Recommender Systems

Alexandros Karatzoglou
Research Scientist @ Telefonica Research, Barcelona
alexk@tid.es
@alexk_z

Telefonica Research in Barcelona

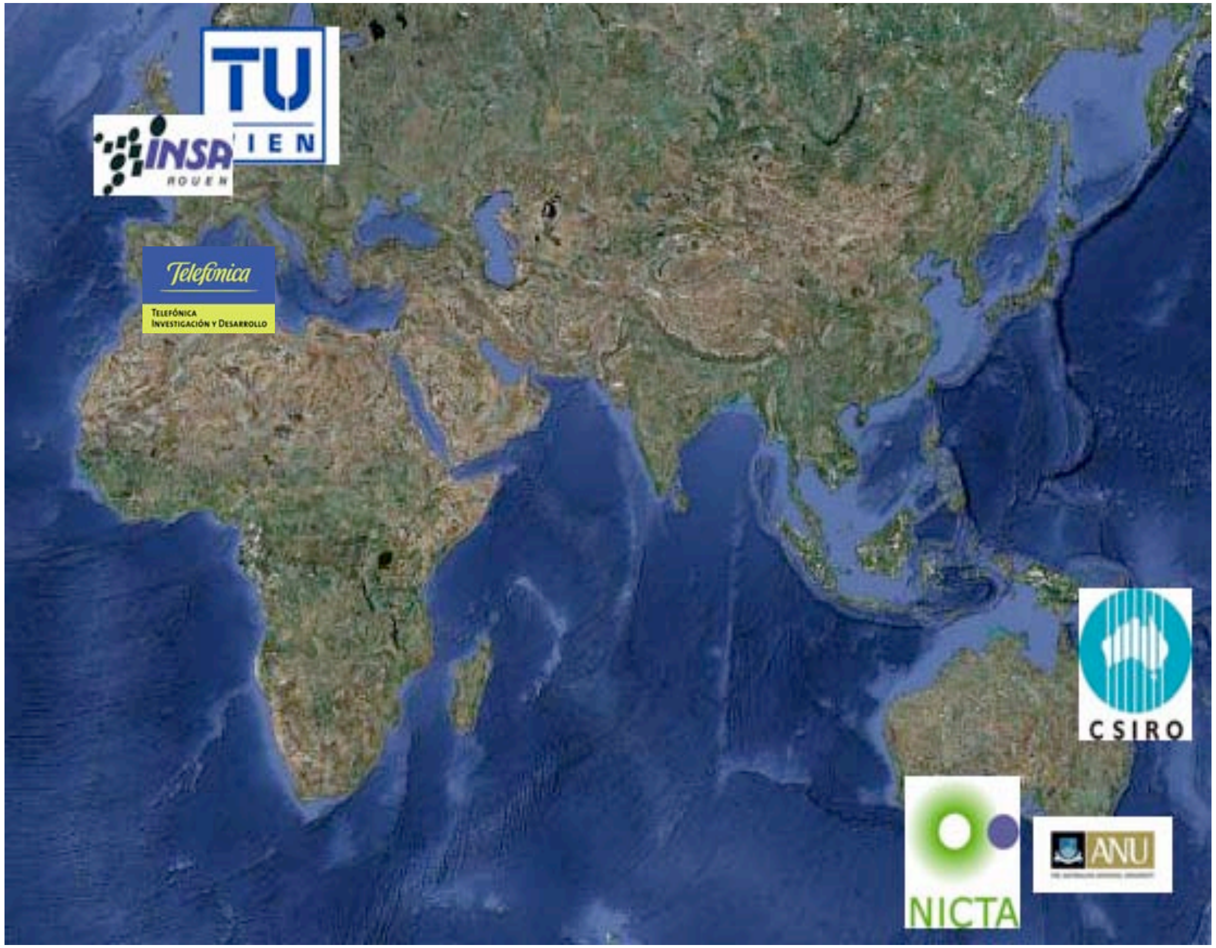
- Machine Learning & Recommender Systems
- Data Mining, Social Networks
- Multimedia Indexing & Analysis
- HCI
- System & Networking

We are looking for interns!

http://www.tid.es

Studies: Undergrad → PhD

Research


TU
IEN
INSA
ROUEN
Telefonica
Telefónica
Investigación y Desarrollo
CSIRO
NICTA
ANU

Recent Publications

CIKM 2013: *GAPfm: Optimal Top-N Recommendations for Graded Relevance Domains*

RecSys 2013: *xCLiMF: Optimizing Expected Reciprocal Rank for Data with Multiple Levels of Relevance*

ECML/PKDD 2013: Socially Enabled Preference Learning from Implicit Feedback Data

AAAI 2013 Workshop: Games of Friends: a Game-Theoretical Approach for Link Prediction in Online Social Networks

CIKM 2012: Climbing the App Wall: Enabling Mobile App Discovery through Context-Aware Recommendations

*RecSys 2012: CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering * Best Paper Award*

SIGIR 2012: TFMAP: Optimizing MAP for Top-N Context-aware Recommendation

NIPS 2011 Workshop: Collaborative Context-Aware Preference Learning

RecSys 2011: Collaborative Temporal Order Modeling

RecSys 2011: Implicit Feedback Recommendation via Implicit-to-Explicit Ordinal Logistic Regression Mapping

RecSys 2010: Multiverse Recommendation: N-dimensional Tensor Factorization for Context-Aware Collaborative Filtering

EC-Web 2010: Quantile Matrix Factorization for Collaborative Filtering

AISTATS 2010: Collaborative Filtering on a Budget

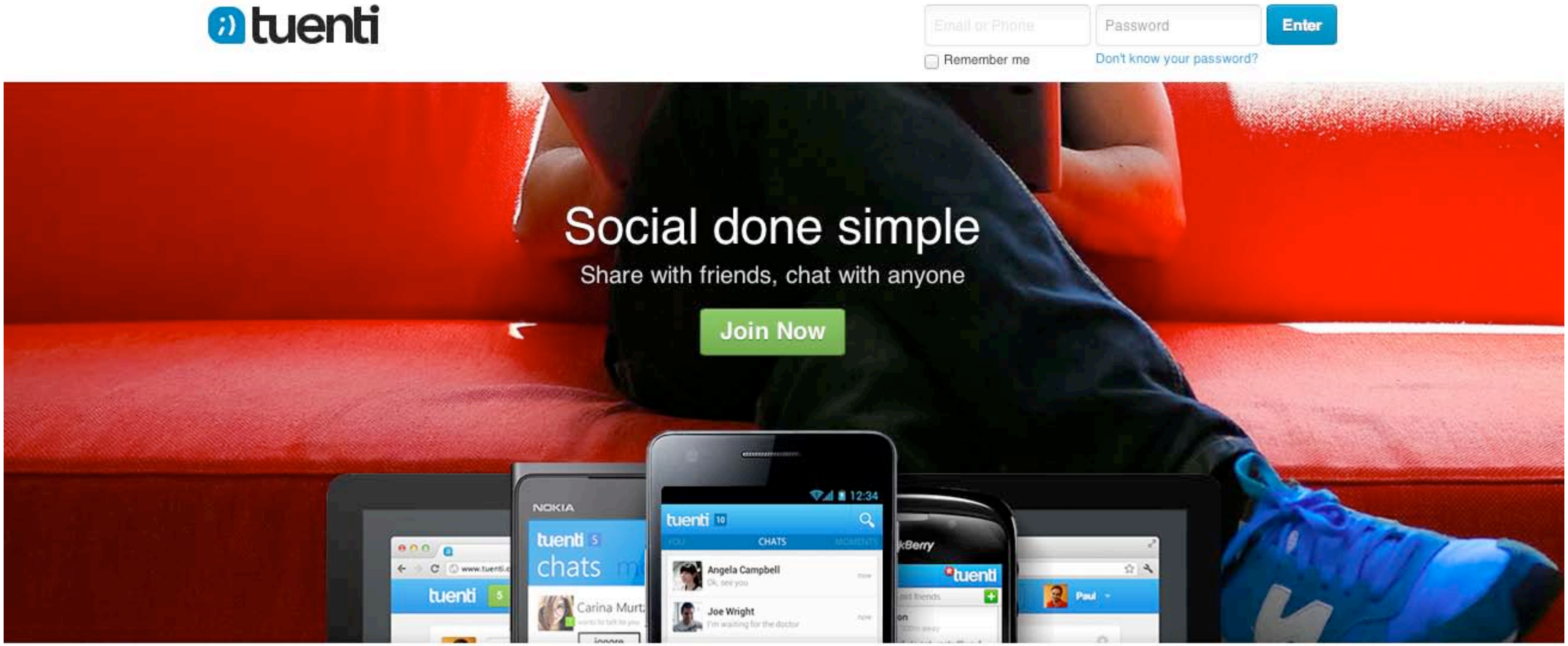
RecSys 2009: Maximum Margin Code Recommendation

RecSys 2008: Adaptive Collaborative Filtering

*Machine Learning Journal, 2008: Improving Maximum Margin Matrix Factorization * Best Machine Learning Paper Award at ECML PKDD 2008*

NIPS 2007: CoFiRank - Maximum Margin Matrix Factorization for Collaborative Ranking

Recommenders @Telefonica



Recommenders @Telefonica

Recommenders @Telefonica

Index

1. Introduction: What is a Recommender System?
2. Approaches
    1. Collaborative Filtering
    2. Content-based Recommendations
    3. Context-aware Recommendations
    4. Other Approaches
    5. Hybrid Recommender Systems
3. Research Directions
4. Conclusions
5. References

Index

1. Introduction: What is a Recommender System?
2. Approaches
    1. Collaborative Filtering
    2. Content-based Recommendations
    3. Context-aware Recommendations
    4. Other Approaches
    5. Hybrid Recommender Systems
3. Research Directions
4. Conclusions
5. References

From Search to Recommendation

"The Web is leaving the era of search and entering one of discovery. What's the difference?

***Search** is what you do when you're looking for something. **Discovery** is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you." – CNN Money, "The race to create a 'smart' Google*

The value of recommendations

- Netflix: 2/3 of the movies watched are recommended
- Google News: recommendations generate 38% more click-throughs
- Amazon: 35% sales from recommendations
- Choicestream: 28% of the people would buy more music if they found what they liked.

The “Recommender problem”

Estimate a **utility function**
to **predict** how
a user will **like** an item.

The “Recommender problem”

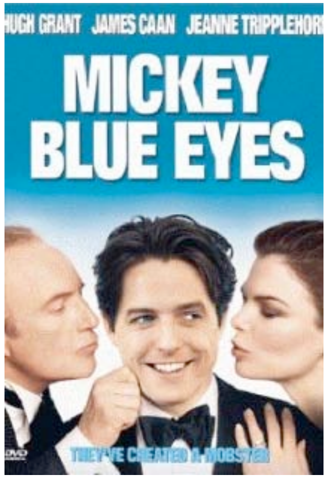
- C:= {users}
- S:= {recommendable items}
- u:= utility function, measures the usefulness of item s to user c,

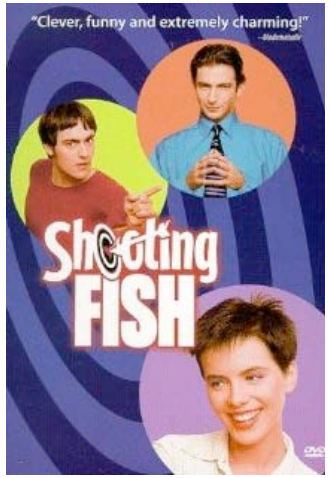
  $$u : C \times S \rightarrow R$$

  where R:= {recommended items}.
- For each user c, we want to choose the items s that maximize u.

$$c \in C \qquad s'_c = argmax_u u(c, s)$$

A good recommendation





is relevant to the user: *personalized*

A good recommendation

- is diverse:

it represents all the possible interests of one user

A good recommendation

- Does not recommend items the user already knows or would have found anyway.
- Expands the user's taste into neighboring areas.

***Serendipity** = Unsought finding*

Top k recommendations

Users take into account only few suggestions.
There is a need to do better on the top scoring recommended items

Index

1. Introduction: What is a Recommender System?
2. Approaches
    1. Collaborative Filtering
    2. Content-based Recommendations
    3. Context-aware Recommendations
    4. Other Approaches
    5. Hybrid Recommender Systems
3. Research Directions
4. Conclusions
5. References

What works?

- Depends on the domain and particular problem

- Currently, the best approach is Collaborative Filtering.
- Other approaches can be combined to improve results

- What matters?
  - Data preprocessing: outlier removal, denoising, removal of global effects
  - “Smart” dimensionality reduction
  - Combining methods

Collaborative Filtering

The task of **predicting** (filtering) user preferences on new items by **collecting** taste information from many users (collaborative).

Challenges:

- many items to choose from
- very few recommendations to propose
- few data per user
- no data for new user
- very large datasets

Index

1. Introduction: What is a Recommender System?
2. Approaches
    1. Collaborative Filtering:
        1. Memory-based CF
            1. User-based CF
            2. Item-based CF
        2. Model-based CF

Memory-Based CF:
User-based CF & Item-based CF

Example

2			4	5	
5		4			1
		5		2	
	1		5		4
		4		2	
4	5		1		

Each user has expressed an opinion for some items:

- Explicit opinion: rating score
- Implicit: purchase records or listen to tracks

Example: User-based CF

2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

Target (or Active) user for whom the CF recommendation task is performed

Example: User-based CF

2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

1. Identify set of items rated by the target user

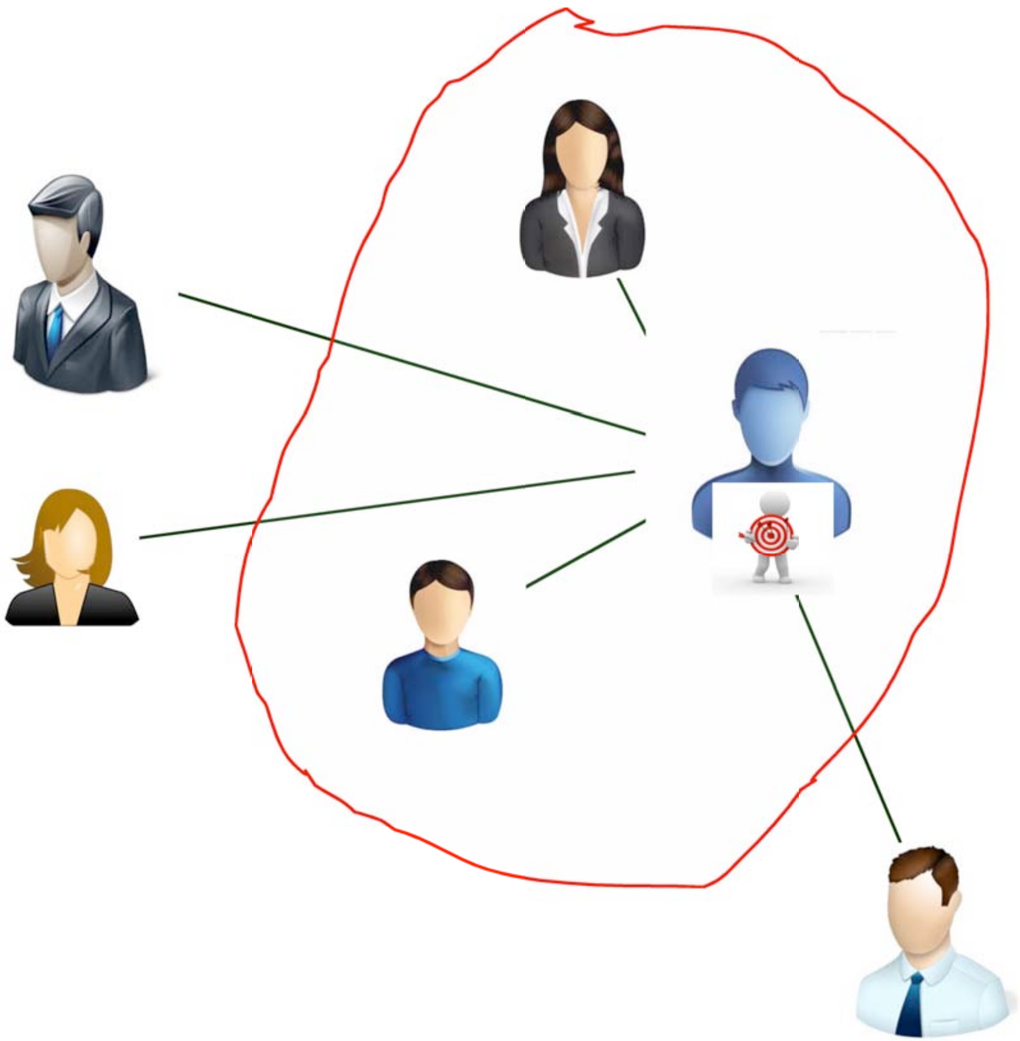
Example: User-based CF

2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

1. Identify set of items rated by the target user

2. Identify which other users rated 1+ items in this set (neighborhood formation)

User-based Similarity



3. Compute how similar each neighbor is to the target user (similarity function)

4. In case, select k most similar neighbors

User-based CF

5. Predict ratings for the target user's unrated items (prediction function)

6. Recommend to the target user the top N products based on the predicted ratings

User-based CF

- Target user u, ratings matrix Y
- Yv,i → rating by user v for item i
- Similarity Pearson r correlation sim(u,v) between users u & v

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} (y_{u,i} - \hat{y}_u)(y_{v,i} - \hat{y}_v)}{\sqrt{\sum_{i \in I_{uv}} (y_{u,i} - \hat{y}_u)^2 \sum_{i \in I_{uv}} (y_{v,i} - \hat{y}_v)^2}}$$

- Predicted rating $y^*(u,i)$

$$y^*(u,i) = \hat{y}_u + \frac{\sum_{j \in I_{y_{*j} \neq 0}} sim(v_j,u)(y_{v_j,i} - \hat{y}_{v_j})}{\sum_{j \in I_{y_{*j} \neq 0}} |sim(v_j,u)|}$$

Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	
		5		2		
	1		5		4	
		4			2	
4	5		1			NA

Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		
	1		5		4	
		4			2	
4	5		1			NA

Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	
		4			2	
4	5		1			NA

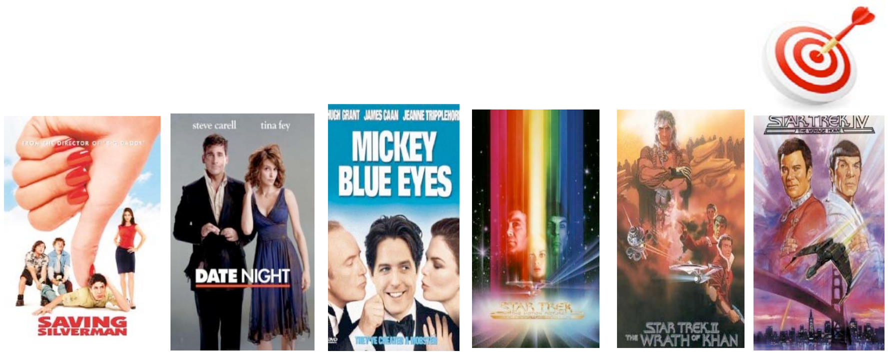
Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	-1
		4			2	
4	5		1			NA

Example: User-based CF

						sim(u,v)
2			4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	-1
3.51*	3.81*	4	2.42*	2.48*	2	
4	5		1			NA

Example: Item-based CF



2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

Target item: item for which the CF prediction task is performed.

Item-based CF

The basic steps:

- Identify set of users who rated the target item i
- Identify which other items (neighbours) were rated by the users set
- Compute similarity between each neighbour & target item (similarity function)
- In case, select k most similar neighbours
- Predict ratings for the target item (prediction function)

Item Based Similarity

Item Based Similarity

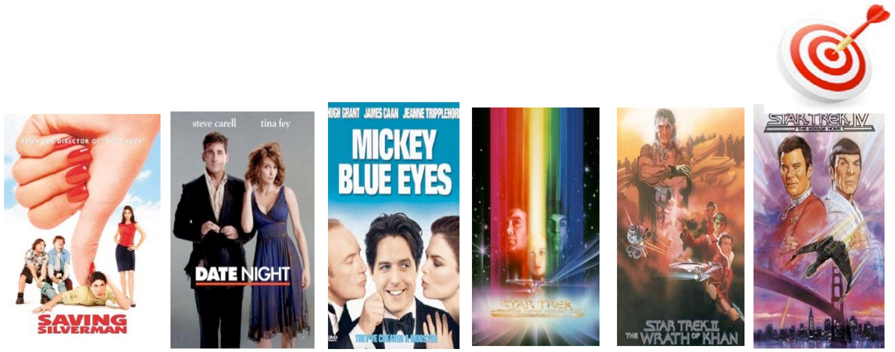
- Target item I
- Yu,j → rating of user u for item j, $\hat{y}_j$ average rating for j.
- Similarity sim(i,j) between items i and j (Pearson-correlation)

$$sim(i,j) = \frac{\sum_{u \in I_{ij}} (y_{u,i} - \hat{y}_i)(y_{u,j} - \hat{y}_j)}{\sqrt{\sum_{u \in I_{ij}} (y_{u,i} - \hat{y}_i)^2 \sum_{u \in I_{ij}} (y_{u,j} - \hat{y}_j)^2}}$$

- Predicted rating $y^*(u,i)$

$$y^*(u,i) = \hat{y}_i + \frac{\sum_{v \in I_{y_{u*} \neq 0}} sim(i,j_v)(y_{u,j_v} - \hat{y}_{j_v})}{\sum_{v \in I_{y_{u*} \neq 0}} |sim(i,j_u)|}$$

Example: Item-based CF



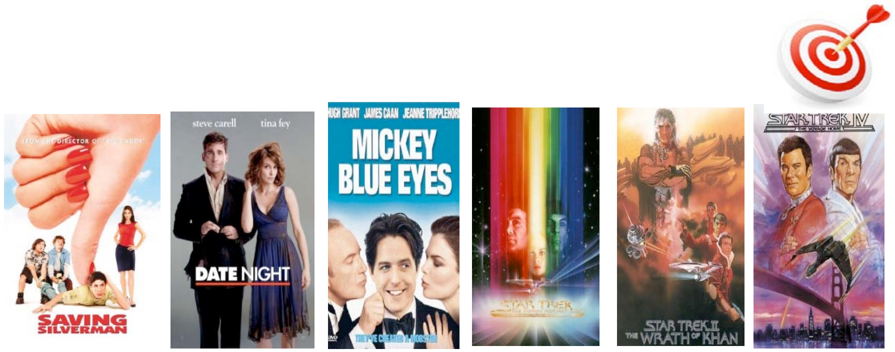
2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

sim(i,j) -1

Example: Item-based CF

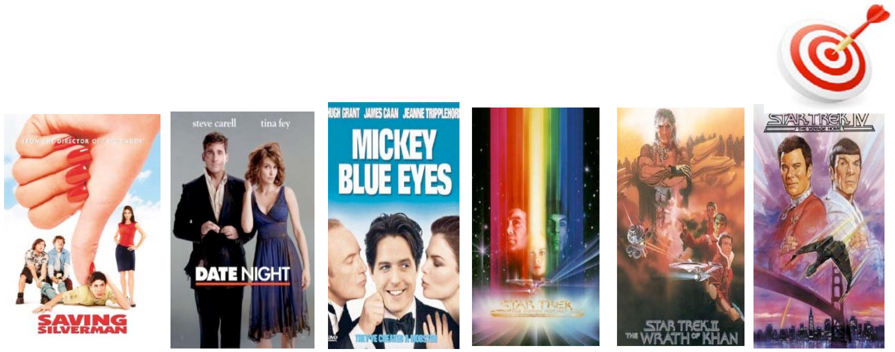
2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		
sim(i,j) -1	-1				

Example: Item-based CF



	2			4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		
sim(i,j)	-1	-1	0.86			

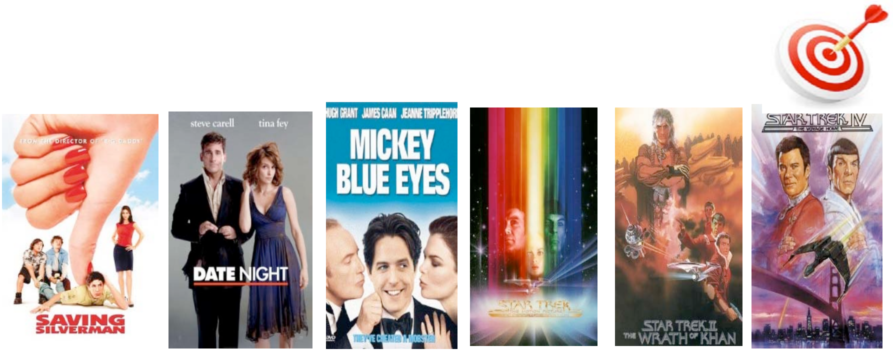
Example: Item-based CF



2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

sim(i,j) -1 -1 0.86 1

Example: Item-based CF

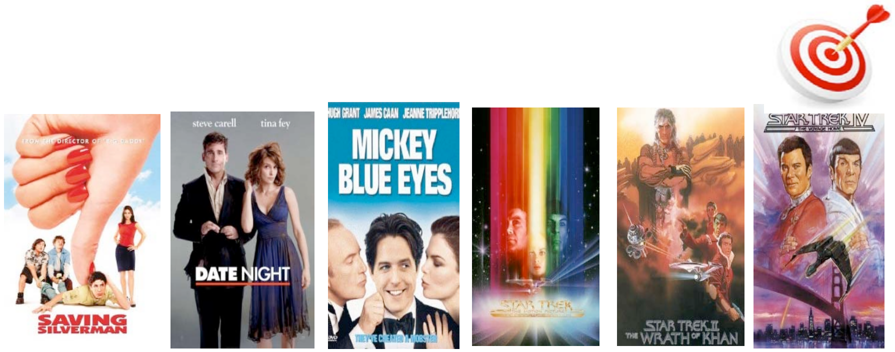


2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		
-1	-1	0.86	1	NA	

sim(i,j): -1, -1, 0.86, 1, NA

sim(6,5) cannot be calculated

Example: Item-based CF



	2			4	5	2.94*
	5		4			1
			5		2	2.48*
		1		5		4
			4			2
	4	5		1		1.12*
sim(i,j)	-1	-1	0.86	1	NA	

Item Similarity Computation

- ***Pearson r correlation-based Similarity***

  does *not* account for user rating biases

- **Cosine-based *Similarity***

  does *not* account for user rating biases

$$cos(i,j) = \frac{\sum_{u \in I_{ij}} y_{u,i} y_{u,j}}{\sqrt{\sum_{u \in I_{ij}} y_{u,i}^2 \sum_{u \in I_{ij}} y_{u,j}^2}}$$

- ***Adjusted Cosine Similarity***

  takes care of user rating biases as each pair in the co-rated set corresponds to a different user.

$$sim(i,j) = \frac{\sum_{u \in I_{ij}} (y_{u,i} - \hat{y}_u)(y_{u,j} - \hat{y}_u)}{\sqrt{\sum_{u \in I_{ij}} (y_{u,i} - \hat{y}_u)^2 \sum_{j \in I_{uv}} (y_{u,j} - \hat{y}_u)^2}}$$

Performance Implications

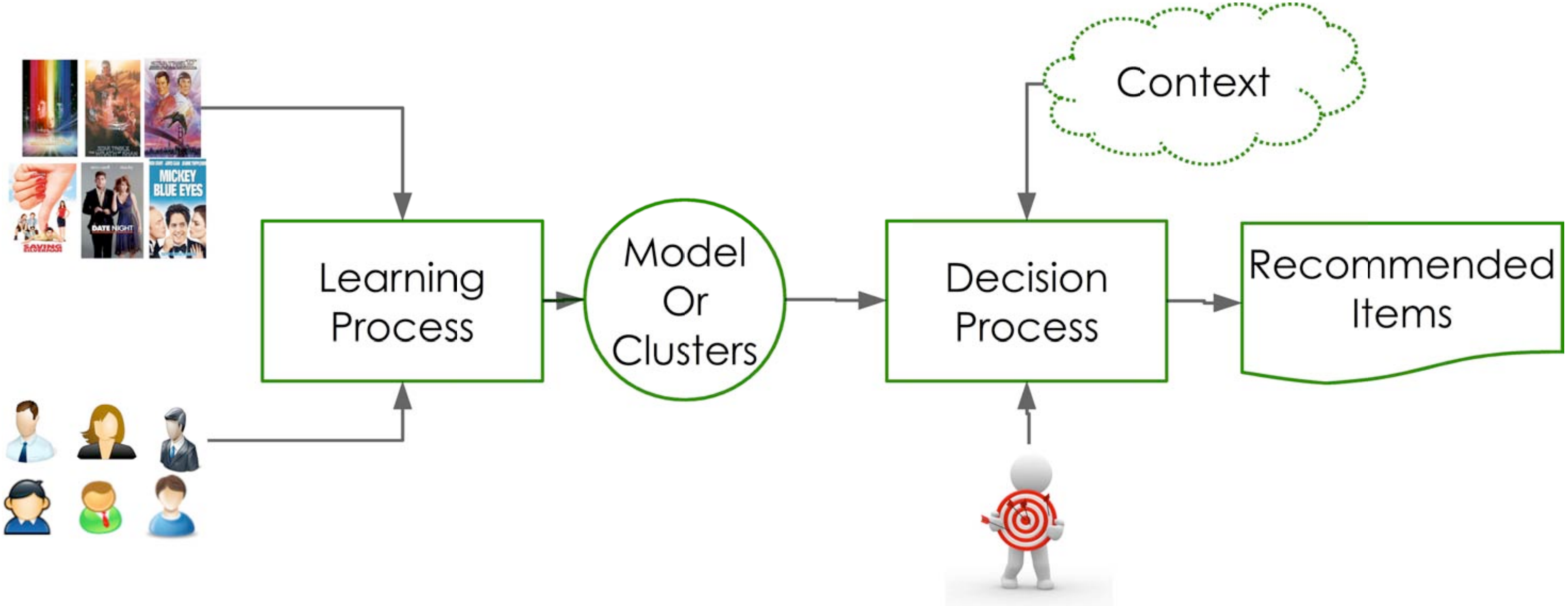
- Bottleneck: Similarity computation.

  Time complexity, highly time consuming with millions of users & items in the database.

- Two-step process:
  - "off-line component" / "model":

    similarity computation, precomputed & stored.
  - "on-line component": prediction process.

Two-step process



Offline **Online**

Performance Implications

- User-based similarity is more <u>dynamic</u>.

  Precomputing user neighbourhood can lead to poor predictions.

- Item-based similarity is <u>static</u>.

  We can precompute item neighbourhood. Online computation of the predicted ratings.

Memory based CF

+ Requires minimal knowledge engineering efforts

+ Users and products are symbols without any internal structure or characteristics

+ Produces good-enough results in most cases

- Requires a large number of explicit and reliable "ratings"

- Requires standardized products: users should have bought exactly the same product

- Assumes that prior behaviour determines current behaviour without taking into account "contextual" knowledge

Personalised vs Non-Personalised CF

- CF recommendations are personalized: the prediction is based on the ratings expressed by similar users; neighbours are different for each target user
- A non-personalized collaborative-based recommendation can be generated by averaging the recommendations of ALL users
- How would the two approaches compare?

Personalised vs Non-Personalised CF

Data Set	users	items	total	density	MAE Non Pers	MAE Pers
Jester	48483	100	3519449	0,725	0,220	0,152
MovieLens	6040	3952	1000209	0,041	0,233	0,179
EachMovie	74424	1649	2811718	0,022	0,223	0,151

Mean Average Error Non Personalized:

$$MAE_{NP} = \frac{\sum_{i,j} |v_{ij} - v_j|}{num.ratings}$$

v_{ij} is the rating of user i for product j and v_j is the average rating for product j

The Sparsity Problem

Typically large product sets & few user ratings e.g. Amazon:

- in a catalogue of 1 million books, the probability that two users who bought 100 books each, have a book in common is 0.01
- in a catalogue of 10 million books, the probability that two users who bought 50 books each, have a book in common is 0.0002

- CF must have a number of users ~ 10% of the product catalogue size

The Sparsity Problem

Methods for dimensionality reduction

- Matrix Factorization
- SVD
- Clustering

Model-Based Collaborative Filtering

Model Based CF Algorithms

Models are learned from the underlying data rather than heuristics.

Models of user ratings (or purchases):

- Clustering (classification)
- Association rules
- Matrix Factorization
- Restricted Boltzmann Machines
- Other models:
  - Bayesian network (probabilistic)
  - Probabilistic Latent Semantic Analysis ...

Clustering

- **Cluster** customers into categories based on preferences & past purchases

- **Compute** recommendations at the cluster level:

all customers within a cluster receive the same recommendations

Clustering

	BOOK 1	BOOK 2	BOOK 3	BOOK 4	BOOK 5	BOOK 6
CUSTOMER A	X			X		
CUSTOMER B		X	X		X	
CUSTOMER C		X	X			
CUSTOMER D		X				X
CUSTOMER E	X				X	

B, C & D form 1 CLUSTER vs. A & E form another cluster.

- « Typical » preferences for CLUSTER are:
  - Book 2, very high
  - Book 3, high
  - Books 5 & 6, may be recommended
  - (Books 1 & 4, not recommended)

Clustering

	BOOK 1	BOOK 2	BOOK 3	BOOK 4	BOOK 5	BOOK 6
CUSTOMER A	X			X		
CUSTOMER B		X	X		X	
CUSTOMER C		X	X			
CUSTOMER D		X				X
CUSTOMER E	X				X	

Clustering

+ It can also be applied for selecting the k most relevant neighbours in a CF algorithm

+ Faster: recommendations are per cluster

- less personalized: recommendations are per cluster vs. in CF they are per user

Association rules

Past purchases used to find relationships of common purchases

	BOOK 1	BOOK 2	BOOK 3	BOOK 4	BOOK 5	BOOK 6
CUSTOMER A	X			X		
CUSTOMER B		X	X		X	
CUSTOMER C		X	X			
CUSTOMER D		X				X
CUSTOMER E	X				X	
CUSTOMER F			X		X	

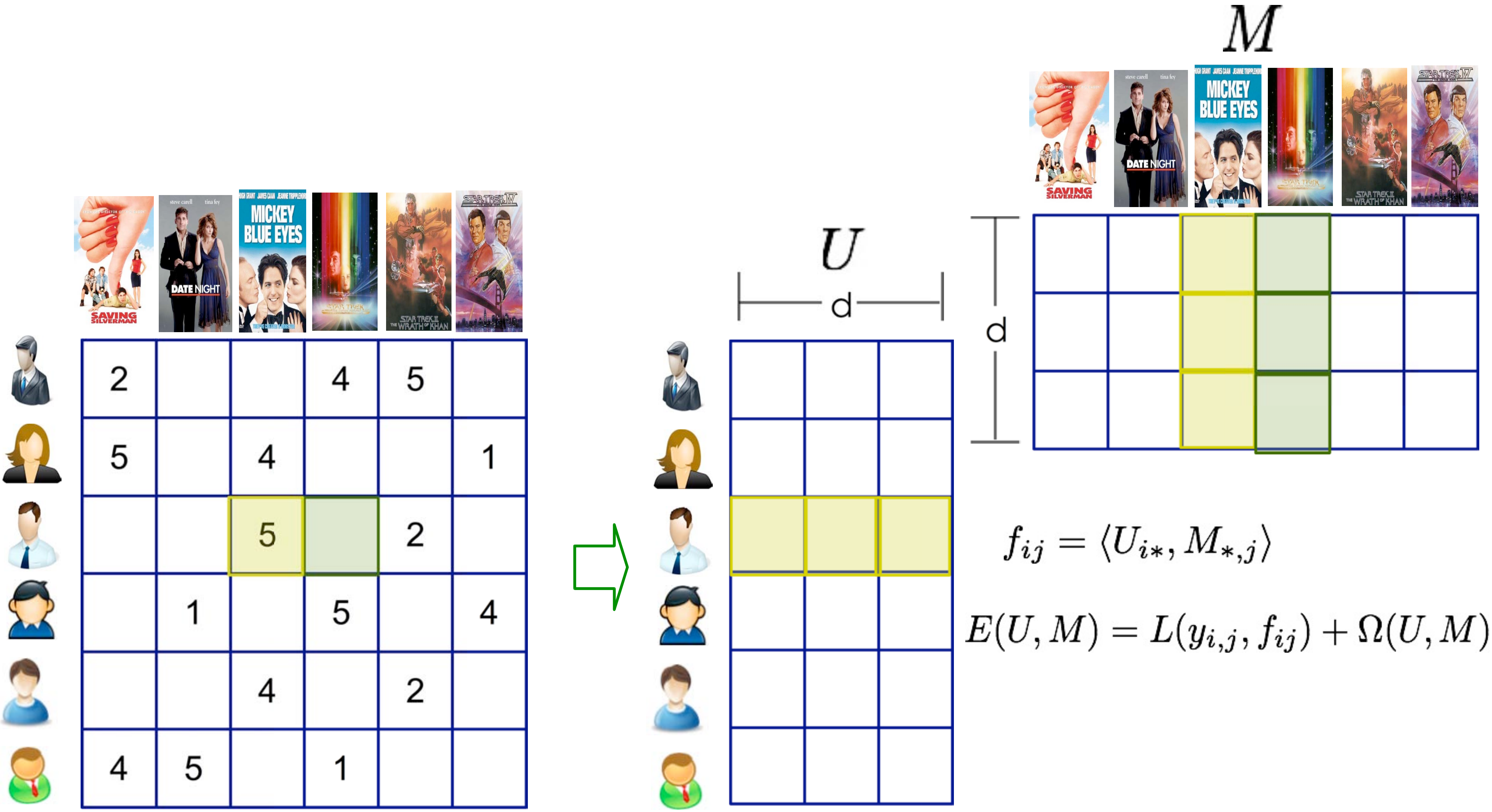
	BOOK 1	BOOK 2	BOOK 3	BOOK 4	BOOK 5	BOOK 6
BOOK 1				1	1	
BOOK 2			2		1	1
BOOK 3		2			2	
BOOK 4	1					
BOOK 5	1	1	2			
BOOK 6		1				

Association rules

+ Fast to implement
+ Fast to execute
+ Not much storage space required
+ Not « individual » specific
+ Very successful in broad applications for large populations, such as shelf layout in retail stores

- Not suitable if preferences change rapidly
- Rules can be used only when enough data validates them. False associations can arise

Matrix Factorization



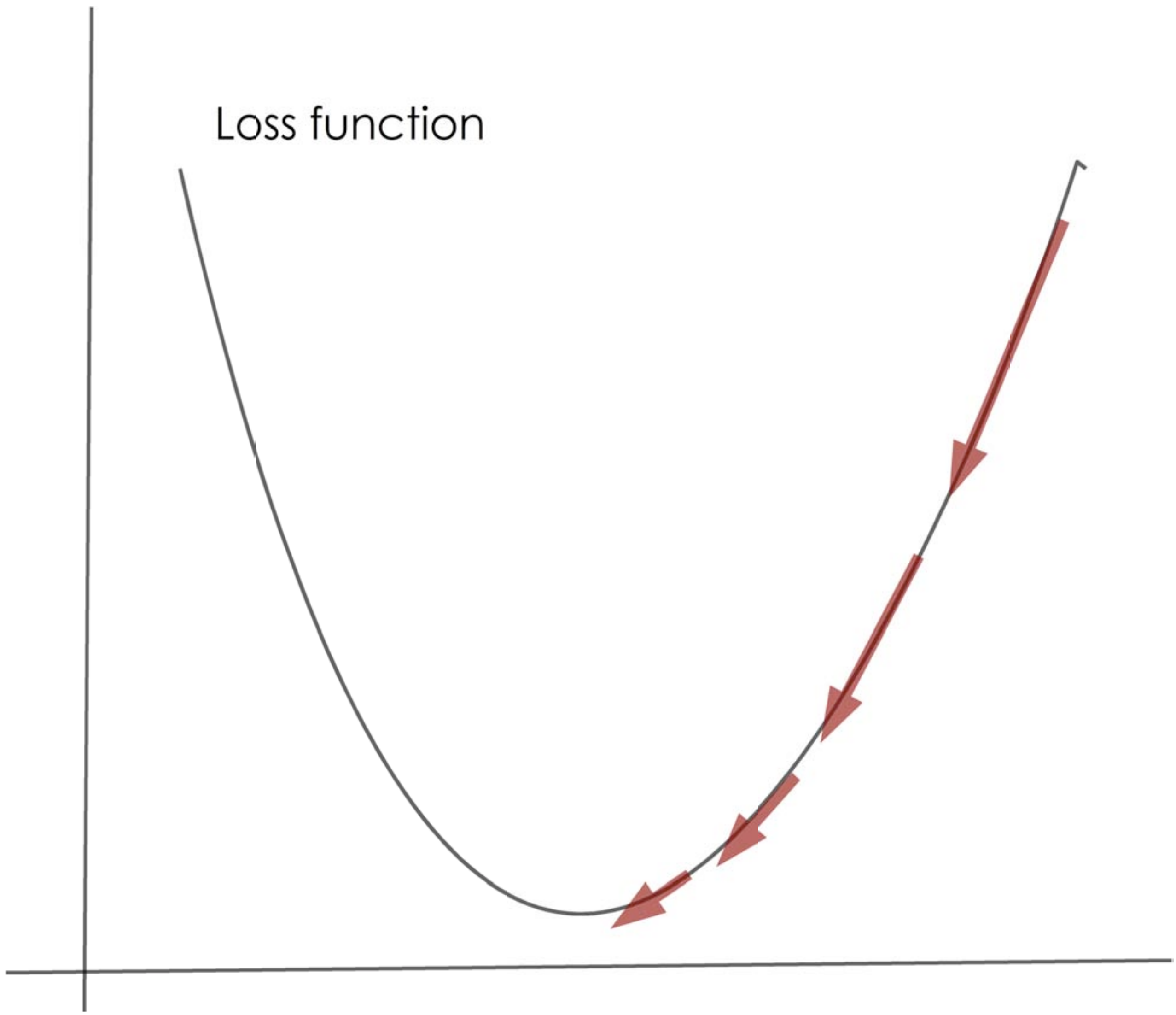
$$f_{ij} = \langle U_{i*}, M_{*,j} \rangle$$

$$E(U, M) = L(y_{i,j}, f_{ij}) + \Omega(U, M)$$

Loss Functions for MF

- Squared error loss: $L(y_{i,j}, f_{i,j}) = \frac{1}{2}(y_{i,j} - f_{i,j})^2$
- Mean Average Error: $L(y_{i,j}, f_{i,j}) = |y_{i,j} - f_{i,j}|$
- Binary Hinge loss: $L(y_{i,j}, f_{i,j}) = max(0, 1 - y_{i,j}, f_{i,j})$

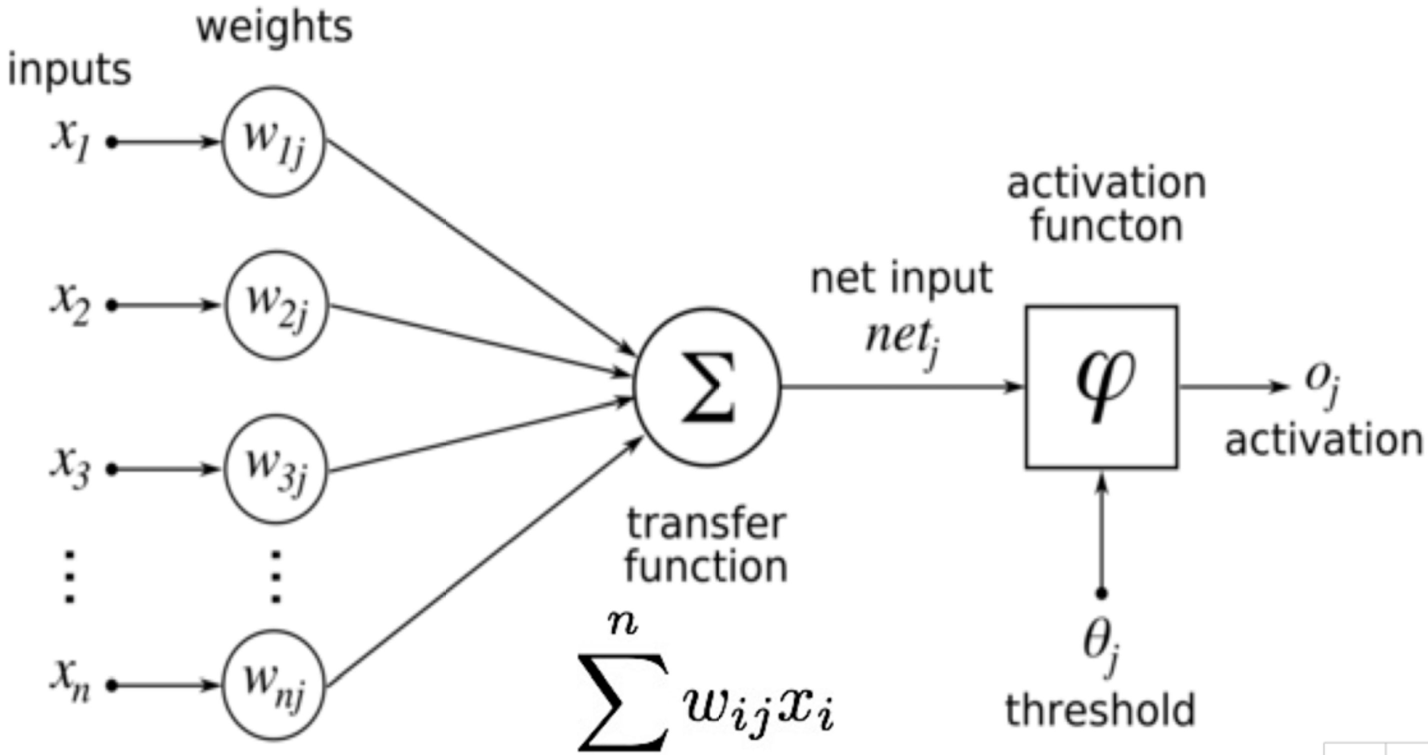
Learning: Stochastic Gradient Descent



Restricted Boltzmann Machines

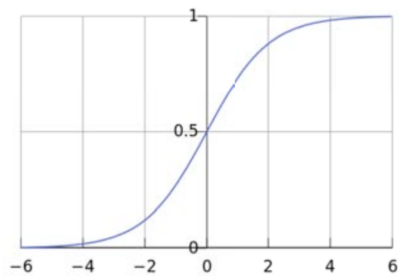
- A (generative stochastic) Neural Network
- Learns a probability distribution over its inputs
- Used in dimensionality reduction, CF, topic modeling, feature learning
- Essential components of Deep Learning methods (DBN's, DBM's)

Restricted Boltzmann Machines



$$\sum_{i=1}^{n} w_{ij} x_i$$

$$\phi = \frac{1}{1 + e^{-\Sigma}}$$



$$P(o_j = 1|x) = \phi\left(\sum_{i=1}^{n} w_{ij} x_i + \theta_j\right)$$

Restricted Boltzmann Machines

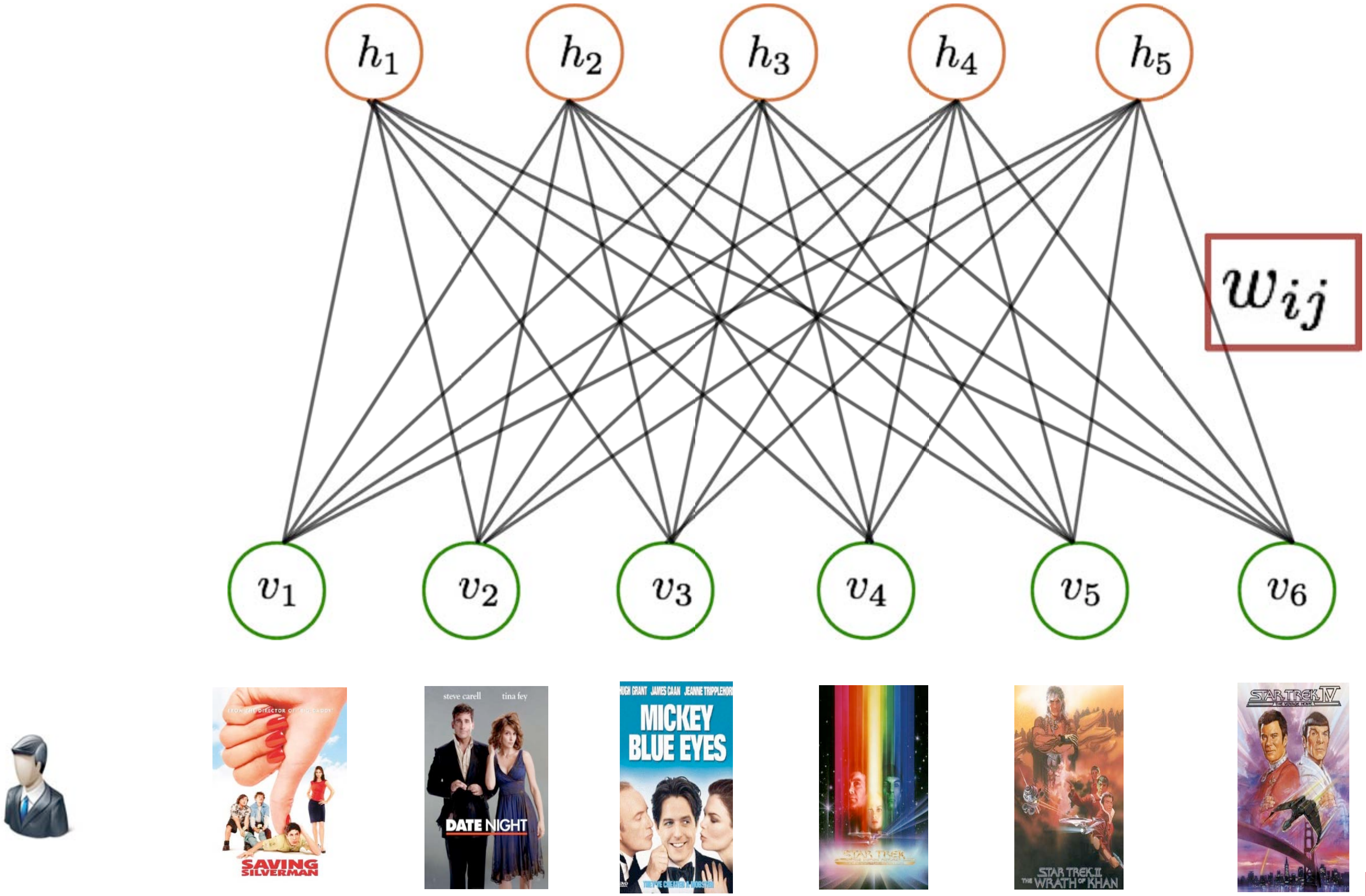
- Each unit is in a state which can be active or not active.
- Each input of a unit is associated to a weight
- The transfer function Σ calculates for each unit a score based on the weighted sum of the inputs
- This score is passed to the activation function ϕ which calculated the probability that the unit state is active.

Restricted Boltzmann Machines

- Each unit in the visible layer vi corresponds to one item
- The number of the hidden units hj is a parameter.
- Each vi is connected to each hj through a weight wij

- In the training phase, for each user:
  - if the user purchased the item the corresponding vi is activated.
  - The activation states of all vi are the input of each hj
  - Based on this input the activation state of each hj is calculated
  - The activation state of all hj become now the input of each vi
  - The activation state of each vi is recalculated
  - For each vi the difference between the present activation state
  - and the previous is used to update the weights wij and thresholds θj

Restricted Boltzmann Machines



Restricted Boltzmann Machines

In the prediction phase, using a trained RBM, when recommending to a user:

- For the items of the user the corresponding v_i is activated.
- The activation states of all v are the input of each h_j
- Based on this input the activation state of each h_j is calculated
- The activation state of all hj become now the input of each v_i
- The activation state of each v_i is recalculated
- The activation probabilities are used to recommend items

Limitations of CF

- Requires User-Item data:
  - It needs to have enough users in the system.
  - New items need to get enough ratings.
  - New users need to provide enough ratings (cold start)

- Sparsity:
  - it is hard to find users who rated the same items.

- Popularity Bias:
  - Cannot recommend items to users with unique tastes.
  - Tends to recommend popular items.

Cold-start

- New User Problem: the system must first learn the user's preferences from the ratings.
  - Hybrid RS, which combines content-based and collaborative techniques, can help.

- New Item Problem: Until the new item is rated by a substantial number of users, the RS is not able to recommend it.

Index

1. Introduction: What is a Recommender System?
2. Approaches
    1. Collaborative Filtering
    2. Content-based Recommendations
    3. Context-aware Recommendations
    4. Other Approaches
    5. Hybrid Recommender Systems
3. Research Directions
4. Conclusions
5. References

Content-Based Recommendations

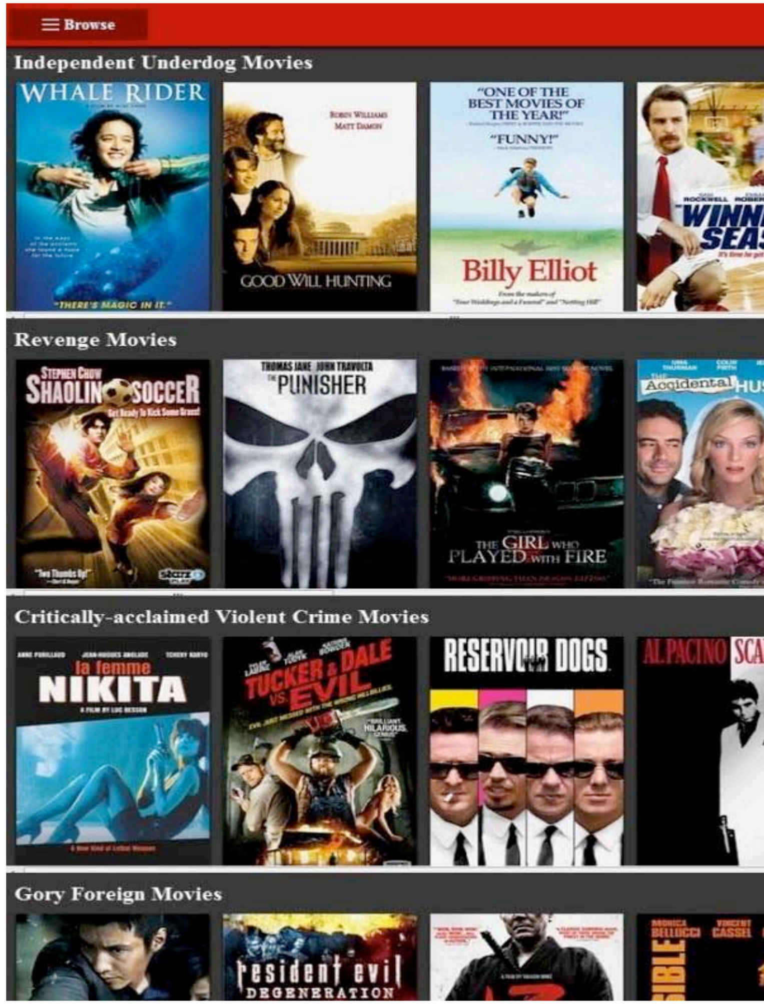
- Recommendations are based on the information on the ***content of items*** rather than on other users' opinions.
- Use a machine learning algorithm to model the users' preferences from examples based on a description of the content.

What is *content* of an item?

- Explicit attributes or characteristics

  e.g. for a movie:

  - Genre: Action / adventure
  - Feature: Bruce Willis
  - Year: 1995



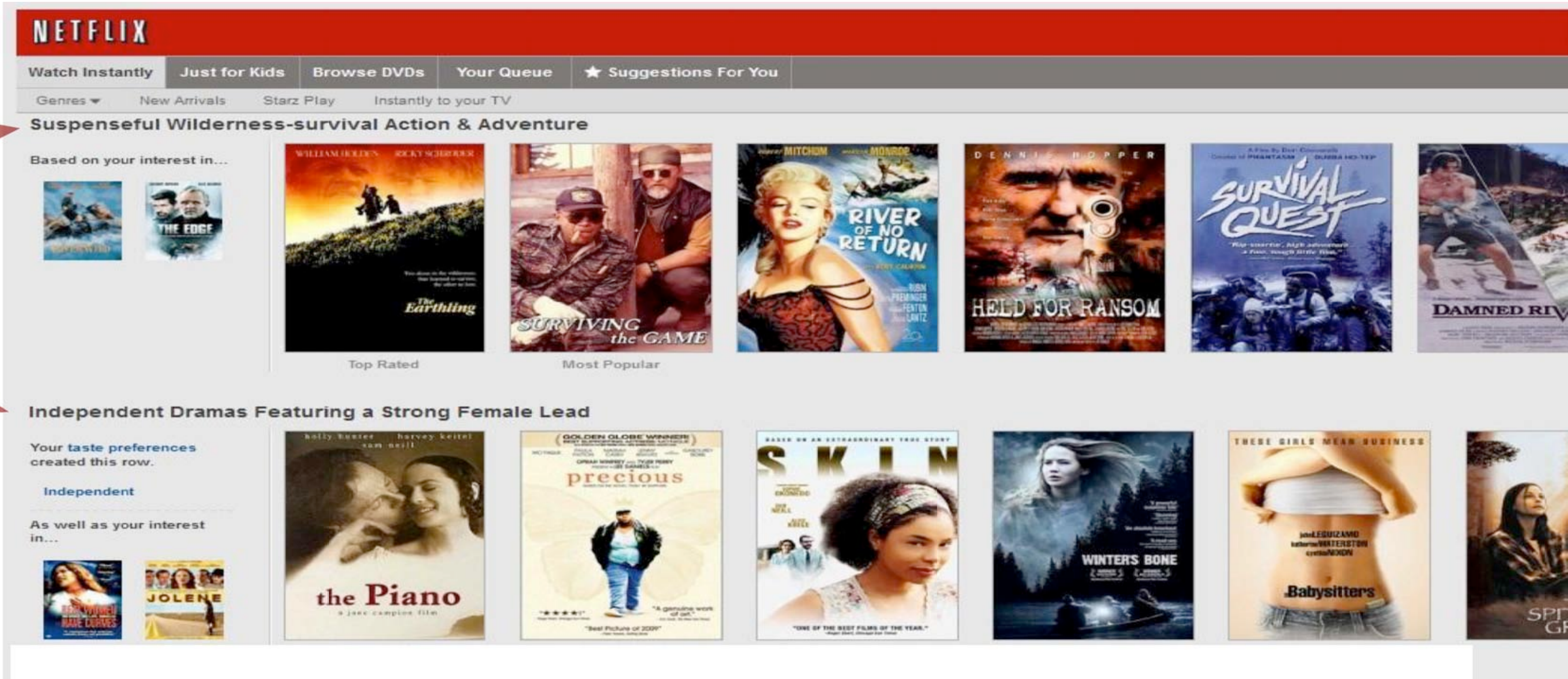
- Textual content

  e.g. for a book:

  - title,
  - description,
  - table of content

In Content-Based Recommendations...

The recommended items for a user are based on the profile built up by analysing the content of the items the user has liked in the past



Content-Based Recommendation

- Suitable for text-based products (web pages, books)
- Items are "described" by their features (e.g. keywords)
- Users are described by the keywords in the items they bought
- Recommendations based on the match between the content (item keywords) and user keywords
- The user model can also be a classifier (Neural Networks, SVM, Naïve Bayes...)

Advantages of CB Approach

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

Disadvantages of CB Approach

- Only for content that can be encoded as meaningful features.

- Some types of items (e.g. movies, music)are not amenable to easy feature extraction methods

- Even for texts, IR techniques cannot consider multimedia information, aesthetic qualities, download time: a positive rating could be not related to the presence of certain keywords

- Users' tastes must be represented as a learnable function of these content features.

- Hard to exploit quality judgements of other users.

- Difficult to implement serendipity

Content-based Methods

- *Content(s):= item profile, i.e.* a set of attributes/keywords characterizing item *s*.
- weight w_{ij} measures the 'Importance" (or "informativeness") of word k_j in document d_j
- *term frequency/inverse document frequency(TF-IDF)* is a popular weighting technique in IR

Content-based User Profile

- *ContentBasedProfile(c)*:= profile of user c
- profiles are obtained by:
  - analysing the content of the previous items
  - using keyword analysis techniques

e.g., ContentBasedProfile(c):=(wc_1, . . . , wc_k)

a vector of weights, where wc_i denotes the importance of keyword k_i to user c

Similarity Measurements

In content-based systems, the utility function u(c,s) is defined as:

$$u(c,s) = score(ContentBasedeProfile(c), Content(s))$$

where ContentBasedProfile(c) of user c and Content(s) of document s are both represented as TF-IDF vectors of keyword weights.

Similarity Measurements

Utility function *u(c,s)* usually represented by some scoring heuristic defined in terms of vectors, such as the cosine similarity measure.

$$u(c,s) = cos(w_c, w_s) = \frac{w_c \times w_s}{\|w_s\| \|w_c\|} = \frac{\sum_{i=1}^{K} w_{ic} w_{is}}{\sum_{i=1}^{K} w_{ic}^2 \sum_{i=1}^{K} w_{is}^2}$$

Content-based Recommendation. An (unrealistic) example

How to compute recommendations of books based only on their title?

- A customer buys the book: *Building data mining applications for CRM*
- 7 Books are possible candidates for a recommendation:

  Accelerating Customer Relationships: Using CRM and Relationship Technologies

  Mastering Data Mining: The Art and Science of Customer Relationship Management

  Data Mining Your Website

  Introduction to marketing

  Consumer behaviour

  Marketing research, a handbook

  Customer knowledge management

COUNT	a	Accelerating	and	applications	art	behavior	Building	Consumer	CRM	customer	data	for	Handbook	Introduction	Knowledge	Management	Marketing	Mastering	mining	of	relationship	Research	science	technology	the	to	using	website	your
Building data mining applications for CRM				1			1		1		1	1							1										
Accelerating customer relationships: using CRM and relationship technologies		1	1						1	1											2			1			1		
Mastering Data Mining: the art and science of Customer Relationship Management			1		1					1	1					1		1	1	1	1		1		1				
Data Mining your website											1								1									1	1
Introduction to Marketing														1			1									1			
Consumer behavior						1		1																					
Marketing Research: a Handbook	1												1				1					1							
Customer Knowledge Management										1					1	1													

Content-based Recommendation

- Computes distances between this book & all others
- Recommends the « closest » books:
  - **#1:** Data Mining Your Website
  - **#2:** Accelerating Customer Relationships: Using CRM and Relationship Technologies
  - **#3:** Mastering Data Mining: The Art and Science of Customer Relationship Management

TFIDF Normed Vectors	a	Accelerating	and	applications	art	behavior	Building	Consumer	CRM	customer	data	for	Handbook	Introduction	Knowledge	Management	Marketing	Mastering	mining	of	relationship	Research	science	technology	the	to	using	website	your
Building data mining applications for CRM				0.502			0.502		0.344		0.251	0.502							0.251										
Accelerating customer relationships: using CRM and relationship technologies		0.432	0.296						0.296	0.216											0.468			0.432			0.432		
Mastering Data Mining: the art and science of Customer Relationship Management			0.256		0.374					0.187	0.187					0.256		0.374	0.187	0.374	0.256		0.374		0.374				
Data Mining your website											0.316								0.316									0.632	0.632
Introduction to Marketing														0.636			0.436									0.636			
Consumer behavior						0.707		0.707																					
Marketing Research: a Handbook	0.537												0.537				0.368					0.537							
Customer Knowledge Management										0.381					0.736	0.522													

Index

1. Introduction: What is a Recommender System?
2. Approaches
    1. Collaborative Filtering
    2. Content-based Recommendations
    3. Context-aware Recommendations
    4. Other Approaches
    5. Hybrid Recommender Systems
3. Research Directions
4. Conclusions
5. References

Context

- Context is a dynamic set of factors describing the state of the user at the moment of the user's experience

- Context factors can rapidly change and affect how the user perceives an item

Context in Recommendations

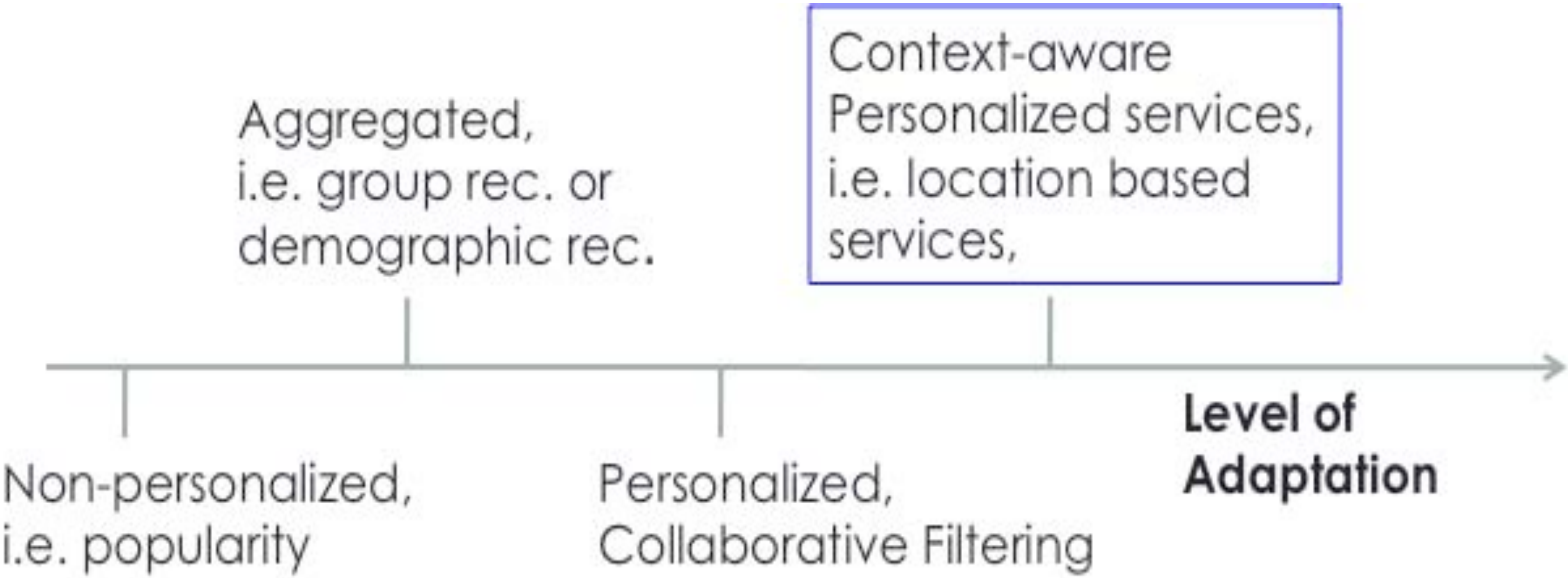
- Temporal: Time of the day, weekday/end

- Spatial: Location, Home, Work etc.

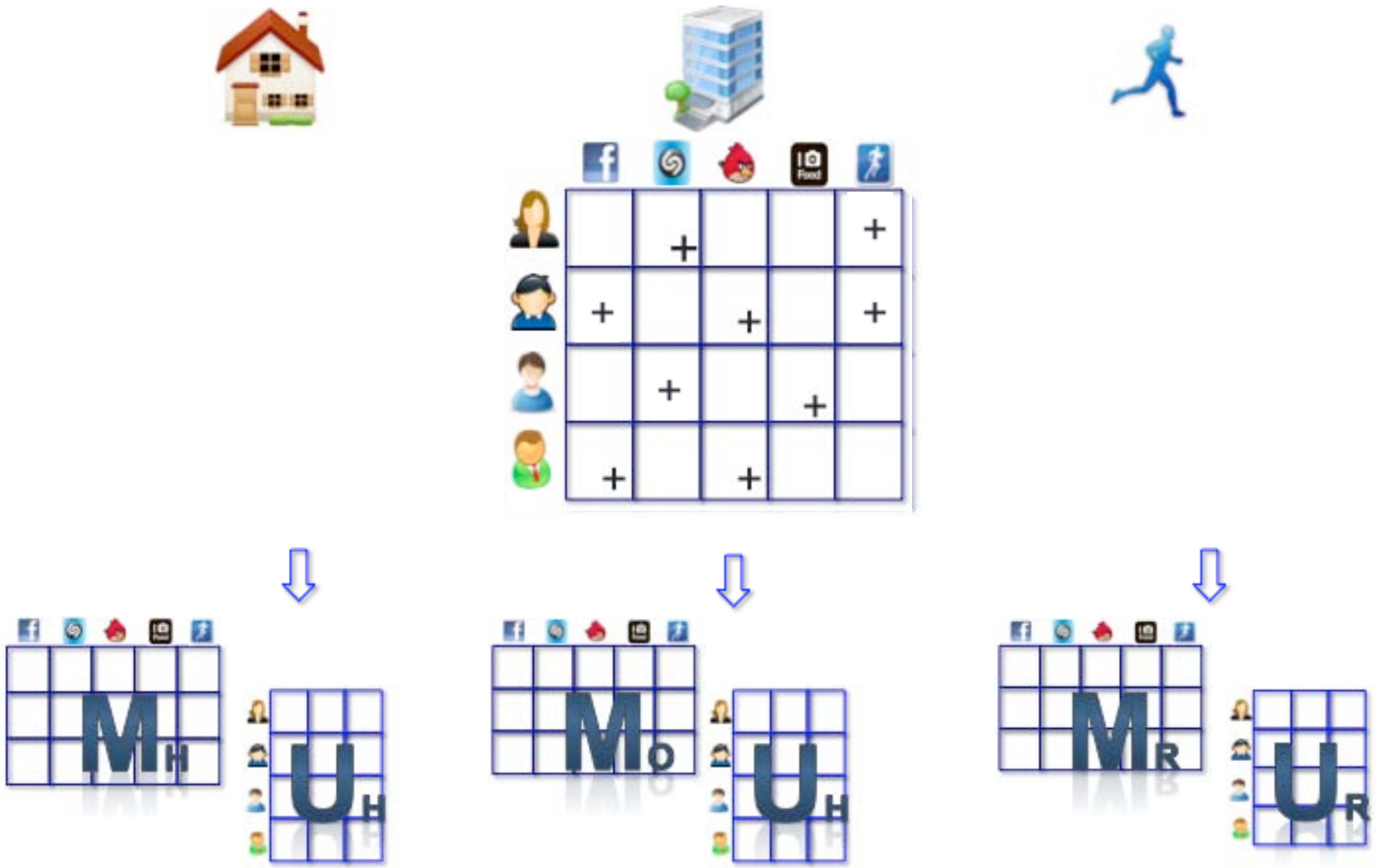
- Social: with Friends, Family

Recommendations should be tailored to the user & to the current *Context* of the user

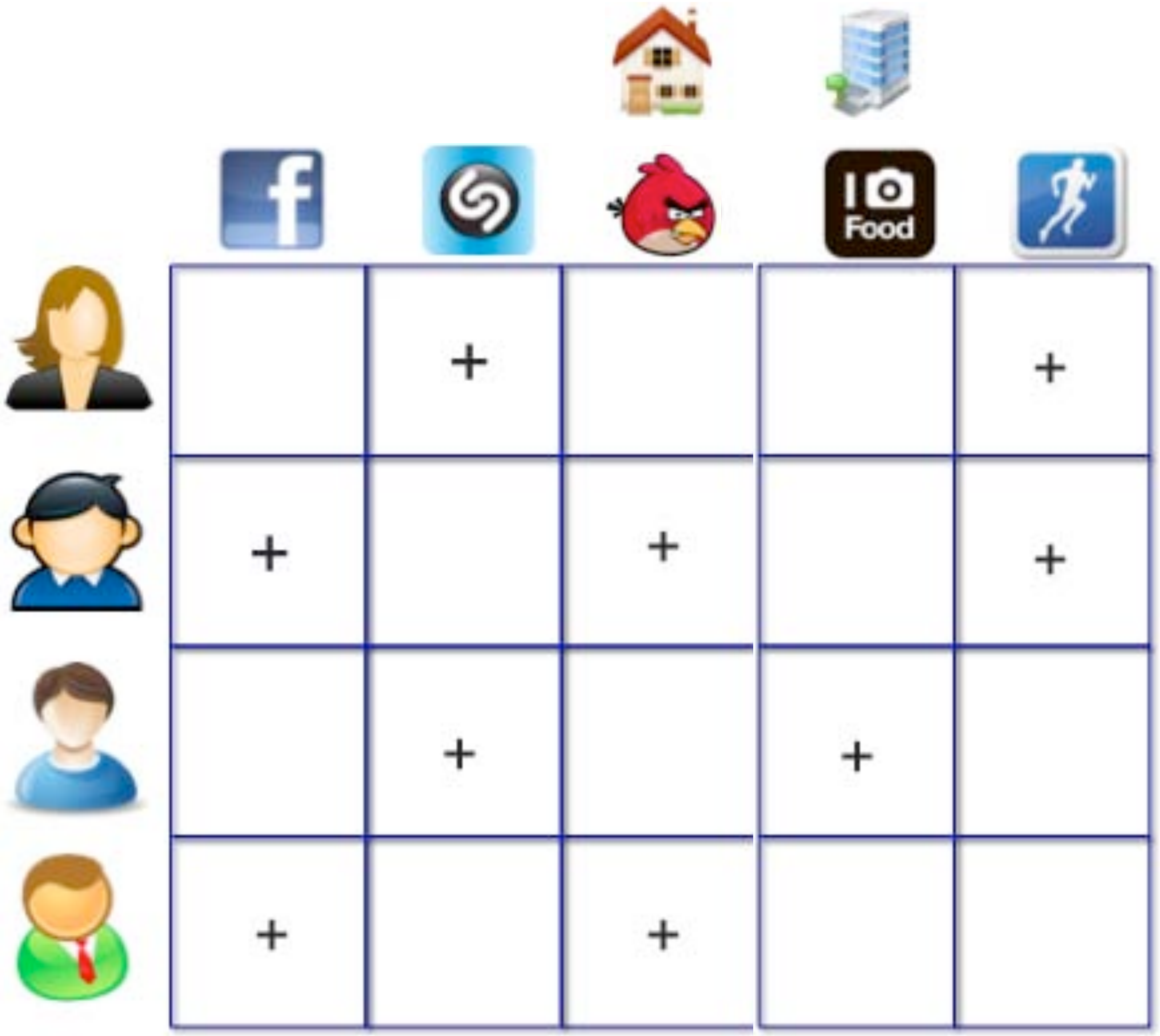
Level of Adaptation



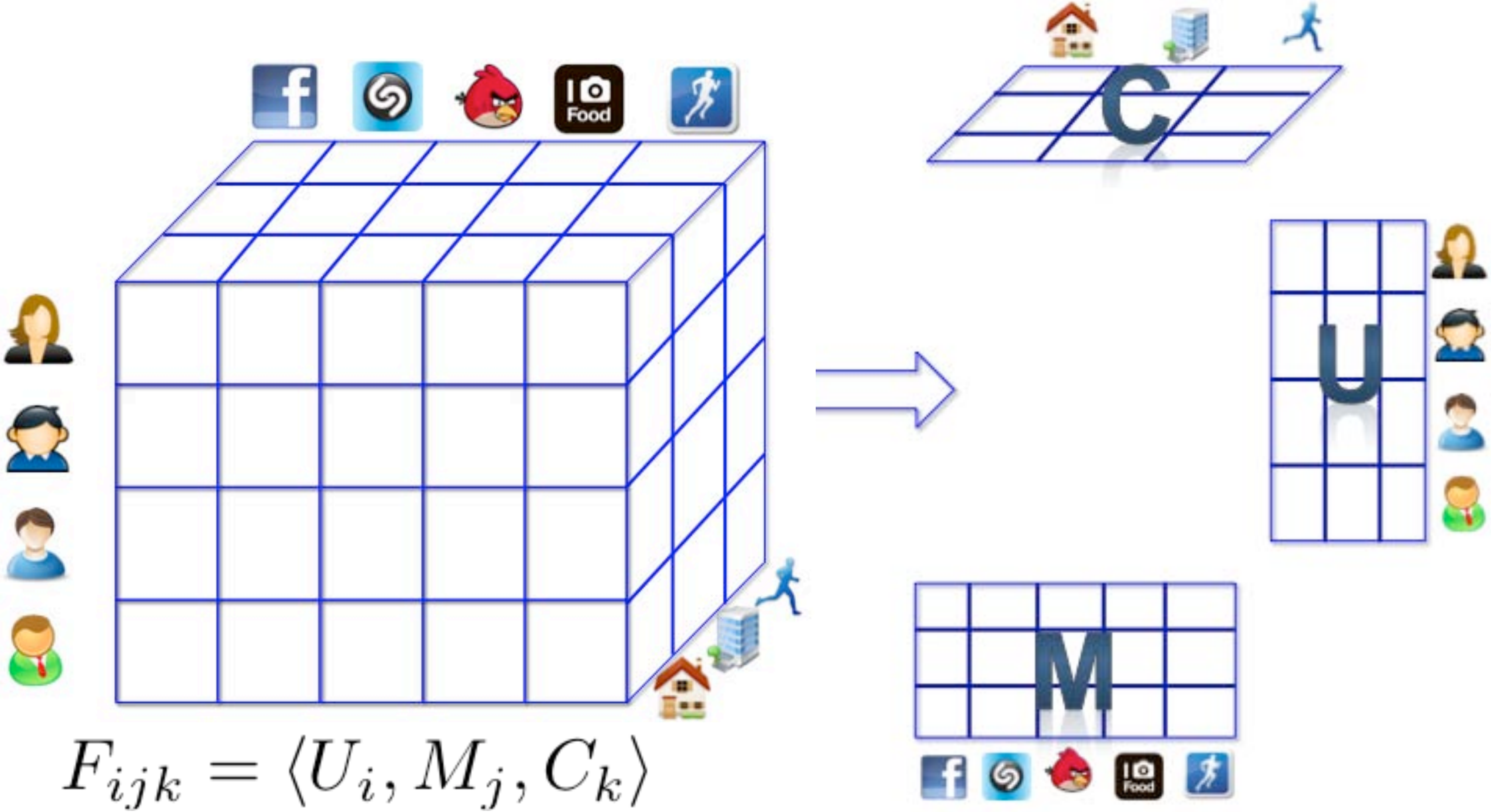
Context-Aware RS: Pre-filtering



Context-Aware RS: Post-filtering



Context-Aware RS: Tensor Factorization



$$F_{ijk} = \langle U_i, M_j, C_k \rangle$$

Context-Aware RS:

Pre-filtering
+ Simple
+ Works with large amounts of data
- Increases sparseness
- Does not scale well with many Context variables

Post-filtering
+ Single model
+ Takes into account context interactions
- Computationally expensive
- Increases data sparseness
- Does not model the Context directly

Tensor Factorization
+ Performance
+ Linear scalability
+ Models context directly

Index

Ranking

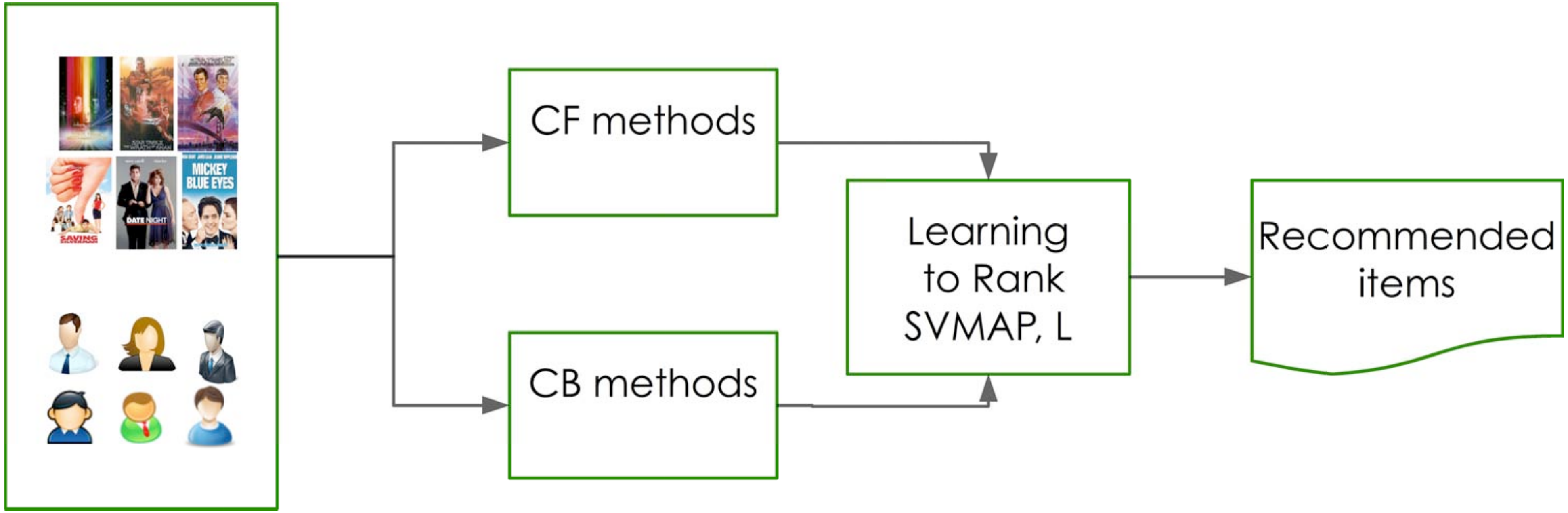
- Most recommendations are presented in a sorted list
- Recommendation is a ranking problem
- Popularity is the obvious baseline
- Users pay attention to few items at the top of the list

Ranking: Approaches

- (I) Re-ranking: based on features e.g. predicted rating, popularity, etc

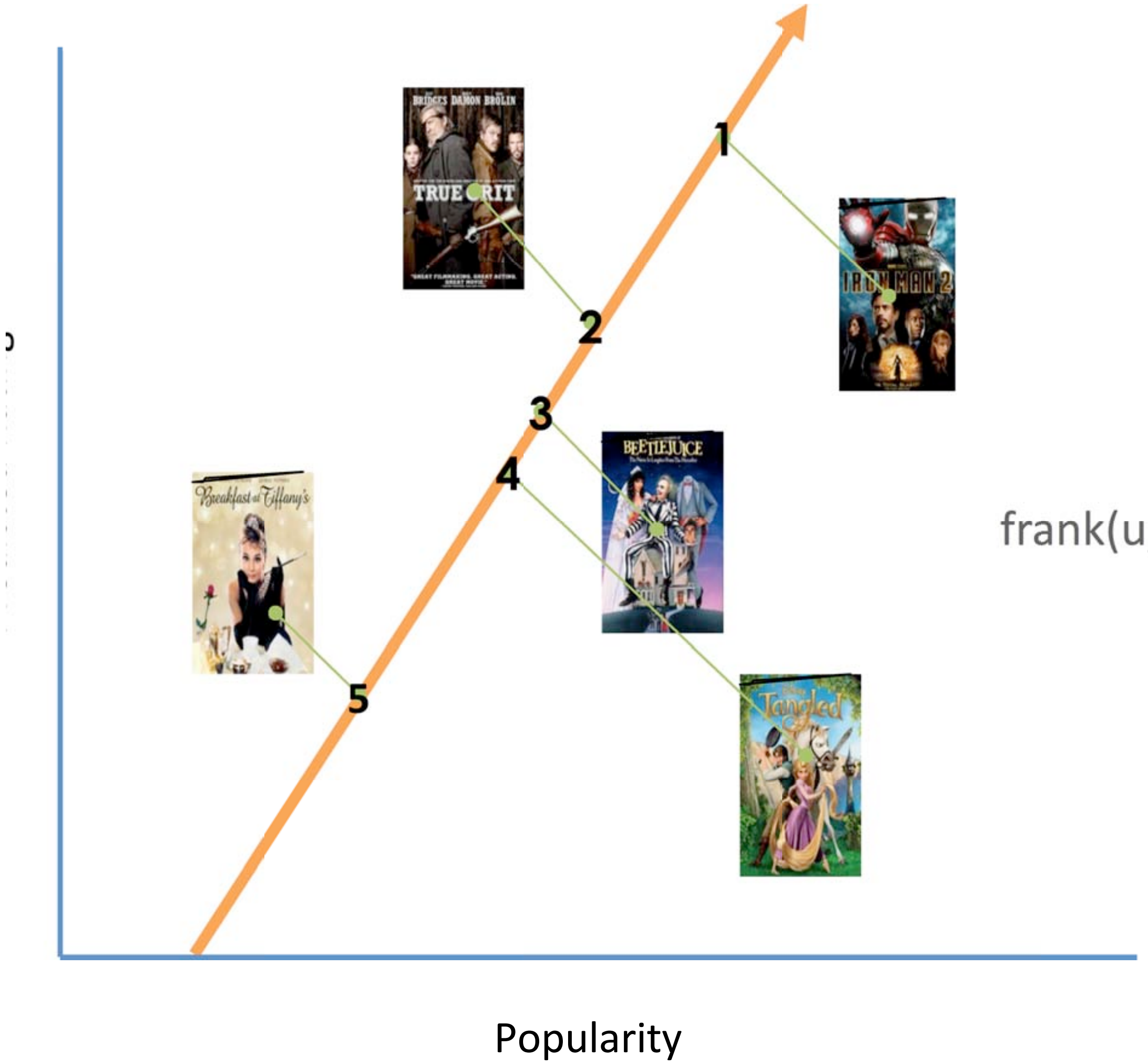
- (II) Learning to Rank: Build Ranking CF models

Re-ranking



CF methods
CB methods
Learning to Rank SVMAP, L
Recommended items

Re-ranking



Linear Model:
$$frank(u,v) = w1\ p(v) + w2\ r(u,v) + b$$

Re-ranking

Predicted Rating
Popularity
1
2
3
4
5
IRON MAN 2
BRIDGES DAMON BROLIN
TRUE GRIT
BEETLEJUICE
Tangled
Breakfast at Tiffany's
Final Ranking

Learning to rank

- Machine learning task: Rank the most relevant items as high as possible in the recommendation list
- Does not try to predict a rating, but the order of preference
- Training data have partial order or binary judgments (relevant/not relevant)
- Can be treated as a standard supervised classification problem

Learning to rank - Metrics

Metrics evaluate the quality of a recommendation list

- Normalized Discounted Cumulative Gain NDCG

- Computed for the first k items
- The NDCG@k of a list of items ratings Y, permuted by π is:

$$DCG(Y,k,\pi) = \sum_{i=0}^{k-1} \frac{2^{Y_\pi[i]} - 1}{log_2(i+2)} \quad , \quad NDCG(Y,k,\pi) = \frac{DCG(Y,k,\pi)}{DCG(Y,k,\pi_s)}$$

- where π_s is the permutation which sorts Y decreasingly

Learning to rank - Metrics

$$DCG = \sum_{i=1} \frac{2^{\#Stars[i]} - 1}{log_2([i] + 2)}$$

Perfect permutation of ratings	★★★★★	★★★★	★★★	★★	★	$\sum$
Contribution to DCG	44.72	13.65	5.05	1.86	0.56	**65.84**
	31/log(3)	15/log(4)	7/log(5)	3/log(6)	1/log(7)	

First wrong	★★★	★★★★	★★★★★	★★	★	$\sum$
Contribution to DCG	10.10	13.65	22.36	1.86	0.56	**48.53**
	7/log(3)		31/log(5)			

Last wrong	★★★★★	★★★★	★	★★	★★★	$\sum$
Contribution to DCG	44.72	13.65	0.72	1.86	3.91	**64.85**
			1/log(5)		7/log(7)	

Example of how DCG values different permutations of the same sequence of ratings.

Learning to rank - Metrics

- Mean Reciprocal Rank (MRR)

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}$$

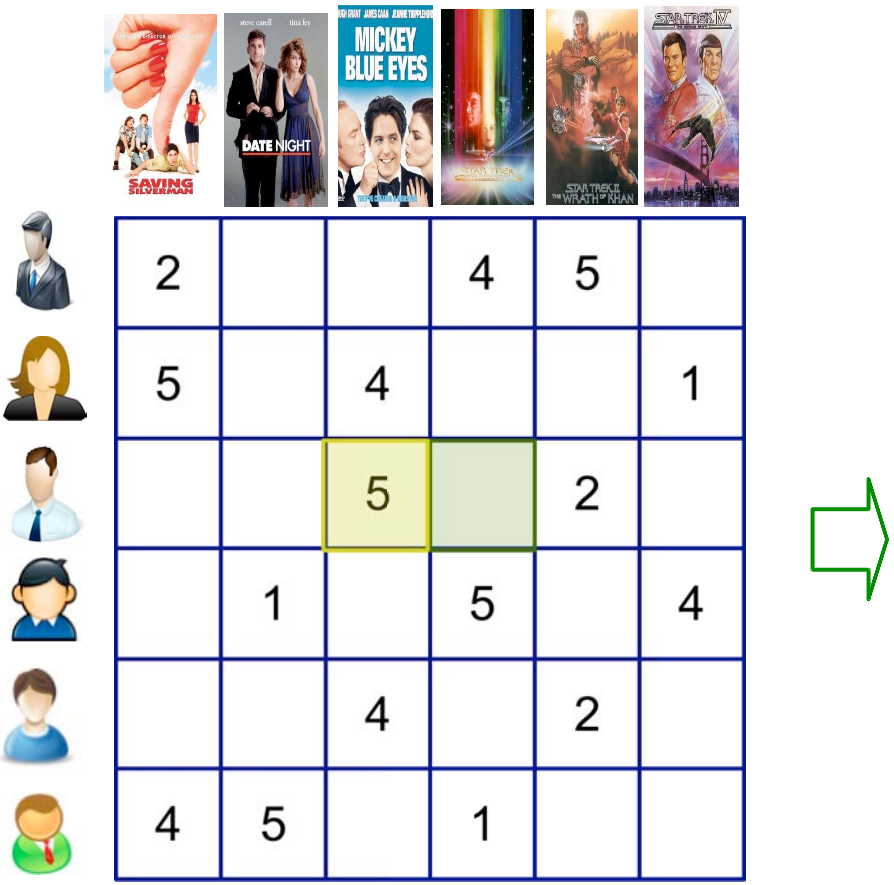
Learning to rank - Metrics

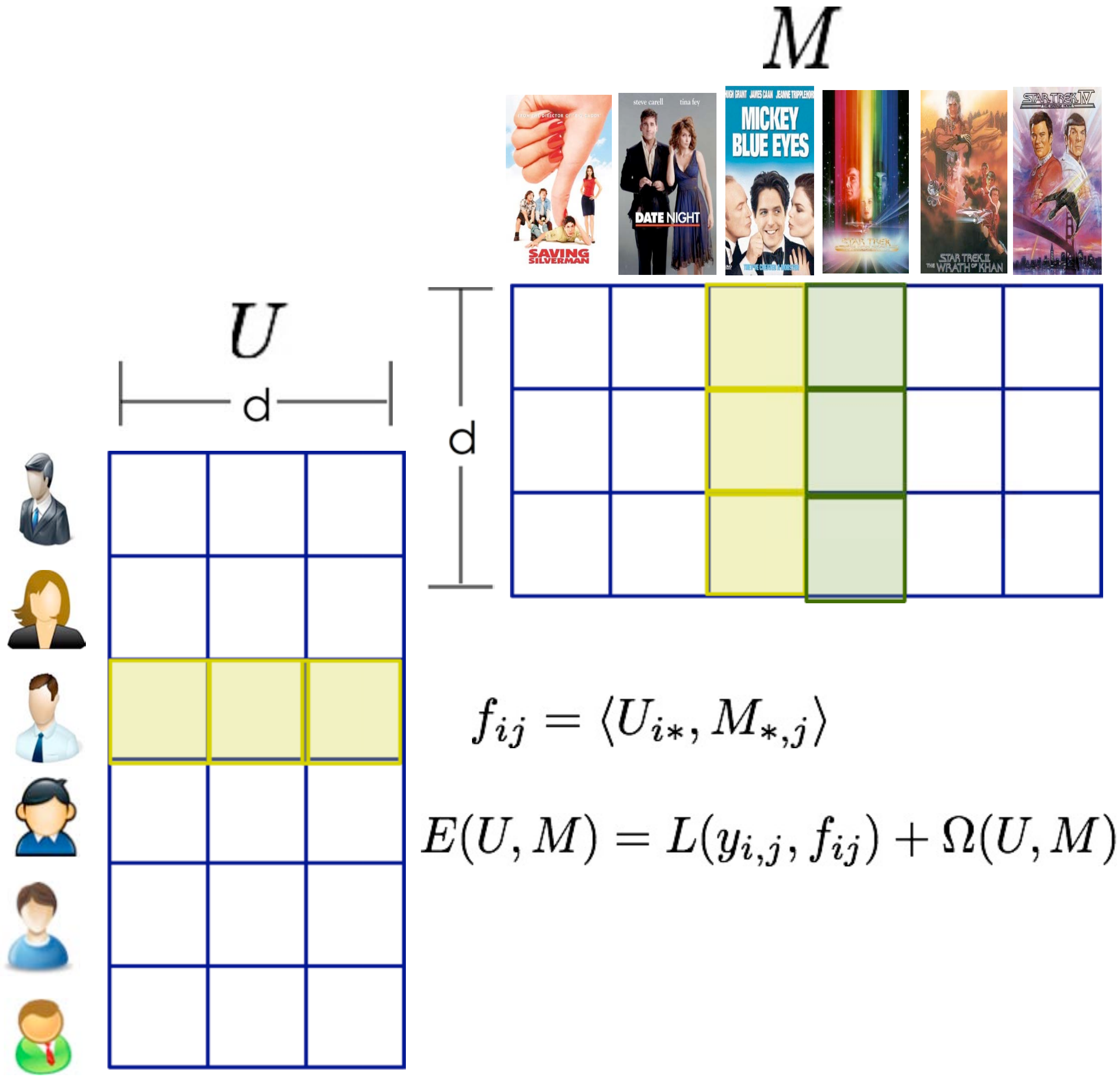
- Mean Average Precision (MAP)

$$\mathrm{AP} = \frac{\sum_{k=1}^{|S|} P(k)}{|S|} \qquad MAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

- S the set of relevant items, N #users

Matrix Factorization for Ranking





$$f_{ij} = \langle U_{i*}, M_{*,j} \rangle$$

$$E(U, M) = L(y_{i,j}, f_{ij}) + \Omega(U, M)$$

Learning to rank - Approaches

1) Pointwise $f(user, item) \rightarrow \mathbb{R}$

- Ranking function minimizes loss function defined on individual relevance judgment e.g.

$$L(y_{i,j}, f_{i,j}) = \frac{1}{2}(y_{i,j} - f_{i,j})^2$$

- Ranking score based on regression or classification
- Ordinal regression, Logistic regression, SVM

Learning to rank - Approaches

2) Pairwise $f(user, item_1, item_2) \rightarrow \mathbb{R}$

- Loss function is defined on pair-wise preferences $\sum_{y_i > y_j} f_i \leq f_j$
- Goal: minimize number of inversions in ranking
- BPR, RankBoost, RankNet, FRank…

Non-smoothness of Metrics

F_i =

0.81

0.75

0.64

0.58

0.55

RR = 0.5

Non-smoothness of Metrics

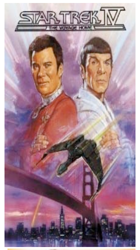
F_i =

0.82

0.80

RR = 0.5

0.63



0.52

0.50

Non-smoothness of Metrics

F_i =

0.83

0.82

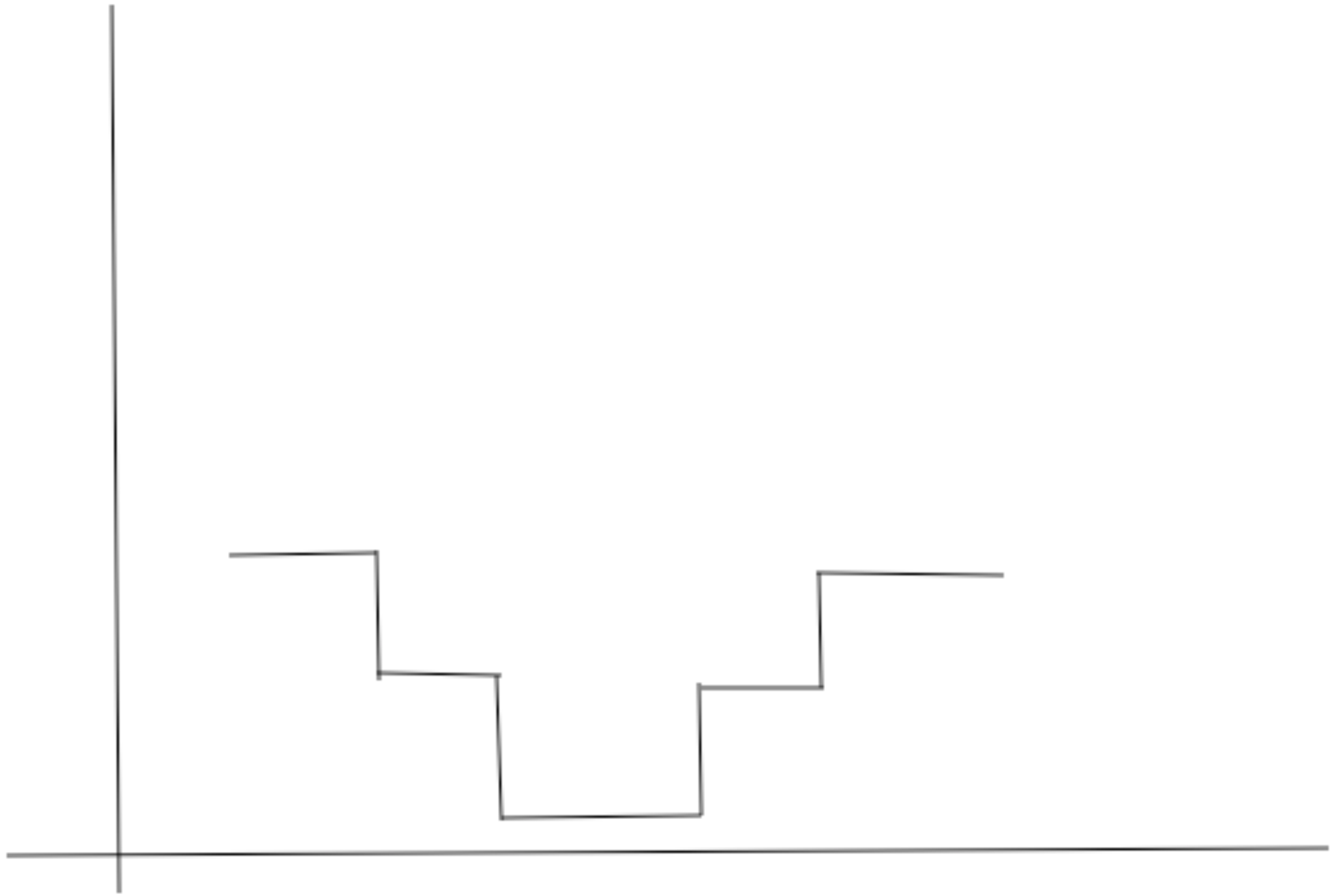
0.62

0.52

0.49

RR = 1

Non-smoothness of Metrics



Learning to rank - Approaches

- 3) Listwise $f(user, item_1, \ldots, item_n) \to \mathbb{R}$
  - Direct optimization of ranking metrics,
  - List-wise loss minimization for CF a.k.a Collaborative Ranking
    - CoFiRank: optimizes an upper bound of NDCG (Smooth version)
    - CLiMF : optimizes a smooth version of MRR
    - TFMAP: optimizes a smooth version of MAP
    - AdaRank: uses boosting to optimize NDCG

Diversity in Recommendation (I)

- Recommendations from a music on-line retailer:



Born This Way	*Pink Friday*	*Dangerously in Love*	*Born This Way – The Remix*	*Femme Fatale*	*Can't be Tamed*	*Teenage Dream*
Lady Gaga	Nicki Minaj	Beyoncé	Lady Gaga	Britney Spears	Miley Cyrus	Katy Perry

- *No diversity:* pop albums from female singers.
- Some are *redundant*.

Diversity in Recommendation (II)

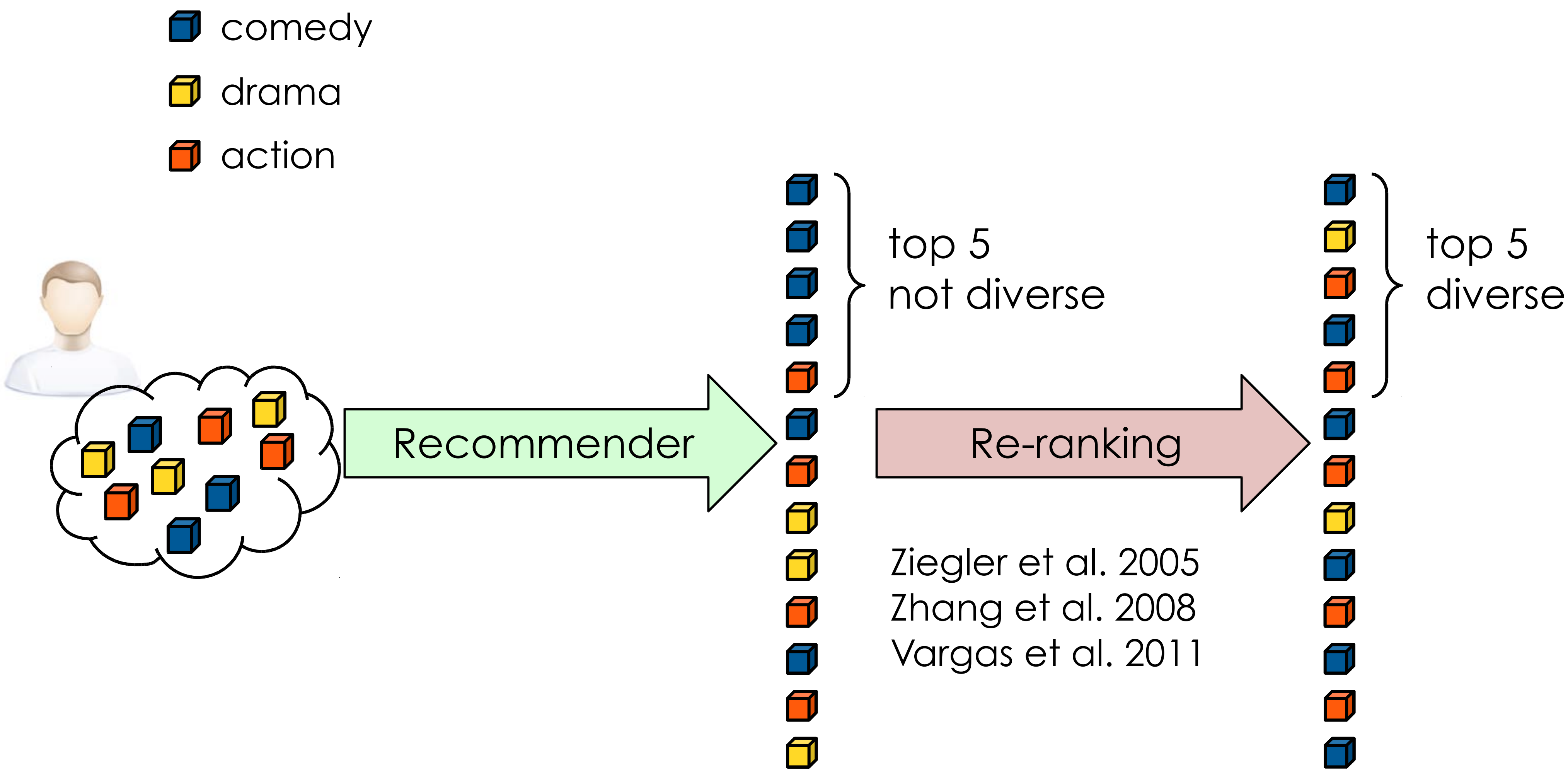
- Some good music recommendations:



Wrecking Ball	*Not your Kind of People*	*Like a Prayer*	*Choice of Weapon*	*Sweet Heart Sweet Light*	*The Light the Dead See*	Little Broken Hearts
B. Springsteen	Garbage	Madonna	The Cult	Spiritualized	Soulsavers	Norah Jones

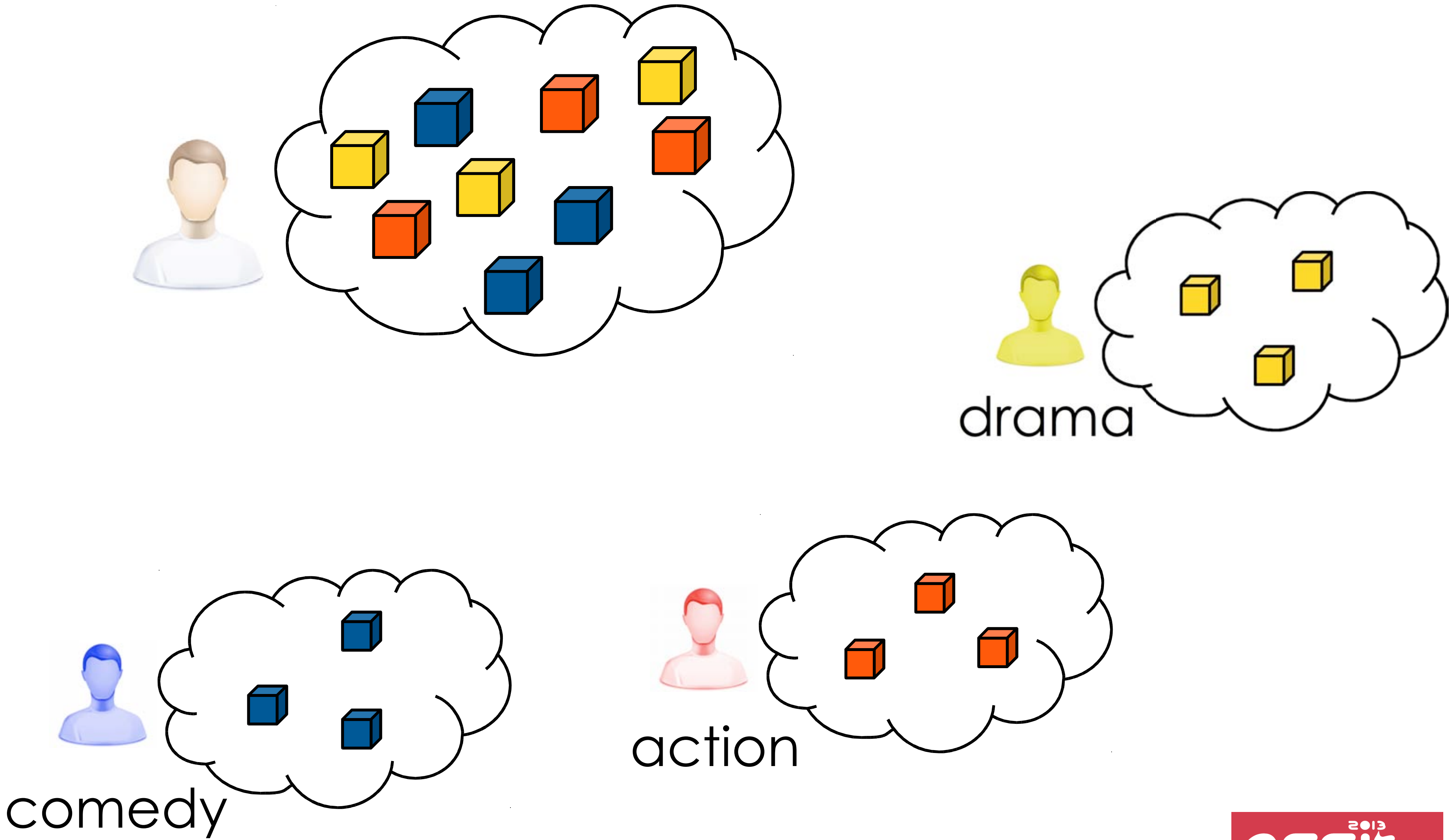
- Different artists and genres.
- Not similar between them.
- These are much better recommendations!

Diversity: Re-Ranking



Ziegler et al. 2005
Zhang et al. 2008
Vargas et al. 2011

Sub-Profiles



Social and Trust-based recommenders

- A social RS recommends items that are "popular" with the friends of the user.
- *Friendship though does not* imply *trust*
- "Trust" in social-based RS can be per-user or topic-specific

Building RS Using Trust

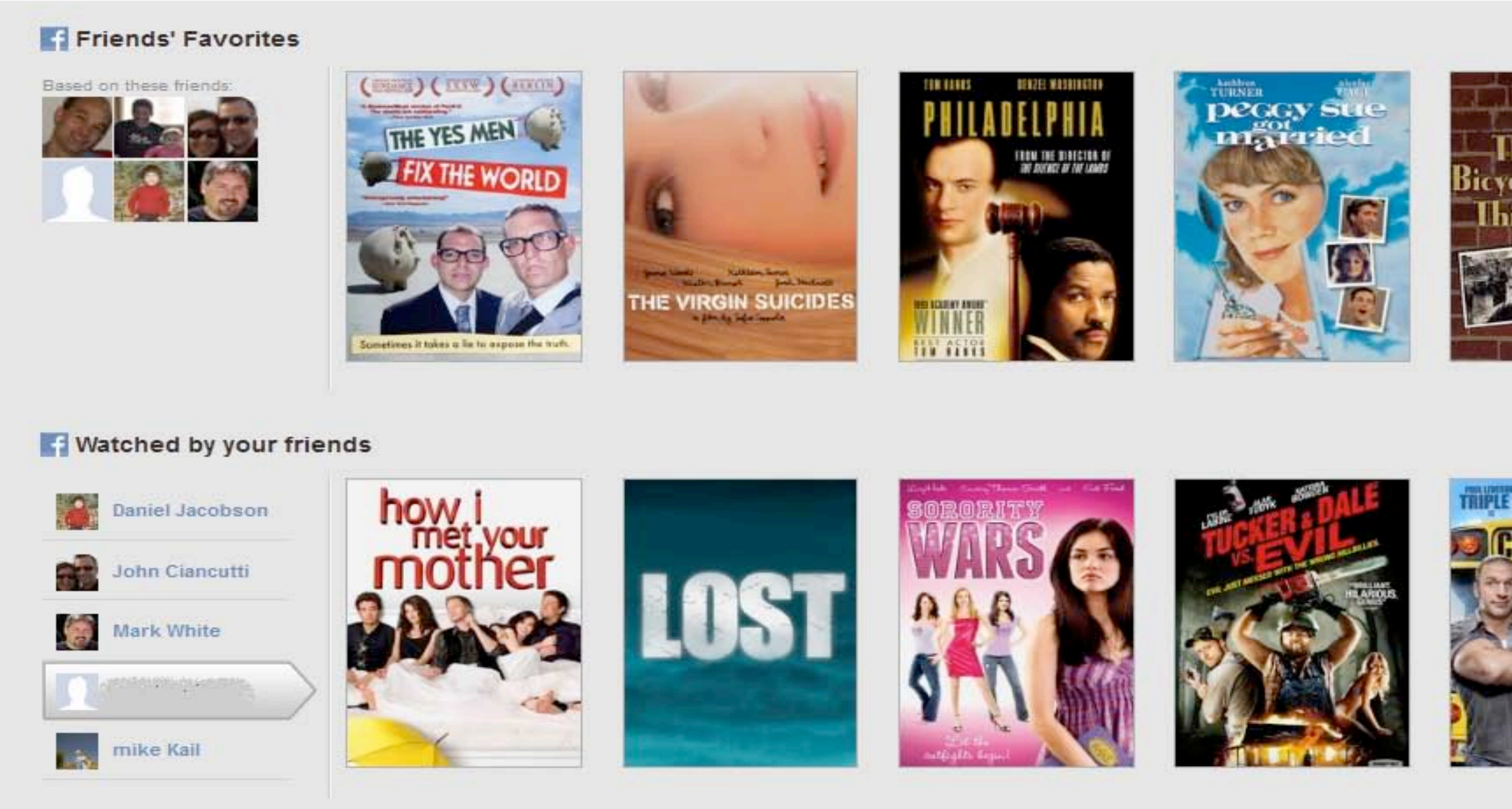
- Trust for CF
  - Use trust to give more weight to some users
  - Use trust in place of (or combined with) similarity

- Trust for sorting & filtering
  - Prioritize information from trusted sources

Other ways to use Social

- Social connections can be used in combination with other approaches
- In particular, "friendships" can be fed into CF methods in different ways
  - e.g. replace or modify user-user similarity by using social network information

Social Recommendations



Index

Hybridization Methods

Hybridization Method	Description
Weighted	Outputs (scores or votes) from several techniques are combined with different degrees of importance to offer final recommendations
Switching	Depending on situation, the system changes from one technique to another
Mixed	Recommendations from several techniques are presented at the same time
Feature combination	Features from different recommendation sources are combined as input to a single technique
Cascade	The output from one technique is used as input of another that refines the result
Feature augmentation	The output from one technique is used as input features to another
Meta-level	The model learned by one recommender is used as input to another

Weighted

- Rating for an item is computed as the weighted sum of ratings produced by a pool of different RS.
- The weights are determined by training and get adjusted as new ratings arrive.
- Assumption: relative performance of the different techniques is uniform. Not true in general: e.g. CF performs worse for items with few ratings.

  e.g.

  - a CB and a CF recommender equally weighted at first. Weights are adjusted as predictions are confirmed or not.
  - RS with consensus scheme: each recommendation of a specific item counts as a vote for the item.

Feature Combination

- CF ratings of users are passed as additional feature to a CB. CB makes recommendations over this augmented data set.

Switching

- The system uses a criterion to switch between techniques
- The main problem is to identify a good switching criterion.

e.g.

- The DailyLearner system uses a CB-CF. When CB cannot predict with sufficient confidence, it switches to CF.

Mixed

- Recommendations from more than one technique are presented together

  e.g.

  - The PTV system recommends a TV viewing schedule for the user by combining recommendations from a CB and a CF system.
  - CB uses the textual descriptions of TV shows; vs CF uses other users' preferences.
  - When collision occurs, the CB has priority.

Cascade

- At each iteration, a first recommendation technique produces a coarse ranking & a second technique refines the recommendation
- Cascading avoids employing the second, lower-priority, technique on items already well-differentiated by the first
- Requires a meaningful ordering of the techniques.
- E.g.: EntreeC is a restaurant RS uses its knowledge of restaurants to make recommendations based on the user's stated interests. The recommendations are placed in buckets of equal preference, and the collaborative technique breaks ties

Feature Augmentation

- Very similar to the feature combination method:

  - Here the output of one RS is incorporated into the processing of a second RS

  e.g.:

  - Amazon.com generates text data ("related authors" and "related titles") using its internal collaborative systems
  - Libra system makes content-based recommendations of books based on these text data found in Amazon.com, using a naive Bayes text classifier

Index

1. Introduction: What is a Recommender System?
2. Approaches
    1. Collaborative Filtering
    2. Content-based Recommendations
    3. Context-aware Recommendations
    4. Other Approaches
    5. Hybrid Recommender Systems
3. Research Directions
4. Conclusions
5. References

Beyond Explicit Ratings

- Implicit feedback is more readily available, and less noisy
- Already many approaches (e.g. SVD++) can make use of implicit feedback
- Ongoing research in combining explicit and implicit feedback

- D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In Proc.of the 18th WWW, 2009.
- Koren Y and J. Sill. OrdRec: an ordinal model for predicting personalized item rating distributions. In Rec-Sys '11, pages 117–124, 2011.
- Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD, 2008.
- Yifan Hu, Y. Koren, and C. Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In Proc. Of the 2008 Eighth ICDM, pages 263–272, 2008.

Personalized Learning to Rank

- Better approaches to learning to rank that directly optimize ranking metrics and allow for personalization (e.g. CliMF & TFMAP)
- Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In Proc. of the sixth Recsys, 2012.
- Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson,A. Hanjalic, and N. Oliver. TFMAP: optimizing MAP for top-n context-aware recommendation. In Proc. Of the 35th SIGIR, 2012.

Context-aware Recommendations

- Beyond the traditional 2D user-item space
- Recommendations should also respond to user context (e.g. location, time of the day...)
- Many different approaches such as Tensor Factorization or Factorization Machines
- A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In Proc. of the fourth ACM Recsys, 2010.
- S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In Proc. of the 34th ACM SIGIR, 2011.

User choice and presentation effects

- We log the recommended items to the users and their choice
- We can use this information as negative feedback (not chosen) and positive feedback (chosen).
- S.H. Yang, B. Long, A.J. Smola, H. Zha, and Z. Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In Proc. of the 34th ACM SIGIR, 2011.

Social Recommendations

- Beyond trust-based
- Cold-starting with Social Information
- Combining Social with CF
- Finding "experts"
- J. Delporte, A. Karatzoglou, T. Matuszczyk, S. Canu. Socially Enabled Preference Learning from Implicit Feedback Data. In Proc. of ECML/PKDD 2013
- N. N. Liu, X. Meng, C. Liu, and Q. Yang. Wisdom of the better few: cold start recommendation via representative based rating elicitation. In Proc. of RecSys'11, 2011.
- M. Jamali and M. Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In Proc. of KDD '09, 2009.
- J. Noel, S. Sanner, K. Tran, P. Christen, L. Xie, E. V. Bonilla, E. Abbasnejad, and N. Della Penna. New objective functions for social collaborative filtering. In Proc. of WWW '12, pages 859–868, 2012.
- X. Yang, H. Steck, Y. Guo, and Y. Liu. On top-k recommendation using social networks. In Proc. of RecSys'12, 2012.

Index

1. Introduction: What is a Recommender System?
2. Approaches
    1. Collaborative Filtering
    2. Content-based Recommendations
    3. Context-aware Recommendations
    4. Other Approaches
    5. Hybrid Recommender Systems
3. Research Directions
4. Conclusions
5. References

Conclusions

- RS are an important application of Machine Learning
- RS have the potential to become as important as Search is now
- However, RS are more than Machine Learning
  - HCI
  - Economical models
  - …

Conclusions

- RS are fairly new but already grounded on well-proven technology
  - Collaborative Filtering
  - Machine Learning
  - Content Analysis
  - Social Network Analysis
  - …
- However, there are still many open questions and a lot of interesting research to do!

Index

1. Introduction: What is a Recommender System?
2. Approaches
    1. Collaborative Filtering
    2. Content-based Recommendations
    3. Context-aware Recommendations
    4. Other Approaches
    5. Hybrid Recommender Systems
3. Research Directions
4. Conclusions
5. References

References

- *"Recommender Systems Handbook." Ricci, Francesco, Lior Rokach, Bracha Shapira, and Paul B. Kantor. (2010).*
- *"Recommender systems: an introduction". Jannach, Dietmar, et al. Cambridge University Press, 2010.*
- *"Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions"*. G. Adomavicious and A. Tuzhilin. 2005. IEEE Transactions on Knowledge and Data Engineering, 17 (6)
- *"Item-based Collaborative Filtering Recommendation Algorithms"*, B. Sarwar et al. 2001. Proceedings of World Wide Web Conference.
- *"Lessons from the Netflix Prize Challenge."*. R. M. Bell and Y. Koren. SIGKDD Explor. Newsl., 9(2):75–79, December 2007.
- *"Beyond algorithms: An HCI perspective on recommender systems"*. K. Swearingen and R. Sinha. In ACM SIGIR 2001 Workshop on Recommender Systems
- *"Recommender Systems in E-Commerce"*. J. Ben Schafer et al. ACM Conference on Electronic Commerce. 1999-
- *"Introduction to Data Mining"*, P. Tan et al. Addison Wesley. 2005

References

- *"Evaluating collaborative filtering recommender systems"*. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. ACM Trans. Inf. Syst., 22(1):5–53, 2004.
- *"Trust in recommender systems"*. J. O'Donovan and B. Smyth. In Proc. of IUI '05, 2005.
- *"Content-based recommendation systems"*. M. Pazzani and D. Billsus. In The Adaptive Web, volume 4321. 2007.
- *"Fast context-aware recommendations with factorization machines"*. S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. In Proc. of the 34th ACM SIGIR, 2011.
- *"Restricted Boltzmann machines for collaborative filtering"*. R. Salakhutdinov, A. Mnih, and G. E. Hinton.In Proc of ICML '07, 2007
- "Learning to rank: From pairwise approach to listwise approach". *Z. Cao and T. Liu. In In Proceedings of the 24th ICML, 2007.*
- *"Introduction to Data Mining"*, P. Tan et al. Addison Wesley. 2005

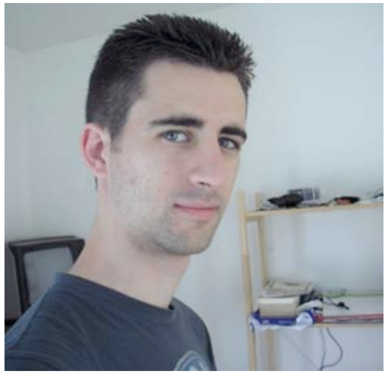
Online resources

- Recsys Wiki: http://recsyswiki.com/
- Recsys conference Webpage: http://recsys.acm.org/
- Recommender Systems Books Webpage: http://www.recommenderbook.net/
- Mahout Project: http://mahout.apache.org/
- MyMediaLite Project: http://www.mymedialite.net/

Thanks

Xavier Amatriain
@Netflix

Linas Baltrunas
@Telefonica Research



Saúl Vargas
@UAM

Yue Shi
@TU Delft

Thank you!

Questions?

Alexandros Karatzoglou

alexk@tid.es

@alexk_z