Recommender Systems

MLSS '14

Collaborative Filtering and other approaches

Xavier Amatriain
Research/Engineering Director @ Netflix

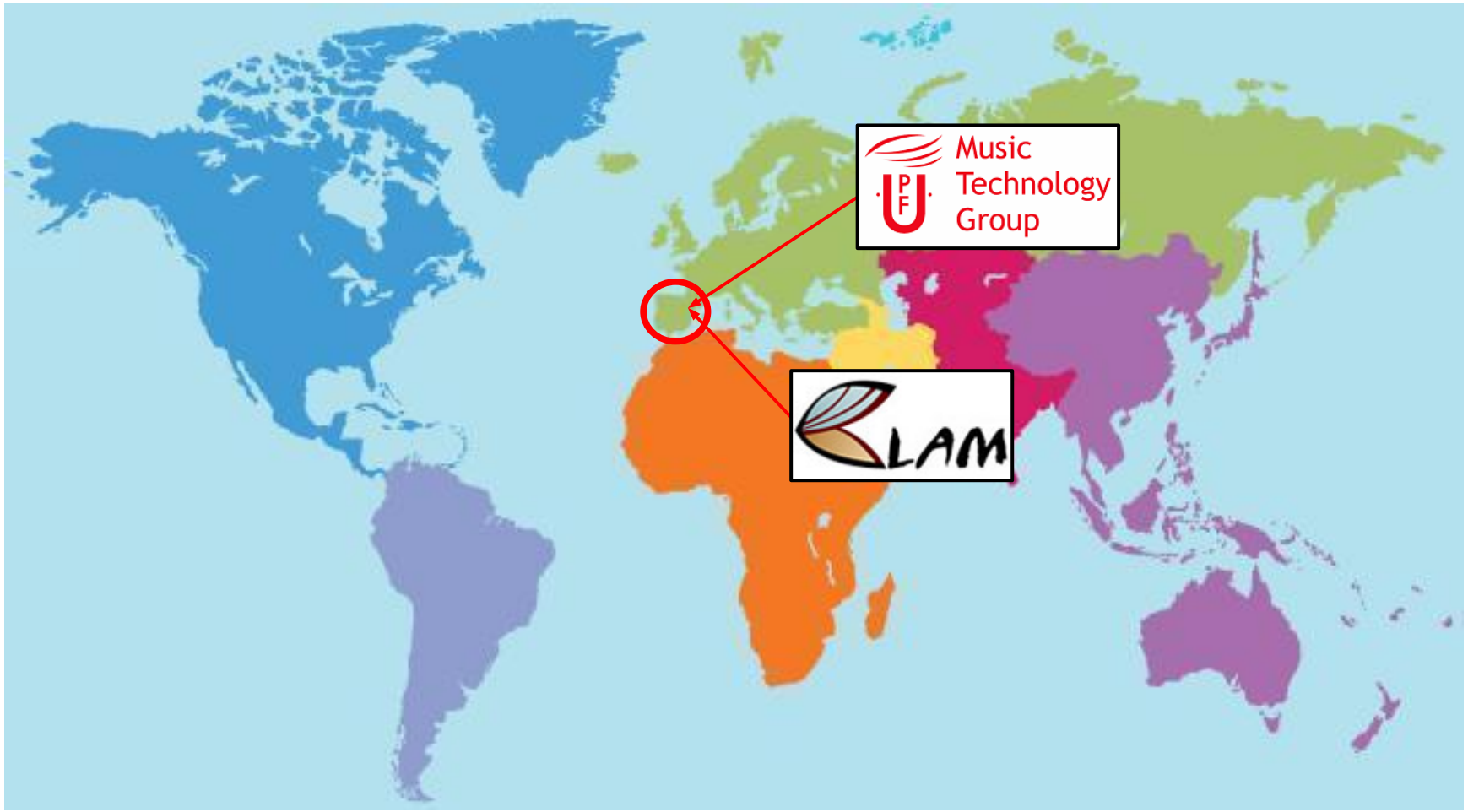
NETFLIX

But first...

NETFLIX

About me

Up until 2005



NETFLIX

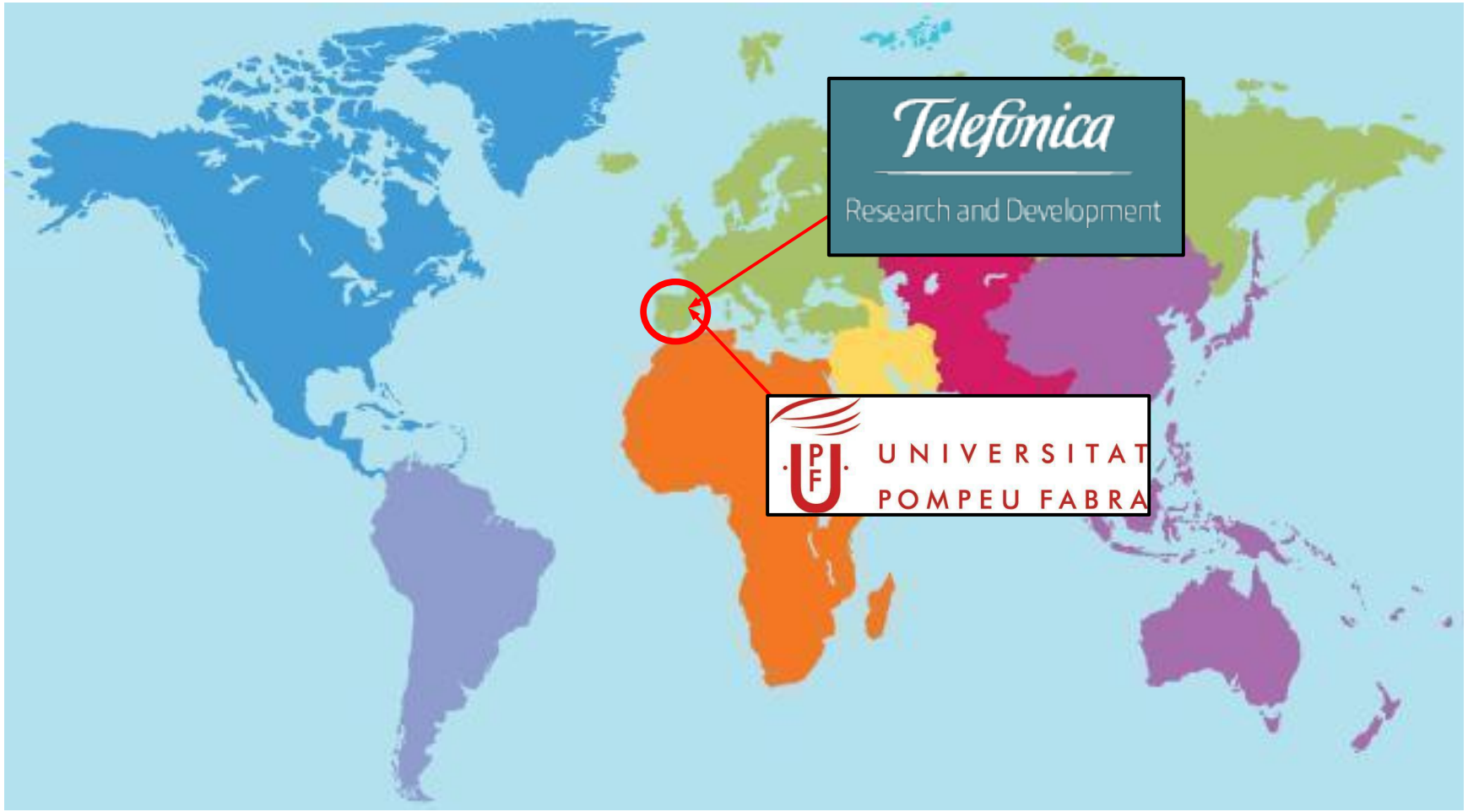
About me

2005 - 2007

NETFLIX

About me

2007 - 2011



NETFLIX

About me

Since 2011

Recent/Upcoming Publications

- The Recommender Problem Revisited. KDD and Recsys 2014 Tutorial
- KDD: Big & Personal: data and models behind Netflix recommendations. 2013
- SIGKDD Explorations: Mining large streams of user data for personalized recommendations. 2012
- Recsys: Building industrial-scale real-world recommender systems. 2012
- Recys - Walk the Talk: Analyzing the relation between implicit and explicit feedback for preference elicitation. 2011
- SIGIR – Temporal behavior of CF. 2010
- Web Intelligence – Expert-based CF for music. 2010
- Recsys – Tensor Factorization. 2010
- Mobile HCI – Tourist Recommendation. 2010
- Recsys Handbook (book) – Data mining for recsys. 2010 & Recommender Systems in Industry. 2014
- SIGIR – Wisdom of the Few. 2009
- Recsys – Denoising by re-rating. 2009
- CARS – Implicit context-aware recommendations. 2009
- UMAP – I like it I like it not. 2009

Collaborators/Contributors

Justin Basilico
(Netflix)

Alexandros Karatzoglou
(Telefonica Research)

Bamhshad Mobasher
(De Paul U)

Francesco Ricci
(University of Bolzano)

Erik Bernhardsson
(Spotify)

NETFLIX

Index

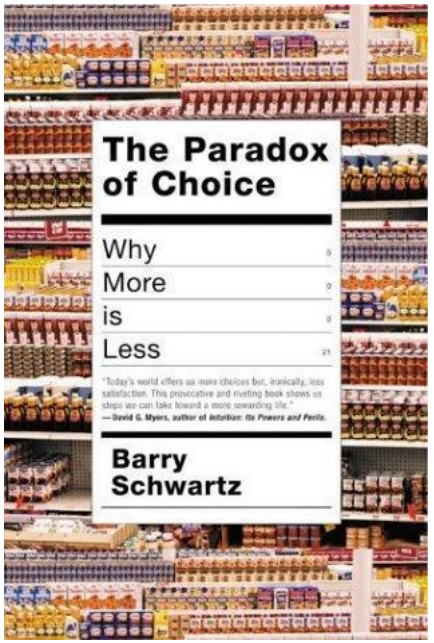
1. Introduction: What is a Recommender System?

The Age of Search has come to an end

- **... long live the Age of Recommendation!**
- **Chris Anderson in "The Long Tail"**
  - *"We are leaving the age of information and entering the age of recommendation"*
- **CNN Money, "The race to create a 'smart' Google":**
  - *"The Web, they say, 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."*

NETFLIX

Information overload



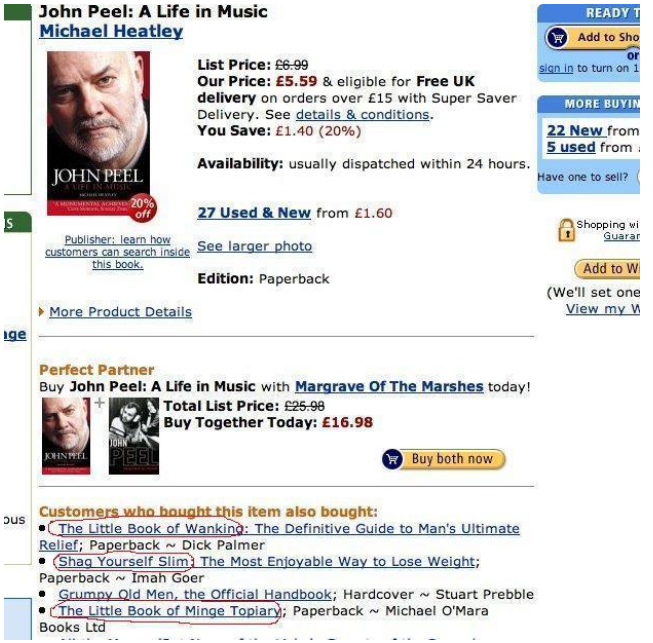
"People read around 10 MB worth of material a day, hear 400 MB a day, and see 1 MB of information every second" - The Economist, November 2006

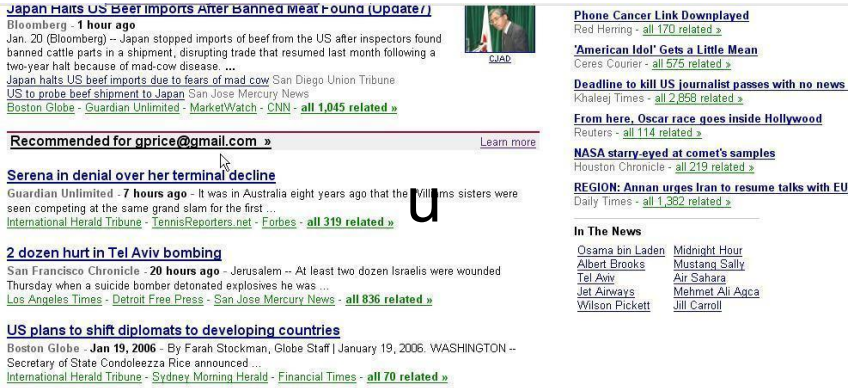
In 2015, consumption will raise to 74 GB a day - UCSD Study 2014

NETFLIX

The value of recommendations

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





NETFLIX

The “Recommender problem”

- Estimate a utility function that automatically predicts how a user will like an item.
- Based on:
  - Past behavior
  - Relations to other users
  - Item similarity
  - Context
  - …

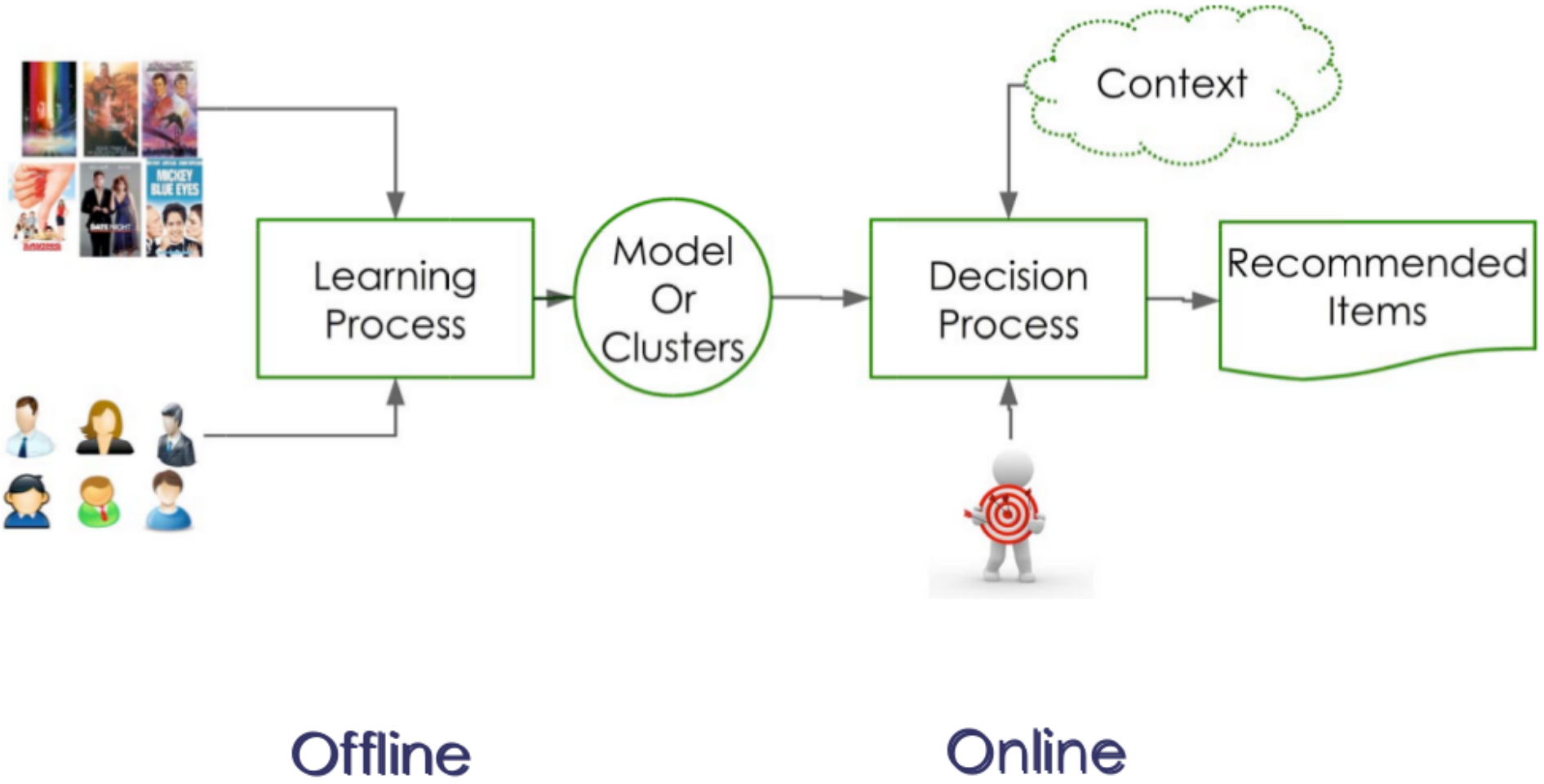
The “Recommender problem”

- Let ***C*** be set of all users and let ***S*** be set of all possible recommendable items
- Let ***u*** be a utility function measuring the usefulness of item ***s*** to user ***c***, i.e., ***u : C X S→R***, where ***R*** is a totally ordered set.
- For each user ***cϵC***, we want to choose items ***sϵS*** that maximize ***u***.
  - Utility is usually represented by rating but can be any function

$$\forall c \in C, s'_c = argmax(u(c,s)) s \in S$$

NETFLIX

Two-step process

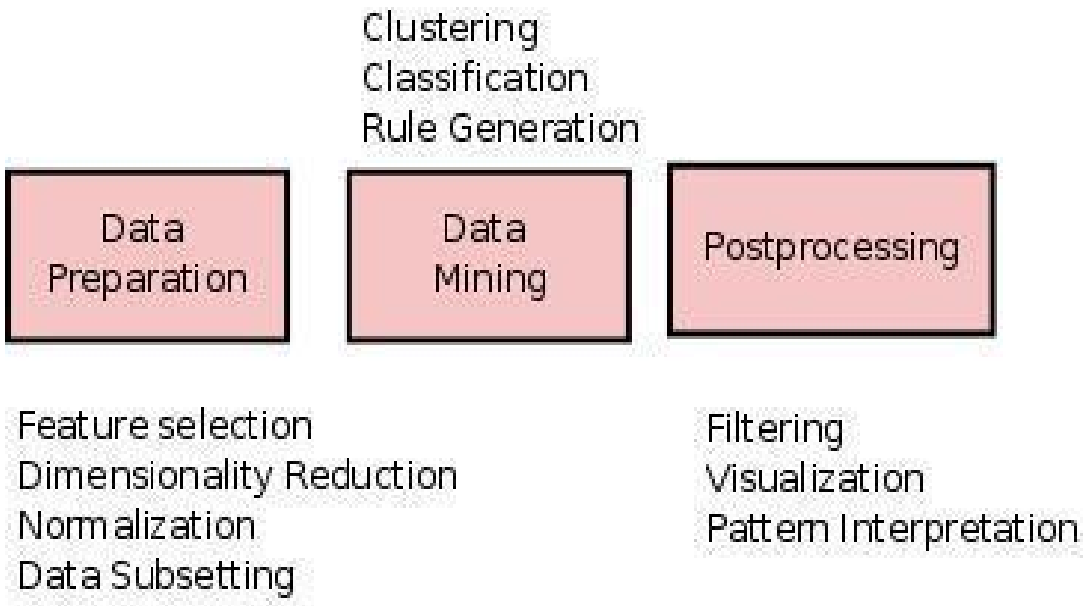


Approaches to Recommendation

- Collaborative Filtering: Recommend items based only on the users past behavior
  - **User-based:** Find similar users to me and recommend what they liked
  - **Item-based:** Find similar items to those that I have previously liked
- Content-based: Recommend based on item features
- Personalized Learning to Rank: Treat recommendation as a ranking problem
- Demographic: Recommend based on user features
- Social recommendations (trust-based)
- Hybrid: Combine any of the above

Recommendation as data mining

The core of the Recommendation Engine can be assimilated to a general data mining problem:



Machine Learning + all those other things

- User Interface
- System requirements (efficiency, scalability, privacy....)
- Serendipity
- ....

Serendipity

- Unsought finding
- Don't recommend items the user already knows or **would have found anyway**.
- Expand the user's taste into neighboring areas by improving the obvious
- Collaborative filtering can offer controllable serendipity (e.g. controlling how many neighbors to use in the recommendation)

What works

- Depends on the **domain** and particular **problem**
- However, in the general case it has been demonstrated that the best isolated approach is CF.
  - Other approaches can be hybridized to improve results in specific cases (cold-start problem...)
- What matters:
  - Data preprocessing: outlier removal, denoising, removal of global effects (e.g. individual user's average)
  - "Smart" dimensionality reduction using MF/SVD
  - Combining methods

Index

2. Traditional Approaches

2.1. Collaborative Filtering

The CF Ingredients

- List of **m Users** and a list of **n Items**
- Each user has a **list of items** with associated **opinion**
  - **Explicit opinion** - a rating score
  - Sometime the rating is **implicitly** – purchase records or listen to tracks
- **Active user** for whom the CF prediction task is performed
- **Metric** for measuring **similarity between users**
- Method for selecting a subset of **neighbors**
- Method for **predicting a rating** for items not currently rated by the active user.

Collaborative Filtering

The basic steps:

1. Identify set of ratings for the **target/active user**
2. Identify set of users most similar to the target/active user according to a similarity function (**neighborhood** formation)
3. Identify the products these similar users liked
4. **Generate a prediction** - rating that would be given by the target user to the product - for each one of these products
5. Based on this predicted rating recommend a set of top N products

Collaborative Filtering

- **Pros:**
  - Requires **minimal knowledge** engineering efforts
  - Users and products are symbols without any internal structure or characteristics
  - Produces good-enough results in most cases
- **Cons:**
  - Requires a large number of **reliable** "user feedback data points" to bootstrap
  - Requires products to be standardized (users should have bought **exactly** the same product)
  - Assumes that **prior behavior determines current behavior** without taking into account "contextual" knowledge (session-level)

Personalised vs Non-Personalised CF

- CF recommendations are **personalized** since the “prediction” is based on the ratings expressed by **similar users**
  - Those **neighbors** are **different** for each target user
- A **non-personalized** collaborative-based recommendation can be generated by averaging the recommendations of **ALL** the 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

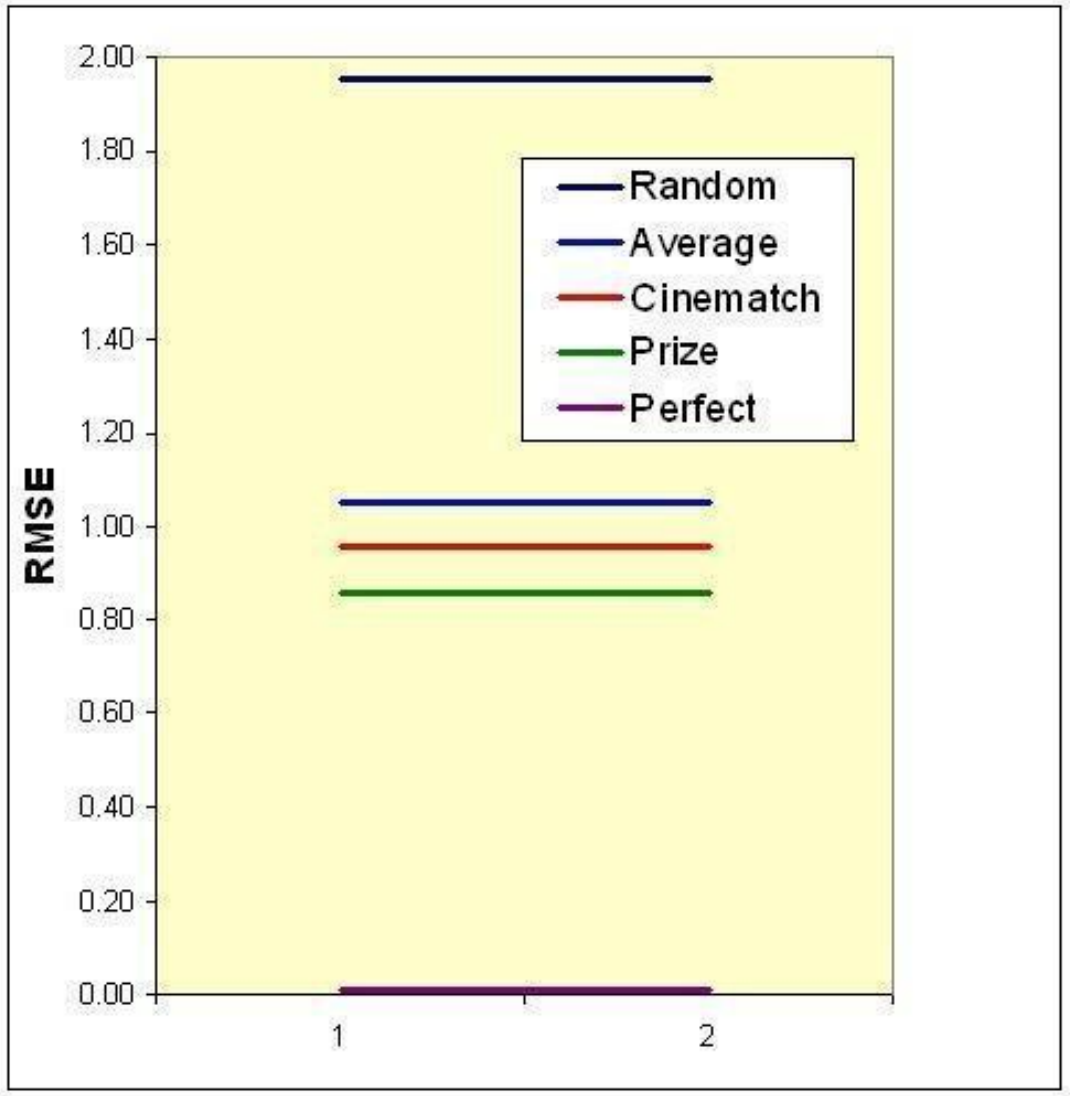
$$MAE_{NP} = \frac{\sum_{i,j} |v_{ij} - v_j|}{num.ratings}$$

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

NETFLIX

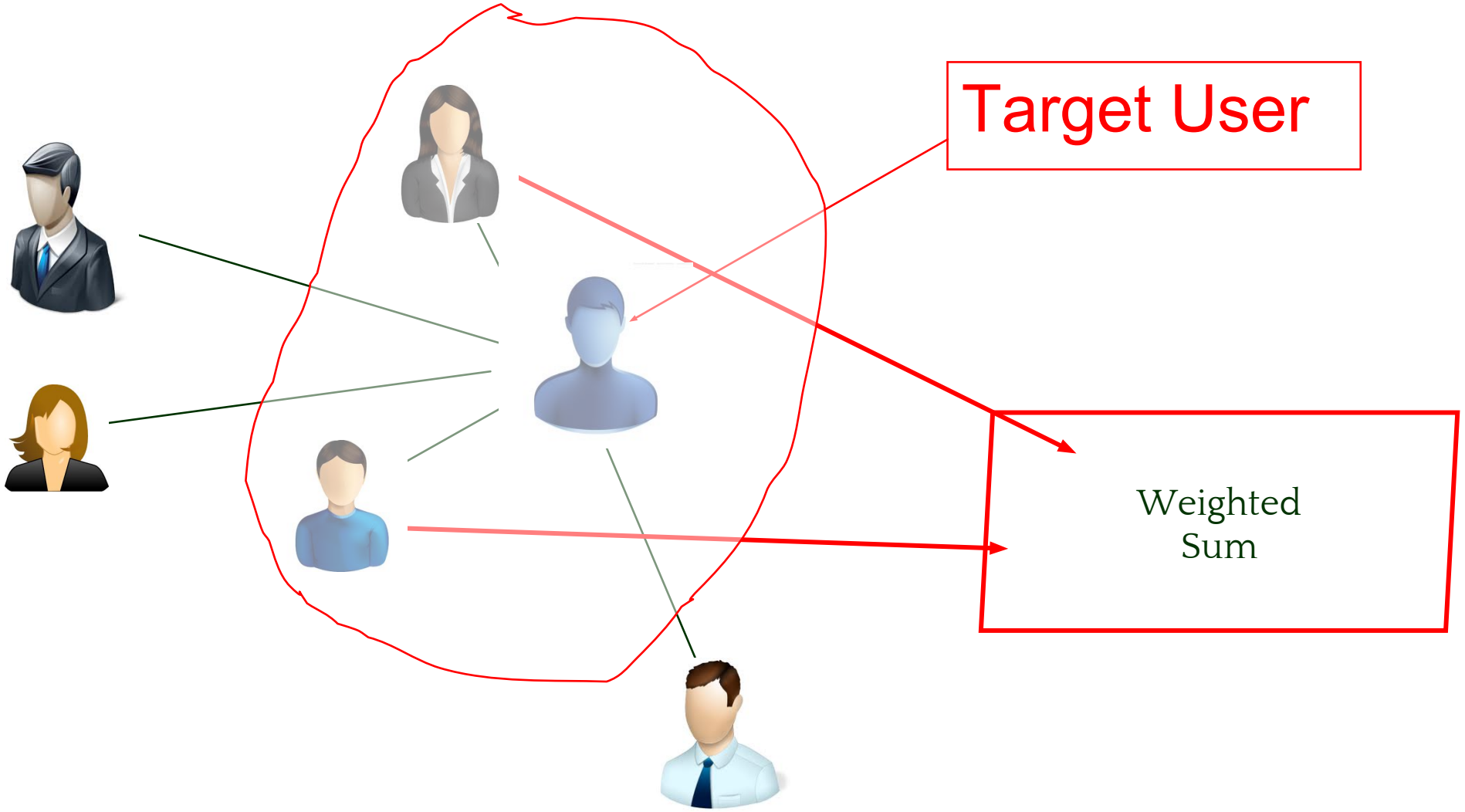
Personalized vs. Not Personalized

- Netflix Prize's first conclusion: it is really extremely simple to produce "reasonable" recommendations and extremely difficult to improve them.



User-based Collaborative Filtering

User-User Collaborative Filtering



UB Collaborative Filtering

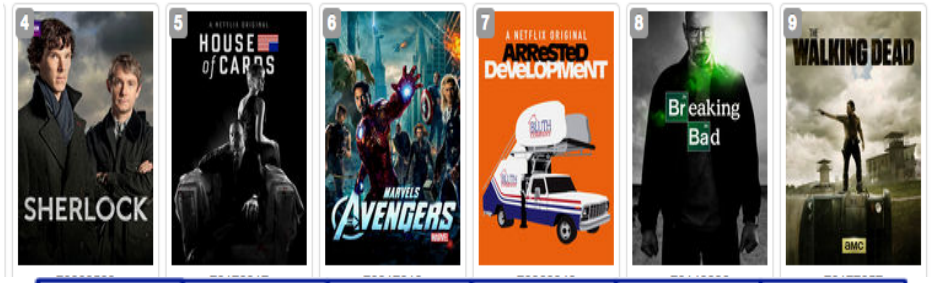
- A collection of user u_i, $i=1, \ldots n$ and a collection of products p_j, $j=1, \ldots, m$
- An $n \times m$ matrix of ratings v_{ij}, with $v_{ij} = ?$ if user i did not rate product j
- Prediction for user i and product j is computed

$$v_{ij}^{*} = K \sum_{v_{kj} \neq ?} u_{jk} v_{kj} \quad \text{or} \quad v_{ij}^{*} = v_i + K \sum_{v_{kj} \neq ?} u_{jk} (v_{kj} - v_k)$$

- Similarity can be computed by Pearson correlation

$$u_{ik} = \frac{\sum_j (v_{ij} - v_i)(v_{kj} - v_k)}{\sqrt{\sum_j (v_{ij} - v_i)^2 \sum_j (v_{kj} - v_k)^2}} \quad \text{or} \quad \cos(u_i, u_j) = \frac{\sum_{k=1}^{m} v_{ik} v_{jk}}{\sqrt{\sum_{k=1}^{m} v_{ik}^2 \sum_{k=1}^{m} v_{jk}^2}}$$

User-based CF Example



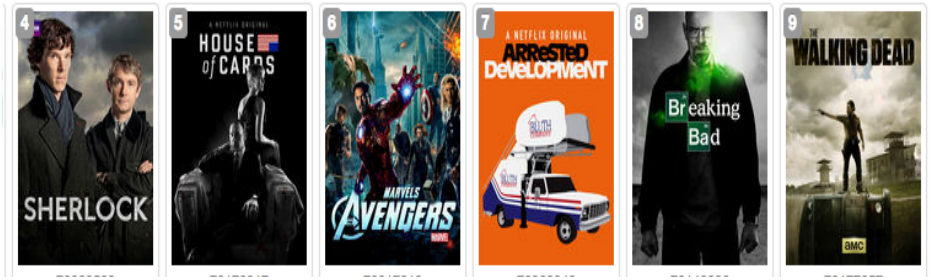
sim(u,v)

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

NA

NA

User-based CF Example

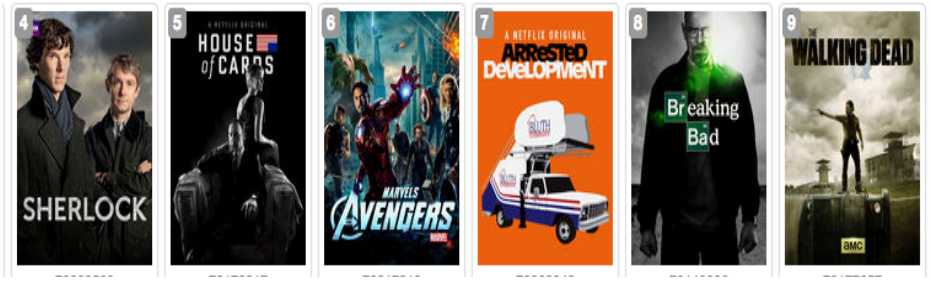


sim(u,v)

	2		2	4	5	
NA						
	5		4			1
0.87						
			5		2	
		1		5		4
			4			2
	4	5		1		
NA						

NETFLIX

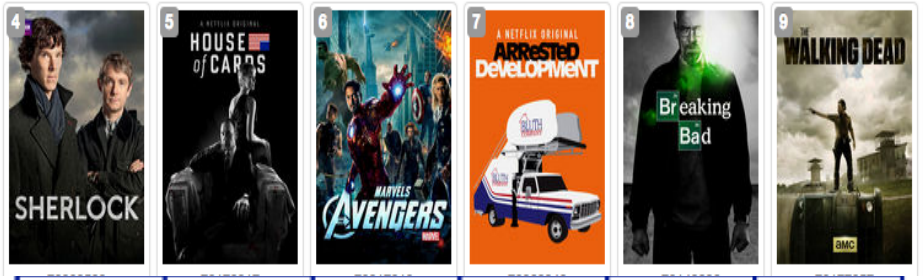
User-based CF Example



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

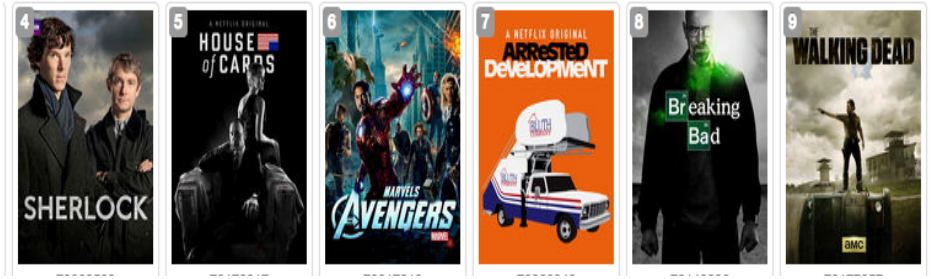
NETFLIX

User-based CF Example



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

User-based CF Example



						sim(u,v)
	2		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

NETFLIX

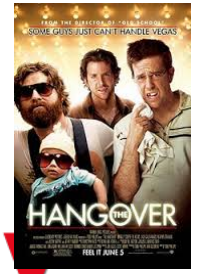
Challenges Of User-based CF Algorithms

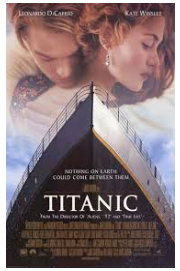
- Sparsity – evaluation of large item sets, users purchases are under 1%.
- Difficult to make predictions based on nearest neighbor algorithms =>Accuracy of recommendation may be poor.
- Scalability - Nearest neighbor require computation that grows with both the number of users and the number of items.
- Poor relationship among like minded but sparse-rating users.
- Solution : usage of latent models to capture similarity between users & items in a reduced dimensional space.

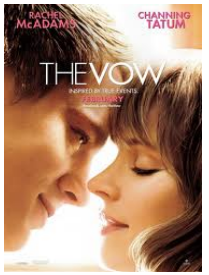
Item-based Collaborative Filtering

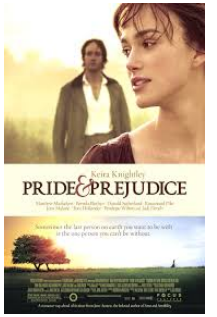
NETFLIX

Item-Item Collaborative Filtering









NETFLIX

Item Based CF Algorithm

- Look into the items the target user has rated
- Compute how similar they are to the target item
  - Similarity **only using** past **ratings** from other users!
- Select k most similar items.
- Compute Prediction by taking weighted average on the target user's ratings on the most similar items.

Item Similarity Computation

- Similarity between items *i* & *j* computed by finding users who have rated them and then applying a similarity function to their ratings.
- Cosine-based Similarity – items are vectors in the m dimensional user space (difference in rating scale between users is not taken into account).

$$S(i,j)=\cos(\vec{i},\vec{j})=\frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2*\|\vec{j}\|_2}$$

NETFLIX

Item Similarity Computation

- Correlation-based Similarity - using the Pearson-r correlation (used only in cases where the users rated both item I & item j).

$$S(i,j) = corr_{\bar{i},\bar{j}} = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_i)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_i)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R}_j)^2}}$$

  - $R_{u,i}$ = rating of user u on item i.
  - R_i = average rating of the i-th item.

Item Similarity Computation

- Adjusted Cosine Similarity – each pair in the co-rated set corresponds to a different user. (takes care of difference in rating scale).

$$S(i,j)=\frac{\sum_{u\in U}(R_{u,i}-\bar{R}_u)(R_{u,j}-\bar{R}_u)}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R}_u)^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R}_u)^2}}$$

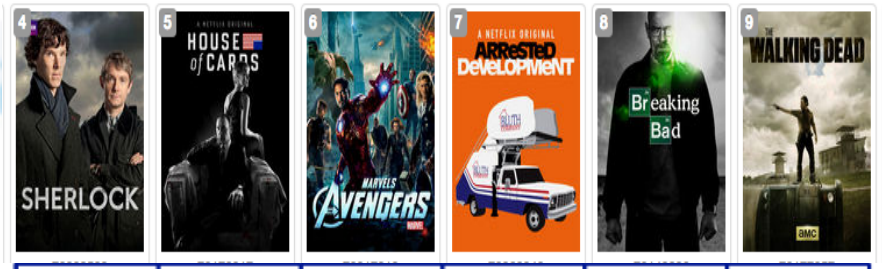
  - $R_{u,i}$ = rating of user u on item i.
  - R_u = average of the u-th user.

Prediction Computation

- Generating the prediction – look into the target users ratings and use techniques to obtain predictions.
- Weighted Sum – how the active user rates the similar items.

$$P_{u,i} = \frac{\sum_{all\ similar\ items,N} (S_{i,N} * R_{u,N})}{\sum_{all\ similar\ items,N} (|S_{i,N}|)}$$

Item-based CF Example

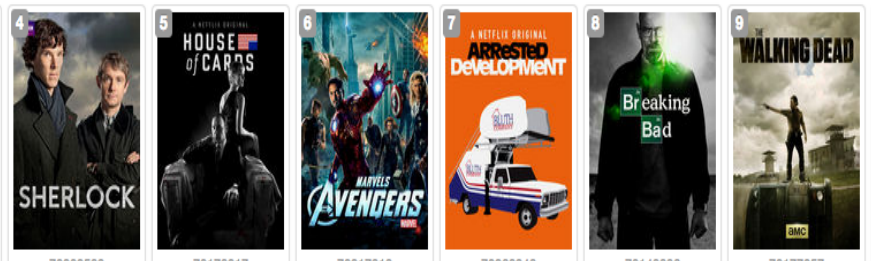


	2			4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		

sim(i,j) -1

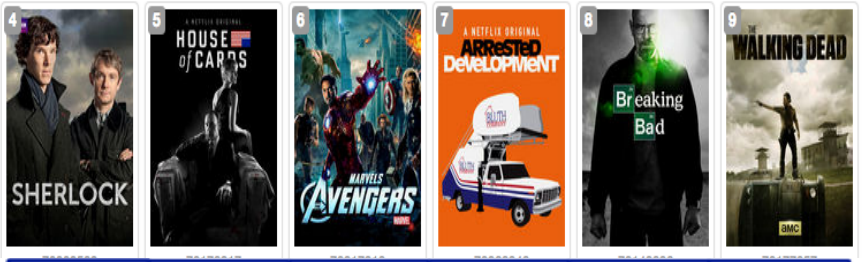
NETFLIX

Item-based CF Example



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

Item-based CF Example

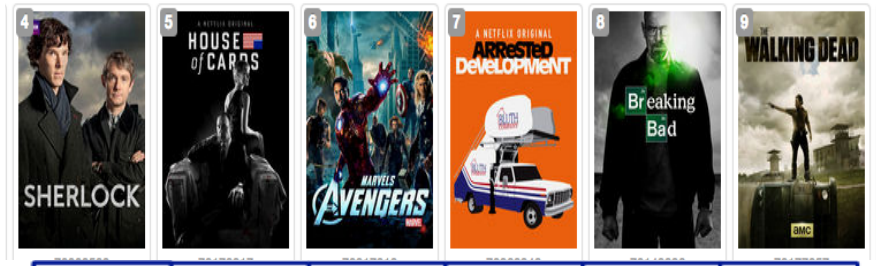


2			4	5	
5		4			1
		5		2	
	1		5		4
		4			2
4	5		1		

sim(i,j)	-1	-1	0.86

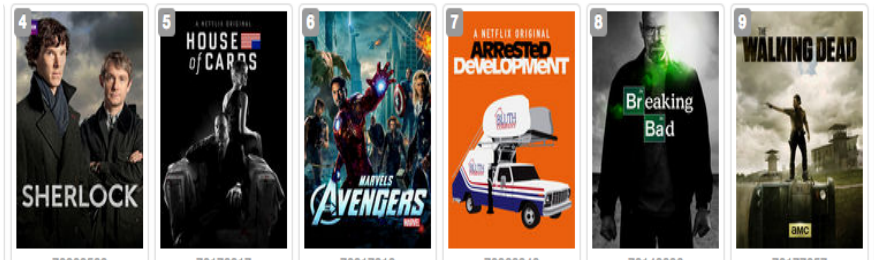
NETFLIX

Item-based CF Example



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

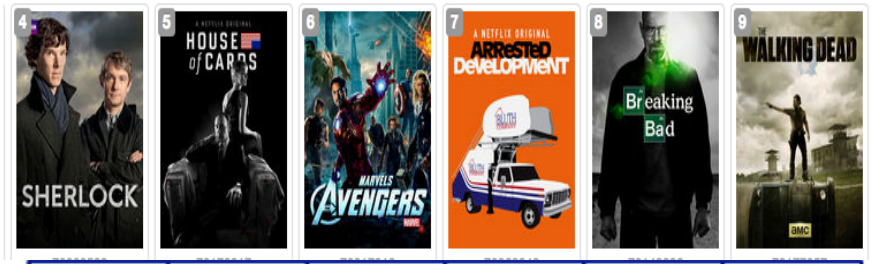
Item-based CF Example



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

sim(6,5) cannot be calculated

Item-based CF Example



	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	

NETFLIX

Performance Implications

- Bottleneck - Similarity computation.
- Time complexity, highly time consuming with millions of users and items in the database.
  - Isolate the neighborhood generation and predication steps.
  - “off-line component” / “model” – similarity computation, done earlier & stored in memory.
  - “on-line component” – prediction generation process.

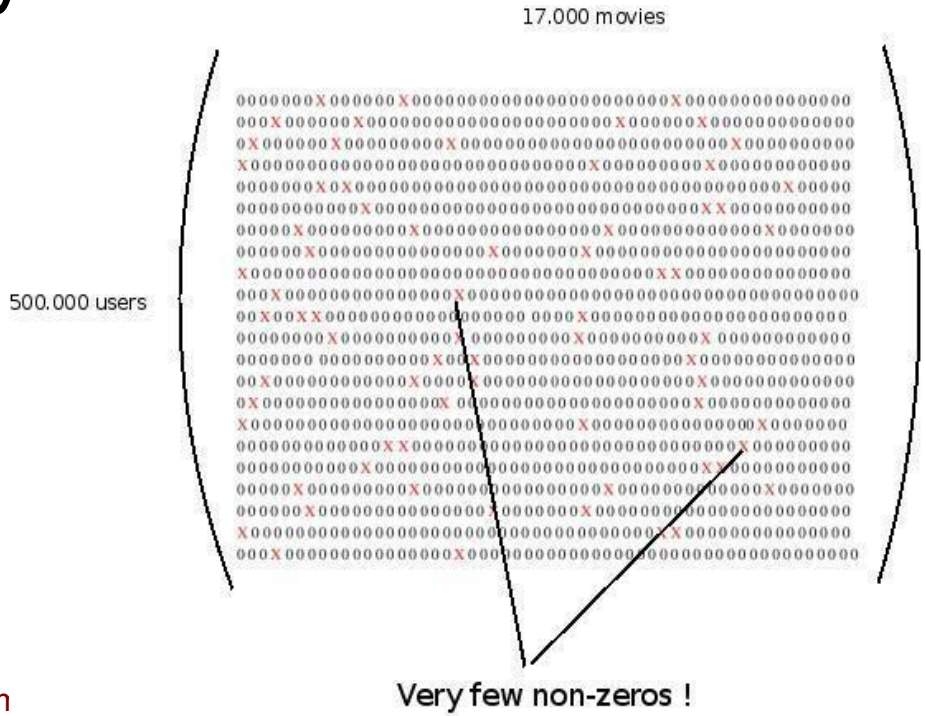
Recap: challenges of Nearest-neighbor Collaborative Filtering

The Sparsity Problem

- Typically: large product sets, user ratings for a small percentage of them
- Example Amazon: millions of books and a user may have bought hundreds of books –
  - the probability that two users that have bought 100 books have a common book (in a catalogue of 1 million books) is 0.01 (with 50 and 10 millions is 0.0002).
- Standard CF must have a number of users comparable to one tenth of the size of the product catalogue

The Sparsity Problem

- If you represent the Netflix Prize rating data in a User/Movie matrix you get...
  - 500,000 x 17,000 = 8,500 M positions
  - Out of which only 100M are not 0's!
- Methods of dimensionality reduction
  - Matrix Factorization
  - Clustering
  - Projection (PCA ...)



The Scalability Problem

- Nearest neighbor algorithms require computations that grows with both the number of customers and products
- With millions of customers and products a web-based recommender can suffer serious scalability problems
- The worst case complexity is O(mn) (m customers and n products)
- But in practice the complexity is O(m + n) since for each customer only a small number of products are considered
- Some clustering techniques like K-means can help

Performance Implications

- User-based CF – similarity between users is dynamic, precomupting user neighborhood can lead to poor predictions.
- Item-based CF – similarity between items is static.
- enables precomputing of item-item similarity => prediction process involves only a table lookup for the similarity values & computation of the weighted sum.

Other approaches to CF

Model-based Collaborative Filtering

NETFLIX

Model Based CF Algorithms

- Memory based
  - Use the entire user-item database to generate a prediction.
  - Usage of statistical techniques to find the neighbors – e.g. nearest-neighbor.
- Memory based
  - First develop a model of user
  - Type of model:
    - Probabilistic (e.g. Bayesian Network)
    - Clustering
    - Rule-based approaches (e.g. Association Rules)
    - Classification
    - Regression
    - LDA
    - ...

NETFLIX

Model-based CF: What we learned from the Netflix Prize

Netflix Prize

COMPLETED

What we were interested in:

- High quality *recommendations*

Proxy question:

- Accuracy in predicted rating
- Improve by 10% = $1million!

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

2007 Progress Prize

- Top 2 algorithms
  - SVD - Prize RMSE: 0.8914
  - RBM - Prize RMSE: 0.8990
- Linear blend Prize RMSE: 0.88
- Currently in use as part of Netflix' rating prediction component
- Limitations
  - Designed for 100M ratings, we have 5B ratings
  - Not adaptable as users add ratings
  - Performance issues

SVD/MF

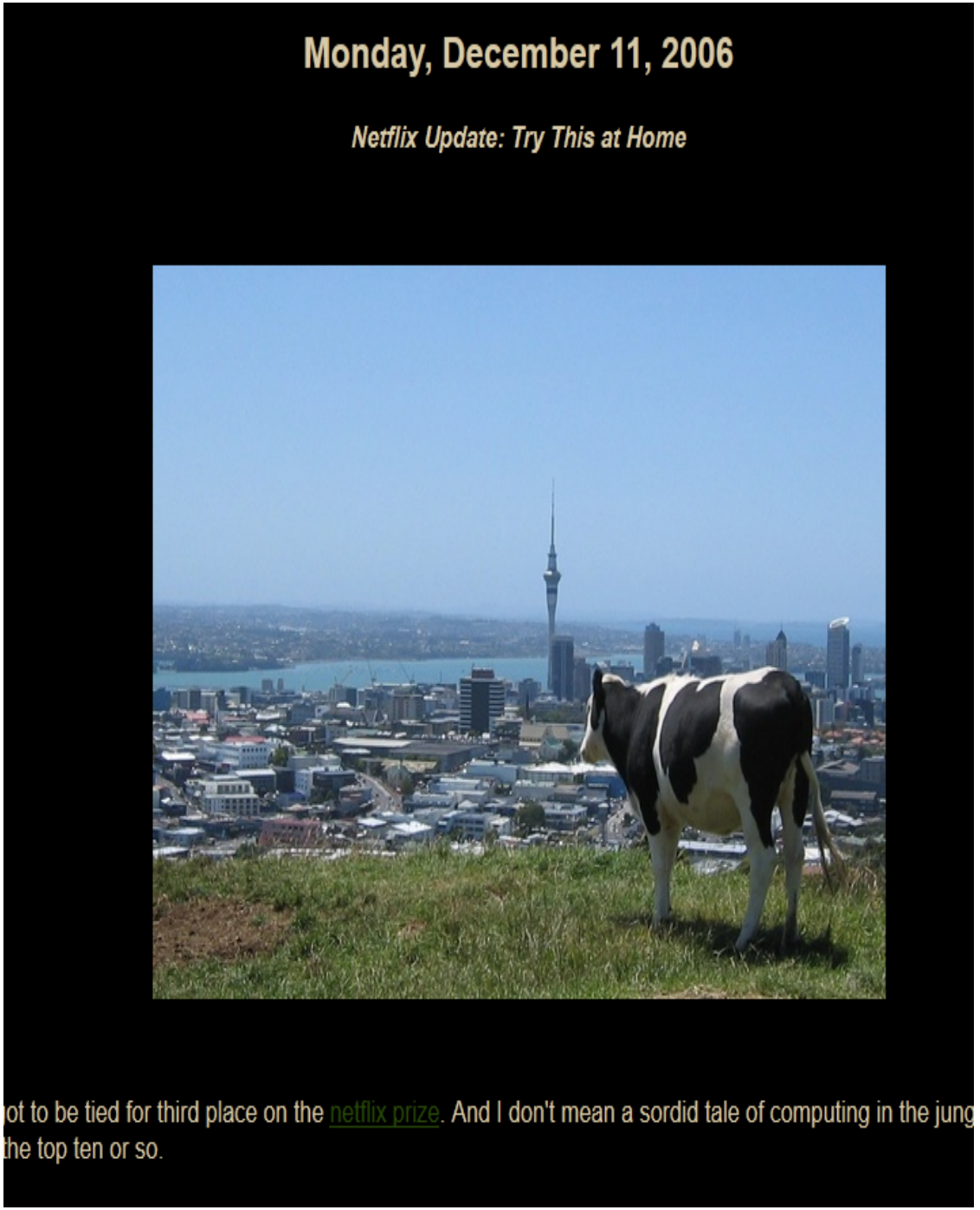
$$X[n \times m] = U[n \times r]\, S[r \times r]\, (V[m \times r])^T$$

$$\underset{m \times n}{X} = \underset{m \times r}{U} \; \underset{r \times r}{S} \; \underset{r \times n}{V^T}$$

- **X**: *m x n* matrix (e.g., m users, n videos)
- **U**: *m x r* matrix (m users, r factors)
- **S**: *r x r* diagonal matrix (strength of each 'factor') (r: rank of the matrix)
- **V**: r x n matrix (n videos, r factor)

Simon Funk's SVD

- One of the most interesting findings during the Netflix Prize came out of a blog post
- Incremental, iterative, and approximate way to compute the SVD using gradient descent



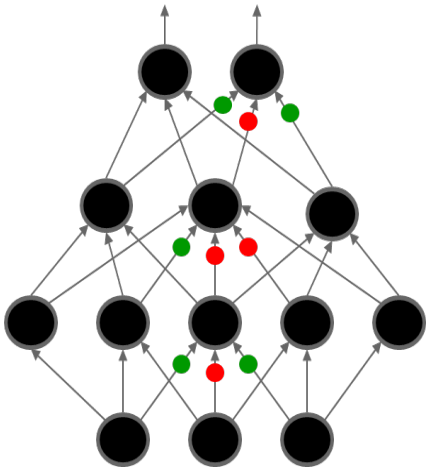
SVD for Rating Prediction

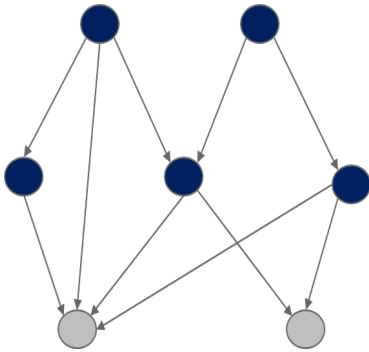
- User factor vectors $p_u \in \Re^f$ and item-factors vector $q_v \in \Re^f$
- Baseline (bias) $b_{uv} = \mu + b_u + b_v$ (user & item deviation from average)
- Predict rating as $r'_{uv} = b_{uv} + p_u^T q_v$
- SVD++ (Koren et. Al) asymmetric variation w. implicit feedback

$$r'_{uv} = b_{uv} + q_v^T \left(|R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) x_j + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

- Where
  - $q_v, x_v, y_v \in \Re^f$ are three item factor vectors
  - Users are not parametrized, but rather represented by:
  - R(u): items rated by user u
  - N(u): items for which the user has given implicit preference (e.g. rated vs. not rated)

Artificial Neural Networks – 4 generations

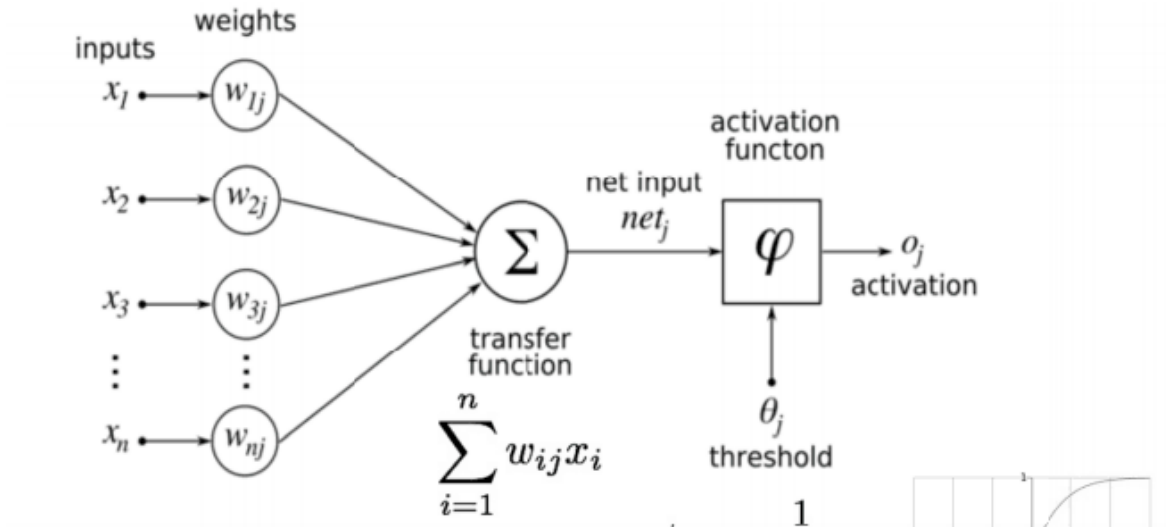




- 1st - Perceptrons (~60s)
  - Single layer of hand-coded features
  - Linear activation function
  - Fundamentally limited in what they can learn to do.
- 2nd - Back-propagation (~80s)
  - Back-propagate error signal to get derivatives for learning
  - Non-linear activation function
- 3rd - Belief Networks (~90s)
  - Directed acyclic graph composed of (visible & hidden) stochastic variables with weighted connections.
  - Infer the states of the unobserved variables & learn interactions between variables to make network more likely to generate observed data.

Restricted Boltzmann Machines

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



Restricted Boltzmann Machines

- Restrict the connectivity to make learning easier.
  - Only one layer of hidden units.
  - Although multiple layers are possible
  - No connections between hidden units.
  - Hidden units are independent given the visible states..
  - So we can quickly get an unbiased sample from the posterior distribution over hidden “causes” when given a data-vector
- RBMs can be stacked to form Deep Belief Networks (DBN) – 4th generation of ANNs

RBM for the Netflix Prize

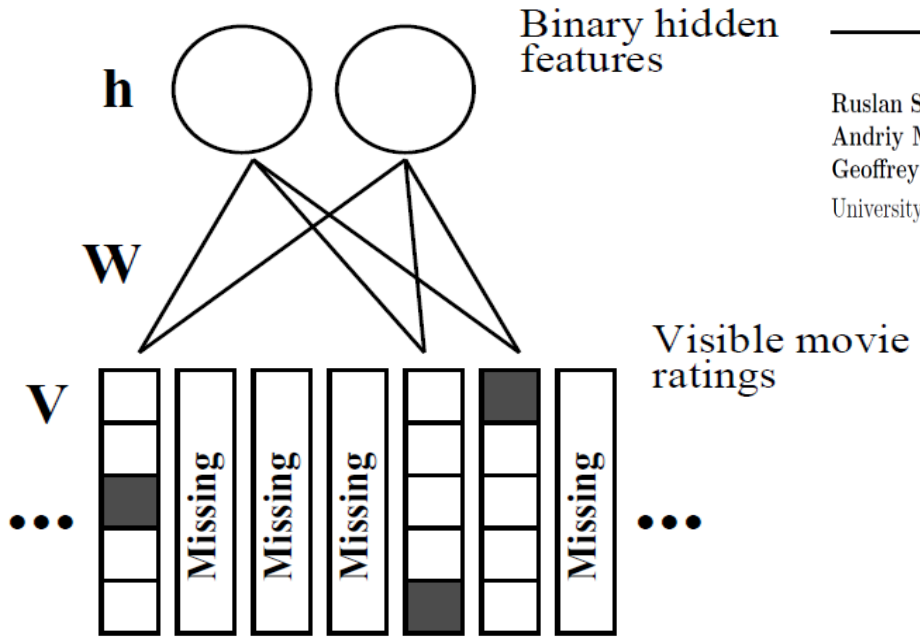


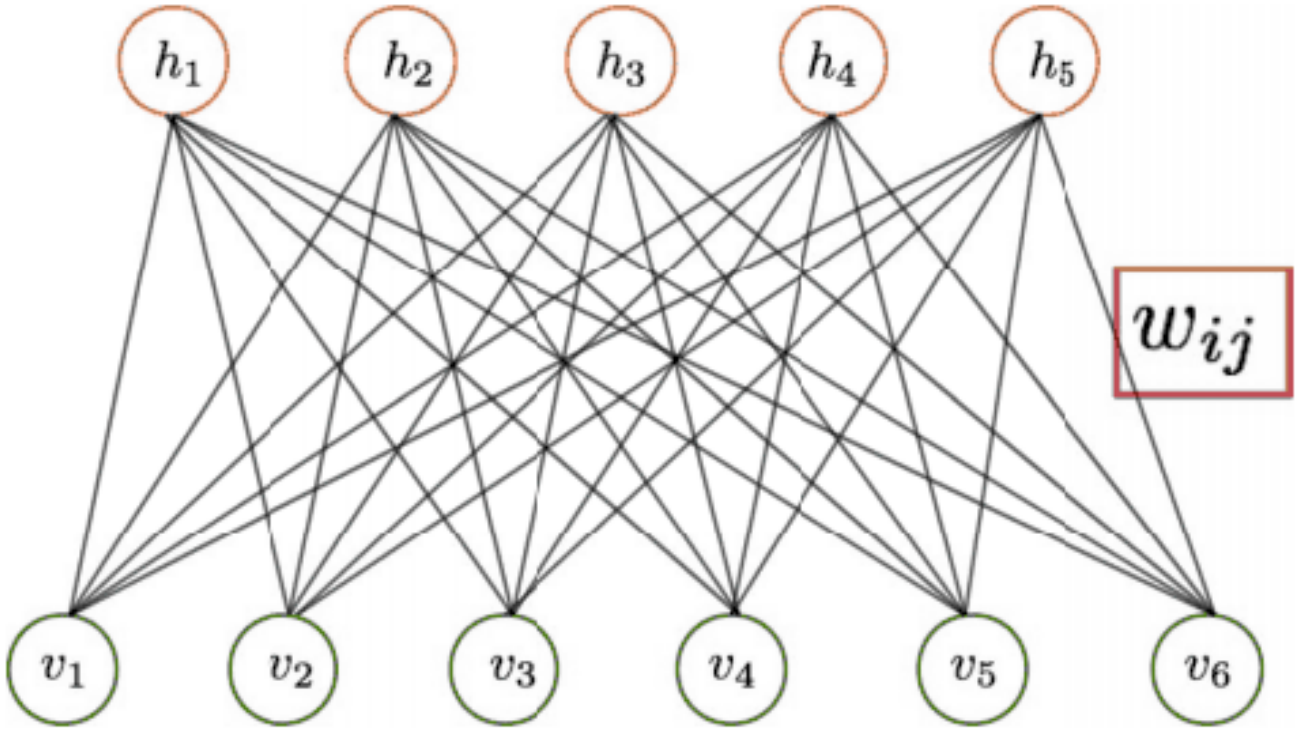
Figure 1. A restricted Boltzmann machine with binary hidden units and softmax visible units. For each user, the RBM only includes softmax units for the movies that user has rated. In addition to the symmetric weights between each hidden unit and each of the $K = 5$ values of a softmax unit, there are 5 biases for each softmax unit and one for each hidden unit. When modeling user ratings with an RBM that has Gaussian hidden units, the top layer is composed of linear units with Gaussian noise.

Restricted Boltzmann Machines for Collaborative Filtering

Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Andriy Mnih AMNIH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
University of Toronto, 6 King's College Rd., Toronto, Ontario M5S 3G4, Canada

NETFLIX

RBM for Recommendations



RBM for Recommendations

- Each unit in the visible layer is an item
- The number of hidden units is a parameter
- In the training phase, for each user:
  - If the user rated the item, vi is activated
  - Activation states of all vi are the inputs to hj
  - Based on the activation, hj is computed
  - The activation state of hj becomes input to vi
  - The activation state of vi is recalculated
  - Difference between current and past activation state for vi is used to update the weights wij and the thresholds

RBM for Recommendations

- In the prediction phase with a trained RBM:
  - For the items of the user the vi are activated
  - Based on this the state of the hj is computed
  - The activation of hj is used as input to recompute the state of vi
  - Activation probabilities are used to recommend items

Putting all together

- Remember that current production model includes an **ensemble** of both SVD++ and RBMs

Clustering

Clustering

- Another way to make recommendations based on past purchases is to **cluster** customers
- Each cluster will be assigned typical preferences, based on preferences of customers who belong to the cluster
- Customers within each cluster will receive recommendations computed at the cluster level

Clustering

	Book1	Book2	Book3	Book4	Book5	Book6
Customer A	X			X		
Customer B		X	X		X	
Customer C		X	X			
Customer D		X				X
Customer E	X				X	

Customers B, C and D are « clustered » together.
Customers A and E are clustered into another separate group

- « Typical » preferences for **CLUSTER** are:
  - Book 2, very high
  - Book 3, high
  - Books 5 and 6, may be recommended
  - Books 1 and 4, not recommended at all

Clustering

	Book1	Book2	Book3	Book4	Book5	Book6
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	

How does it work?

- Any customer that shall be classified as a member of **CLUSTER** will receive recommendations based on preferences of the group:
  - Book 2 will be highly recommended to Customer F
  - Book 6 will also be recommended to some extent

Clustering

Pros:

- Clustering techniques can be used to work on aggregated data
- Can also be applied as a first step for shrinking the selection of relevant neighbors in a collaborative filtering algorithm and improve performance
- Can be used to capture latent similarities between users or items

Cons:

- Recommendations (per cluster) may be less relevant than collaborative filtering (per individual)

Locality-sensitive Hashing (LSH)

- Method for grouping similar items in highly dimensional spaces
- Find a hashing function s.t. similar items are grouped in the same buckets
- Main application is Nearest-neighbors
  - Hashing function is found iteratively by concatenating random hashing functions
  - Addresses one of NN main concerns: performance

Association Rules

Association rules

- Past purchases are transformed into relationships of common purchases

	Book1	Book2	Book3	Book4	Book5	Book6
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	

		Also bought...					
		Book1	Book2	Book3	Book4	Book5	Book6
Customers who bought...	Book1				1	1	
	Book2			2		1	1
	Book3		2			2	
	Book4	1					
	Book5	1	1	2			
	Book6		1				

Association rules

- These association rules are then used to made recommendations
- If a visitor has some interest in Book 5, she will be recommended to buy Book 3 as well
- Recommendations are constrained to some minimum levels of confidence

		Alsobought..					
		Book1	Book2	Book3	Book4	Book5	Book6
Customers whobought..	Book1				1	1	
	Book2			2		1	1
	Book3		2			2	
	Book4	1					
	Book5	1	1	2			
	Book6		1				

NETFLIX

Association rules

Pros:

- Fast to implement (A priori algorithm for frequent itemset mining)
- 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

Cons:

- Not suitable if knowledge of preferences change rapidly
- It is tempting to not apply restrictive confidence rules
  → May lead to literally *stupid* recommendations

Classifiers

Classifiers

- **Classifiers** are general computational models trained using positive and negative examples
- They may take in inputs:
  - Vector of item features (action / adventure, Bruce Willis)
  - Preferences of customers (like action / adventure)
  - Relations among item
- E.g. Logistic Regression, Bayesian Networks, Support Vector Machines, Decision Trees, etc...

Classifiers

- Classifiers can be used in CF and CB Recommenders
- Pros:
  - Versatile
  - Can be combined with other methods to improve accuracy of recommendations
- Cons:
  - Need a relevant training set
  - May overfit (Regularization)

Limitations of Collaborative Filtering

Limitations of Collaborative Filtering

- **Cold Start**: There needs to be enough other users already in the system to find a match. New items need to get enough ratings.
- **Popularity Bias**: Hard to recommend items to someone with unique tastes.
  - Tends to recommend popular items (items from the tail do not get so much data)

Cold-start

- New User Problem: To make accurate recommendations, the system must first learn the user's preferences from the ratings.
  - Several techniques proposed to address this. Most use the hybrid recommendation approach, which combines content-based and collaborative techniques.
- New Item Problem: New items are added regularly to recommender systems. Until the new item is rated by a substantial number of users, the recommender system is not able to recommend it.

Index

NETFLIX

2.2 Content-based Recommenders

Content-Based Recommendations

- Recommendations based on information on the content of items rather than on other users' opinions/interactions
- Use a machine learning algorithm to induce a model of the users preferences from examples based on a featural description of content.
- In *content-based* recommendations, the system tries to recommend items **similar** to those a given user has liked in the past
- A pure content-based recommender system makes recommendations for a user based solely on the profile built up by **analyzing the content** of items which that user has rated in the past.

What is content?

- What is the **content** of an item?
- It can be explicit **attributes** or **characteristics** of the item. For example for a film:
  - Genre: Action / adventure
  - Feature: Bruce Willis
  - Year: 1995
- It can also be **textual content** (title, description, table of content, etc.)
  - Several techniques to compute the distance between two textual documents
  - Can use NLP techniques to extract content features
- Can be extracted from the signal itself (audio, image)

Content-Based Recommendation

- Common for recommending **text-based products** (web pages, usenet news messages,)
- Items to recommend are “described” by their associated **features** (e.g. keywords)
- **User Model** structured in a “similar” way as the content: features/keywords more likely to occur in the preferred documents (lazy approach)
  - Text documents recommended based on a comparison between their content (words appearing) and user model (a set of preferred words)
- The user model can also be a **classifier** based on whatever technique (Neural Networks, Naïve Bayes...)

Advantages of CB 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
  - No first-rater problem.
- Can provide explanations of recommended items by listing content-features that caused an item to be recommended.

Disadvantages of CB Approach

- Requires content that can be encoded as meaningful features.
- Some kind of items are not amenable to easy feature extraction methods (e.g. movies, music)
- Even for texts, IR techniques cannot consider multimedia information, aesthetic qualities, download time…
  - If you rate positively a page it 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
- Easy to overfit (e.g. for a user with few data points we may "pigeon hole" her)

Content-based Methods

- *Let Content(s)* be an *item profile,* i.e. a set of attributes characterizing item *s*.
- Content usually described with keywords.
- "Importance" (or "informativeness") of word k_j in document d_j is determined with some weighting measure w_{ij}
- One of the best-known measures in IR is the term frequency/inverse document frequency (TF-IDF).

$$\mathrm{tf}(t,d) = 0.5 + \frac{0.5 \times \mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}} \qquad \mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D)$$

NETFLIX

Content-based User Profile

- Let *ContentBasedProfile(c)* be the profile of user *c* containing preferences of this user profiles are obtained by:
  - analyzing the content of the previous items
  - using keyword analysis techniques
- For example, *ContentBasedProfile(c)* can be defined as a vector of weights $(w_{c1}, \ldots, w_{ck})$, where weight w_{ci} denotes the importance of keyword *ki* to user *c*

Similarity Measures

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

$$u(c,s) = \text{score}(ContentBasedProfile(c), Content(s))$$

- Both *ContentBasedProfile(c)* of user *c* and *Content(s)* of document *s* can be 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.

$$u(c,s) = \cos(\vec{w}_c, \vec{w}_s) = \frac{\vec{w}_c \cdot \vec{w}_s}{\|\vec{w}_c\|_2 \times \|\vec{w}_s\|_2} = \frac{\sum_{i=1}^{K} w_{i,c} w_{i,s}}{\sqrt{\sum_{i=1}^{K} w_{i,c}^2} \sqrt{\sum_{i=1}^{K} w_{i,s}^2}}$$

Statistical and Machine Learning Approaches

Other techniques are feasible

- Bayesian classifiers and various machine learning techniques, including clustering, decision trees, and artificial neural networks.

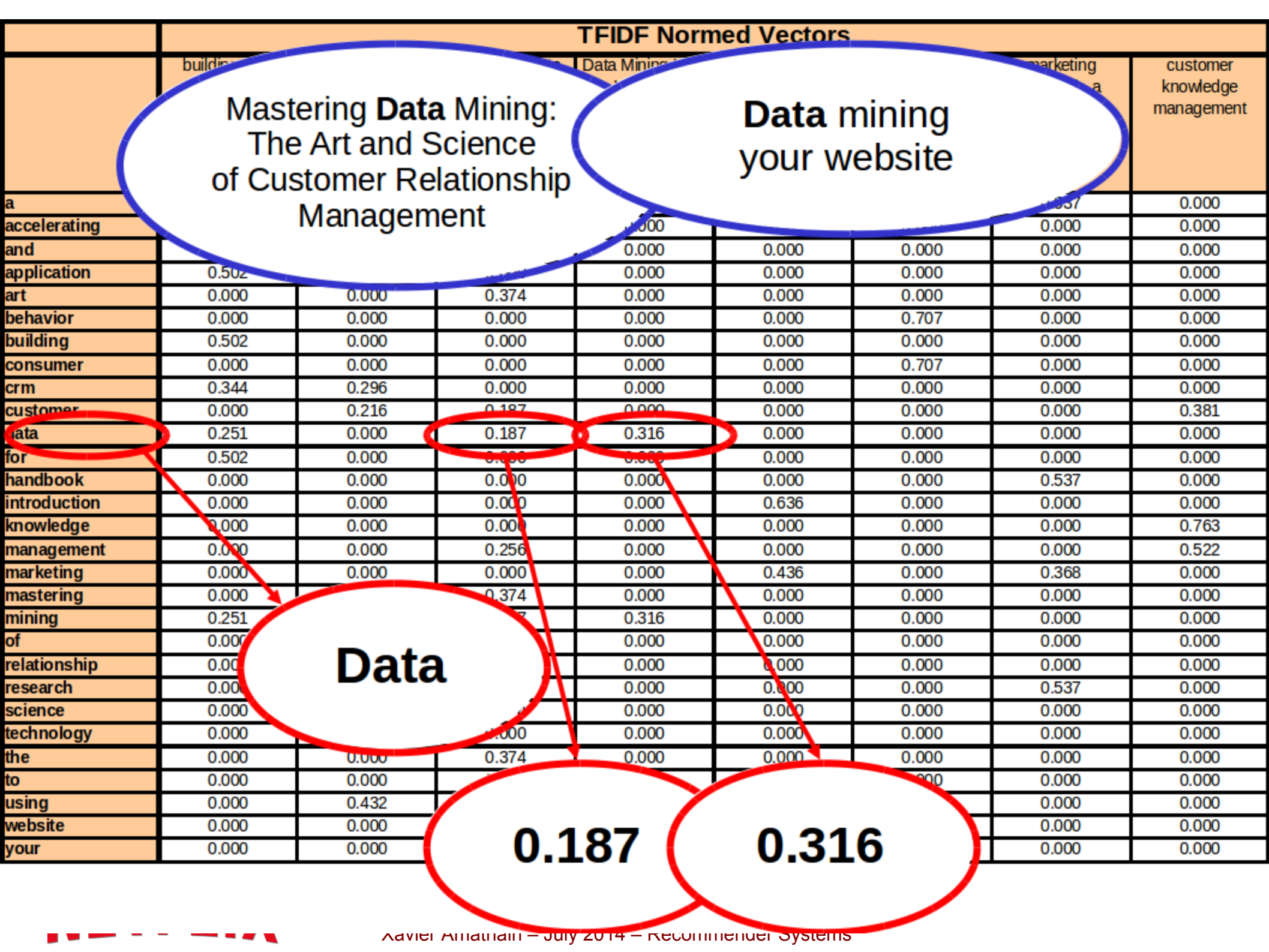
These methods use models learned from the underlying data rather than heuristics.

- For example, based on a set of Web pages that were rated as “relevant” or “irrelevant” by the user, the naive bayesian classifier can be used to classify unrated Web pages.

Content-based Recommendation. An unrealistic example

- An (unrealistic) example: how to compute recommendations between 8 books based only on their title?
- A customer is interested in the following book:”Building data mining applications for CRM”
- Books selected:
  - Building data mining applications for CRM
  - 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 behavior
  - marketing research, a handbook
  - Customer knowledge management

	COUNT							
	building data mining applications for crm	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 behavior	marketing research, a handbook	customer knowledge management
a							1	
accelerating		1						
and		1	1					
application	1							
art			1					
behavior						1		
building	1							
consumer						1		
crm	1	1						
customer		1	1					1
data	1		1	1				
for	1							
handbook							1	
introduction					1			
knowledge								1
management			1					1
marketing					1		1	
mastering			1					
mining	1		1	1				
of			1					
relationship		2	1					
research							1	
science			1					
technology		1						
the			1					
to					1			
using		1						
website				1				
your				1				



								customer knowledge management
a								0.000
accelerating							0.000	0.000
and				0.000	0.000	0.000	0.000	0.000
application	0.502			0.000	0.000	0.000	0.000	0.000
art	0.000	0.000	0.374	0.000	0.000	0.000	0.000	0.000
behavior	0.000	0.000	0.000	0.000	0.000	0.707	0.000	0.000
building	0.502	0.000	0.000	0.000	0.000	0.000	0.000	0.000
consumer	0.000	0.000	0.000	0.000	0.000	0.707	0.000	0.000
crm	0.344	0.296	0.000	0.000	0.000	0.000	0.000	0.000
customer	0.000	0.216	0.187	0.000	0.000	0.000	0.000	0.381
data	0.251	0.000	0.187	0.316	0.000	0.000	0.000	0.000
for	0.502	0.000			0.000	0.000	0.000	0.000
handbook	0.000	0.000	0.000	0.000	0.000	0.000	0.537	0.000
introduction	0.000	0.000	0.000	0.000	0.636	0.000	0.000	0.000
knowledge	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.763
management	0.000	0.000	0.256	0.000	0.000	0.000	0.000	0.522
marketing	0.000	0.000	0.000	0.000	0.436	0.000	0.368	0.000
mastering	0.000		0.374	0.000	0.000	0.000	0.000	0.000
mining	0.251			0.316	0.000	0.000	0.000	0.000
of	0.000			0.000	0.000	0.000	0.000	0.000
relationship	0.000			0.000	0.000	0.000	0.000	0.000
research	0.000			0.000	0.000	0.000	0.537	0.000
science	0.000			0.000	0.000	0.000	0.000	0.000
technology	0.000		0.000	0.000	0.000	0.000	0.000	0.000
the	0.000	0.000	0.374	0.000	0.000	0.000	0.000	0.000
to	0.000	0.000				0.000	0.000	0.000
using	0.000	0.432					0.000	0.000
website	0.000	0.000					0.000	0.000
your	0.000	0.000					0.000	0.000

Content-based Recommendation

- The system computes distances between this book and the 7 others
- The « closest » books are recommended:
  - **#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
  - **Not recommended:** Introduction to marketing
  - **Not recommended:** Consumer behavior
  - **Not recommended:** marketing research, a handbook
  - **Not recommended:** Customer knowledge management

A word of caution

Recommending New Movies: Even a Few Ratings Are More Valuable Than Metadata

István Pilászy *
Dept. of Measurement and Information Systems
Budapest University of Technology and Economics
Magyar Tudósok krt. 2.
Budapest, Hungary
pila@mit.bme.hu

Domonkos Tikk *,†
Dept. of Telecom. and Media Informatics
Budapest University of Technology and Economics
Magyar Tudósok krt. 2.
Budapest, Hungary
tikk@tmit.bme.hu

ABSTRACT

The Netflix Prize (NP) competition gave much attention to collaborative filtering (CF) approaches. Matrix factorization (MF) based CF approaches assign low dimensional feature vectors to users and items. We link CF and content-based filtering (CBF) by finding a linear transformation that transforms user or item descriptions so that they are as close as possible to the feature vectors generated by MF for CF.

We propose methods for explicit feedback that are able to handle 140 000 features when feature vectors are very sparse. With movie metadata collected for the NP movies we show that the prediction performance of the methods is comparable to that of CF, and can be used to predict user preferences on new movies.

We also investigate the value of movie metadata compared to movie ratings in regards of predictive power. We compare

1. INTRODUCTION

The goal of recommender systems is to give personalized recommendation on items to users. Typically the recommendation is based on the former and current activity of the users, and metadata about users and items, if available.

There are two basic strategies that can be applied when generating recommendations. Collaborative filtering (CF) methods are based only on the activity of users, while content-based filtering (CBF) methods use only metadata. In this paper we propose hybrid methods, which try to benefit from both information sources.

The two most important families of CF methods are matrix factorization (MF) and neighbor-based approaches. Usually, the goal of MF is to find a low dimensional representation for both users and movies, i.e. each user and movie is associated with a feature vector. Movie metadata (which

Index

3. Novel approaches to Recommendation: Beyond “traditional” CF and CB

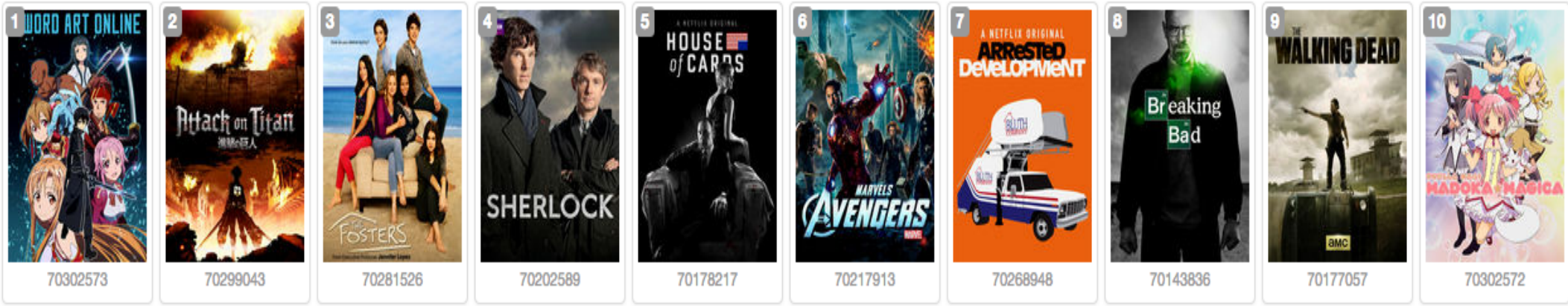
NETFLIX

Ranking

Ranking

- Most recommendations are presented in a sorted list
- Recommendation can be understood as a ranking problem
- Popularity is the obvious baseline
- Ratings prediction is a clear secondary data input that allows for personalization
- Many other features can be added

Ranking by ratings



4.7 4.6 4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5

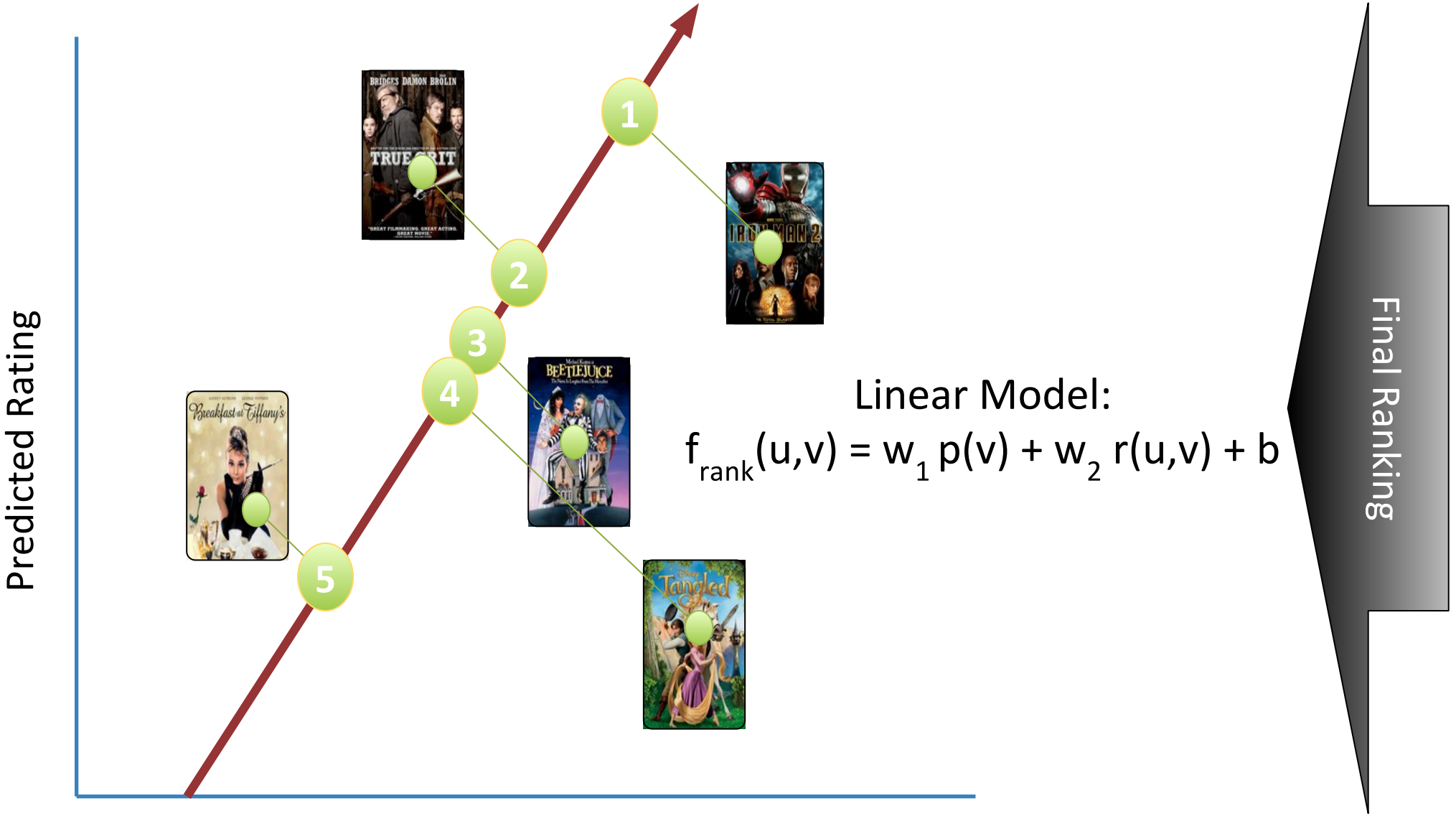
Niche titles

High average ratings… by those who would watch it

NETFLIX

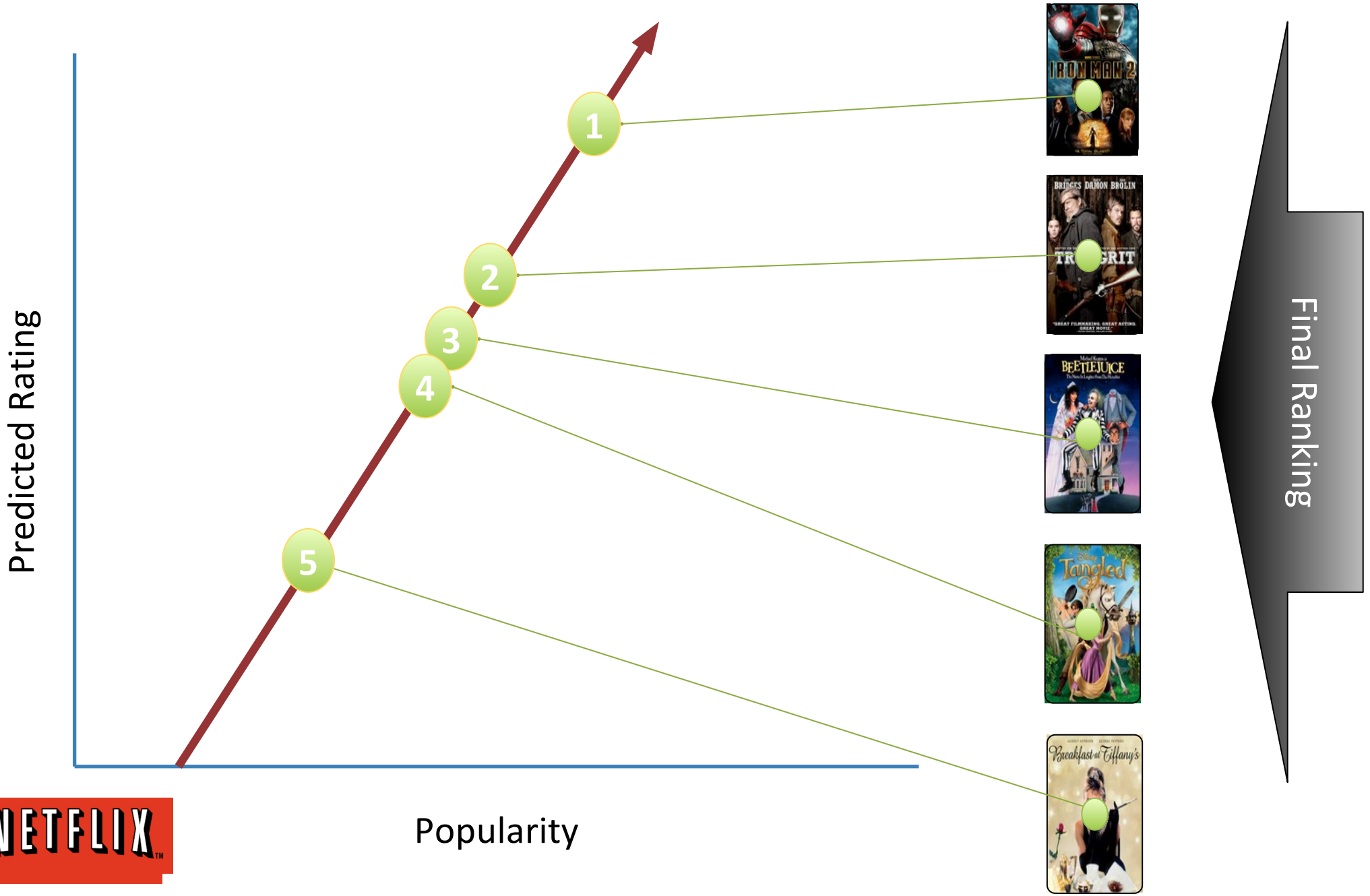
NETFLIX

Example: Two features, linear model

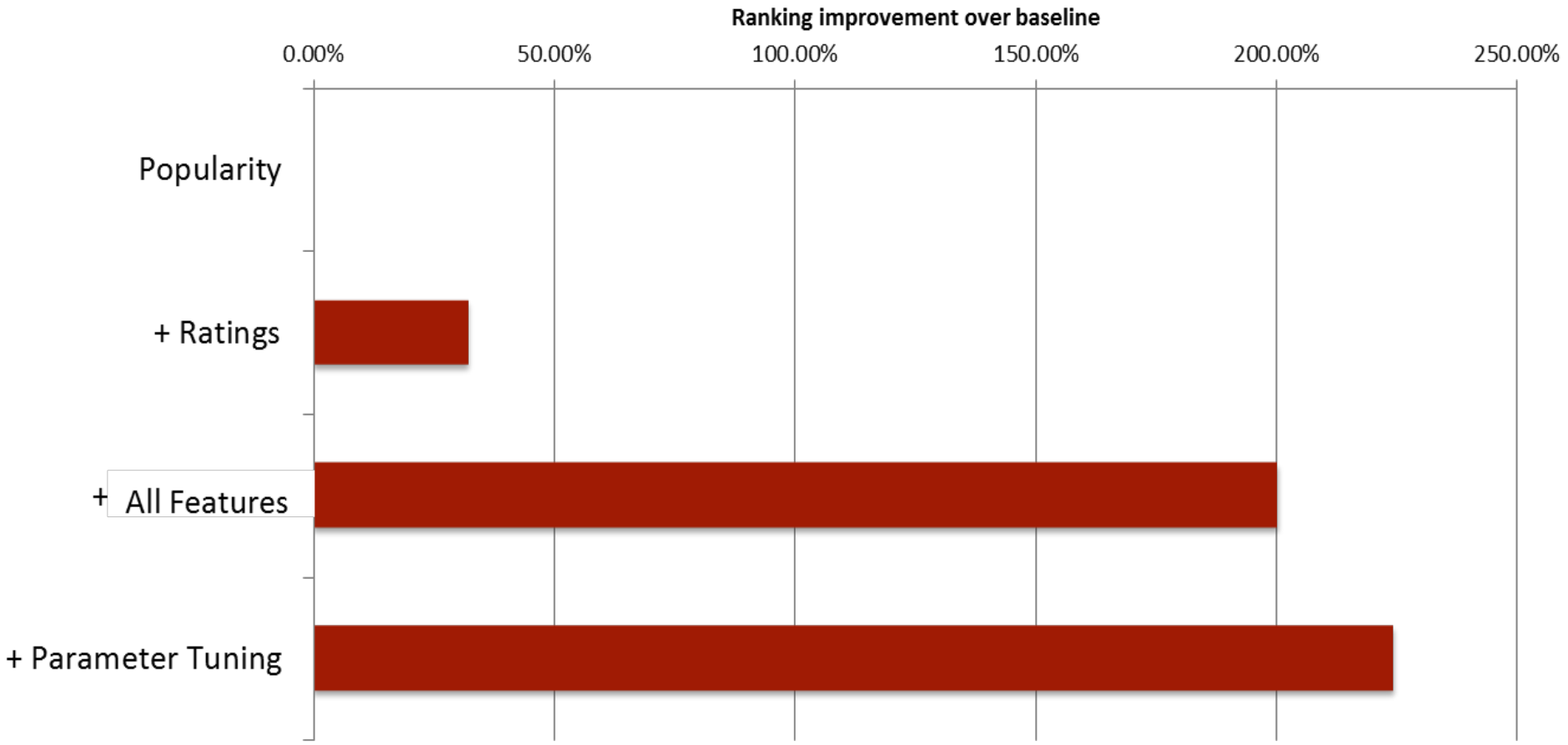


NETFLIX

Example: Two features, linear model



Ranking


Ranking improvement over baseline
0.00%
50.00%
100.00%
150.00%
200.00%
250.00%
Popularity
+ Ratings
+ All Features
+ Parameter Tuning

Learning to rank

- Machine learning problem: goal is to construct ranking model from training data
- Training data can be a partial order or binary judgments (relevant/not relevant).
- Resulting order of the items typically induced from a numerical score
- Learning to rank is a key element for personalization
- You can treat the problem as a standard supervised classification problem

Learning to rank - Metrics

- Quality of ranking measured using metrics as
  - Normalized Discounted Cumulative Gain
  - Mean Reciprocal Rank (MRR)
  - Fraction of Concordant Pairs (FCP)
  - Others…
- But, it is hard to optimize machine-learned models directly on these measures (they are not differentiable)
- Recent research on models that directly optimize ranking measures

Learning to rank - Approaches

1. Pointwise
   - Ranking function minimizes loss function defined on individual relevance judgment
   - Ranking score based on regression or classification
   - Ordinal regression, Logistic regression, SVM, GBDT, …
2. Pairwise
   - Loss function is defined on pair-wise preferences
   - Goal: minimize number of inversions in ranking
   - Ranking problem is then transformed into the binary classification problem
   - RankSVM, RankBoost, RankNet, FRank…

Learning to rank - Approaches

3. Listwise
   - Indirect Loss Function
     - RankCosine: similarity between ranking list and ground truth as loss function
     - ListNet: KL-divergence as loss function by defining a probability distribution
     - Problem: optimization of listwise loss function may not optimize IR metrics
   - Directly optimizing IR measures (difficult since they are not differentiable)

Learning to rank - Approaches

3. Listwise
   - Directly optimize IR measures through Genetic Programming
   - Directly optimize measures with Simulated Annealing
   - Gradient descent on smoothed version of objective function (e.g. CLiMF presented at Recsys 2012 or TFMAP at SIGIR 2012)
   - SVM-MAP relaxes the MAP metric by adding it to the SVM constraints
   - AdaRank uses boosting to optimize NDCG

Index

Context-aware Recommendation

Context-aware Recommendations

Context is an important factor to consider in personalized Recommendation

Context-aware Recommendations

Context is an important factor to consider in personalized Recommendation

Context-aware Recommendations

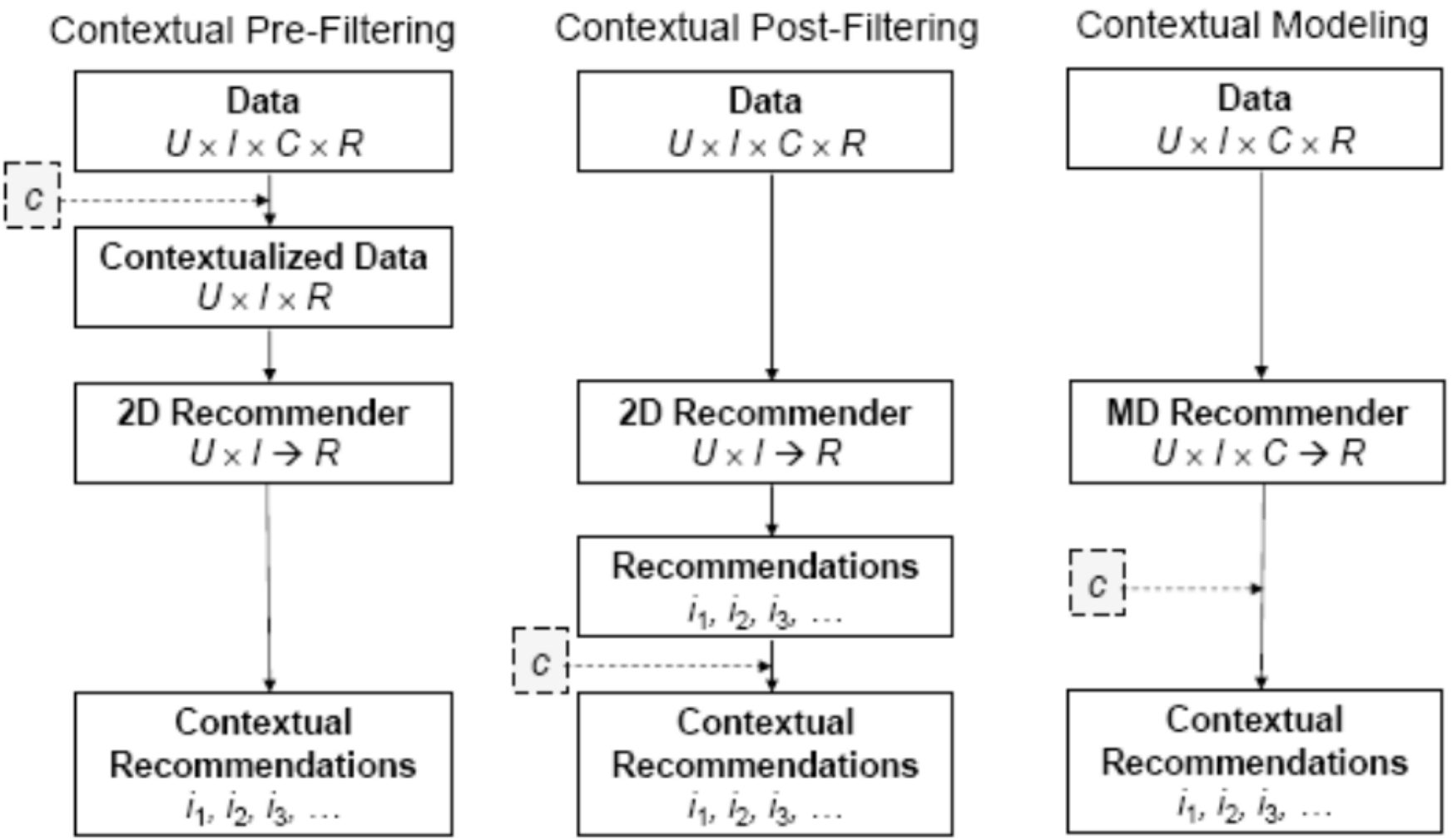
- Pre-Filtering Techniques
- Post-Filtering Techniques
- Contextual modeling

Context-aware Recommendations

- **Three types of Architecture for using context in recommendation (Adomavicius, Tuzhilin, 2008)**
  - Contextual Pre-filtering
    - Context information used to select relevant portions of data
  - Contextual Post-filtering
    - Contextual information is used to filter/constrain/re-rank final set of recommendations
  - Contextual Modeling
    - Context information is used directly as part of learning preference models
- **Variants and combinations of these are possible**
- **Originally introduced based on the representational view**
  - Though these architectures are also generally applicable in the interactional view

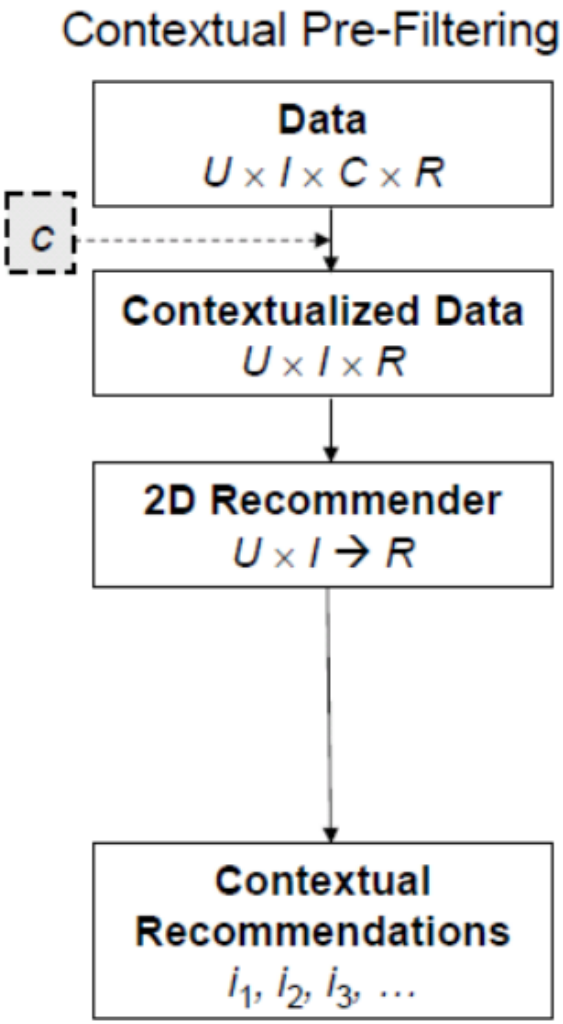
Context-aware Recommendations

From Adomavicius, Tuzhilin, 2008



Context-aware Recommendations

- *Pre-Filtering*: using contextual information to select the most relevant data for generating recommendations
  - Context *c* serves as a *query* to select relevant ratings data Data(User, Item, Rating, Context), i.e.,
  - SELECT User, Item, Rating
    FROM Data
    WHERE Context = *c*
- *Example:* if a person wants to see a movie on Saturday, *only* the Saturday rating data is used to recommend movies



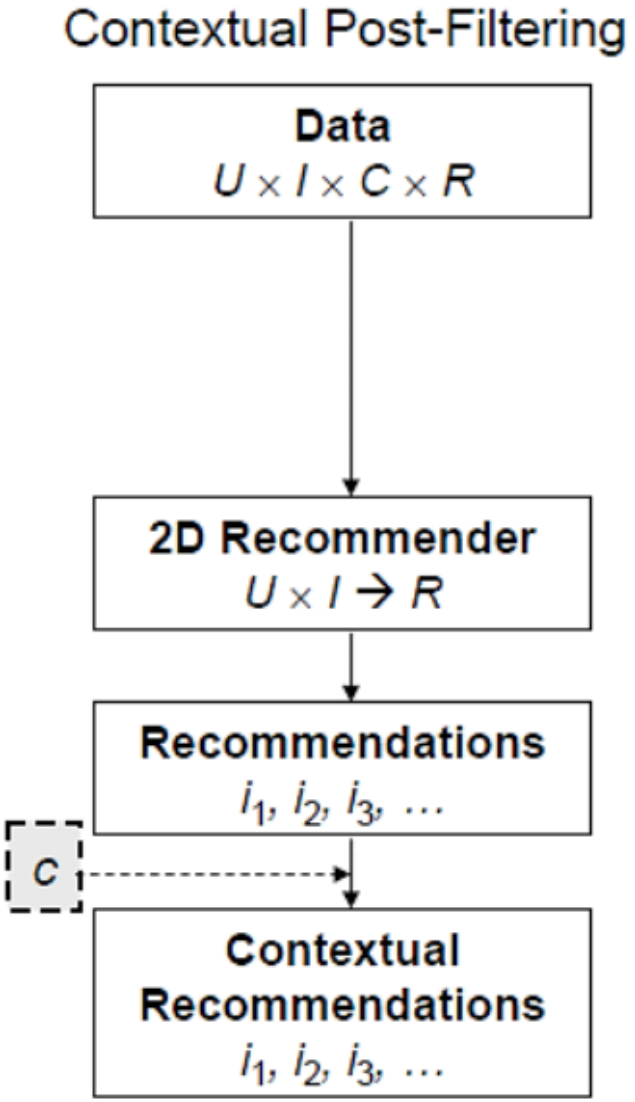
Context-aware Recommendations

- Context Over-Specification
  - Using an exact context may be too narrow:
    - Watching a movie with a girlfriend in a movie theater on Saturday
  - Certain aspects of the overly specific context may not be significant (e.g., Saturday vs. weekend)
  - Sparsity problem: overly specified context may not have enough training examples for accurate prediction
- Pre-Filter Generalization
  - Different Approaches
  - "Roll up" to higher level concepts in context hierarchies
    - E.g., Saturday → weekend, or movie theater → any location
  - Use latent factors models or dimensionality reduction approaches (Matrix factorization, LDA, etc.)

NETFLIX

Context-aware Recommendations

- Ignore context in the data selection and modeling phases, but filter or (re-rank) recommendations based on contextual information
- *Example: Context → Watching a movie with family*
  - Suppose the user generally watches comedies and dramas when going to theater with her family.
  - First, generate recommendations using standard recommender.
  - Then filter out action movies from the recommendation list.

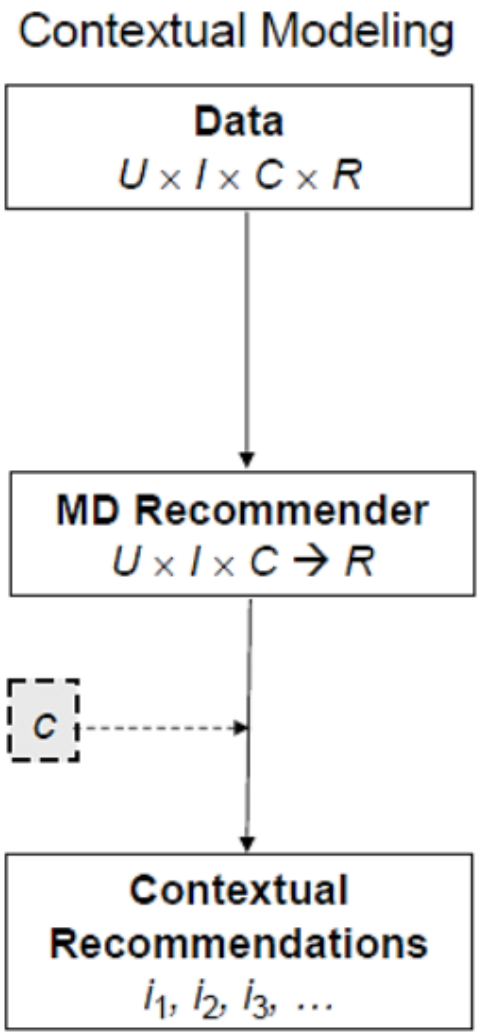


Context-aware Recommendations

- Contextual Post-Filtering is generally heuristic in nature
  - Basic Idea: Treat the context as an additional constraint
  - Many different approaches are possible
- Example: Filtering Based on Context Similarity
  - Can be represented as a set of features commonly associated with the specified context
  - Adjust the recommendation list by favoring those items that have more of the relevant features
  - Similarity-based approach (but the space of features may be different than the one describing the items)
- Example: Filtering Based on Social/Collaborative Context Representation
  - Mine social features (e.g., annotations, tags, tweets, reviews, etc.) associated with the item and users in a given context *C*
  - Promote items with frequently occurring social features from *C*

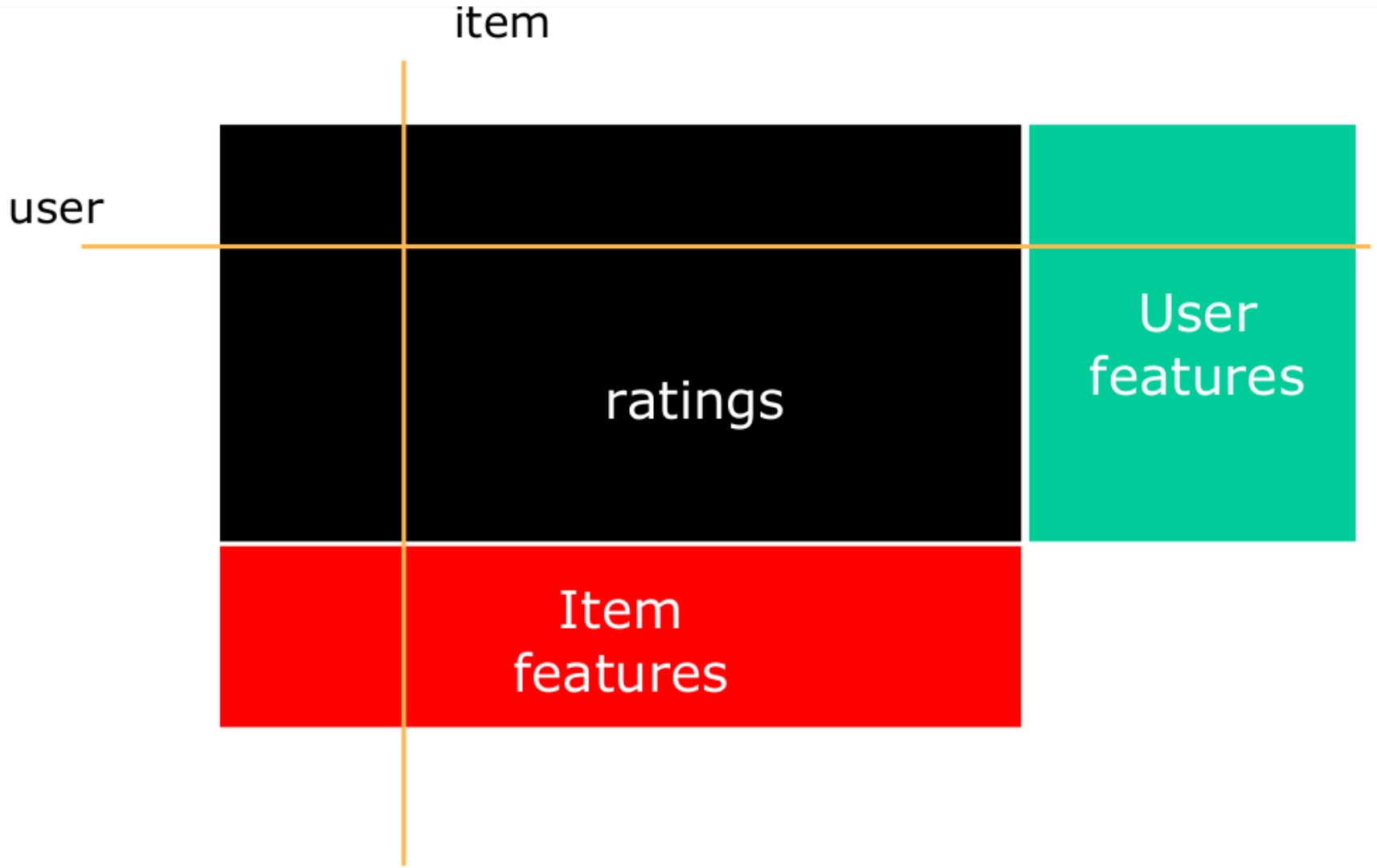
Context-aware Recommendations

- Using contextual information *directly* in the modeling learning phase
  - Multi-dimensional recommendation models
- Contextual variables are added as dimensions $D_1, \ldots, D_n$ in the feature space in addition to the *Users* and *Items* dimensions
  - R: $U \times I \times D_1 \times \ldots \times D_n \rightarrow Rating$
- Example: Dimensions for movie recommendation application
  - ***User***
  - ***Movie***
  - ***Time*** (weekday, weekend)
  - ***Company*** (alone, partner, family, etc.)
  - ***Place*** (movie theater, at home)

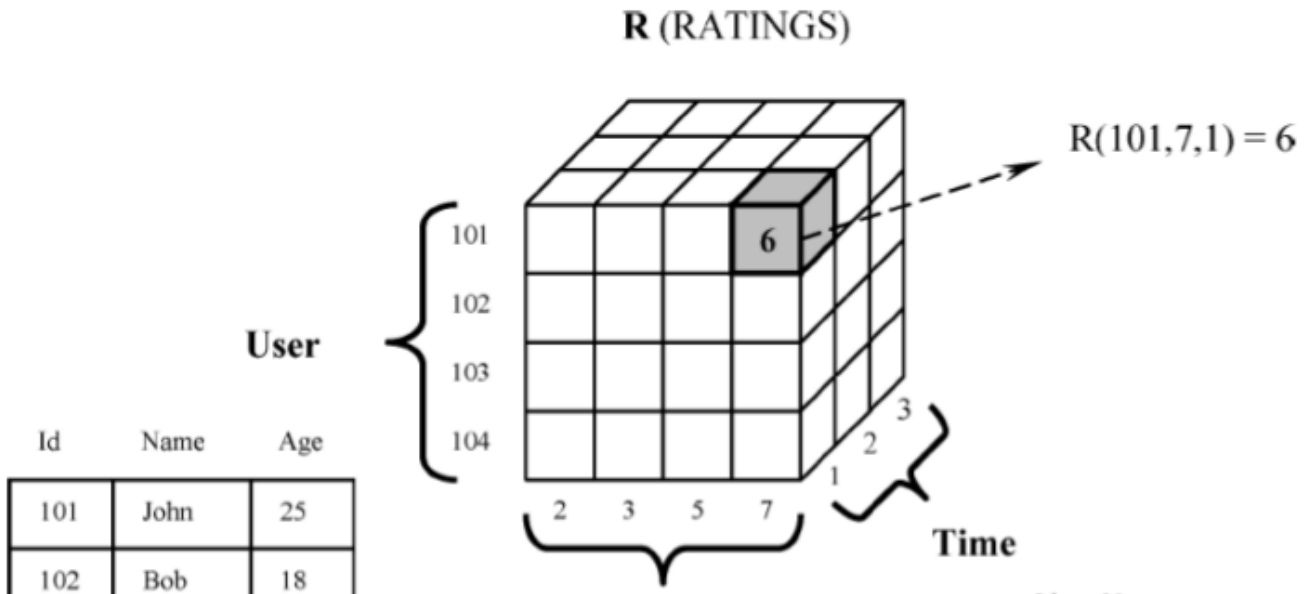


NETFLIX

Two dimensional model



N-dimensional model



Id	Name	Age
101	John	25
102	Bob	18
103	Alice	27
104	Mary	21

Id	Name	Cost
2	AB17	250.00
3	AB23	299.95
5	XY70	150.00
7	ZZ55	115.50

Id	Name
1	Weekday
2	Weekend
3	Holiday

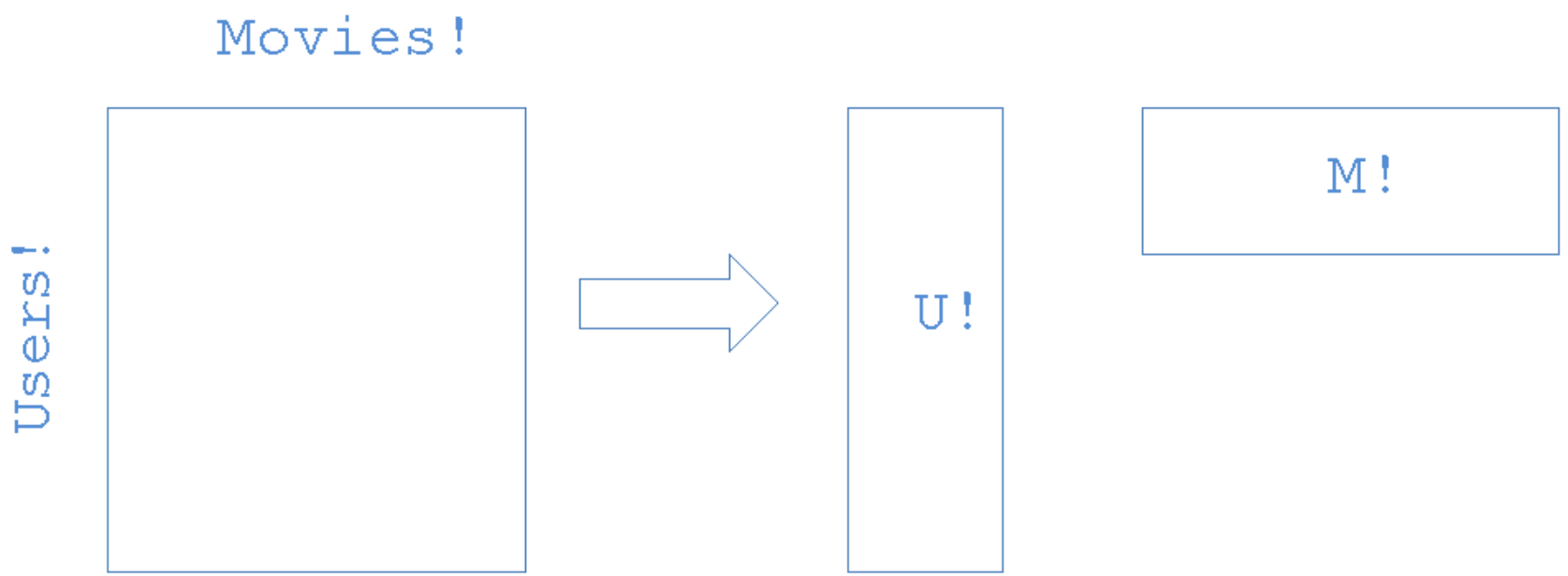
[Adomavicius et al., 2005]

Index

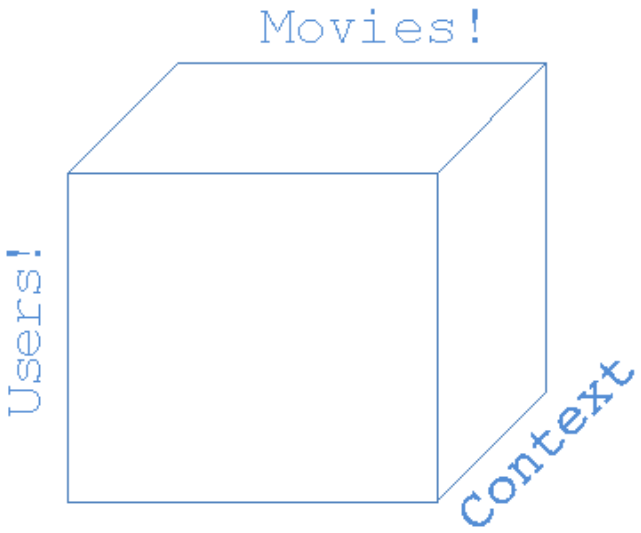
Tensor Factorization

Tensor Factorization

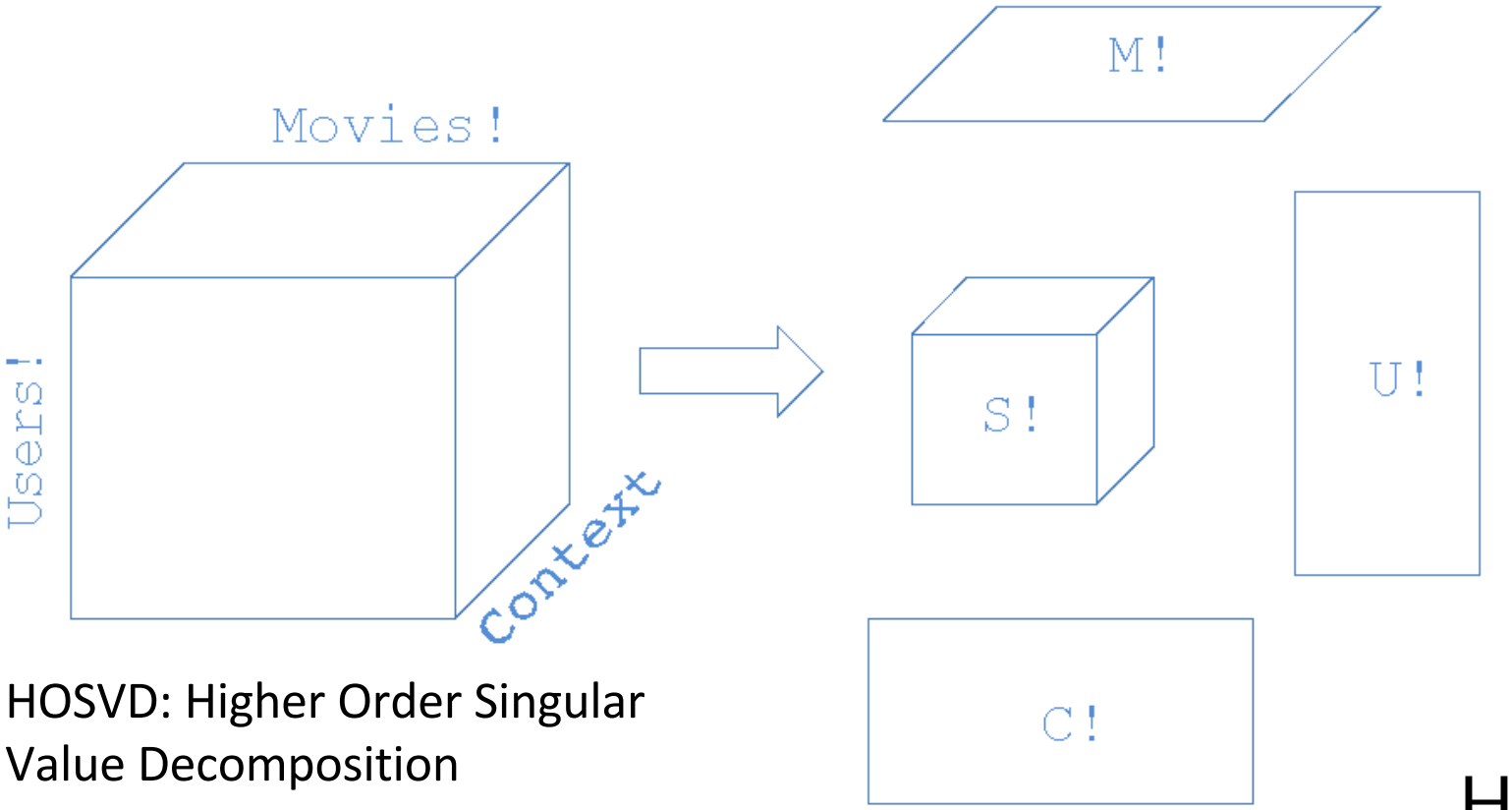
Find $U \in R^{n \to d}$ and $M \in R^{d \to m}$ so that $F = UM$



Tensor Factorization



Tensor Factorization



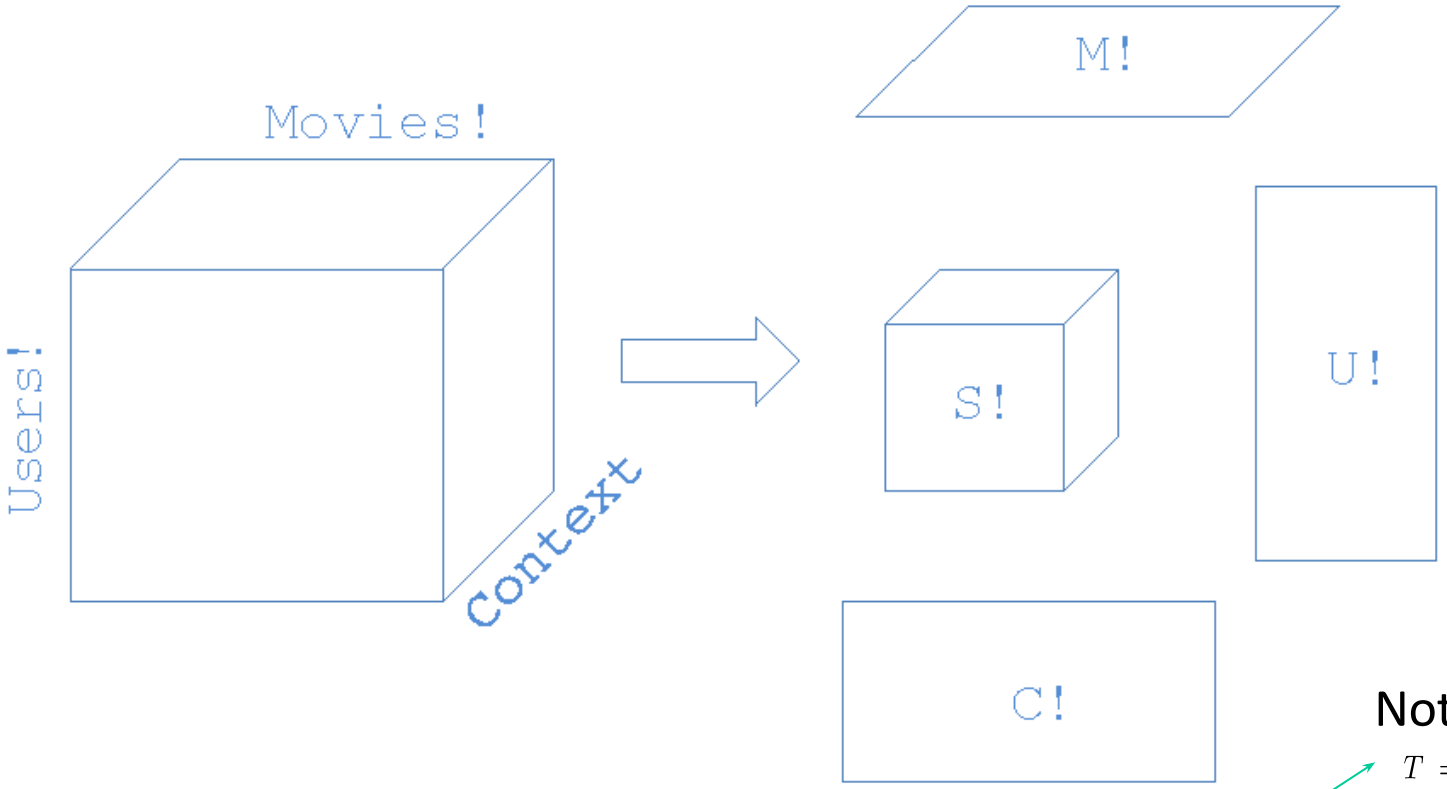
HOSVD: Higher Order Singular Value Decomposition

HOSVD Model

$$U \in \mathbb{R}^{n \times d_U}, M \in \mathbb{R}^{m \times d_M} \text{ and } C \in \mathbb{R}^{c \times d_C}$$

$$S \in \mathbb{R}^{d_U \times d_M \times d_C}$$

Tensor Factorization



Note:

$$T = Y \times_U U$$

$$T_{ljk} = \sum_{i=1}^{n} Y_{ijk} U_{ij}$$

$$F_{ijk} = S \rightarrow_U U_{i\leftarrow} \rightarrow_M M_{j\leftarrow} \rightarrow_C C_{k\leftarrow}$$

$$R[U, M, C, S] := L(F, Y) + \Omega[U, M, C] + \Omega[S]$$

Tensor Factorization

$$\Omega[F] = \lambda_M \|M\|_F^2 + \lambda_U \|U\|_F^2 + \lambda_C \|C\|_F^2$$

$$\Omega[S] := \lambda_S \|S\|_{\mathrm{F}}^2 \tag{1}$$

Tensor Factorization

Many implementations of MF used a simple squared error regression loss function

$$l(f, y) = \frac{1}{2}(f - y)^2$$

thus the loss over all users and items is:

$$L(F, Y) = \sum_{i}^{n} \sum_{j}^{m} l(f_{ij}, y_{ij})$$

Tensor Factorization

Alternatively one can use the absolute error loss function

$$l(f, y) = |f - y|$$

thus the loss over all users and items is:

$$L(F, Y) = \sum_{i}^{n} \sum_{j}^{m} l(f_{ij}, y_{ij})$$

Tensor Factorization

The partial gradients with respect to U, M, C and S can then be written as:

$$\partial_{U_{i*}} l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_M M_{j*} \times_C C_{k*}$$
$$\partial_{M_{j*}} l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_U U_{i*} \times_C C_{k*}$$
$$\partial_{C_{k*}} l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_U U_{i*} \times_M M_{j*}$$
$$\partial_S l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) U_{i*} \otimes M_{j*} \otimes C_{k*}$$

Tensor Factorization

The partial gradients with respect to U, M, C and S can then be written as:

$$\partial_{U_{i*}} l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_M M_{j*} \times_C C_{k*}$$
$$\partial_{M_{j*}} l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_U U_{i*} \times_C C_{k*}$$
$$\partial_{C_{k*}} l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_U U_{i*} \times_M M_{j*}$$
$$\partial_S l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) U_{i*} \otimes M_{j*} \otimes C_{k*}$$

We then iteratively update the parameter matrices and tensors using the following update rules:

Learning rate

$$U_{i*}^{t+1} = U_{i*}^{t} - \eta \partial_U L - \eta \lambda_U U_{i*}$$
$$M_{j*}^{t+1} = M_{j*}^{t} - \eta \partial_M L - \eta \lambda_M M_{j*}$$
$$C_{k*}^{t+1} = C_{k*}^{t} - \eta \partial_C L - \eta \lambda_C C_{k*}$$
$$S^{t+1} = S^{t} - \eta \partial_S l(F_{ijk}, Y_{ijk}) - \eta \lambda_S S$$

NETFLIX

Tensor Factorization

We evaluate our model on contextual rating data and computing the *Mean Absolute Error (MAE)*,using 5-fold cross validation defined as follows:

$$MAE = \frac{1}{K} \sum_{ijk}^{n,m,c} D_{ijk} | Y_{ijk} - F_{ijk} |$$

Data set	Users	Movies	Context Dim.	Ratings	Scale
Yahoo!	7642	11915	2	221K	1-5
Adom.	84	192	5	1464	1-13
Food	212	20	2	6360	1-5

Table: Data set statistics

Tensor Factorization

- Pre-filtering based approach, (*G. Adomavicius et.al*), computes recommendations using *only* the ratings made in the same context as the target one
- Item splitting method (*L. Baltrunas, F. Ricci*) which identifies items which have significant differences in their rating under different context situations.

Tensor Factorization

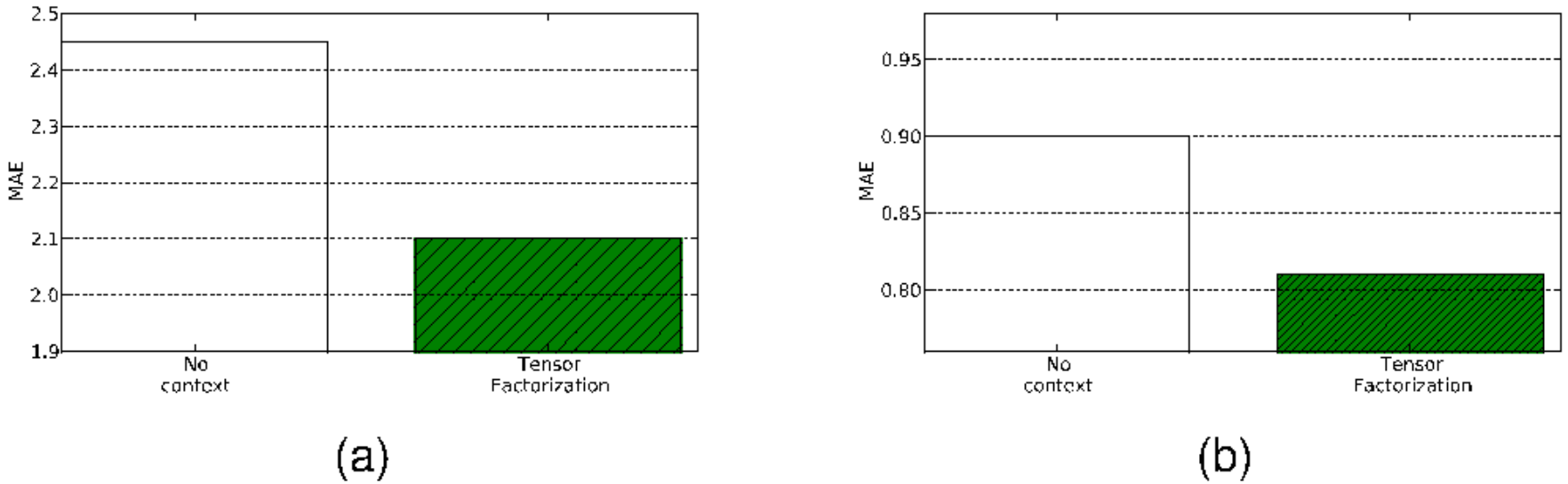


Figure: Comparison of matrix (no context) and tensor (context) factorization on the Adom and Food data.

Tensor Factorization

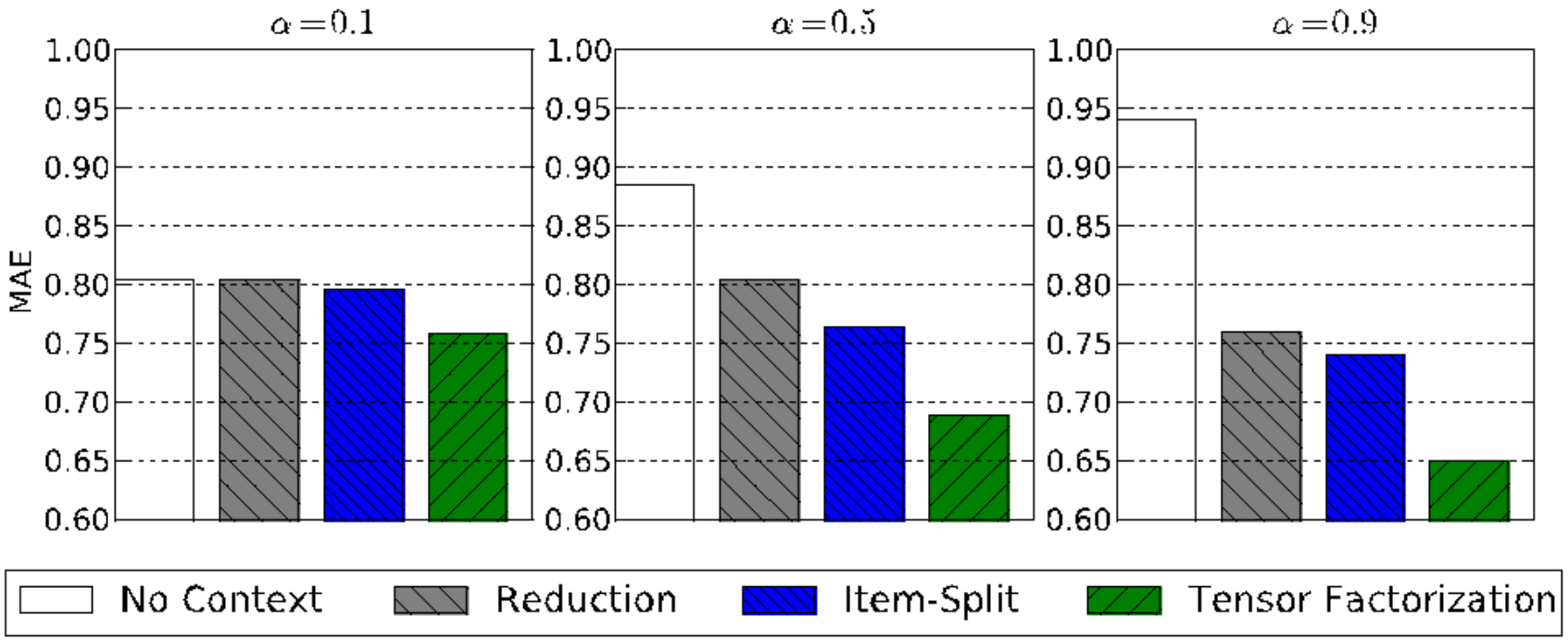
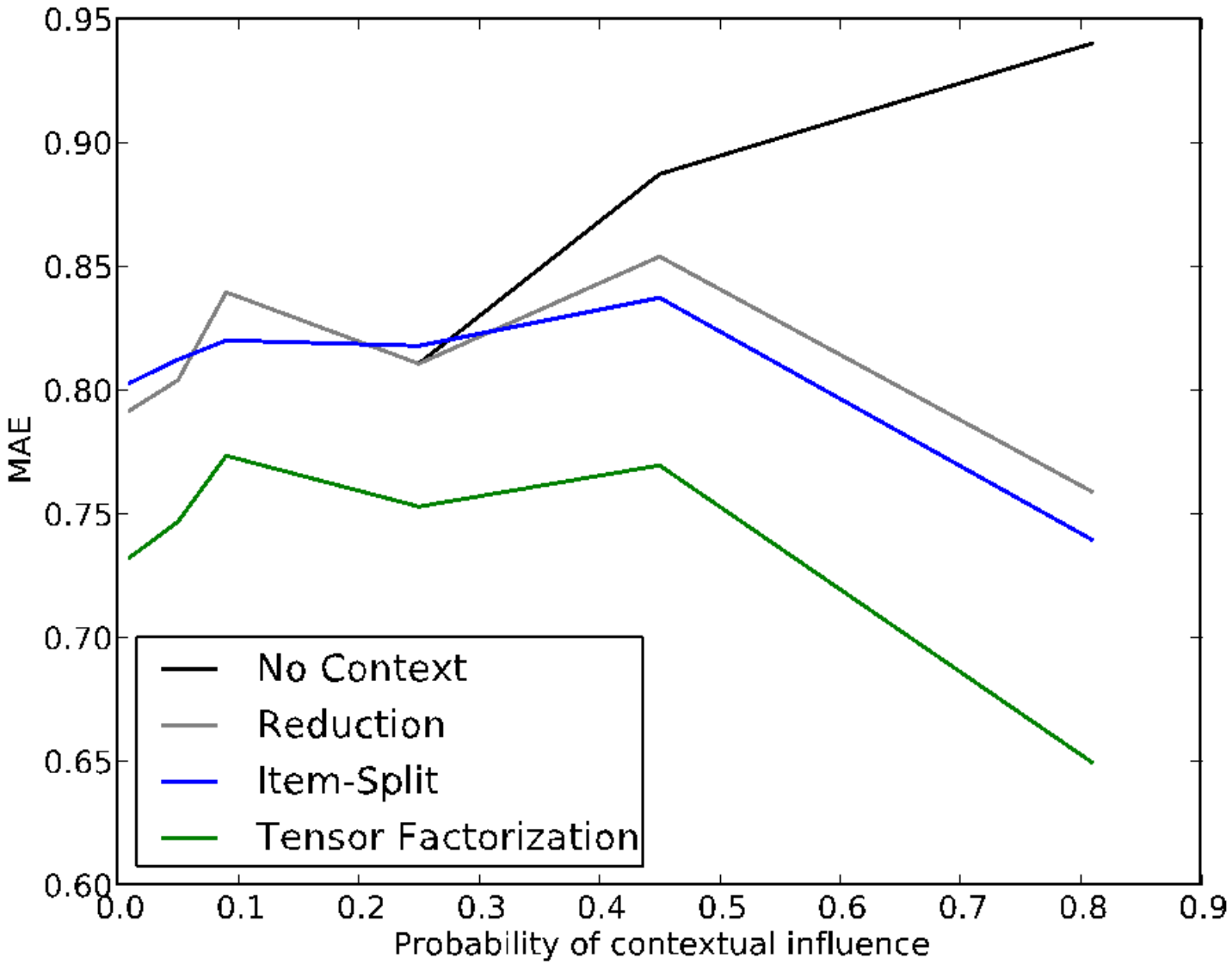


Figure: Comparison of context-aware methods on the Yahoo! artificial data

Tensor Factorization



Tensor Factorization

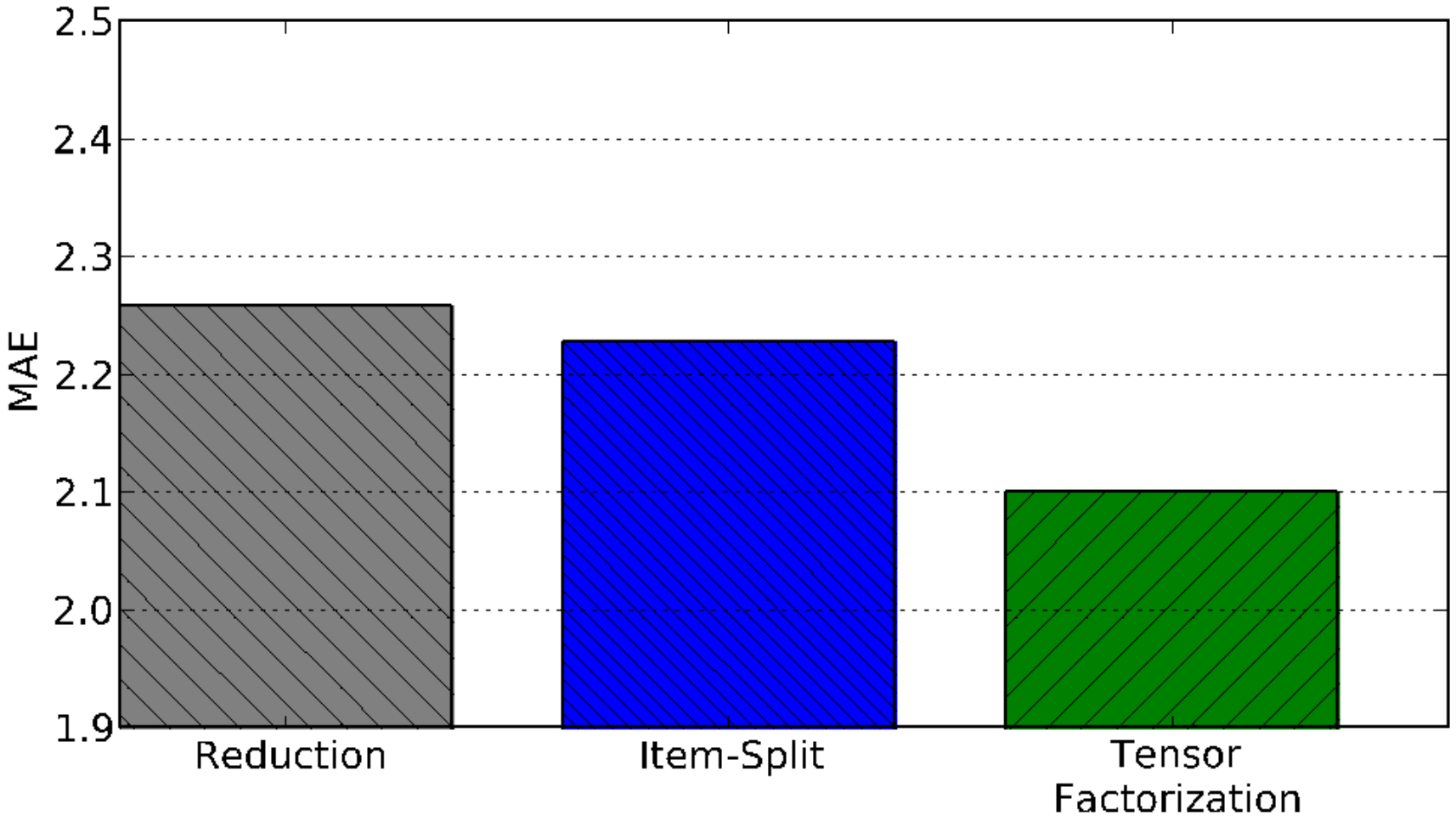


Figure: Comparison of context-aware methods on the Adom data.

Tensor Factorization

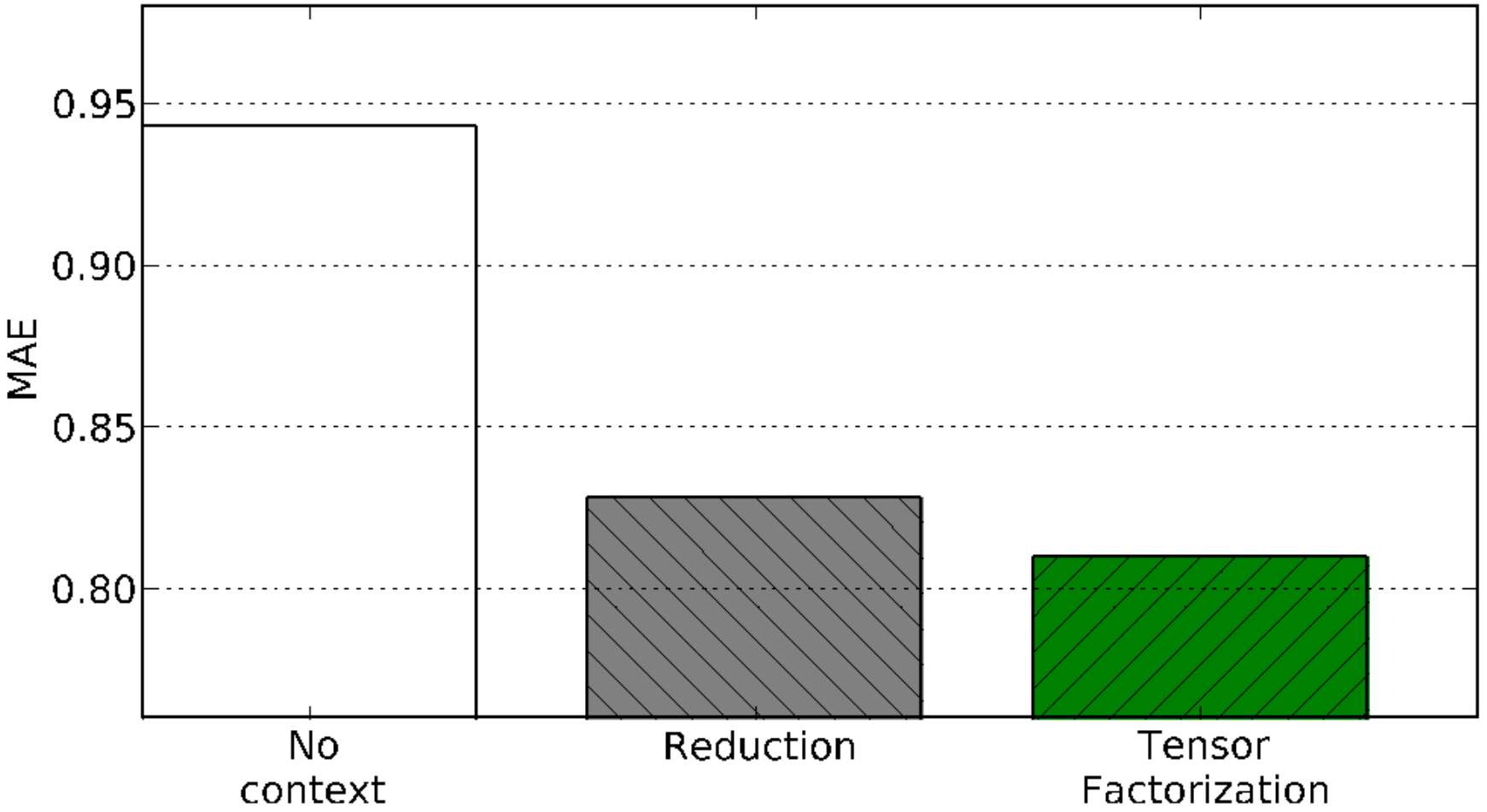


Figure: Comparison of context-aware methods on the Food data.

Tensor Factorization

- Context does matter
- TF offers a way to integrate as many contextual variables as needed
- Can be “easily” trained in a similar way to MF
- However…
  - Factorization Machines seem to work better, especially for higher dimensional spaces.

Index

Factorization Machines

Factorization Machines (Rendle, 2010)

- Generalization of regularized matrix (and tensor) factorization approaches combined with linear (or logistic) regression
- Problem: Each new adaptation of matrix or tensor factorization requires deriving new learning algorithms
  - Hard to adapt to new domains and add data sources
  - Hard to advance the learning algorithms across approaches
  - Hard to incorporate non-categorical variables

Factorization Machines (Rendle, 2010)

- Approach: Treat input as a real-valued feature vector
  - Model both linear and pair-wise interaction of *k* features (i.e. polynomial regression)
  - Traditional machine learning will over-fit
  - Factor pairwise interactions between features
  - Reduced dimensionality of interactions promote generalization
  - Different matrix factorizations become different feature representations
  - Tensors: Additional higher-order interactions
- Combines "generality of machine learning/regression with quality of factorization models"

Factorization Machines (Rendle, 2010)

- Each feature gets a weight value and a factor vector
  - O(*dk*) parameters

$$b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d, \mathbf{V} \in \mathbb{R}^{d \times k}$$

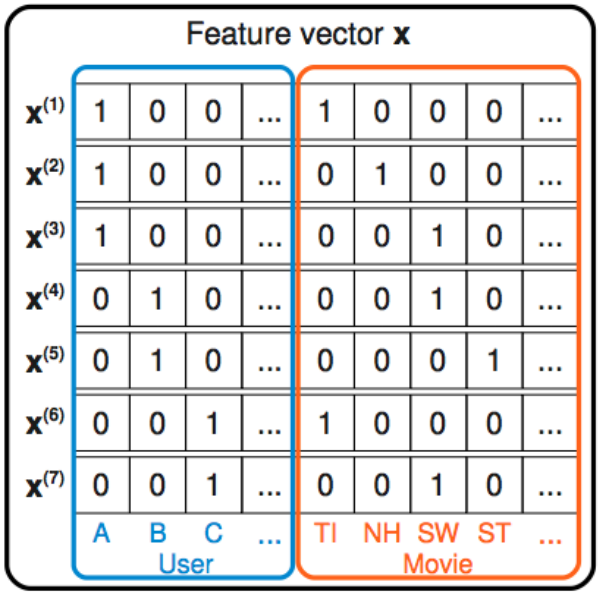
- Model equation:

$$f(\mathbf{x}) = b + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} x_i x_j \mathbf{v}_i^T \mathbf{v}_j \qquad O(d^2)$$

$$= b + \sum_{i=1}^{d} w_i x_i + \frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{d} x_i v_{i,f} \right)^2 - \sum_{i=1}^{d} x_i^2 v_{i,f}^2 \right) \qquad O(kd)$$

Factorization Machines (Rendle, 2010)

- Two categorical variables (*u*, *i*) encoded as real values:



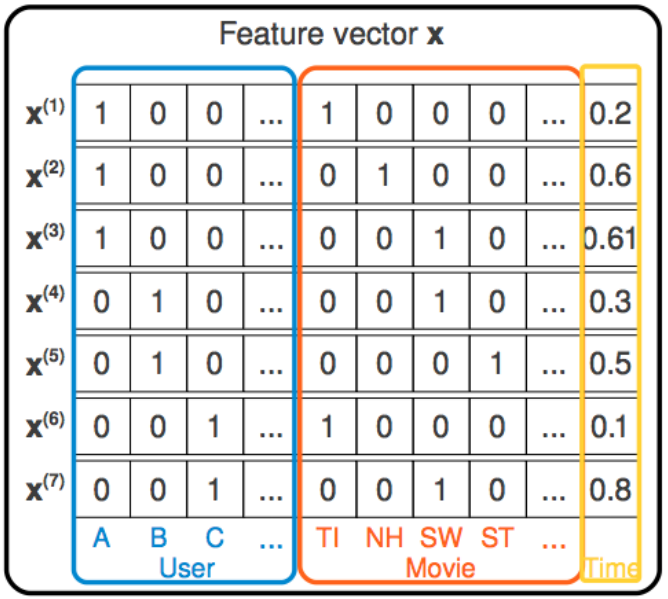
- FM becomes identical to MF with biases:

$$f(\mathbf{x}) = b + w_u + w_i + \mathbf{v}_u^T \mathbf{v}_i$$

From Rendle (2012) KDD Tutorial

Factorization Machines (Rendle, 2010)

- Makes it easy to add a time signal



- Equivalent equation:

$$f(\mathbf{x}) = b + w_u + w_i + x_t w_t + \mathbf{v}_u^T \mathbf{v}_i + x_t \mathbf{v}_u^T \mathbf{v}_t + x_t \mathbf{v}_i^T \mathbf{v}_t$$

From Rendle (2012) KDD Tutorial

Factorization Machines (Rendle, 2010)

- L2 regularized
  - Regression: Optimize RMSE
  - Classification: Optimize logistic log-likelihood
  - Ranking: Optimize scores
- Can be trained using:
  - SGD
  - Adaptive SGD
  - ALS
  - MCMC
  - Efficient training

Gradient:

$$\frac{\partial}{\partial\theta} f(\mathbf{x}) = \begin{cases} 1 & \text{if } \theta \text{ is } b \\ x_i & \text{if } \theta \text{ is } w_i \\ x_i \sum_{j=1}^{d} v_{j,f} x_j - v_{i,f} x_i^2 & \text{if } \theta \text{ is } v_{i,f} \end{cases}$$

Least squares SGD:

$$\theta' = \theta - \eta \left((f(\mathbf{x}) - y) \frac{\partial}{\partial\theta} f(\mathbf{x}) + \lambda_\theta \theta \right)$$

Factorization Machines (Rendle, 2010)

- Learning parameters:
  - Number of factors
  - Iterations
  - Initialization scale
  - Regularization (SGD, ALS) – Multiple
  - Step size (SGD, A-SGD)
  - MCMC removes the need to set those hyperparameters

Index

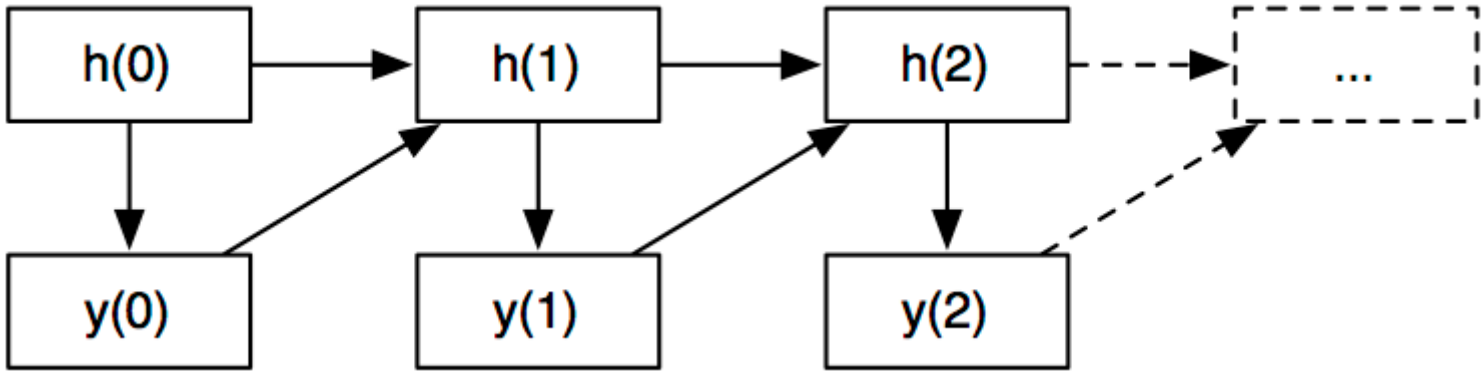
NETFLIX

3.3 Deep Learning for Recommendation

NETFLIX

Deep Learning for Collaborative Filtering

- Let's look at how Spotify uses Recurrent Networks for Playlist Prediction (http://erikbern.com/?p=589)



Deep Learning for Collaborative Filtering

- We assume $P(y_i|h_i)$ is a normal distribution, log-likelihood of the loss is just the (negative) L2 loss: $-(y_t - h_t)^2$
- We can specify that $h_{i+1} = \tanh(U y_i + V h_i)$ and that $h_0 = 0$
  - Model is now completely specified and we have $3k^2$ unknown parameters
  - Find U, V, and W to maximize log likelihood over all examples using backpropagation

$$\log L = \sum_{\text{all examples}} \left(\sum_{i=0}^{t-1} -(y_i - h_i)^2 \right)$$

Deep Learning for Collaborative Filtering

- In order to predict the next track or movie a user is going to watch, we need to define a distribution $P(y_i|h_i)$
  - If we choose Softmax as it is common practice, we get:

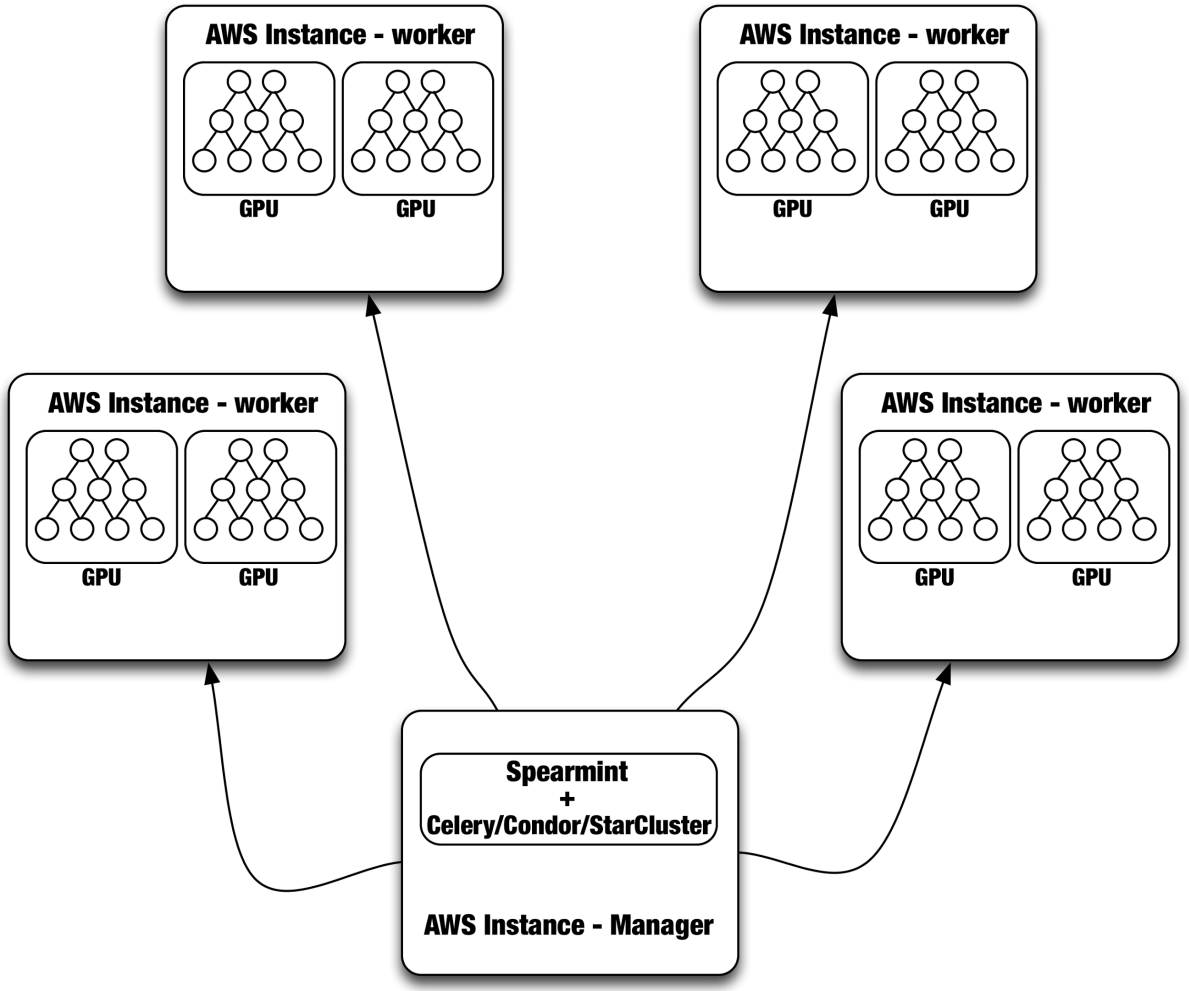
$$P(y_i|h_i) = \frac{\exp(h_t^T a_j)}{\sum_k \exp(h_t^T a_k)}$$

    - Problem: denominator (over all examples is very expensive to compute)
    - Solution: build a tree that implements a hierarchical softmax

- More details on the blogpost

- How did we implement our ANN solution at Netflix?

http://techblog.netflix.com/2014/02/distributed-neural-networks-with-gpus.html



ANN Training over GPUS and AWS

- Level 1 distribution: machines over different AWS regions
- Level 2 distribution: machines in AWS and same AWS region
  - Use coordination tools
    - Spearmint or similar for parameter optimization
    - Condor, StarCluster, Mesos… for distributed cluster coordination
- Level 3 parallelization: highly optimized parallel CUDA code on GPUs

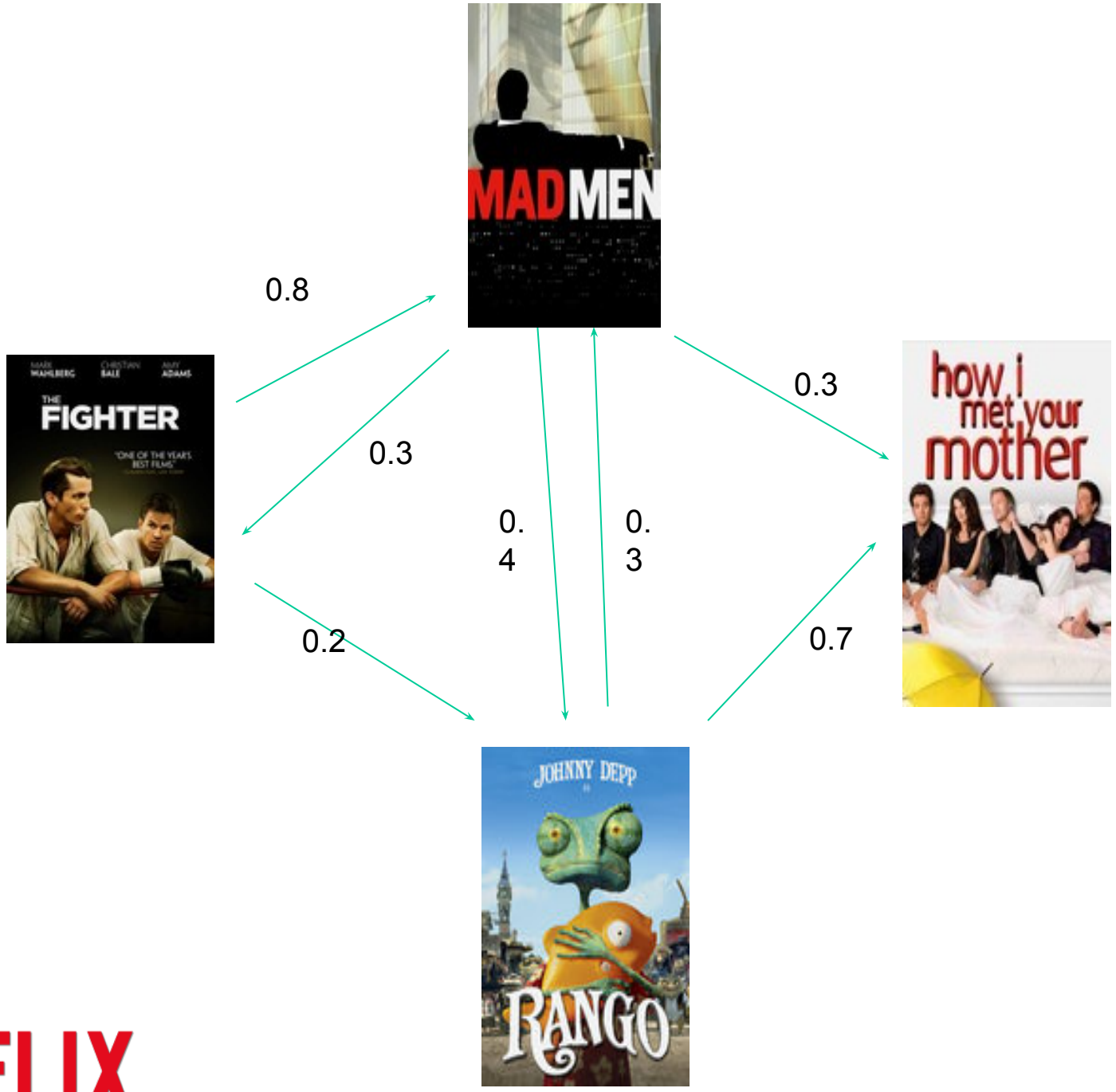
Index

3.4 Similarity as Recommendation

What is similarity?

- Similarity can refer to different dimensions
  - Similar in metadata/tags
  - Similar in user play behavior
  - Similar in user rating behavior
  - …
- You can learn a model for each of them and finally learn an ensemble

Graph-based similarities



NETFLIX

Example of graph-based similarity: SimRank

- SimRank (Jeh & Widom, 02): “two objects are similar if they are referenced by similar objects.”

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b))$$

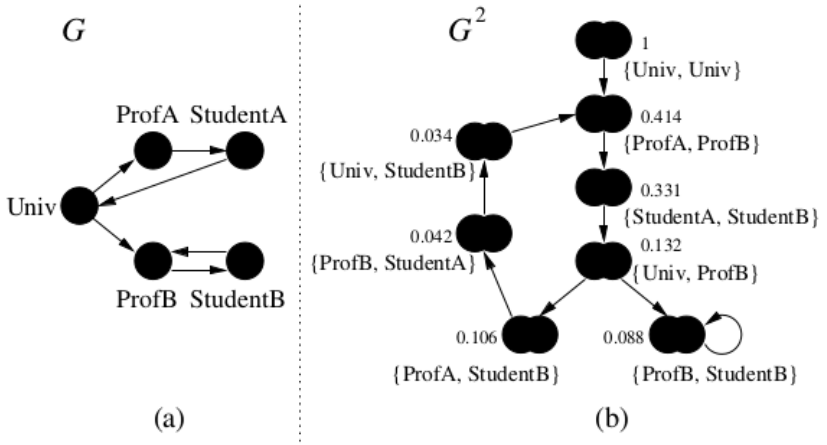


Figure 1: A small Web graph G and simplified node-pairs graph G^2. SimRank scores using parameter $C = 0.8$ are shown for nodes in G^2.

Similarity ensembles

- Come up with a score of play similarity, rating similarity, tag-based similarity…
- Combine them using an ensemble
  - Weights are learned using regression over existing response
  - Or… some MAB explore/exploit approach
- The final concept of “similarity” responds to what users vote as similar

Index

3.5 Social Recommendations

Social and Trust-based recommenders

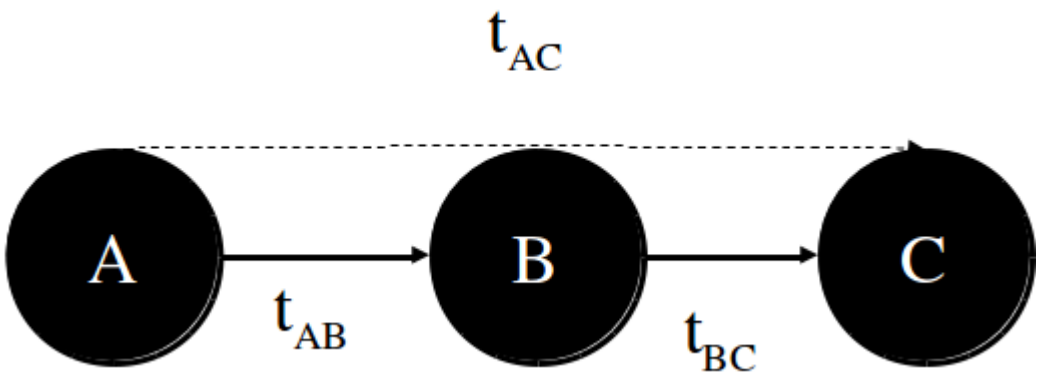
- A social recommender system recommends items that are “popular” in the social proximity of the user.
- A person being close in our social network does not mean we **trust** their judgement
- This idea of trust is central in social-based systems
- It can be a general per-user value that takes into account social proximity but can also be topic-specific

Defining Trust

- Trust is very complex
  - Involves personal background, history of interaction, context, similarity, reputation, etc.
- Sociological definitions
  - Trust requires a belief and a commitment
  - E.g. Bob believes Frank will provide reliable information thus Bob is willing to act on that information
  - Similar to a bet
- In the context of recommender systems, trust is generally used to describe similarity in opinion
  - Ignores authority, correctness on facts

Trust Inference

The Goal: Select two individuals - the *source* (node A) and *sink* (node C) - and recommend to the source how much to trust the sink.



NETFLIX

Major Algorithms - Networks

- Advogato (Levien)
- Appleseed (Ziegler and Lausen)
- MoleTrust (Massa and Avesani)
- TidalTrust (Golbeck)

Building Recommender Systems Using Trust

- Use trust as a way to give more weight to some users
- Trust for collaborative filtering
  - Use trust in place of (or combined with) similarity
- Trust for sorting and 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 collaborative filtering methods in different ways
  - e.g. replace or modify user-user “similarity” by using social network information

Demographic Methods

- Aim to categorize the user based on personal attributes and make recommendation based on demographic classes
- Demographic groups can come from marketing research – hence experts decided how to model the users
- Demographic techniques form people-to-people correlations

Demographic Methods

- Demographic features in general are asked
- But can also induced by classifying a user using other user descriptions (e.g. the home page) – you need some user for which you know the class (e.g. male/female)
- Prediction can use whatever learning mechanism we like (nearest neighbor, naïve classifier, etc.)

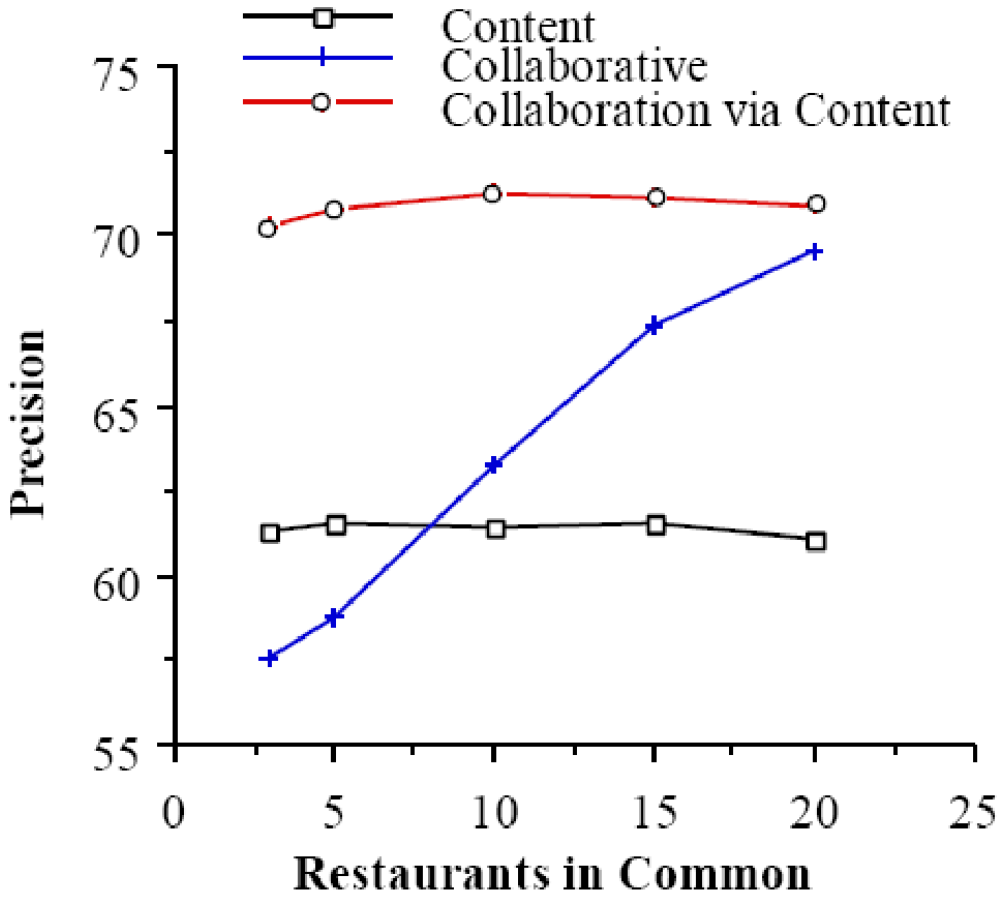
	gender	age	area code	education	employed	Dolce
Karen	F	15	714	HS	F	+
Lynn	F	17	714	HS	F	–
Chris	M	35	714	C	T	+
Mike	F	40	714	C	T	–
Jill	F	10	714	E	F	?

Index

4 Hybrid Approaches

Comparison of methods (FAB system)

- Content–based recommendation with Bayesian classifier
- Collaborative is standard using Pearson correlation
- Collaboration via content uses the content-based user profiles



Averaged on 44 users

Precision computed in top 3 recommendations

Hybridization Methods

Hybridization Method	Description
Weighted	Outputs from several techniques (in the form of scores or votes) 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

- 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
- Assumption: relative value of the different techniques is more or less uniform across the space of possible items
  - Not true in general: e.g. a collaborative recommender will be weaker for those items with a small number of raters.

Switching

- The system uses criterion to switch between techniques
  - **Example**: The DailyLearner system uses a content-collaborative hybrid in which a content-based recommendation method is employed first
  - If the content-based system cannot make a recommendation with sufficient confidence, then a collaborative recommendation is attempted
  - Note that switching does not completely avoid the cold-start problem, since both the collaborative and the content-based systems have the “new user” problem
- The main problem of this technique is to identify a GOOD switching condition.

Mixed

- Recommendations from more than one technique are presented together
- The mixed hybrid avoids the “new item” start-up problem
- It does not get around the “new user” start-up problem, since both the content and collaborative methods need some data about user preferences to start up.

Feature Combination

- Features can be combined in several directions. E.g.
  - (1) Treat collaborative information (ratings of users) as additional feature data associated with each example and use content-based techniques over this augmented data set
  - (2) Treat content features as different dimensions for the collaborative setting (i.e. as other ratings from virtual specialized users)

Cascade

- One recommendation technique is employed first to produce a coarse ranking of candidates and a second technique refines the recommendation
  - Example: EntreeC 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 is employed to break ties
- Cascading allows the system to avoid employing the second, lower-priority, technique on items that are already well-differentiated by the first
- But requires a meaningful and constant ordering of the techniques.

Feature Augmentation

- Produce a rating or classification of an item and that information is then incorporated into the processing of the next recommendation technique
  - Example: Libra system makes content-based recommendations of books based on data found in Amazon.com, using a naive Bayes text classifier
  - In the text data used by the system is included “related authors” and “related titles” information that Amazon generates using its internal collaborative systems
- Very similar to the feature combination method:
  - **Here** the output is used for a second RS
  - In **feature combination** the representations used by two systems are combined.

Index

NETFLIX

5. Netflix as a practical example

NETFLIX

Netflix Prize

COMPLETED

What we were interested in:

- High quality *recommendations*

Proxy question:

- Accuracy in predicted rating
- Improve by 10% = $1million!

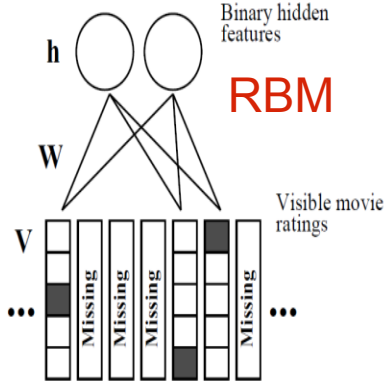
$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Results

- Top 2 algorithms still in production

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} | & | \\ u_1 & u_2 \\ | & | \end{bmatrix} \times \begin{bmatrix} \lambda_1 & \varnothing \\ \varnothing & \lambda_2 \end{bmatrix} \times \begin{bmatrix} — & v_1 & — \\ — & v_2 & — \end{bmatrix}$$

SVD



NETFLIX

What about the final prize ensembles?

- Our offline studies showed they were too computationally intensive to scale
- Expected improvement not worth the engineering effort
- Plus…. Focus had already shifted to other issues that had more impact than rating prediction.

From the Netflix Prize to today

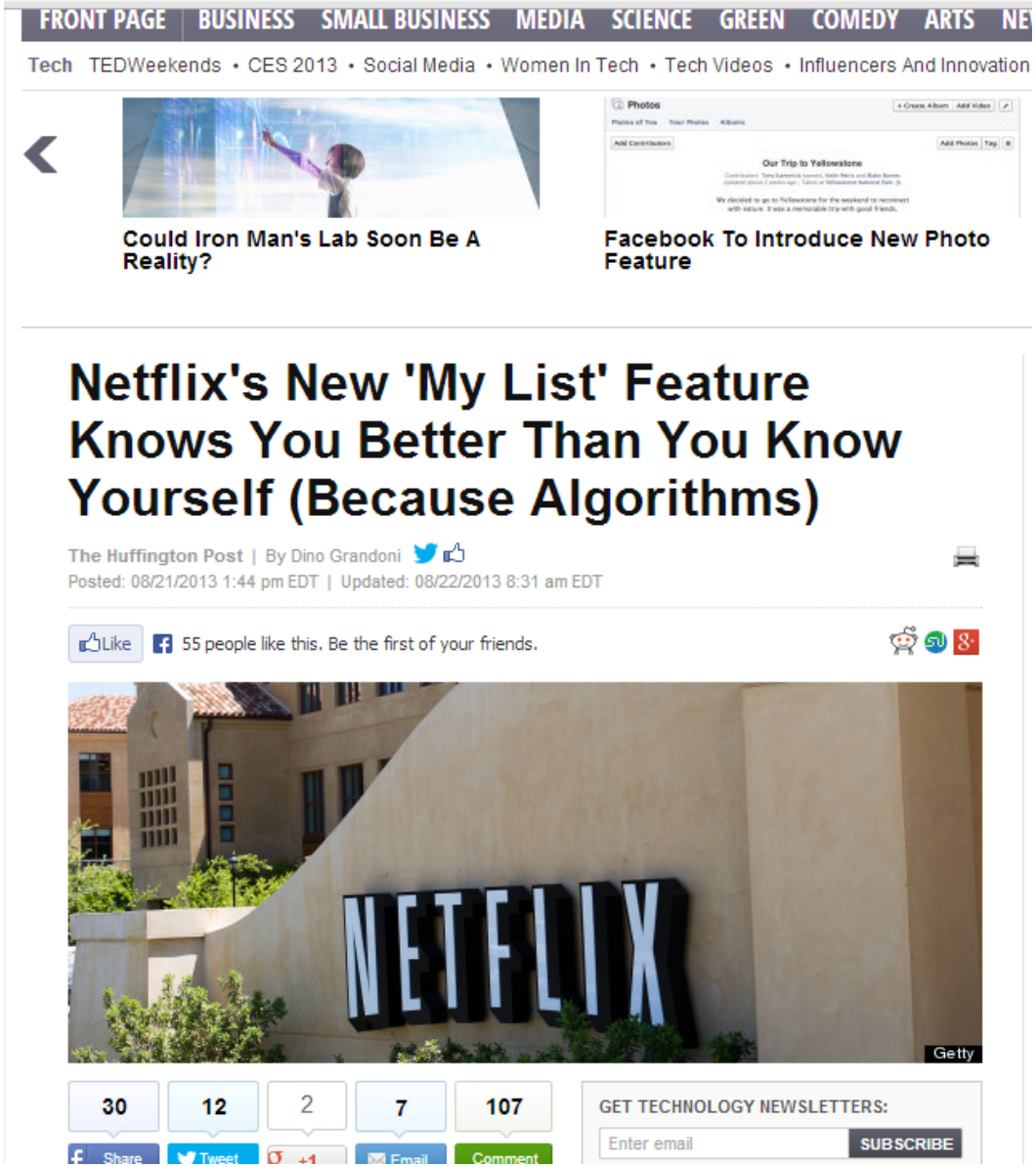
2006

2014

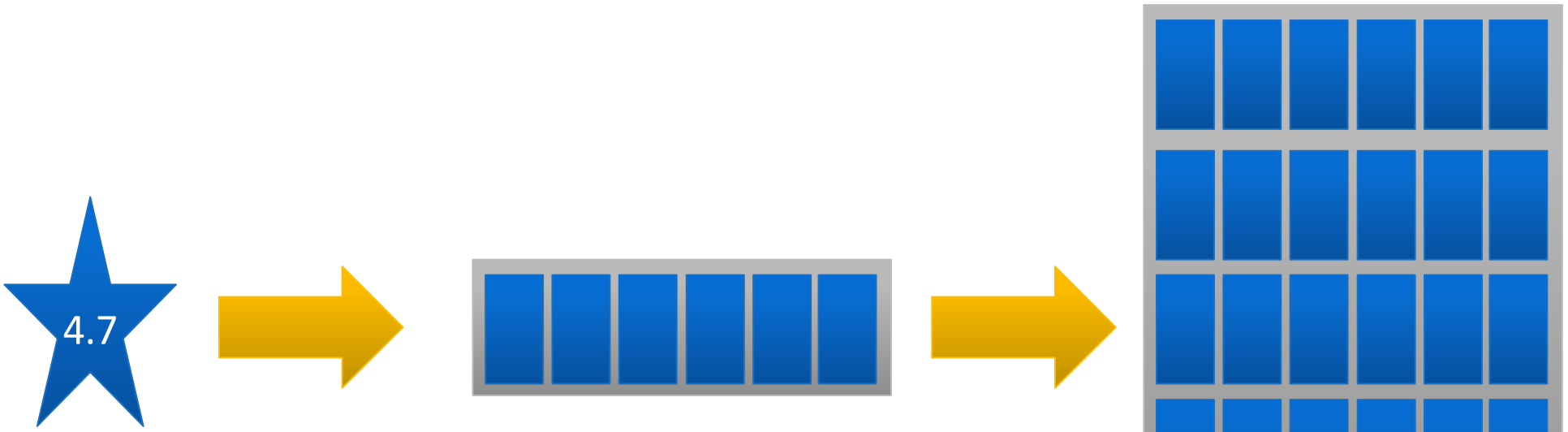
Anatomy of Netflix Personalization

Everything is a Recommendation

NETFLIX


FRONT PAGE BUSINESS SMALL BUSINESS MEDIA SCIENCE GREEN COMEDY ARTS NE
Tech TEDWeekends • CES 2013 • Social Media • Women In Tech • Tech Videos • Influencers And Innovation
Could Iron Man's Lab Soon Be A Reality?
Facebook To Introduce New Photo Feature
Netflix's New 'My List' Feature Knows You Better Than You Know Yourself (Because Algorithms)
The Huffington Post | By Dino Grandoni
Posted: 08/21/2013 1:44 pm EDT | Updated: 08/22/2013 8:31 am EDT
Like
55 people like this. Be the first of your friends.
NETFLIX
Getty
30
12
2
7
107
Share
Tweet
+1
Email
Comment
GET TECHNOLOGY NEWSLETTERS:
Enter email
SUBSCRIBE

Evolution of Recommendation Approach



Rating

Ranking

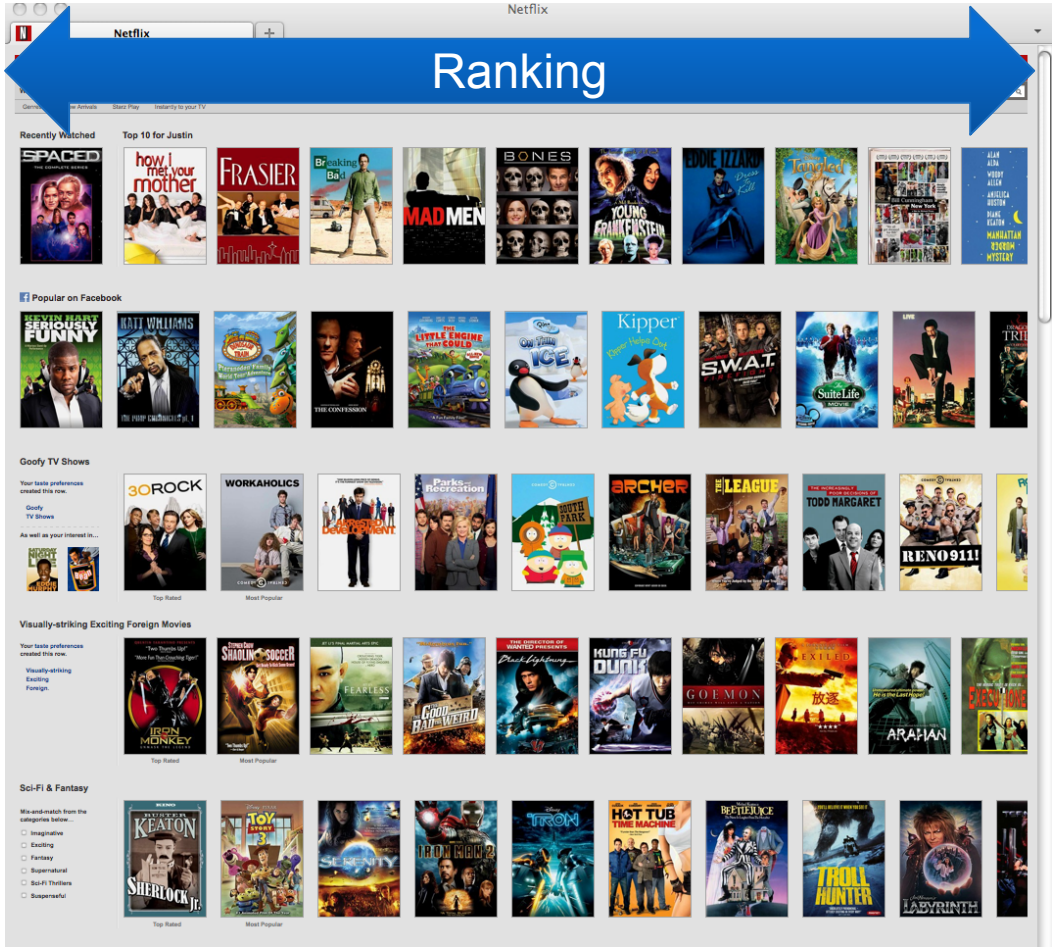
Page Generation

NETFLIX

Everything is personalized

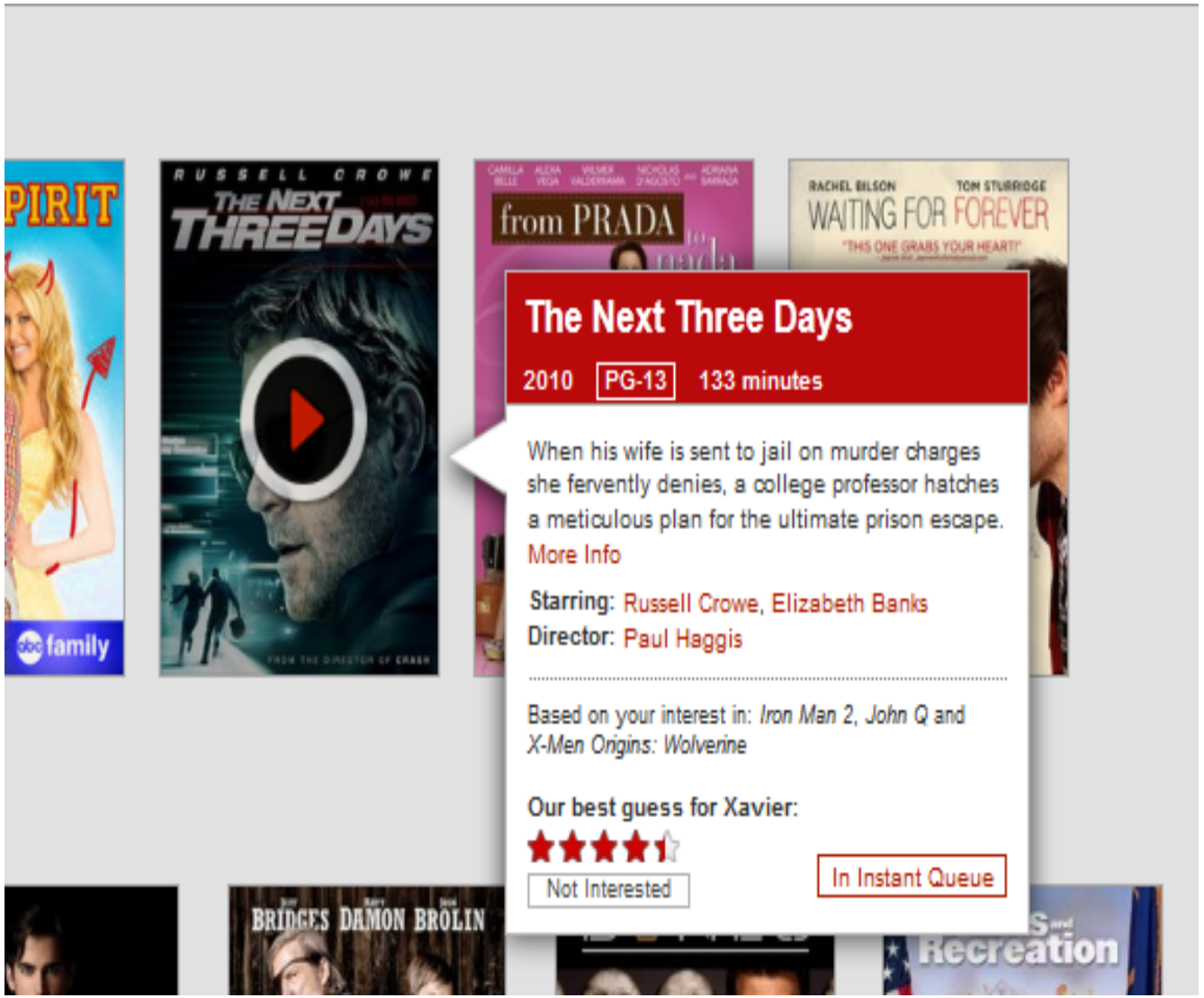
Ranking

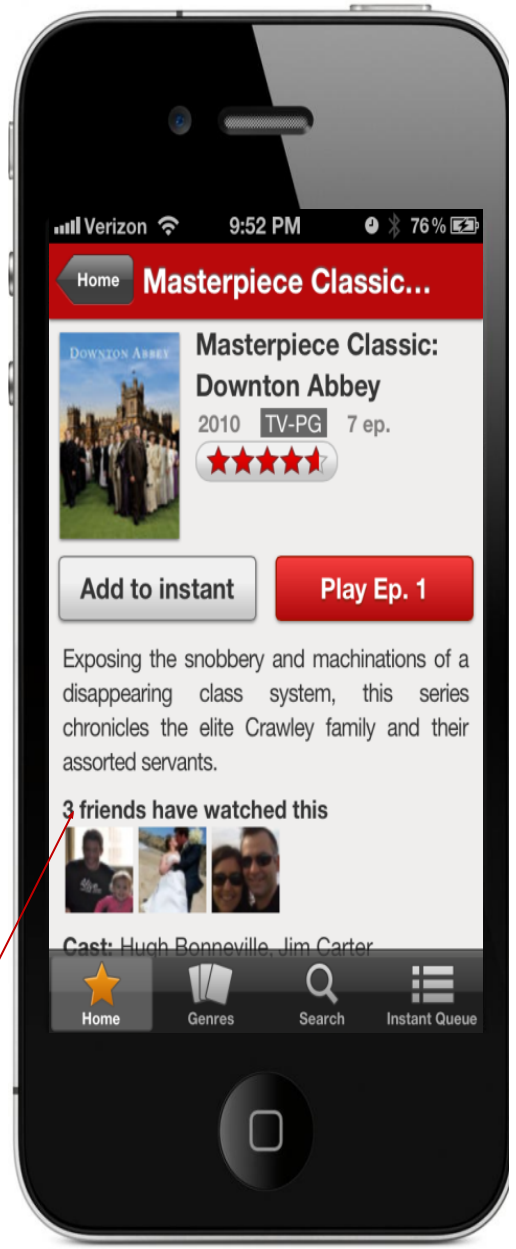
Key algorithm, sorts titles in most contexts



NETFLIX

Support for Recommendations

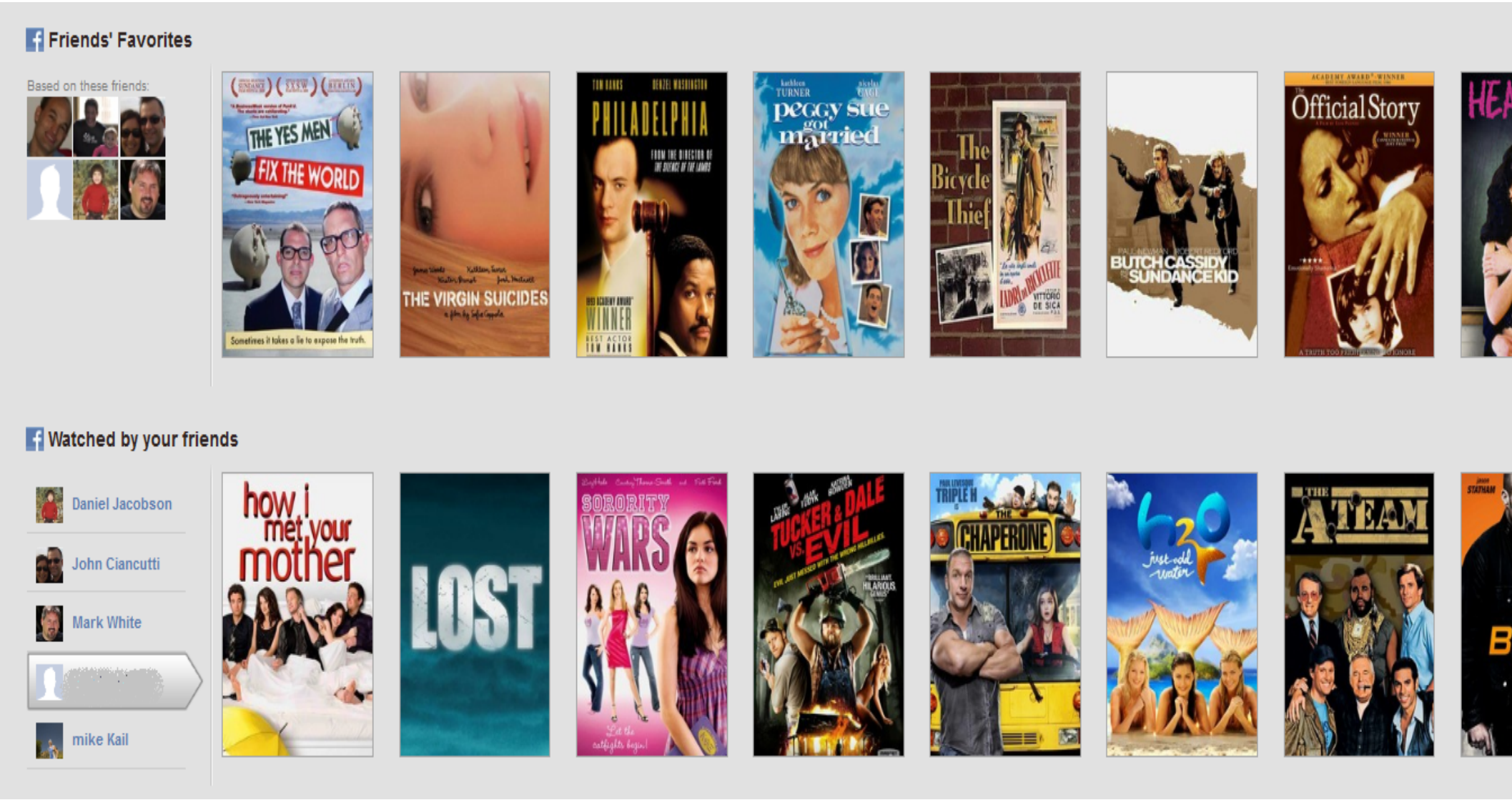




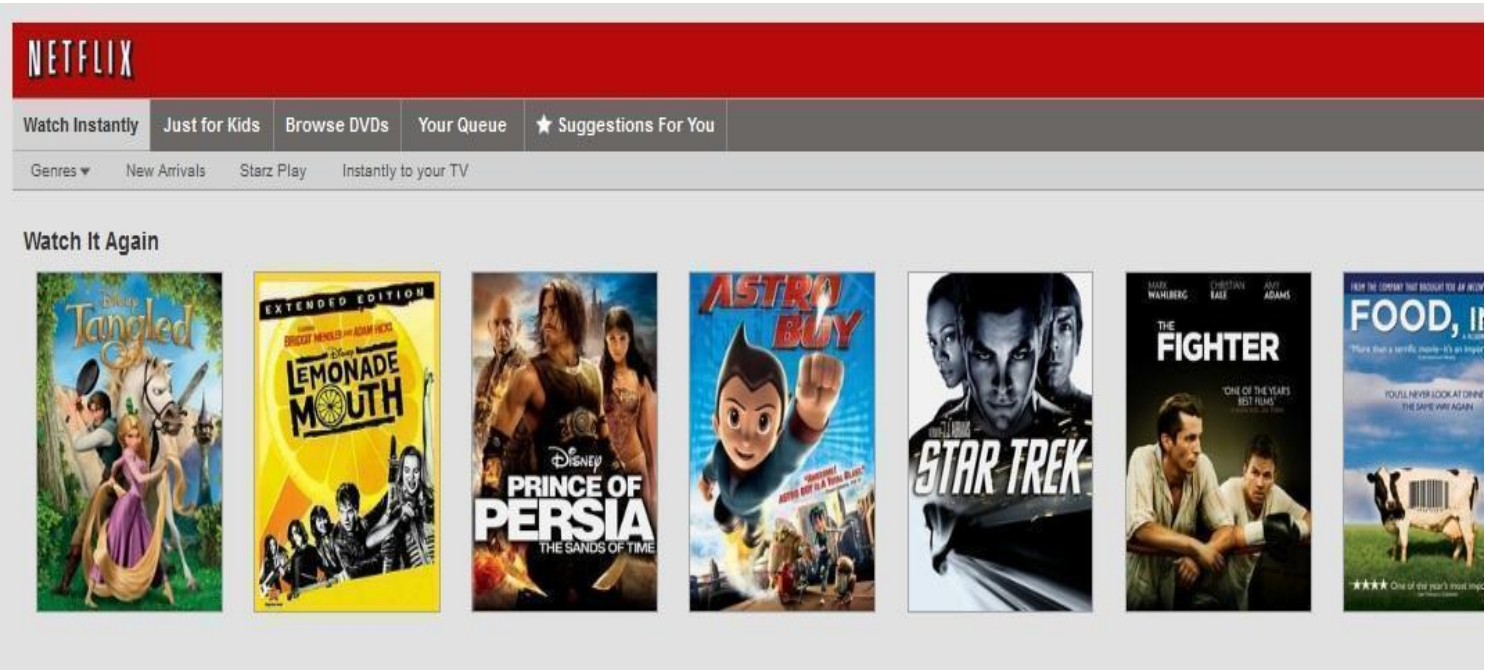
Social Support

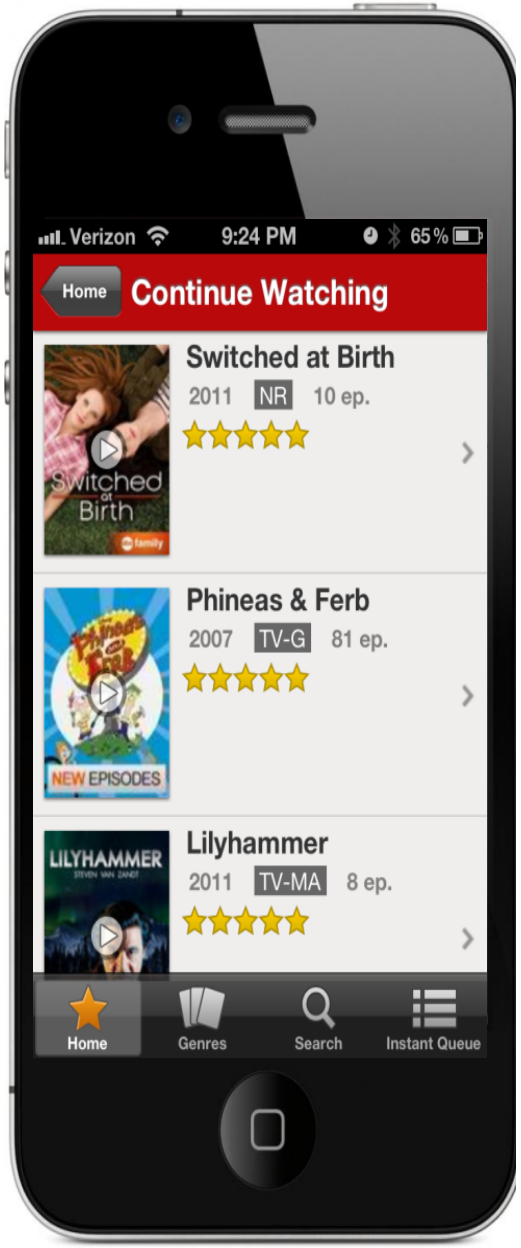
NETFLIX

Social Recommendations



Watch again & Continue Watching





NETFLIX

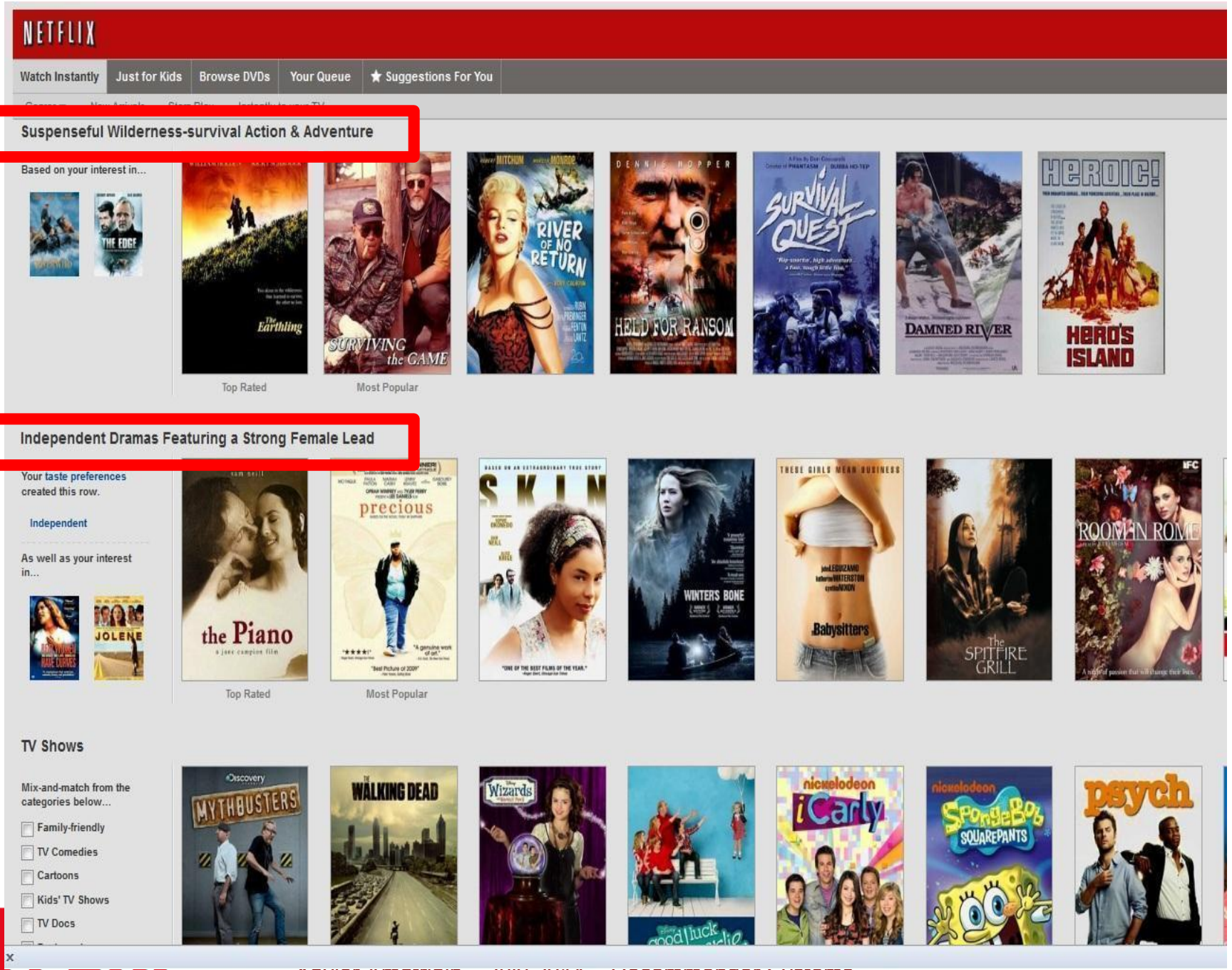
Genres

NETFLIX

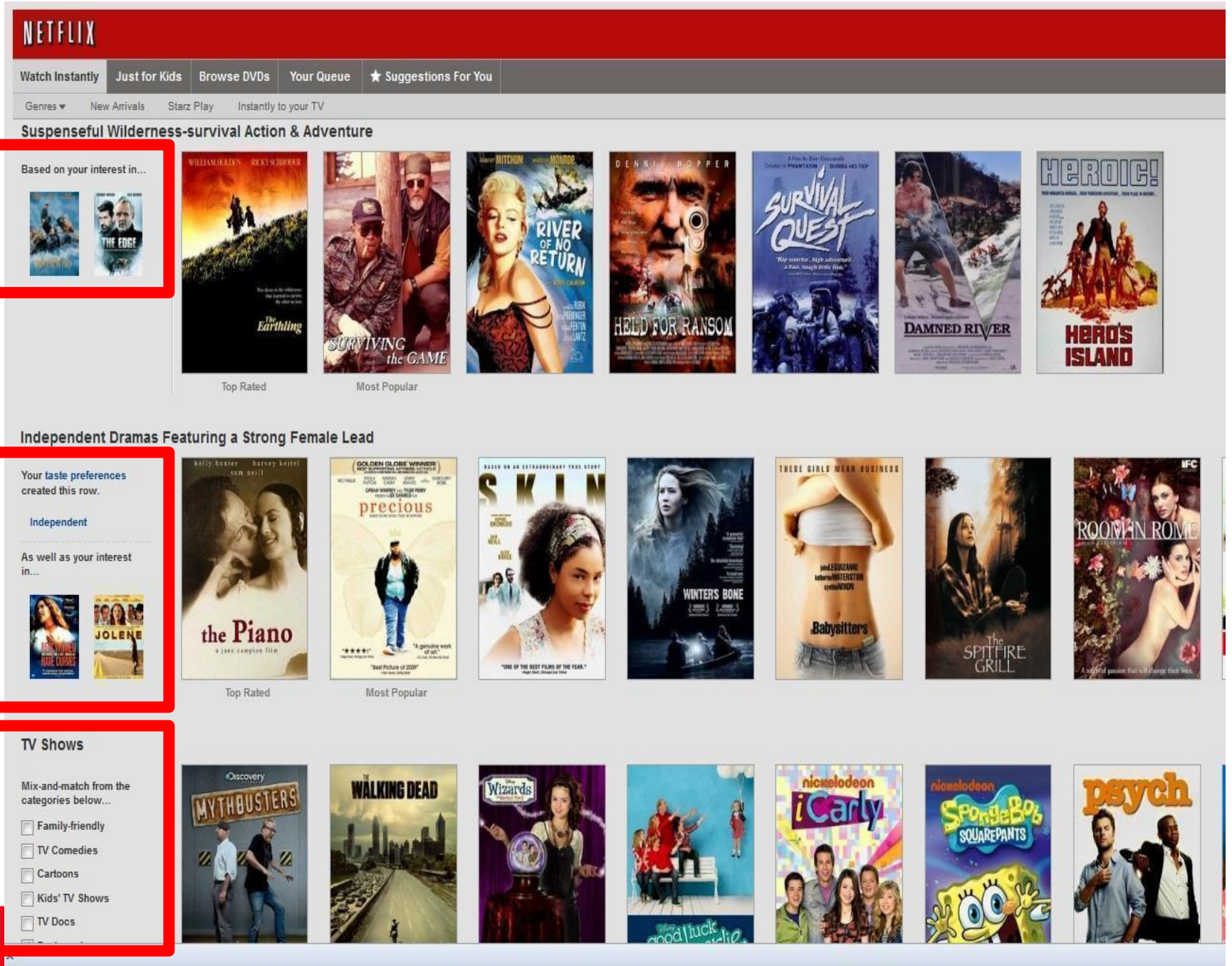
Genre rows

- Personalized genre rows focus on user interest
  - Also provide context and "evidence"
  - Important for member satisfaction – moving personalized rows to top on devices increased retention
- How are they generated?
  - **Implicit**: based on user's recent plays, ratings, & other interactions
  - **Explicit** taste preferences
  - **Hybrid**:combine the above
- Also take into account:
  - Freshness - has this been shown before?
  - Diversity– avoid repeating tags and genres, limit number of TV genres, etc.

Genres - personalization



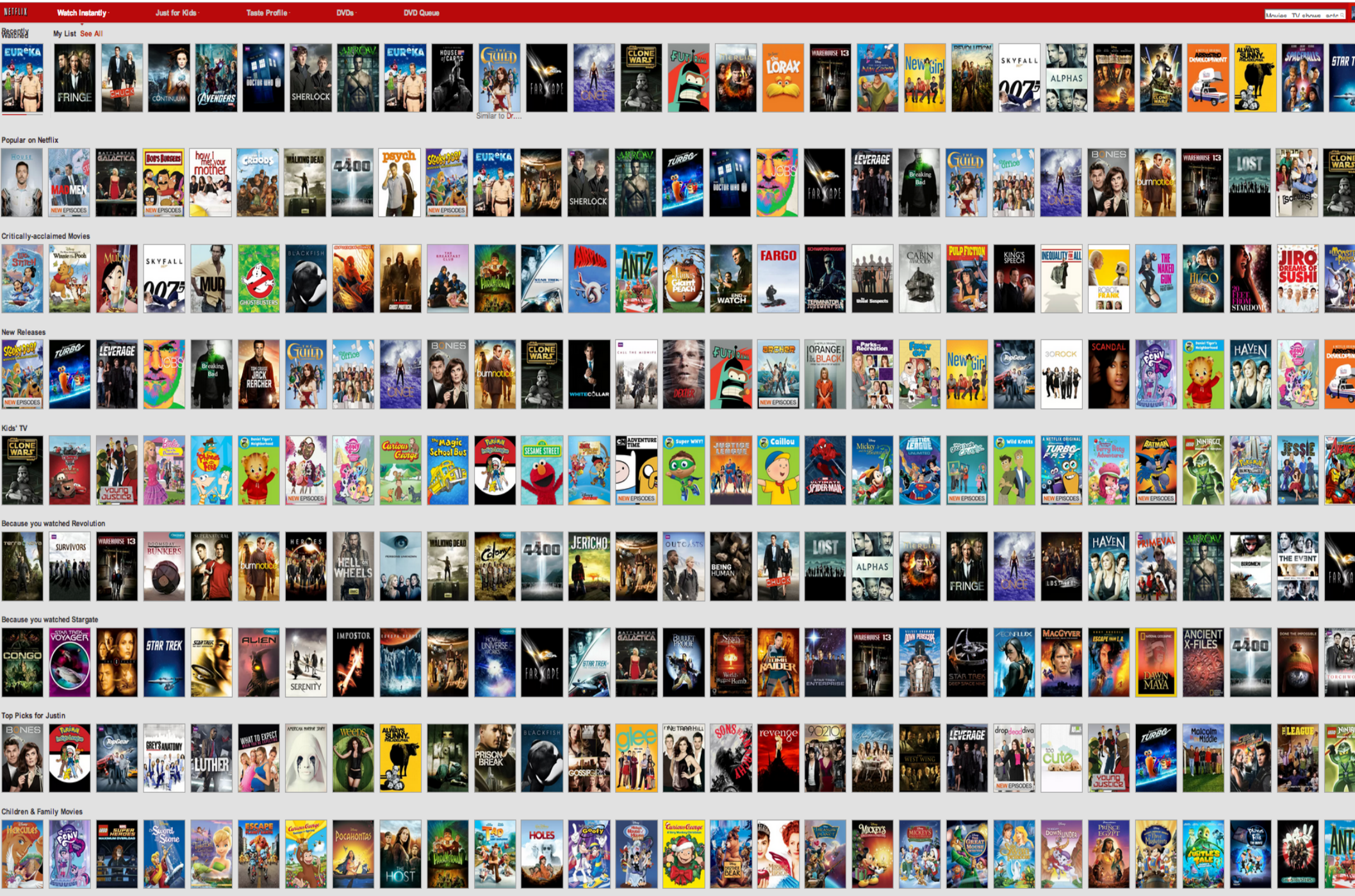
Genres - personalization



Similars

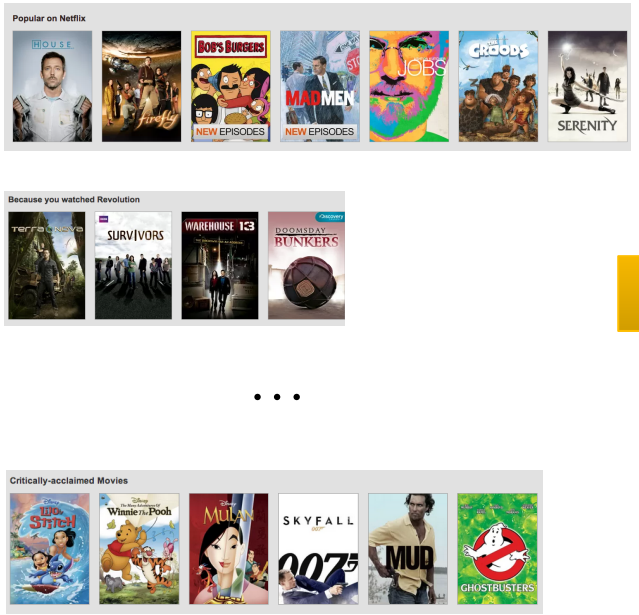
- Displayed in many different contexts
  - In response to user actions/context (search, queue add…)
  - More like… rows

Page Composition



Page Composition

10,000s of possible rows



Variable number of possible videos per row (up to thousands)

1 personalized page

10-40 rows

per device

Page Composition

Accurate vs. Diverse

Discovery vs. Continuation

Depth vs. Coverage

Freshness vs. Stability

Recommendations vs. Tasks

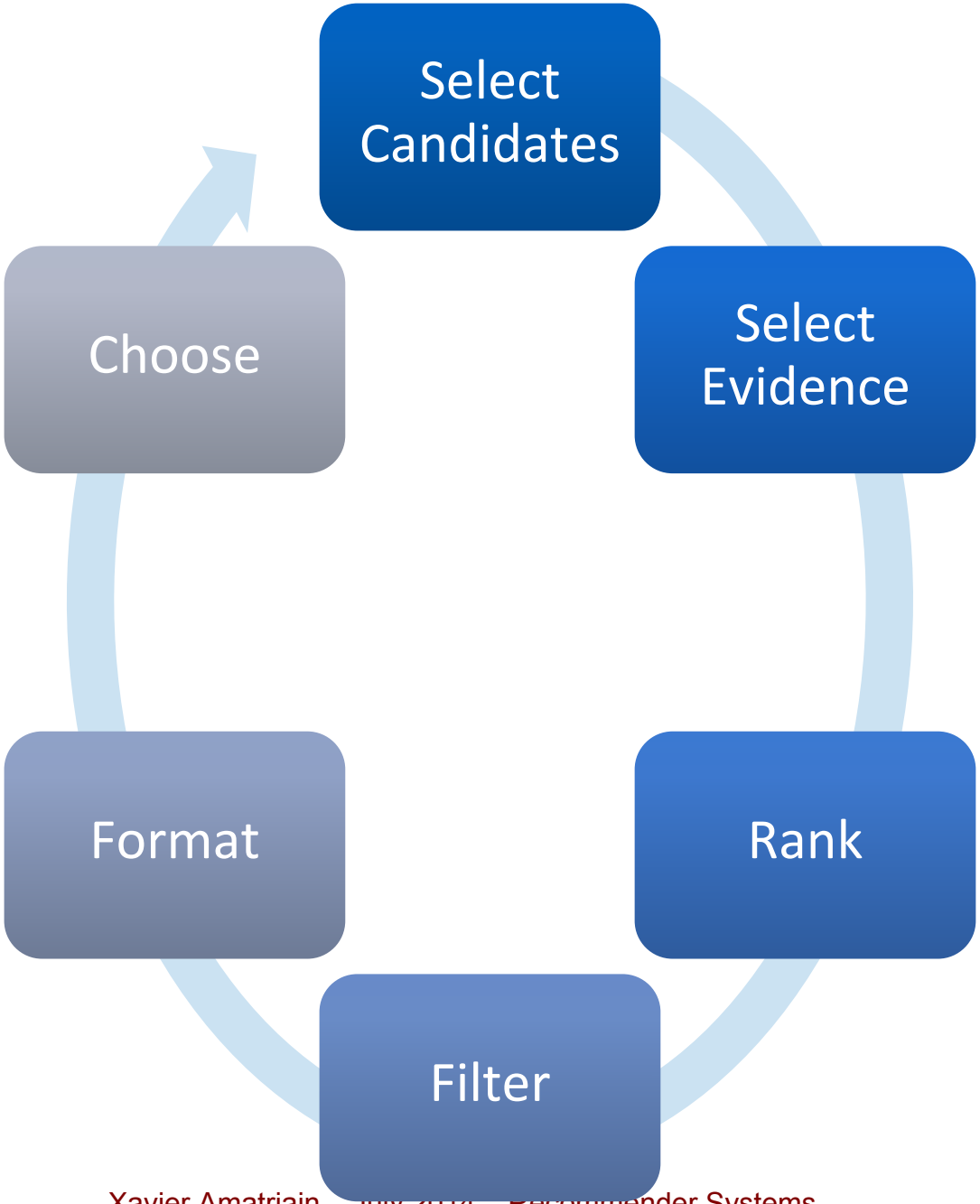
Page Composition

More likely to see

Less likely

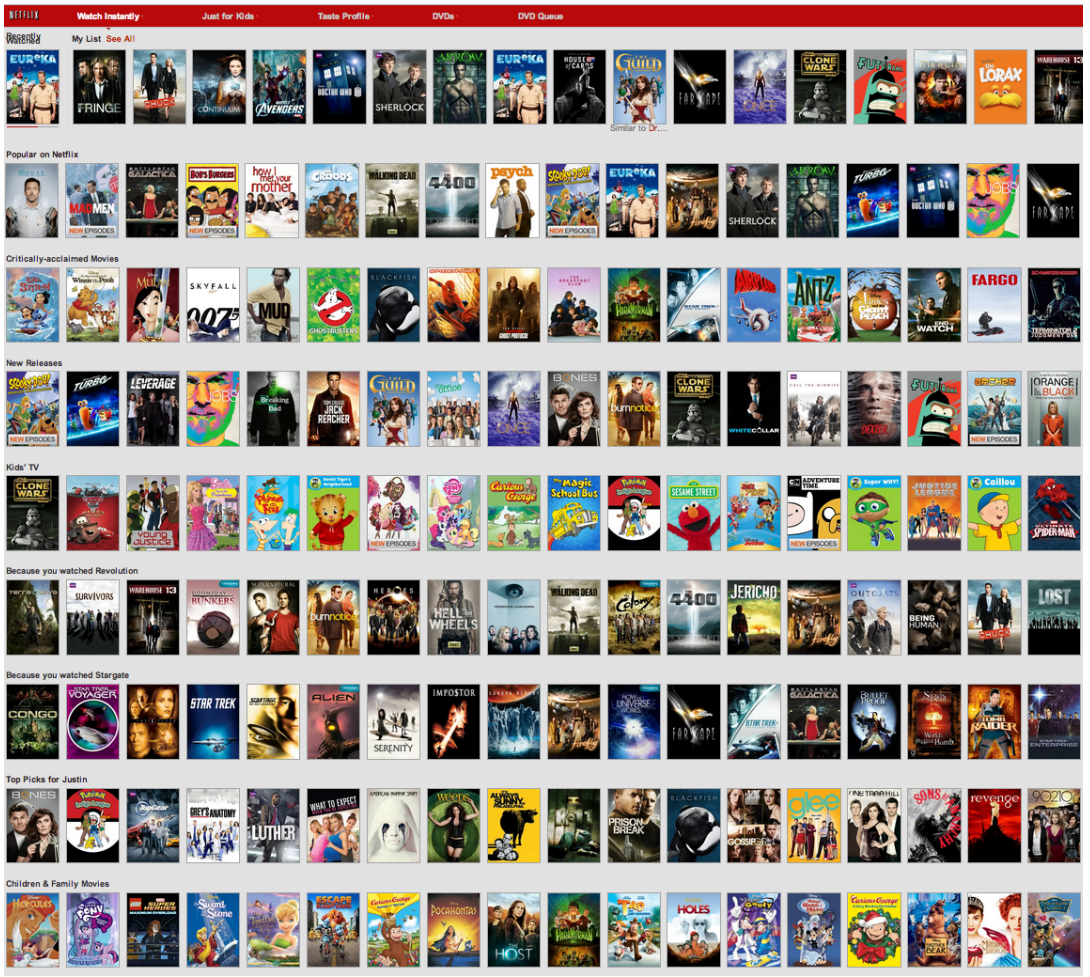
NETFLIX

Page Composition



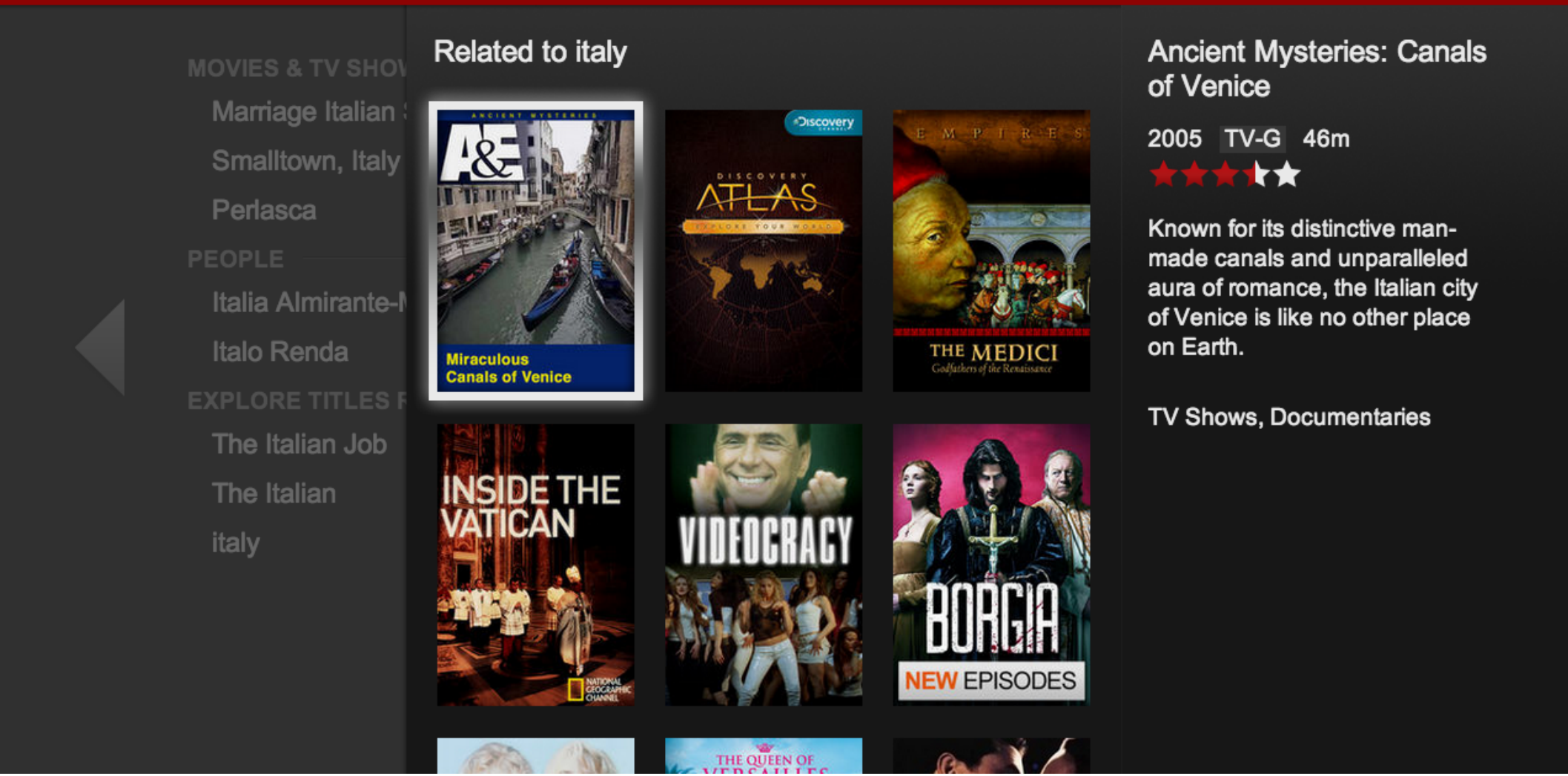
Row Features

- Quality of items
- Features of items
- Quality of evidence
- User-row interactions
- Item/row metadata
- Recency
- Item-row affinity
- Row length
- Position on page
- Context
- Title
- Diversity
- Freshness
- …

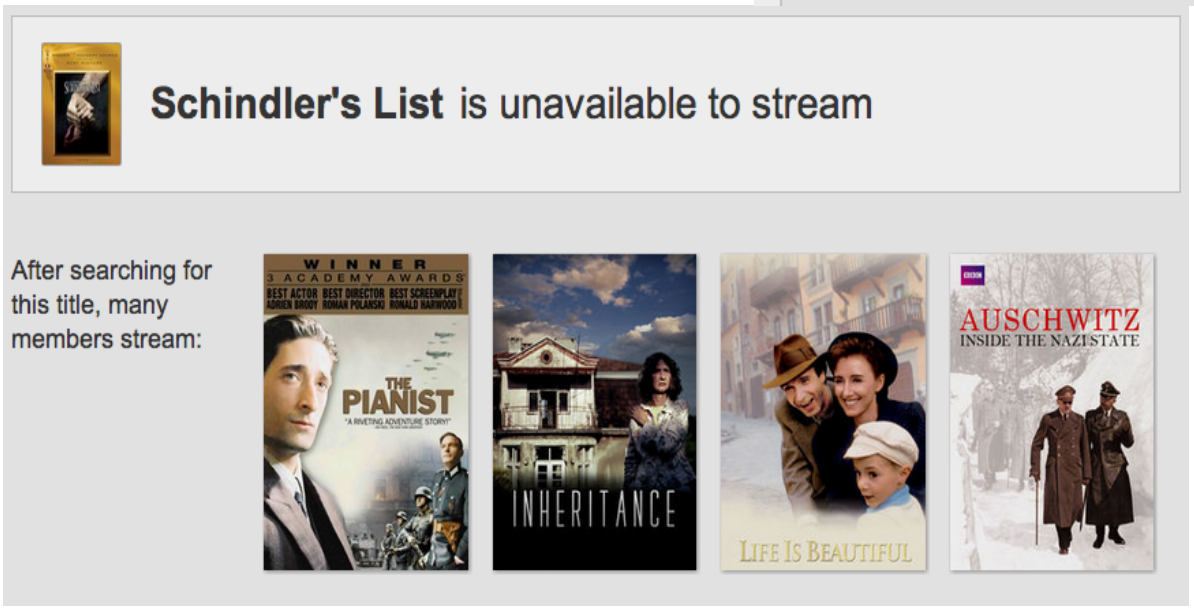


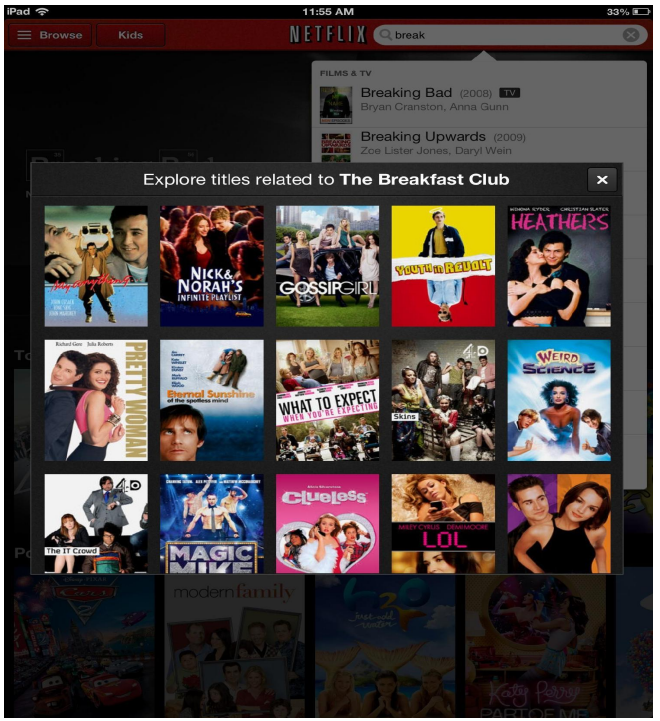
NETFLIX

Search Recommendations



Unavailable Title Recommendations





NETFLIX

Search Recommendations

Combination of Different sources

- Transition on Play after Query
- Similarity between user's (query, play)
- Editorial
- …

Metaphor: a System for Related Search Recommendations

Azarias Reda
University of Michigan
azarias@umich.edu

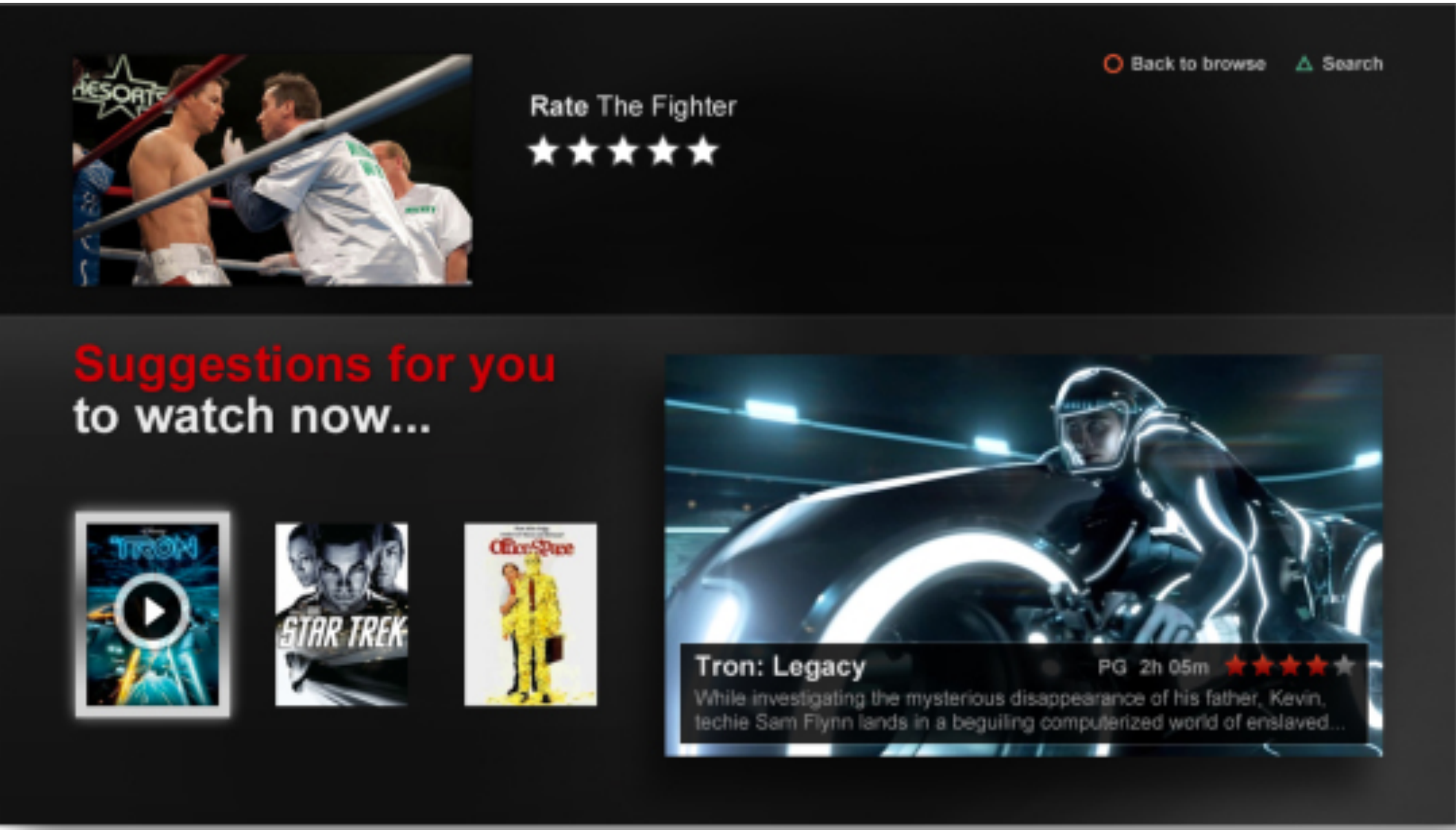
Yubin Park
University of Texas at Austin
yubin.park@utexas.edu

Mitul Tiwari
LinkedIn
mtiwari@linkedin.com

Christian Posse
LinkedIn
cposse@linkedin.com

Sam Shah
LinkedIn
samshah@linkedin.com

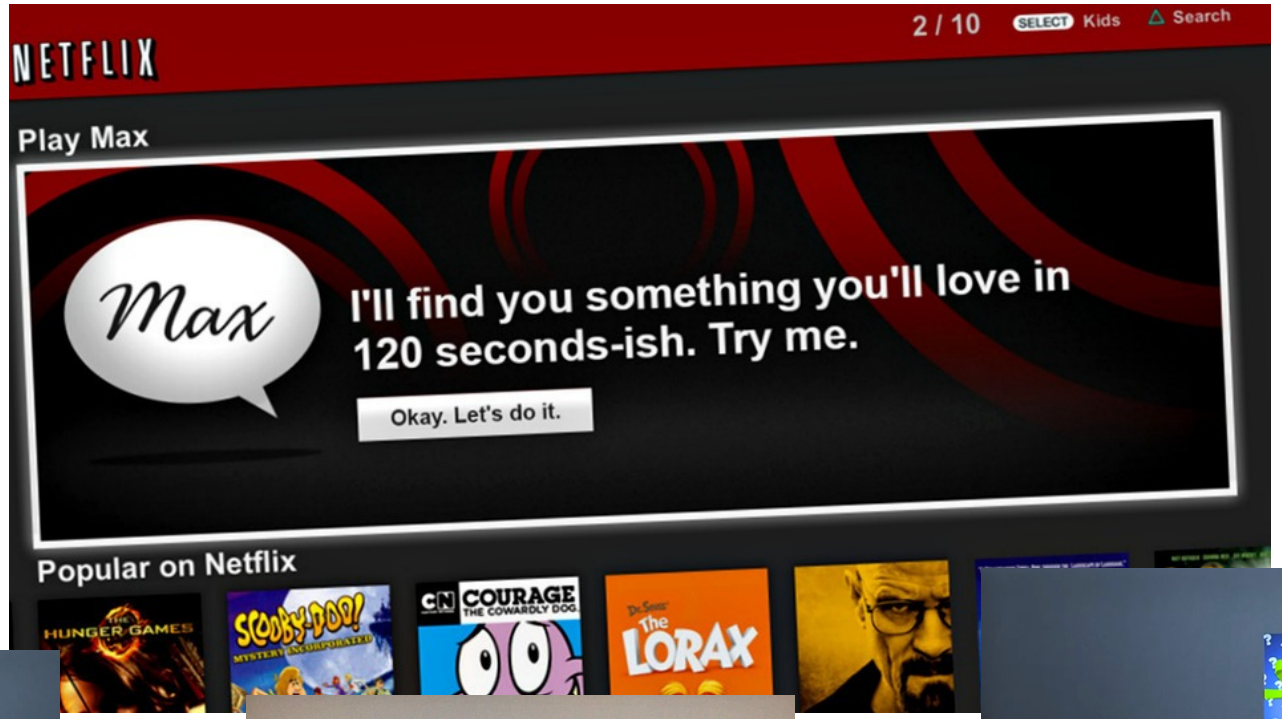
Postplay


Back to browse
Search
Rate The Fighter
Suggestions for you
to watch now...
TRON
STAR TREK
Tron: Legacy
PG 2h 05m
While investigating the mysterious disappearance of his father, Kevin, techie Sam Flynn lands in a beguiling computerized world of enslaved...

Billboard


iPad
4:20 AM
32 %
Browse
NETFLIX
Search
RANGO
Rango
2009 PG 2h 06min
Johnny Depp lends his voice to the title character, an adventurous family pet who leaves home to learn more about himself in this comic animated tale.
Play Now
Continue Watching
Psych: Season 3: Ep. 2 - Six Feet Under t...
Zombieland
The Office: Season 2, Ep.3 – Take Your ...
Top Picks for Navin

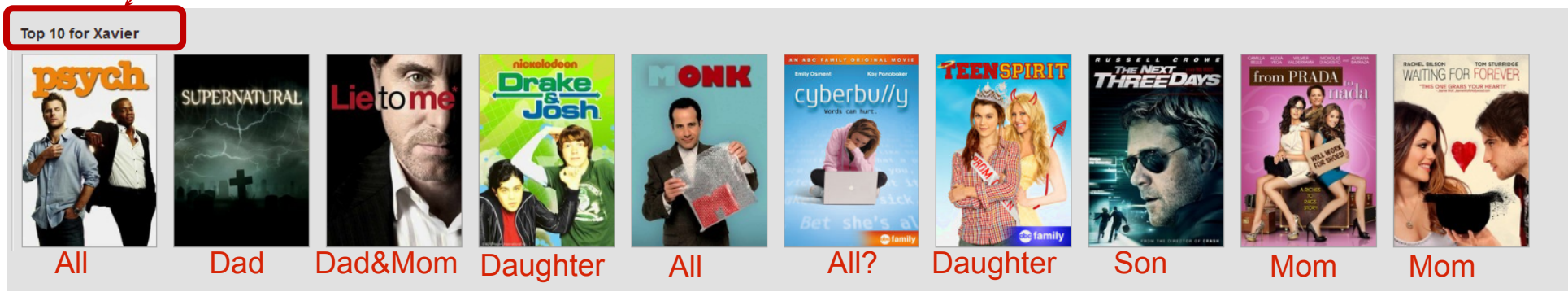
Gamification



NETFLIX

Diversity & Awareness

Personalization awareness



Diversity

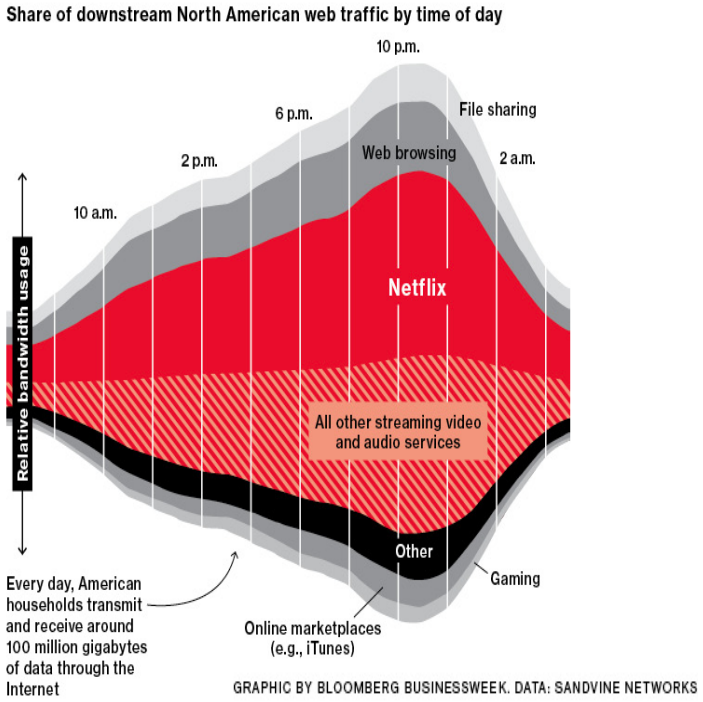
Data

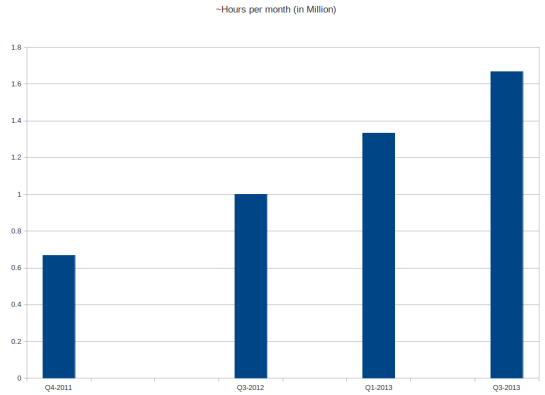
&

Models

NETFLIX

Netflix Scale

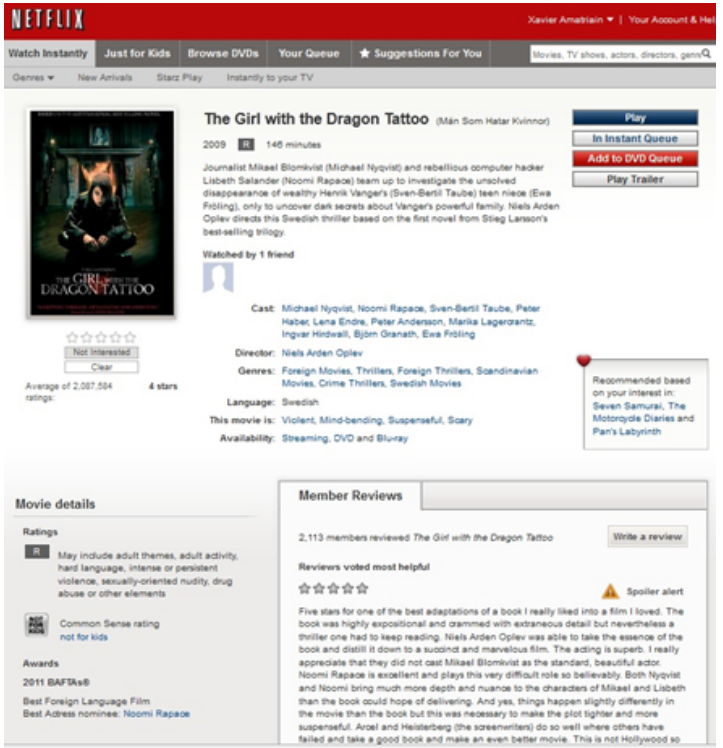




- > 48M members
- > 40 countries
- > 1000 device types
- > 5B hours in Q3 2013
- Plays: > 50M/day
- Searches: > 3M/day
- Ratings: > 5M/day
- Log 100B events/day
- 34.2% of peak US downstream traffic

Metadata

- Tag space is made of thousands of different concepts
- Items are manually annotated
- Metadata is useful
  - Especially for coldstart



How Netflix Reverse Engineered Hollywood

To understand how people look for movies, the video service created 76,897 micro-genres. We took the genre descriptions, broke them down to their key words, ... and built our own new-genre generator.

ALEXIS C. MADRIGAL | JAN 2 2014, 11:58 AM ET

The Atlantic's Netflix-Genre Generator

Slapstick Medical Underdog Comedies About Cats For Ages 0 to 2 With a Strong Female Lead

CREATE A GENRE: Gonzo (ultraniche genres) | Hollywood (film-making cliches) | Netflix (mimicking their style)

If you use Netflix, you've probably wondered about the specific genres that it suggests to you. Some of them just seem *so specific* that it's absurd. Emotional Fight-the-System Documentaries? Period Pieces About Royalty Based on Real Life? Foreign Satanic Stories from the 1980s?

NETFLIX

Social

- Can your “friends” interests help us better predict yours?
- The answer is similar to the Metadata case:
  - If we know enough about you, social information becomes less useful
  - But, it is very interesting for coldstarting
- And… social support for recommendations has been shown to matter

NETFLIX

Impressions

- We track what users get shown and their interactions
- Impression tracking is costly from an infrastructure perspective
- But, it can have very important applications
  - Correcting for presentation bias
  - Different forms of positive and negative feedback
  - ...

NETFLIX

Smart Models

- Regression models (Logistic, Linear, Elastic nets)
- GBDT/RF
- SVD & other MF models
- Factorization Machines
- Restricted Boltzmann Machines
- Markov Chains & other graphical models
- Clustering (from k-means to HDP)
- Deep ANN
- LDA
- Association Rules
- …

NETFLIX

Index

6. Conclusions

Conclusions

- For many applications such as Recommender Systems (but also Search, Advertising, and even Networks) understanding data and users is vital
- Algorithms can only be as good as the data they use as input
  - But the inverse is also true: you need a good algorithm to leverage your data
- Importance of User/Data Mining is going to be a growing trend in many areas in the coming years

Conclusions

- Recommender Systems (RS) is an important application of User Mining
- RS have the potential to become as important as Search is now
- However, RS are more than User Mining
  - 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

7. 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

References

- 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.
- 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.

References

- 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.
- 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.
- 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.

References

- 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.

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!

Questions?

Xavier Amatriain

xavier@netflix.com

@xamat

NETFLIX