@xamat

The Recommender Problem *Revisited*

Xavier Amatriain
Research/Engineering Director @ Netflix

Bamshad Mobasher
Professor @ DePaul University

NETFLIX

Index

PART I - Xavier Amatriain (2h)

1. The Recommender Problem
2. Traditional Recommendation Methods
3. Beyond Traditional Methods

PART 2 - Bamshad Mobasher (1h)

1. Context-aware Recommendations

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

Index

NETFLIX

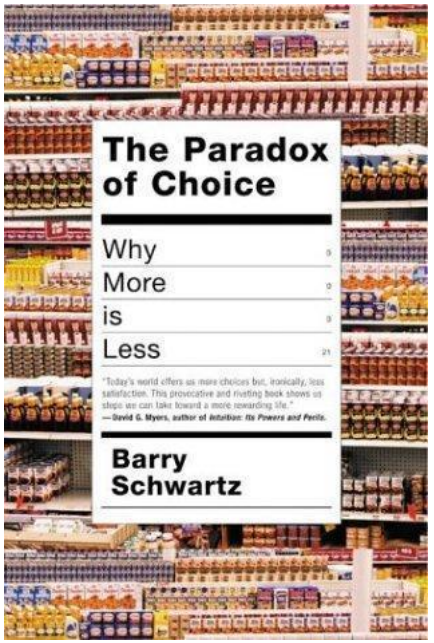
1. The Recommender Problem

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

Everything is personalized

NETFLIX

The “Recommender problem”

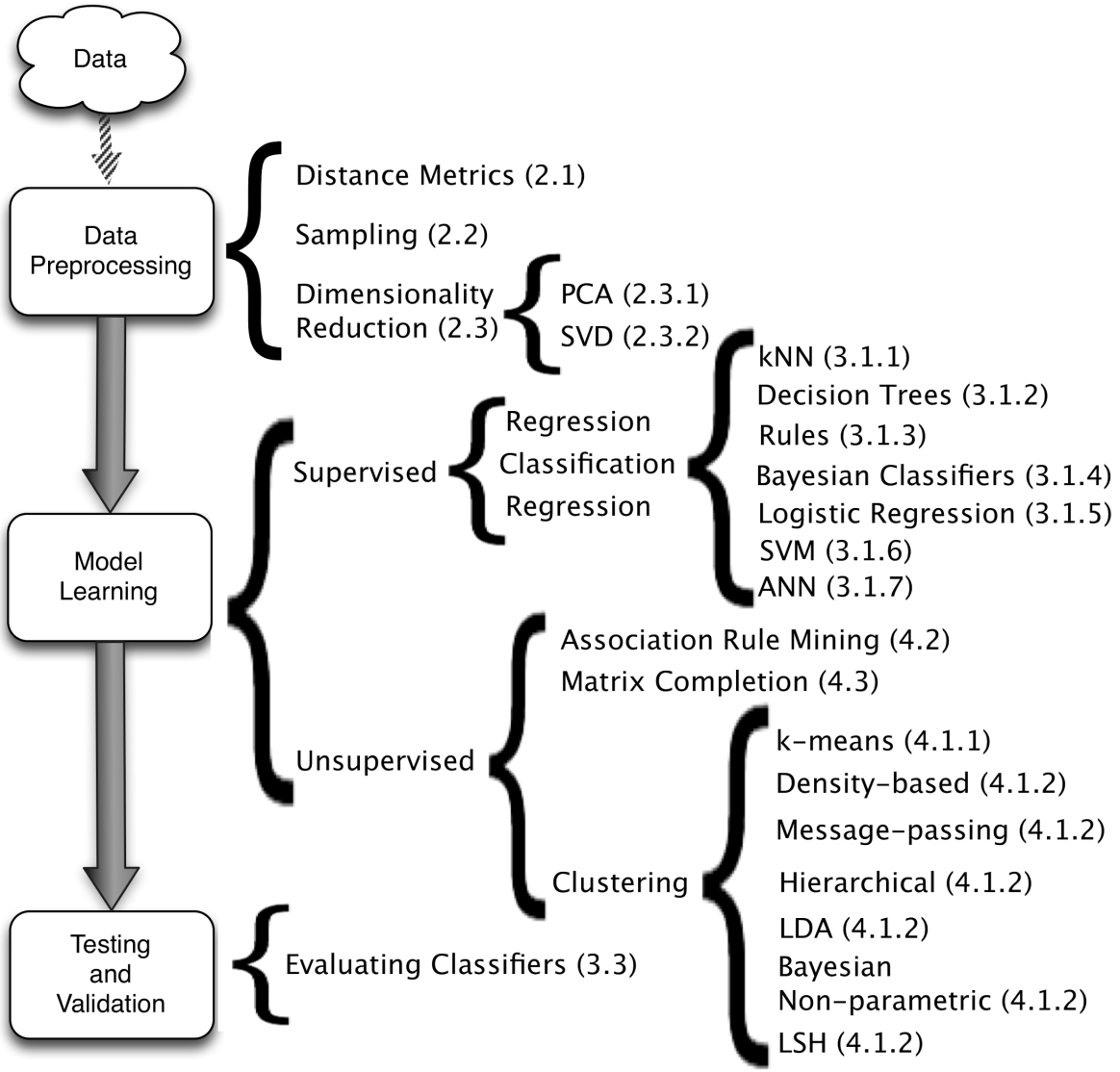
- *Traditional* definition: Estimate a utility function that automatically predicts how a user will like an item.
- Based on:
  - Past behavior
  - Relations to other users
  - Item similarity
  - Context
  - …

NETFLIX

Recommendation as data mining

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

(Amatriain et al. Data Mining Methods for Recommender Systems in Recommender Systems Handbook)



Machine Learning + all those other things

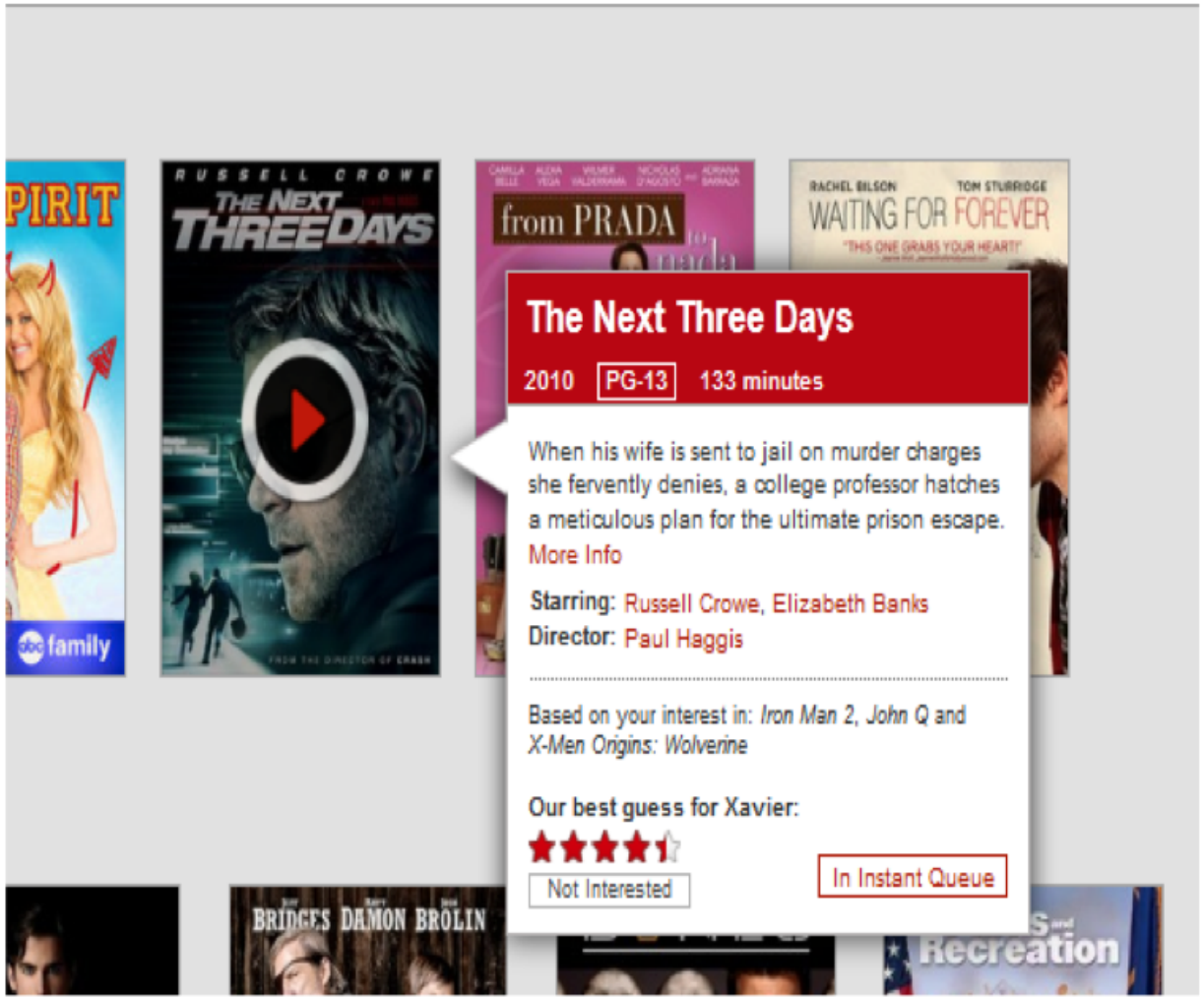
- User Interface
- System requirements (efficiency, scalability, privacy....)
- Serendipity
- Diversity
- Awareness
- Explanations
- …

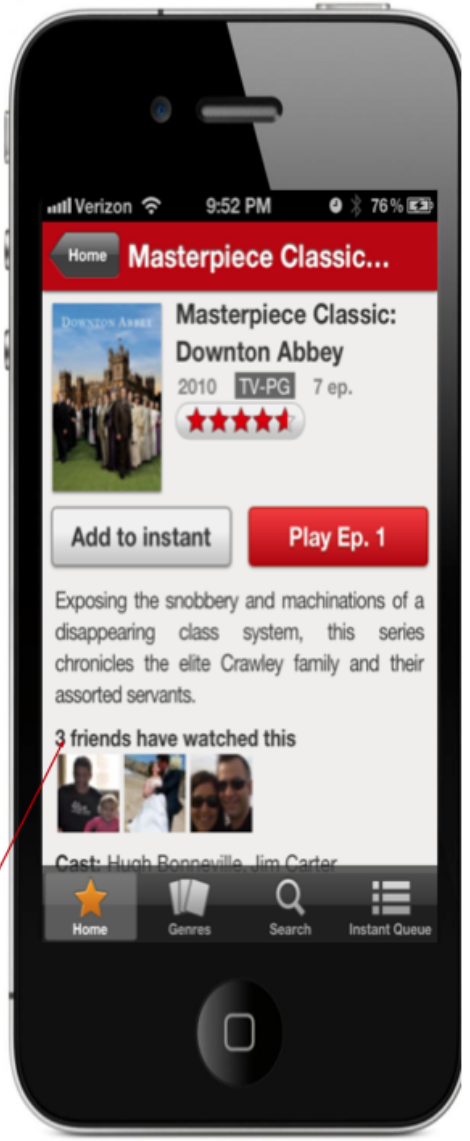
NETFLIX

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)

Explanation/Support for Recommendations

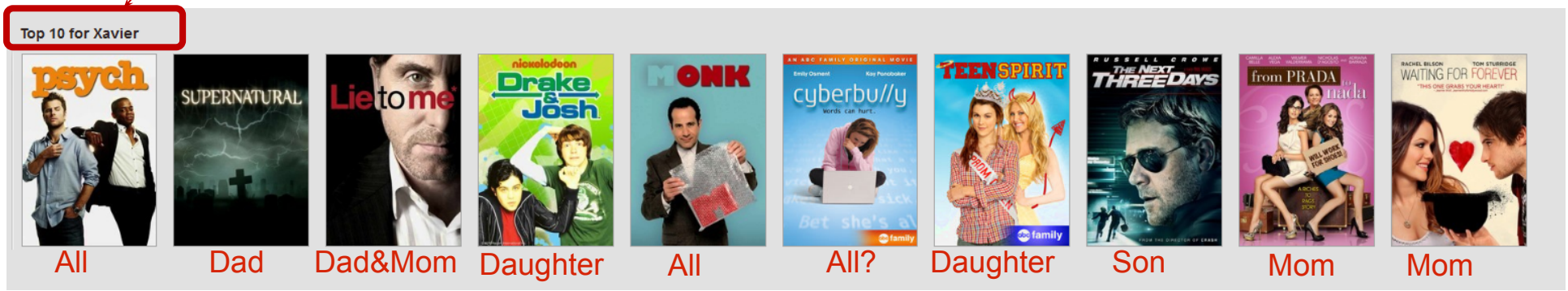




Social Support

Diversity & Awareness

Personalization awareness

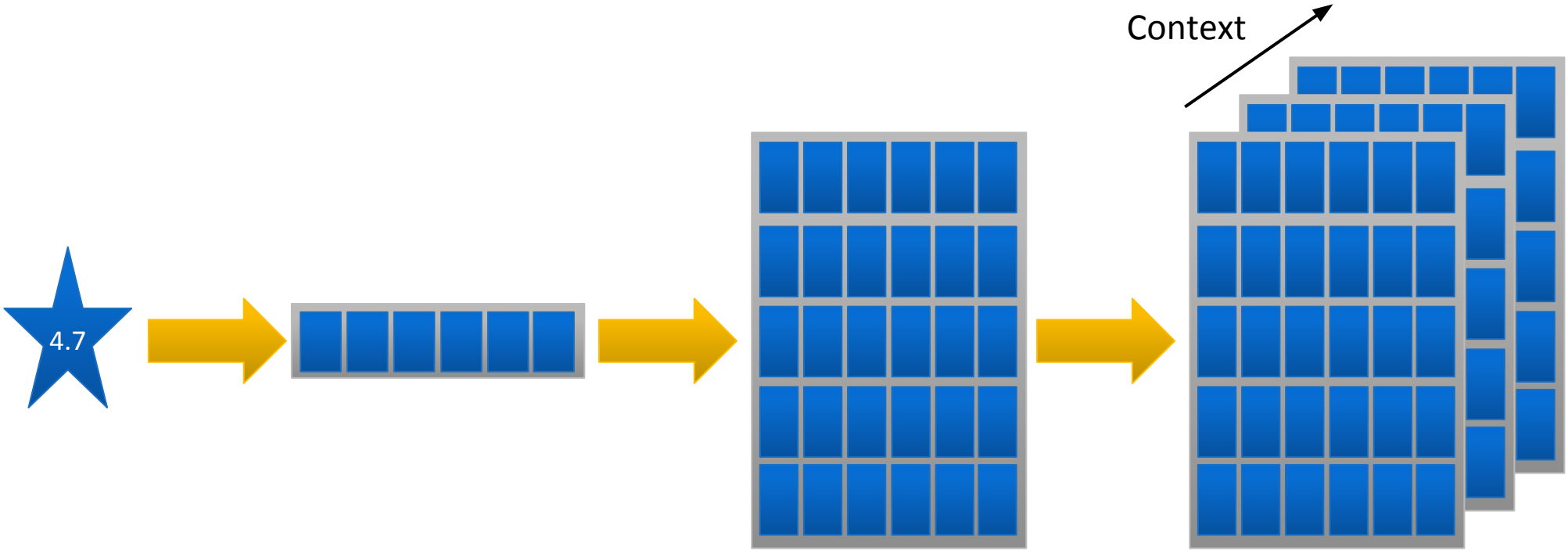


Diversity

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 through ensembles

Evolution of the Recommender Problem



Rating

Ranking

Page Optimization

Context-aware Recommendations

NETFLIX

Index

NETFLIX

2. Traditional Approaches

2.1. Collaborative Filtering

The CF Ingredients

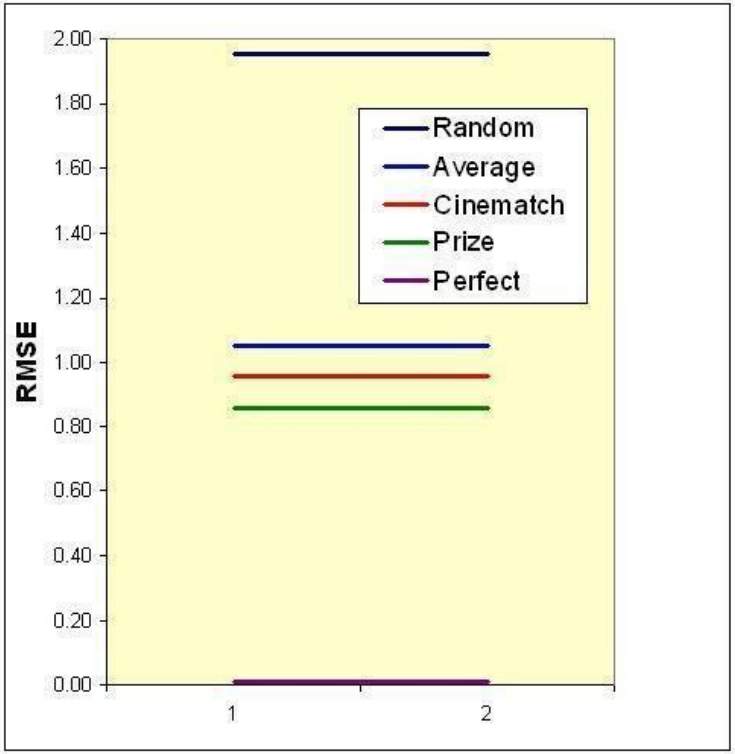
- List of **m Users** and a list of **n Items**
- Each user has a **list of items** with associated **opinion**
  - **Explicit opinion** - a rating score
  - Sometime the rating is **implicitly** – purchase records or listen to tracks
- **Active user** for whom the CF prediction task is performed
- **Metric** 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.

Personalized vs Non-Personalised CF

- CF recommendations are **personalized**: prediction based only on **similar users**
- **Non-personalized** collaborative-based recommendation: averagge the recommendations of **ALL** the users
- How would the two approaches compare?

Personalized vs. Not Personalized

- Netflix Prize: it is very simple to produce "reasonable" recommendations and extremely difficult to improve them to become "great"
- But there is a huge difference in business value between reasonable and great



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

NETFLIX

User-based Collaborative Filtering

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

NETFLIX

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

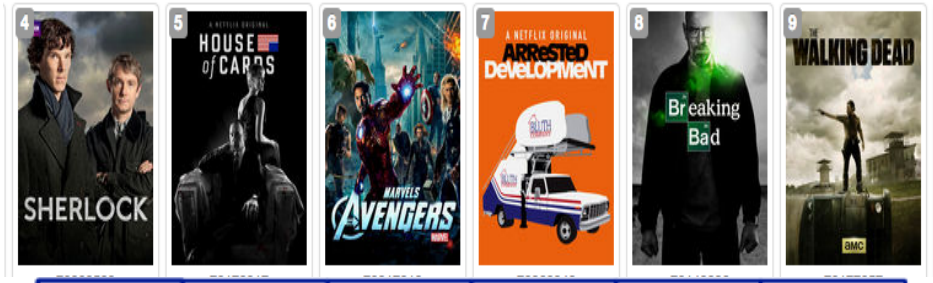
$$v_{ij}^* = K \sum_{v_{kj} \neq ?} u_{jk} v_{kj} \quad \text{or} \quad v_{ij}^* = v_i + K \sum_{v_{kj} \neq ?} u_{jk} (v_{kj} - v_k)$$

- Similarity can be computed by Pearson correlation

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

NETFLIX

User-based CF Example



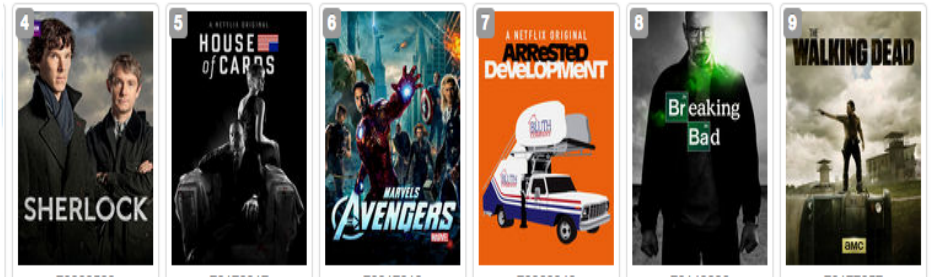
sim(u,v)

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

NA

NA

User-based CF Example

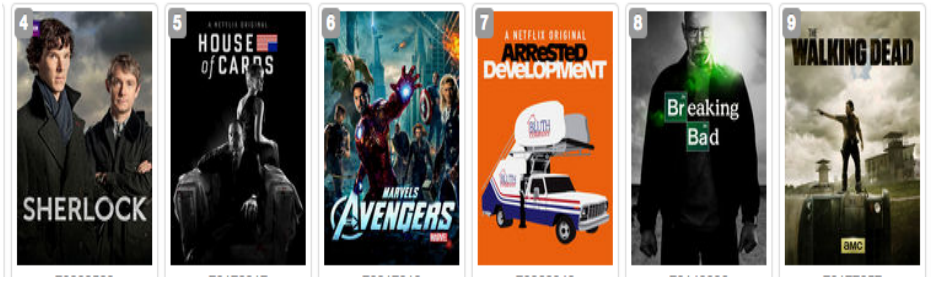


	2		2	4	5		NA
	5		4			1	0.87
			5		2		
		1		5		4	
 			4			2	
	4	5		1			NA

sim(u,v)

NETFLIX

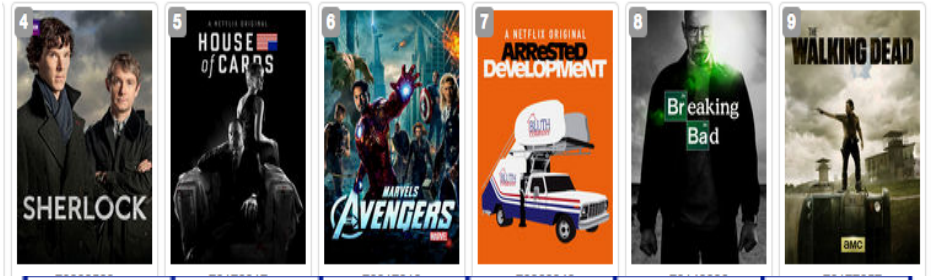
User-based CF Example



						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

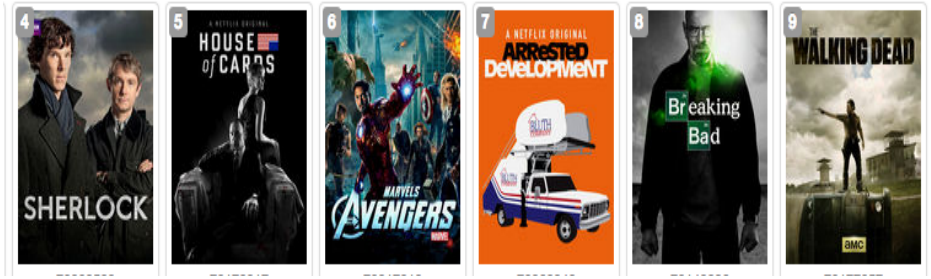
NETFLIX

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

Item-based 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: find users who have rated items and apply a similarity function to their ratings.
- Cosine-based Similarity (difference in rating scale between users is not taken into account)

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

- Adjusted Cosine Similarity (takes care of difference in rating scale)

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

NETFLIX

Challenges of memory-based CF

NETFLIX

CF: Pros/Cons

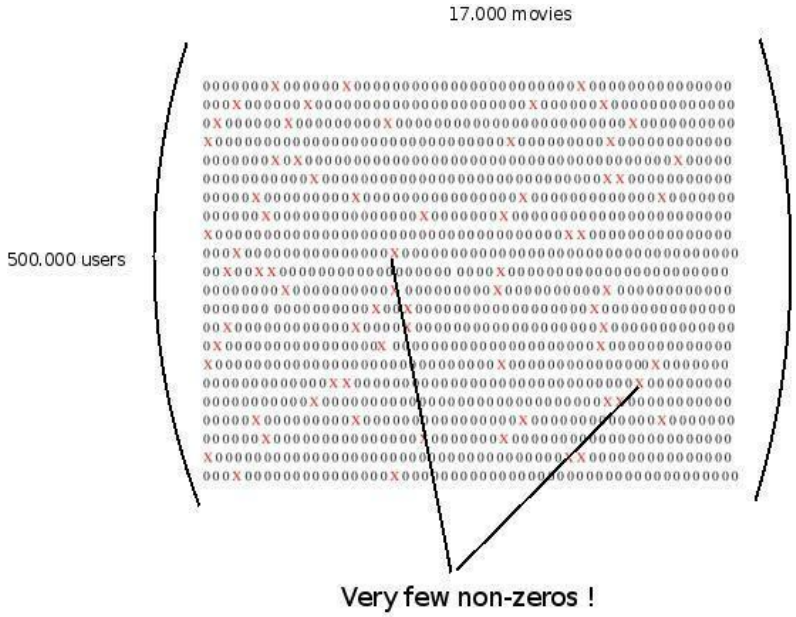
- Requires **minimal knowledge** engineering efforts
- Users and products are symbols without any internal structure or characteristics
- Produces good-enough results in most cases

Challenges:

- **Sparsity** – evaluation of large itemsets where user/item interactions are under 1%.
- **Scalability** - Nearest neighbor require computation that grows with both the number of users and the number of items.

The Sparsity Problem

- Typically: large product sets, user ratings for a small percentage of them
  (e.g. Amazon: millions of books and a user may have bought hundreds of books)
- If you represent the Netflix Prize rating data in a User/Movie matrix you get…
  - 500,000 x 17,000 = 8.5 B positions
  - Out of which only 100M are non-zero
- Number of users needs to be ~ 0.1 x size of the catalog



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.
- Model-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

Netflix Prize

COMPLETED

What we were interested in:

- High quality *recommendations*

Proxy question:

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

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

NETFLIX

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

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

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

- **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)

NETFLIX

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)

NETFLIX

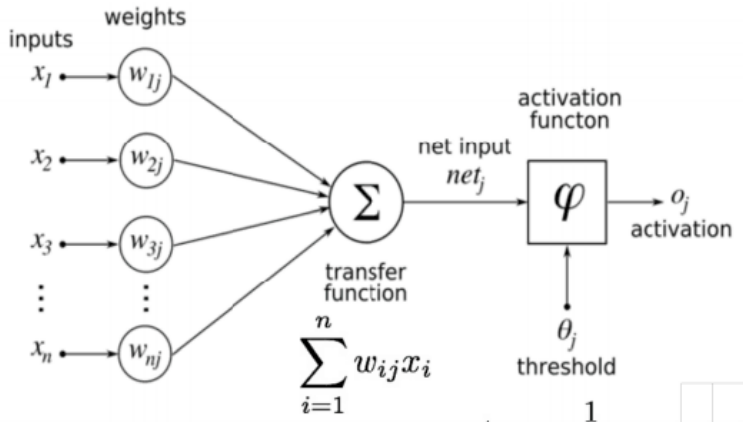
Restricted Boltzmann Machines

- 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
- Restrict the connectivity to make learning easier.

  Only one layer of hidden units.

  No connections between hidden units.

  Hidden units are independent given visible states



RBM for Recommendations

- Each visible unit = an item
- Num. of hidden units a is parameter
- In training phase, for each user:
  - If user rated item, v_i is activated
  - Activation states of v_i = inputs to h_j
  - Based on activation, h_j is computed
  - Activation state of h_j becomes input to v_i
  - Activation state of v_i is recalculated
  - Difference between current and past activation state for v_i used to update weights w_{ij} and thresholds
- In prediction phase:
  - For the items of the user the v_i are activated
  - Based on this the state of the h_j is computed
  - The activation of h_j is used as input to recompute the state of v_i
  - Activation probabilities are used to recommend items

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

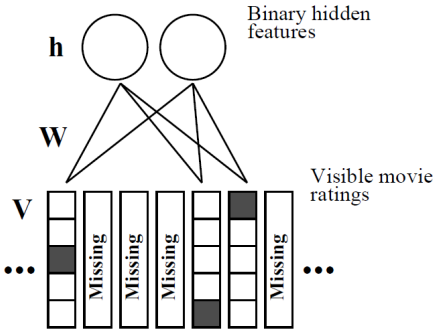
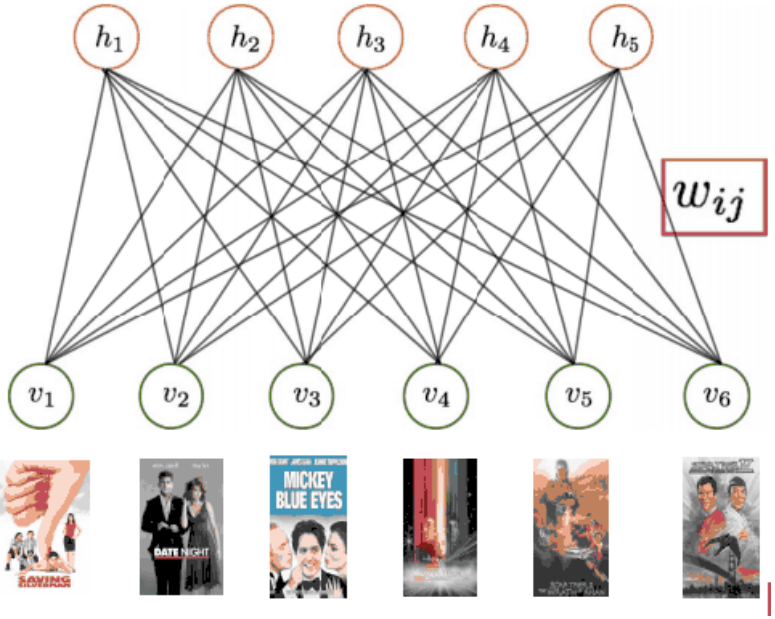


Figure 1. A restricted Boltzmann machine with binary hidden units and softmax visible units. For each user, the RBM only includes softmax units for the movies that user has rated. In addition to the symmetric weights between each hidden unit and each of the $K = 5$ values of a softmax unit, there are 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.



NETFLIX

Putting all together

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

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.

Clustering

NETFLIX

Clustering

- Goal: **cluster** users and compute per-cluster "typical" preferences
- Users receive recommendations computed at the cluster level

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

Other “interesting” clustering techniques

- k-means and all its variations
- Affinity Propagation
- Spectral Clustering
- Non-parametric Bayesian Clustering (e.g. HDPs)

Association Rules

NETFLIX

Association rules

- Past purchases are interpreted as transactions of “associated” items
- 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
- Fast to implement and execute (e.g. A Priori algorithm)

		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				

NETFLIX

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)
- E.g. Logistic Regression, Bayesian Networks, Support Vector Machines, Decision Trees, etc...

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)

Index

NETFLIX

2.2 Content-based Recommenders

Content-Based Recommendation

- Recommendations based on content of items rather than on other users' opinions/interactions
- Goal: recommend items **similar** to those the user liked
- 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)
- The user model can be a **classifier** based on whatever technique (Neural Networks, Naïve Bayes...)

Pros/cons of CB Approach

Pros

- No need for data on other users: No cold-start or sparsity
- Able to recommend to users with unique tastes.
- Able to recommend new and unpopular items
- Can provide explanations by listing content-features

Cons

- Requires content that can be encoded as meaningful features (difficult in some domains/catalogs)
- Users represented as learnable function of content features.
- Difficult to implement serendipity
- Easy to overfit (e.g. for a user with few data points)

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

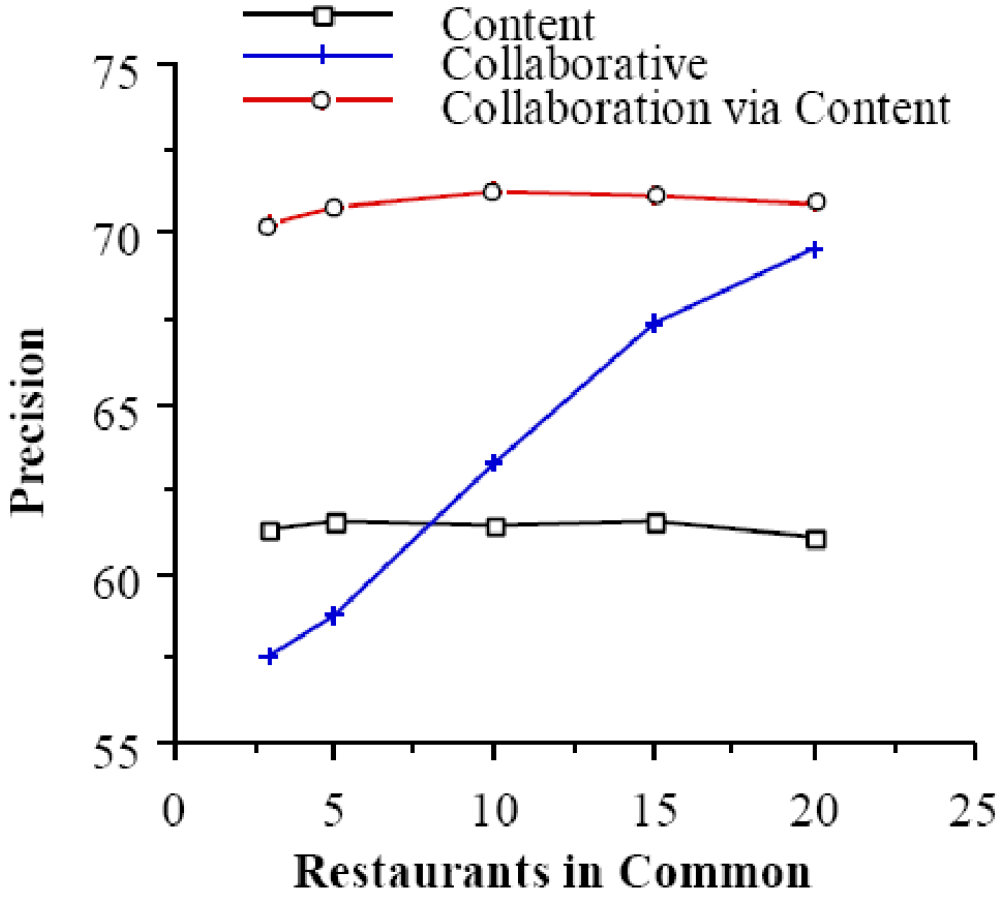
Index

NETFLIX

2.3 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

NETFLIX

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

NETFLIX

Index

NETFLIX

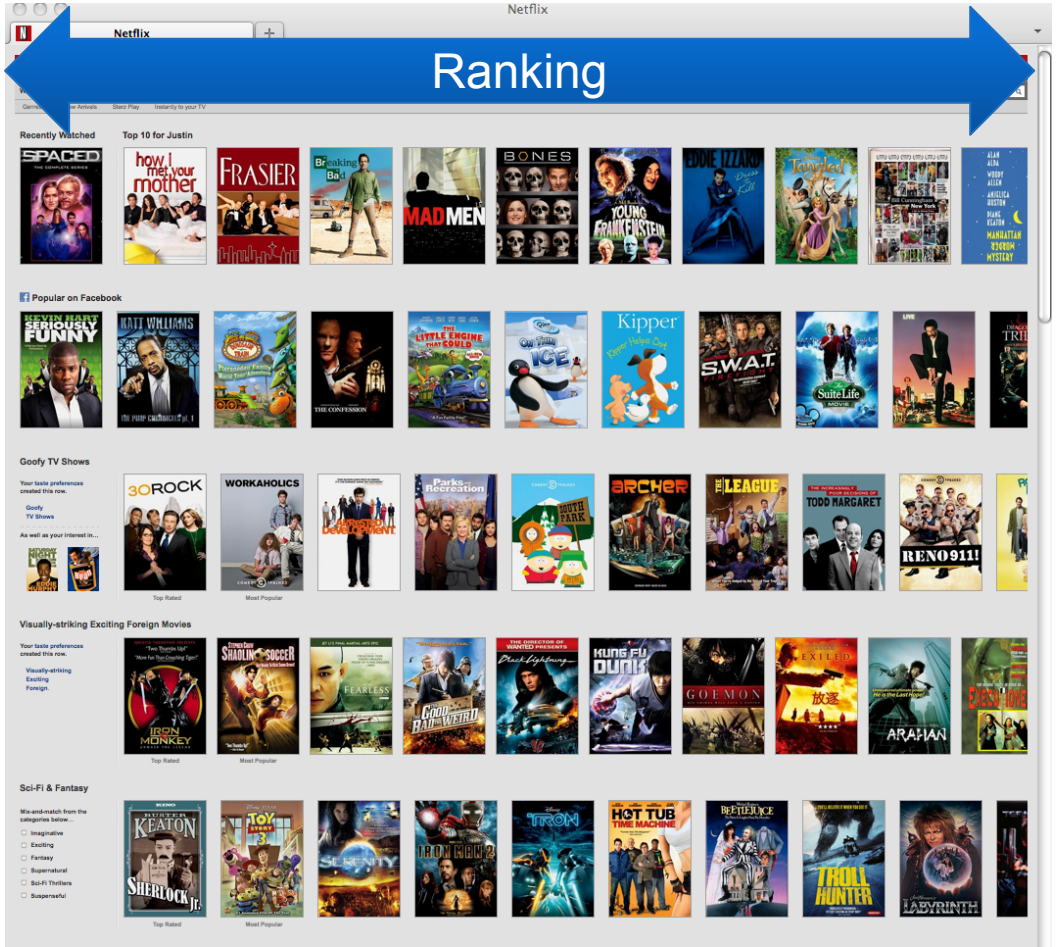
3. Beyond traditional approaches to Recommendation

3.1 Ranking

NETFLIX

Ranking

Key algorithm, sorts titles in most contexts

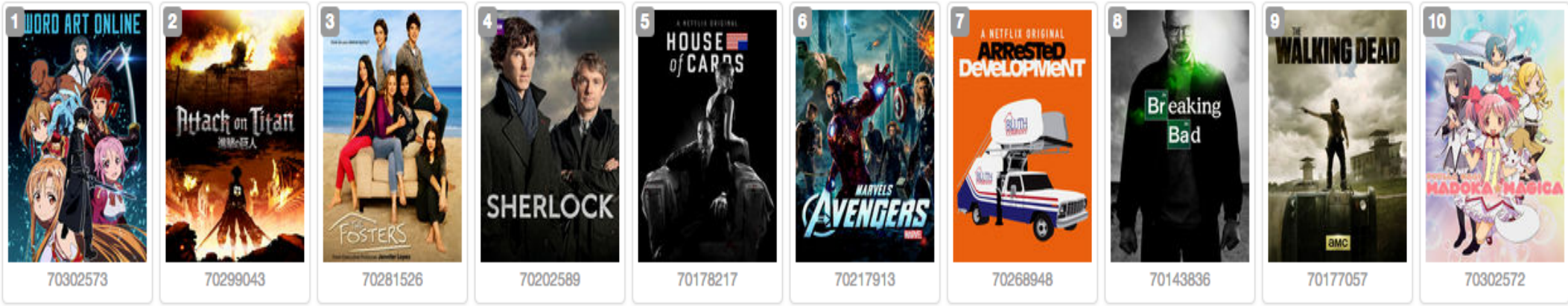


NETFLIX

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

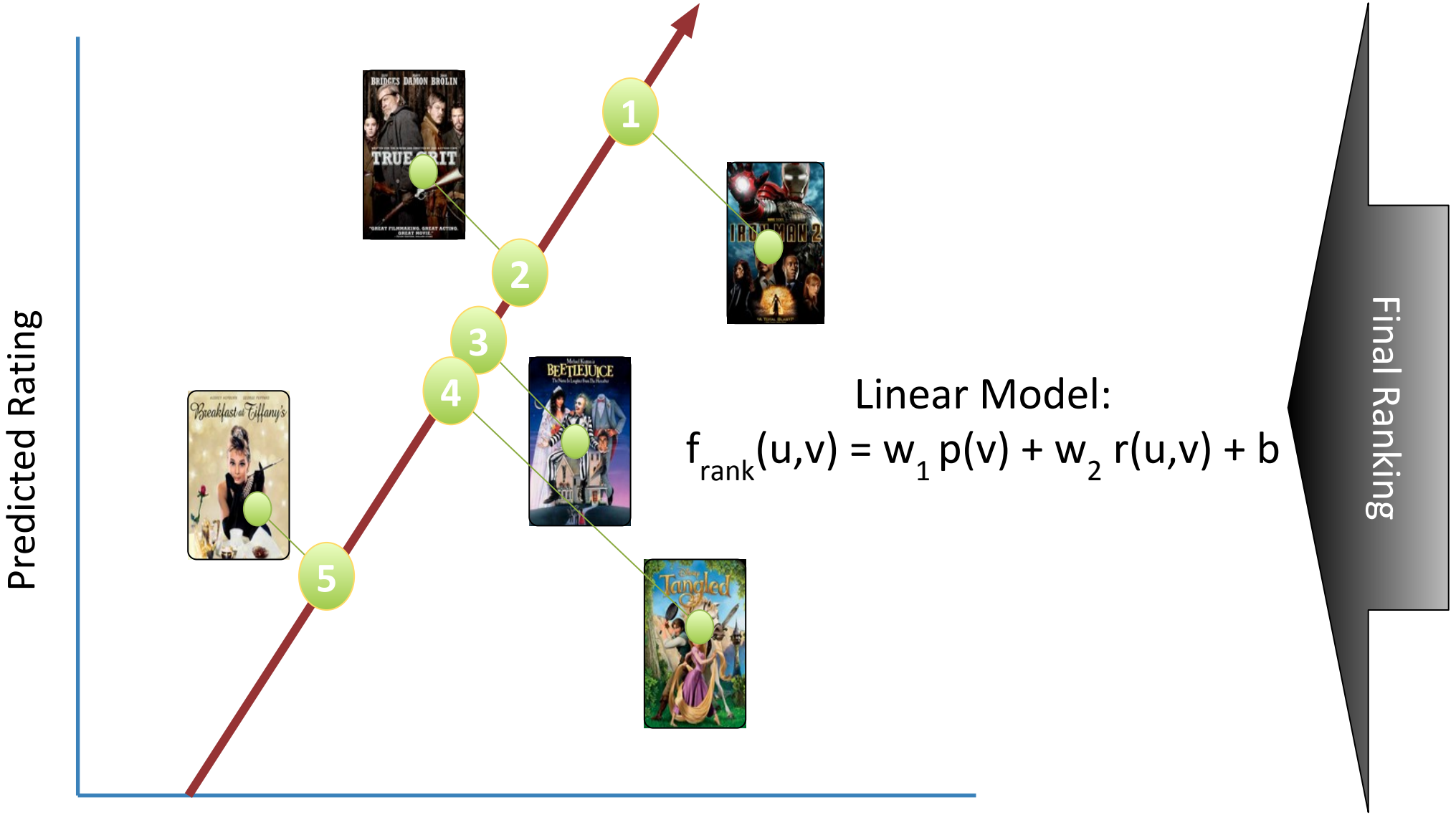
Niche titles

High average ratings… by those who would watch it

NETFLIX

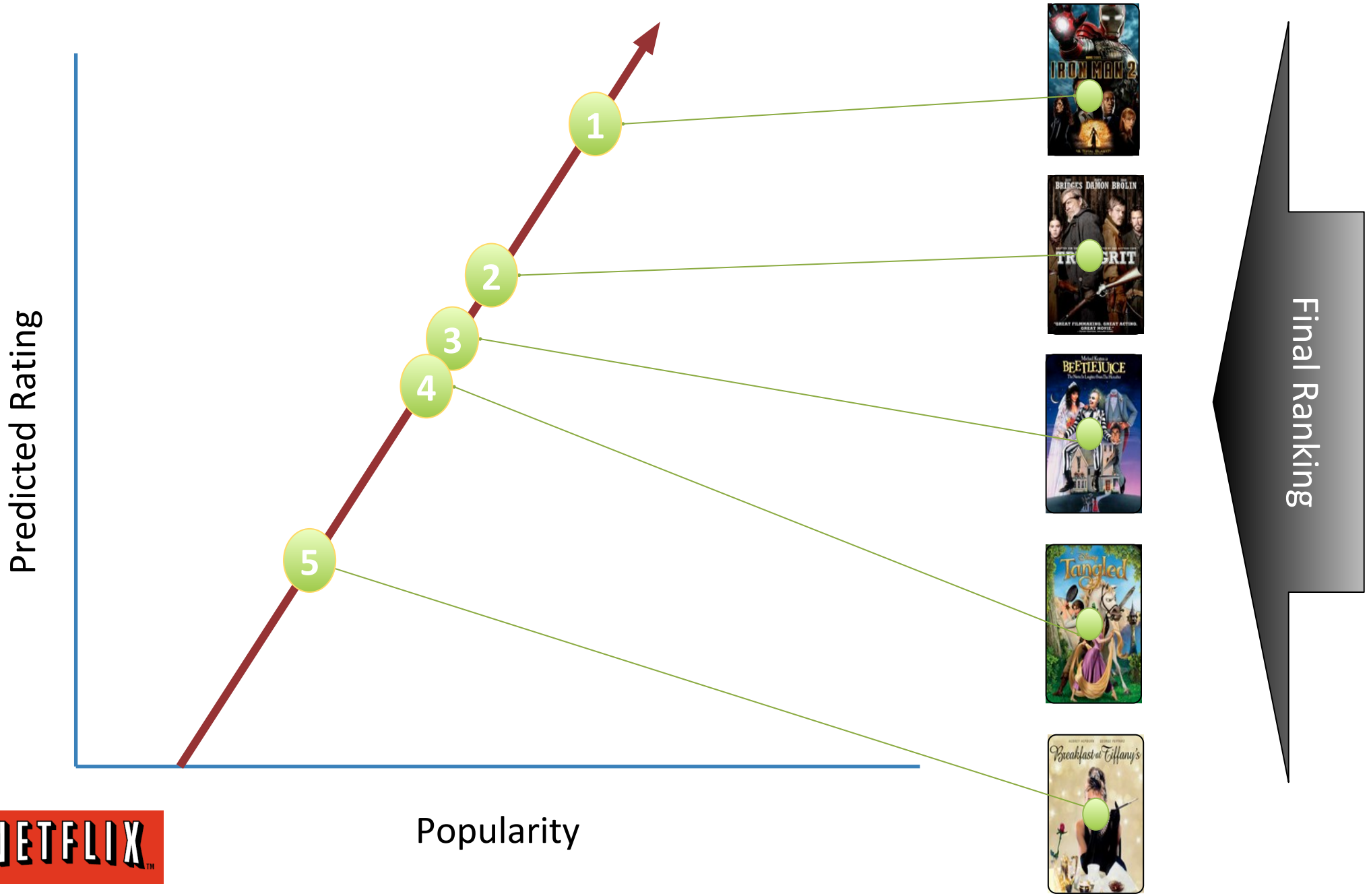
NETFLIX

Example: Two features, linear model



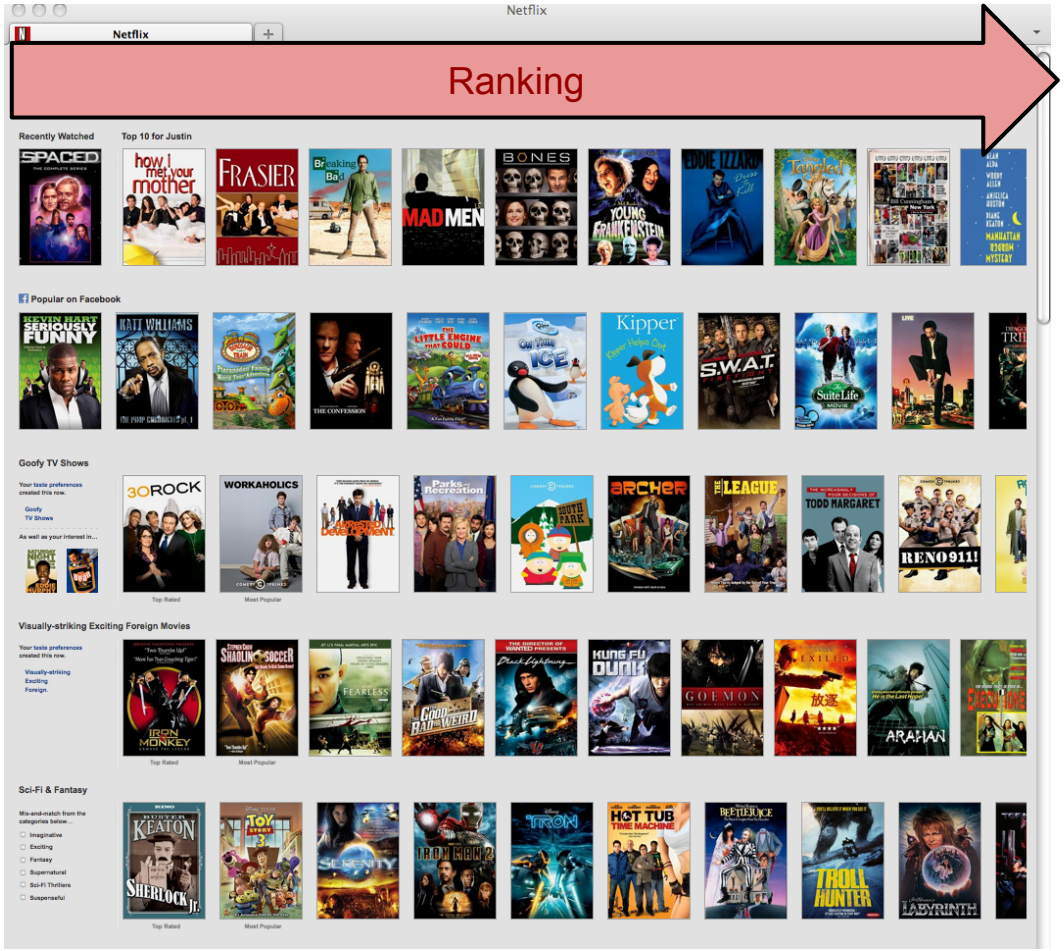
NETFLIX

Example: Two features, linear model



NETFLIX

Ranking



NETFLIX

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 (e.g. non-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
   - LambdaMart, 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)
     - Genetic Programming or Simulated Annealing
     - Gradient descent on smoothed version of objective function (e. g. CLiMF or TFMAP)
     - SVM-MAP relaxes MAP metric by adding to SVM constraints
     - AdaRank uses boosting to optimize NDCG

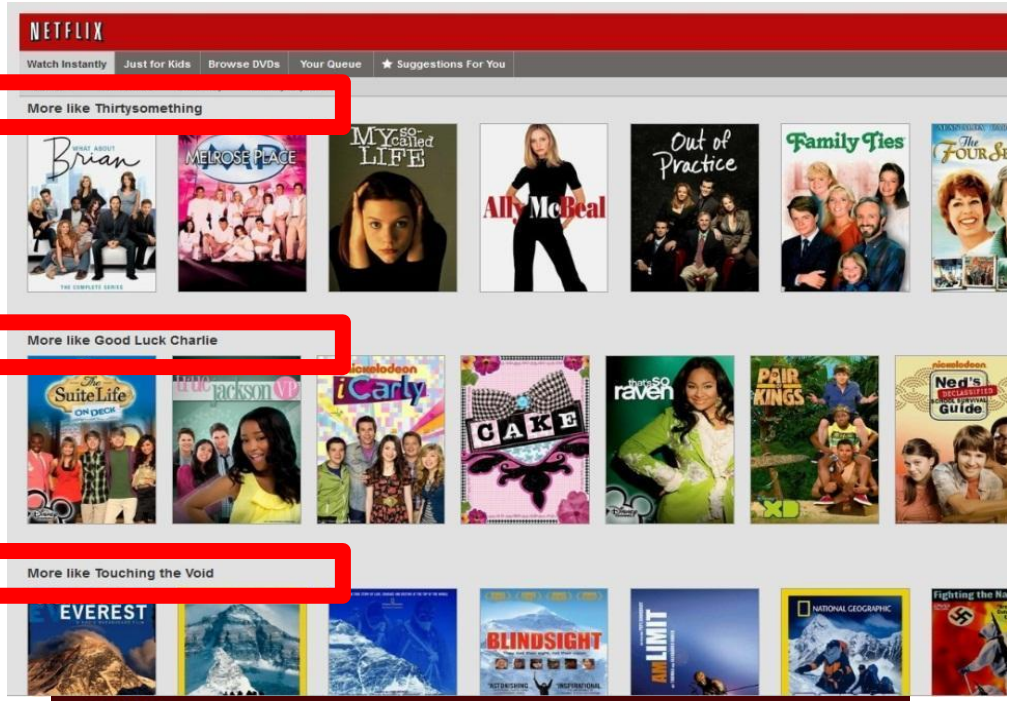
Index

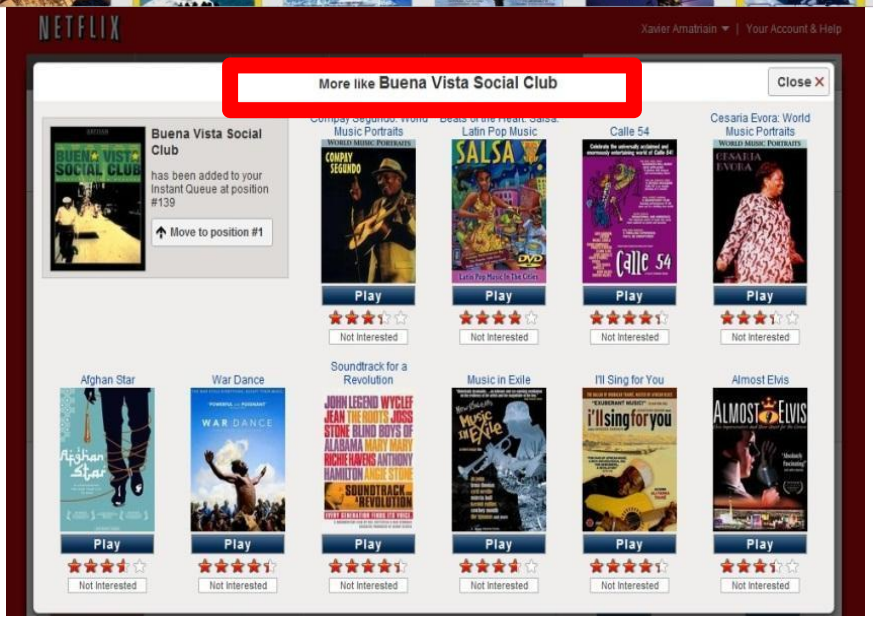
NETFLIX

3.2 Similarity as Recommendation

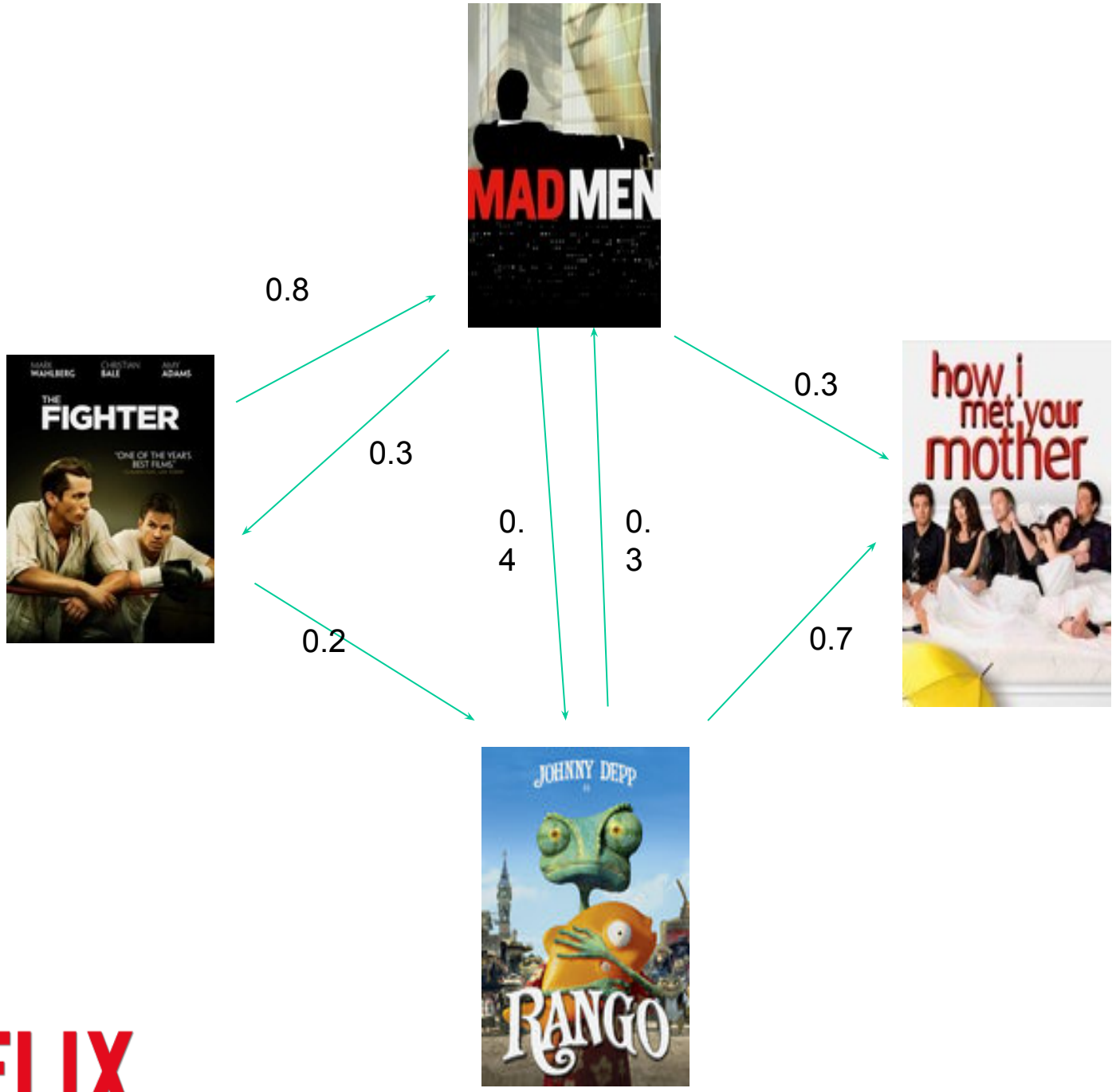
Similars

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





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.”

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

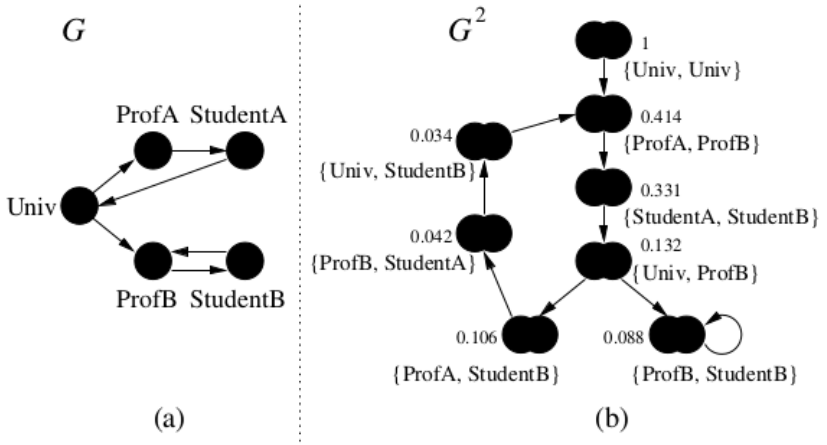


Figure 1: A small Web graph G and simplified node-pairs graph G^2. SimRank scores using parameter $C = 0.8$ are shown for nodes in G^2.

Similarity ensembles

- Similarity can refer to different dimensions
  - Similar in metadata/tags
  - Similar in user play behavior
  - Similar in user rating behavior
  - …
- 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

NETFLIX

Index

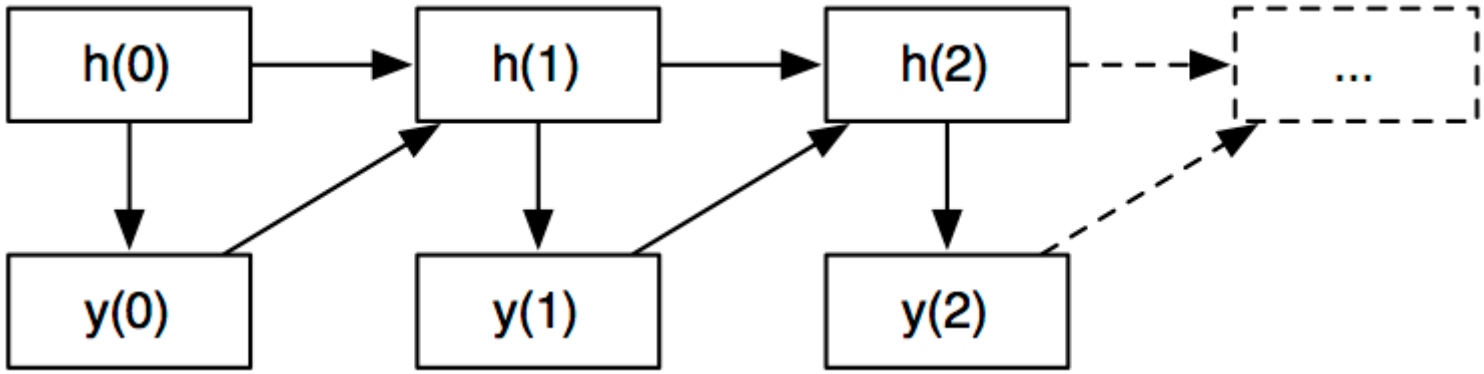
NETFLIX

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-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

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

NETFLIX

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:

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

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

- More details on the blogpost

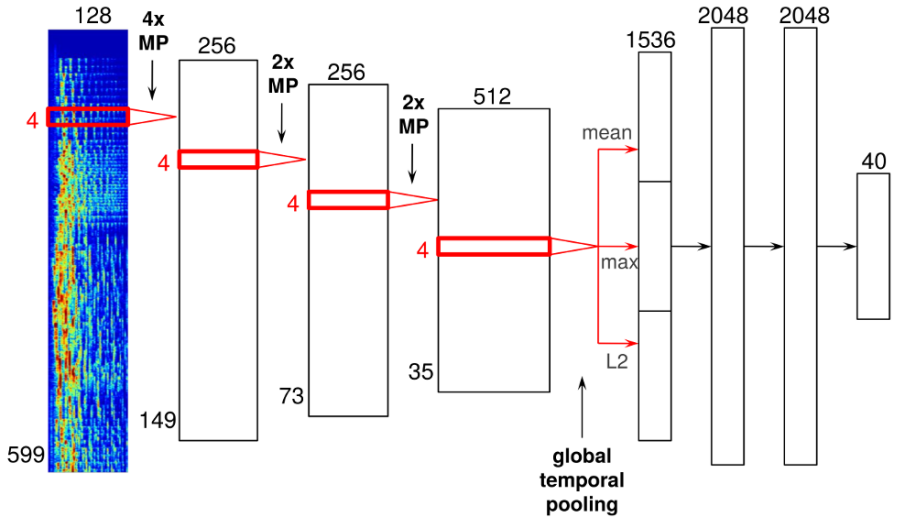
Deep Learning for Content-based Recommendations

- Another application of Deep Learning to recommendations also from Spotify
  - http://benanne.github.io/2014/08/05/spotify-cnns.html also *Deep content-based music recommendation, Aäron van den Oord, Sander Dieleman and Benjamin Schrauwen, NIPS 2013*
- Application to coldstart new titles when very little CF information is available
- Using mel-spectrograms from the audio signal as input
- Training the deep neural network to predict 40 latent factors coming from Spotify's CF solution

NETFLIX

Deep Learning for Content-based Recommendations

- Network architecture made of 4 convolutional layers + 4 fully connected dense layers

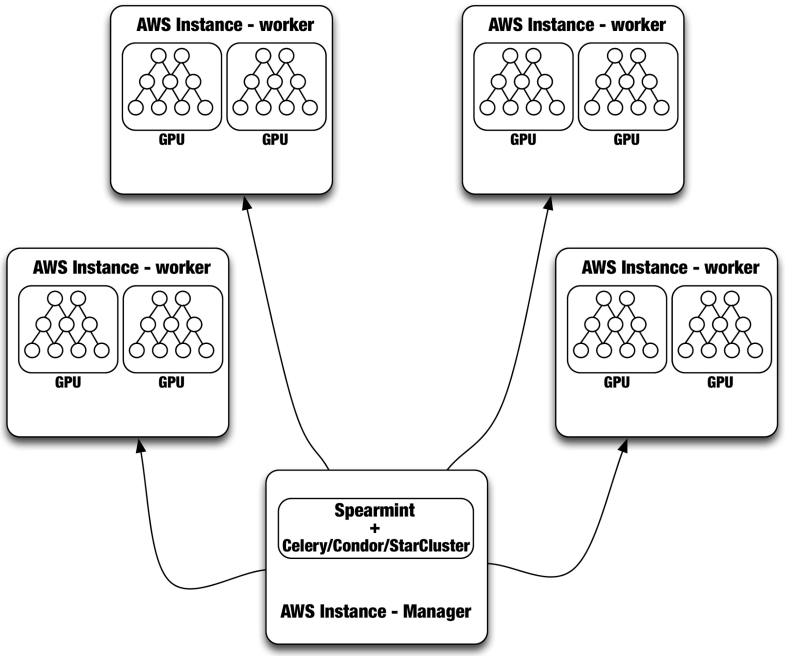


- One dimensional convolutional layers using RELUs (Rectified Linear Units) with activation max(0,x)
- **Max-pooling operations** between convolutional layers to downsample intermediate representations in time, and add time invariance
- **Global temporal pooling layer** after last convolutional layer: pools across entire time axis, computing statistics of the learned features across time:: mean, maximum and L2-norm
- Globally pooled features are fed into a series of **fully-connected layers** with 2048 RELUs

NETFLIX

ANN Training over GPUS and AWS

- How did we implement our ANN solution at Netflix?
  - 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



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

NETFLIX

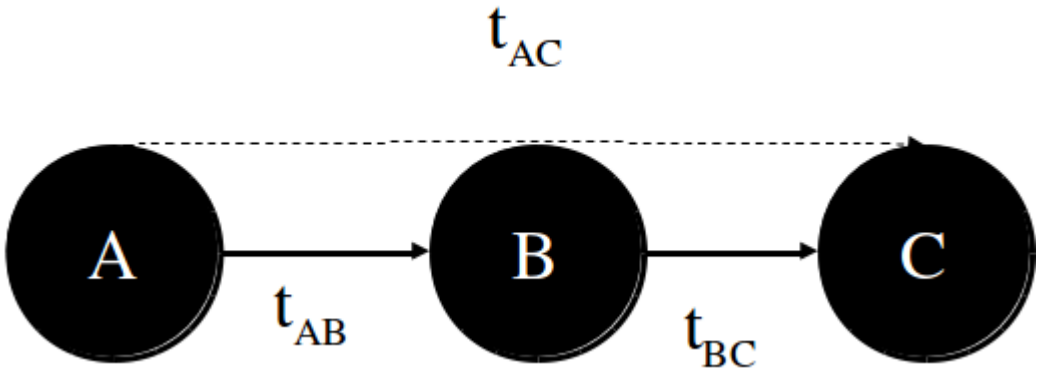
Index

NETFLIX

3.4 Social Recommendations

Social and Trust-based recommenders

- A social recommender system recommends items that are “popular” in the social proximity of the user.
- Social proximity = trust (can also be topic-specific)
- Given two individuals - the *source* (node A) and *sink* (node C) - derive how much the source should trust the sink.
- Algorithms
  - Advogato (Levien)
  - Appleseed (Ziegler and Lausen)
  - MoleTrust (Massa and Avesani)
  - TidalTrust (Golbeck)



NETFLIX

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
  - replace or modify user-user “similarity” by using social network information
  - use social connection as a part of the ML objective function as regularizer
  - …

NETFLIX

Demographic Methods

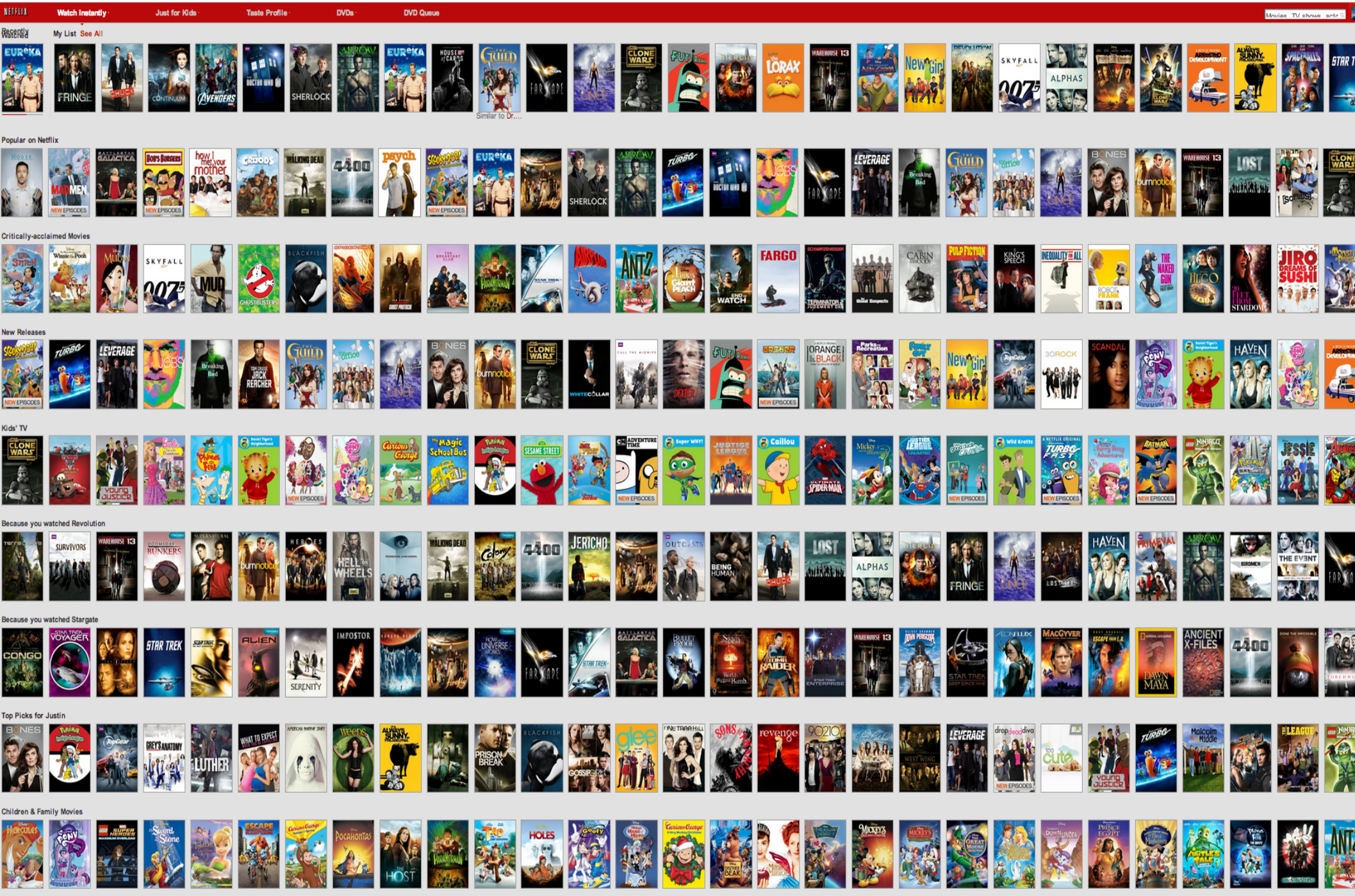
- Aim to categorize the user based on personal attributes and make recommendation based on demographic classes
- Demographic groups can come from marketing research – hence experts decided how to model the users
- Demographic techniques form people-to-people correlations

Index

NETFLIX

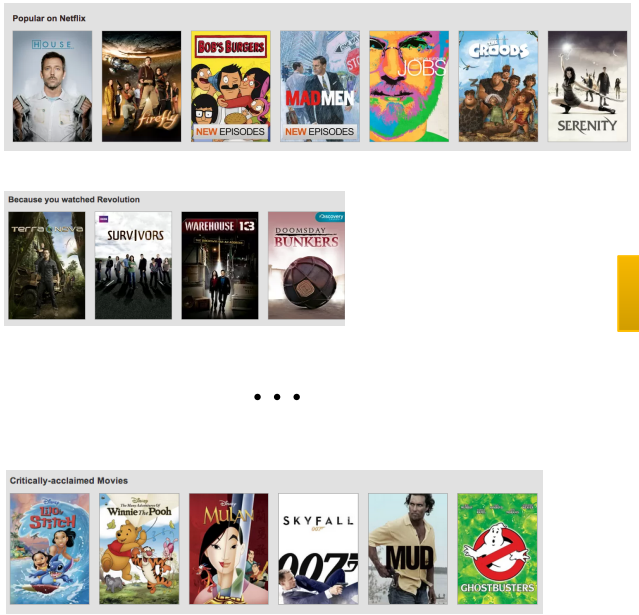
3.5. Page Optimization

Page Composition



Page Composition

10,000s of possible rows



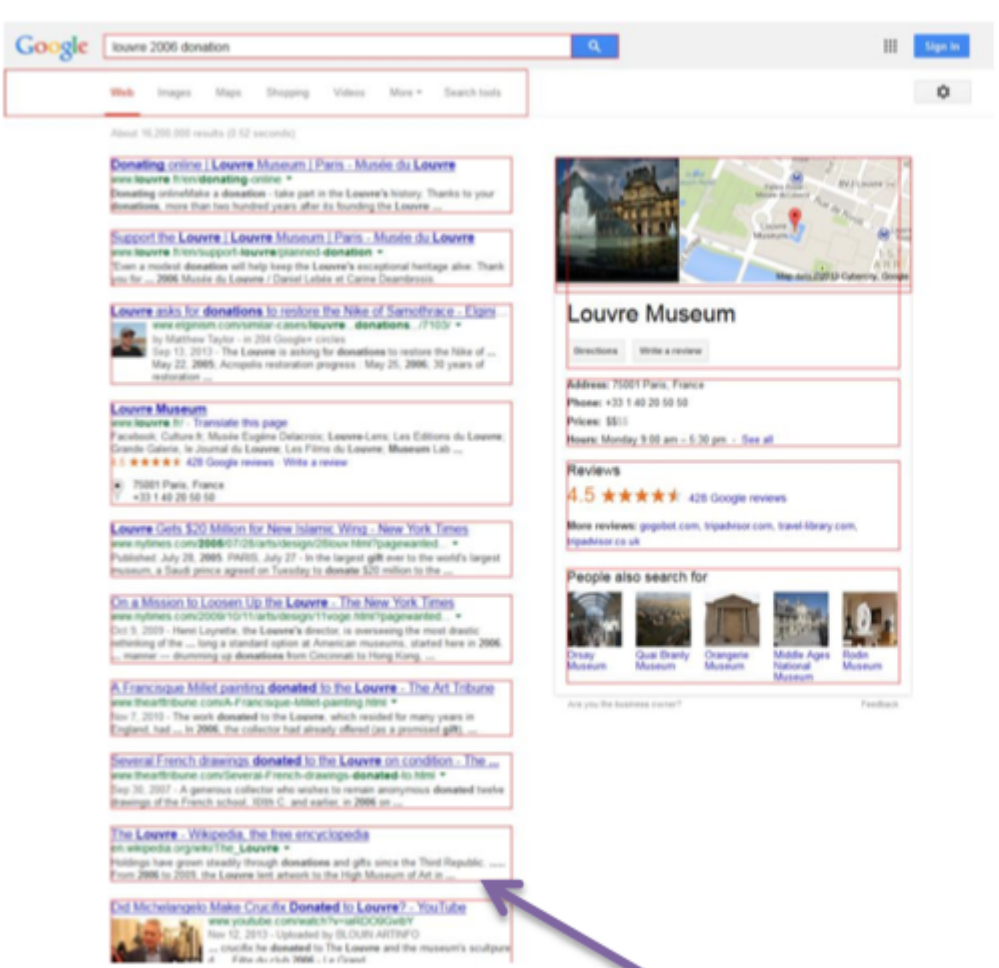
Variable number of possible videos per row (up to thousands)

1 personalized page

10-40 rows

per device

Page Composition



From “Modeling User Attention and Interaction on the Web” 2014 - PhD Thesis by Dmitry Lagun (Emory U.)

NETFLIX

User Attention Modeling

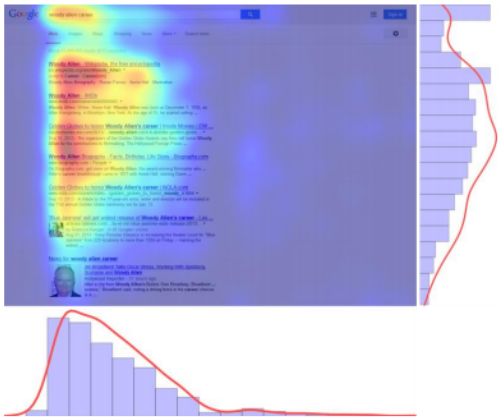
More likely to see

Less likely

NETFLIX

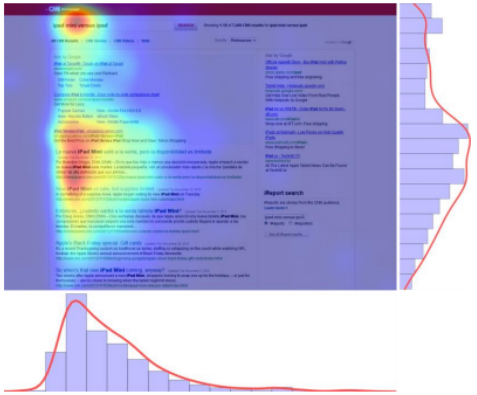
User Attention Modeling

Web Search (Google)



Social Network (Twitter)

News (CNN)



Shopping (Amazon)

From “Modeling User Attention and Interaction on the Web” 2014 - PhD Thesis by Dmitry Lagun (Emory U.)

Page Composition

Accurate vs. Diverse
Discovery vs. Continuation
Depth vs. Coverage
Freshness vs. Stability
Recommendations vs. Tasks

- To put things together we need to combine different elements
  - Navigational/Attention Model
  - Personalized Relevance Model
  - Diversity Model

Fair and Balanced: Learning to Present News Stories

Amr Ahmed*[1], Choon Hui Teo*[1], S.V. N. Vishwanathan[2], Alex Smola[1]
* Co-first authors.
[1]Yahoo! Research, Santa Clara, CA 95053, USA
[2]Purdue University, West Lafayette, IN 47907, USA
{amahmed,choonhui,smola}@yahoo-inc.com, vishy@stat.purdue.edu

NETFLIX

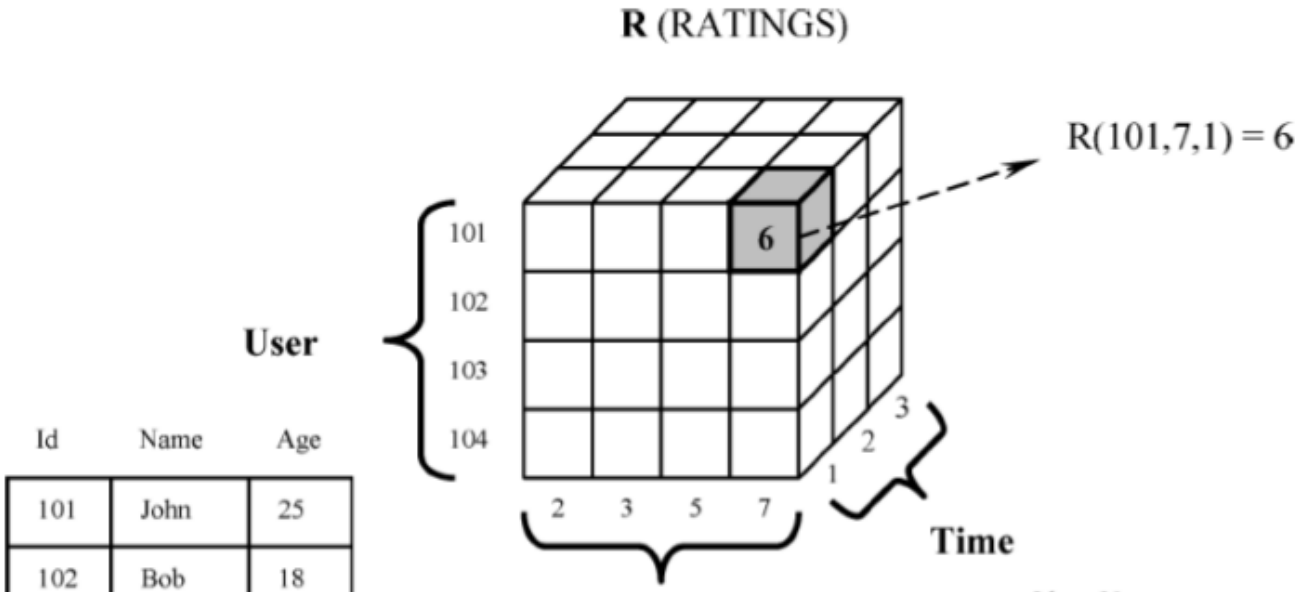
Index

NETFLIX

3.6 Tensor Factorization & Factorization Machines

NETFLIX

N-dimensional model



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

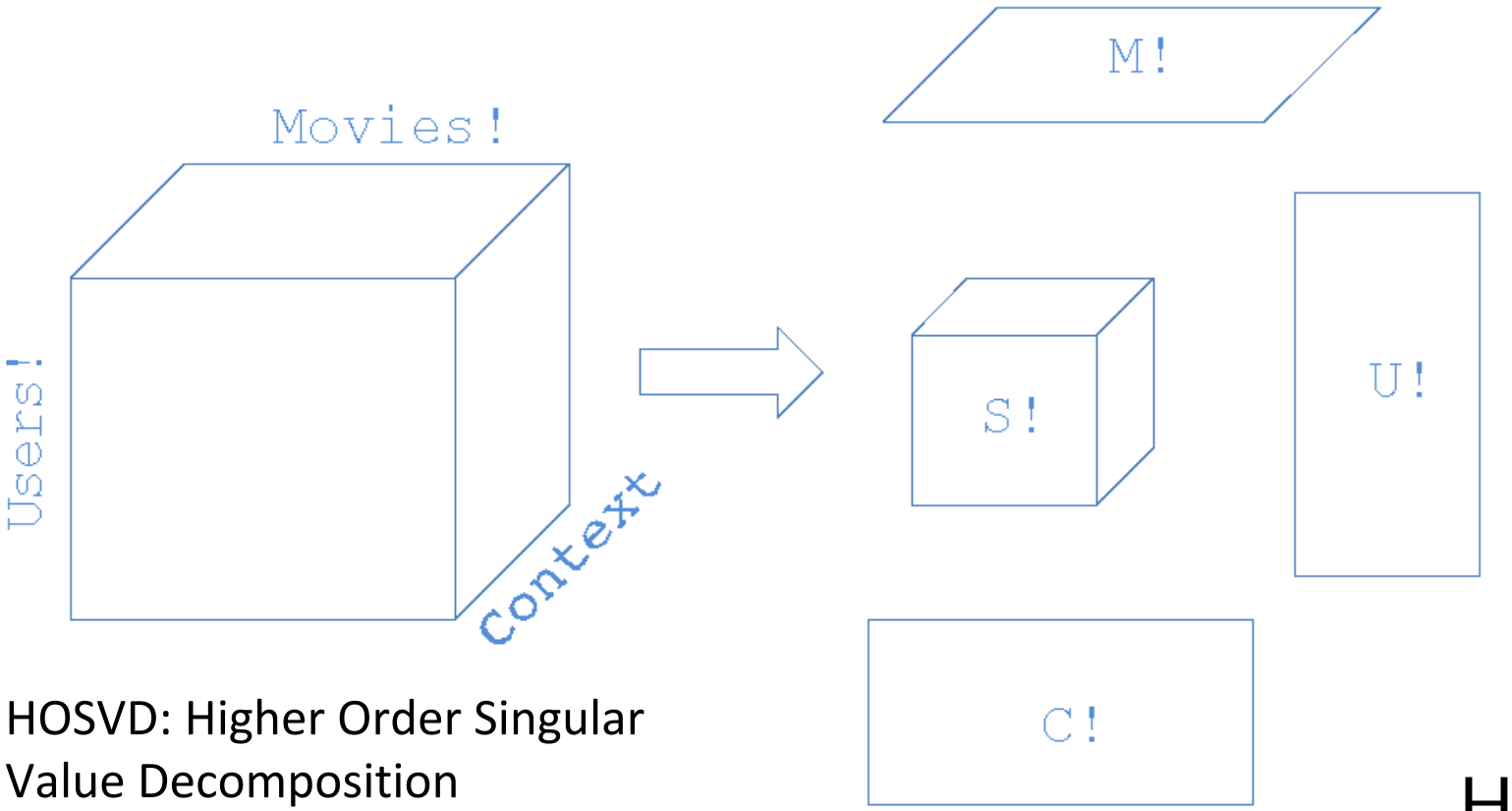
Item

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]

Tensor Factorization



HOSVD: Higher Order Singular Value Decomposition

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

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

HOSVD Model

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

NETFLIX

Tensor Factorization

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

Where:

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

- We can use a simple squared error loss function:

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

- Or the absolute error loss

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

- The loss function over all users becomes

$$L(F, Y) = \sum_i^n \sum_j^m l(f_{ij}, y_{ij})$$

Factorization Machines

- 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

- 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 overfit
  - 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

- 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:

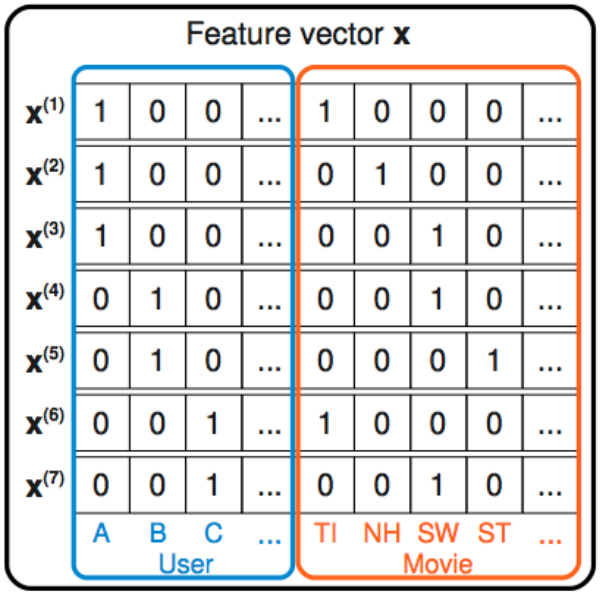
$$f(\mathbf{x}) = b + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} x_i x_j \mathbf{v}_i^T \mathbf{v}_j \qquad \text{O}(d^2)$$

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

NETFLIX

Factorization Machines

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



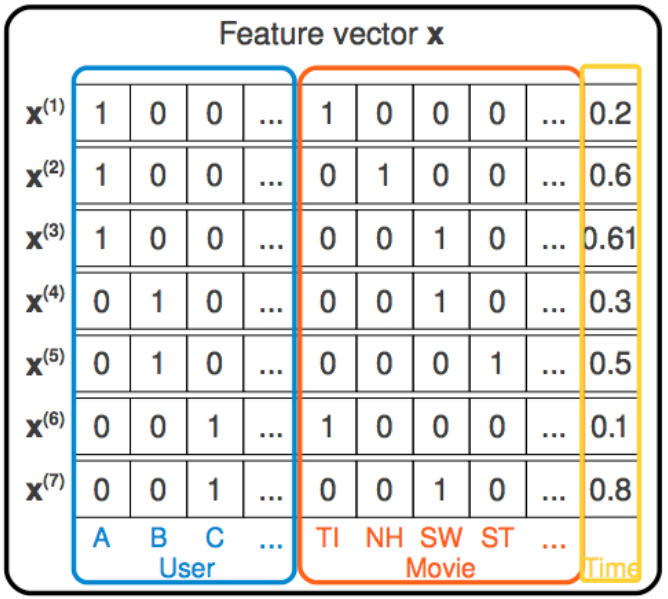
- FM becomes identical to MF with biases:

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

From Rendle (2012) KDD Tutorial

Factorization Machines

- 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

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

3.7 MAB Explore/Exploit

Explore/Exploit

- One of the key issues when building any kind of personalization algorithm is how to trade off:
  - **Exploitation**: Cashing in on what we know about the user right now
  - **Exploration**: Using the interaction as an opportunity to learn more about the user
- We need to have informed and optimal strategies to drive that tradeoff
  - **Solution**: pick a reasonable set of candidates and show users only "enough" to gather information on them

Multi-armed Bandits

- Given possible strategies/candidates (slot machines) pick the arm that has the maximum potential of *being good* (minimize **regret**)

- Naive strategy:
  - Explore with ε probability (e.g. 5%) -> choose an arm at random
  - Exploit with a high probability ($1-\varepsilon$) (e.g. 95%) -> choose the best-known arm so far
- Translation to recommender systems:
  - Choose an arm = choose an item/choose an algorithm (MAB testing)

$\varepsilon - greedy$

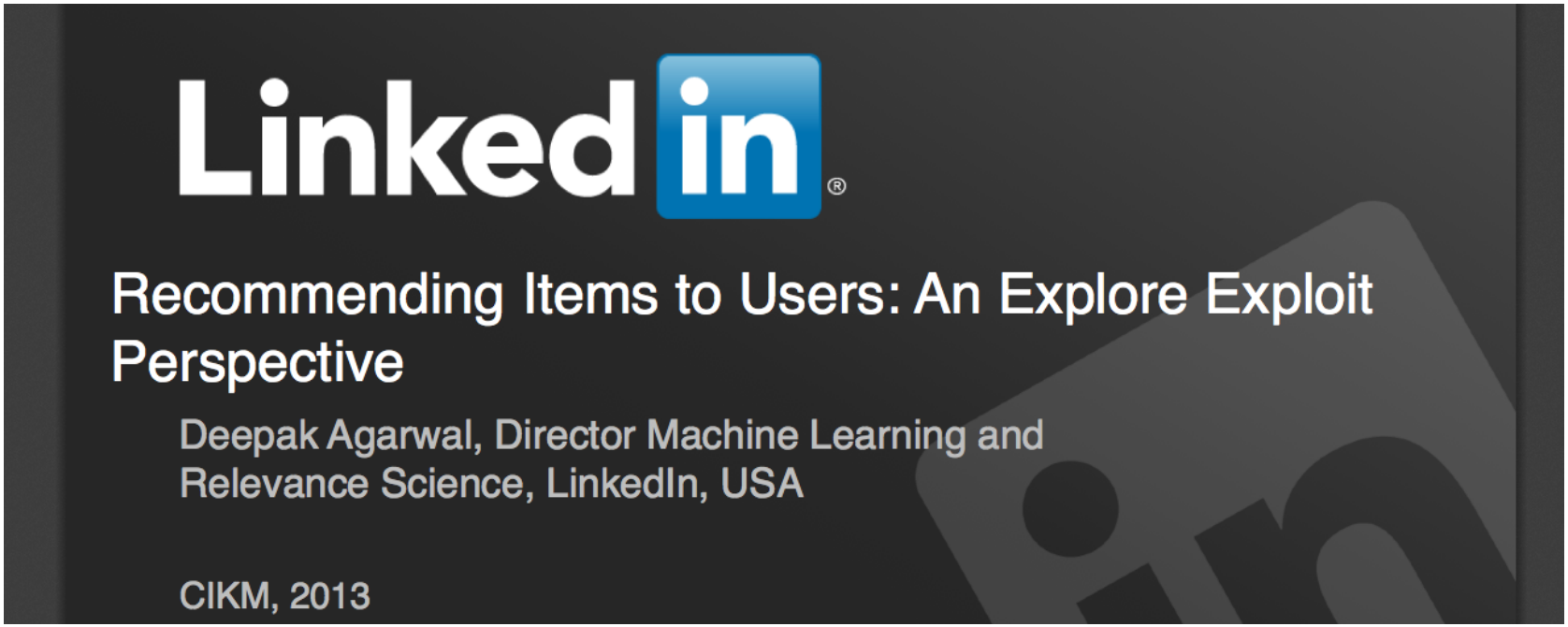
Multi-armed Bandits

- Better strategies not only take into account the mean of the posterior, but also the variance
- Upper Confidence Bound (UCB)
  - Show item with maximum score
  - Score = Posterior mean + Δ
  - Where Δ can be computed differently depending on the variant UCB (e.g. $\propto$ to the number of trials or the measured variance)
- Thompson Sampling
  - Given a posterior distribution
  - Sample on each iteration and choose the action that maximizes the expected reward

$$P(\theta|\mathcal{D}) \propto P(\mathcal{D}|\theta)P(\theta)$$

NETFLIX

Multi-armed Bandits



NETFLIX

Index

NETFLIX

4. 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.
- 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
- 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.
- C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. MSFT Technical Report

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.
- Dmitry Lagun. "*Modeling User Attention and Interaction on the Web*" 2014 PhD Thesis (Emory Un.)
- *"Robust Models of Mouse Movement on Dynamic Web Search Results* Pages - F. Diaz et al. In Proc. of CIKM 2013
- Amr Ahmed et al. "*Fair and balanced".* In Proc. WSDM 2013
- Deepak Agarwal. 2013. *Recommending Items to Users: An Explore Exploit Perspective*. CIKM '13

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!

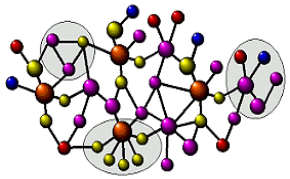
Questions?

Xavier Amatriain

xavier@netflix.com


@xamat

NETFLIX

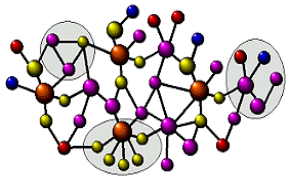


Context Aware Recommendation

Bamshad Mobasher

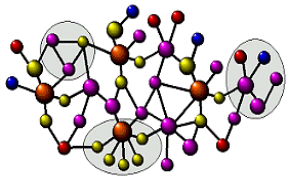
Center for Web Intelligence

School of Computing
DePaul University, Chicago



Outline

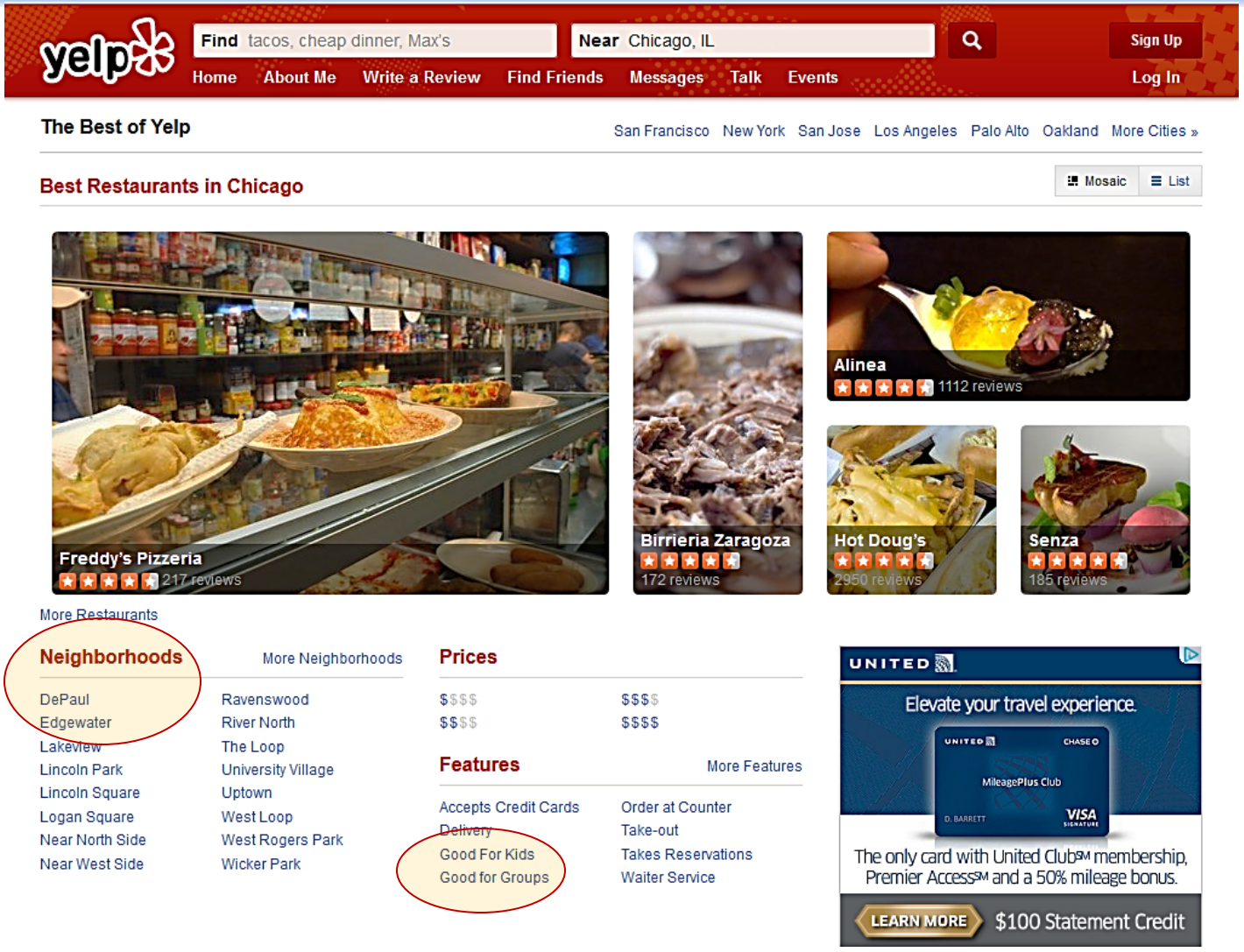
- **General views of context and their relevance to recommendation problem**
  - Key issues for context-aware system
  - Representational versus Interactional Views
- **Key Concepts in Contextual Aware Recommendation**
  - Architectures for integrating context in recommender systems
  - Highlighted Approaches in the Representational Framework
    - Item / User Splitting
    - Differential Contextual Modeling
    - Approaches based on Matrix Factorization
  - Interactional Context
    - Example Architecture: A Framework based on human memory
    - Highlighted Approach: Latent Variable Context Modeling



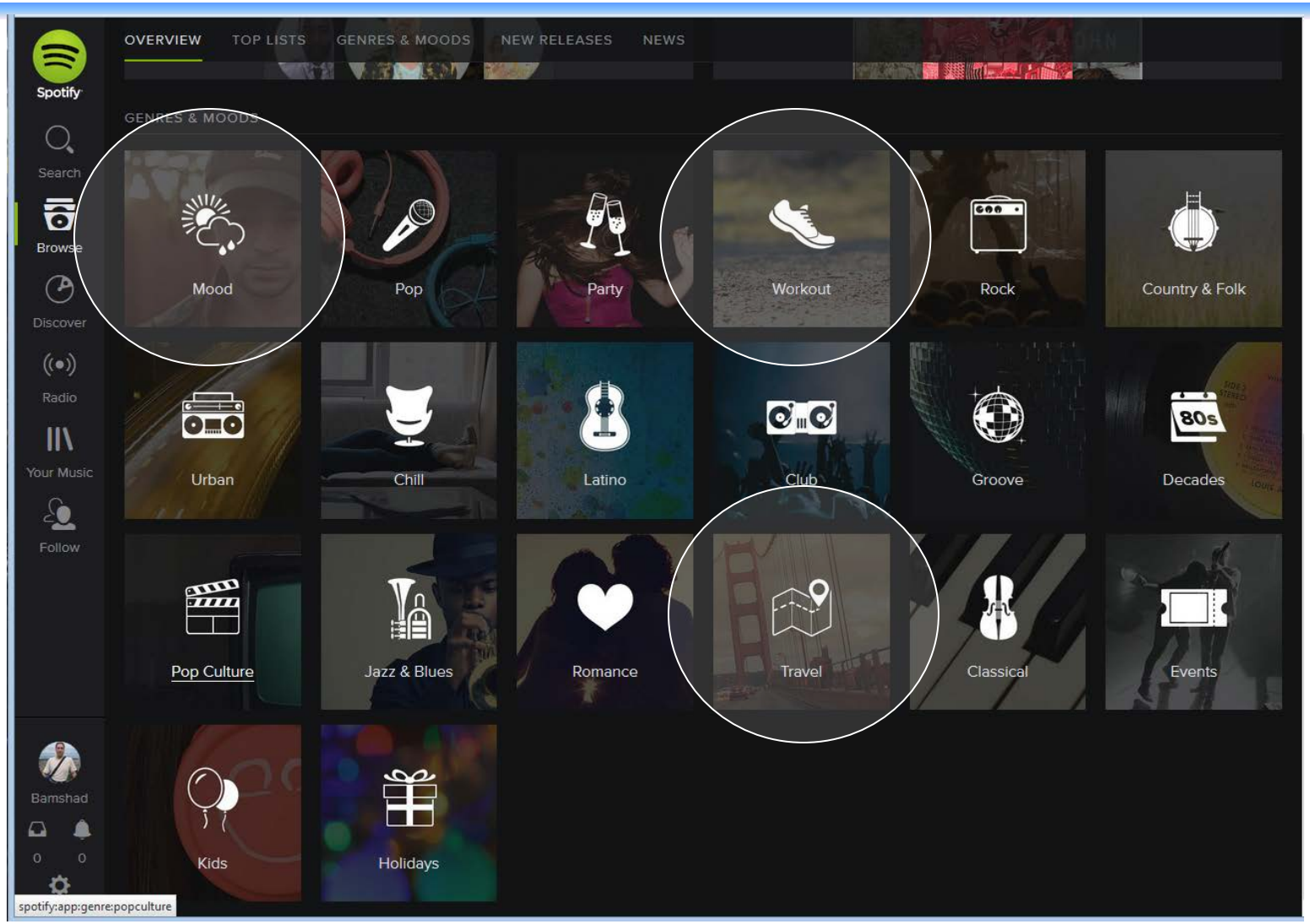
Context in Recommendation

- **Recommendation Scenario**
  - Dan's purchases on Amazon:
    - mystery-detective fiction "Da Vinci Code" (for himself)
    - "Python Programming" (for work)
    - "Green Eggs and Ham" (gift for his daughter)
  - How should we represent Dan's *interest in books*?
  - System needs to know the difference between children books and computer books, i.e., the contexts in which Dan interacts with the system
  - What should be recommended if Dan is reading reviews for a book on Perl Scripting?

Context in Recommendation



Context in Recommendation



Types of Context

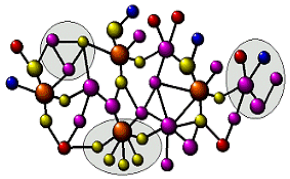
[Fling, 2009]

- **Physical context**
  - time, position, and activity of the user, weather, light, and temperature ...
- **Social context**
  - the presence and role of other people around the user
- **Interaction media context**
  - the device used to access the system and the type of media that are browsed and personalized (text, music, images, movies, …)
- **Modal context**
  - The state of mind of the user, the user's goals, mood, experience, and cognitive capabilities.

What the system knows about context

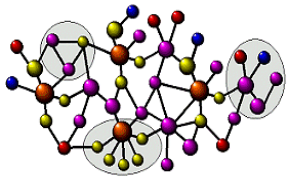
How Contextual Factors Change	Knowledge of the RS about the Contextual Factors		
	Fully Observable	Partially Observable	Unobservable
Static	Everything Known about Context	Partial and Static Context Knowledge	Latent Knowledge of Context
Dynamic	Context Relevance Is Dynamic	Partial and Dynamic Context Knowledge	Nothing Is Known about Context

See: Adomavicius, Mobasher, Ricci, and Tuzhilin. Context Aware Recommender Systems. *AI Magazine*, Fall 2011



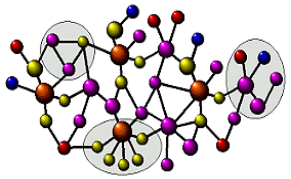
Defining Context

- **Entities interact with their environment through “situated actions”**
  - “Any information that can be used to characterize the situation of entities.” (Dey et al., 2001)
- **Context of an entity exist independently and outside of the entity’s actions**
  - Everything that affects computation except its explicit input and output.” (Lieberman and Selker, 2000)



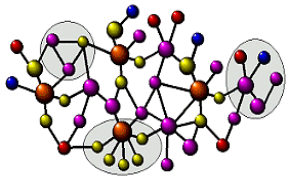
Different Views of Context

- **Paul Dourish (2004) distinguished between two views of context**
- **Representational view:**
  - Context is information that can be described using a set of “appropriate” attributes that can be observed and are distinguishable from features describing the underlying activity undertaken by the user within the context
- **Interactional View of Context**
  - The scope of contextual features is defined dynamically, and is occasioned rather than static
  - Rather than assuming that context defines the situation within which an activity occurs, there is a cyclical relationship between context and activity:
    - Context gives rise to the activity and activity changes the context



Representational View: Assumptions & Implications

- **Context can be represented as an explicit, enumerated set of static attributes (i.e., it's "extensional")**
  - Typically attributes are predefined based on the characteristics of the domain and environment
    - E.g., time, date, location, mood, task, device, etc.
  - Contextual variable can have associated structure
    - E.g., Sunday < Weekend
- **Implications:**
  - Must identify and acquire contextual information as part of data collection before actual recommendations are made
  - Relevant contextual variables (and their structures) must be identified at the design stage
- **Drawbacks**
  - The "qualification problem" – as in AI & Knowledge Representation
  - Context is static. No "situated action"

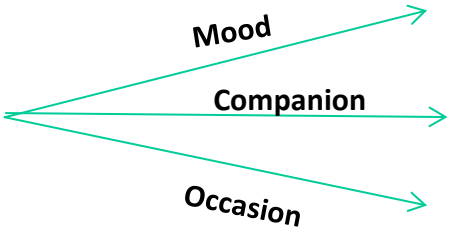


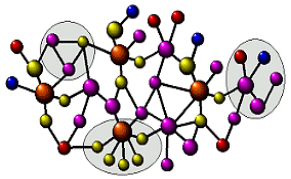
Interactional View: Assumptions & Implications

- **Properties of Context**
  - Context gives rise to a behaivor that is observable, though context itself may not be observable (it's "intensional")
    - Context exists (usually implicitly) in relation to the ongoing interaction of the user with the system
  - not static
    - Can be derived: a stochastic process with d states $\{c_1, c_2, \ldots, c_d\}$ representing different contextual conditions
- **Context aware recommendation**
  - Explicit representation of context may not be as important as
    - recognizing behavior arising from the context
    - adapting to the needs of the user within the context
- **Drawback:** Ability to explain recommendations

Context-Aware RS (CARS)

- **Traditional RS:** Users × Items → Ratings
- **Contextual RS**: Users × Items × Contexts → Ratings

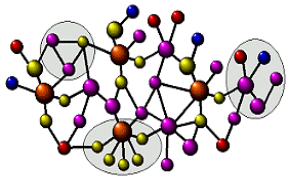




Example of Ratings Data in Representational Framework

User	Movie	Time	Location	Companion	Rating
U1	*Titanic*	Weekend	Home	Family	4
U2	*Titanic*	Weekday	Home	Family	5
U3	*Titanic*	Weekday	Cinema	Friend	4
U1	*Titanic*	Weekday	Home	Friend	?

- But what constitutes context:
  - <Weekday, Home, Friend>?
  - A subset of contextual variables, e.g., <Time, Location>?
- How do we select or weigh the most relevant set of variables

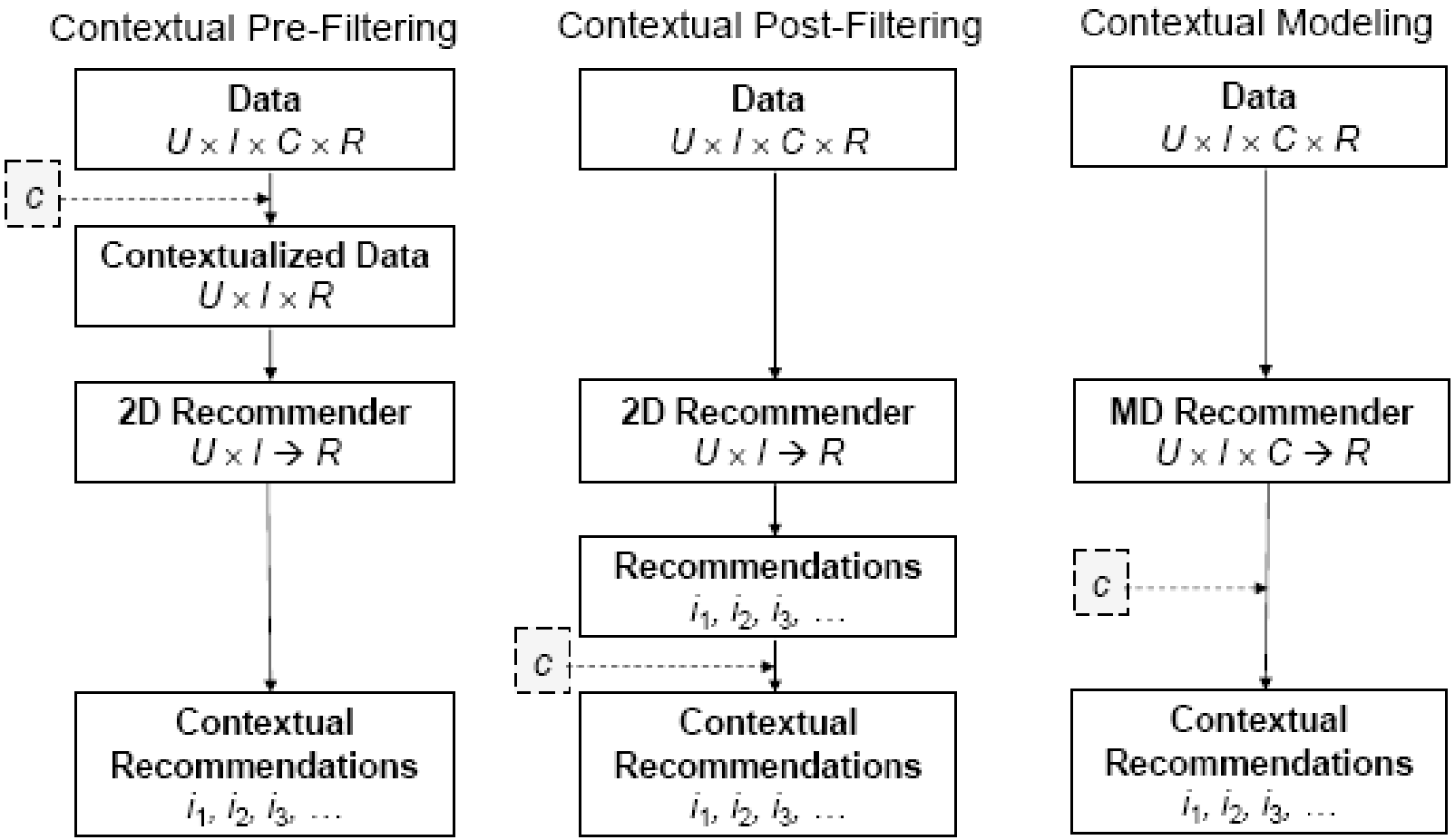


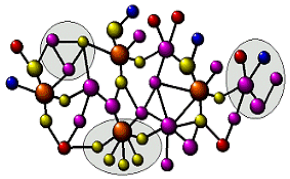
CARS Architectural Models

- **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**

CARS Architectural Models

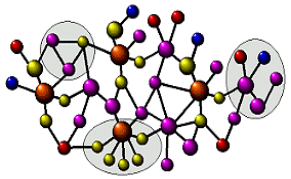
From Adomavicius, Tuzhilin, 2008





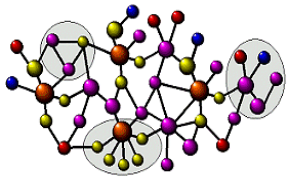
Contextual Pre-Filtering Challenges

- 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.)



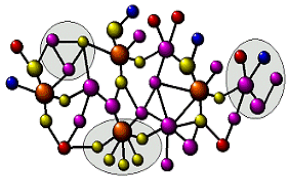
Contextual Post-Filtering

- 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*

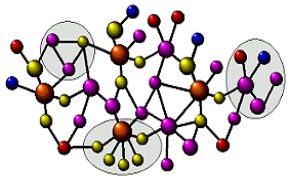


Contextual Recommendation Approaches

- Many different approaches in recent years, both in representational and interactional frameworks
  - Extensions of standard collaborative filtering
    - *CF after Item / user splitting pre-filters*
    - *Differential Context Modeling*
  - Heuristic distance-based approaches
    - Extend items-item, user-user similarities to contextual dimensions
    - Requires, possibly domain specific, similarity/distance metrics for various contextual dimensions
  - Approaches based on matrix/tensor factorization
    - Model the data as a tensor and apply higher-order factorization techniques (HoSVD, HyPLSA, etc) to model context in a latent space
    - *Context-Aware Matrix Factorization*
  - *Probabilistic latent variable context models*

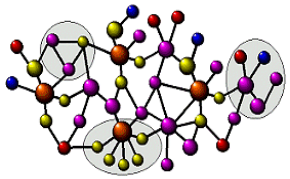


Highlighted Approach: Item / User Splitting



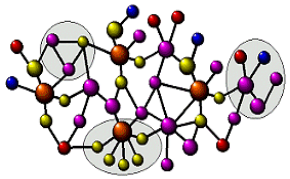
Context-Aware Splitting Approaches

- **Generally based on contextual pre-filtering**
  - May be combined with contextual modeling techniques
- **Goal: produce a 2D data set that incorporates context information associated with preference scores**
  - Advantage: can use a variety of well-known traditional recommendation algorithms in the modeling phase
  - Disadvantages:
    - Determining the variables based on which to split
    - May lead to too much sparsity
- **There are three approaches to splitting:**
  - Item Splitting (Baltrunas et al., 2009, RecSys)
  - User Splitting (Baltrunas et al., 2009, CARS)
  - UI Splitting (Zheng et al., 2013)



Item Splitting and User Splitting

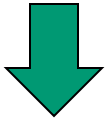
- **Item Splitting**
  - Assumption: the nature of an item, from the user's point of view, may change in different contextual conditions (values of contextual variables)
  - Hence we may consider the item as multiple items – one for each contextual condition
- **User splitting**
  - It may be useful to consider one user as multiple users, if he or she demonstrates significantly different preferences in different contexts
- **Good deal of recent work on these approaches:**
  - L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. RecSys 2009
  - L. Baltrunas and X. Amatriain. Towards time-dependent recommendation based on implicit feedback. RecSys 2009 Workshop on CARS
  - A. Said, E. W. De Luca, and S. Albayrak. Inferring contextual user profiles – improving recommender performance. RecSys 2011 Workshop on CARS



Example: Item Splitting

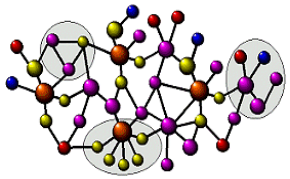
User	Movie	Rating	Time	Location	Companion
U1	M1	3	Weekend	Home	Friend
U1	M1	5	Weekend	Theater	Spouse
U1	M1	?	Weekday	Home	Family

Assume Location (Home vs. Theater) is the best split condition



User	Item	Rating
U1	M11	3
U1	M12	5
U1	M11	?

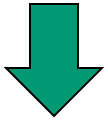
M11: M1 seen at home; M12 = M1 seen not at home



Example: User Splitting

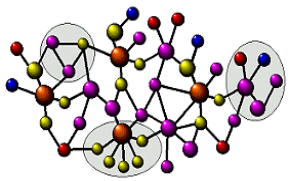
User	Movie	Rating	Time	Location	Companion
U1	M1	3	Weekend	Home	Friend
U1	M1	5	Weekend	Theater	Alone
U1	M1	?	Weekday	Home	Family

Assume Companion (Family vs. Non-Family) is the best split condition



User	Item	Rating
U12	M1	3
U12	M1	5
U11	M1	?

U11: U1 saw the movie with family; U12 = U1 saw the movie alone or with a friend



User-Item (UI) Splitting

- **New approach combining User and Item splitting**
  - The process is simply an application of item splitting followed by user splitting on the resulting output
  - Y. Zheng, B. Mobasher, R. Burke. Splitting approaches for Context-aware Recommendation. (To appear)
- **Using the same conditions as previous example:**

Item Splitting

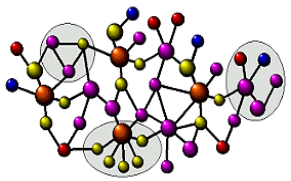
User	Item	Rating
U1	M11	3
U1	M12	5
U1	M11	?

User Splitting

User	Item	Rating
U12	M1	3
U12	M1	5
U11	M1	?

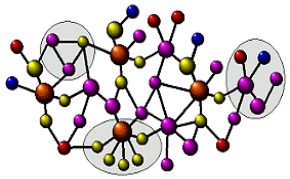
UI Splitting

User	Item	Rating
U12	M11	3
U12	M12	5
U11	M11	?



Determining the Best Conditions for Splitting

- **Impurity Criteria**
  - Statistical criteria to evaluate whether items are being rated significantly differently under an alternative contextual condition
  - e.g. the location → home vs. not-home
- **Commonly used criteria**
  - t_{mean} (t-test)
  - t_{prop} (z-test)
  - t_{chi} (chi-square test)
  - t_{IG} (Information gain)
- **Thresholds:**
  - P-value is used to judge significance



Splitting Criteria

- **Example: t_{mean}**
  - Uses the two-sample *t* test and computes how significantly different are the means of the ratings in the two rating subsets, when the split *c* (context condition) is used
  - $$t_{mean} = \left| \frac{\mu_{i_c} - \mu_{i_{\bar{c}}}}{\sqrt{s_{i_c}/n_{i_c} + s_{i_{\bar{c}}}/n_{i_{\bar{c}}}}} \right|$$
  - *S* is the rating variance, and *n* is the number of ratings in the given contextual condition, *c* and $\bar{c}$ denote alternative conditions
  - The bigger the *t* value of the test is, the more likely the difference of the means in the two partitions is significant
  - This process is iterated over all contextual conditions

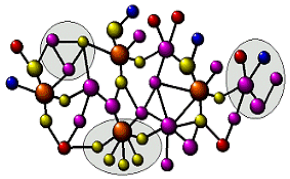
Splitting Criteria Example

- **t_{mean} ; condition: time= weekend and not weekend**

$$t_{mean} = \left| \frac{\mu_{i_c} - \mu_{i_{\bar{c}}}}{\sqrt{s_{i_c}/n_{i_c} + s_{i_{\bar{c}}}/n_{i_{\bar{c}}}}} \right|$$

User	Item	Rating	Time	Location	Companion
U1	T1	3	Weekend	Home	Sister
U1	T1	5	Weekend	Cinema	Girlfriend
U2	T1	4	Weekday	Home	Family
U3	T1	2	Weekday	Home	Sister

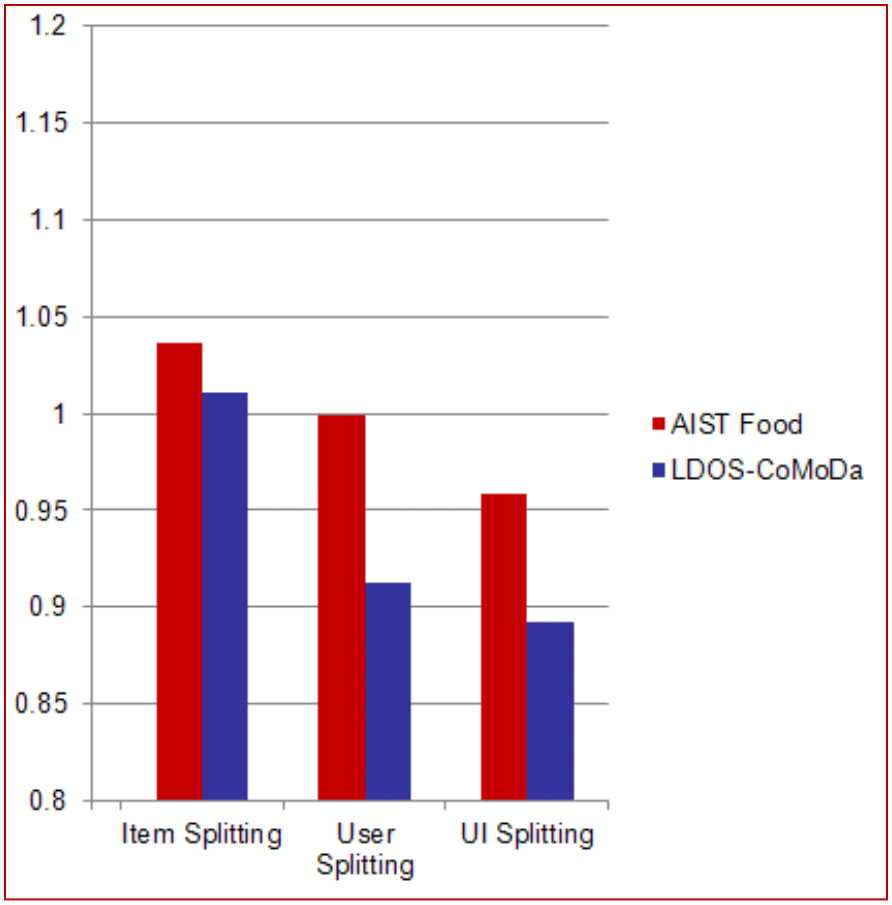
- **mean1=4 , mean2 =3 , s1 =1 , s2 =1 , n1= 2, n2=2**
- **Impurity criteria t_{mean} = (4-3)/1 = 1**
- **P-value of t-test used to determine significance (0.05 as threshold)**

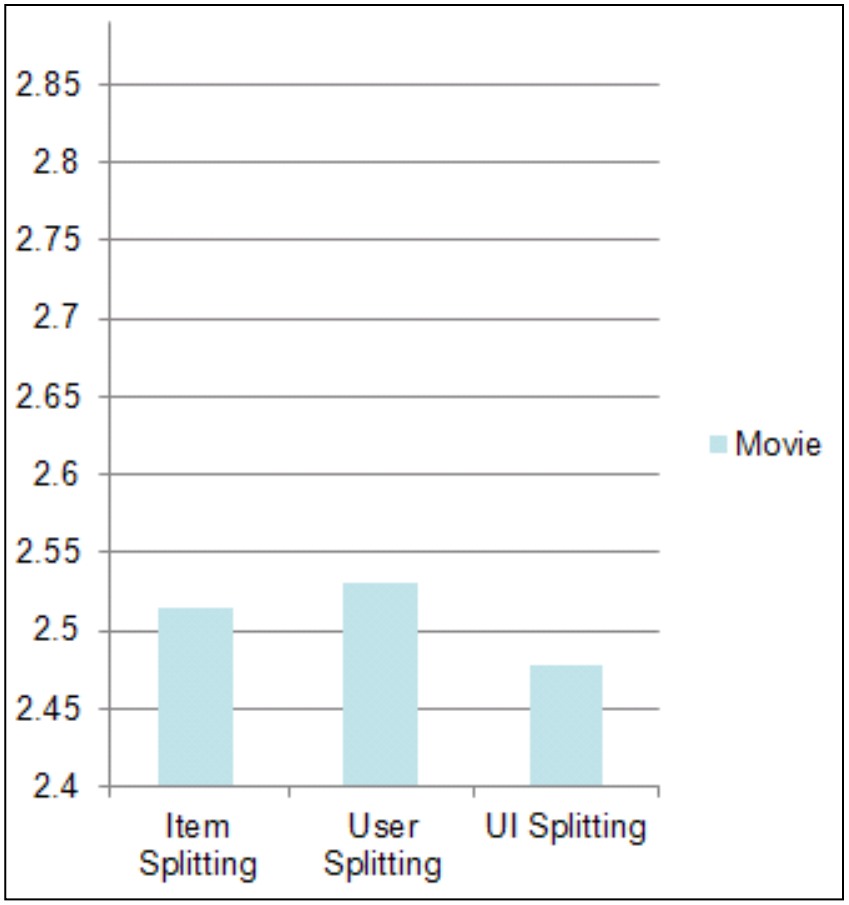


Example Results

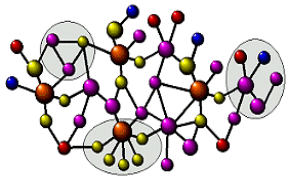
	Food Data	Movie Data
Ratings	6360	1010
Users	212	69
Items	20	176
Contexts	Real hunger (full/normal/hungry) Virtual hunger	Time (weekend, weekday) Location (home, cinema) Companions (friends, alone, etc)
Contexts-linked Features	User gender food genre, food style, food stuff	User gender, year of the movie
Density	Dense in contexts	Sparse in contexts

Example Results (RMSE)

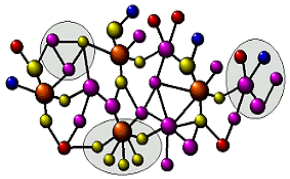




Results based on splitting followed by Matrix Factorization (discussed next)



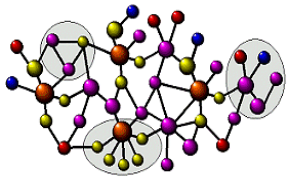
Highlighted Approach: Differential Context Modeling



Differential Context Modeling

User	Movie	Time	Location	Companion	Rating
U1	*Titanic*	Weekend	Home	Family	4
U2	*Titanic*	Weekday	Home	Family	5
U3	*Titanic*	Weekday	Cinema	Friend	4
U1	*Titanic*	Weekday	Home	Friend	?

- Context Matching ➔ only the exact context <Weekday, Home, Friend>?
- Context Selection ➔ use only the most relevant contexts
- Context Relaxation ➔ relax set of constraints (dimensions) defining the context, e.g. use only time
- Context Weighting ➔ use all dimensions, but weight them according to co-occurrence relationships among contexts

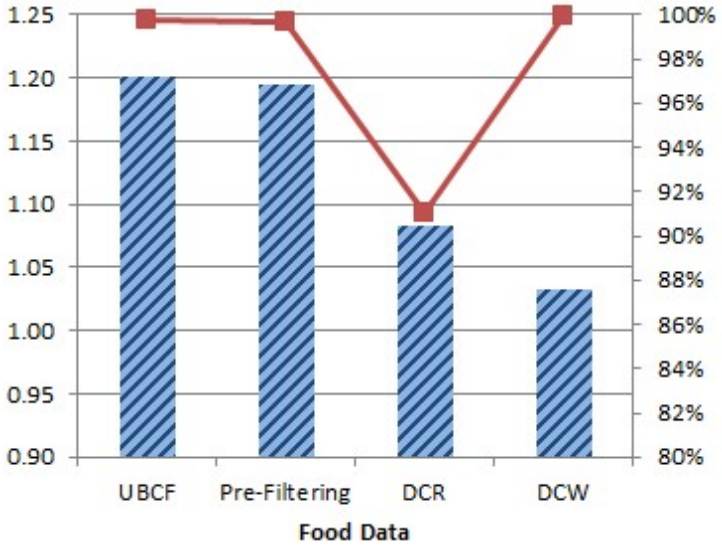


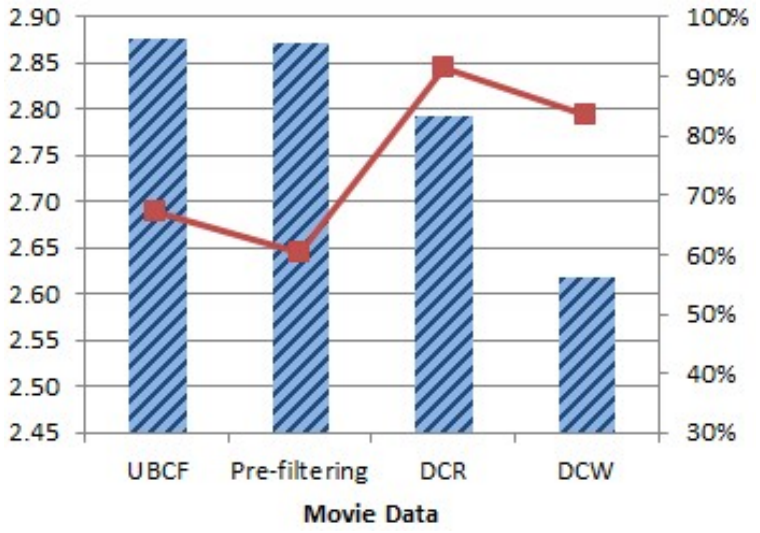
Differential Context Weighting

User	Movie	Time	Location	Companion	Rating
U1	*Titanic*	Weekend	Home	Friend	4
U2	*Titanic*	Weekday	Home	Friend	5
U3	*Titanic*	Weekday	Cinema	Family	4
U1	*Titanic*	Weekday	Home	Family	?

- **Goal: Use all dimensions, but weight them based on the similarity of contexts**
  - Assumption: the more similar two contexts are, the more similar the ratings will be in those contexts
  - Similarity can be measured by Weighted Jaccard similarity $J(c,d,\sigma) = \frac{\sum_{f \in c \cap d} \sigma_f}{\sum_{f \in c \cup d} \sigma_f}$
  - Example:
    - c and d are two contexts (two red regions in the Table)
    - σ is the weighting vector <w1, w2, w3> for three dimensions.
    - Assume they are equal weights, w1 = w2 = w3 = 1
    - J(c, d, σ) = # of matched dimensions / # of all dimensions = 2/3

Predictive Performance

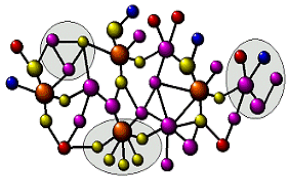




Blue bars are RMSE values, ***Red lines*** are coverage curves

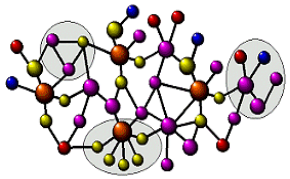
Findings:

1) t-test shows DCW works better significantly in movie data
2) DCW can further alleviate sparsity of contexts
3) DCW offers better coverage over baselines!



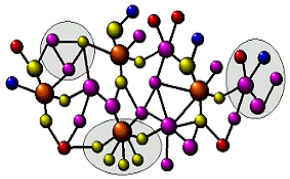
Differential Context Modeling

- **Some relevant work**
  - Y. Zheng, R. Burke, B. Mobasher. "Differential Context Relaxation for Context-aware Travel Recommendation". In EC-WEB, 2012 [DCR]
  - Y. Zheng, R. Burke, B. Mobasher. "Optimal Feature Selection for Context-Aware Recommendation using Differential Relaxation". In ACM RecSys Workshop on CARS, 2012 [DCR + Optimizer]
  - Y. Zheng, R. Burke, B. Mobasher. "Recommendation with Differential Context Weighting". In UMAP, 2013 [DCW]
  - Y. Zheng, R. Burke, B. Mobasher. “Differential Context Modeling in Collaborative Filtering". In SOCRS-2013, DePaul University, Chicago, IL, May 31, 2013 [DCM]
- **Future Work**
  - Try other similarity measures of contexts instead of the simple Jaccard
  - Introduce semantics into the similarity of contexts to further alleviate the sparsity of contexts, e.g., Rome is closer to Florence than Paris
  - Parallel PSO to speed up optimizer
  - Additional recommendation algorithms on DCR or DCW



Highlighted Approach:

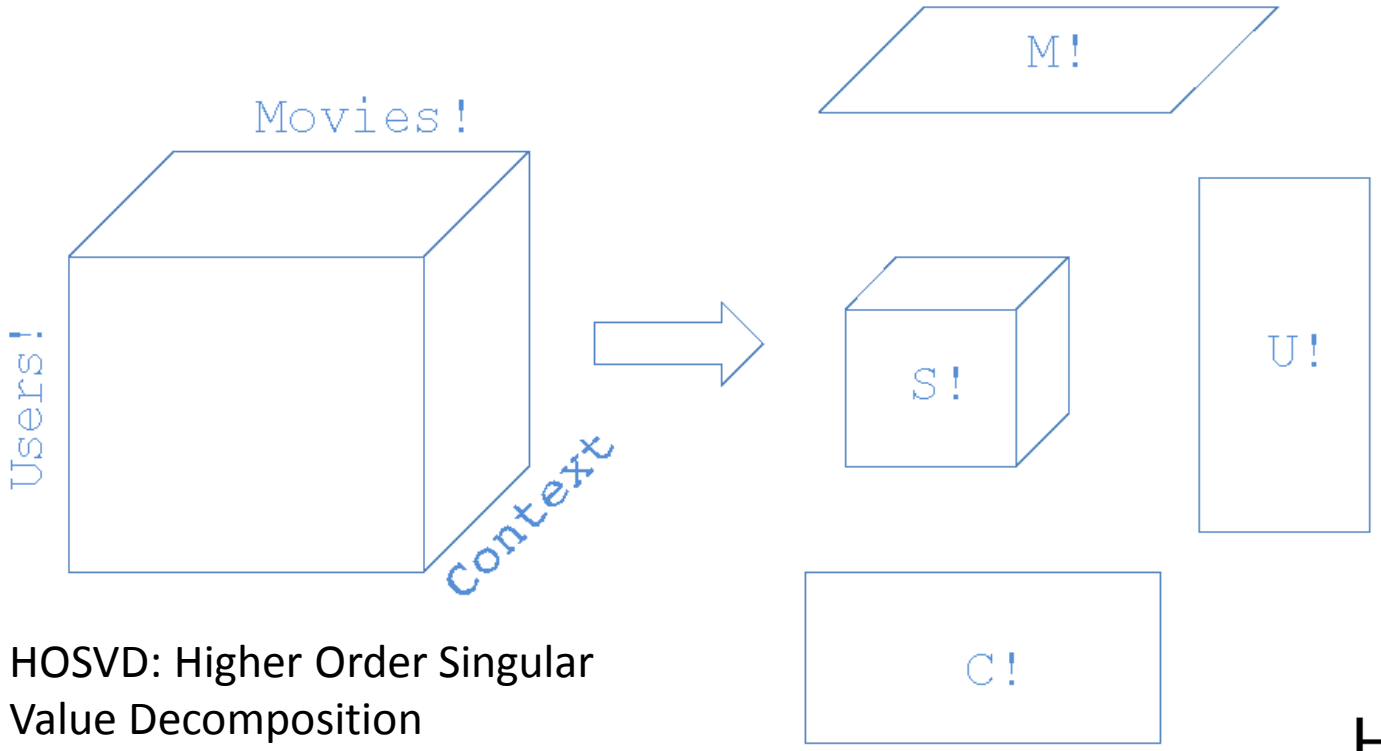
Contextual Modeling using Matrix Factorization



Contextual Modeling via Factorization

- Recent approaches to contextual modeling attempt to fit the data using various regression models
  - Prominent examples: Tensor Factorization (TF), Factorization Machines (FM)
- Tensor Factorization
  - Extends the two-dimensional matrix factorization problem into an multi-dimensional version of the same problem
  - Multi-dimensional matrix is factored into lower-dimensional representation, where the user, the item and each contextual dimension are represented with a lower dimensional feature vector
- TF can introduce a huge number of model parameters that must be learned using the training data
  - the number of model parameters grow exponentially with the number of contextual factors

Recall: Tensor Factorization



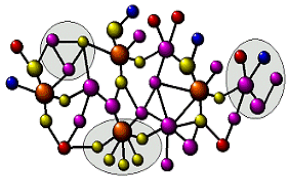
HOSVD: Higher Order Singular Value Decomposition

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

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

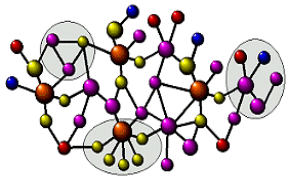
HOSVD Model

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



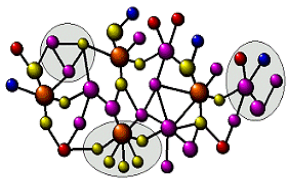
Context-Aware Matrix Factorization

- **Recall the difference between MF and BiasMF:**
  - Standard MF: Predict(u, i) = $q_i^T p_u$
  - BiasMF: Predict(u, i) = $q_i^T p_u + \mu + b_u + b_i$
  - b_u and b_i are user bias and item bias respectively, μ is the global mean rating
- **Context-Aware MF**
  - CAMF replaces the simple item bias, b_i, by the interaction between item and contextual conditions
    - Predicted rating will now be also a function of contextual conditions, c_1, …, c_k giving rise to a particular context
    - The item bias is modeled by how that item is rated in different contexts
    - i.e., sum of biases for the given item across all contextual conditions, c_1, …, c_k
    - Different levels of granularity in aggregating item biases can lead to different variants of CAMF



Recall: Factorization Machines

- **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 overfit
  - 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”**



Recall: Factorization Machines

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

Feature vector **x**

$x^{(1)}$	1	0	0	...	1	0	0	0	...
$x^{(2)}$	1	0	0	...	0	1	0	0	...
$x^{(3)}$	1	0	0	...	0	0	1	0	...
$x^{(4)}$	0	1	0	...	0	0	1	0	...
$x^{(5)}$	0	1	0	...	0	0	0	1	...
$x^{(6)}$	0	0	1	...	1	0	0	0	...
$x^{(7)}$	0	0	1	...	0	0	1	0	...
	A	B	C	...	TI	NH	SW	ST	...
	User				Movie				

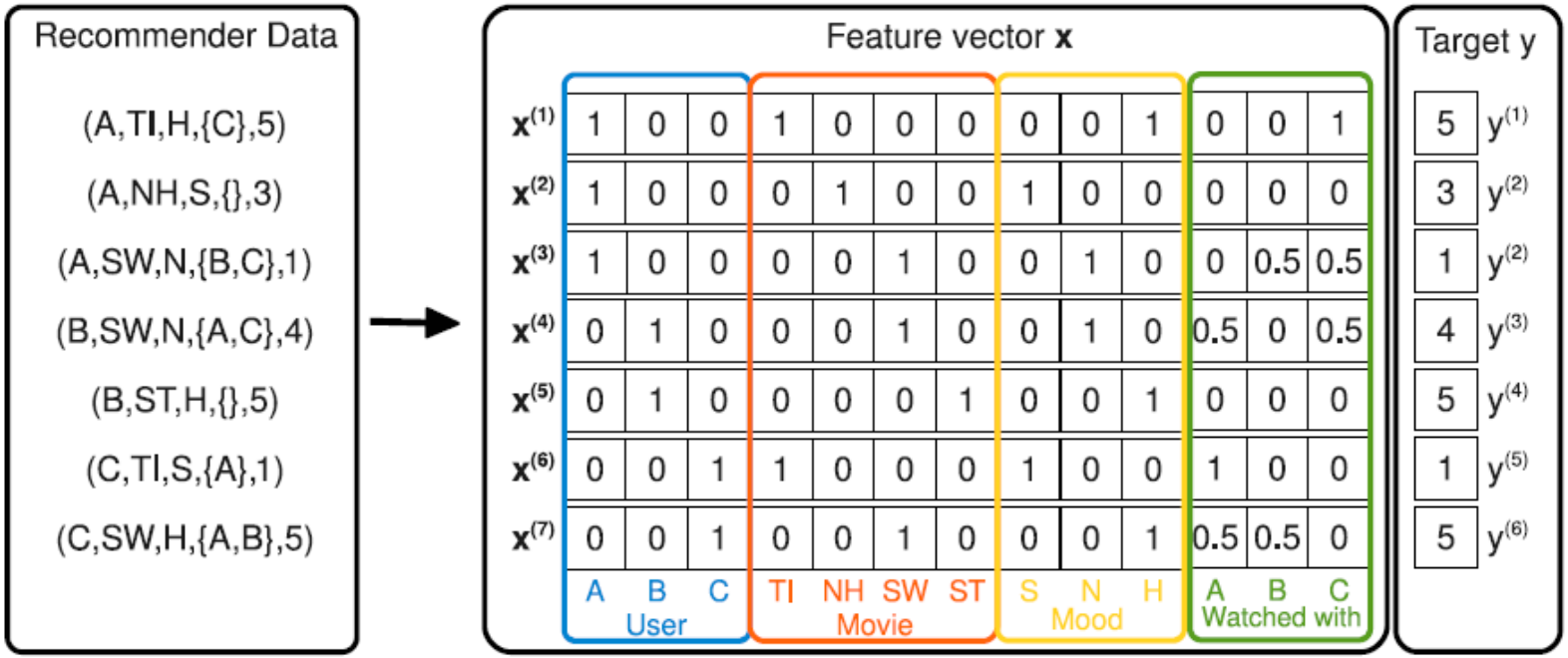
- **FM becomes identical to MF with biases:**

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

From Rendle (2012) KDD Tutorial

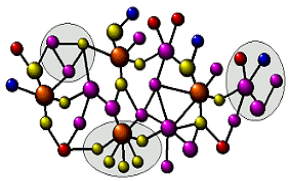
Recall: Factorization Machines

- **Can easily map different factorization models in FM:**



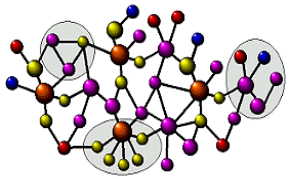
Context-aware recommendation data (left side) is transformed into a prediction problem from real-valued features by encoding the categorical and set categorical variables.

From Rendle SIGIR, 2011



Some Additional References for MF Based Approaches

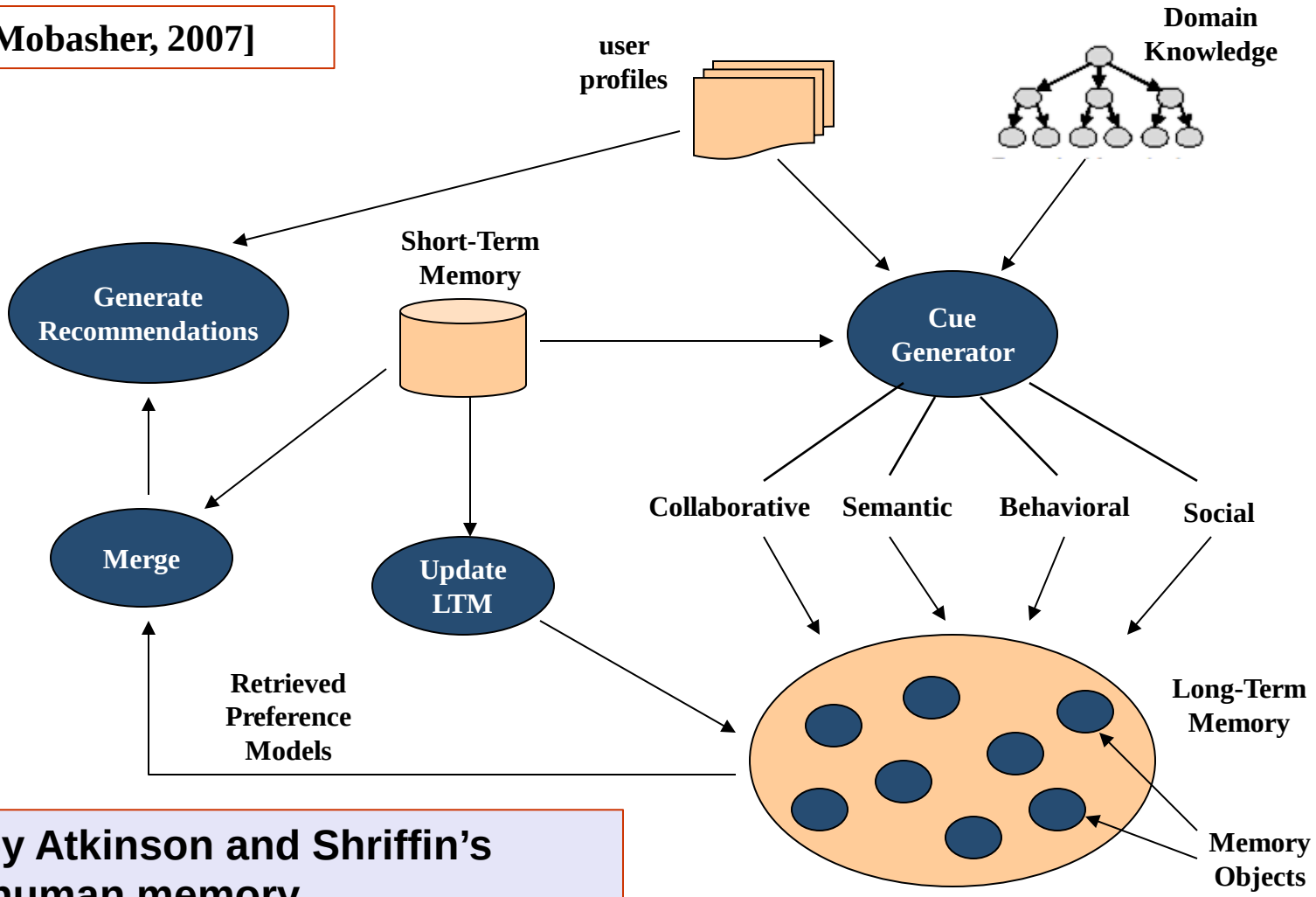
- Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model. KDD 2008
- Yehuda Koren, Collaborative filtering with temporal dynamics. KDD 2009
- A. Karatzoglou , X. Amatriain , L. Baltrunas , N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. RecSys 2010
- L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. RecSys 2011
- L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Luke, and R. Schwaiger. InCarMusic: Context-aware music recommendations in a car. ECWeb 2011
- Rendle, Gantner, Freudenthaler, Schmidt-Thieme: Fast context-aware recommendations with factorization machines. SIGIR 2011



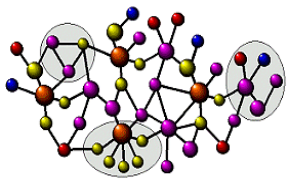
The Interactional View of Context

An Interactional Framework for Contextual Recommendation

[Anand and Mobasher, 2007]

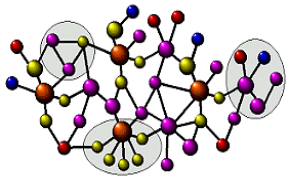


Inspired by Atkinson and Shriffin's model of human memory



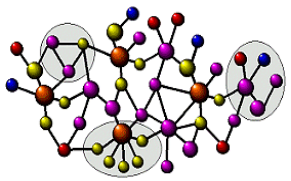
Contextual Recommendation Generation

- **Explicit or implicit preferences for items from the active interaction are stored in the STM**
- **Contextual cues are derived from this data and used to retrieve relevant preference models from LTM**
  - Relevant = belong to the same context as the active interaction
- **Merged with STM preferences and used to predict preferences for unseen items**
- **New Observations used to update preference models in LTM**
- **Lots of variations on how LTM preference models are organized**
  - based on ontological or semantic relationships
  - aggregate models based on similarities among users
  - based on connections in social or information networks
  - etc.

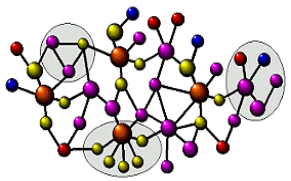


Characteristics of the Framework

- **The Framework Emphasizes**
  - The distinction between local, transient preference models in STM and the long-term established models in LTM
  - The importance of user's interaction with the system in deriving contextual cues
  - The mutually reinforcing relationship between user activity and the context model
    - Emphasizes the dynamic nature of context
- **Does Not Emphasize**
  - Explicit knowledge-based representation of contextual attributes
  - A rigid formulation of contextual modeling approaches
    - Very general framework and many implementations are possible (we will look at several next)

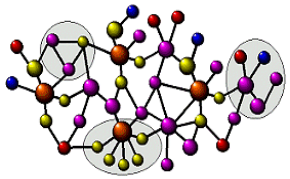


Highlighted Approach:
Latent Variable Context Models



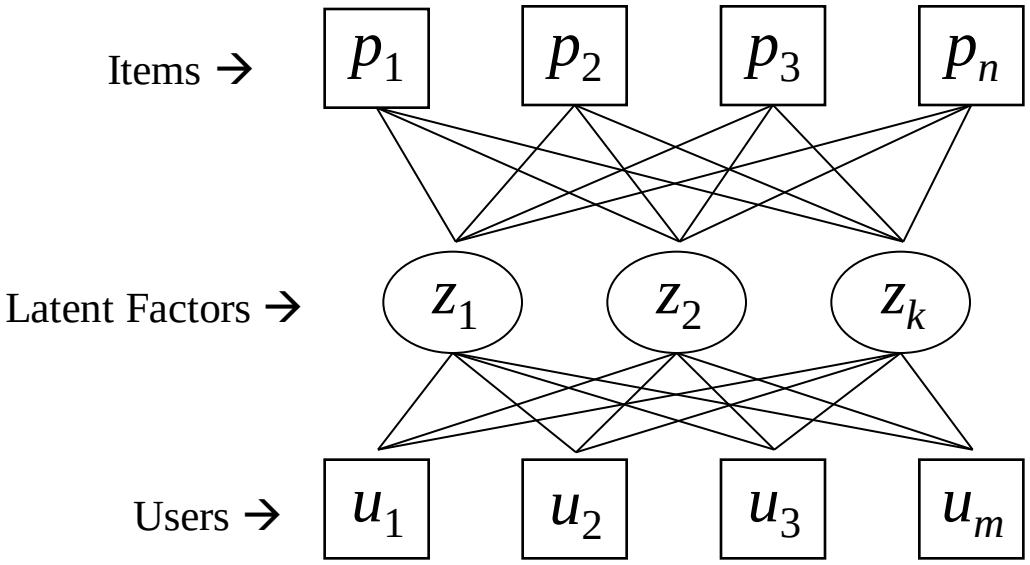
Latent Variable Context Model

- **Generative approach to modeling user context**
- **Basic assumption:**
  - users' interactions involve a relatively small set of contextual states that can "explain" users' behavior at different points during their interactions
- **Have been used effectively in applications involving user's performing informational or functional tasks**
- **Contexts correspond to tasks/activities and are derived as latent factors from the observed user data**
- **Latent variable models such as PLSA or LDA can be used to automatically characterize these "contexts," as well as their relationships to items or users**



Latent Variable Models

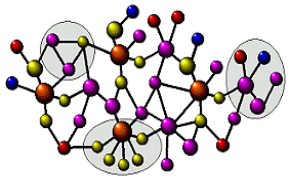
- **Assume the existence of a set of latent (unobserved) variables (or factors) which "explain" the underlying relationships between two sets of observed variables.**



Advantages:

Probabilistically determine the association between each latent factor and items, or between each factor and users.

In navigational/interactional data, the latent factors correspond to distinguishable patterns usually associated with performing certain informational or functional tasks. Context = Task!



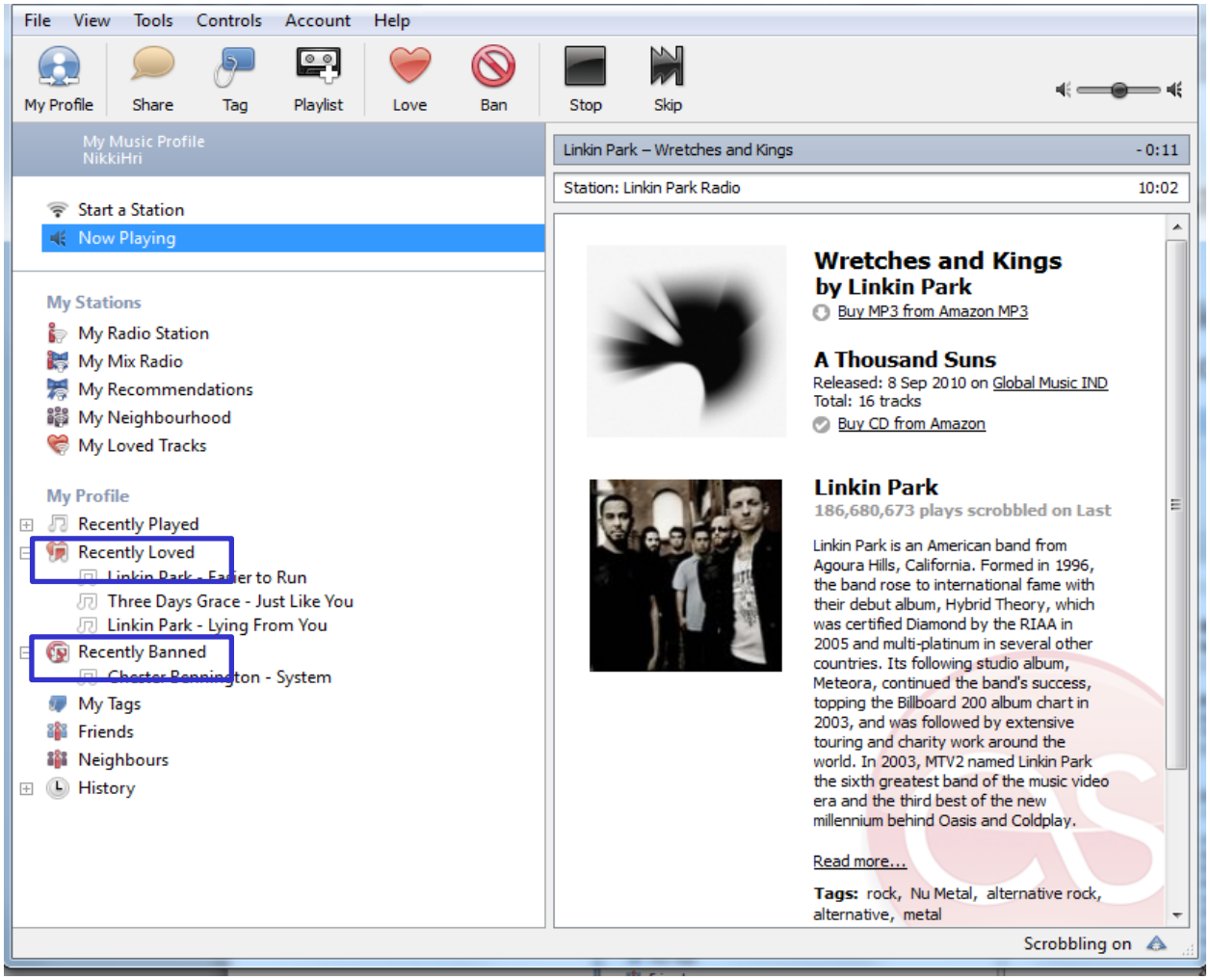
Example Implementation: Inferring Latent Contexts From Social Annotation

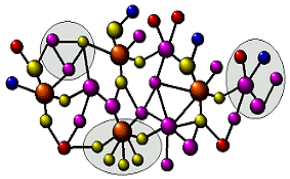
Hariri, Mobasher, Burke, RecSys 2012

- **Context-Aware Music Recommendation Based on Latent Topic Sequential Patterns**
- **In the domain of music recommendation:**
  - Context is usually not fully observable & may depend on many factors
    - Types of user activity (exercising, relaxing, driving, dancing)
    - User's moods or emotional states
    - Occasion or social setting
  - Contextual information is dynamic and should be inferred from users' interactions with the system such as:
    - Liking/disliking/skipping songs
    - Creating different stations by selecting different track seeds or artists
- **Different applications:**
  - Song recommendation, Playlist generation, Playlist recommendation

User Interactions

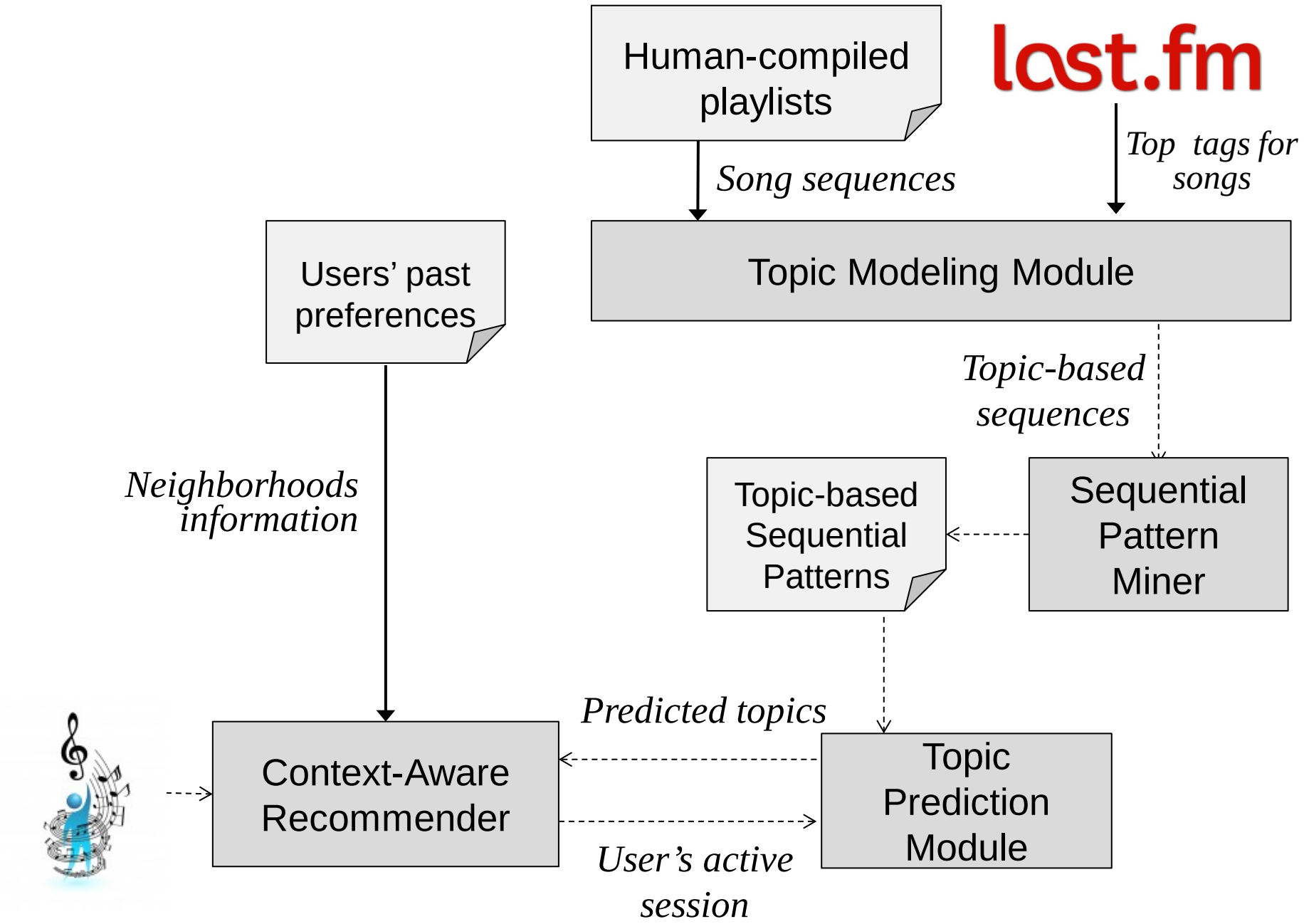
Context is reflected in the sequence of songs liked/disliked or played by the user in her current interaction with the system





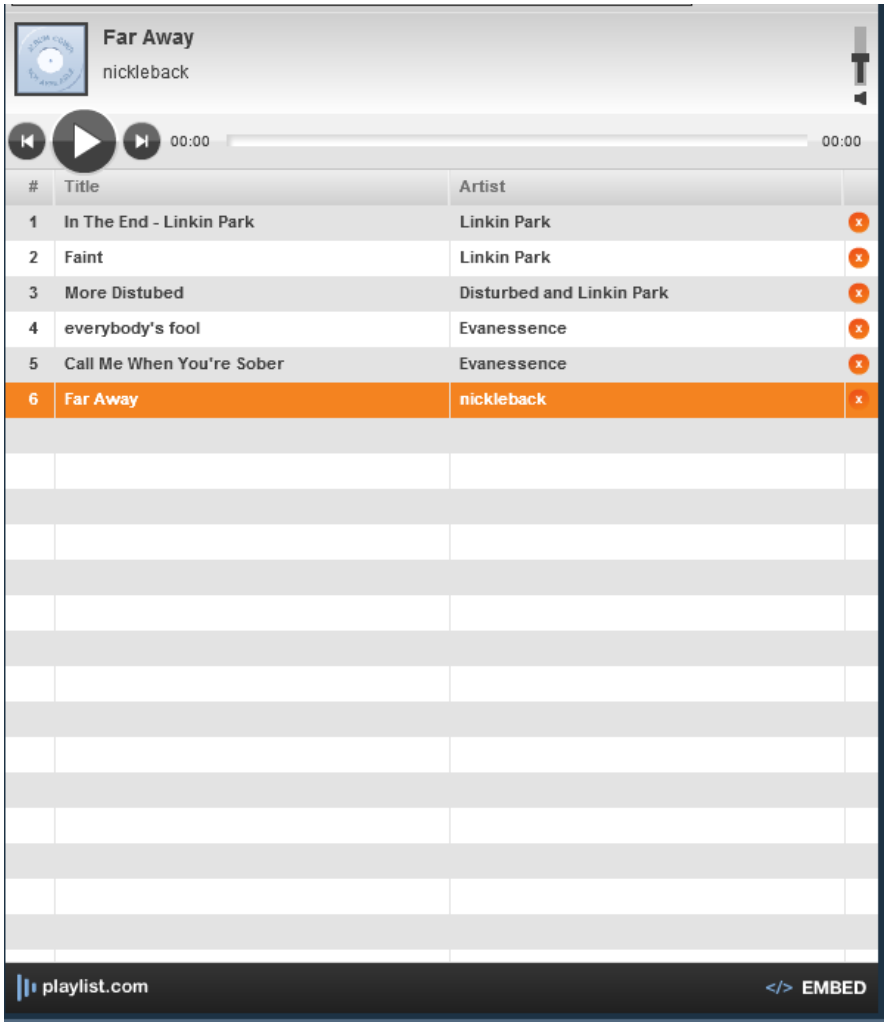
Goals of the Contextual Music Recommendation Framework

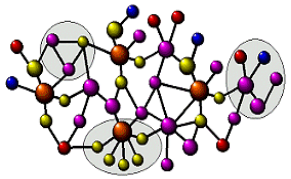
- Infer users' contextual states based on the most recent sequence of songs liked, played, or added to a playlist
- Utilize sequential information in user's history of interaction to identify and predict changes in context
- Adapt the system's recommendations to user's interest corresponding to these changes in context


Human-compiled playlists
last.fm
Top tags for songs
Song sequences
Users' past preferences
Topic Modeling Module
Topic-based sequences
Neighborhoods information
Topic-based Sequential Patterns
Sequential Pattern Miner
Predicted topics
Context-Aware Recommender
Topic Prediction Module
User's active session

Example Usage Scenario: Playlist Generation

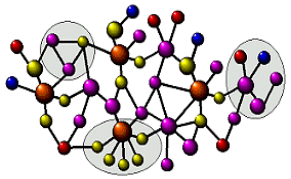
1. The user selects an initial sequence of songs for the playlist.
2. The system *infers* user's context and recommends a set of songs.
3. The user adds one of the recommendations (or a new song outside the recommendation set) to the playlist
4. The system updates its knowledge about the user's preferences before the next interaction





Topic Modeling for Song Context Representation

- **We use LDA topic modeling approach to map user's interaction sequence to a sequence of latent topics**
  - Better at capturing more general trends in user's interests
- **The latent topics are generated from the top most frequent tags associated with songs**
  - Tags obtained from social tagging Web sites such as last.fm.
  - Tags may characterize song features, user's situation, mood, etc.
  - For LDA, songs are taken as documents and tags as words
  - After fitting the topic model for *K* topics, the probability distribution over topics can be inferred for any given song
  - For each song, the set of dominant topics are selected that have probabilities higher than a specific threshold value

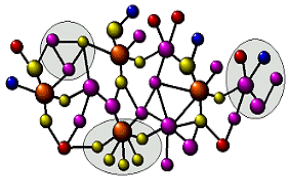


Top Most Frequent Tags for a Sample of Topics

Topic#1	Topic#2	Topic#3	Topic#4	Topic#5	Topic#6	Topic#7
ambient	latin	death	60s	chill	beautiful	electronic
instrumental	world	thrash	oldies	downtempo	sad	electronica
soundtrack	streamable	black	roll	chillout	mellow	house
classical	spanish	heavy	50s	christmas	melancholy	techno
beautiful	para	doom	rockabilly	lounge	acoustic	tranc
age	bossa	brutal	top	electronic	chill	electro
chillout	fusion	melodic	500	trip-hop	soft	bass
experimental	musica	california	radio	electronica	slow	drum
movie	que	power	rolling	trip	melancholic	ambient
atmospheric	nova	progressive	1960s	ambient	favourite	beat
world	brazilian	gods	rhythm	hop	chillout	idm
ethereal	african	seixas	time	easy	ballad	experimental
chill	party	speed	elvis	cool	singer-songwriter	club
calm	brasil	swedish	soundtrack	sexy	life	minimal
electronic	espanol	old	american	radio	easy	party

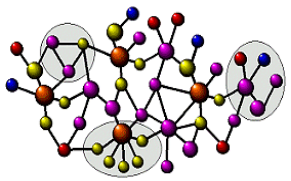
An Example Playlist (Mapped to Tags and Topics)

Time	Popular Tags	Dominant Topics
1	Singer-songwriter, mellow, relaxing, chill, male vocalist, easy listening, acoustic, 00's, guitar, rock, happy	6
2	Singer-songwriter, chill, acoustic, mellow, rock, summer, surf, male vocalist, pop, relaxing, guitar, happy	6
3	singer,-songwriter, indie rock, folk, acoustic, mellow, chill out, relaxing, bittersweet, lo-fi	6, 20, 23
4	Alternative rock, ballads, calm, beautiful, nice, soundtrack, favorites	6, 28
5	Electronic, electronica, French, chill out, trip-hop, ambient, down-tempo, sexy, 90s, alternative, easy listening, guitar, mellow, relax, female vocal	7, 5
6	Soundtrack, 90s, alternative, atmospheric, female vocalist, indie, dreamy	23
7	Singer-songwriter, acoustic, chill, alternative, rock, male vocalist, easy listening, driving	6, 25
8	Cover, Beatles cover, rock, 90s, soundtrack, brass , pop rock, alternative, rock, folk, brass	30, 18
9	Indie, rock, acoustic, 90s, cover, mellow, pop, folk, dreamy, singer-songwriter, sad-core, summery, sweet, alternative rock, female vocalist	6, 20



Sequential Pattern Mining and Topic Prediction

- **Using a training set of human-compiled playlists, sequential patterns are mined over the set of corresponding latent topic sequences**
  - Each pattern represents a frequent sequence of transitions between topics/contexts
- **Why Topic Level Sequential Patterns?**
  - Mining SPs on topics instead of songs is useful in capturing user interests based on common characteristics of the current context
  - Makes it easier to track and detect changes in the users' preferences due to changes in contextual states
  - Topic-based patterns are useful in managing the cold start problem: a new songs may still match topic-based patterns



Song Recommendation Based on Contextual Post-filtering

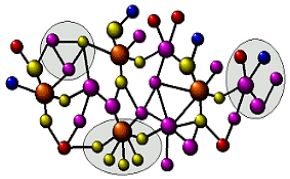
- **The predicted topics are used to contextualize the recommendations**

$$contextScore(h_u, s) = \frac{\sum_{t_i \in predictedTopics(h_u)} p(t_i|s)}{|predictedTopics(h_u)|}$$

  - *ContextScore*(h_u, *s*) represents the suitability of song *s* for the current context of user *u* (determined based on user's active session, h_u)

- **Next, recommendations are re-ranked using the contextual information**

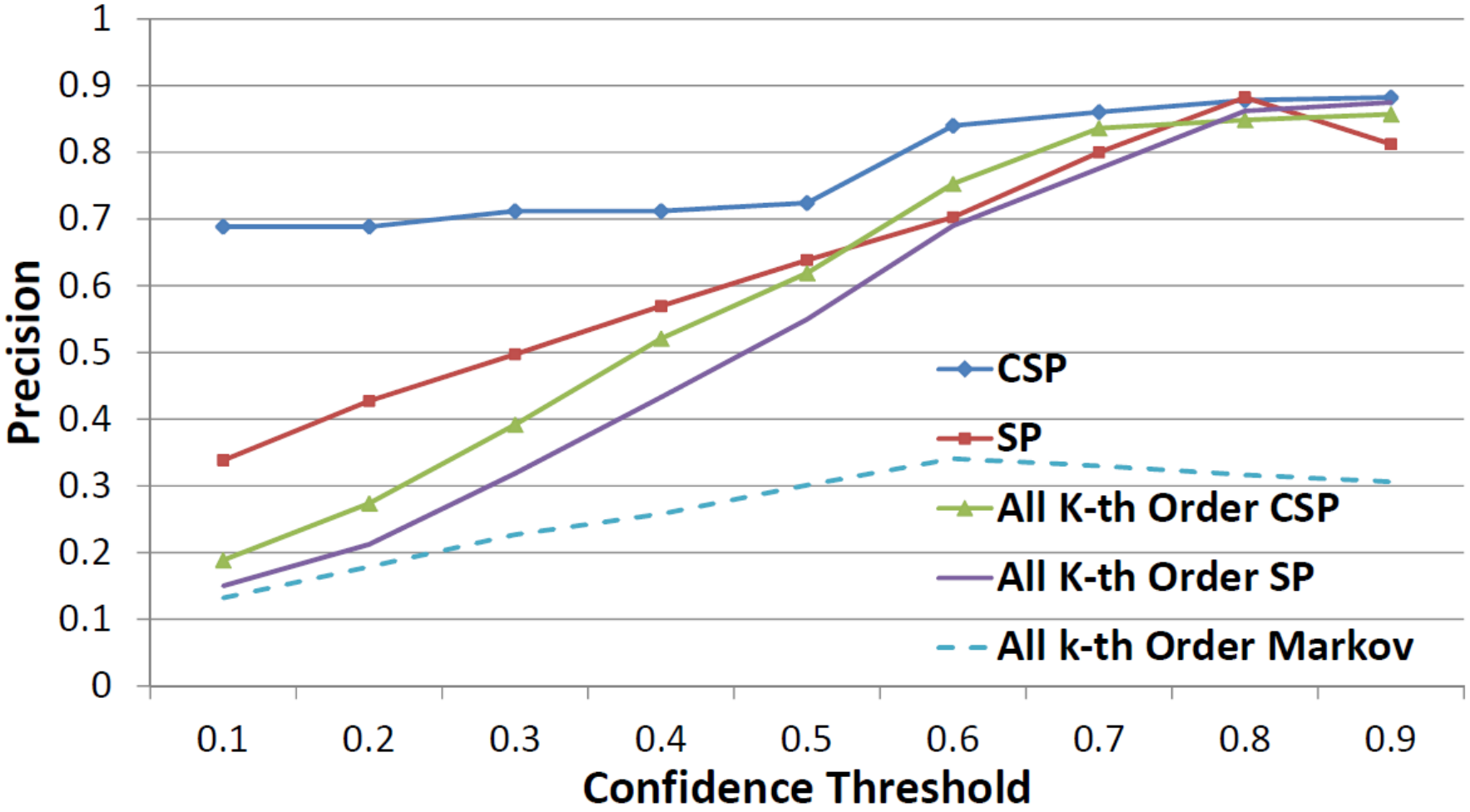
$$predictionScore(s) = (contextScore(s) + \alpha_1) \cdot (CFScore(s) + \alpha_2)$$



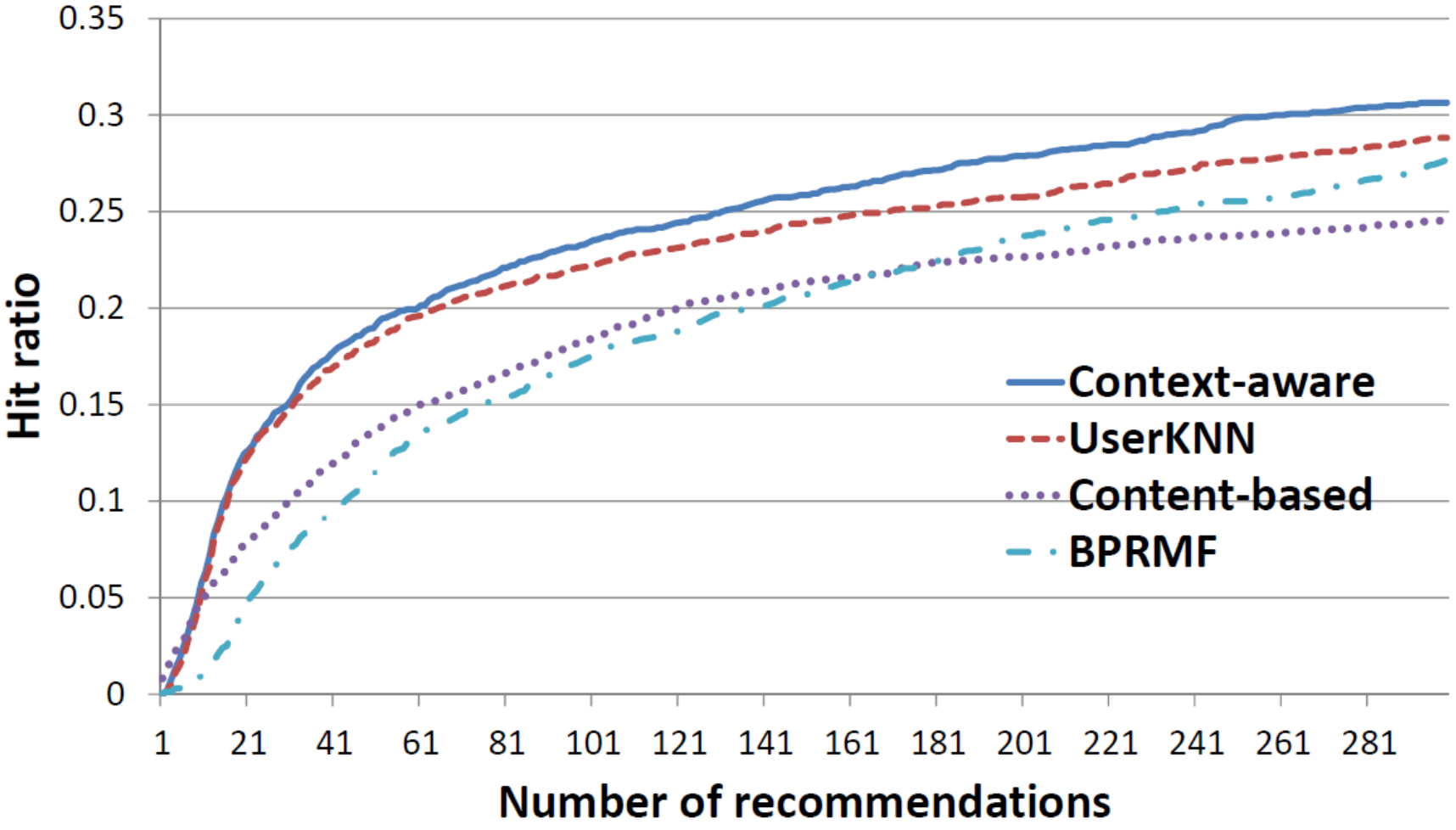
Evaluation

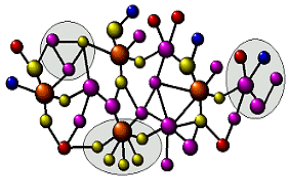
- **Dataset and Methodology:**
  - 28,963 user-contributed playlists from *Art of the Mix* website in January 2003
  - This dataset consists of 218,261 distinct songs for 48,169 distinct artists
  - Top tags were retrieved from the last.fm website for about 71,600 songs in our database
  - 48K songs with min. of 5 tags were used to build a 30-topic LDA model
  - The last $w = 7$ songs were selected as the user's active session, the last song was removed and the dominant topics associated with that song were used as target set

Topic Prediction Precision



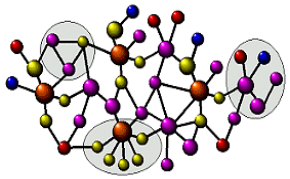
Song Recommendation Performance





Conclusions

- **Incorporating context in recommendation generation *can* improve the effectiveness of recommender systems**
- **What does it take?**
  - **In representational models:** careful selection of relevant contextual attributes for the specific domain (the classic knowledge engineering task)
  - **In Interactional Models:** effective methods for extraction of contextual cues from user behavior & ways of coping with domains that don't lend themselves to user interactions
- **Work on Interactional Models Suggests:**
  - observable behavior is "conditioned" on the underlying context
  - The context can be inferred (and predicted) effectively in certain kinds of applications
  - The integration of semantic knowledge or social cues with user activity can be particularly effective in contextual user modeling



?