W10D1

Recommender Engines I

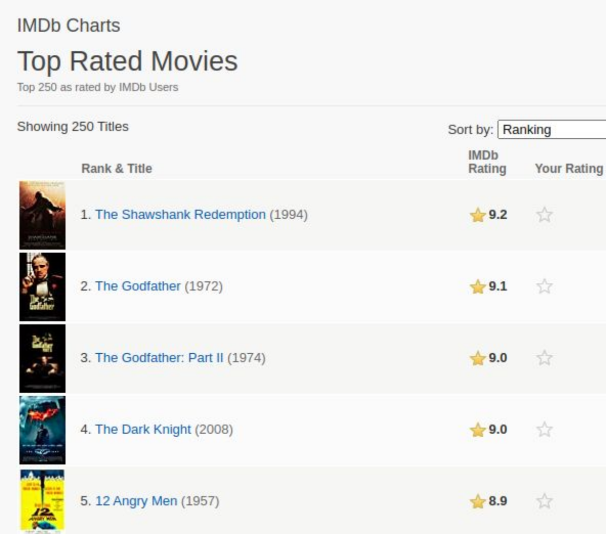
Instructor: Eric Elmoznino

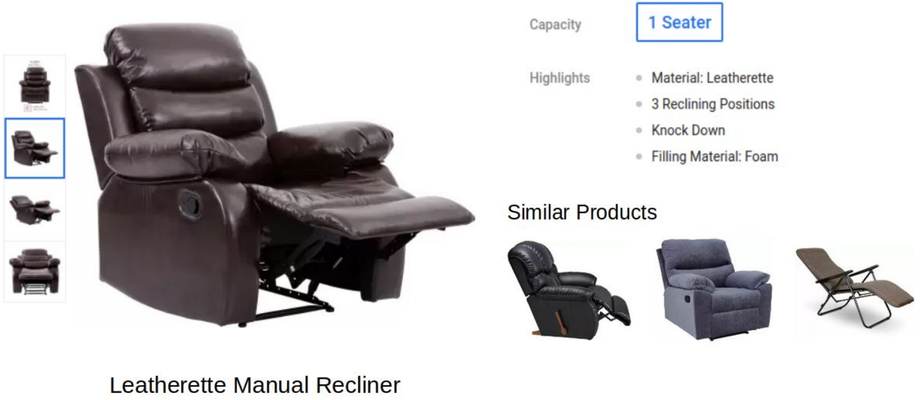
Outline for today

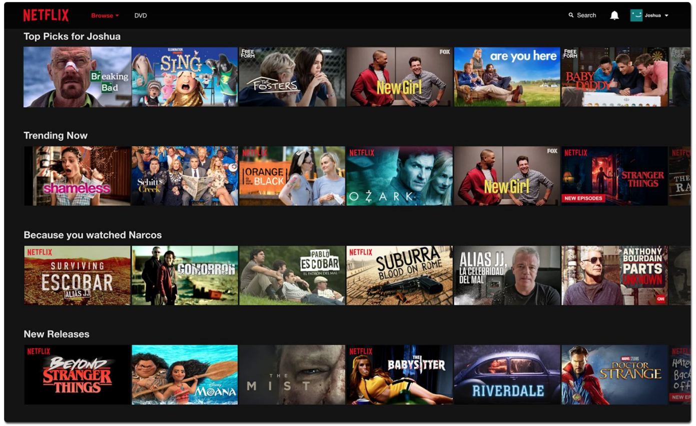
- Overview
- Content-based vs. collaborative
- Content-based recommender
    - Step 1: Define item features
    - Step 2: Define a distance metric
    - Step 3: Recommend similar items
- Activity: movie feature engineering
- Case study: co-occurrence feature engineering
- Demo

Overview

Examples

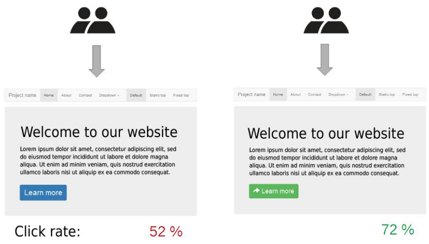

IMDb Charts
Top Rated Movies
Top 250 as rated by IMDb Users
Showing 250 Titles
Sort by: Ranking
Rank & Title
IMDb Rating
Your Rating
1. The Shawshank Redemption (1994) 9.2
2. The Godfather (1972) 9.1
3. The Godfather: Part II (1974) 9.0
4. The Dark Knight (2008) 9.0
5. 12 Angry Men (1957) 8.9


Capacity
1 Seater
Highlights
Material: Leatherette
3 Reclining Positions
Knock Down
Filling Material: Foam
Similar Products
Leatherette Manual Recliner


NETFLIX
Top Picks for Joshua
Trending Now
Because you watched Narcos
New Releases

Recommender engines

- **Users**: purchaser of Amazon products, Netflix binge-watcher, social media subscriber, etc.
- **Items**: Amazon products, Netflix shows, social media posts, etc.
- **Goal**: optimize some business (e.g. clicks, longer screen time, revenue, conversion rate)
    - Algorithms will often be compared to baselines and alternatives using A/B experiments



Content-based vs. collaborative

Content-based vs. collaborative

- **Content-based recommender**: use knowledge of each item to recommend a similar one (item-based recommendation)
  - Example: customer looking at a computer with 8GB RAM, 125 GB HDD, 6 hour battery life
- **Collaborative filtering**: use knowledge of a user's past purchases/selections to recommend what similar users did (user-based recommendation).
  - Example: Netflix recommending me shows based on what others who have watched similar shows to me have also watched

Content-based vs. collaborative

Content-based recommender

Advantages:
- Works without user data

Disadvantages:
- Requires descriptive data of products
- Doesn’t expand user interests

Collaborative filtering

Advantages:
- Works without descriptive product data

Disadvantages:
- Requires user data
- Difficult to make recommendations to new users (“cold start” problem)

Content-based recommender

Step 1: define item features

Movie	Action	Comedy	Romance	Drama	Runtime	Actor 1	Actor 2	Actor 3	...
Movie 1	1	0	0	0	123	1	0	1	...
Movie 2	0	0	1	0	96	0	0	1	...
Movie 3	0	1	0	0	89	0	0	0	...
...	...	...	...	...	...	...	...	...	...

Step 1: define item features — feature engineering

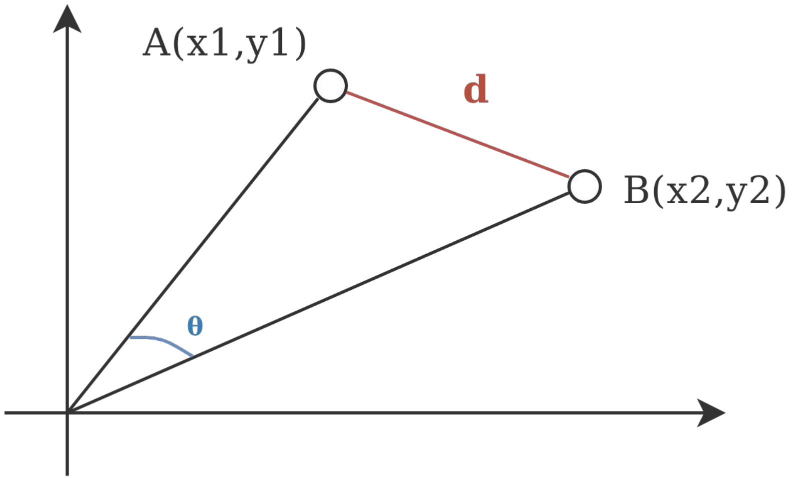
Movie	**Action**	**Runtime**	**Description word soup**	...
Movie 1	1	123	jealousy toy boy tomhanks timallen donrickles ...	...
Movie 2	0	96	boardgame disappearance basedonchildren'sbook ...	...
...	...	...	...	...

Step 2: define a distance metric

Euclidean distance

- Geometric distance between two points
- Considers magnitude and direction of vectors
- Range: (0, ∞)
- Interpretation: lower is more similar

```
def euclidean_distance(x, y):
  return np.sqrt(np.sum((x - y) ** 2))
```



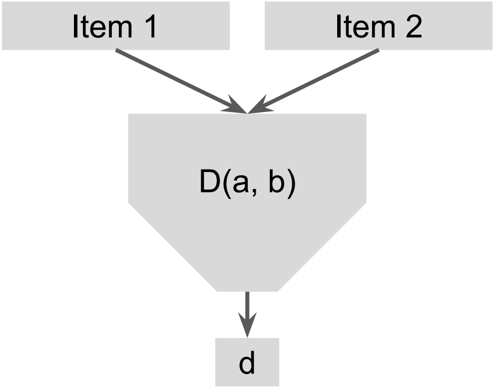
Cosine similarity

- Angular distance between two points
- Considers direction of vectors
- Range: (-1, 1)
- Interpretation: higher is more similar

```
def cosine_similarity(x, y):
  return np.dot(x, y) / (np.sqrt(np.dot(x, x)) * np.sqrt(np.dot(y, y)))
```

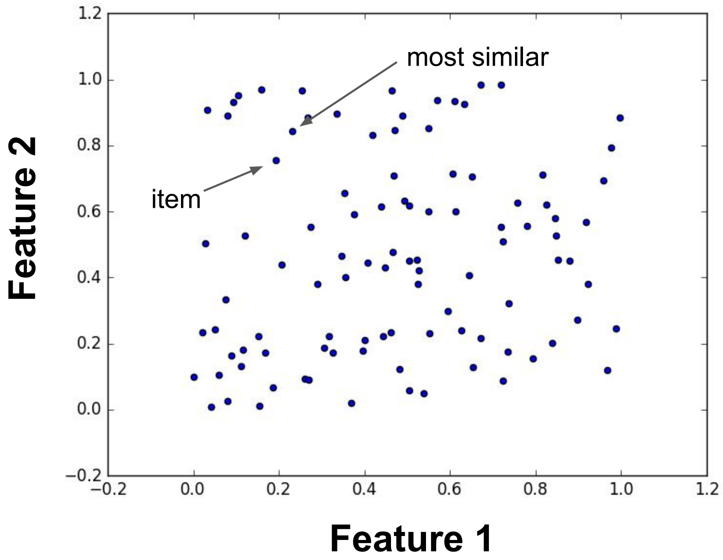
Step 2: *learn* a distance metric?

- Instead of picking a distance metric that may not work well for our feature space, we can also *learn* a distance metric. Called “distance metric learning”
- Function that takes 2 sets of features and outputs a positive number
- Has the potential to:
  - Learn relative feature importance
  - Account for feature interactions
  - Reduce the load on feature engineering
- Challenges: how to train?
- Comprehensive overview and technical tutorial



Step 3: recommend items

- Based on the user’s items, (e.g. currently viewing, cart, history), recommend items with the *highest similarity* (i.e. *lowest distance*)

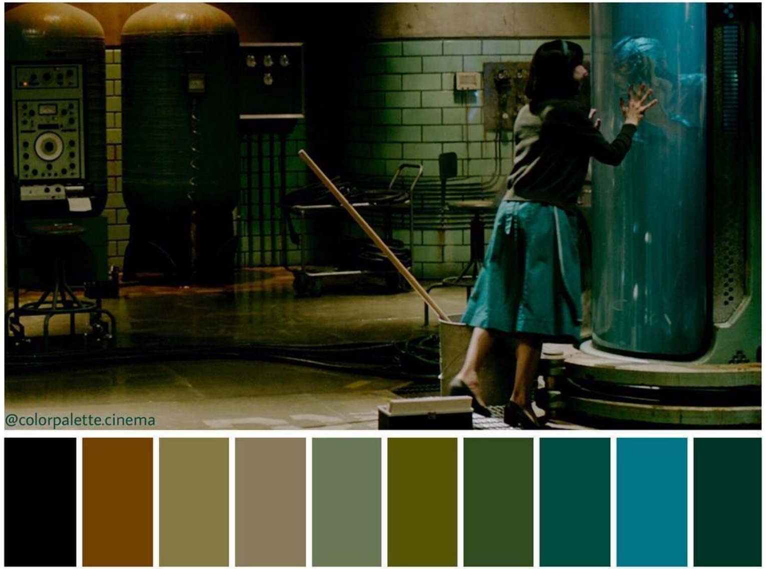


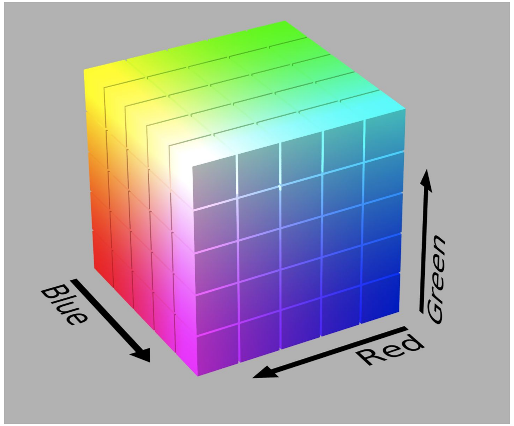
Activity: movie feature engineering

Activity: movie feature engineering

- Imagine you are a data scientist at Netflix trying to come up with movie features for a content-based recommender
- Assume access to video, audio, dialogue, genre, description, twitter, etc.
- In groups, come up with 1 or more features that you could use
  - What is the feature?
  - Why would this feature be useful? What relevant information would it carry?
  - How would you engineer (or learn) the feature from the raw data? (i.e. numeric representation)
  - What would be the challenges, if any?
- 15 minutes in groups, then verbally tell us your ideas

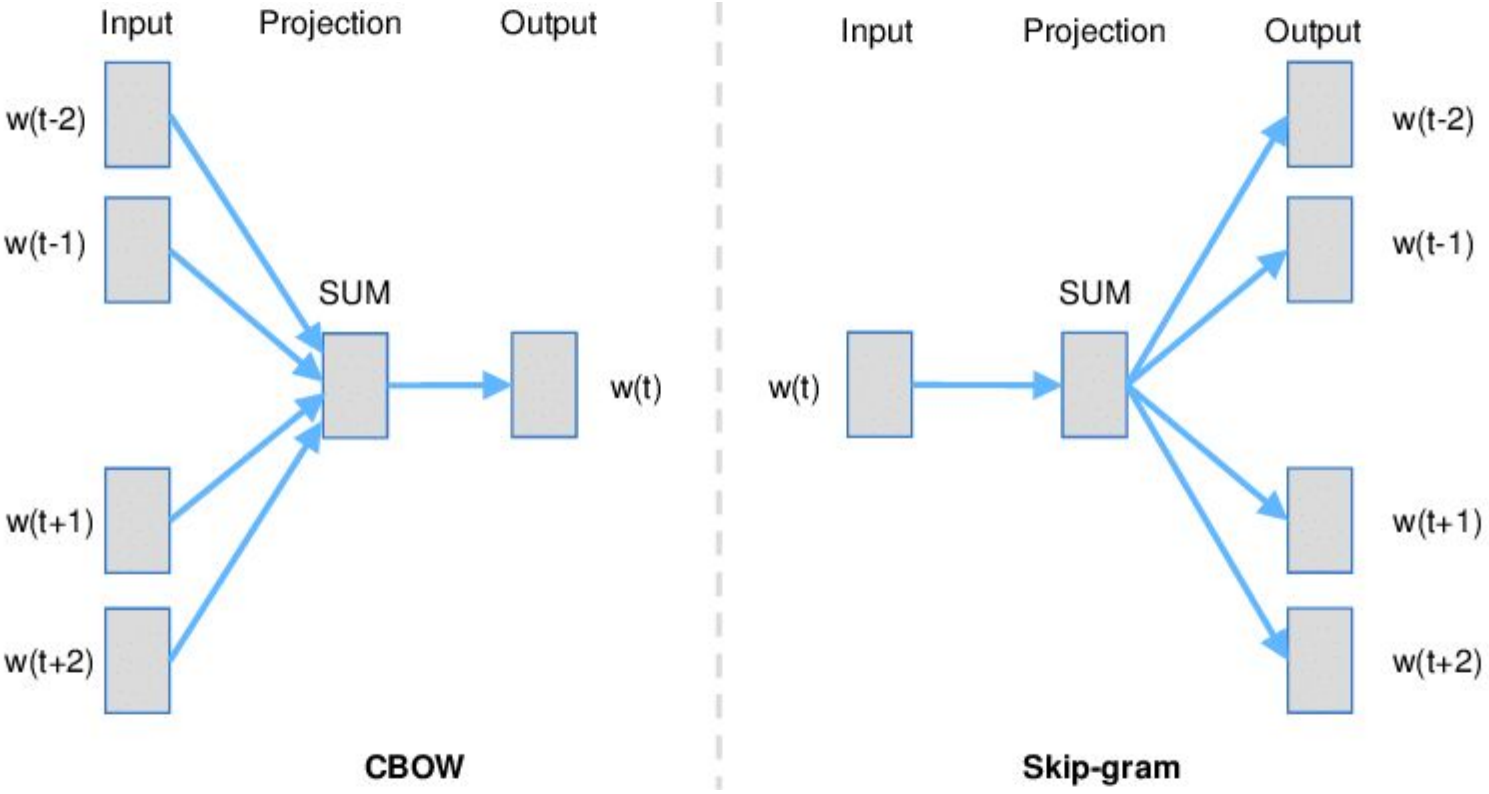
Example: colour palette



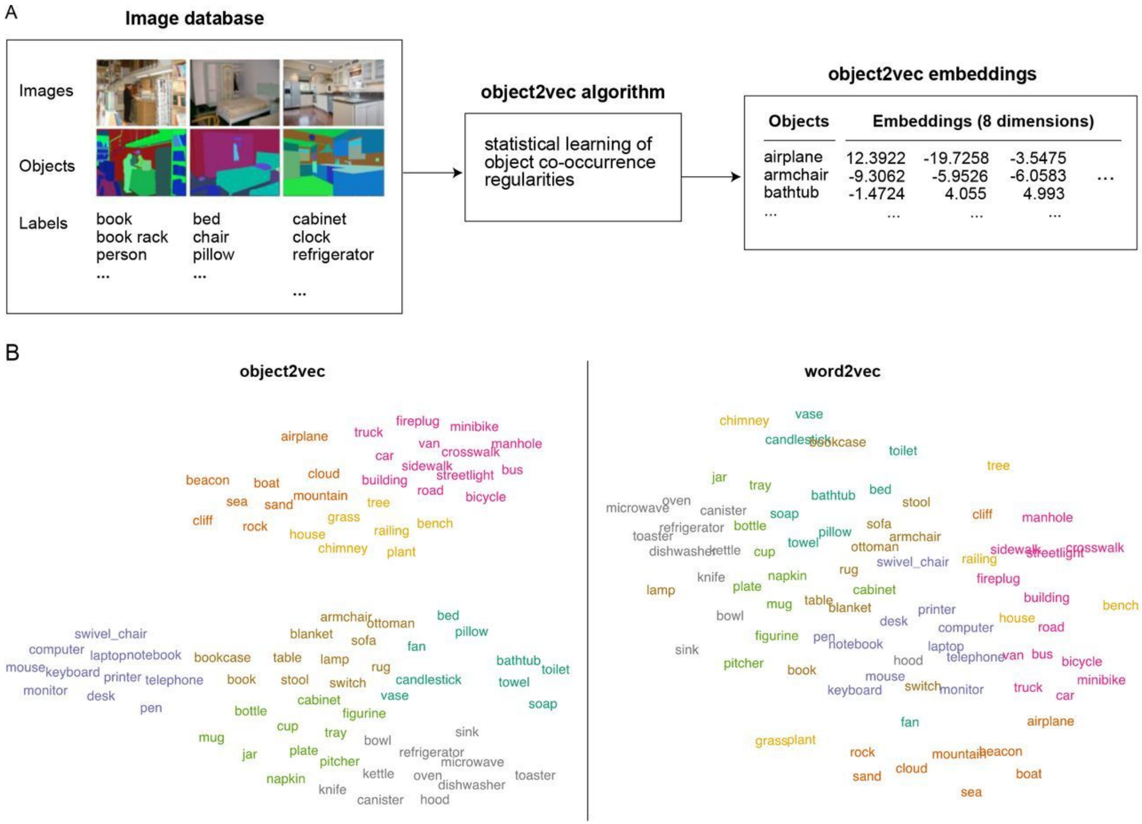


Case study:
co-occurrence feature engineering

Word2Vec


Input
Projection
Output
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
CBOW
Input
Projection
Output
w(t-2)
w(t-1)
SUM
w(t)
w(t+1)
w(t+2)
Skip-gram

Word2Vec for arbitrary data



Word2Vec for content recommendation features

Demo