AI6102: Machine Learning Methodologies & Applications

L12: Recommender Systems

Sinno Jialin Pan

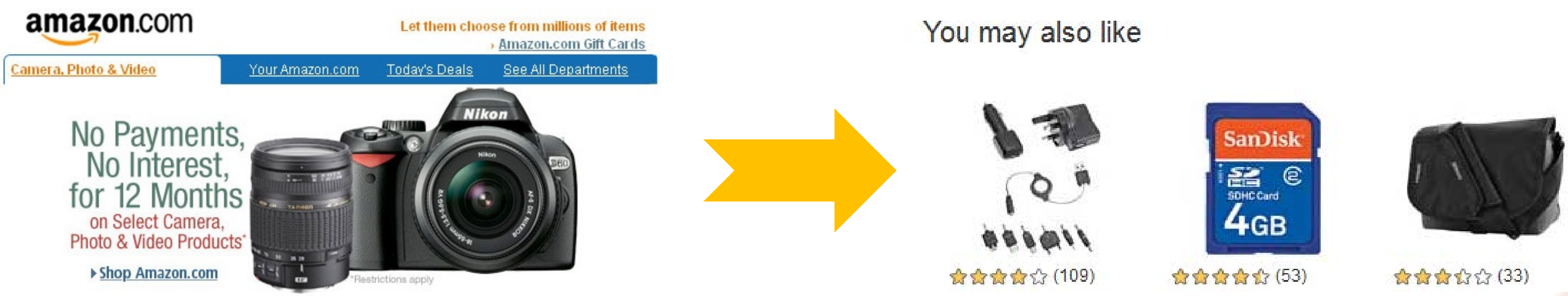
Nanyang Technological University, Singapore

Homepage: http://www.ntu.edu.sg/home/sinnopan

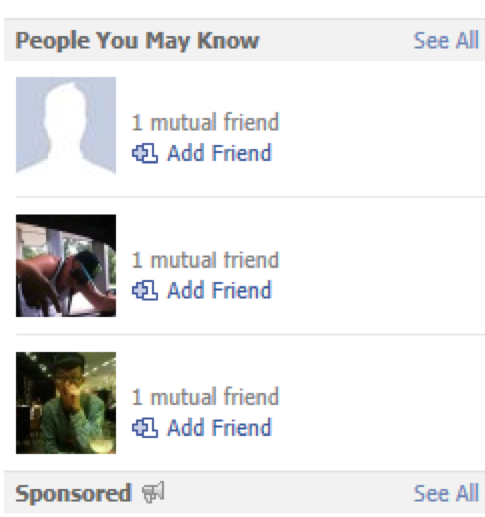
Acknowledgements: slides are adapted from the lecture notes of the book "Recommender Systems: An Introduction" Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich.

NANYANG TECHNOLOGICAL UNIVERSITY

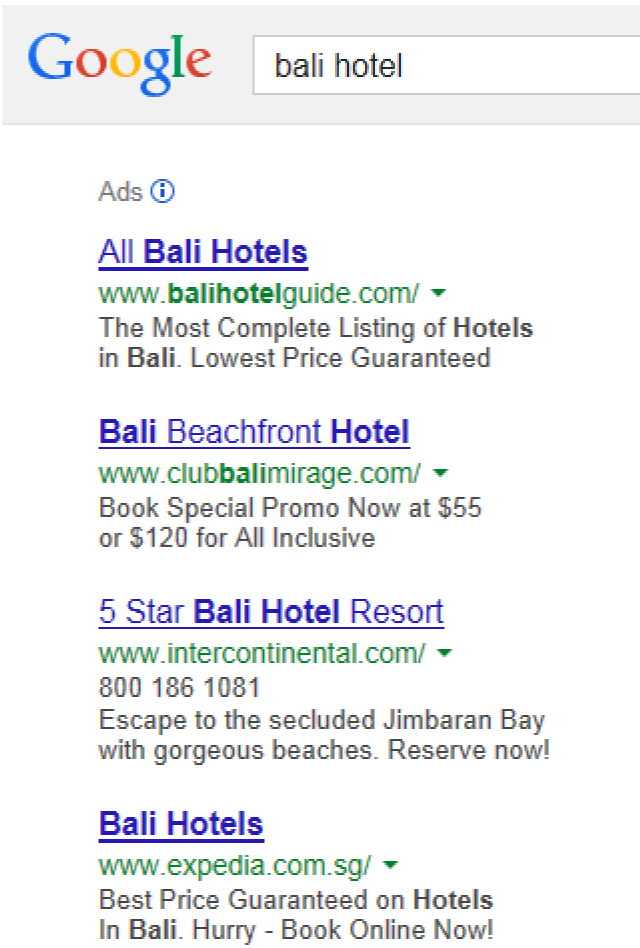
Recommender Systems

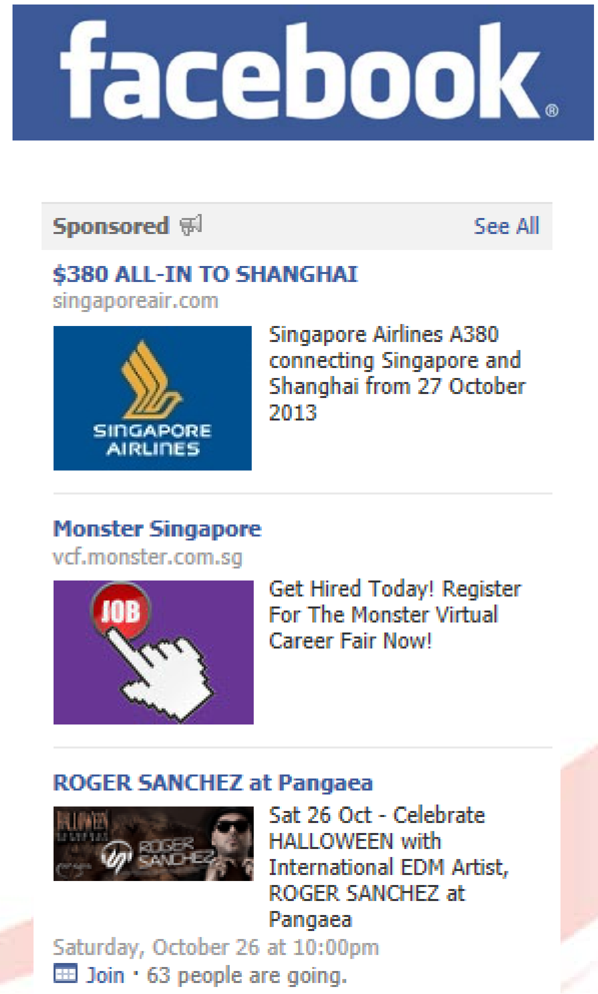


Social Media



Computational Advertising





Roadmap

- Introduction
- Collaborative filtering
  - Memory-based approaches
  - Model-based approaches
- Content-based recommendation
- Evaluation techniques

Problem Domain

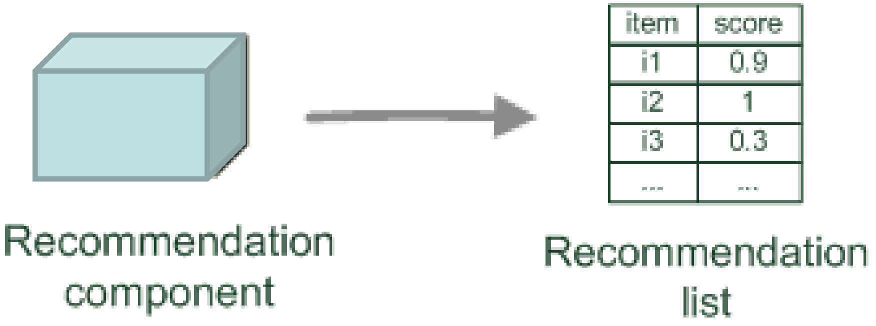
- Recommender systems help to match users with items
  - Software agents that elicit the interests and preferences of individual consumers and make recommendations on items accordingly
- Different system designs / paradigms
  - Based on availability of exploitable data
  - Implicit and explicit user feedback
  - Domain characteristics

Problem Statement (cont.)

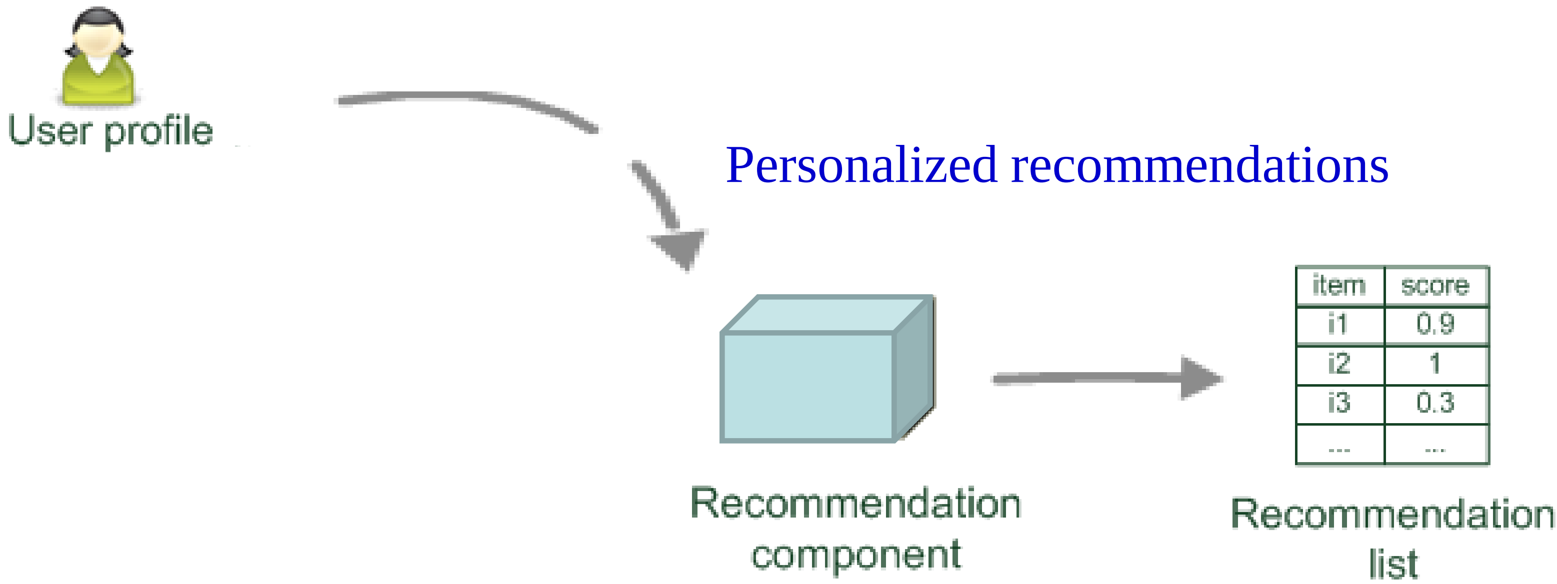
- A recommender system can be seen as a function
- Given
  - User model (e.g. ratings, preferences, demographics, situational context)
  - Items (with or without description of item characteristics)
- Estimate
  - Relevance scores of items, used for ranking
- Finally:
  - Recommend items that are assumed to be relevant

Paradigms

Recommender systems reduce information overload by estimating relevance

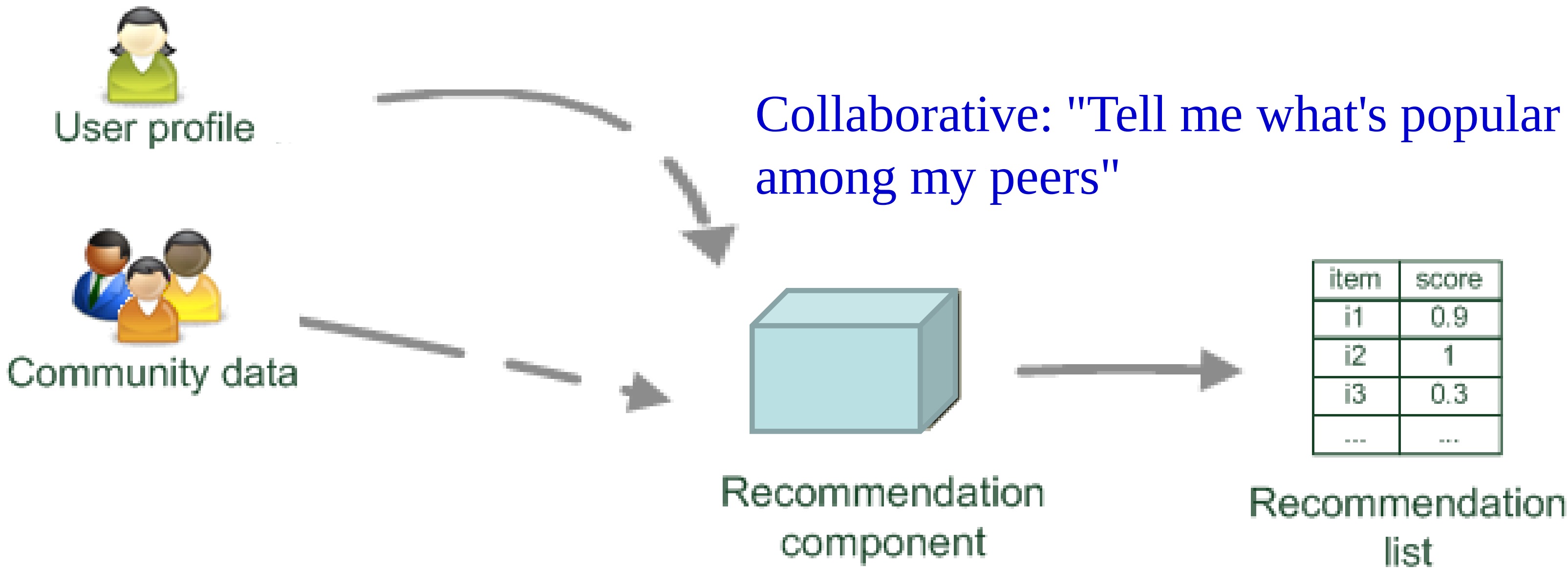


Paradigms (cont.)



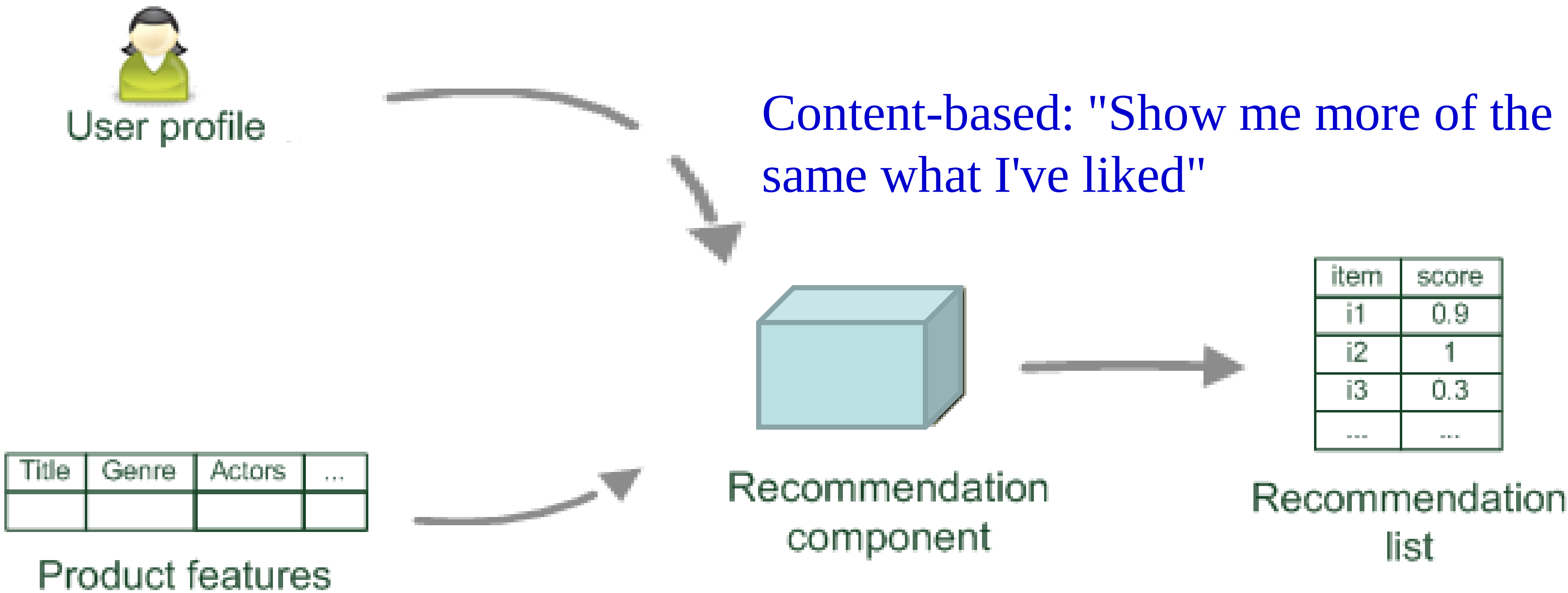
Paradigms (cont.)

Collaborative: "Tell me what's popular among my peers"

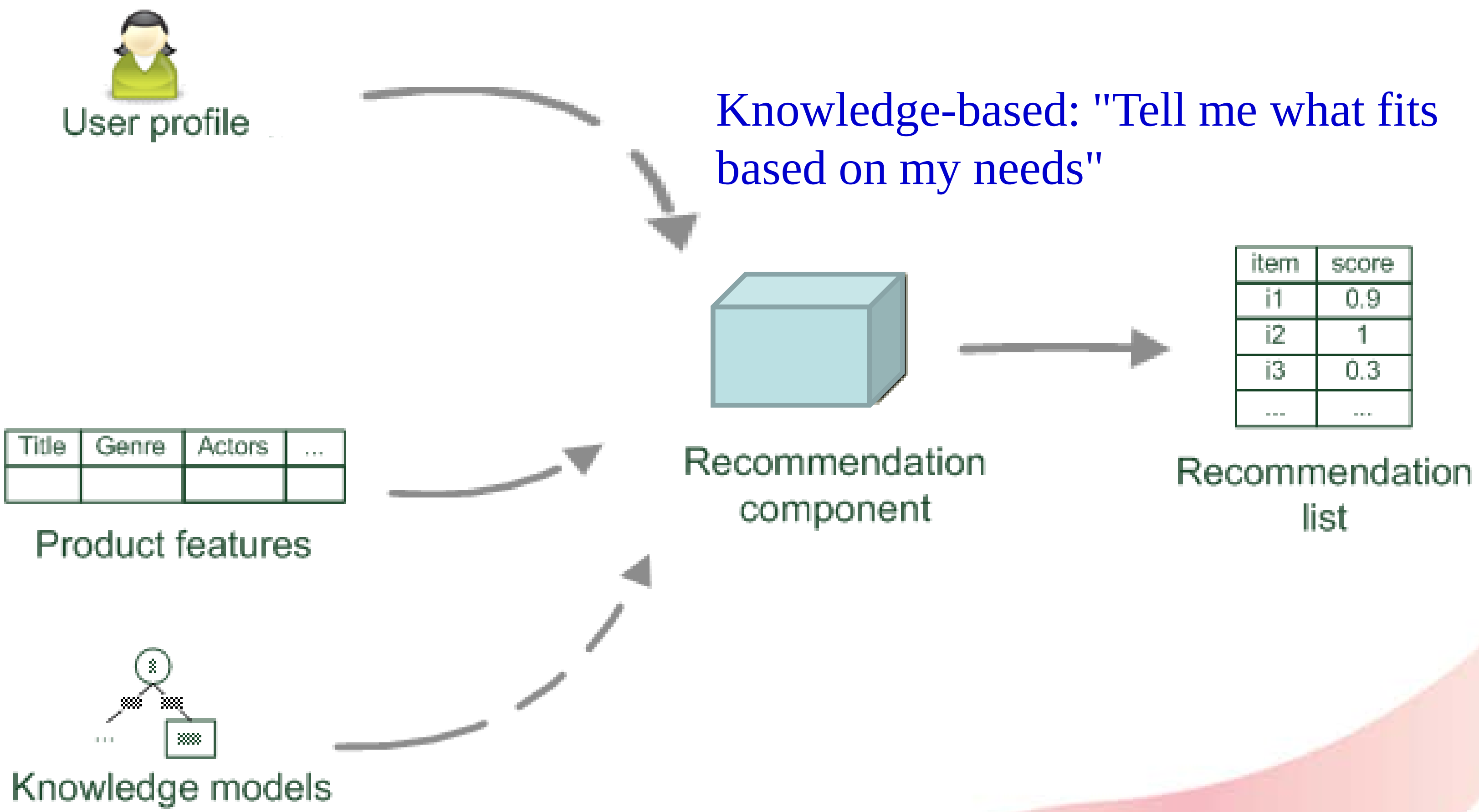


Paradigms (cont.)

Content-based: "Show me more of the same what I've liked"

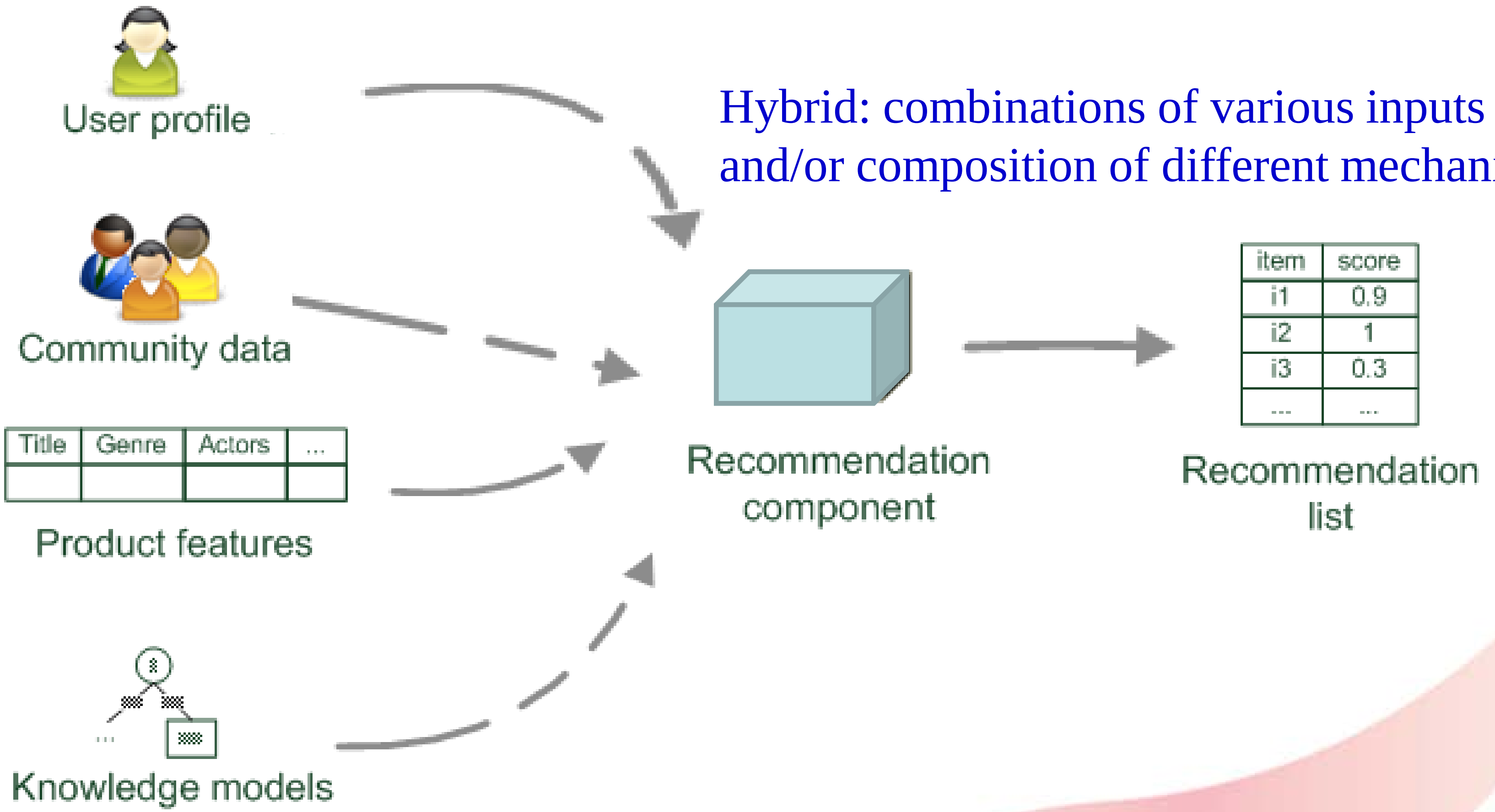


Paradigms (cont.)



Paradigms (cont.)

Hybrid: combinations of various inputs and/or composition of different mechanism



Basic Techniques

	Pros	Cons
Collaborative filtering	No knowledge-engineering effort	Requires some form of rating feedback, cold start for new users and new items
Content-based	No community required	Content descriptions necessary
Knowledge-based	Deterministic recommendations, no cold-start	Knowledge engineering effort to bootstrap, basically static, does not react to short-term trends

Roadmap

- Introduction
- Collaborative filtering
  - Memory-based approaches
  - Model-based approaches
- Content-based recommendation
- Evaluation techniques

Collaborative Filtering (CF)

- The most prominent approach to generate recommendations
  - Used by large, commercial e-commerce sites
  - Well-understood, various algorithms and variations exist
  - Applicable in many domains (book, movies, DVDs, ..)
- Approach
  - Use the "wisdom of the crowd" to recommend items
- Basic assumption
  - Users give ratings to catalog items (implicitly or explicitly)
  - Customers who had similar tastes in the past, will have similar tastes in the future

Input to CF Approaches

- Pure CF-based systems only rely on the rating matrix
- Explicit ratings
  - Most commonly used (e.g., 1 to 5 rating scales)
  - Users are not always willing to rate many items: sparse rating matrices

	Item 1	**Item 2**	**Item 3**	**…**	**Item M**
User 1	1	3	?	..	?
User 2	?	?	2	..	2
…	…	…	…	…	…
User $N-1$	?	2	?	…	4
User N	?	?	5	…	?

Explicit feedback (e.g., ratings on items)

Input to CF Approaches (cont.)

- Implicit ratings
  - Clicks, page views, time spent on some page, demo downloads, etc.

	Item 1	Item 2	Item 3	…	Item M
User 1	1	0	?	..	?
User 2	?	?	1	..	1
…	…	…	…	…	…
User $N-1$	?	1	?	…	0
User N	1	?	0	…	?

Implicit feedback (e.g., click-through records)

CF Approaches

- Memory-based approaches
  - User-based approaches
  - Item-based approaches
- Model-based approaches

User-based Collaborative Filtering

- Given a target user (e.g., Alice) and an item v without rating from Alice
- The goal is to estimate Alice's rating for this item, e.g., by
  - Find a set of similar users (i.e., neighbors) who liked the same items as Alice in the past and who have rated item v
  - Use, e.g. the average of their ratings, to predict Alice’s rating on item v
  - Apply this to all items Alice has not rated and recommend the (estimated) best-rated items to Alice

	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	5	3	4	4	?
User 1	3	1	2	3	3
User 2	4	3	4	3	5
User 3	3	3	1	5	4
User 4	1	5	5	2	1

User-based CF (cont.)

- How do we measure similarity between users?
- How many neighbors should we consider based on the similarities?
- How do we generate a prediction from the neighbors' ratings?

	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	5	3	4	4	?
User 1	3	1	2	3	3
User 2	4	3	4	3	5
User 3	3	3	1	5	4
User 4	1	5	5	2	1

User Similarity Measure

- A popular similarity measure in user-based CF: Pearson correlation coefficient
- Recall Pearson correlation coefficient:

$$\mathrm{PCC}(X_i, X_j) = \frac{1}{N}\sum_{k=1}^{N}\left(\left(\frac{x_{ik} - \mu_{X_i}}{\sigma_{X_i}}\right) \times \left(\frac{x_{jk} - \mu_{X_j}}{\sigma_{X_j}}\right)\right)$$

- Pearson correlation between users

Rating of the item k given by user u_i

The average rating given by user u_i over the items in I

$$\mathrm{sim}(u_i, u_j) = \mathrm{PCC}(u_i, u_j) = \frac{1}{\boxed{|I|}}\sum_{k\in I}\left(\left(\frac{\boxed{r_{ik}} - \boxed{\bar{r}_{u_i}}}{\boxed{\sigma_{u_i}}}\right) \times \left(\frac{r_{jk} - \bar{r}_{u_j}}{\sigma_{u_j}}\right)\right)$$

The set of co-rated items by u_i and u_j

The standard deviation of ratings over the items in I given by user u_i

User Similarity Measure (cont.)

- The output of Pearson correlation coefficient is $[-1, 1]$, where 1 means perfectly positively correlated, and -1 means perfectly negatively correlated
- Use $|\mathrm{PCC}(u_*, u_j)|$ to rank u_j's in non-increasing order to find top K (hyper-parameter) neighbors to u_*

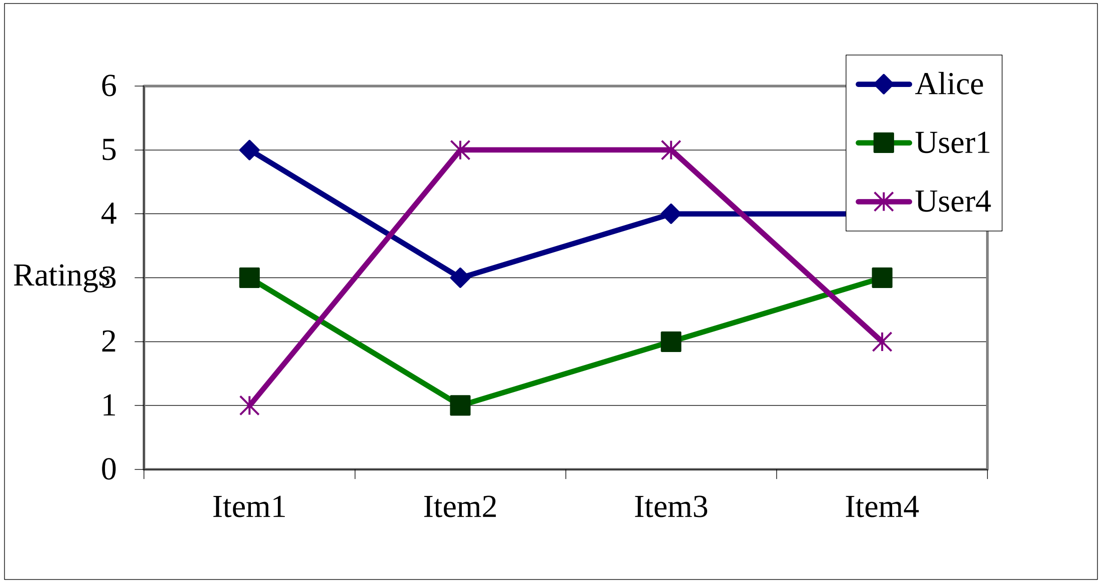
	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	5	3	4	4	?
User 1	3	1	2	3	3
User 2	4	3	4	3	5
User 3	3	3	1	5	4
User 4	1	5	5	2	1

0.85

-0.79

User Similarity Measure (cont.)

- PCC takes differences in rating behavior into account



- Works well, compared with alternative measures
  - such as cosine similarity

Generate A Prediction

- A common prediction function:

$$r_{iv} = \boxed{\bar{r}_{u_i}} + \boxed{\frac{\sum_{u_j \in \mathcal{N}} \left(\mathrm{sim}(u_i, u_j) \left(\boxed{r_{jv} - \bar{r}_{u_j}} \right) \right)}{\sum_{u_j \in \mathcal{N}} \mathrm{sim}(u_i, u_j)}}$$

- Calculate whether the neighbors' ratings for the unseen item v are higher or lower than their average
- Combine the rating differences using the similarity as a weight
- Add/subtract the neighbors' bias from the active user's average and use this as a prediction

Potential Issues

- The similarity between two users is computed based on the their co-rated items
- The numbers of co-rated items between different pairs of users can be very different
- The similarity estimated based on many co-rated items is more reliable than that estimated based on a few co-rated items
  – Define “significance weight” based on the number of co-rated items

Potential Issue of PCC (cont.)

target user

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Alice	5	3	5	4	4	?
User 1	?	1	1	?	?	3
User 2	?	?	1	?	3	5
User 3	3	3	?	?	5	4
User 4	4	2	4	1	2	1

- Significant weights: $w_{u_4} > w_{u_3} > w_{u_2} = w_{u_1}$
- E.g., $w_{u_4} = \frac{5}{5+3+2+2} = \frac{5}{12}$, $w_{u_3} = \frac{3}{5+3+2+2} = \frac{1}{4}$, $w_{u_1} = w_{u_2} = \frac{2}{5+3+2+2} = \frac{1}{6}$
- $\widetilde{\mathrm{sim}}(\mathrm{Alice}, u_i) = w_{u_i} \times \mathrm{sim}(\mathrm{Alice}, u_i)$

Limitations of User-based CF

- The scalability issue arises if there are many users
- Space complexity $O(N^2)$ when pre-computed, where N is the number of users
  - Time complexity for computing similarity is $O(N^2M)$, where M is the number of items
- High sparsity leads to few common ratings between two users
- If $M \ll N$, we can use Item-based CF instead
  - Exploit relationships between items for recommendation

Item-based Collaborative Filtering

- Basic idea:
  - Use the similarity between items to make predictions
- Example:
  - Look for items that are similar to Item 5
  - Take Alice's ratings on the similar items to predict the rating on Item 5

	Item 1	Item 2	Item 3	Item 4	Item 5
Alice	5	3	4	4	?
User 1	3	1	2	3	3
User 2	4	3	4	3	5
User 3	3	3	1	5	4
User 4	1	5	5	2	1

Item-based CF (cont.)

- Each item is considered as a vector of ratings in N-dimensional space (with missing values), where N is the number of users
- Revised cosine similarity between two items

rating on item v_j
given by user k

$$\mathrm{rcos}(v_i, v_j) = \frac{\sum_{k \in U} (r_{ki} - \bar{r}_k)(\boxed{r_{kj}} - \boxed{\bar{r}_k})}{\sqrt{\sum_{k \in U} (r_{ki} - \bar{r}_k)^2} \sqrt{\sum_{k \in U} (r_{kj} - \bar{r}_k)^2}}$$

  - U: the set of users who have rated both items v_i and v_j
  - Take average user ratings into account

The average rating
given by user k

Generate A Prediction

- A common prediction function:

$$r_{ui} = \frac{\sum_{v_j \in \mathcal{N}} \text{sim}(v_i, v_j) r_{uj}}{\sum_{v_j \in \mathcal{N}} \text{sim}(v_i, v_j)}$$

Notes on Item-based CF

- The scalability issue arises if there are many items
- Space complexity $O(M^2)$ when pre-computed, where M is the number of items
  - Time complexity for computing similarity $O(M^2N)$, where N is the number of users
- If $M \ll N$, item-based, otherwise if $N \ll M$, user-based
- If both M and N are very large, both item-based and user-based are computationally expensive
- Item similarities are supposed to be more stable than user similarities

Summary: Memory-based

- Memory-based approaches including both user-based and item-based:
  - The rating matrix is directly used to find neighbors and make predictions
  - Does not scale for most real-world scenarios
    - Large e-commerce sites have tens of millions of customers and millions of items

Model-based CF Approaches

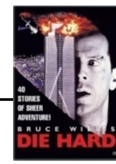
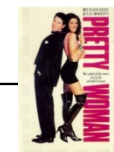
- Based on an offline model-learning phase to learn a model
- At run-time, only the learned model is used to make predictions
- Models are updated / re-trained periodically
- Model-building and updating can be computationally expensive

Model-based CF Approaches (cont.)

- Matrix factorization
  - Consider rating prediction as a matrix completion problem
  - Use matrix factorization to solve the problem
- Association rule mining
  - Analysis on item associations
- Probabilistic models
  - Output rating probabilities
- Various other machine learning approaches

Matrix Completion

Sparse rating matrix: $\mathbf{R}$ $(N \times M)$

						…
Alice	1	?	3	?	?	?
Bob	2	?	?	5	?	?
Mary	?	3	?	?	5	?
Sue	?	?	4	1	?	?
…	?	?	?	?	?	?

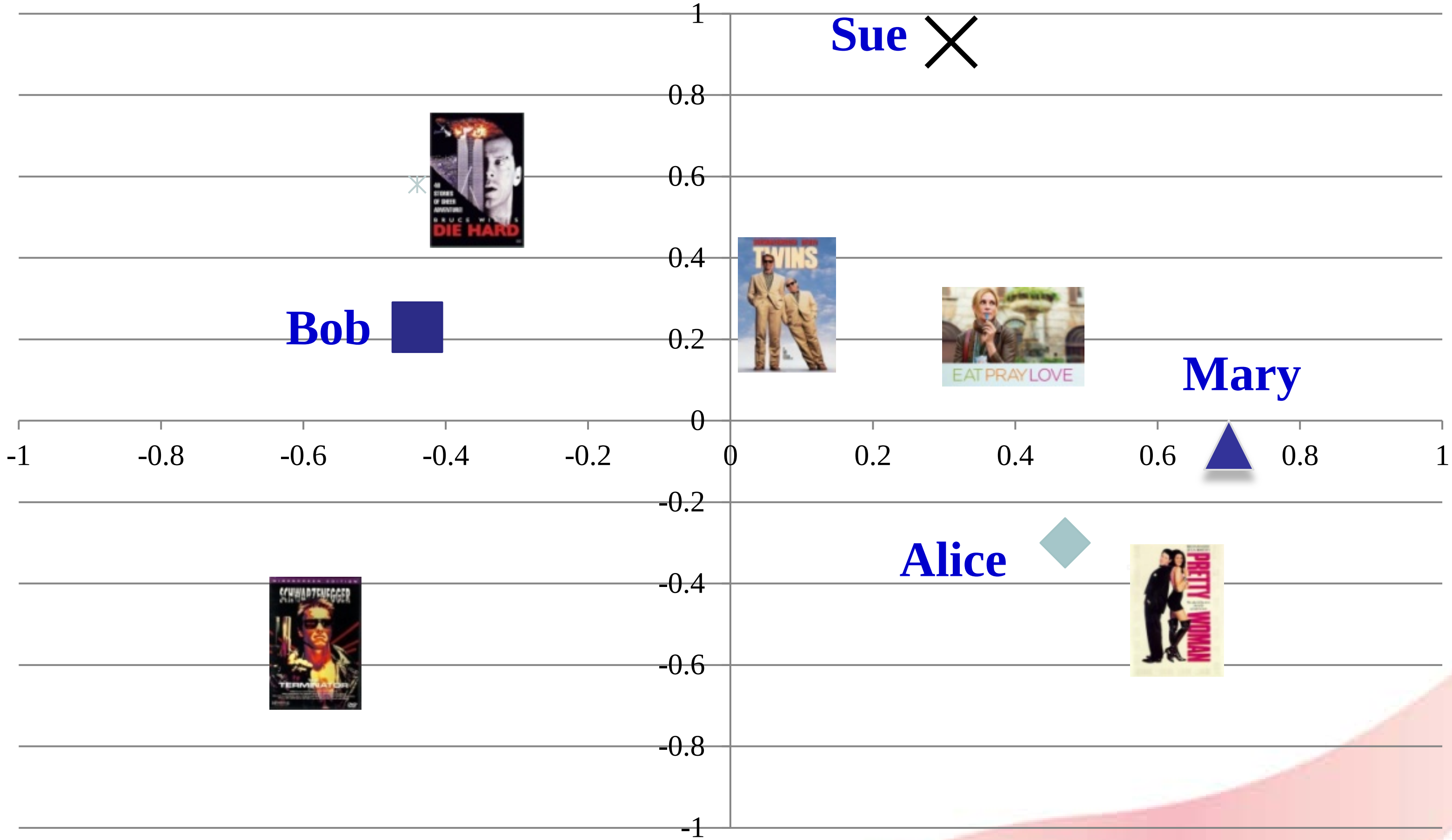
N users

M items

Matrix Factorization Methods

- Each user can be represented by an M-dimensional vector
- Each item can be represented by a N-dimensional vector
- High dimensional observations are controlled by some latent factors
  - Idea of dimensionality reduction
  - For a user, factors can be interests, ages, etc., but also implicit ones
  - For an item, factors can be actors, genre, etc., but also implicit ones
- Assume there are K factors that capture signals of items and users, respectively
  - Each user is represented by a K-dimensional vector
  - Each item is also represented by a K-dimensional vector

Special Case $K = 2$

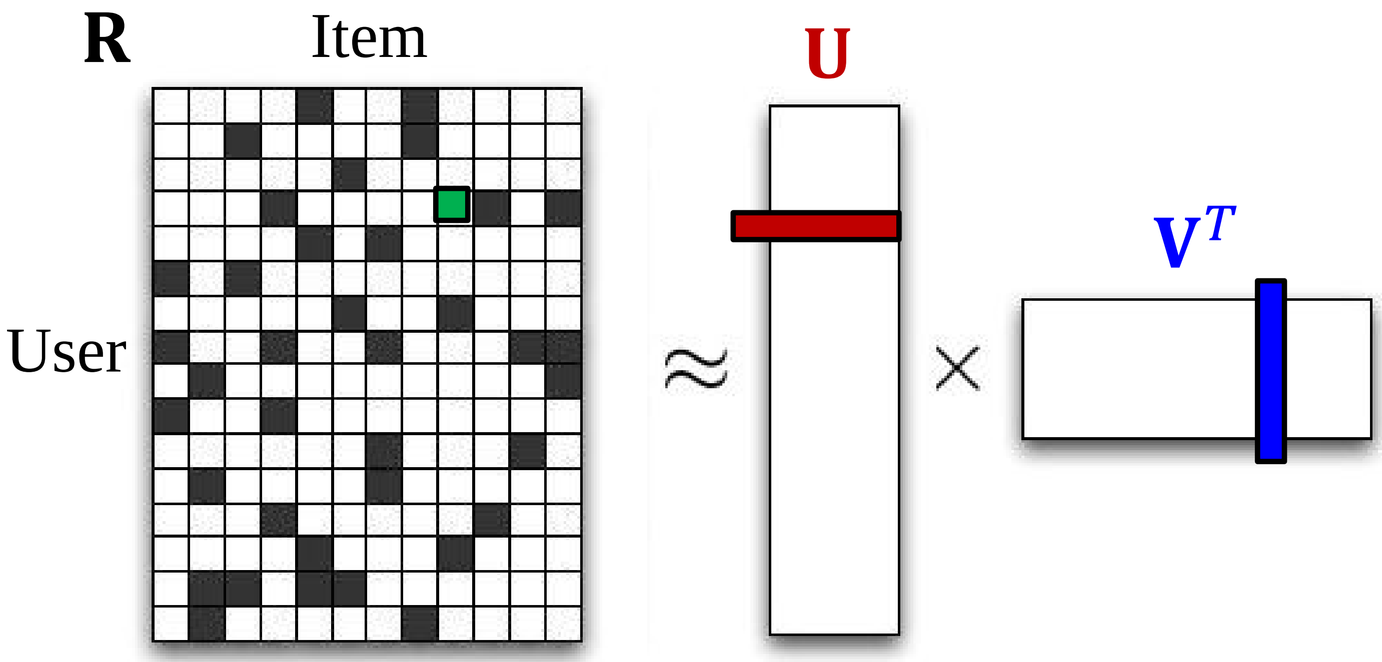


Matrix Factorization (MF)

- An $N-$by$-M$ rating matrix $\mathbf{R}$ can be approximate by the multiplication of an $N-$by$-K$ matrix and a $K-$by$-M$ matrix

$$\mathbf{R} \approx \mathbf{U}\mathbf{V}^T$$

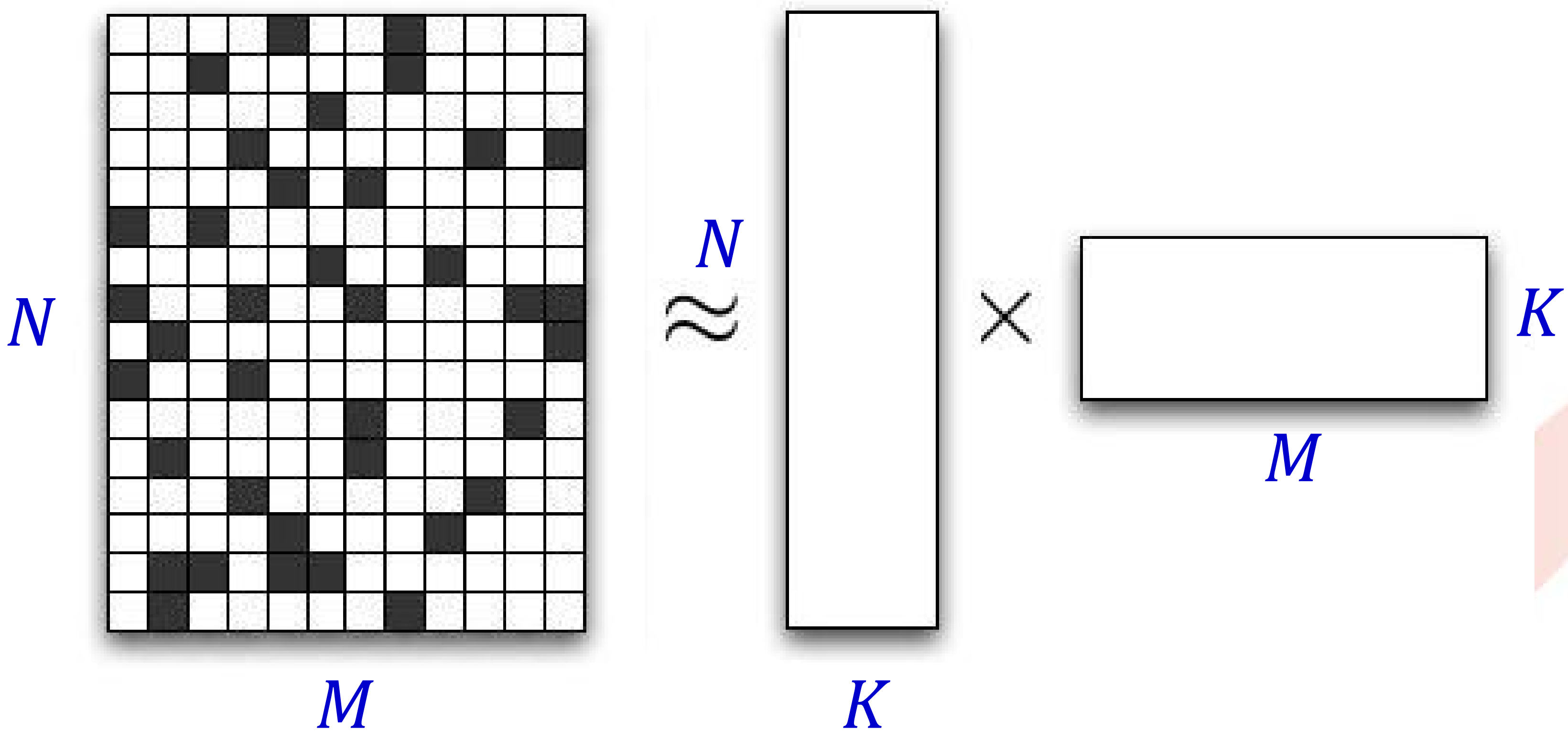
$\mathbf{U}$ is an $N-$by$-K$ matrix
$\mathbf{V}$ is a $M-$by$-K$ matrix



$$r_{ij} \approx \boldsymbol{u}_i \boldsymbol{v}_j^T$$

Advantage of MF

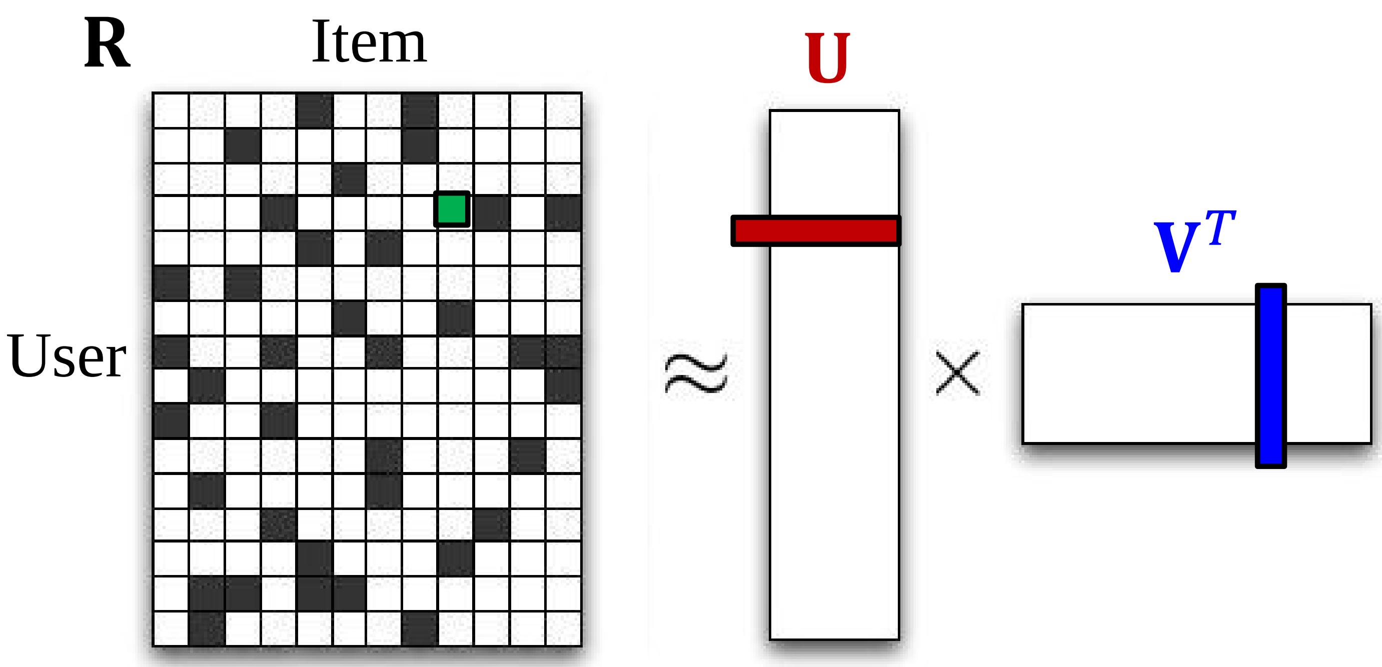
- Suppose there are L ratings ($L \ll N \times M$), for predicting every ratings on unrated items, we need to estimate $N \times M - L \approx N \times M$
- With MF, only need to estimate $(N + M) \times K$, where $K \ll \min(N, M)$



Objective of MF

$$\min_{\mathbf{U},\mathbf{V}} \sum_{\boxed{\{i,j\}\in O}} (r_{ij} - \hat{r}_{ij})^2 \text{, where } \hat{r}_{ij} = \boldsymbol{u}_i \boldsymbol{v}_j^T$$

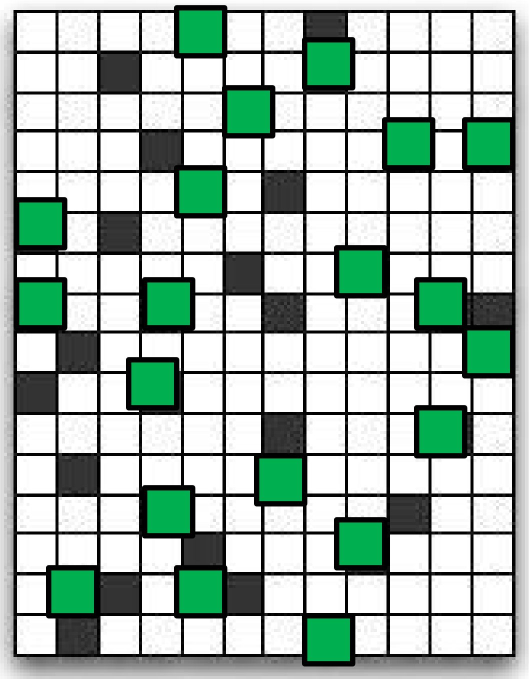
O denotes the observed elements in **R** (rated items)



A gradient-decent-based algorithm can be applied

Cross-Validation for CF

- Each "labeled" data instance is a tuple (uid, tid, r)
- Random sample a subset of tuples for training
- The rest are for testing



Data Sparsity Problem

- Extreme case: cold start problem
  - How to recommend new items? What to recommend to new users?
- Solutions
  - Ask the new user to rate a set of items or ask existing users to rate the new item
  - Use another method (e.g., content-based approaches) in the initial phase
- Sparse issues (not cold start)
  - For the items or users with very few ratings, the corresponding feature vector $\boldsymbol{u}_i$'s and $\boldsymbol{v}_i$'s are not accurate

Summary: CF Approaches

- Pros:
  - well-understood, no knowledge engineering required
- Cons:
  - requires user community, sparsity problems, no integration of knowledge sources, no explanation of results

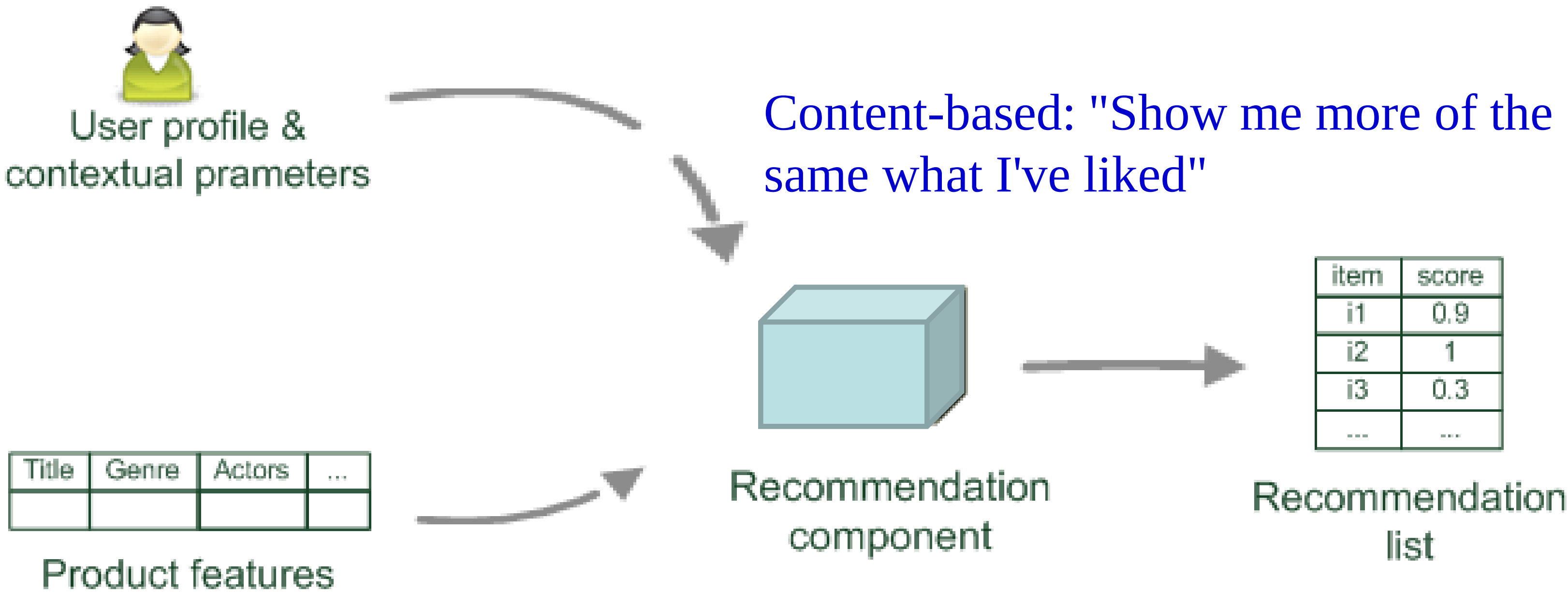
Roadmap

- Introduction
- Collaborative Filtering
  - Memory-based approaches
  - Model-based approaches
- Content-based recommendation
- Evaluation techniques

Item & User Information

- Collaborative filtering does NOT require any information about the items or users
  - It might be reasonable to exploit such information
- What do we need:
  - Some information about the available items
  - Some sort of *user profile* describing his/her preferences
- The task:
  - Learn user preferences
  - Recommend items whose content match the user preferences

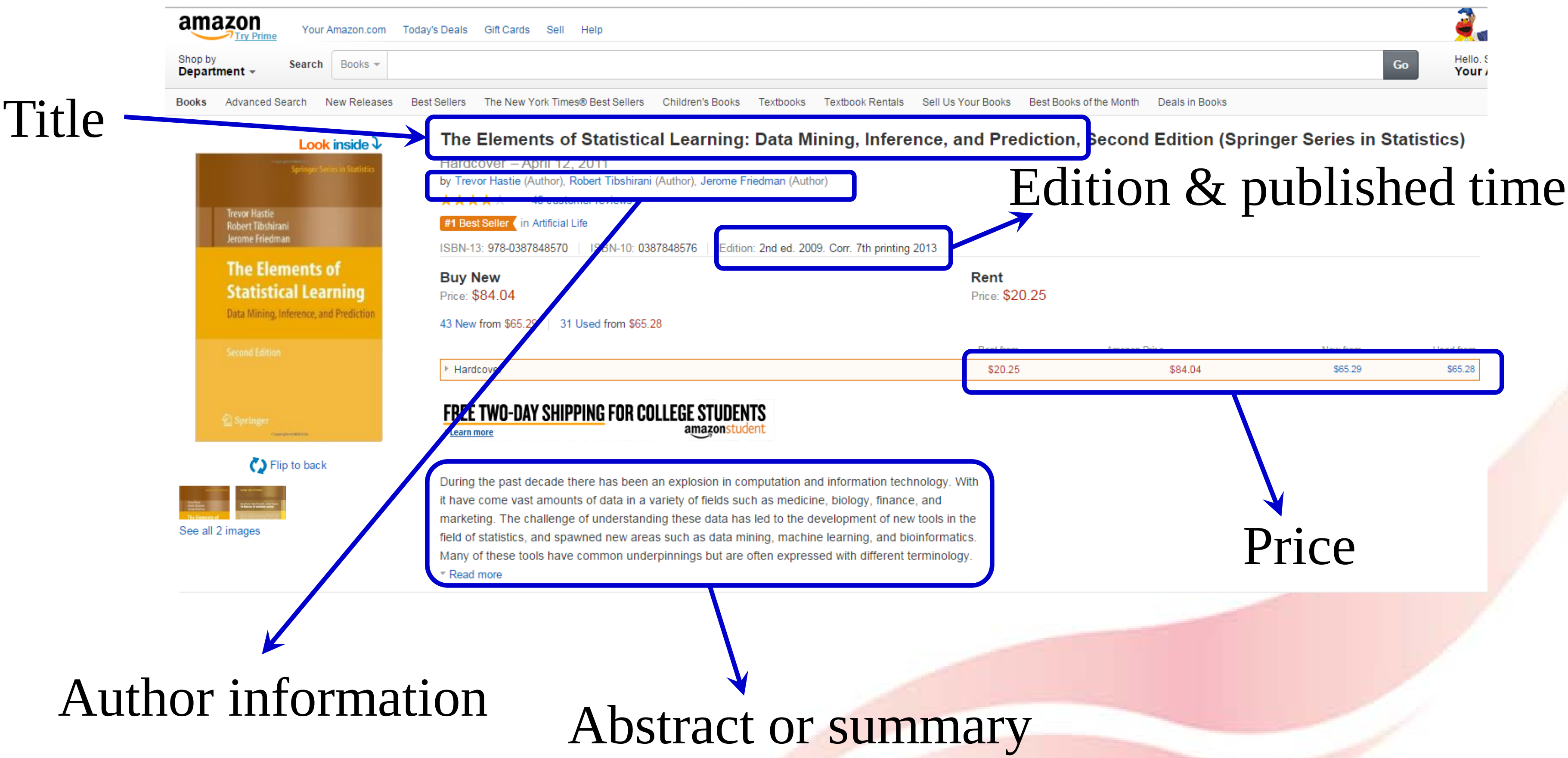
Content-based Recommendation



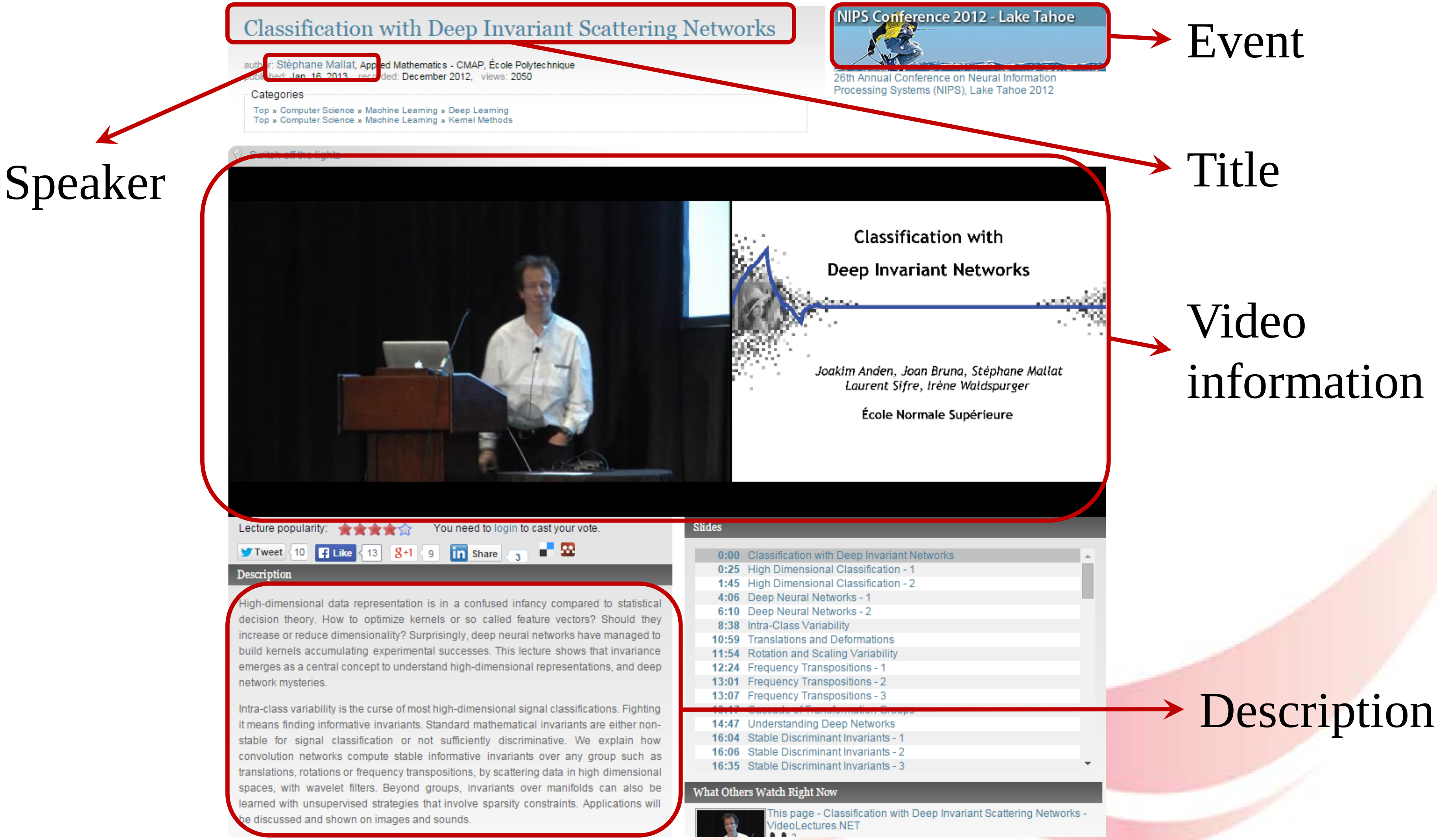
Content-based: "Show me more of the same what I've liked"

What is the “Content”

- In different domains, the definition of “content” is different



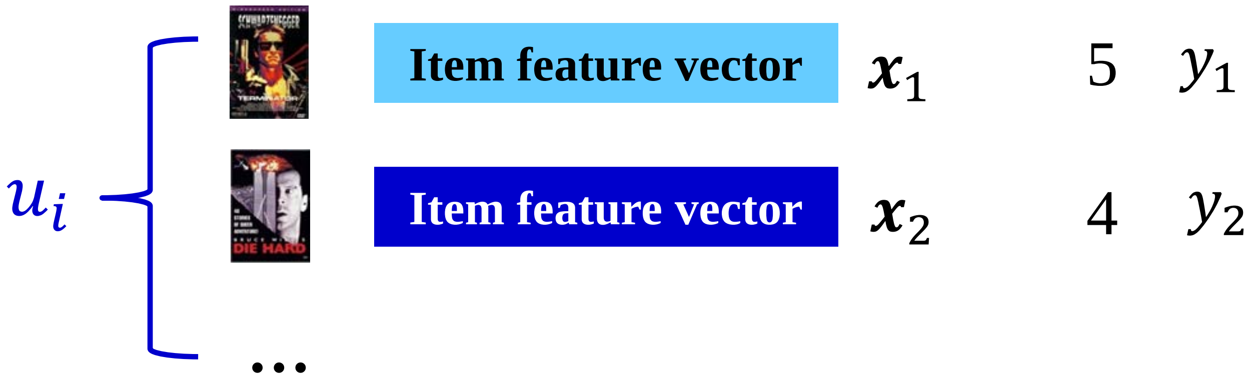
What is the “Content” (cont.)



Content-based Approaches

- Simple method:
  - For each user u_i, use the rated items to construct a training set $\{\boldsymbol{x}_j, y_j\}, j \in I_{u_i}$, where I_{u_i} denotes the set of items that user u_i has rated, $\boldsymbol{x}_j$ is the input feature vector of item j and y_j is the corresponding rating given by u_i
  - Train a specific classifier f_{u_i} for each user u_i
  - For an unrated item $\boldsymbol{x}_*$, estimate its rating via $f_{u_i}(\boldsymbol{x}_*)$

Content-based Approaches (cont.)



- Issue:
  - For each user, the size of training data may be too small to train precise classifier

Content-based Approaches (cont.)

- Combine training data from different users?

u_i

Item feature vector $\boldsymbol{x}_1^{(u_i)}$ 5 $y_1^{(u_i)}$

Item feature vector $\boldsymbol{x}_2^{(u_i)}$ 4 $y_2^{(u_i)}$

…

u_j

Item feature vector $\boldsymbol{x}_1^{(u_j)}$ 3 $y_1^{(u_j)}$

Item feature vector $\boldsymbol{x}_2^{(u_j)}$ 4 $y_2^{(u_j)}$

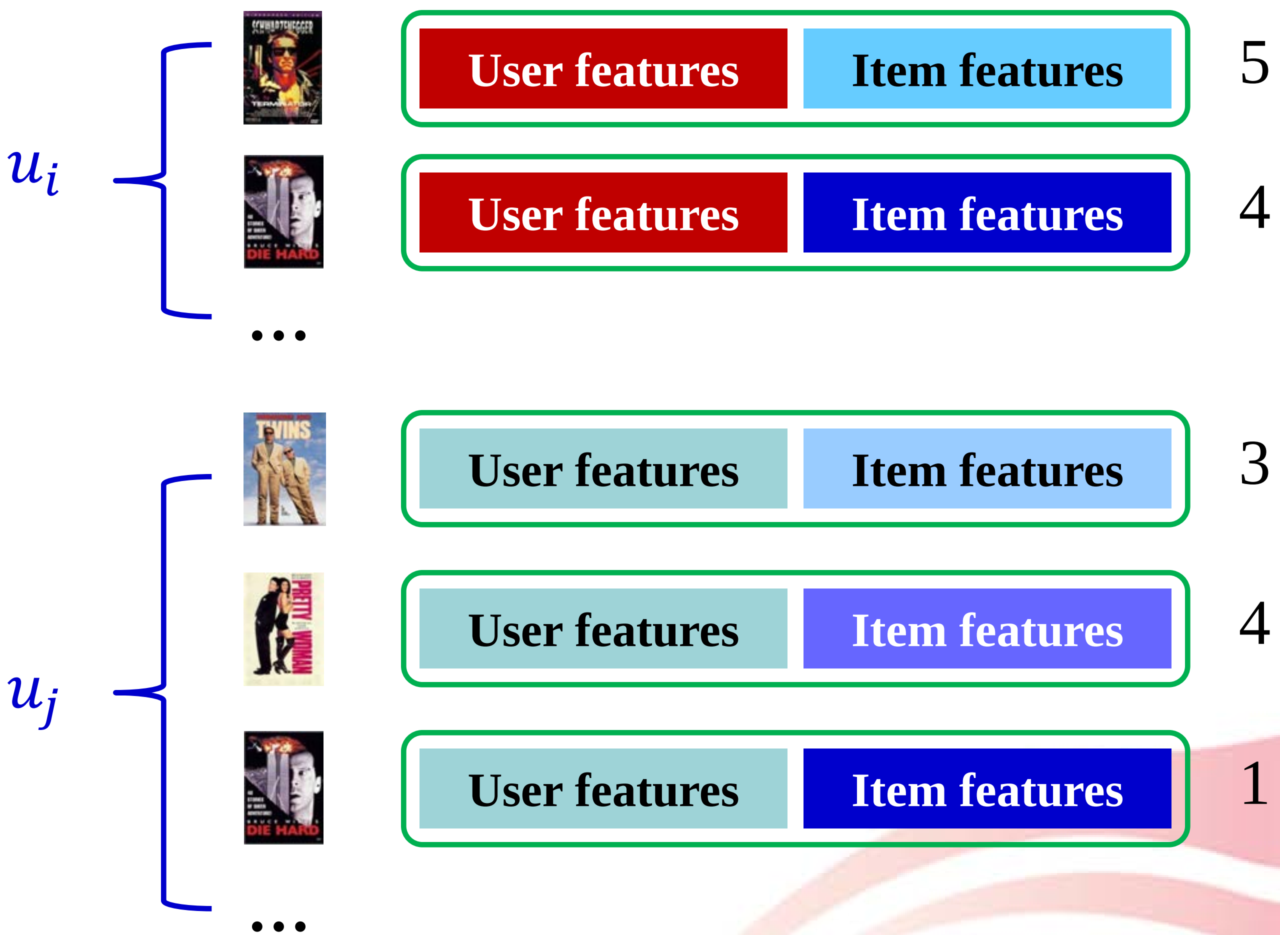
Item features vector $\boldsymbol{x}_3^{(u_j)}$ 1 $y_3^{(u_j)}$

…

For a same item, ratings given by different users can be inconsistent

Content-based Approaches (cont.)

- Solution: construct vectors of user-item pairs



Summary: Content-based

- Pros:
  - Can address cold-start problems
- Cons:
  - Difficult to extract features from unstructural data (text, images, videos)

Roadmap

- Introduction
- Collaborative filtering
  - Memory-based approaches
  - Model-based approaches
- Content-based recommendation
- Evaluation techniques

Evaluation

- Before a recommender system is launched
  - Offline evaluation
  - More focused on technical aspects
- After a recommender system is launched
  - Online evaluation
  - More focused on business benefit

Technical Evaluation

- How accurate is the estimated ratings on unrated items
  - Difference between the predicted ratings and the true ratings
- Hide some items with known ground truth for validation
  - Mean Absolute Error (MAE)

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |\hat{r}_i - r_i|$$

  - Root Mean Square Error (RMSE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (\hat{r}_i - r_i)^2}$$

Technical Evaluation (cont.)

- Final goal of a recommender system is to generate a ranking list of recommended items for each user
  - Measure the quality of the ranking
  - Recommendation is viewed as information retrieval task
    - Retrieve (or recommend) all items which are predicted to be "good" or "relevant"

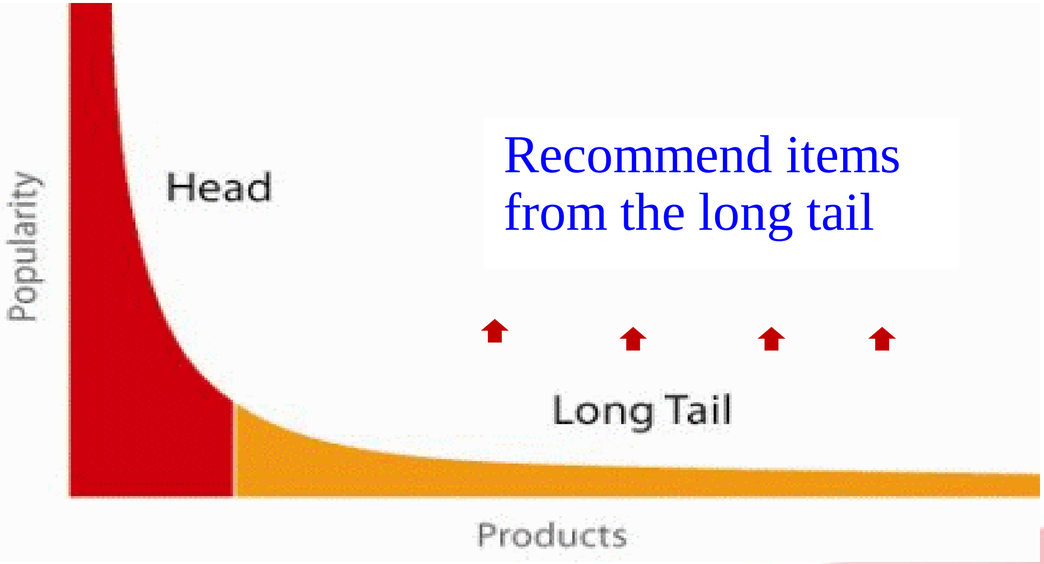
		Reality	
		Actually Good	Actually Bad
Prediction	Rated Good	True Positive (TP)	False Positive (FP)
	Rated Bad	False Negative (FN)	True Negative (TN)

Business Evaluation

- What are the measures in practice?
  - Total sales numbers
    - Promotion of certain items
  - Click-through-rates
    - Interactivity on platform
  - Customer return rates
    - Customer satisfaction and loyalty
  - etc.

When a RS does its Job Well?

- "Recommend widely unknown items that users might actually like!"
- 20% of items accumulate 74% of all positive ratings



Thank you!