Introduction to the Social Web

Recommendation and Mining

Sihem Amer-Yahia

updated by Vincent Leroy

Sihem Amer-Yahia

- Ph.D. in CS, 1999, Univ. of Paris-Orsay & INRIA, France
- Research Scientist, at&t labs: 1999-2006
- Senior Research Scientist, Yahoo! Research: 2006-2011
  – Member of the jury of a young PhD student, *Vincent Leroy*
- Principal Research Scientist, QCRI: 2011-12
- Since Dec 2011: DR1 CNRS@LIG
  – Big Data Management and Query Processing for Search and Recommendation and their application to Social Computing, Large-scale information exploration algorithms
  – Head of the SLIDE team (ScaLable Information Discovery and Exploitation) at LIG (among which …)

Social Content Sites

- **Web destinations that let users:**
  - Consume and produce content
    - Videos / photos / articles /…
    - tags / ratings / reviews /…
  - Engage in social activities with
    - friends / family / colleagues / acquaintances /…
    - people with similar interests / located in the same area /…
- **Two major driving factors:**
  - Social activities improve the attractiveness of traditional content sites
    - the "similar traveler" feature improves user engagement
  - Content is critical to the value of social networking sites
    - a significant amount of user time is spent browsing other people's photos, posts, etc.

Social Content Sites

- **Users engage the system**
  - Contribute content
  - Disclose information about themselves
  - Need help navigating the ever-growing cyber-city maze
- **Ultimate goal**
  - Personalize search and information discovery
  - Predict what a user's interests will be in the future
  - Understand user behavior
- **Many social content sites, collaborative tagging sites are one particular kind**
  - *Flickr*, *YouTube*, *Delicious*, photo tagging in *Facebook*

Recommendation Outline

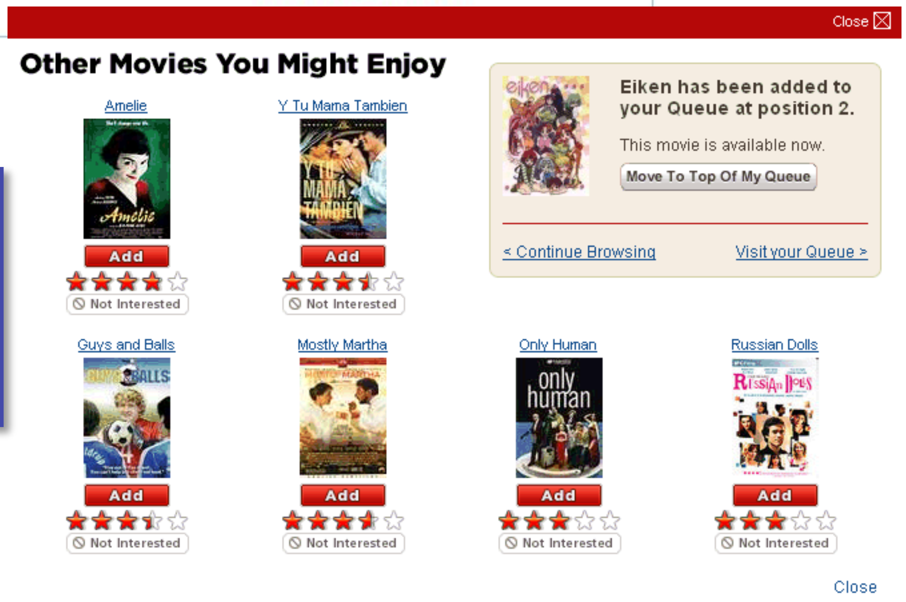
- Recommender Systems
  - **What are recommender systems** and how do they work?
  - Example application: Hotlist Recommendation on Delicious
  - How are recommender systems evaluated?

- Recommendation challenges
  - Well-known challenges
  - Recommendation diversity
  - Group recommendation

Recommender Systems

All are social content sites that thrive on User Generated Content (UGC)!

Recommender System



- *Predict ratings for unrated items*
- *Recommend top-k items*

Motivation

- Amazon makes 20-30% of its sales from recommendations. Only 16% of people go to Amazon with explicit intent to buy something
- Collected data matters more than the algorithm.
  - Amazon's algorithm is essentially a large product-product correlation matrix for the past hour, but it works for them because they collect so much data through user actions
- A lot of types of data can be used: votes, ratings, clicks, page-view time, purchases, tagging…

Academia: An Overview

- **Early days: 3 papers by HCI researchers (1995)**
- **Today: over 1000 papers**
  - ACM RecSys09
    - 203 submissions, thereof 140 long and 63 short papers
    - acceptance rate for long papers of 17% and of 34% overall
  - Fields: CS/IS, marketing, DM/statistics, MS/OR
- **Netflix $1M Prize Competition**
  - Data: ≈18K movies, ≈500K customers, 100M ratings
  - $1M Prize: improve Netflix RMSE rates by 10%
  - ≈ 40K contestants from 179 countries
  - Winners in June 2009: a coalition of four: BellKor's Pragmatic Chaos with statisticians, machine learning experts and computer engineers from America, Austria, Canada and Israel — declared that it had produced a program that improves the accuracy of the predictions by 10.05 percent.

Recommendation Outline

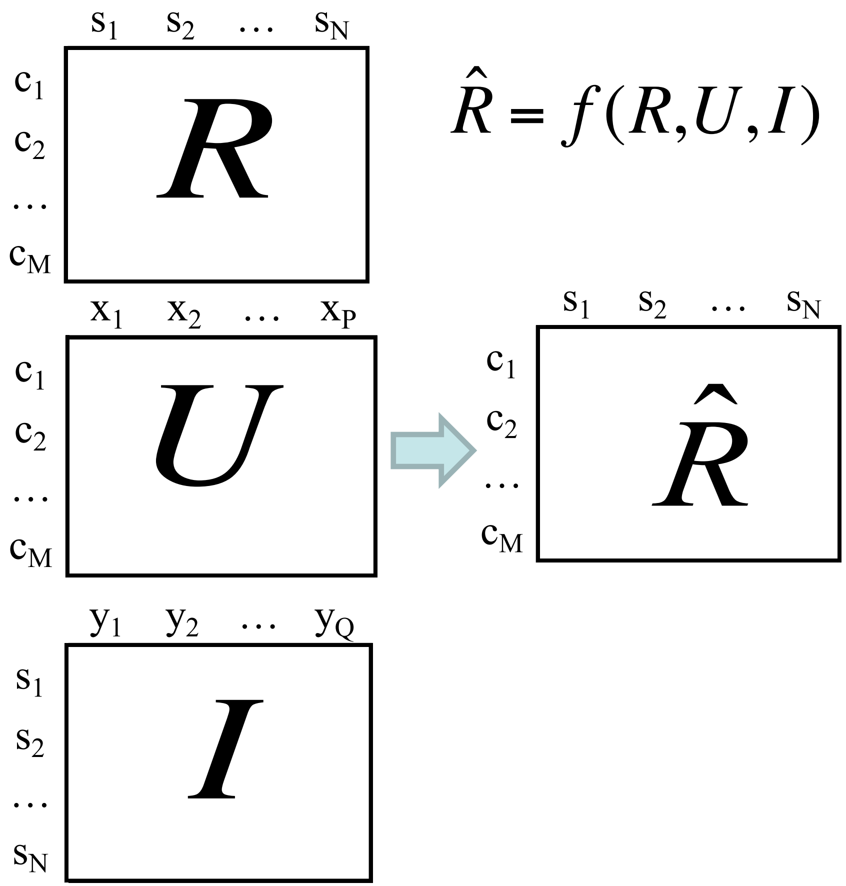
- Recommender Systems
  - What are recommender systems and **how do they work**?
  - Example application: Hotlist Recommendation on Delicious
  - How are recommender systems evaluated?

- Recommendation challenges
  - Well-known challenges
  - Recommendation diversity
  - Group recommendation

Recommendation Model

- **Input**
  - Rating matrix *R:* r_{ij} – rating user c_i assigns to item s_j (*explicit* Vincent rates Westworld 5/5, or *implicit* Vincent listened to Explosions in the sky 659 times)
  - User attribute matrix *U:* x_{ij} – *attribute* x_j *of user* c_i (e.g. demographic attributes)
  - Item attribute matrix *I:* y_{ij} – *attribute* y_j *of item* s_i (e.g. product category, tags)
- **Output**
  - Predicted new matrix $\hat{R}$

$$\hat{R} = f(R, U, I)$$



Types of Recommendations

- **Content-based**
  - How similar is an item *i* to items *u* has liked in the past?
  - Uses metadata for measuring similarity
  - Works even when no ratings are available on items
  - Requires metadata!

- **Collaborative filtering**
  - Treat items and users as vectors of ratings, compute vector distance

Taxonomy of Traditional Recommendation Methods

- Recommendation approach [Balabanovic & Shoham 1997]
  - Content-based, collaborative filtering
- Nature of the prediction technique
  - Heuristic-based (uses matrix as is), model-based
- Support for rating/transaction data
  - Both, rating-only [R], transaction-only [T]

	Heuristic-based	Model-based
Content-based		
Collaborative filtering		

Content-based, Heuristic-based

- Item similarity methods
  - Information Retrieval (IR) Techniques
  - Treat each item as a document
  - Item similarity computed as document similarity

	Heuristic-based	Model-based
Content-based		
Collaborative filtering		

Similarity Measures

- Use attributes of items to build an item profile
- User profile v_i of user c_i constructed by aggregating profiles of items c_i has experienced
- Ex:
  - Justin Bieber (Pop 723, R&B 428, Canada 109)
  - Selena Gomez (Pop 341, Female Vocalist 156)

  → Similarity = 0.77

$$\hat{r}_{ij} = score(\mathbf{v}_i, \mathbf{y}_j)$$

$$\hat{r}_{ij} = \cos(\mathbf{v}_i, \mathbf{y}_j) = \frac{\mathbf{v}_i \bullet \mathbf{y}_j}{\| \mathbf{v}_i \|_2 \cdot \| \mathbf{y}_j \|_2}$$

TF-IDF: relevance in Information Retrieval

- Some attributes are very frequent (e.g. *rock* or *pop* tags on music)
  - Not able to differentiate items accurately
- *Romantic ballads* is much less frequent
  - Sharing this tag is much more meaningful
- Term Frequency:
  - The more a term is present in a document the more meaningful it is for this document (equivalent to tag frequency for an item)
- Inverse Document Frequency:
  - The fewer documents contain this term, the more meaningful it is (equivalent to a tag only used on a few items is more meaningful than a tag used on all items)

Term Frequency

Variants of TF weight

weighting scheme	TF weight
binary	$0, 1$
raw frequency	$f_{t,d}$
log normalization	$1 + \log(f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Inverse Document Frequency

Variants of IDF weight

weighting scheme	IDF weight ($n_t = \|\{d \in D : t \in d\}\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t}$
inverse document frequency smooth	$\log(1 + \frac{N}{n_t})$
inverse document frequency max	$\log\left(1 + \frac{\max_{\{t' \in d\}} n_{t'}}{n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

Item Similarity based on IR

- Account for TF and IDF when building the vector of an item / user
- Item attributes are word occurrences in each document

$$y_{ij} = TF_{ij} \cdot IDF_j$$

- TF_{ij} – term frequency: frequency of word y_j occurring in the description of item s_i;
- IDF_j – inverse document frequency: inverse of the frequency of word y_j occurring in descriptions of all items

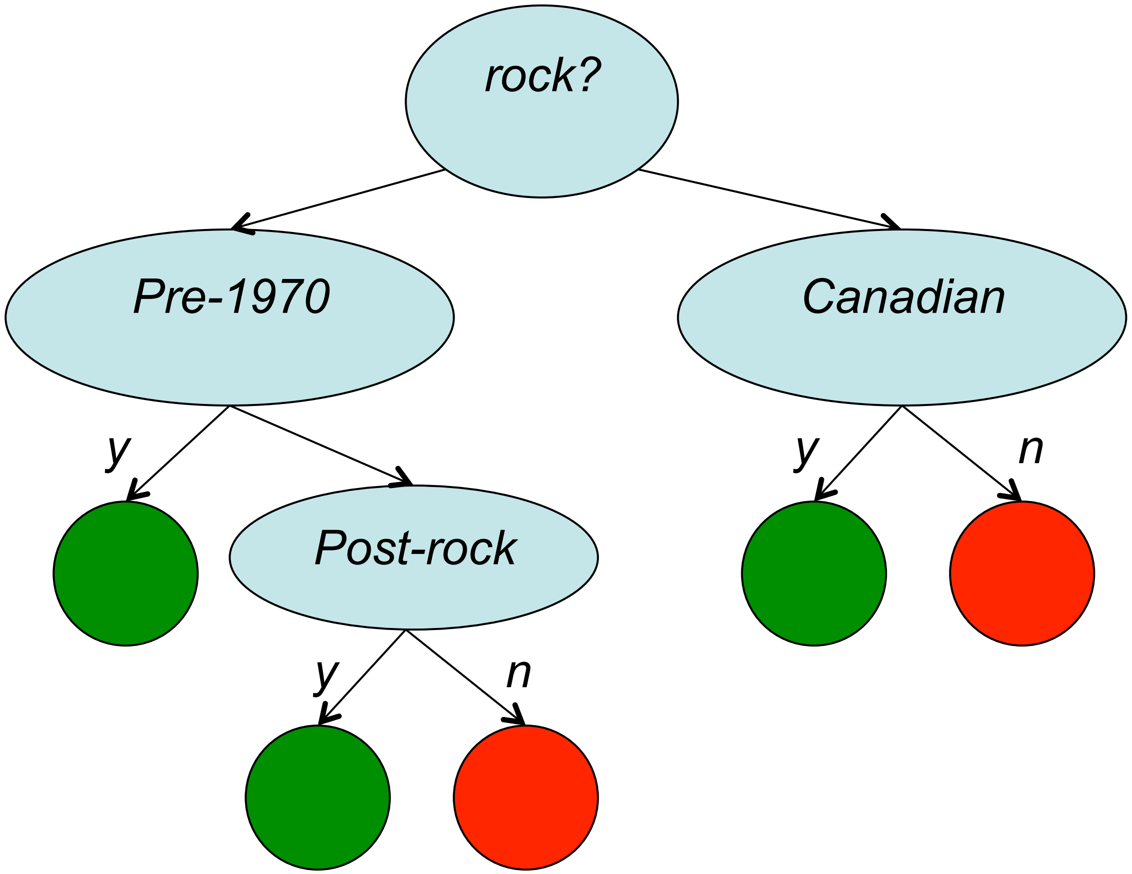
Content-based, Model-based

- Classification models [Pazzani & Billsus 1997; Mooney & Roy 1998]
- One-class Naïve Bayes classifier [Schwab et al. 2000]
- Latent-class generative models [Zhang et al. 2002]

	Heuristic-based	Model-based
Content-based		
Collaborative filtering		

Tree-based classification model

- Train a classifier using attributes to predict 2 classes:
  - Liked
  - Disliked



Collaborative Filtering Algorithms

- Non-Personalized Summary Statistics
- K-Nearest Neighbor
- Dimensionality Reduction
- Content + Collaborative Filtering
- Graph Techniques
- Clustering
- Classifier Learning

	Heuristic-based	Model-based
Content-based		
Collaborative filtering		
Hybrid		

Collaborative Filtering, Heuristic-based

- **Neighborhood methods**
  - User-based algorithm [Breese et al. 1998; Resnick et al. 1994; Sarwar et al. 1998]
  - Item-based algorithm [Deshpande & Karypis 2004; Linden et al. 2003; Sarwar et al. 2001]
  - Similarity fusion [Wang et al. 2006]
  - Weighted-majority [Delgado and Ishii 1999]
  - Matrix reduction methods (SVD, PCA processing) [Goldberg et al. 2001; Sarwar et al. 2000]
- **Association rule mining [Lin et al. 2002]**
- **Graph-based methods [Aggarwal et al. 1999; Huang et al. 2004, 2007]**

	Heuristic-based	Model-based
Content-based		
Collaborative filtering		
Hybrid		

Collaborative Filtering, Heuristic-based (examples from Rajaraman and Ullman book)

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Jaccard

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Jaccard(A,B) = 1/5 < 2/4 = Jaccard(A,C)

Cosine

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

, where A_i and B_i are components of vector A and B respectively.

cos(A,B) = 0.380 > 0.322 = cos(A,C)

Normalizing ratings

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	-7/3		
B	1/3	1/3	-2/3				
C				-5/3	1/3	4/3	
D		0					0

Replace each rating with its difference with the mean (average) for that user
Low ratings become negative
High ratings are positive

Cosine: users with opposite views on common movies will have vectors in opposite directions and users with similar opinions about movies rated in common will have a small angle.

cos(A,B) = 0.092 > -0.559 = cos(A,C)

K Nearest Neighbors recommendation

- Using Ratings Matrix select *k* most similar users
- Aggregate their ratings to create a ranking of items
  - E.g. 3 users that love the same series as Jon love *Stranger Things*, and I haven't seen it → recommend *Stranger Things* to Jon

Collaborative Filtering, Model-based

- Matrix reduction methods [Takacs et al. 2008; Toscher et al. 2008]
- Latent-class generative model [Hofmann 2004; Kumar et al. 2001; Jin et al. 2006]
- User-profile generative model [Pennock et al. 2000; Yu et al. 2004]
- User-based classifiers [Billsus & Pazzani 1999; Pazzani & Billsus 1997]
- Item dependency (Bayesian) networks [Breese et al. 1998; Heckerman et al. 2000]

	Heuristic-based	Model-based
Content-based		
Collaborative filtering		
Hybrid		

Alternating Least Squares (ALS)

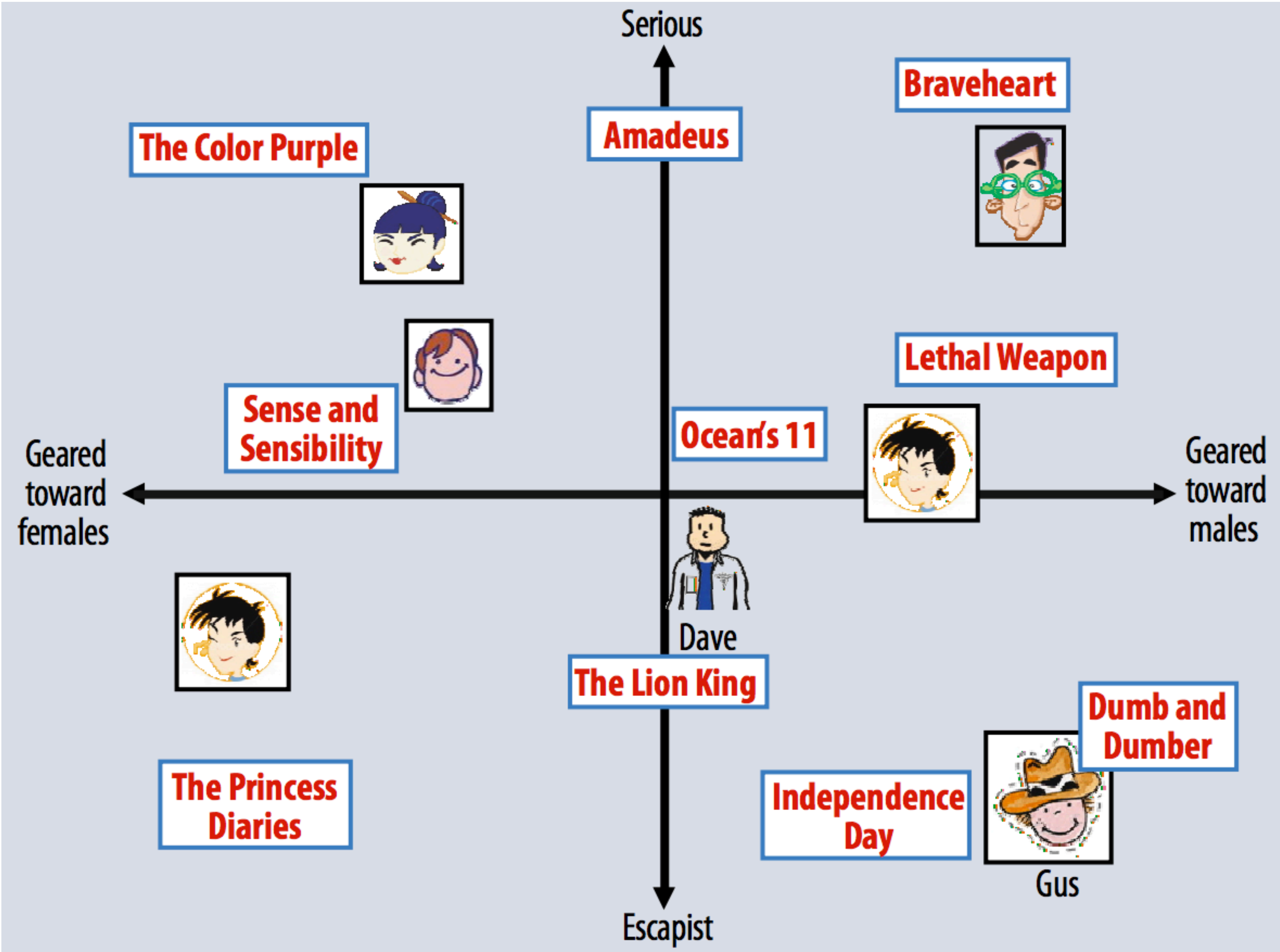
- The type of approach that won the Netflix prize!
- Matrix Factorization method
  - Represent users and items as vectors p_u and q_i
  - Prediction $\hat{r}_{ui} = q_i^T p_u$
- How do you learn these vectors?

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

  - Minimize prediction error on known ratings (κ) while keeping the model simple (λ) to avoid *overfitting*
  - 2 parameters: number of dimensions of vectors (hidden features, called rank), and regularization parameter λ

Read more: https://goo.gl/6z09EG

ALS on 2 dimensions



Solving ALS

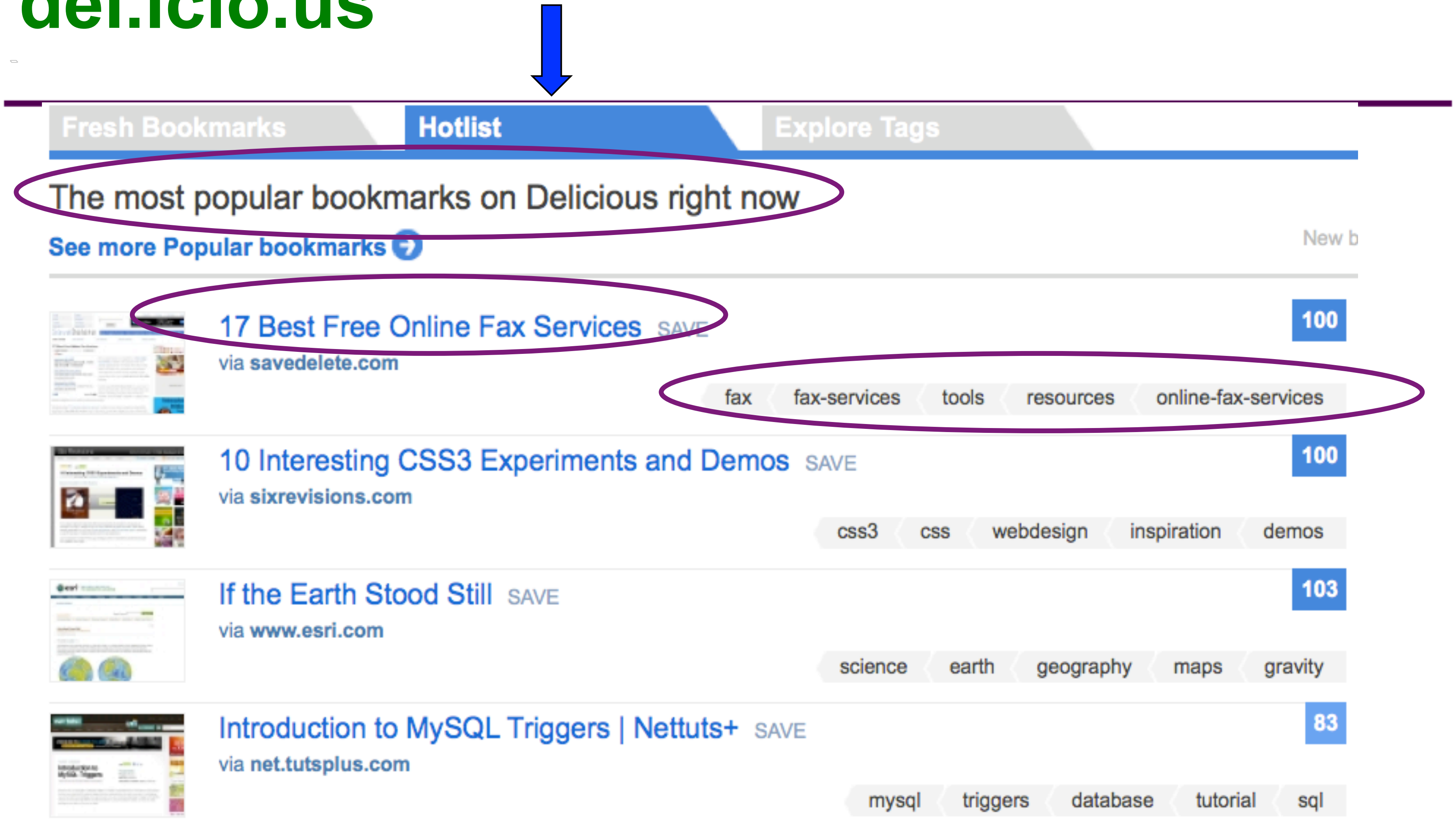
- Fix user vectors
  - Solve equation to find optimal items vectors
- Fix item vector
  - Solve equation to find optimal user vectors
- Execute until convergence (or for *x* iterations)
- There is a Spark implementation of this!
  - Mllib

Recommendation Outline

- Recommender Systems
  - What are recommender systems and how do they work?
  - **Example application: Hotlist Recommendation on Delicious**
  - How are recommender systems evaluated?

- Recommendation challenges
  - Well-known challenges
  - Recommendation diversity
  - Group recommendation

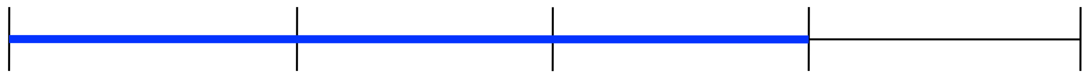
del.icio.us



Fresh Bookmarks
Hotlist
Explore Tags
The most popular bookmarks on Delicious right now
See more Popular bookmarks
New b
17 Best Free Online Fax Services SAVE
via savedelete.com
100
fax
fax-services
tools
resources
online-fax-services
10 Interesting CSS3 Experiments and Demos SAVE
via sixrevisions.com
100
css3
css
webdesign
inspiration
demos
If the Earth Stood Still SAVE
via www.esri.com
103
science
earth
geography
maps
gravity
Introduction to MySQL Triggers | Nettuts+ SAVE
via net.tutsplus.com
83
mysql
triggers
database
tutorial
sql

del.icio.us Hotlists Experiment

- **116,177 del.icio.us users**
  - who tagged 175,691 distinct URLs
  - using 903 tags
  - for a total of 2,322,458 tagging actions
  - for 1 month in 2006
- **Evaluate how networks predict user's interest**
  - *J. Stoyanovich, S. Amer-Yahia, C. Yu, C. Marlow: Leveraging Tagging Behavior to Model Users' Interest in del.icio.us (AAAI Workshop on Social Information Processing 2008)*



A/B testing: user behavior in first 3 weeks to predict 4th week

Data Model

- users $u \in U$, tags $t \in T$, items $i \in I$
- *friends(u)* directional

- *tags(u)*
- *items(u)* & *items(u,t)*

- *taggers(i)* & *taggers(i,t)*

Tagging data has a long tail

- we have to clean it for efficiency (relational processing)
- we removed unpopular tags (< 4 uses) & URLs (< 10 uses), reduced to 27% of original size

Global

10 URLs that are tagged most often over-all

Performance

coverage (**`global`**) = 3%

scope (**`global`**) = 100%

Global Top-10

Rank	***URL***	***Votes***
1	*google.com*	*980*
2	*facebook.com*	*820*
3	*iTunes.com*	*729*
4	*twitter.com*	*720*
5	*jonasbrothers.com*	*680*
6	*cnn.com*	*678*
7	*amazon.com*	*620*
8	*yahoo.com*	*525*
9	*youtube.com*	*524*
10	***techcrunch.com***	*492*

Items(Chris)

URL	***Tag***
jars.com	*java*
java.sun.com	*java*
techcrunch.com	*news*
devshed.com	*tutorial*

Items(Ben)

URL	***Tag***
bbc.co.uk	*news*
pbs.org	*news*
tomwaits.com	*music*
nick-cave.com	*music*
loureed.com	*music*

Tag-based

- **If a user tags with *sports*, he is interested in sports-related content**
  – interest(u,t) = |items(u,t)| / |items(u)|

Top-10 for "news"

Rank	*URL*	*Votes*
1	*cnn.com*	*610*
2	*bbc.co.uk*	*503*
3	*npr.org*	*427*
4	*nytimes.com*	*414*
5	*slashdot.org*	*392*
6	*reuters.com*	*330*
7	*news.cnet.com*	*290*
8	*msnbc.msn.com*	*250*
9	*news.yahoo.com*	*180*
10	*digg.com*	*149*

Top-10 for "music"

Rank	*URL*	*Votes*
1	*iTunes.com*	*542*
2	*eMusic.com*	*420*
3	*pandora.com*	*350*
4	*thebeatles.com*	*330*
5	*jonasbrothers.com*	*215*
6	*madonna.com*	*175*
7	*rhapsody.com*	*148*
8	*rollingstones.com*	*133*
9	*lastfm.com*	*120*
10	*beyonce.com*	*107*

Items(Ben)

URL	*Tag*
bbc.co.uk	*news*
pbs.org	*news*
tomwaits.com	*music*
nick-cave.com	*music*
rollingstones.com	*music*

Build one global hotlist per tag, use in one of two ways

- `best_tag`
  hotlist = top-10 for tag for which user has highest interest

- `dominant_tags`
  hotlist is a combination of up to 3 top-10 lists s.t. interest(u,t) $\geq$ 0.3 (user has *strong interest* for these tags)

Performance of Tag-based

`best_tag`

coverage = 9%

***scope* = 100%**

`dominant_tags`

1 tag	coverage = 10%	scope = 32%
2 tags	coverage = 14%	scope = 14%
3 tags	coverage = 18%	scope = 6%

Network-based

Choose 10 most popular URLs from those tagged by a user's friends.

coverage (`friends`) = 43%

scope (`friends`) = 31%

Common Interest Networks: URL-interest

Identify the seed -- a set of users who tag many of the same URLs as the user* u *("agree with* u*"). Hotlist = 10 most popular URLs tagged by users in seed.

$agr(u,f) = |items(u) \cap items(f)| \,/\, |items(u)|$
$U_{scope} = \{u \in U \mid \exists f \in U,\ agr(u, f) > threshold\}$
$U_{seed} = \{f \in U \mid agr(u, f) > threshold\}$

thresh = 0.3	coverage = 61%	scope = 1.2%
thresh = 0.5	coverage = 71%	scope = 0.7%

Common Interest Networks: Tag-URL-Interest

Agreement across the board is rare, let's look at agreement per-tag: may agree with adviser on research, but with mom on cooking.

agr (u,f,t)=|items(u,t) ∩ items(f,t)| / |items(u,t)|

U_{scope}, U_{scope} defined as for `url-interest`, combined as in `dominant-tags`.

scope (`tag-url-interest`) = 7%

Tag/Interest-based Methods: a Comparison

Users in the intersection of `dominant-tags`, `url-interest` and `tag-url-interest`, with a strong interest in 2 tags, all thresholds = 0.3

	$\lvert U_{scope} \rvert$	avg ($\lvert U_{seed} \rvert$)	coverage
`dominant-tags`	1235	26,856	17%
`tag-url-interest`	1235	227	82%
`url-interest`	205	203	85%

Recommendation Outline

- Recommender Systems
  - What are recommender systems and how do they work?
  - Example application: Hotlist Recommendation on Delicious
  - **How are recommender systems evaluated?**

- Recommendation challenges
  - Well-known challenges
  - Recommendation diversity
  - Group recommendation

Evaluation Approaches

- **Industry outcome**
  - Add-on sales
  - Click-through rates
- **In research**
  - Offline: To anticipate the above beforehand
    - No actual users are involved and an existing dataset is split into a test and a training set
    - Using the ratings in the training set, predict the ratings in the test set
    - Predicted ratings are compared with ratings in the test set using different measures
    - In K-fold cross validation (a common cross validation technique), the data set is partitioned into K equal-sized subsets: one is retained and used as the test set, the other subsets are used as training set. This process is repeated K times, each time with a different test set.
  - Online: User satisfaction

Evaluation Metrics

- Accuracy Metrics
  - measure how well a user's ratings can be reproduced by the recommender system, and also how well a user's ranked list is predicted
  - 3 kinds of accuracy metrics
    - Predictive
    - Classification
    - Rank
- Other metrics:
  - Coverage, Confidence, Diversity, Novelty and Serendipity

Predictive Metrics

- measure to what extent a recommender system can predict ratings of users.
- useful for systems that display the predicted ratings to their users.
- MAE = (|0|+|1|+|3|+|0|+|− 2| + |0| + |2|)/7 = 1.143

$$MAE = \frac{1}{|B_i|} \sum_{b_k \in B_i} |r_i(b_k) - p_i(b_k)|$$

Also RMSE (Root Mean Squared Errors) as discussed for ALS
→ Several small errors is better than one big errors

	Ranking		Rating	
Item	**User**	**RS**	**User**	**RS**
A	1	1	5	5
B	2	5	4	3
D	3	4	4	4
G	4	6	4	2
E	5	3	3	5
C	6	2	2	5
F	7	7	2	2

Classification Metrics

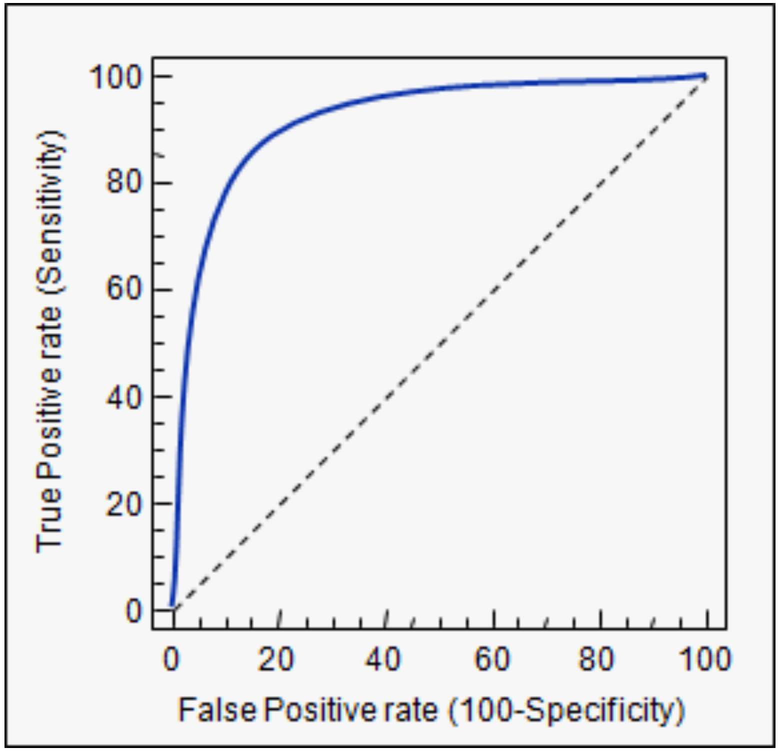
- measure to what extent a RS is able to correctly classify items as interesting or not.
- Ignores rating difference

		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

- *Precision: TP/(TP+FP)*
  - measures proportion of recommended items that are good
- *Recall: TP/(TP+FN)*
  - measures proportion of all good items recommended

ROC curve

- Combine Recall and Precision
- *Imagine a recommender that orders items from the most likely to the least likely*



Rank Metrics
DCG, nDCG for list comparison

- A measure of effectiveness of a web search engine algorithm or related applications
- DCG measures the usefulness, or *gain*, of a document based on its position in the result list
- Two assumptions are made in using DCG:
  - Highly relevant documents are more useful when appearing earlier in a search engine result list (have higher ranks)
  - Highly relevant documents are more useful than marginally relevant documents, which are in turn more useful than irrelevant documents.
- DCG originates from an earlier, more primitive, measure called Cumulative Gain.

Cumulative Gain: CG

It is the sum of the graded relevance values of all results in a search result list.

The CG at a particular rank position p is defined as:
where rel_i is the graded relevance of the result at position i.

$$\mathrm{CG_p} = \sum_{i=1}^{p} rel_i$$

CG Example

$$D_1, D_2, D_3, D_4, D_5, D_6$$

the user provides the following relevance scores:

$$3, 2, 3, 0, 1, 2$$

$$\mathrm{CG_p} = \sum_{i=1}^{p} rel_i = 3 + 2 + 3 + 0 + 1 + 2 = 11$$

does not account for document ordering.

Discounted Cumulative Gain: DCG

DCG is that highly relevant documents appearing lower in a search result list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result.
The discounted CG accumulated at a particular rank position is defined as:

$$\mathrm{DCG_p} = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i)}$$

No theoretical justification for using a logarithmic reduction factor other than it produces a smooth reduction.
An alternative formulation of DCG places stronger emphasis on retrieving relevant documents:

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

DCG Example

$D_1, D_2, D_3, D_4, D_5, D_6$

the user provides the following relevance scores:

$3, 2, 3, 0, 1, 2$

i	rel_i	$\log_2 i$	$\frac{rel_i}{\log_2 i}$
1	3	0	N/A
2	2	1	2
3	3	1.585	1.892
4	0	2.0	0
5	1	2.322	0.431
6	2	2.584	0.774

So the DCG_6 of this ranking is:

$$\mathrm{DCG_6} = rel_1 + \sum_{i=2}^{6} \frac{rel_i}{\log_2 i} = 3 + (2 + 1.892 + 0 + 0.431 + 0.774) = 8.10$$

Normalized DCG

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p}$$

Search result lists vary in length depending on the query.

Comparing a search engine's performance from one query to the next cannot be consistently achieved using DCG alone.

The cumulative gain at each position for a chosen value of should be normalized across queries.

Ideal DCG (IDCG) at position is obtained by sorting documents of a result list by relevance, producing the maximum possible DCG till position p.

nDCG Example

$$D_1, D_2, D_3, D_4, D_5, D_6$$

the user provides the following relevance scores:

$$3, 2, 3, 0, 1, 2$$

$$3, 3, 2, 2, 1, 0$$

The DCG of this ideal ordering, or *IDCG*, is then:

$$\mathrm{IDCG}_6 = 8.69$$

And so the nDCG for this query is given as:

$$\mathrm{nDCG}_6 = \frac{DCG_6}{IDCG_6} = \frac{8.10}{8.69} = 0.932$$

Recommendation Outline

- Recommender Systems
  - What are recommender systems and how do they work?
  - Example application: Hotlist Recommendation on Delicious
  - How are recommender systems evaluated?

- **(Some) Recommendation challenges**
  - Well-known challenges
  - Recommendation diversity
  - Group recommendation

Well-Known Challenges

- The new user problem
- The recurring startup problem
- The sparse rating problem
- The scaling problem

The New User Problem

- To be able to make accurate predictions, the system must first learn the user's preferences from the input the user provides (e.g., movie ratings, URL tagging).
- If the system does not show quick progress, a user may lose patience and stop using the system

The Recurring Startup Problem

- New items are added regularly to recommender systems.
- A system that relies solely on users' preferences to make predictions would not be able to make accurate predictions on these items.
- This problem is particularly severe with systems that receive new items regularly, such as an online news article recommendation system.

The Sparse Rating Problem

- In any recommender system, the number of ratings already obtained is very small compared to the number of ratings that need to be predicted.
- Effective generalization from a small number of examples is thus important.
- This problem is particularly severe during the startup phase of the system when the number of users is small.

The Scaling Problem

- Recommender systems are normally implemented as a centralized algorithm and may be used by a very large number of users.
- Sometimes, predictions need to be made in real time and many predictions may potentially be requested at the same time.
- The computational complexity of the algorithms needs to scale well with the number of users and items in the system.

Recommendation Outline

- Recommender Systems
  - What are recommender systems and how do they work?
  - Example application: Hotlist Recommendation on Delicious
  - How are recommender systems evaluated?

- Recommendation challenges
  - Well-known challenges
  - **Recommendation diversity**
  - Group recommendation

Diversification

From the pool of relevant items, identify a list of items that are dissimilar to each other and maintain a high cumulative relevance, i.e., strike a good balance between relevance and diversity.

Existing Solutions

- **Attribute-based diversification in 3 steps:**
  - pair-wise item-to-item distance function on item attributes
  - Perform Diversification:
    - Optimize an overall score as a weighted combination of relevance and distance
    - Constrain either relevance or distance, maximizing the other
  - Overhead of retrieving item attributes

- **Explanation-Based Diversification**

Recommendation Strategy

- **Estimate the rating of an unrated item (*i*) by the user (*u*) based on its similarity to items already rated and how *u* rated those items.**

$$\texttt{relevance}(u, i) = \Sigma_{i' \in \mathcal{I}} \texttt{ItemSim}(i, i') \times \texttt{rating}(u, i')$$

- **Similarly, one could define a user-based strategy**

$$\texttt{relevance}(u, i) = \Sigma_{u' \in \mathcal{U}} \texttt{UserSim}(u, u') \times \texttt{rating}(u', i)$$

Explanation

- **Basic Notion**
  - The set of objects because of which a particular item is recommended to the user

- **Explanation for Item-Based Strategies**

$$\mathtt{Expl}(u, i) = \{i' \in \mathcal{I} \mid \mathtt{ItemSim}(i, i') > 0 \;\&\; i' \in \mathtt{Items}(u)\}$$

- **Explanation for User-Based Strategies**

$$\mathtt{Expl}(u, i) = \{u' \in \mathcal{U} \mid \mathtt{UserSim}(u, u') > 0 \;\&\; i \in \mathtt{Items}(u')\}$$

Explanation-Based Diversity

- **Pair-wise diversity distance between two recommended items**
  - Standard similarity measures like *Jaccard similarity* and *cosine similarity*
  - E.g. (Distance based on Jaccard similarity)

$$DD_u^J(i, i') = 1 - \frac{|\mathtt{Expl}_{(u,i)} \cap \mathtt{Expl}_{(u,i')}|}{|\mathtt{Expl}_{(u,i)} \cup \mathtt{Expl}_{(u,i')}|}.$$

- **Diversity for the set of recommended items (*S*)**

$$DD_u(S) = avg\{DD_u(i, i') \mid i, i' \in S\}$$

Diverse Recommendation Problem

Top-K Recommendation with Diversification

Given a user u, find a subset S from the set of candidate items, such that |S| = k and the overall relevance of items in S and the diversity of S are balanced.

Cong Yu, Laks V. S. Lakshmanan, Sihem Amer-Yahia: Recommendation Diversification Using Explanations. ICDE 2009: 1299-1302

Recommendation Outline

- Recommender Systems
  - What are recommender systems and how do they work?
  - Example application: Hotlist Recommendation on Delicious
  - How are recommender systems evaluated?

- Recommendation challenges
  - Well-known challenges
  - Recommendation diversity
  - **Group recommendation**

Group Recommendation (motivation)

- How do you decide where to go to dinner with friends?
  - email/text/phone
  - not optimal for reaching consensus
- What if there was a system that knew each user’s preferred list?
- What is the best way to model consensus?
- How to *evaluate* that?
- How to *efficiently* compute *group recommendations*?

Group Recommendation by Example

- **Task: recommend a movie to group G ={u1, u2 ,u3}**
  - predictedRating(u1,"God Father") = 5
  - predictedRating(u2, "God Father") = 1
  - predictedRating(u3, "God Father") = 1

  - predictedRating(u1, "Roman Holiday") = 3
  - predictedRating(u2, "Roman Holiday") = 3
  - predictedRating(u3, "Roman Holiday") = 1

- ***Average Rating*** **and** ***Least Misery*** **fail to distinguish between "God Father" and "Roman Holiday"**

Group Reco Problem Definition

Consensus function *combines* ***relevance*** *(average or least misery) and* ***disagreement*** *(average pair-wise or variance) in the score of a group recommendation*

$\mathcal{F}(\mathcal{G}, i) = w_1 \times \mathtt{rel}(\mathcal{G}, i) + w_2 \times (1 - \mathtt{dis}(\mathcal{G}, i))$, *where* $w_1 + w_2 = 1.0$ *and each specifies the relative importance of relevance and disagreement in the overall recommendation score.*

Problem: Given a user group G (formed on-the-fly) and a consensus function F, find the k best items according to F, such that each item is new to all users in G

S. Amer-Yahia, S. B. Roy, A. Chawla, G. Das, C. Yu: Group Recommendation: Semantics and Efficiency. VLDB 2009.

In practice

- Choose your similarity measure wisely, you will have to try more than one
- Define your goal early with the domain expert to determine how to evaluate your approach
- Build a prototype ASAP
- Use existing tools whenever possible

Main references

- Mining of Massive Datasets: A. Rajaraman and J. Ullman
- Overview of Recommendation Systems

http://web.stanford.edu/class/ee378b/papers/adomavicius-recsys.pdf

- Collaborative Filtering: Chapter 9 of Mining Massive Datasets book

http://infolab.stanford.edu/~ullman/mmds/book.pdf

- Delicious recommendations

J. Stoyanovich, S. Amer-Yahia, C. Yu, C. Marlow: Leveraging Tagging Behavior to Model Users' Interest in del.icio.us (AAAI Workshop on Social Information Processing 2008)

- Diverse recommendations

Cong Yu, Laks V. S. Lakshmanan, Sihem Amer-Yahia: Recommendation Diversification Using Explanations. ICDE 2009: 1299-1302

- Group recommendations

S. Amer-Yahia, S. B. Roy, A. Chawla, G. Das, C. Yu: Group Recommendation: Semantics and Efficiency. VLDB 2009.

- Evaluating recommender systems

http://essay.utwente.nl/59711/1/MA_thesis_J_de_Wit.pdf