UNRESYST

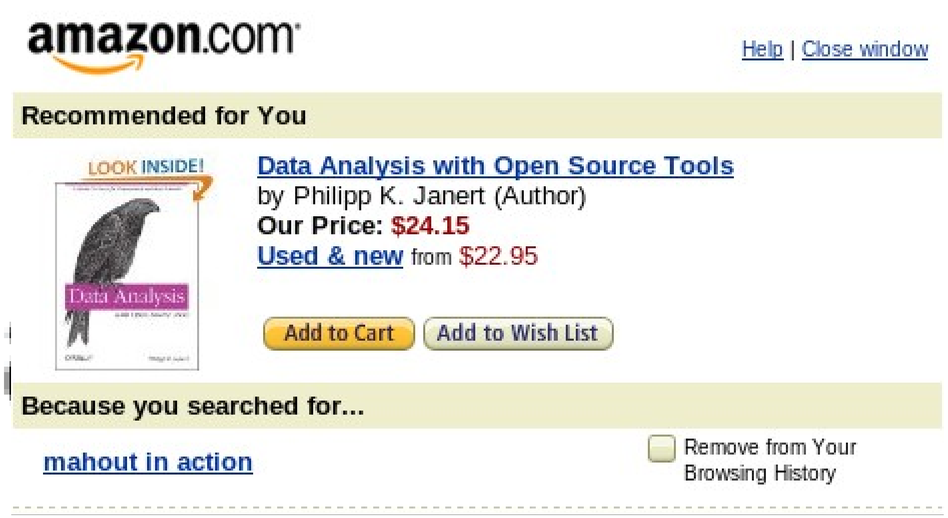
Master thesis presentation
8th March 2011

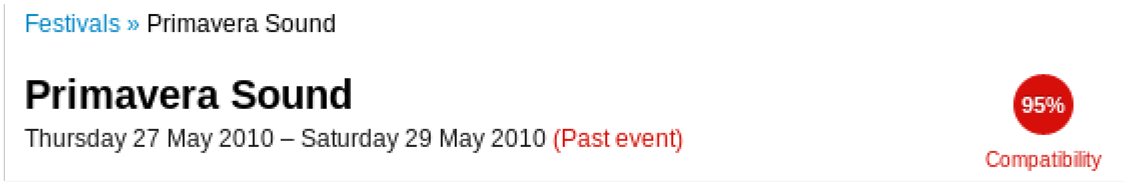
Presentation overview

1. Quick Introduction to Recommender Systems
2. Problem Analysis
3. System Design
4. Adaptation to Datasets
5. Challenges
6. Finale, Discussion

1. Intro
Recommenders

- User actions
  - Find items I will like
  - Find me novel items
  - Sort items by preference
  - Advise me on particular item





1. Intro:
Recommender Applicability

- Many items in the domain
- Choice based on taste
- Taste data
- Homogeneous items

1. Intro: Recommender as a Research Area

- Gathering user preference
- Algorithms transforming past user actions to recommendations
- Privacy, legacy and other aspects
- Measuring recommender efficiency
- Recommender system implementation

ACM Recommender Systems
?+?+?=!

GroupLens Research

1. Intro:
Recommender as a Research Area

- Gathering user preference
- Algorithms transforming past user actions to recommendations
- Privacy, legacy and other aspects
- Measuring recommender efficiency
- Recommender system implementation

ACM Recommender Systems
?+?+?=!

GroupLens Research

1. Intro
Thesis: Universal Recommender

Features:

- Domain Independence
- Using and combining multiple data sources
- Simple and developer-friendly interface
- Verification on various domains

Thesis type:

- Implementace
- Výzkumný problém
- Analýza a návrh řešení zadaného problému
- Srovnávací studie

2. Analysis
Implementing Recommender in a Web System

Options for a system holder:

(a) Implement their own recommendations

(b) Use a recommender framework

(c) Use the Universal Recommender System

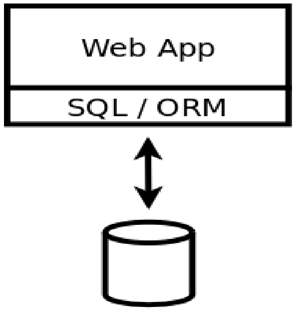
(d) (Use a Google API)

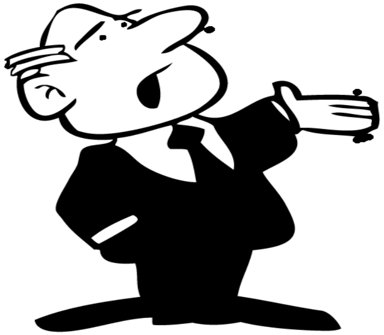
Product ID # \ Customer ID #	1	2	3	4	5
1	0	0	5	0	10
2	3	2	0	20	0
3	8	5	1	40	1
4	0	1	10	2	10
5	1	0	0	1	1

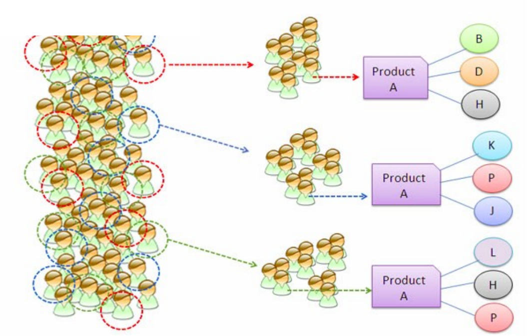
WEKA The University of Waikato

mahout

$$w_{a,u} = \frac{\sum_{i=1}^{M}(r_{a,i} - \bar{r}_a)(r_{u,i} - \bar{r}_u)}{\sigma_a \sigma_u}$$







2. Analysis
(a) Implement Their Own Recommendations

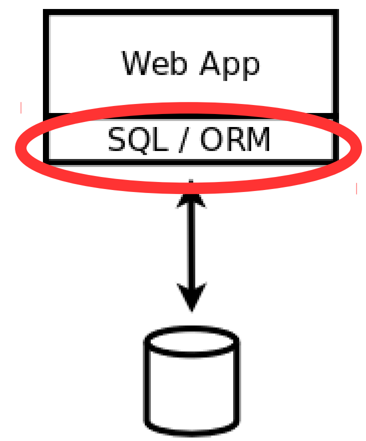
Benefits and drawbacks:

+ no framework needed

+ efficient for simple filtering

- maintenance

- modern algorithms have to be implemented if needed



2. Analysis
(b) Use a Recommender Framework

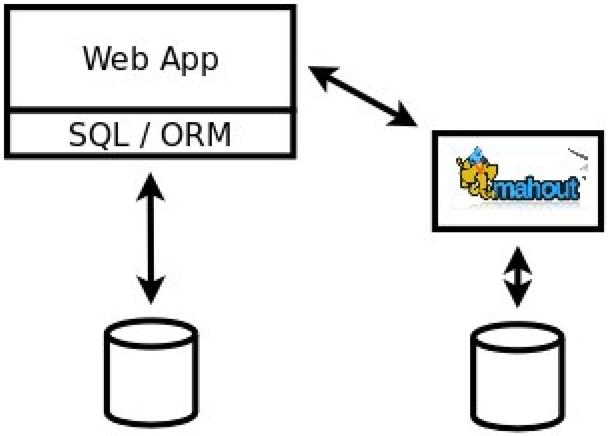
Benefits and drawbacks

+ variety of modern algorithms

+ scalable implementations

- incorporating the framework to the system (DB setup, interface adapter)

- a single source of preference data



2. Analysis
(c) Use the Universal Recommender System

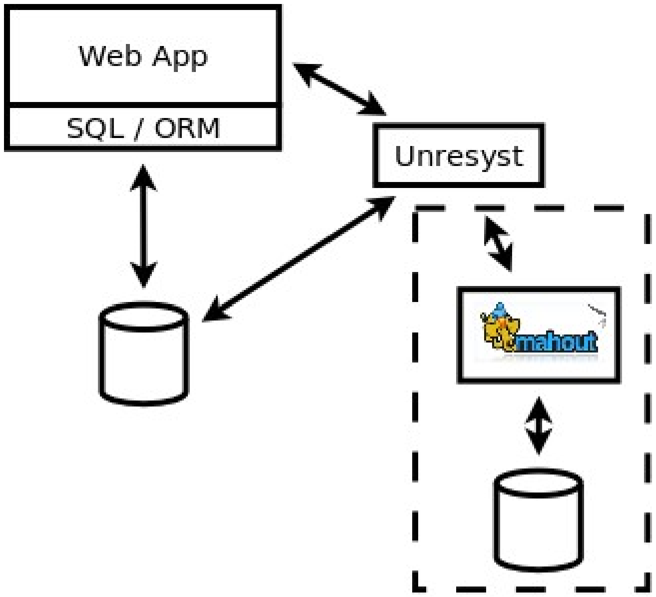
Benefits and drawbacks

+ easy-to-use interface

+ can share the system DB

+ can use external algorithm implementation

- needs setup



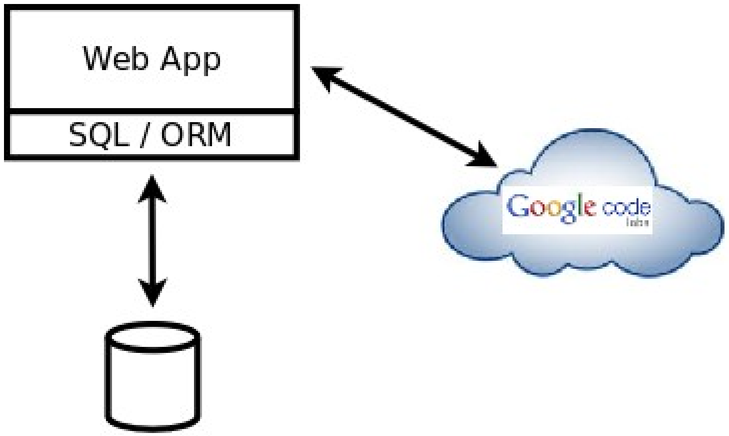
2. Analysis
(d) (Use the Google Prediction API)

Benefits and drawbacks

? little information available

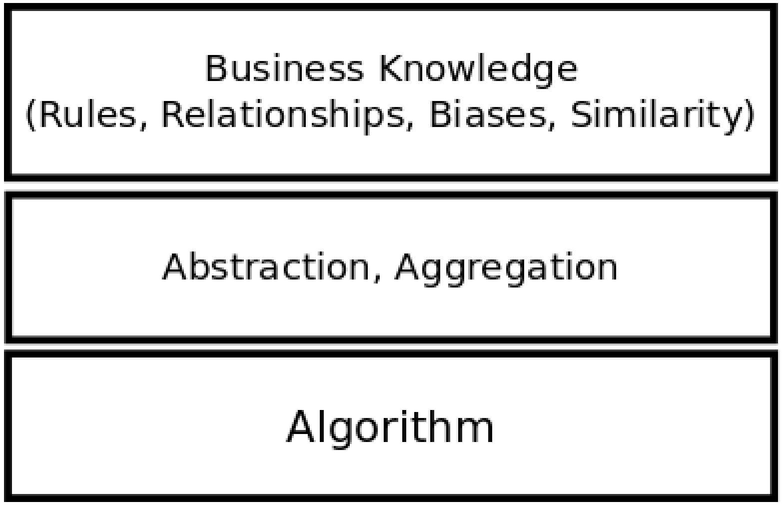
? access on request through a waiting list

- passing customer data to Google

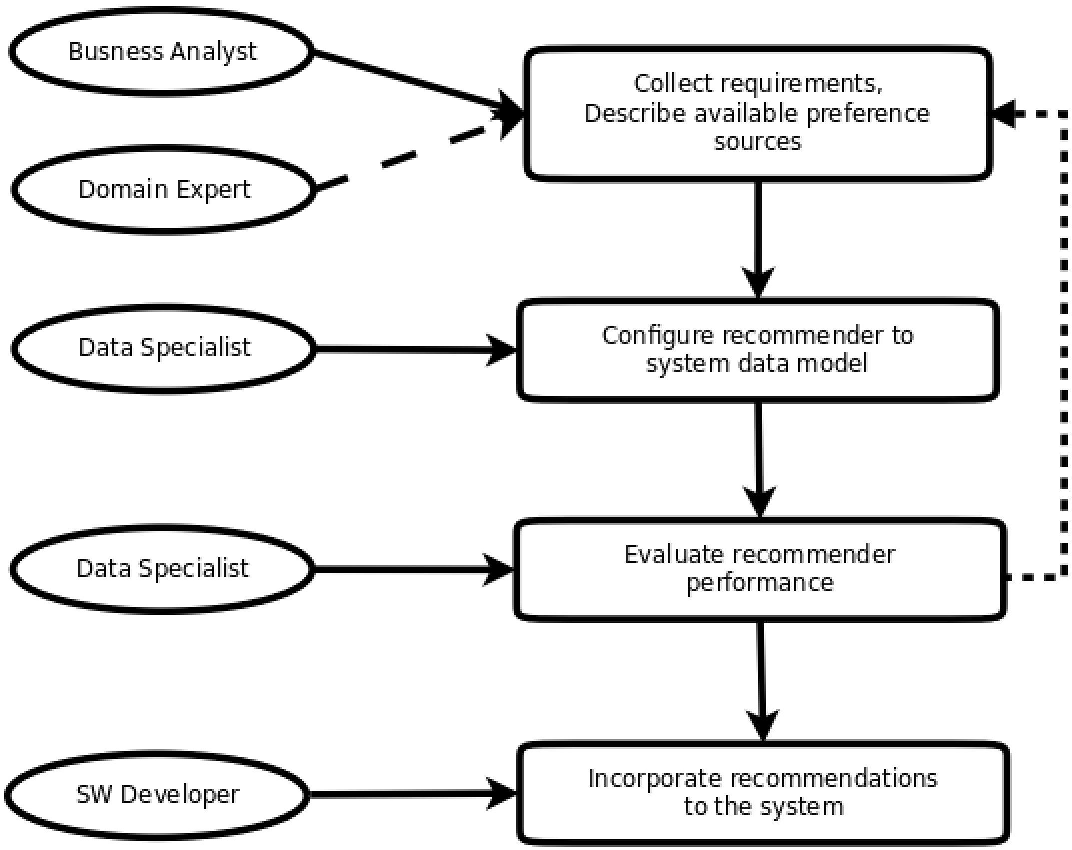


2. Analysis
UNRESYST Added Value

- Multiple data sources and their combination
- Multiple recommenders in one system
- Using both implicit and explicit feedback, not just rating
- Domain specific rules
  (e.g. no double GPS purchase)
- Isolating business knowledge
  from recommender algorithms



2. Analysis
The Process of Implementing a Recommender

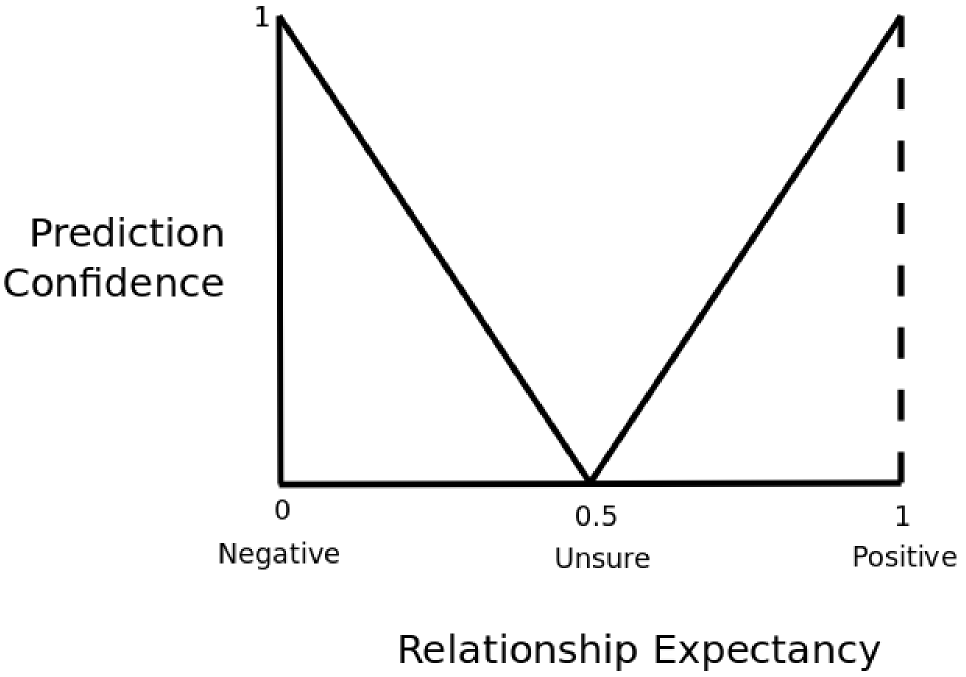


3. UNRESYST Design
Interface

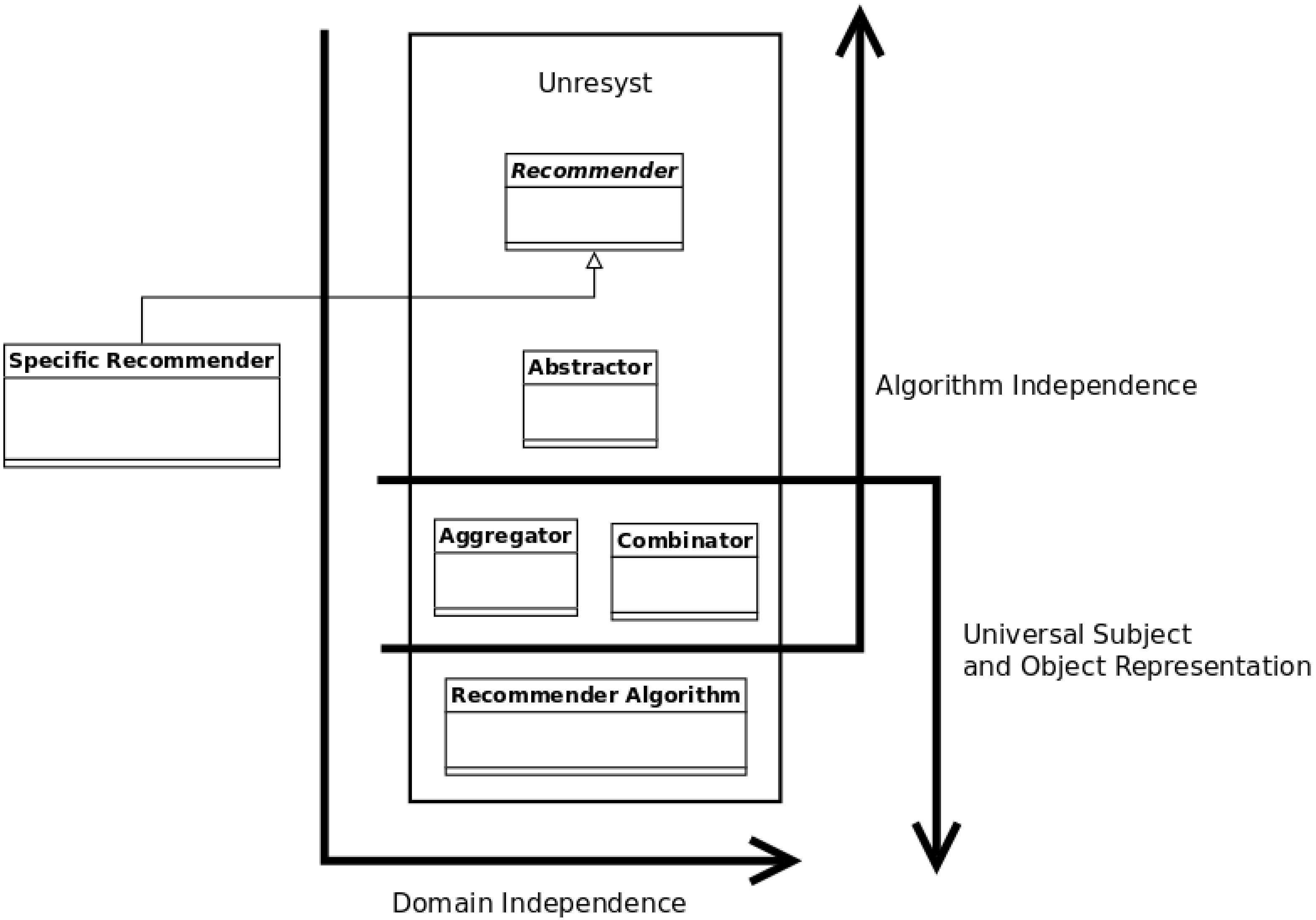
- Adaptation :
  - subjects, objects, predicted relationship
  - rules and relationships
  - clusters
  - bias
- Runtime:
  - build
  - update
  - predict, recommend – using system classes

3. UNRESYST Design
Defining rules

- Rule semantics:
  - Preference
  - Similarity
- Parameters
  - Generator/Condition
  - Positiveness
  - Confidence
  - Name, description

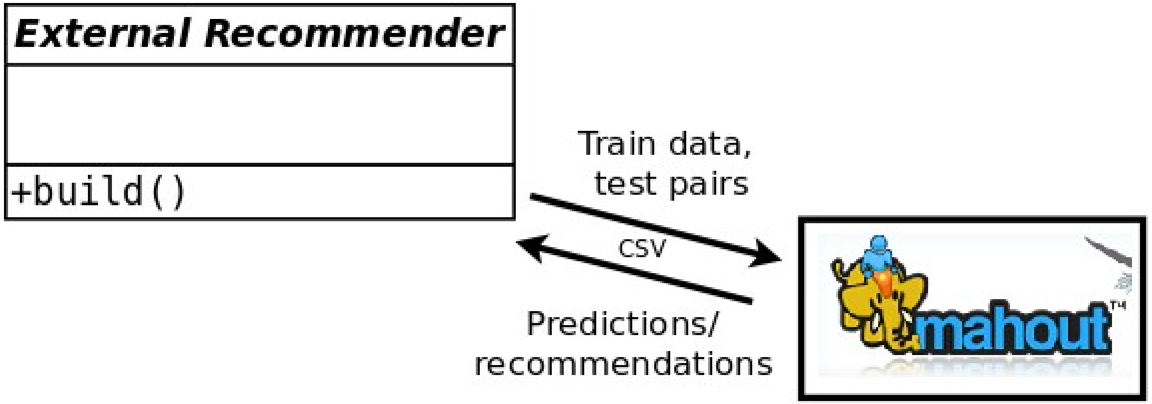


3. UNRESYST Design
Architecture - Layers



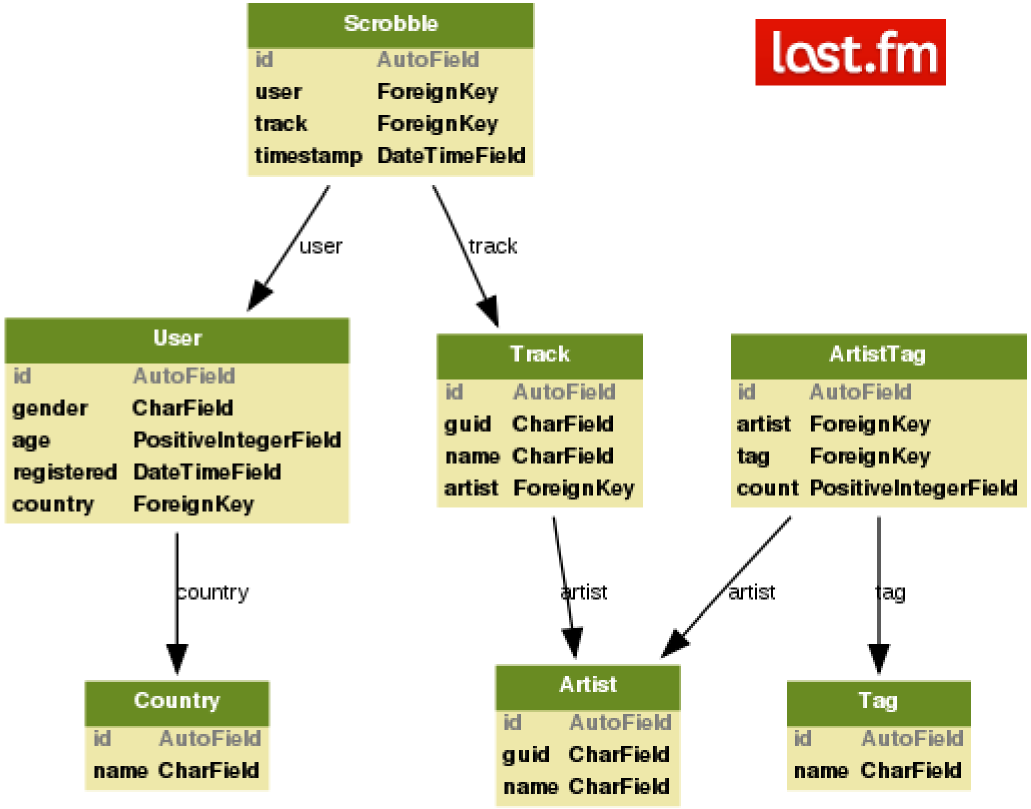
4. Adaptation to Datasets, Comparison Overview

- Adaptation
  - Create data model, import data from csv file (or its subset)
  - Configure Unresyst
  - Divide data into train and test set
  - Build the recommender with the train set
  - Run evaluation
- Comparison
  - Mahout implementation of collaborative filtering algorithms



4. Adaptation to Datasets, Comparison
Last.fm Data Set

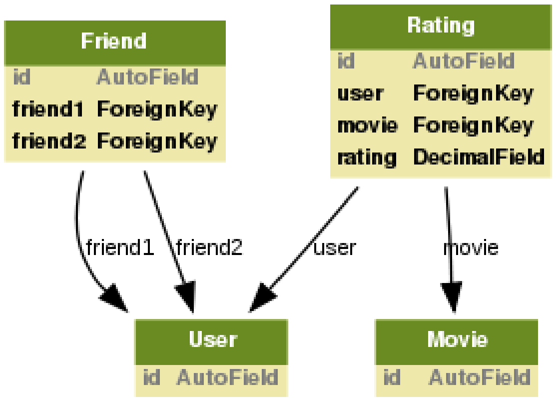
- Users listening to Tracks by Artists
  - Originally 20 mil. scrobbles, 1000 users
  - Reduced to 100 users, 6000 scrobbles
- Artist social tags
  - Available for 10% of artists
- Recommending artists to users



- http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html
- http://musicmachinery.com/2010/11/10/lastfm-artisttags2007/

4. Adaptation to Datasets, Comparison Flixster Data Set

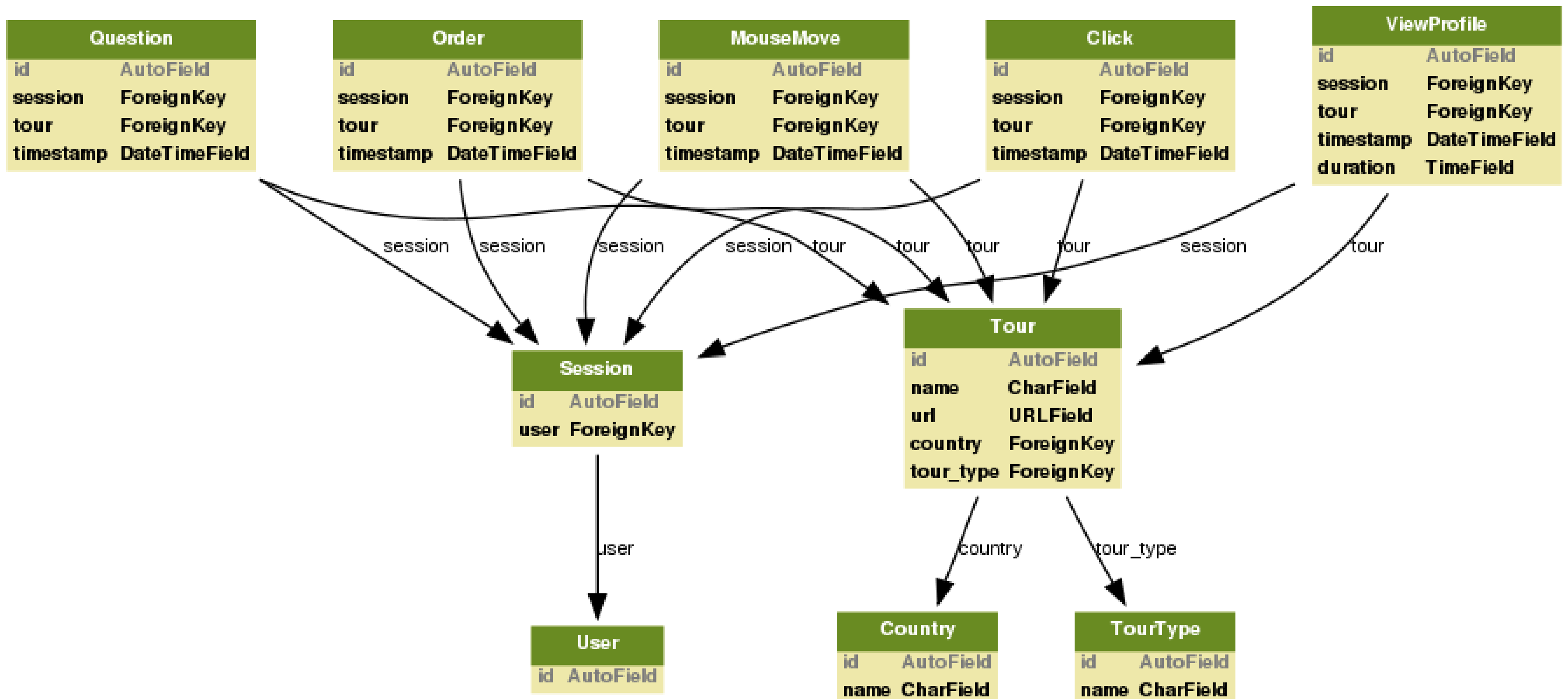
- Users rating movies
  - Originally 8 mil. ratings
- Social links between users
  - Originally 7 mil. links
- Classical collaborative filtering data set extended by social links
- No timestamps available



- http://www.cs.sfu.ca/~sja25/personal/datasets/

4. Adaptation to Datasets, Comparison
Travel Agency Data Set

- Users viewing and ordering tours
- Various kinds of implicit feedback



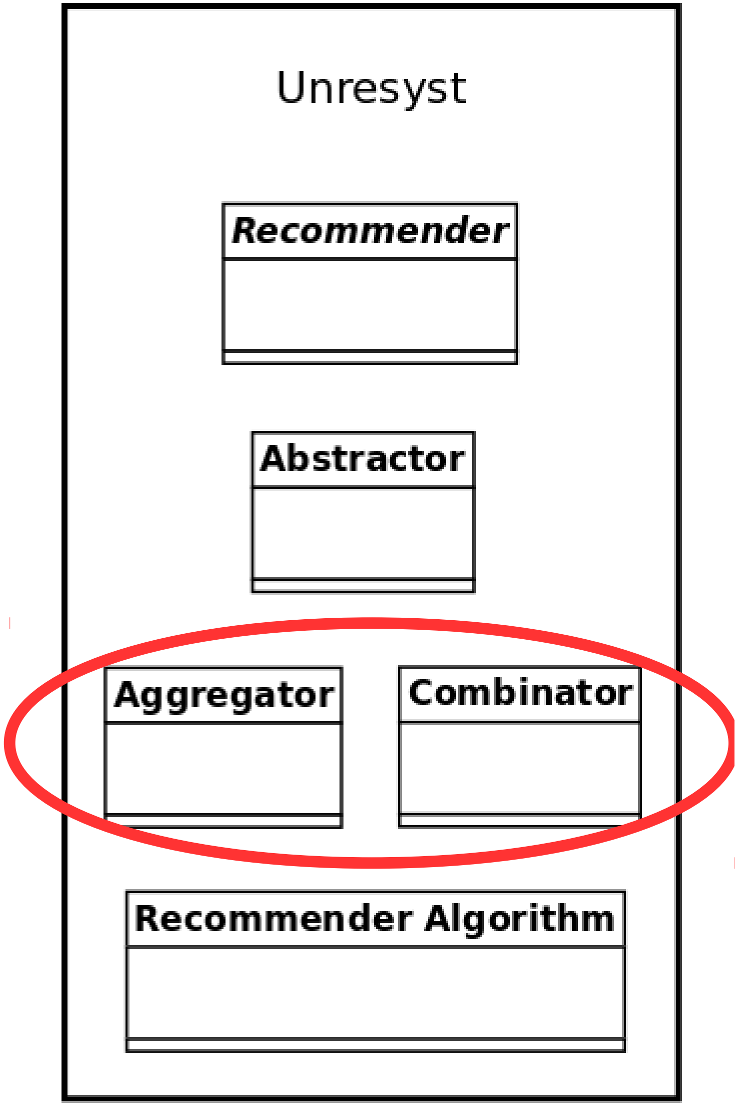
5. Challenges
Preference/Similarity/Bias combination

Combining Expectancies

- Aggregator:
  - Subject, object similarity, clusters
  - Subject, object biases

  Output: entity pair similarity, entity bias
- Combinator:
  - Preference rules and relationships
  - Aggregated biases, similarity

  Output: preference prediction



5. Challenges Preference/Similarity/Bias combination

- Expectancy: probability estimated by an isolated rule

C_{so}: event, s chooses o
E_{Rso} event, rule R covers s - o prediction

$$P(C_{so}|E_{Rso}) = \frac{P(E_{Rso}|C_{so})P(C_{so})}{P(E_{Rso})}$$

$CPair$: a set of s-o pairs, s has chosen o
$Pair_R$: a set of s-o pairs, R predicts s-o expectancy
S: a set of subjects
O: a set of objects
$P(C_{so}) \simeq$ expectancy given by R
$P(E_{Rso}) \simeq \frac{|Pair_R|}{|S||O|}$
$P(E_{Rso}|C_{so}) \simeq \frac{|CPair \cap Pair_R|}{CPair}$

- Multiple rules applied to an s-o pair:

$$P(C_{so}|E_{R_1so} \cap E_{R_2so}) = \frac{P(C_{so})P(E_{R_1so}|C_{so})P(E_{R_2so}|C_{so} \cap E_{R_1so})}{P(E_{R_1so})P(E_{R_2so}|E_{R_1so})}$$

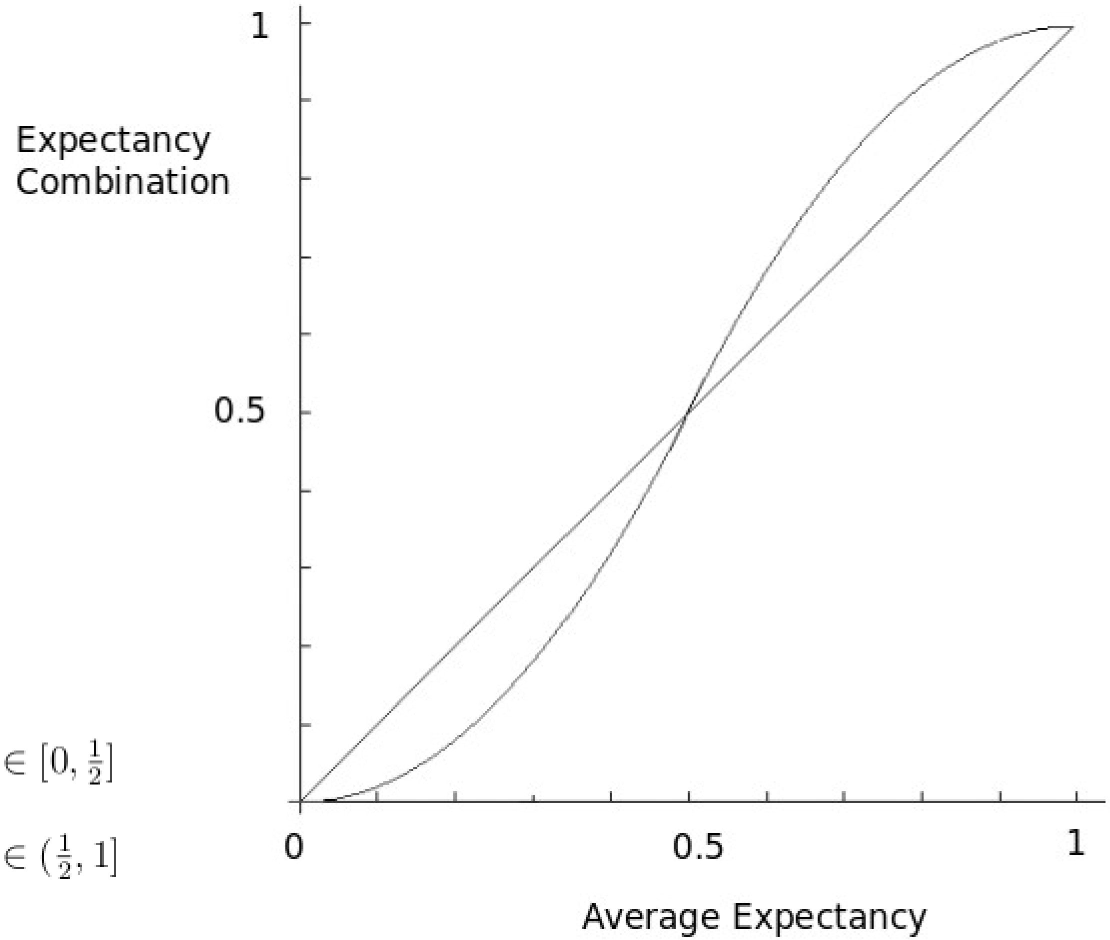
Rules taken as independent?

5. Challenges
Preference/Similarity/Bias combination

Combining preferences

- Intuitively:
  - Positive + Negative
    => Neutral
  - Positive + Positive
    => More positive
  - Negative + Negative
    => More negative

$$f(x) = \begin{cases} 2^{n-1}x^n & \text{if } x \in [0, \frac{1}{2}] \\ 1 - 2^{n-1}(x-1)^n & \text{if } x \in (\frac{1}{2}, 1] \end{cases}$$



5. Challenges
Efficiency Measure: Prediction

- Trying to predict the expectancies of the pairs in the test set
- Root Mean Square Error

$$RMSE = \sqrt{\frac{\sum_{(s,o)\in TestPairs}(p_{so} - \widetilde{p}_{so})^2}{|TestPairs|}}$$

p_{so}: preference of s to o taken from the test set
$\widetilde{p}_{so}$: prediction of the preference of s to o

+ widely used

+ large errors highly increase the resulting error

- only for datasets where negative feedback is also available

5. Challenges
Efficiency Measure: Recommendation

- Generating N recommendations for each subject, evaluating how many of them appear in test pairs
- Precision/Recall

$$Prec_s = \frac{|R_s \cap T_s|}{N} \qquad Rec_s = \frac{|R_s \cap T_s|}{|T_s|}$$

+ using only first N objects (relative)

+ more intuitive

- scaling (different object count for subjects in test set)

- multiple appearance of an object in the test set

- not a single number

- rarely used for evaluating recommender systems

$Prec_s$: precision for subject s.
Rec_s: recall for subject s
R_s: a set of objects recommended to s
T_s: a set of objects in test set for s
N: number of recommended objects

6. Finale
Questions, discussion

6. Finale
Contacts, References

- Unresyst project homepage:
  - http://code.google.com/p/unresyst/
- Recommender system leaders:
  - http://www.amazon.com/
  - http://www.last.fm/
  - http://news.google.com/
- Open source recommender algorithm libraries
  - http://mahout.apache.org/
  - http://www.cs.waikato.ac.nz/ml/weka/
  - http://duineframework.org/

Thanks for your attention.