context
- Order of Predictions
- “Offline Evaluation”: Based on historic data
- “Online Evaluation”: A/B Testing with users

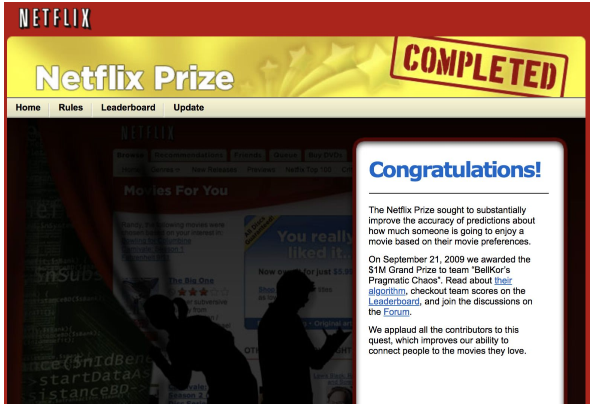
Best way to evaluate…..

What makes you the most money! (Or keeps your customers satisfied)

Often done with “online evaluation”

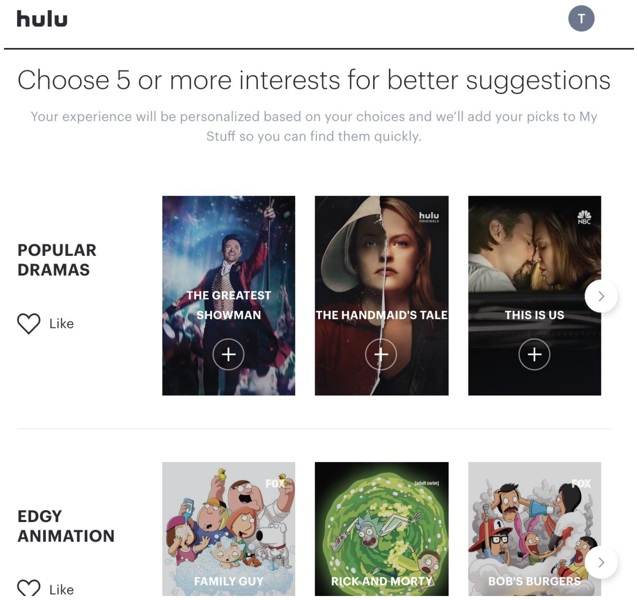
Problems with Evaluating Systems

- Sometimes the best performing systems are not optimized for reality!
- Netflix Starting RMSE: 0.9525
- BellKor's Pragmatic Chaos RMSE: 0.8567

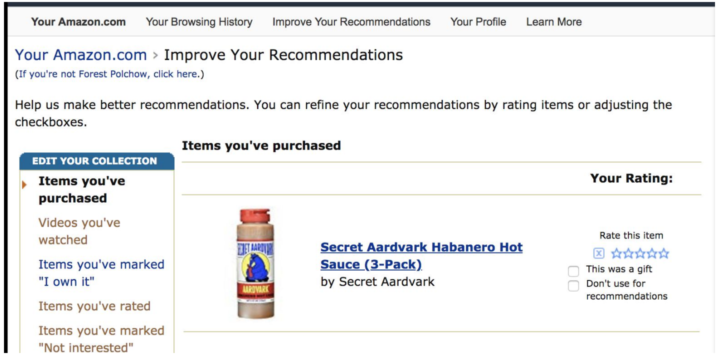


Issues with Recommendation Engines

- The “Cold Start” Problem
  - When a new user signs up, we don’t know anything about their preferences. What can we do?



- Force users to rate a certain number of items when they join
- Force users to integrate their social media information
- Show users the most popular items



More Models!!

- Hybrid Methods
  - Models using a combination of content-based similarity and collaborative filtering. Some also include demographic information, previous behavior, etc.
  - Survey of Hybrid Recommender Systems
  - Recommendation systems based on previous behavior

- Deep Learning: Some of the best performing models
  - Survey of Deep Learning Recommendation Systems
  - Git Repo with a ton of resources on deep learning recommendation systems
- It's not all about the accuracy
  - http://ir.ii.uam.es/rim3/publications/ddr11.pdf

Python Libraries

- http://surpriselib.com/
- https://spark.apache.org/docs/2.2.0/mllib-collaborative-filtering.html

Smaller Scale

Large Scale

THANK YOU !

User-User or Item-Item, which wins??

Item-Item Based Algorithm

- for every item i that u has no preference for yet
  - for every item j that u has a preference for
    - compute a similarity s between i and j
    - add u's preference for j, weighted by s, to a running average
- return the top items, ranked by weighted average

User-User Based Algorithm

- for every item i that u has no preference for yet
  - for every other user v that has a preference for i
    - compute a similarity s between u and v
    - add v's preference for i, weighted by s, to a running average
- return the top items, ranked by weighted average

Item-Item Similarity Breakout

Items					
Users		SW	HP	LOTR	Avengers
	Cersei	2	?	5	1
	Daenerys	5	4	?	1
	Joffrey	**?**	1	1	3
	Jon	3	?	5	2
	Tyrion	?	4	?	2

Using Item-Item similarity, how much would Joffrey like Star Wars?
Assume N = 2 Try different similarity metrics!

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

Item-Item Similarity Breakout

Items					
Users		**SW**	**HP**	**LOTR**	**Avengers**
	Cersei	2	?	5	1
	Daenerys	5	4	?	1
	Joffrey	**1**	1	1	3
	Jon	3	?	5	2
	Tyrion	?	4	?	2