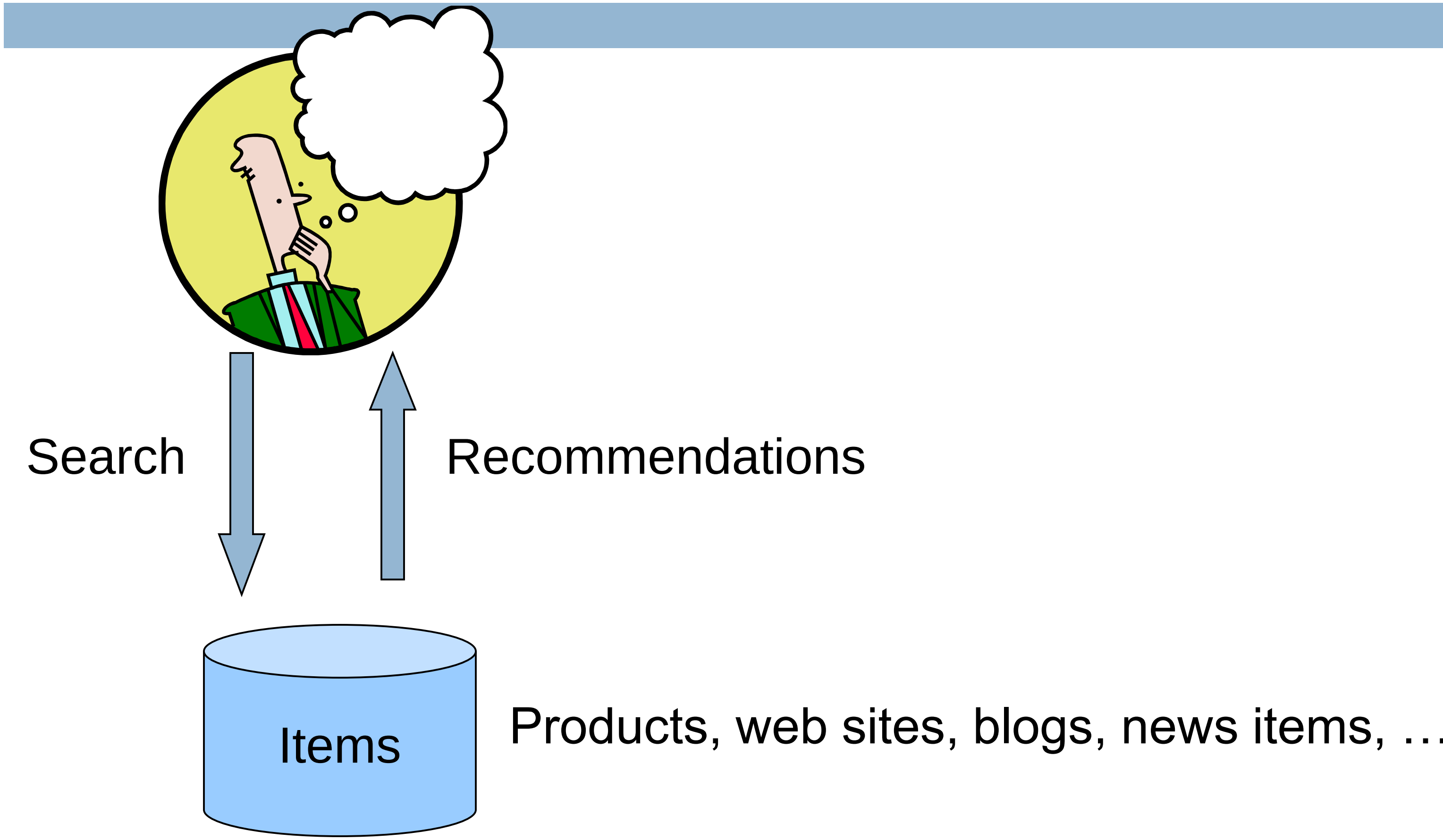
RECOMMENDER SYSTEMS

Recommendations



From scarcity to abundance

- Shelf space is a scarce commodity for traditional retailers
  - Also: TV networks, movie theaters,…
- The web enables near-zero-cost dissemination of information about products
  - From scarcity to abundance
- More choice necessitates better filters
  - Recommendation engines

Recommendation Types

- Editorial
- Simple aggregates
  - Top 10, Most Popular, Recent Uploads
- Tailored to individual users
  - Amazon, Netflix, …

Formal Model

- C = set of Customers
- *S* = set of Items
- Utility function *u*: C X *S* -> *R*
  - *R* = set of ratings
  - *R* is a totally ordered set
  - e.g., 0-5 stars, real number in [0,1]

Utility Matrix

	King Kong	LOTR	Matrix	Nacho Libre
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

Key Problems

- Gathering "known" ratings for matrix
- Extrapolate unknown ratings from known ratings
  - Mainly interested in high unknown ratings
- Evaluating extrapolation methods

Gathering Ratings

- Explicit
  - Ask people to rate items
  - Does not work well in practice – people cannot be bothered
- Implicit
  - Learn ratings from user actions
  - e.g., purchase implies high rating
  - What about low ratings?

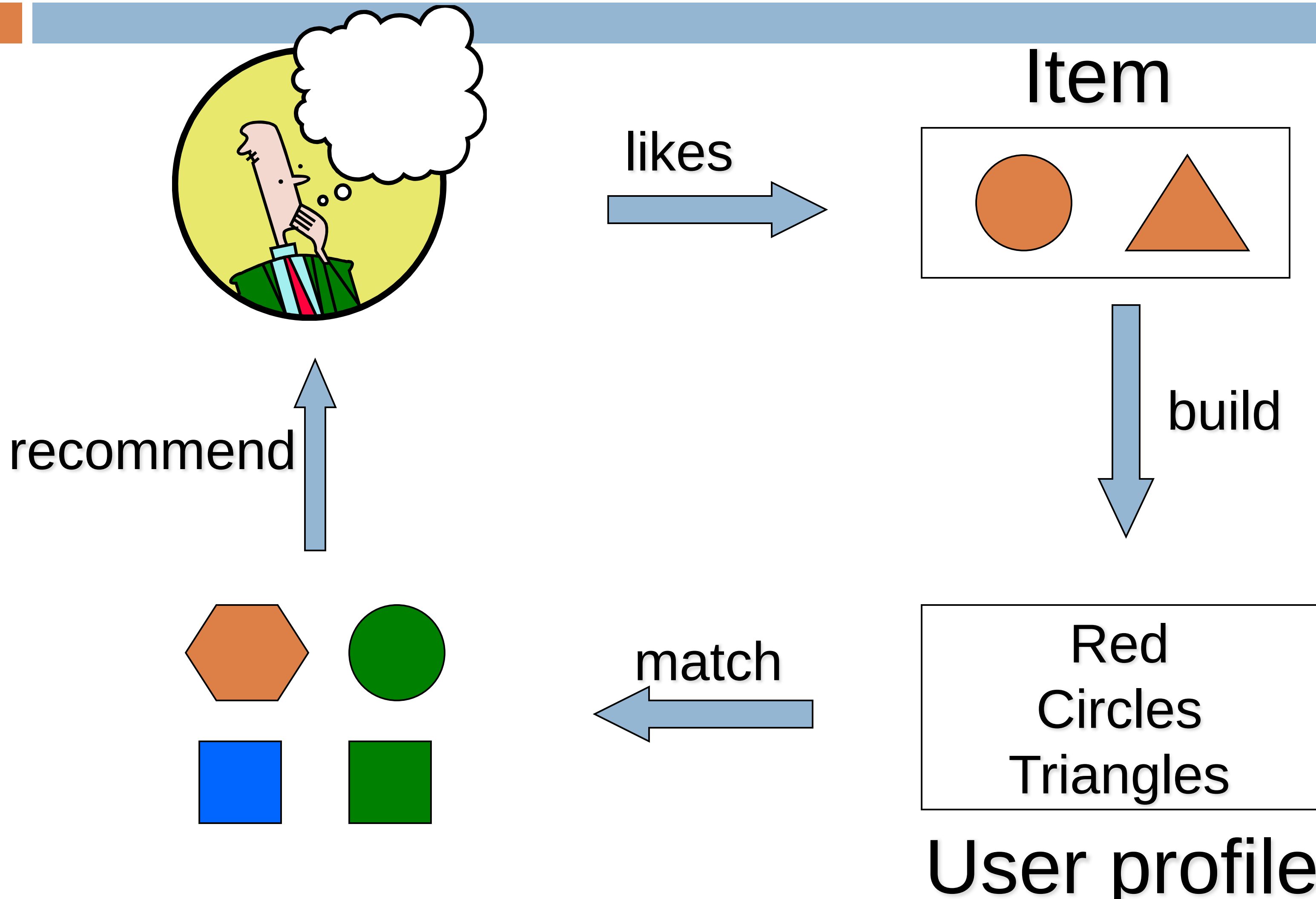
Extrapolating Utilities

- Key problem: matrix U is sparse
  - most people have not rated most items
- Three approaches
  - Content-based
  - Collaborative
  - Hybrid

Content-based recommendations

- Main idea: recommend items to customer C similar to previous items rated highly by C
- Movie recommendations
  - recommend movies with same actor(s), director, genre, …
- Websites, blogs, news
  - recommend other sites with "similar" content

Plan of action



Item Profiles

- For each item, create an item profile
- Profile is a set of features
  - movies: author, title, actor, director,…
  - text: set of "important" words in document
- How to pick important words?
  - Usual heuristic is TF.IDF (Term Frequency times Inverse Doc Frequency)

TF.IDF

f_{ij} = frequency of term t_i in document d_j

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

n_i = number of docs that mention term i

N = total number of docs

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

TF.IDF score $w_{ij} = Tf_{ij} \text{ X } IDF_i$

Doc profile = set of words with highest TF.IDF scores, together with their scores

User profiles and prediction

- User profile possibilities:
  - Weighted average of rated item profiles
  - Variation: weight by difference from average rating for item
  - …
- Prediction heuristic
  - Given user profile **c** and item profile **s**, estimate $u(\mathbf{c},\mathbf{s}) = \cos(\mathbf{c},\mathbf{s}) = \mathbf{c}.\mathbf{s}/(|\mathbf{c}||\mathbf{s}|)$
  - Need efficient method to find items with high utility: later

Machine Learning Approaches

- Various machine learning techniques are feasible
  - E.g.,decision trees, and neural networks.
- These methods use models learned from the underlying data
  - For example, based on a set of Web pages that were rated as “relevant” or “irrelevant” by the user, the machine learning classifier can be used to classify unrated Web pages.

Advantages of Content-Based Approach

- No need for data on other users
  - No cold-start or sparsity problems
- Able to recommend to users with unique tastes
- Able to recommend new and unpopular items
- Can provide explanations of recommended items by listing content-features that caused an item to be recommended

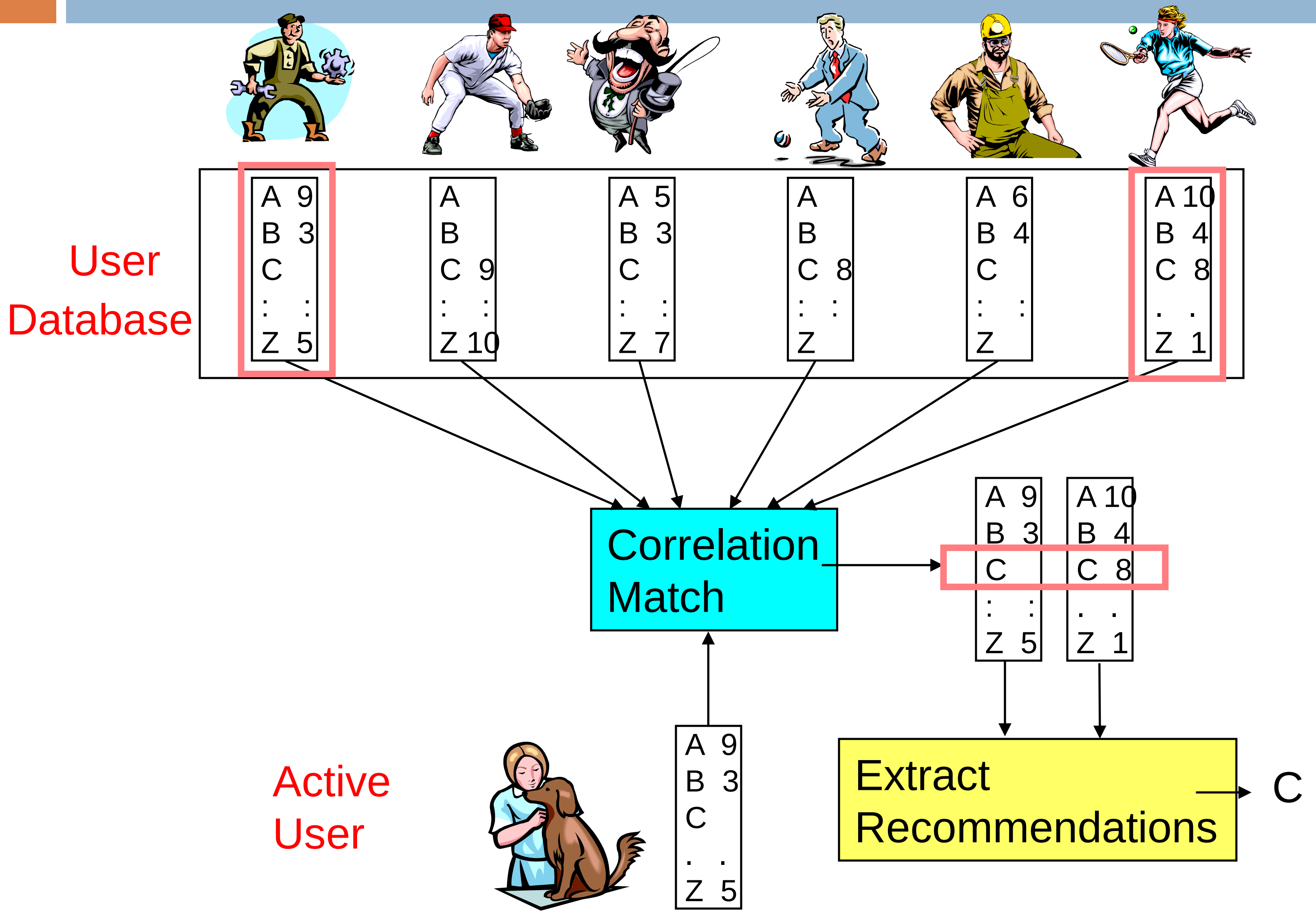
Disadvantages of Content-Based Method

- Requires content that can be encoded as meaningful features
- Users' tastes must be represented as a learnable function of these content features
- Unable to exploit quality judgments of other users
  - Unless these are somehow included in the content features

Collaborative Filtering

- Maintain a database of many users' ratings of a variety of items
- For a given user, find other similar users whose ratings strongly correlate with the current user
- Recommend items rated highly by these similar users, but not rated by the current user

Collaborative Filtering



Collaborative Filtering Method

- Weight all users with respect to similarity with the active user
- Select a subset of the users (*neighbors*) to serve as predictors
- Normalize ratings and compute a prediction from a weighted combination of the selected neighbors' ratings
- Present items with highest predicted ratings as recommendations

Similarity Weighting

- use Pearson correlation coefficient between ratings for active user *a* and another user *u*

$$c_{a,u} = \frac{\mathrm{cov}(r_a, r_u)}{\sigma_{r_a}\sigma_{r_u}}$$

r_a and r_u : rating vectors for the m items rated by **both** *a* and *u*

Significance Weighting

- not to trust correlations based on very few co-rated items
- Include *significance weights*, $s_{a,u}$, based on number of co-rated items, *m*

$$w_{a,u} = s_{a,u} c_{a,u}$$

$$s_{a,u} = \begin{Bmatrix} 1 \text{if } m > 50 \\ \dfrac{m}{50} \text{ if } m \leq 50 \end{Bmatrix}$$

Neighbor Selection

- For a given active user *a*, select correlated users to serve as source of predictions
  - use the most similar *n* users, *u*, based on similarity weights $w_{a,u}$
- Alternatively, include all users whose similarity weight is above a given threshold

Rating Prediction

- Predict a rating, $p_{a,i}$, for each item i, for active user, a, by using the n selected neighbor users, $u \in \{1,2,\ldots n\}$
- To account for users different ratings levels, base predictions on *differences* from a user's *average* rating
- Weight users' ratings contribution by their similarity to the active user

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n} w_{a,u}(r_{u,i} - \bar{r}_u)}{\sum_{u=1}^{n} w_{a,u}}$$

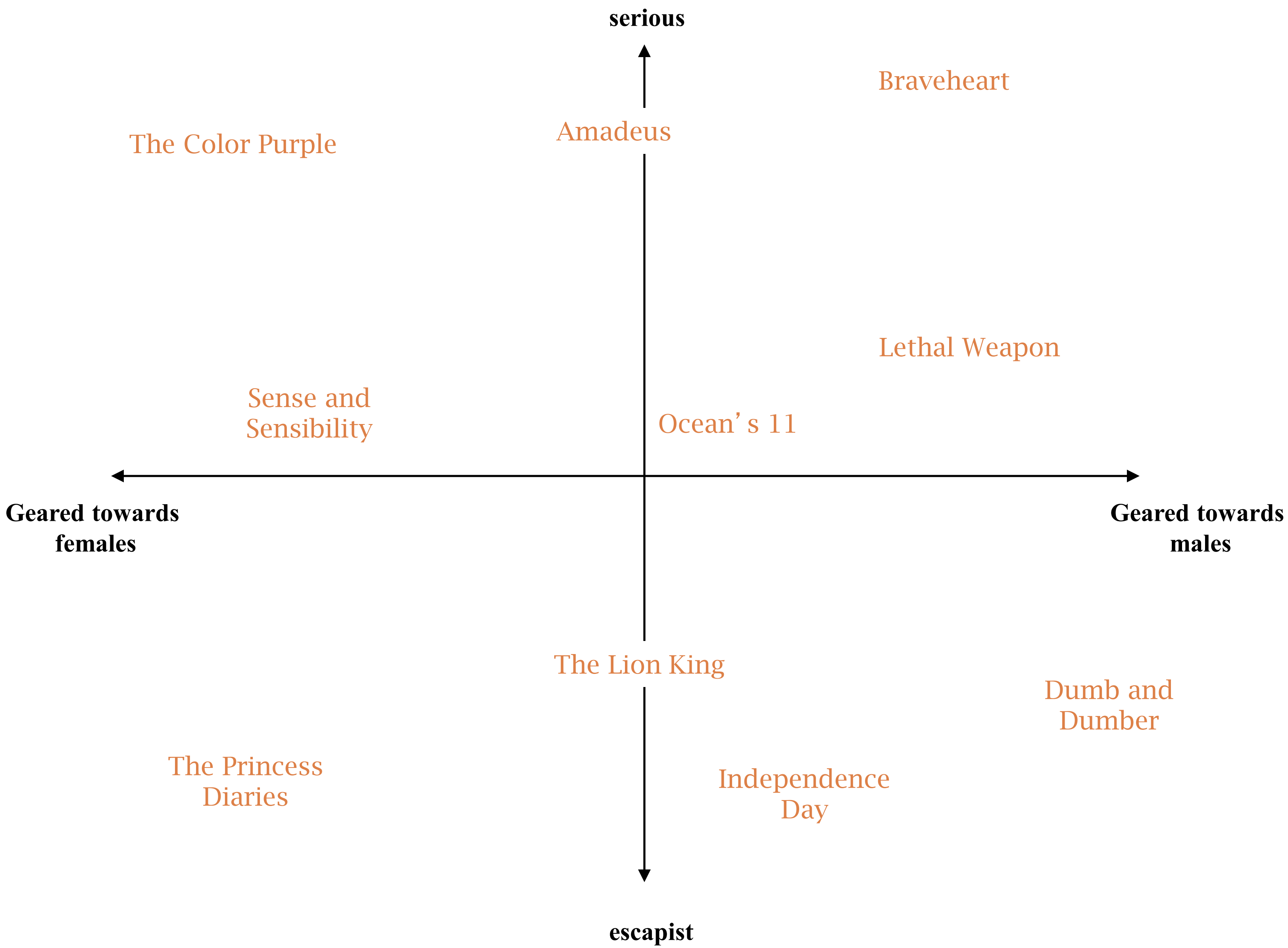
Challenges Of User-based CF Algorithms

- Sparsity – evaluation of large item sets, users purchases are under 1%.
- Difficult to make predictions based on nearest neighbor algorithms =>Accuracy of recommendation may be poor
- Scalability - Nearest neighbor require computation that grows with both the number of users and the number of items.

Other approaches to CF

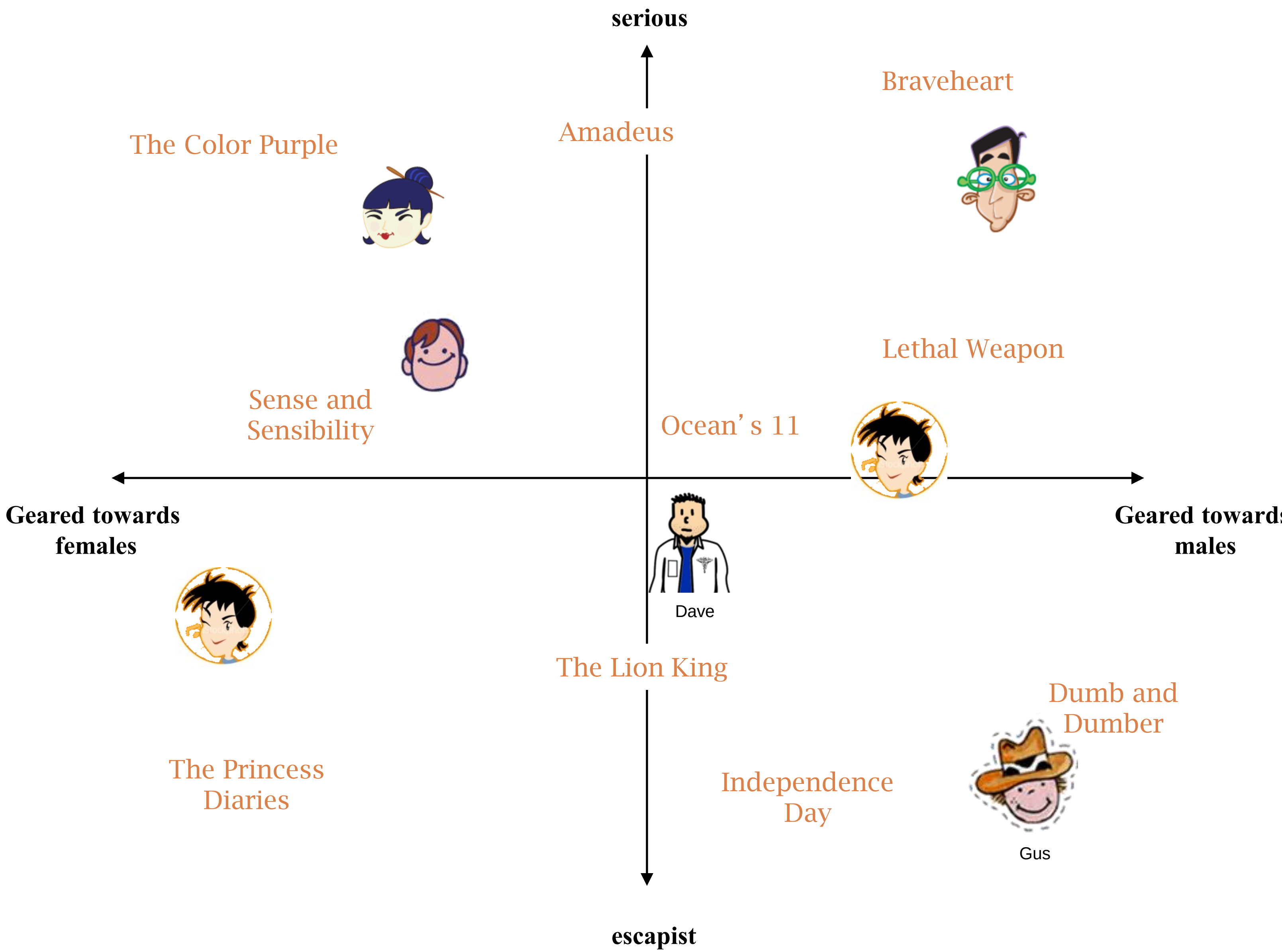
Matrix Factorization

- Dimension reduction technique for matrices
  - X[n x m] = U[n x d] V[m x d]^T
- Each item summarized by a *d*-dimensional vector q_i
- Similarly, each user summarized by p_u
- Choose *d* much smaller than number of items or users
  - *e.g.*, $d = 50 \ll 18{,}000$ or $480{,}000$
- Predicted rating for Item *i* by User *u*
  - Inner product of q_i and p_u
  - $\hat{r}_{ui} = q_i^T p_u \quad \text{or} \quad \hat{r}_{ui} = m + a_u + b_i + q_i' p_u$


serious
Braveheart
Amadeus
The Color Purple
Lethal Weapon
Sense and Sensibility
Ocean's 11
Geared towards females
Geared towards males
The Lion King
Dumb and Dumber
The Princess Diaries
Independence Day
escapist

Example

- This graph shows a hypothetical layout of movies in two dimensions.
- In the example, the horizontal dimension contrasts “chick flicks” from “macho movies”, while the vertical dimension measures the seriousness of the movie.
- In a real application of SVD, an algorithm would determine the layout, so it night not be easy to label the axes.


serious
Braveheart
Amadeus
The Color Purple
Lethal Weapon
Sense and Sensibility
Ocean's 11
Geared towards females
Geared towards males
Dave
The Lion King
Dumb and Dumber
The Princess Diaries
Independence Day
Gus
escapist

Example…

- Users fall into the same space as movies, where a user's position in a dimension reflects the user's preference for (or against) movies that score high on that dimension.
- For example, Gus tends to like male-oriented movies, but dislikes serious movies. Therefore, we would expect him to love "Dumb and Dumber" and hate "The Color Purple".
- Note that these two dimensions do not characterize Dave's interests very well; additional dimensions would be needed.

Collaborative filtering: Pros

- No feature selection needed

Collaborative filtering: Cons

- **Sparsity**
  - If there are many items to be recommended, even if there are many users, the user/ratings matrix is sparse, and it is hard to find users that have rated the same items
- **Cold Start**
  - New User Problem
    - To make accurate recommendations, the system must first learn the user's preferences from the ratings
  - New Item Problem
    - Until the new item is rated by a substantial number of users, the recommender system is not able to recommend it
- **Popularity Bias**
  - Cannot recommend items to someone with unique tastes
  - Tends to recommend popular items

Hybrid

- Combine the results of different recommendation techniques into a single recommendation list
  - **Example 1**: a linear combination of recommendation scores
  - **Example 2**: treats the output of each recommender (collaborative, content-based and demographic) as a set of votes, which are then combined in a consensus scheme