CS361
(Software Engineering Program)
Artificial Intelligence II - Applied Machine Learning

Lecture 6
Unsupervised Learning (Clustering)
A Basic Introduction to Recommendation Systems

Amr S. Ghoneim
(Assistant Professor, Computer Science Dept.)
Helwan University

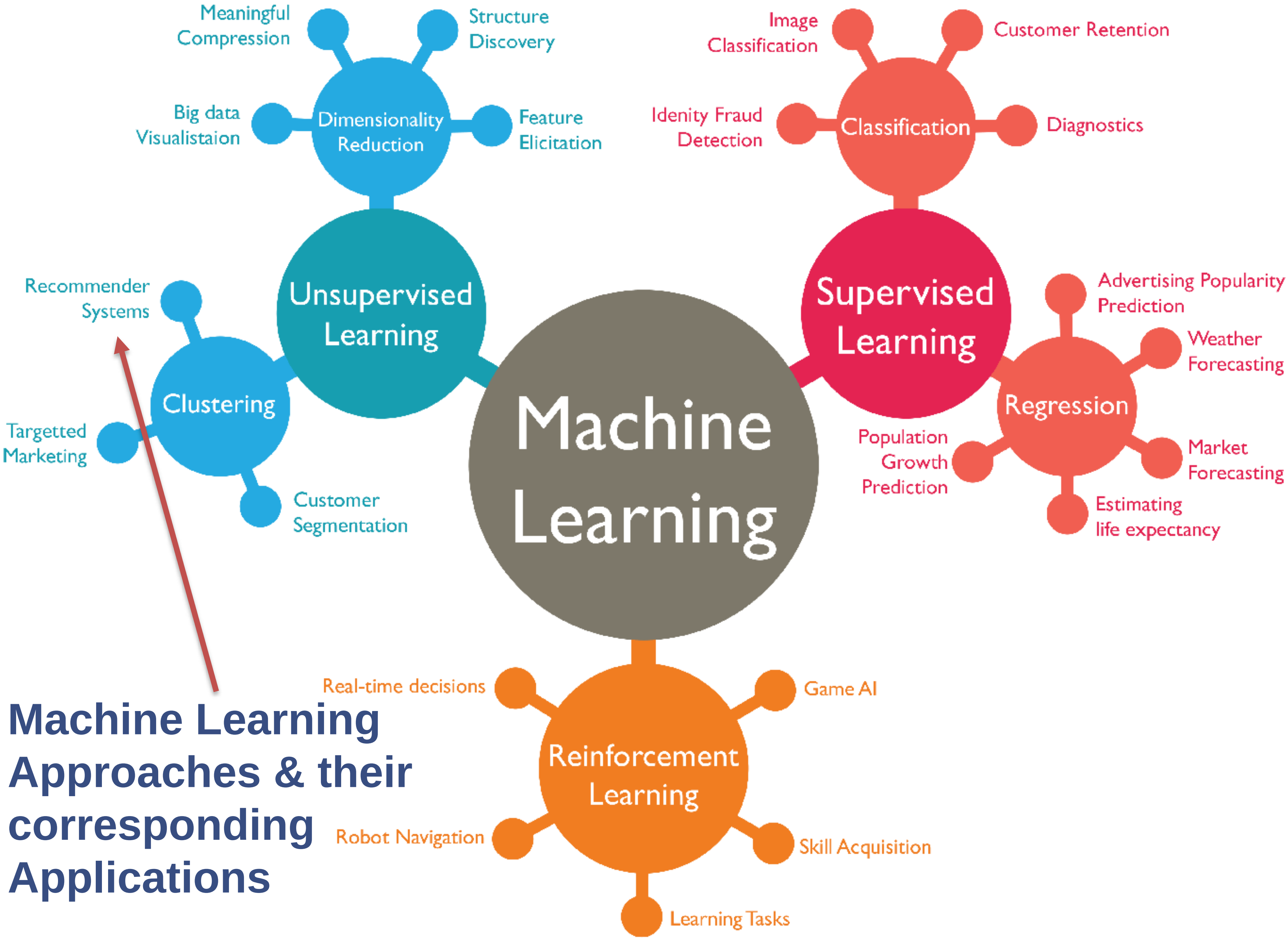
Fall 2019

Lecture is based on its counterparts in the following courses (and the following resources):

- *Recommender Systems: An introduction, lecture at the* **University of Szeged (Hungary),** by Dietmar Jannach **(Technische Universität "TU" Dortmund, Germany)**
- *PV254 Recommender Systems,* **Masaryk University (Czech Republic)**, Department of Machine Learning and Data Processing – Faculty of Informatics
- ***Recommender Systems: An Introduction.*** D. Jannach, M. Zanker, A. Felfernig, G. Friedrich, 2010.
- ***Recommender Systems Handbook.*** F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, 2015 (second edition).

Today's Key Concepts

- Machine Learning Approaches & their corresponding Applications
- Motivation
  - the Jam experiment ..
  - Examples of Applications
  - Recommendation Systems & Education
- Recommendation Systems
  - An "often-cited" Problem Characterization
  - Various Paradigms
    - Collaborative Filtering (CF)
      - User-based Nearest Neighbor CF
      - Pearson Correlation
      - Item-based CF
    - Content-based Recommendation
      - What is the "Content" ?
      - Content Representation
      - Item Similarities
      - Term-Frequency (TF)
      - Inverse Document Frequency (IDF)
      - Example: TF–IDF Representation
      - Improving the Vector Space Model
      - Cosine Similarity
      - Recommending Items



Machine Learning Approaches & their corresponding Applications

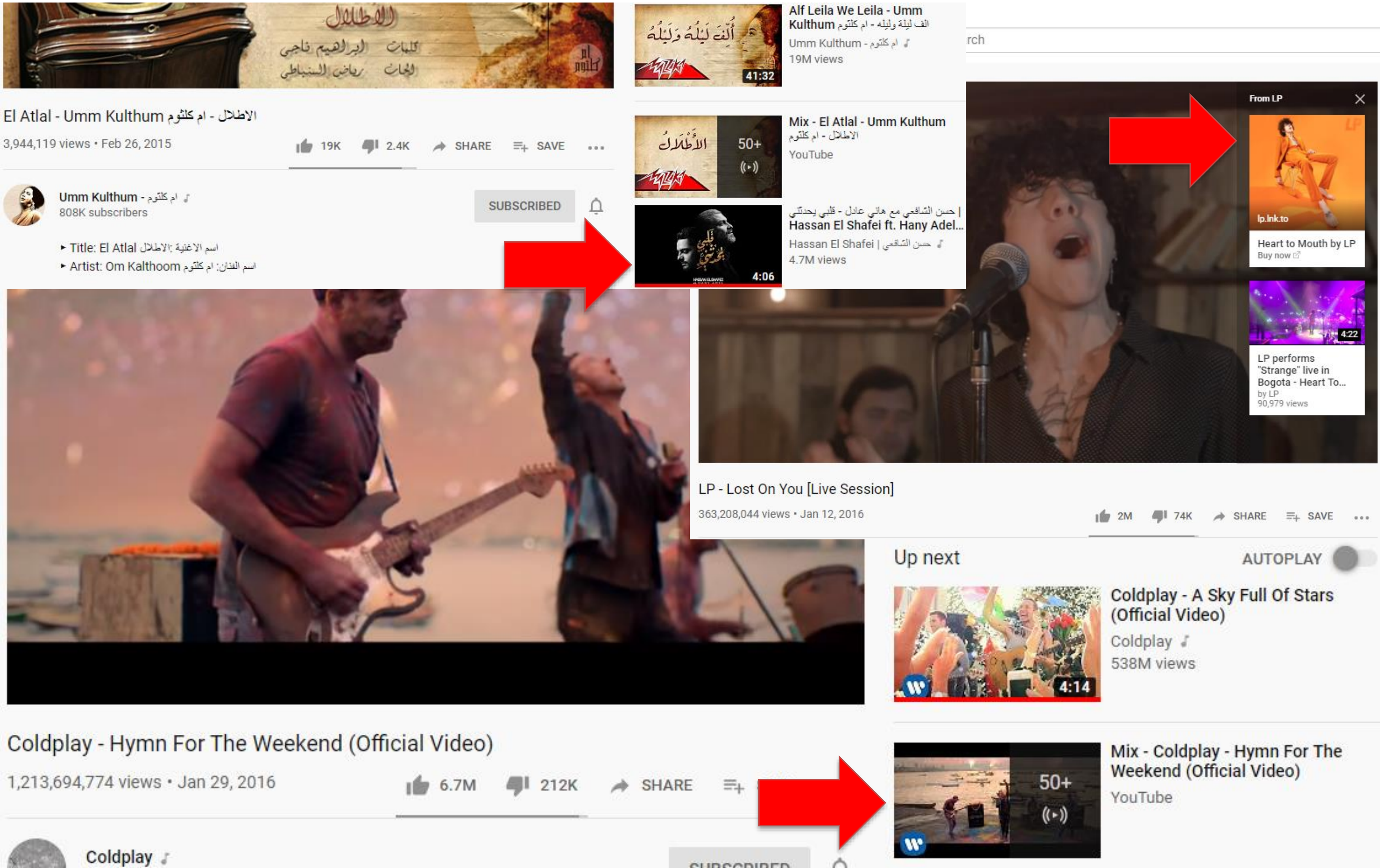
Motivation .. *the Jam experiment ..*

Motivation & Examples of Applications

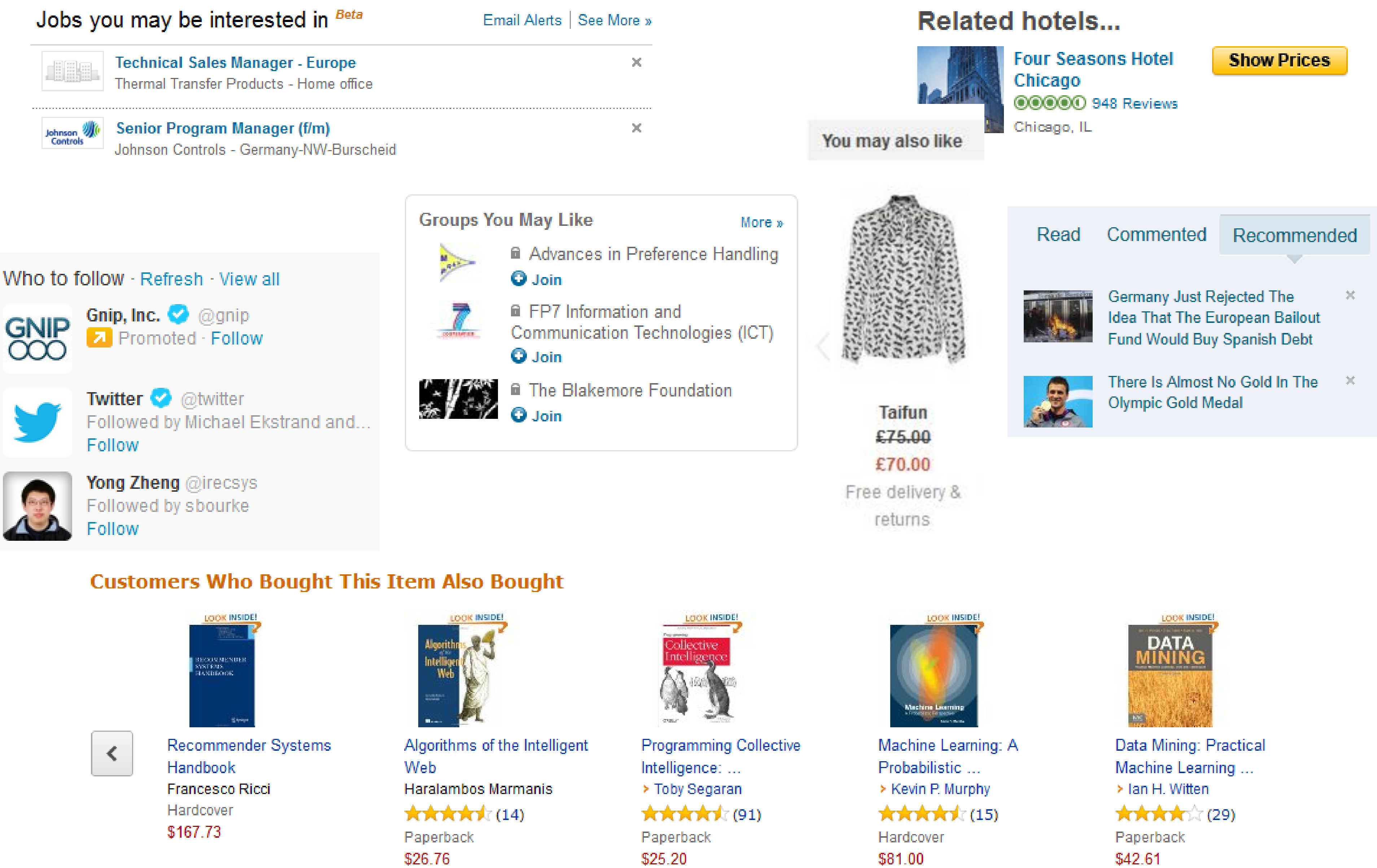
- **Information overload;**
  - Many choices available ..
  - "the paradox of choice" *(Choice Overload, .. "Why more is less")* ..
- **Recommender System;**
  - Provides aid ..
  - Set of items + user "context" ⇒ selection of items (predicted to be "good" for the user)

Movies, online videos, music, books, software *(apps)*, products in general, people *(dating, friends)*, services *(restaurants, accommodation, .., etc.)*, research articles, jokes, .. etc.

Motivation & Examples of Applications



Motivation & Examples of Applications



Motivation & Examples of Applications
Recommendation Systems & Education

- **Learning materials – direct application**
- **Problems, exercises:**
  - **Users ~ Students**
  - **Items ~ Problems**
  - **Ratings ~ Performance *(correctness of answers, problem solving times)***

Personalization in Education:

- **Adaptive learning, Personalized learning, ..**

Value of Recommendations ..

Value for the customer:

- **Find things that are interesting.**
- **Narrow down the set of choices.**
- **Help me explore the space of options.**
- **Discover new things.**
- **Entertainment.**
- **..**

Value for the provider:

- **Additional & probably unique personalized service for the customer.**
- **Increase trust and customer loyalty.**
- **Increase sales, click trough rates, conversion, .. etc.**
- **Opportunities for promotion, persuasion.**
- **Obtain more knowledge about customers.**
- **..**

Value of Recommendations ..

- **Netflix:** ~ 2/3 of the movies watched
- **Amazon:** ~ 35% of the sales
- **Google News:** Recommendations ⇒ ~ 38% more click through.

An “often-cited” Problem Characterization

Given ..

- **The profile of the "active" user and possibly some situational context.**

Compute ..

- **A relevance (ranking) score for each recommendable item.**

The Profile:

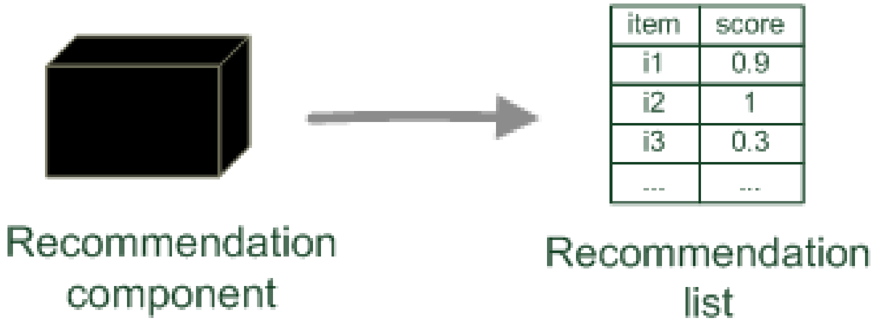
- **Can include past user ratings (explicit or implicit), demographics and interest scores for item features.**

The Problem:

- **is to learn a function that predicts the relevance score for a given *(typically unseen)* item.**

Models of Recommender Systems

Recommender systems reduce information overload by estimating relevance.

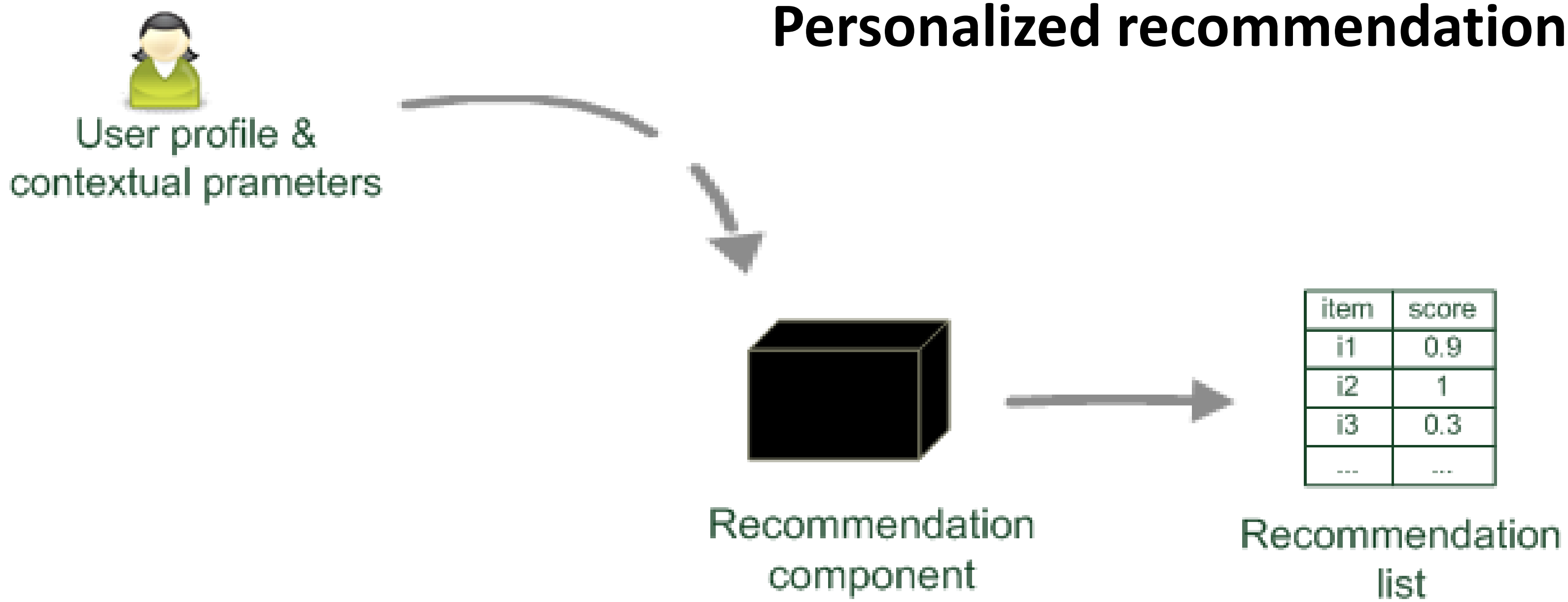


Usefulness of Recommendations; Is it worthwhile? It depends ..

- Is there s “large” number of items?
- Do users know exactly what are they looking for?

Models of Recommender Systems

Personalized recommendations



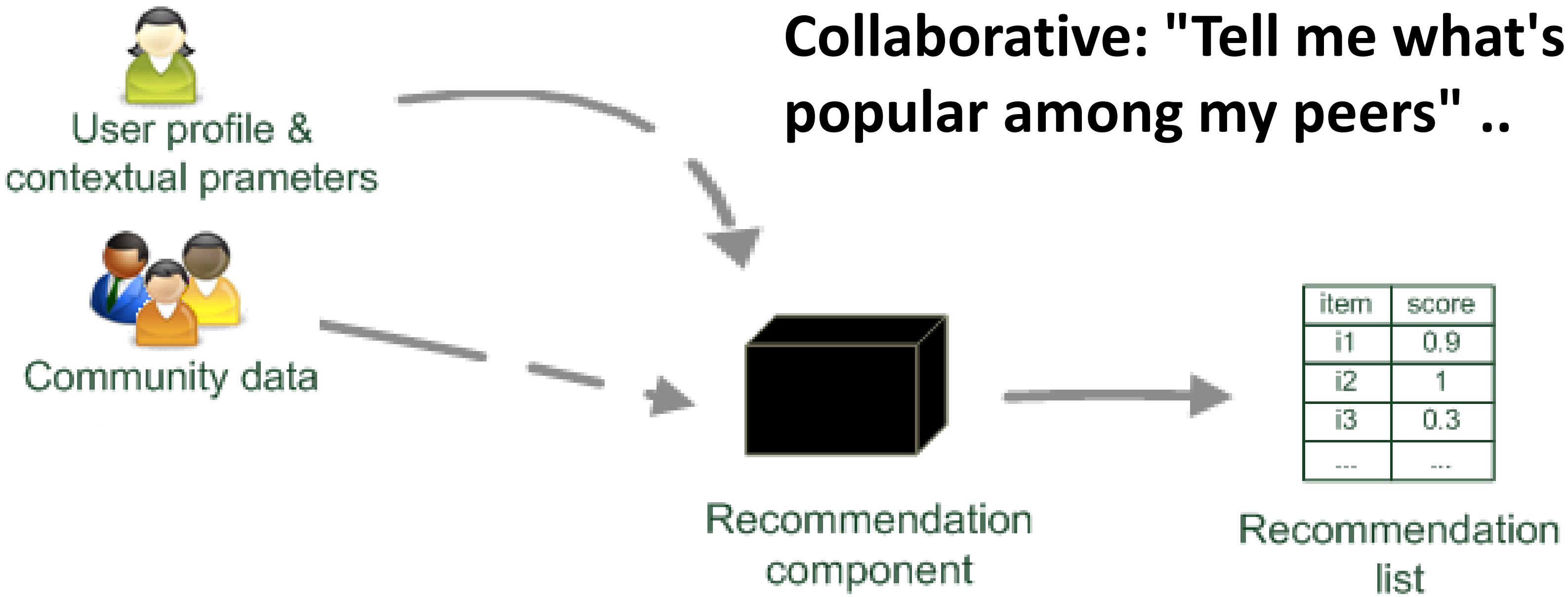
Implementing Personalized Systems is Difficult;

(always) difficult debugging, testing, evaluation ..

.. *personalization* ⇒ different behaviour for each user, thus, hard to distinguish bugs and surprising results ..

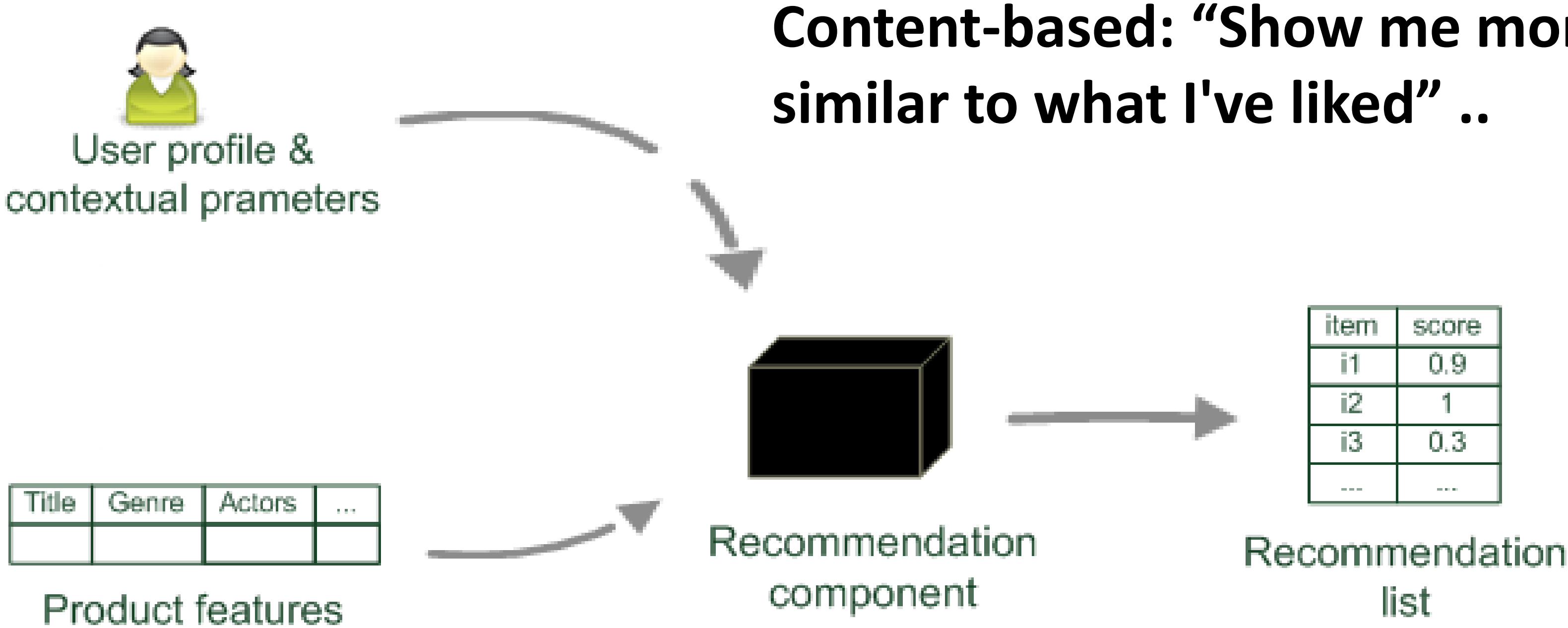
Models of Recommender Systems

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



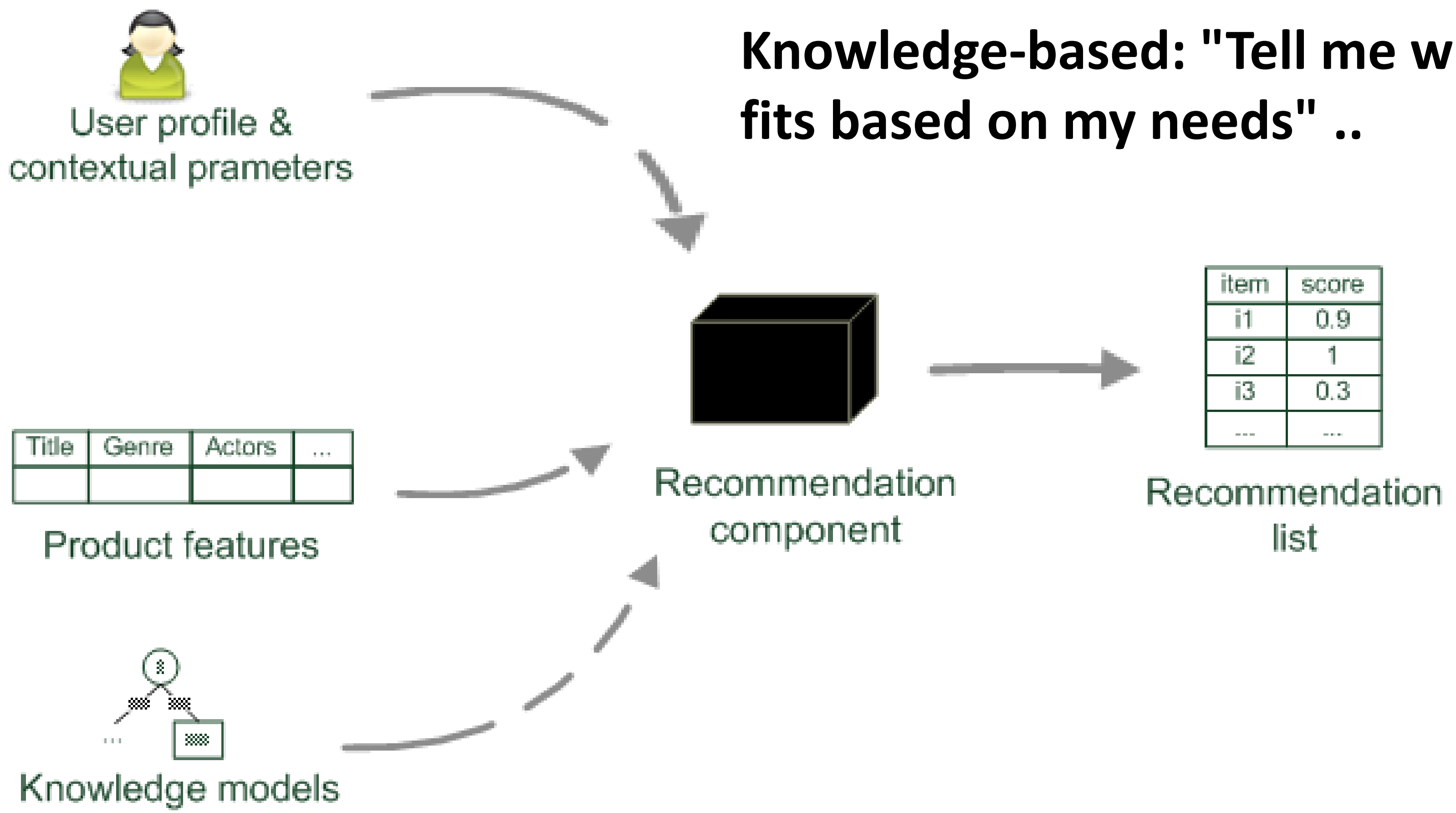
Models of Recommender Systems

Content-based: “Show me more similar to what I've liked” ..



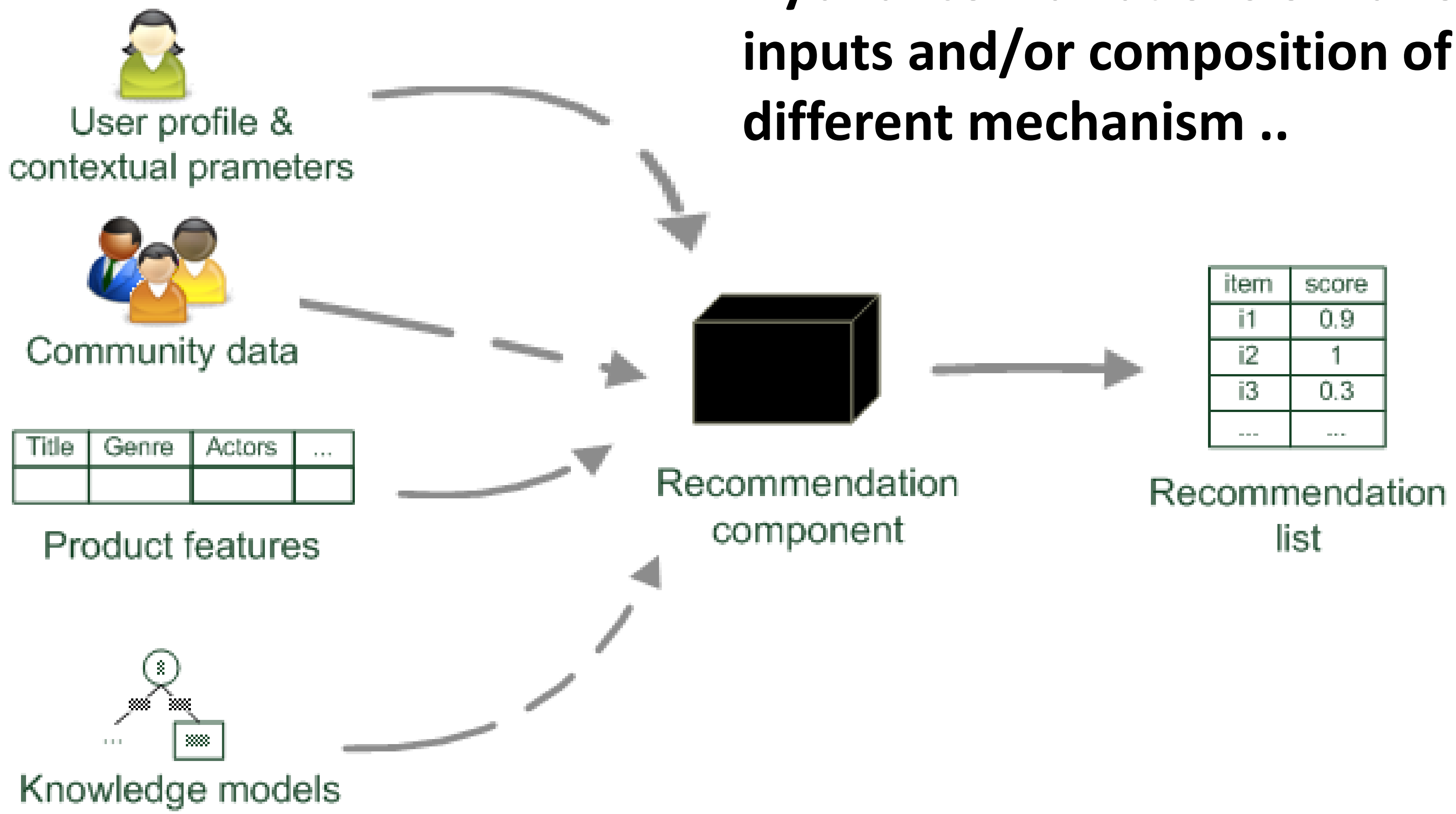
Models of Recommender Systems

Knowledge-based: "Tell me what fits based on my needs" ..



Models of Recommender Systems

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



Recommender Systems .. *versus* .. Information Retrieval

- **Information retrieval (IR)** is the activity of obtaining information resources relevant to an information need from a collection of information resources. *(Wikipedia)*

- The goal of a **Recommender System** is to generate meaningful recommendations to a collection of users for items or products that might interest them. *(Melville, Sindhwani)*

Recommender Systems and Information Retrieval ***are closely connected (many similar techniques)***, yet ..

.. different goals:

- Information Retrieval – *"I know what I'm looking for"*
- Recommender Systems – *"I'm not sure what I'm looking for"*

Collaborative Filtering .. & .. Rating

“tell me what’s popular among my peers (= similar users)”

- **one of the most often and successfully used techniques widely applicable, does not need any domain knowledge.**

Collaborative Filtering .. & .. Rating

Recommender systems *(particularly collaborative filtering)* rely on user "ratings" ..

- **Rating of item ~ how much the user likes the item.**

There're many different forms of ratings, for example:

- **Explicit**
  - Like-it scale (5 stars), like/dislike require additional effort from users.
- **Implicit**
  - Click through rate, buying an item, visiting a page, viewing a video, dwell time ..
  - Easier to collect, less precise ..
  - More "honest" *(Netflix example: highly rated vs watched).*

Potential Downsides

An example .. **Personalization in general, collaborative filtering specifically "filter bubbles" news, social media .. That is, users only see what they are expected to like.**

- **Good for business** *(in the short term)* **..**
- **Potentially bad** *(in the long term)* **for users and society ..**

However, ..

- Do not "throw away" collaborative filtering techniques, just be aware of the limitations ..
- Try to address limitations in a suitable way *(depending on the application)* ..

- Collaborative Filtering (CF)
  - User-based Nearest Neighbor CF
  - Pearson Correlation
  - Item-based CF
- Content-based Recommendation
  - Term-Frequency (TF)
  - Inverse Document Frequency (IDF)
  - Improving the Vector Space Model
  - Cosine Similarity

Various Paradigms *that will be used in* Recommendation Systems

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 and idea:

- Users give ratings to catalog items (implicitly or explicitly).
- Customers who had similar tastes in the past, will have similar tastes in the future.

Pure *Collaborative Filtering (CF)* **Approaches**

Input:

- Only a matrix of given user–item ratings.

Output types:

- A (numerical) prediction indicating to what degree the current user will like or dislike a certain item.
- A top-N list of recommended items.

User-based Nearest-Neighbor *Collaborative Filtering ..*

The basic technique:

- Given an "active user" (e.g. Alice) and an item i not yet seen by Alice:

1. Find a set of users (peers/nearest neighbors) who liked the same items as Alice in the past **and** who have rated item i.
2. Use, e.g. the average of their ratings to predict, if Alice will like item i.
3. Do this for all items Alice has not seen and recommend the best-rated.

Basic assumption and idea:

- If users had similar tastes in the past they will have similar tastes in the future.
- User preferences remain stable and consistent over time.

User-based Nearest-Neighbor
Collaborative Filtering ..

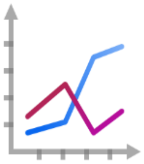
Example:

A database of ratings of the current user, Alice, and some other users is given:

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Determine whether Alice will like or dislike *Item5*, which Alice has not yet rated or seen.

User-based Nearest-Neighbor
Collaborative Filtering ..



Some initial questions:

- How do we measure similarity?
- How many neighbors should we consider?
- How do we generate a prediction from the neighbors' ratings?

	Item1	Item2	Item3	Item4	Item5
Alice	**5**	**3**	**4**	**4**	**?**
User1	**3**	**1**	**2**	**3**	**3**
User2	**4**	**3**	**4**	**3**	**5**
User3	**3**	**3**	**1**	**5**	**4**
User4	**1**	**5**	**5**	**2**	**1**

Measuring User Similarity: Pearson Correlation

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

$\boldsymbol{a, b}$: users

$\boldsymbol{r_{a,p}}$: rating of user a for item p

$\boldsymbol{P}$: set of items, rated both by a and b

Possible similarity values between -1 and 1 ..

$$\boldsymbol{sim(a,b)} = \frac{\sum_{\boldsymbol{p} \in \boldsymbol{P}}(\boldsymbol{r_{a,p}} - \boldsymbol{\bar{r}_a})(\boldsymbol{r_{b,p}} - \boldsymbol{\bar{r}_b})}{\sqrt{\sum_{\boldsymbol{p} \in \boldsymbol{P}}(\boldsymbol{r_{a,p}} - \boldsymbol{\bar{r}_a})^2}\sqrt{\sum_{\boldsymbol{p} \in \boldsymbol{P}}(\boldsymbol{r_{b,p}} - \boldsymbol{\bar{r}_b})^2}}$$

Measuring User Similarity: Pearson Correlation

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

$\boldsymbol{a, b}$: users

$\boldsymbol{r_{a,p}}$: rating of user a for item p

$\boldsymbol{P}$: set of items, rated both by a and b

Possible similarity values between -1 and 1 ..

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0,85

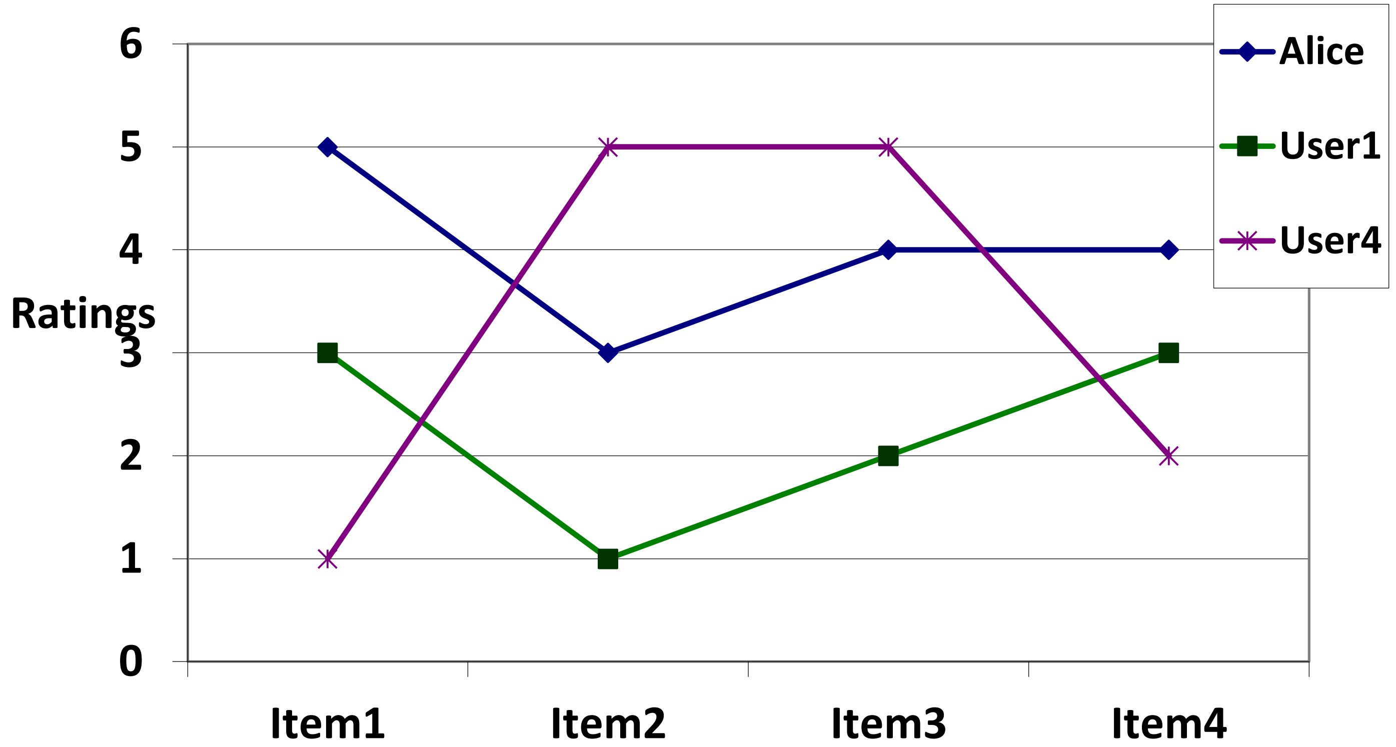
sim = 0,00

sim = 0,70

sim = -0,79

Measuring User Similarity: Pearson Correlation

Takes differences in rating behavior into account ..



Works well in usual domains, compared with alternative measures such as cosine similarity.

Making Predictions

A common prediction function:

$$\boldsymbol{pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}}$$

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

Item-based *Collaborative Filtering*

Basic idea:

- Use the similarity between items (and not users) to make predictions.

Example:

- Look for items that are similar to Item5.
- Take Alice's ratings for these items to predict the rating for Item5.

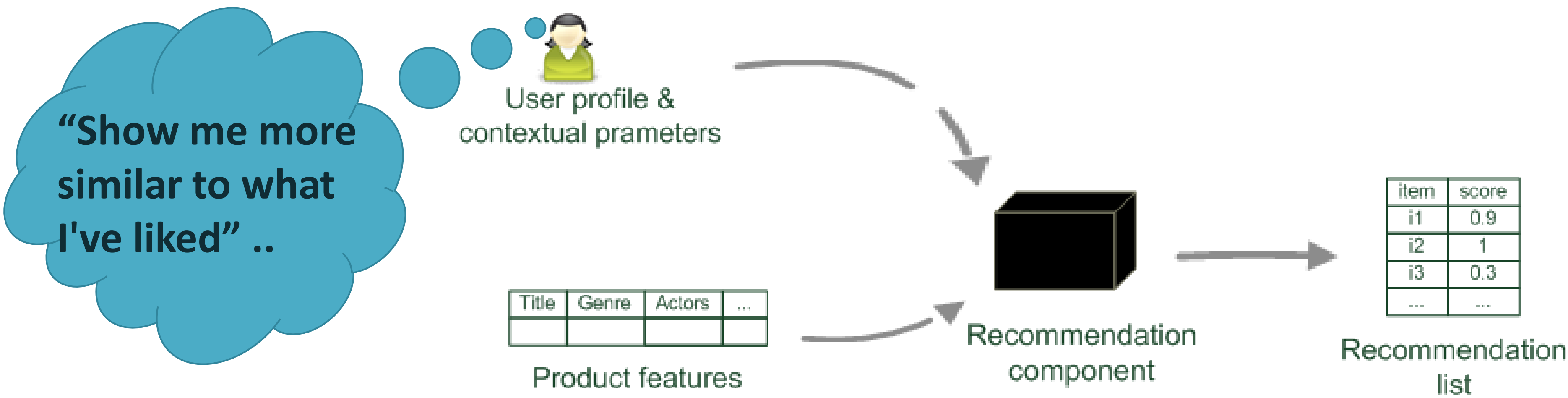
	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

Content-based Recommendation

- **While CF – methods do not require any information about the items,**
  - It might be reasonable to exploit such information; and ..
  - Recommend fantasy novels to people who liked fantasy novels in the past.
- **What do we need:**
  - Some information about the available items such as the genre ("content").
  - Some sort of user profile describing what the user likes (the preferences).

Content-based Recommendation

- **The task:**
  - Learn user preferences.
  - Locate/recommend items that are "similar" to the user preferences.



What is the “Content” ?

Most CB-recommendation techniques were applied to recommending text documents.

- Like web pages or newsgroup messages for example.

Content of items can also be represented as text documents.

- With textual descriptions of their basic characteristics.
- Structured: Each item is described by the same set of attributes

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

Content Representation .. & .. Item Similarities ..

- **Item Representation:**

Title	Genre	Author	Type	Price	Keywords
The Night of the Gun	Memoir	David Carr	Paperback	29.90	Press and journalism, drug addiction, personal memoirs, New York
The Lace Reader	Fiction, Mystery	Brunonia Barry	Hardcover	49.90	American contemporary fiction, detective, historical
Into the Fire	Romance, Suspense	Suzanne Brockmann	Hardcover	45.90	American fiction, murder, neo-Nazism

- **User profile:**

Title	Genre	Author	Type	Price	Keywords
…	Fiction	Brunonia, Barry, Ken Follett	Paperback	25.65	Detective, murder, New York

$keywords(b_j)$ describes Book b_j with a set of keywords.

Simple approach:

Compute the similarity of an unseen item with the user profile based on the keyword overlap (e.g. using the Dice coefficient), or use and combine multiple metrics.

$$\frac{2 \times |keywords(b_i) \cap keywords(b_j)|}{|keywords(b_i)| + |keywords(b_j)|}$$

Term-Frequency – Inverse Document Frequency ($TF - IDF$)

Simple keyword representation has its problems, in particular when automatically extracted, as:

- Not every word has similar importance
- Longer documents have a higher chance to have an overlap with the user profile

Standard measure: TF-IDF

- **Encodes text documents in multi-dimensional Euclidian space**
  - Weighted term vector.
- **TF: Measures, how often a term appears *(density in a document)*.**
  - Assuming that important terms appear more often.
  - Normalization has to be done in order to take document length into account.
- **IDF: Aims to reduce the weight of terms that appear in all documents.**

Term-Frequency – Inverse Document Frequency ($TF - IDF$)

Given a keyword i and a document j . .

$\boldsymbol{TF(i,j)}$: term frequency of keyword i in document j ..

$\boldsymbol{IDF(i)}$: inverse document frequency calculated as $\boldsymbol{IDF(i) = log\,\frac{N}{n(i)}}$

- $\boldsymbol{N}$: number of all recommendable documents.
- $\boldsymbol{n(i)}$: number of documents from N in which keyword i appears.

$\boldsymbol{TF - IDF}$ is calculated as $\boldsymbol{TF\text{-}IDF(i,j) = TF(i,j) * IDF(i)}$

Example: TF–IDF Representation

Term Frequency ***(TF)*****:**

Each document is a $\boxed{\text{count vector}}$ in $\mathbb{N}^{|v|}$

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	1.51	0	3	5	5	1
worser	1.37	0	1	1	1	0

Vector v with dimension $|v| = 7$

Example: TF–IDF Representation

Combined TF–IDF weights:

- **Each document is now represented by a real-valued vector of $TF - IDF$ weights $\in \mathbb{R}^{|v|}$**

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4					
Caesar	232					
Calpurnia	0					
Cleopatra	57					
mercy	1.51					
worser	1.37					

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	5.25	3.18	0	0	0	0.35
Brutus	1.21	6.1	0	1	0	0
Caesar	8.59	2.54	0	1.51	0.25	0
Calpurnia	0	1.54	0	0	0	0
Cleopatra	2.85	0	0	0	0	0
mercy	1.51	0	1.9	0.12	5.25	0.88
worser	1.37	0	0.11	4.15	0.25	1.95

Improving the Vector Space Model

- **Vectors are usually long and sparse ..**
  - **Remove Stop–Words**
    - They will appear in nearly all documents.
    - E.g., "a", "the", "on", …
  - **Use Stemming**
    - Aims to replace variants of words by their common stem
    - E.g., "went" → "go", "stemming" → "stem", …
  - **Size Cut-offs**
    - Only use top n most representative words to remove "noise" from data.
    - E.g., use top 100 words.

Improving the Vector Space Model

- **Use lexical knowledge .. Use more elaborate methods for feature selection:**
  - Remove words that are not relevant in the domain.
- **Detection of phrases as terms:**
  - More descriptive for a text than single words .
  - E.g., "United Nations“.
- **Limitations:**
  - Semantic meaning remains unknown.
  - Example: usage of a word in a negative context.
    - *"there is nothing on the menu that a vegetarian would like.."*
    - The word "vegetarian" will receive a higher weight then desired.

    an unintended match with a user interested in vegetarian restaurants.

Cosine Similarity

Usual similarity metric to compare vectors: Cosine similarity (angle)

- Cosine similarity is calculated based on the angle between the vectors:

$$sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|}$$

Adjusted cosine similarity

- Take average user ratings into account ($\bar{r}_u$), transform the original ratings.
- U: set of users who have rated both items a and b.

$$sim(\vec{a}, \vec{b}) = \frac{\sum_{u \in U}(r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,a} - \bar{r}_u)^2}\sqrt{\sum_{u \in U}(r_{u,b} - \bar{r}_u)^2}}$$

Recommending Items

Simple method: Nearest Neighbors ..

- **Given a set of documents D already rated by the user *(like/dislike)*:**
  - Either explicitly via user interface.
  - Or implicitly by monitoring user's behavior.
- **Find the n nearest neighbors of an not-yet-seen item i in D:**
  - Use similarity measures *(like cosine similarity)* to capture similarity of two documents.
- **Take these neighbors to predict a rating for i:**
  - E.g., $k = 5$ most similar items to i. Four of k items were liked by current user ➡ item i will also be liked by this user.

Thanks! . . . *Questions?*