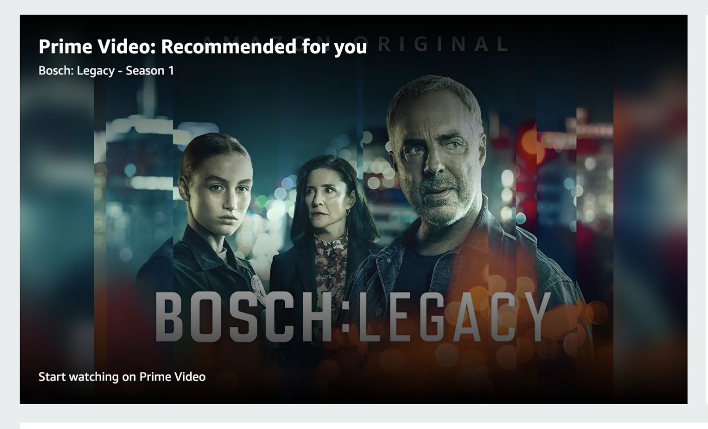
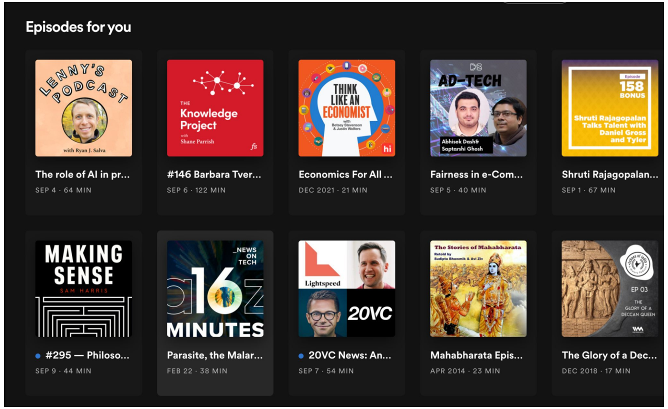
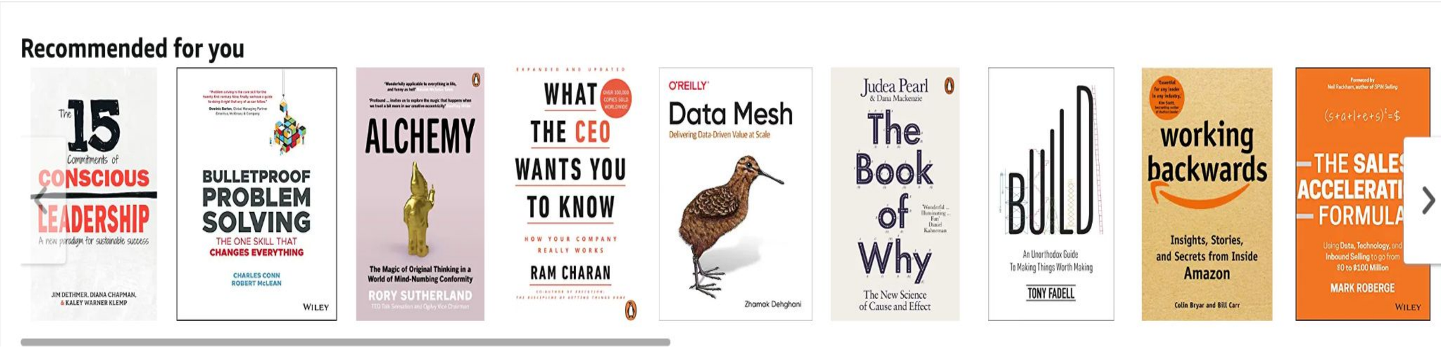
Recommender Systems

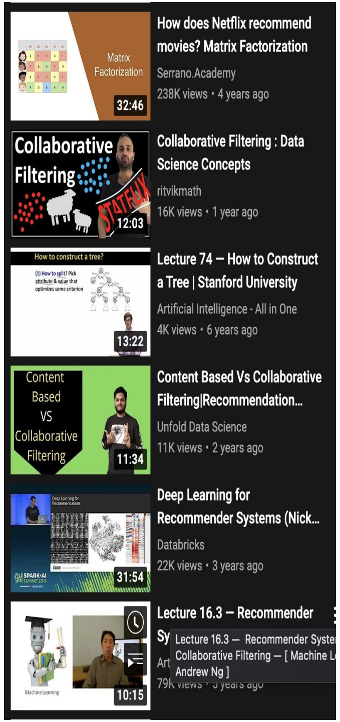
E0 259: Data Analytics
Lecture 1

Where Do You Encounter Recommender Systems?

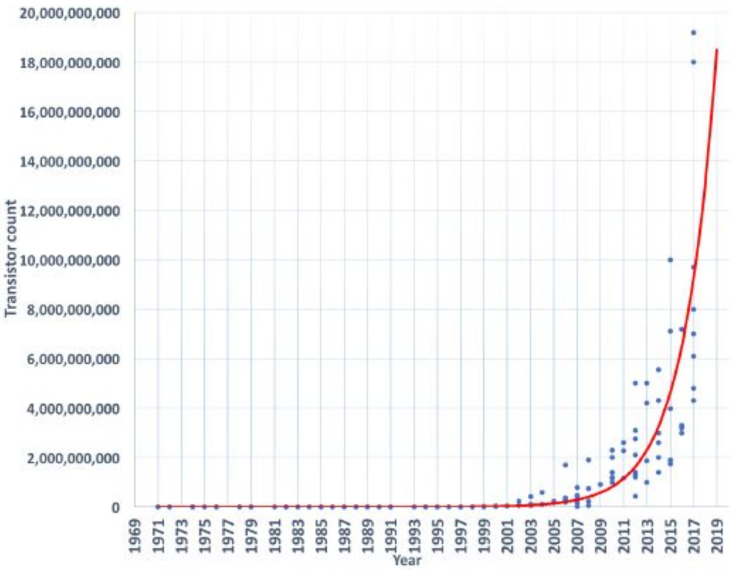




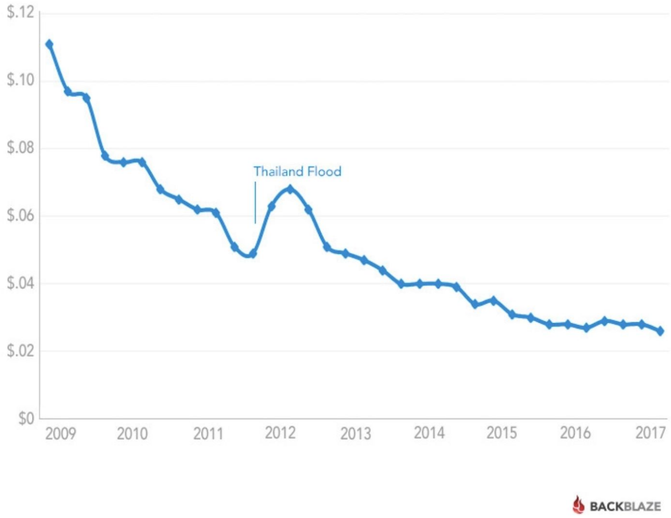




Economics of Content Distribution

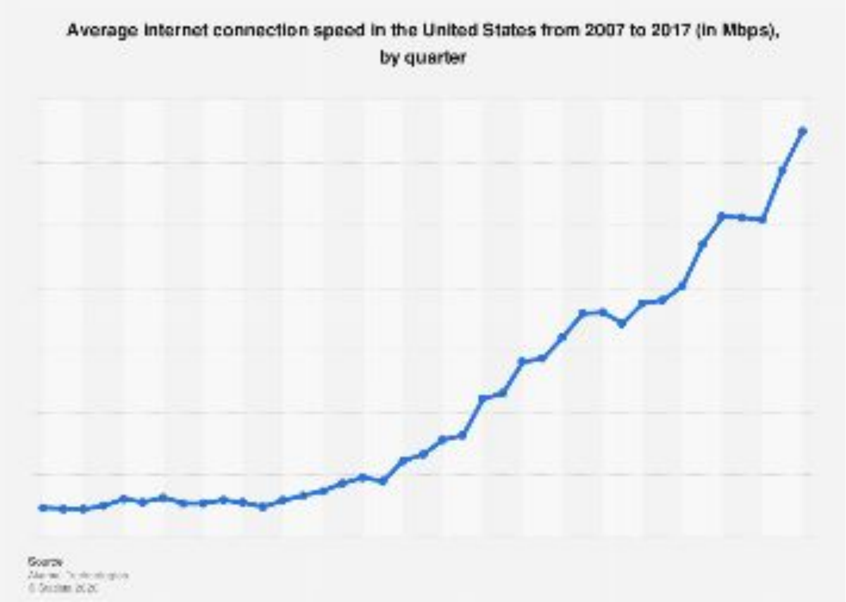


Compute Power: Moore's Law



Storage Costs: Cost/GB

Economics of Content Distribution



Internet Speed over time

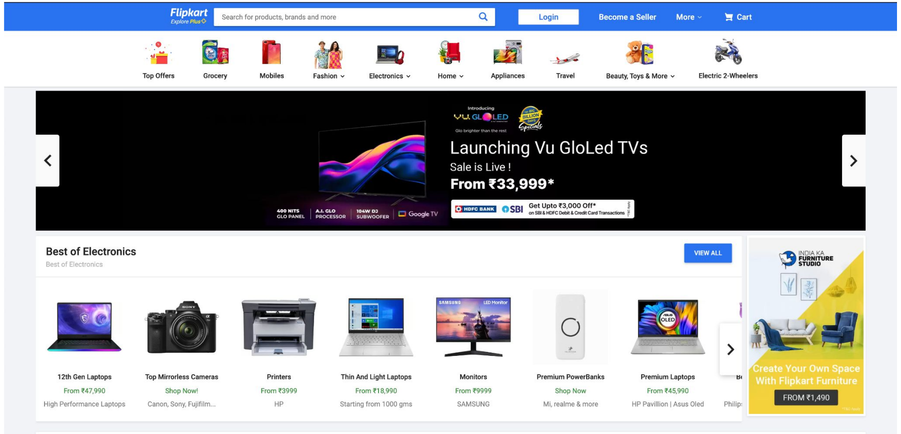
US Commercial Real Estate Prices over Time

Impact of Changing Distribution Economics

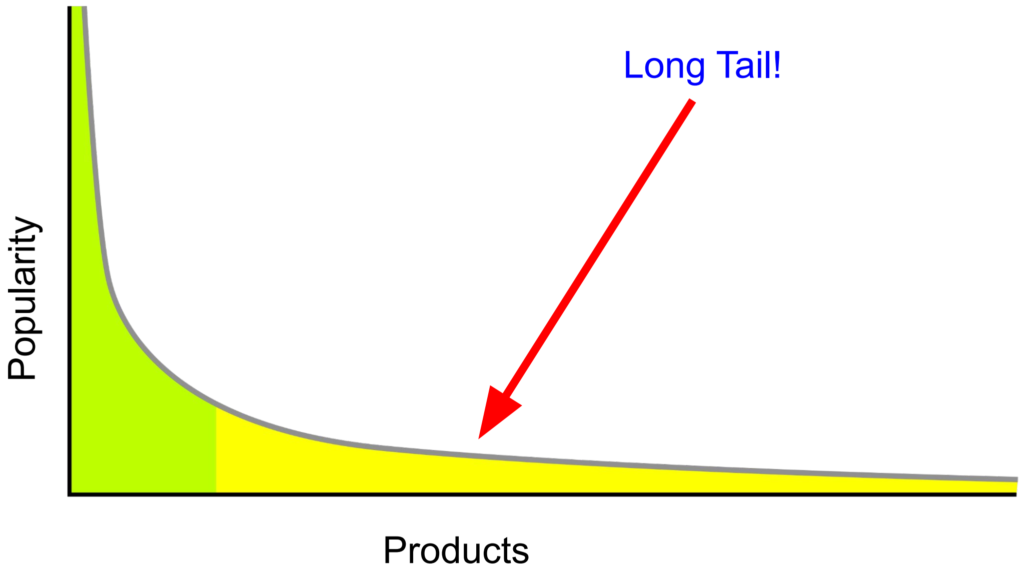
amazon books

Impact of Changing Distribution Economics

Impact of Changing Distribution Economics



What Products do you Stock in Your Store?

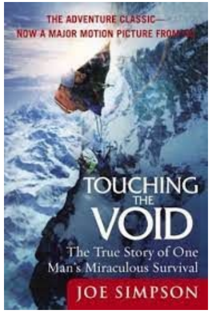


- **Physical Store:**
  - Expensive real estate
  - Limited shelf space
  - Store most popular items - stuff in green area!
- **Online Store:**
  - Cost of storage low
  - Cost of distribution low
  - Should we store stuff in yellow area also?

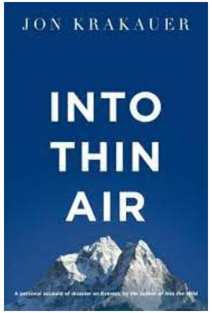
What is area of Green part versus Yellow part?
Ans: Significant fraction of total area!

The Long Tail

- Total demand for long tail items is comparable to the head or popular items!
  - If cost dynamics lowered, then long tail items are profitable to sell!
- Enter Amazon!
- How do users discover long tail items?
  - Recommendation engines

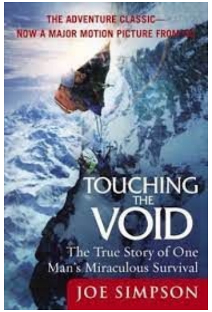


1988
Didn't do well



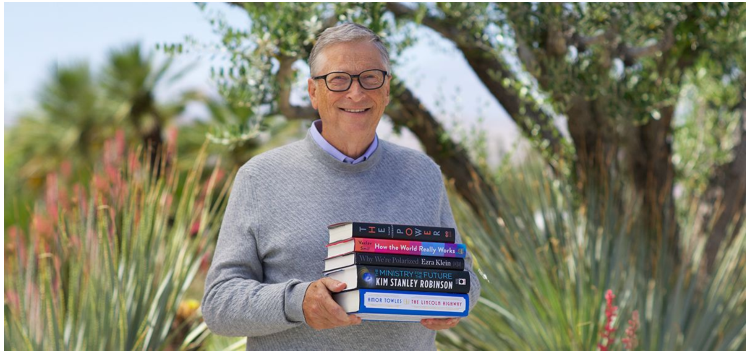
1997
Bestseller

Recommendation



Bestseller

How do we know what to recommend?

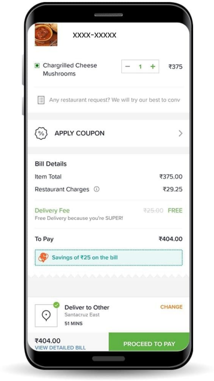


Expert Top Picks

Wisdom of Crowd

Neither Surfaces the Long Tail!

Get More Nuanced - Use Other Signals From Users

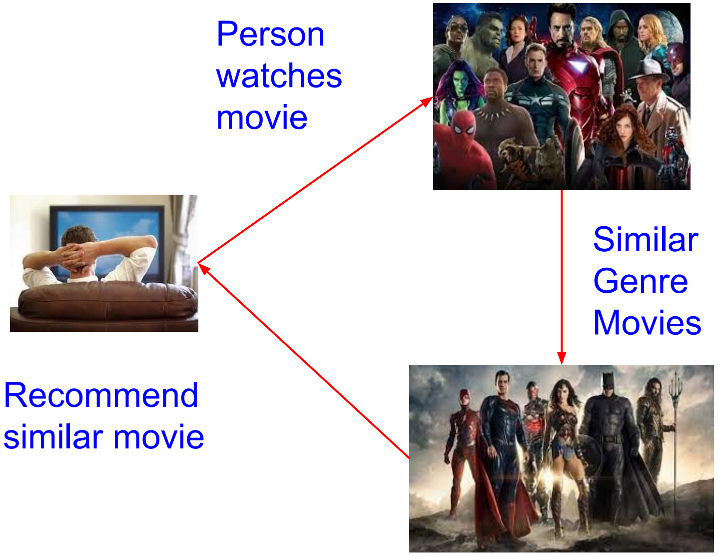


- Users Seldom Rate
- Users have different rating scales

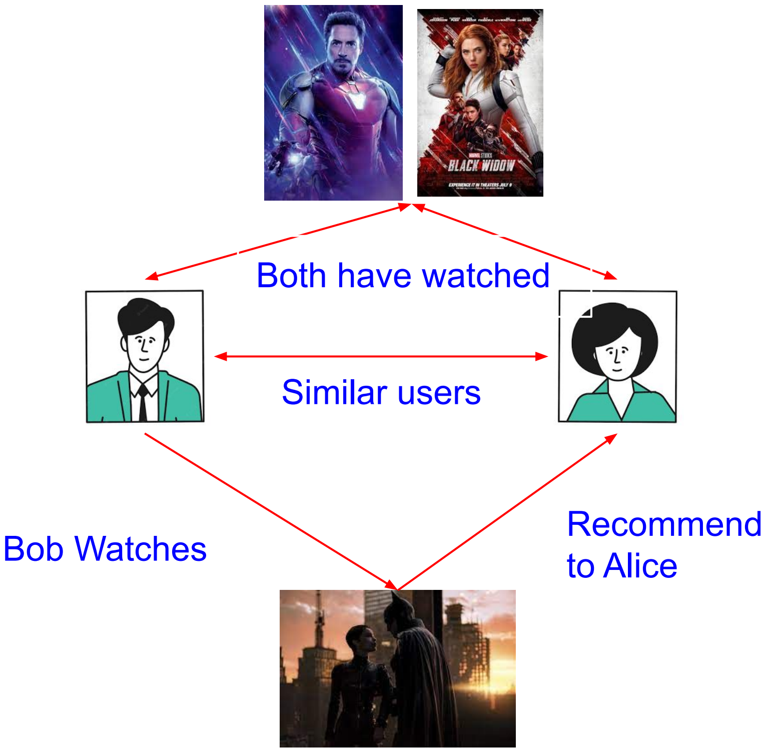
- Use Implicit signals - purchase, add to cart, clicks
- Very binary signal

Types of Recommendation Systems

Content Based Recommendation System



Collaborative Filtering



Content Based Recommendation Systems

- Recommend items with similar features to other items user has rated highly!

- **Examples:**
    - **Movies:** genre, actors, director
    - **Books:** genre, authors
    - **Music:** musicians, genre etc.

How Do We Go About This?

1. Define a universe of features {f} - e.g. genre, actors, director
2. For each item → create an item profile using the features of the item
3. For each user → based on items user has rated, create user profile.
4. Recommend items to the user that have features similar to what the user likes

Step 2: Item Profile - TF-IDF

- Assume you want to recommend books. Go beyond genre, author
- Use the content of the book!
- Features - every word possible is a feature.
- Use important words as distinguishing feature - ignore common words, *to, the* etc.

- How do we pick important words?

Step 2: Item Profile - TF-IDF

- TF = Term Frequency - how often does the word occur in the book?

$$f_{ij} = \text{number of times word } i \text{ occurs in book } j$$

$$TF_{ij} = \frac{f_{ij}}{max_k f_{kj}}$$

Normalize values

- IDF = Inverse Document Frequency - how rarely does the word appear across corpus?

$$n_i = \text{number of documents with word } i$$

$$N = \text{total number of documents}$$

$$IDF_i = log \frac{N}{n_i}$$

Log to bound range in large corpora

Step 2: Item Profile - TF-IDF

$$w_{ij} = TF_{ij} * IDF_i$$

TF-IDF score of term i in document j

$$\text{Feature vector of document } j \text{ is } \{w_{ij}\}_{i \in V}$$

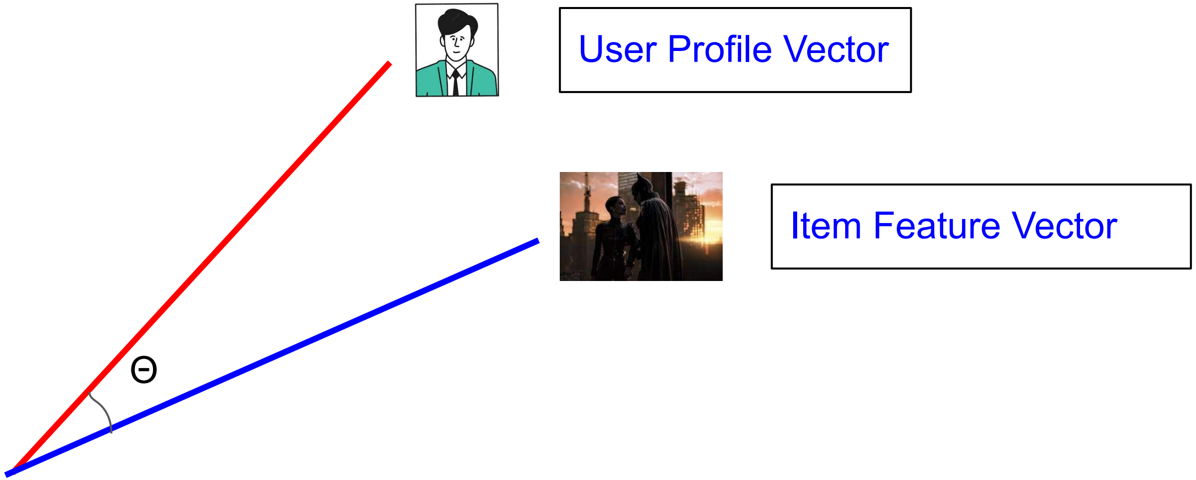
Can remove words with TF-IDF score less than a threshold to compress storage.

Step 3: Create User Profile

- Assume user has rated k items, $i_1, i_2, \ldots, i_k$
- Let $f(i_1), f(i_2), \ldots, f(i_k)$ be feature vectors of the k items
- Let $r(i_1), r(i_2), \ldots, r(i_k)$ be the ratings by the user for the k items.
- How do we construct a user profile from this? E.g. Weighted average of rating and feature vectors

$$\text{user profile} = \frac{\sum_{j \in k} r(i_j) f(i_j)}{\sum_{j \in k} r(i_j)}$$

Step 4: Recommend New Item to User



- Euclidean Distance? - If dimension is large, all vectors likely to be far apart

- Cosine similarity: $sim(\mathbf{u}, \mathbf{i}) = \frac{\mathbf{u}^{\mathbf{T}}\mathbf{i}}{||\mathbf{u}|| \, ||\mathbf{i}||}$

Pros and Cons

Pros:

- Can work even if user base is small
- Completely personalized
- No cold start - new unrated items can be recommended

Cons:

- Lot of feature engineering needed
- User lives in bubble - reinforcing loop gets created

Collaborative Filtering

- **Step 1:** For a user ***u*** find most similar users S

- **Step 2:** For any item *i*, unrated by user ***u***
  - estimate the rating that ***u*** would give *i*
  - Using the ratings given by users in S

Key Question: How do we find Similar Users?

- We first need a distance metric
  - How do we compare distances between vectors?
  - Any L_p norm:
    - L_2 = Euclidean distance, L_1 = absolute difference = $||\mathbf{x} - \mathbf{y}||$
    - Doesn't work well in high dimensional spaces
  - Jaccard Distance:
    -

$$sim(\mathbf{x}, \mathbf{y}) = \frac{|\mathbf{r_x} \cap \mathbf{r_y}|}{|\mathbf{r_x} \cup \mathbf{r_y}|}$$

Intersection of rating vectors over union

Example

Movie/User	Iron Man	Dr. Strange	Thor	Superman	Batman
Alice	4	4		2	
Bob	2		2	5	4
Charlie	5		4		

sim(Alice, Bob) = 2/5, sim(Alice, Charlie) = 4

Jaccard similarity ignores actual ratings! Looks only at set intersections:(

Cosine Similarity

Movie/User	Iron Man	Dr. Strange	Thor	Superman	Batman
Alice	4	4		2	
Bob	2		2	5	4
Charlie	5		4		

sim(Alice, Bob) = (8 + 10)/sqrt(34)*sqrt(49) = 0.44

sim(Alice, Charlie) = 20/sqrt(34)*sqrt(41) = 0.53

Alice more similar to Charlie than Bob!

Can we make this gap larger?

Do Pearson Correlation

Movie/User	Iron Man	Dr. Strange	Thor	Superman	Batman
Alice	4 - 10/3	4 - 10/3		2 - 10/3	
Bob	2 - 13/4		2 - 13/4	5 - 13/4	4 - 13/4
Charlie	5 - 9/2		4 - 9/2		

sim(Alice, Bob) = - 0.94
sim(Alice, Charlie) = = 0.32
Alice way more similar to Charlie than Bob now!

Rating Predictions

Now for user ***u***, item ***i,***

S(***u,i***) = most similar users to ***u*** who have rated ***i***

Prediction:

$$r_u(i) = \frac{\sum_{v \in S(u,i)} r_v(i) * sim(u,v)}{\sum_{v \in S(u,i)} sim(u,v)}$$

Predicted rating of item *i*, weighted average over similar users

Item-Item Collaborative Filtering

- So far: User-User collaborative filtering
  - Look at similar users to rate unrated item
- Item-Item:
  - Look at similar items to rate unrated item.
  - Same ideas as in user-user model, but along item axis

Item-Item Normalized Matrix

User/Movies	Alice	Bob	Charlie
Iron Man	4 - 11/3	2 - 11/3	5 - 11/3
Dr. Strange	4 - 4/1		
Thor		2 - 6/2	4 - 6/2
Superman	2 - 7/5	5 - 7/5	
Batman		4 - 4/1	

Similarity between i and j

$$r_u(i) = \frac{\sum_{j \in T(i,u)} r_u(j) * sim(i,j)}{\sum_{j \in T(i,u)} sim(i,j)}$$

Items most similar to i, rated by u

Item-Item or User-User

- In practice, item-item works better than user-user
- Items belong to a small set - e.g. genre
- Users can have varied tastes in different genres

Collaborative Filtering

- Popularity Bias
- Cold start problem - what to do with new items?
- Sparsity - find similar items which are rated
- No feature engineering needed.
- People use hybrid methods - collaborative + content based

Collaborative Filtering

- Dimensionality Reduction
  - SVD, latent factor models

Global Baseline

- Mean movie rating - 4.1
- Batman is 0.3 below average
- User typically rates movies 0.2 above average.
- Baseline for User’s rating for Batman = 4.1 - 0.3 + 0.2 = 4.0

Combining Global Baseline and CF

- User's global baseline for Batman is 4.0
- Related movie Superman that User has rated is 2 points below his average rating.
- Predicted rating for Batman = 4 - 2 = 2

$$r_{ui} = b_{ui} + \frac{\sum_{j \in T(i,u)} (r_u(j) - b_{xj}) * sim(i,j)}{\sum_{j \in T(i,u)} sim(i,j)}$$

$$\boxed{b_{ui} = \mu + b_x + b_i}$$

- μ : global base line rating for all items.
- b_u : user u's average rating across all items
- b_i : item i's average rating across all users