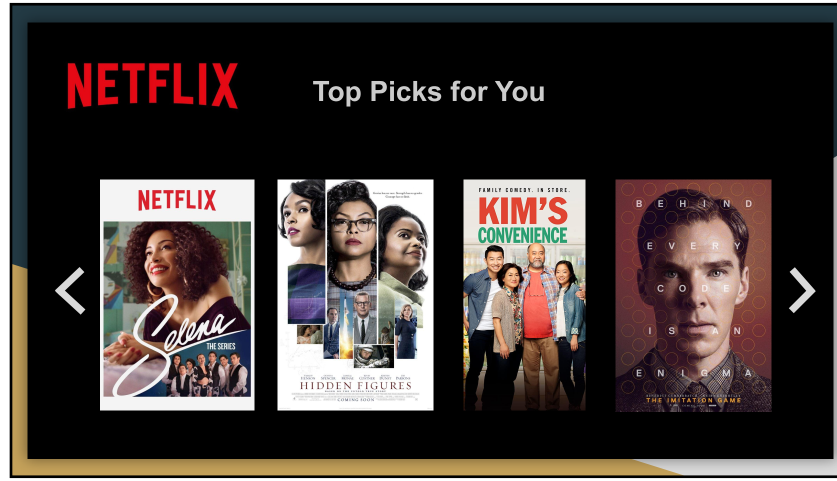
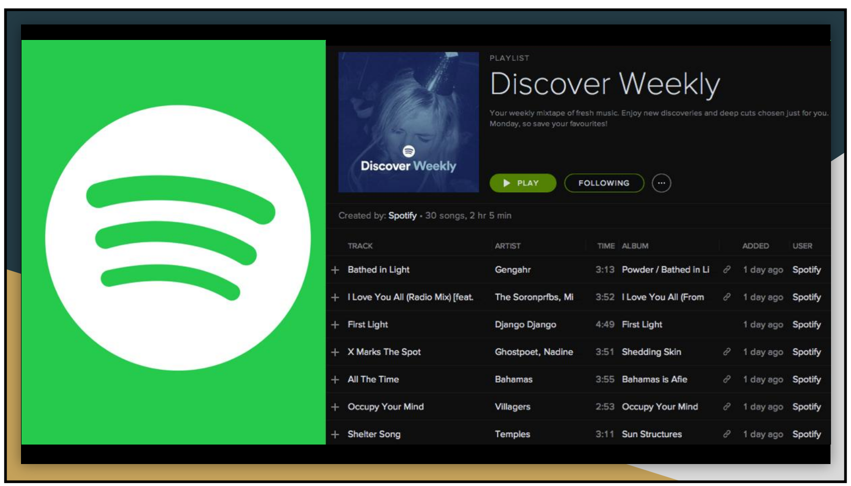
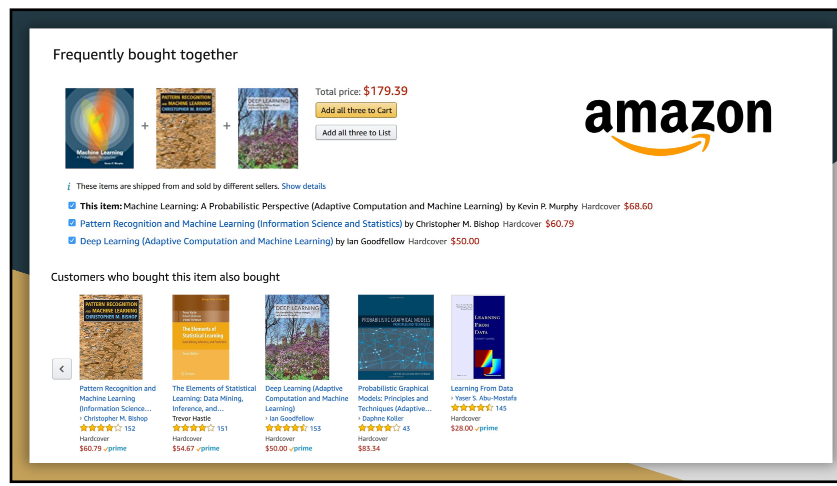
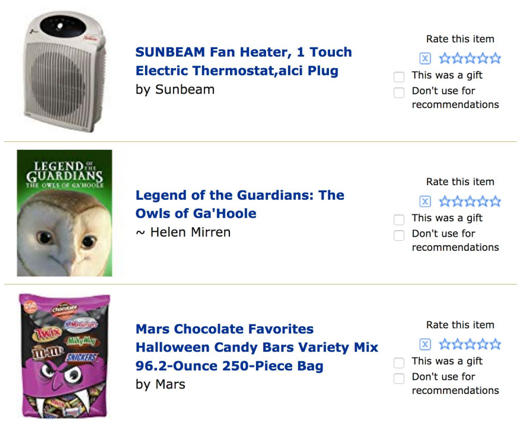
Recommender Systems







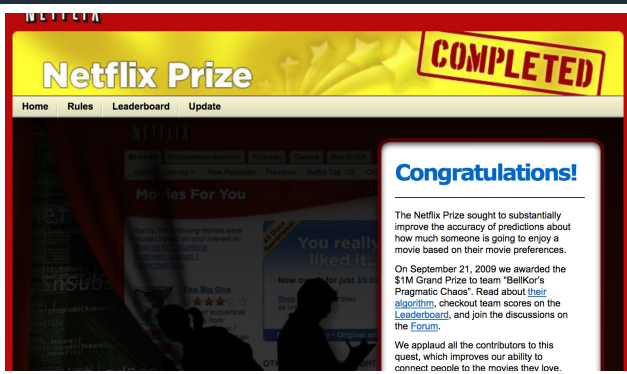


amazon

How might my ratings change my recommendations?

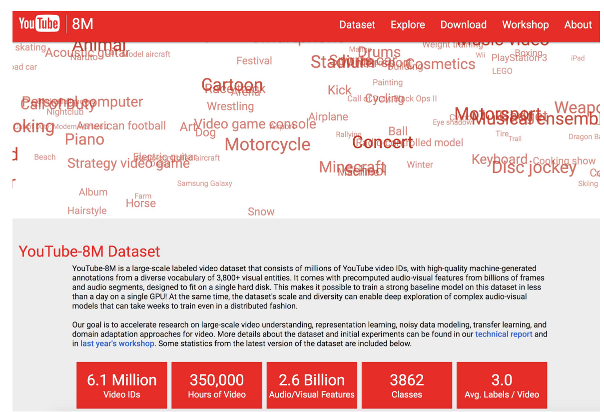
Goals of Recommender Systems

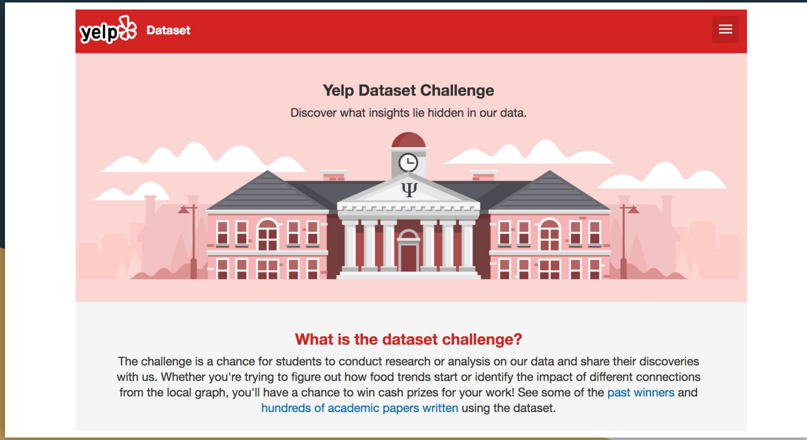
- Show content that we're interested in
- Suggest new content that would interest us
- Suggest new content that is generally popular
- Adjust recommendations based on our feedback



$1M winning algorithm not actually used by Netflix

Researchers were able to de-anonymize data by comparing with IMDB ratings, resulting in a lawsuit





Recommender Systems

Collaborative Filtering

- What makes two (Amazon) *users* similar?
  - Purchased the same set of items
  - Liked and disliked the same set of items
- What makes two *items* similar?
  - The same set of users purchased/liked them
  - Their titles, description, prices, other metadata

Content Based Recommendation

Collaborative Filtering

Create a *user-item* matrix

Sohie	5		2	
Brian				4
Christine		1	4	5
Orit	3	5	3	
Catherine	4	4		4

Similarity: Jaccard

- Measure similarity between a pair of user vectors (or a pair of item vectors)

Problem: does not work for non-binary vectors

$U_A = [1, 0, 1, 0]$

$U_B = [0, 0, 1, 0]$

$$Jaccard(U_A, U_B) = \frac{|U_A \cap U_B|}{|U_A \cup U_B|}$$

When is result 0? When is it 1?

Similarity: Cosine

- Measure similarity between a pair of user vectors (or a pair of item vectors)

$U_A = [0, 5, 2, 0]$

$U_B = [1, 0, 4, 2]$

$$CosineSim(U_A, U_B) = \frac{U_A \cdot U_B}{||U_A||\ ||U_B||}$$

When is result 0? When is it 1?

User-Based Collaborative Filtering

Task: predict rating on new user-item entry in matrix: U_A, I_P

- Among *users* that have rated I_P, select a set S_K of the K most similar *users* to U_A
- Predicted rating for U_A, I_P is average rating of I_P from *users* in S_K:

$$R(U_A, I_P) = \frac{\sum_{U_B \in S_K} R(U_B, I_P)}{K}$$

User-Based Collaborative Filtering

Task: predict rating on new user-item entry in matrix: U_A, I_P

- Some users have a tendency to be more or less generous
- Use deviation from a user's average rating, rather than a user's absolute rating

$$R(U_A, I_P) = \frac{\sum_{U_B \in S_K} R(U_B, I_P) - \mathrm{mean}(U_B)}{K}$$

Item-Based Collaborative Filtering

Task: predict rating on new user-item entry in matrix: U_A, I_P

- Among *items* that have been rated by U_A, select a set S_K of the K most similar *items* to I_P
- Predicted rating for U_A, I_P is average rating of U_A from *items* in S_K:

$$R(U_A, I_P) = \frac{\sum_{I_Q \in S_K} R(U_A, I_Q)}{K}$$

Item-Based Collaborative Filtering

Task: predict rating on new user-item entry in matrix: U_A, I_P

- Among all *items* that have been rated by U_A, compute weighted average of U_A's ratings (weighted by similarity to I_P)

$$R(U_A, I_P) = \frac{\sum_{I_Q \in S_{all}} \mathrm{Sim}(I_P, I_Q) R(U_A, I_Q)}{\sum_{I_Q \in S_{all}} \mathrm{Sim}(I_P, I_Q)}$$

Weighted Average

Mean is 73%

Weighted Mean is 80%

Compute final score in some class:

	Score	Weight
Class participation	60%	50 points
Homework	95%	200 points
Midterm Exam	50%	100 points
Final Exam	87%	150 points

$$\text{Weighted Mean} = \frac{\sum \text{Weight} \cdot \text{Score}}{\sum \text{Weight}} = \frac{50*0.60 + 200*0.95 + 100*0.50 + 150*0.87}{50 + 200 + 100 + 150}$$

Problems with Collaborative Filtering?

- If user-item matrix is too sparse, may not be useful
- “Cold-start problem”: how to handle new users and items?
- Won’t encourage diverse results (echo chamber effect)

Content-Based Recommendations: Approach 1

- Define similarity between users (or similarity between items) in terms of content features, not rating patterns
  - ➢ Examples of item features: restaurant cuisine type, director or actors in movie, product details
  - ➢ Examples of user features: demographic information
- Apply same methods as for collaborative filtering

Content-Based Recommendations: Approach 2

- Featurize users and items under the *same* set of features
  - ➢ Features: words
    - ○ user feature values = word counts in reviews
    - ○ item feature values = word counts in descriptions
  - ➢ Features: demographics
    - ○ user feature values = demographic info
    - ○ item feature values = target demographics
- Compute similarity between a given user and item

Featurizing Text

- Bag of words: tokenizing, counting, tf-idf weighting

	bland	but	fast	food	good	no	parking	service
Fast service but bland food.	1	1	1	1	0	0	0	1
Good fast food.	0	0	1	1	1	0	0	0
No service, no parking, no good.	0	0	0	0	1	3	1	1

bland	but	fast	food	good	no	parking	service
0.5	0.5	0.4	0.4	0	0	0	0.4
0	0	0.6	0.6	0.6	0	0	0
0	0	0	0	0.2	0.9	0.3	0.2

- N-Grams

Fast service but bland food.
Good fast food.
No service, no parking, no good.

service no
fast food
good fast
fast service
parking no
but bland
service but
bland food
no service
no good
no parking

Evaluation

- **Task Type A:** Given test set of (user, item) pairs, predict ratings
  - ➢ Raw accuracy, e.g., percentage of ratings predicted exactly Too strict!
  - ➢ Root mean squared error (RMSE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

- **Task Type B:** Given test set of users, predict set of items to recommend
  - ➢ Precision, Recall, F1 Score

TP: Recommended items user actually buys
FP: Recommended items user does not buy
TN: Items not recommended and user does not buy
FN: Items not recommended and user buys

Vectorization (Array Programming)

- Many scientific and numerical computing libraries, such as NumPy in Python, provide *vectorized* operations, i.e., operations that can be applied to an entire array (matrix)

```
np.median(a)    a[a>10]
np.random.randint(...)
np.dot(a,b)
a**2    np.sum(a)    np.mean(a)    np.ones(...)
```

- Whenever possible, it is usually a good idea to use *vectorization* rather than looping through an array and applying an operation to each element

Overview