UNRESYST

Universal Recommender System
30th May 2011

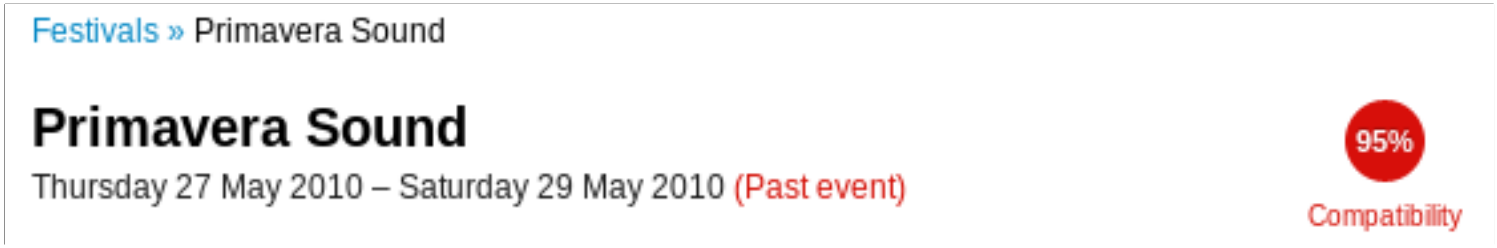
Presentation Overview

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

1. Intro
Recommenders

last.fm Google news Flixster

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



1. Intro:
Recommenders 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

GroupLens Research

ACM Recommender Systems

?+?+?=!

2. Analysis: Chosen Research Areas

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

Thesis type:

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

2. Analysis Requirements

Features:

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

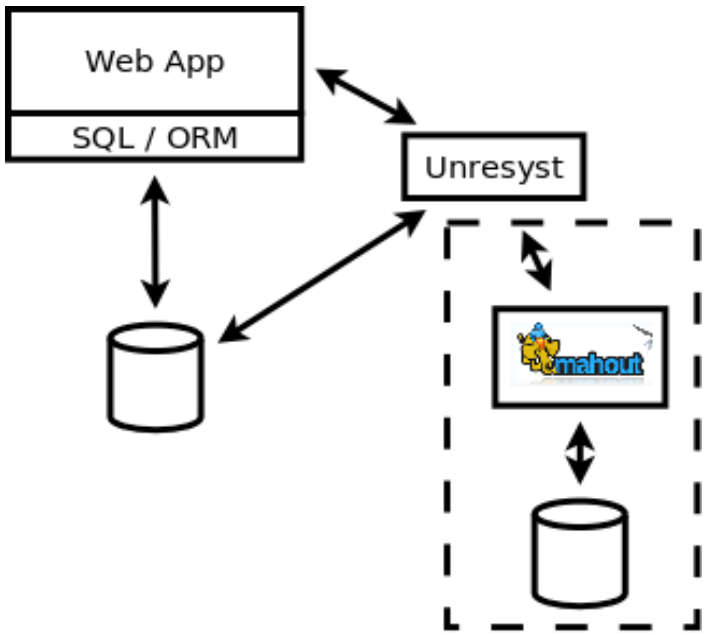
Adaptation :

- subjects, objects
- rules and relationships
- predicted relationship
- bias

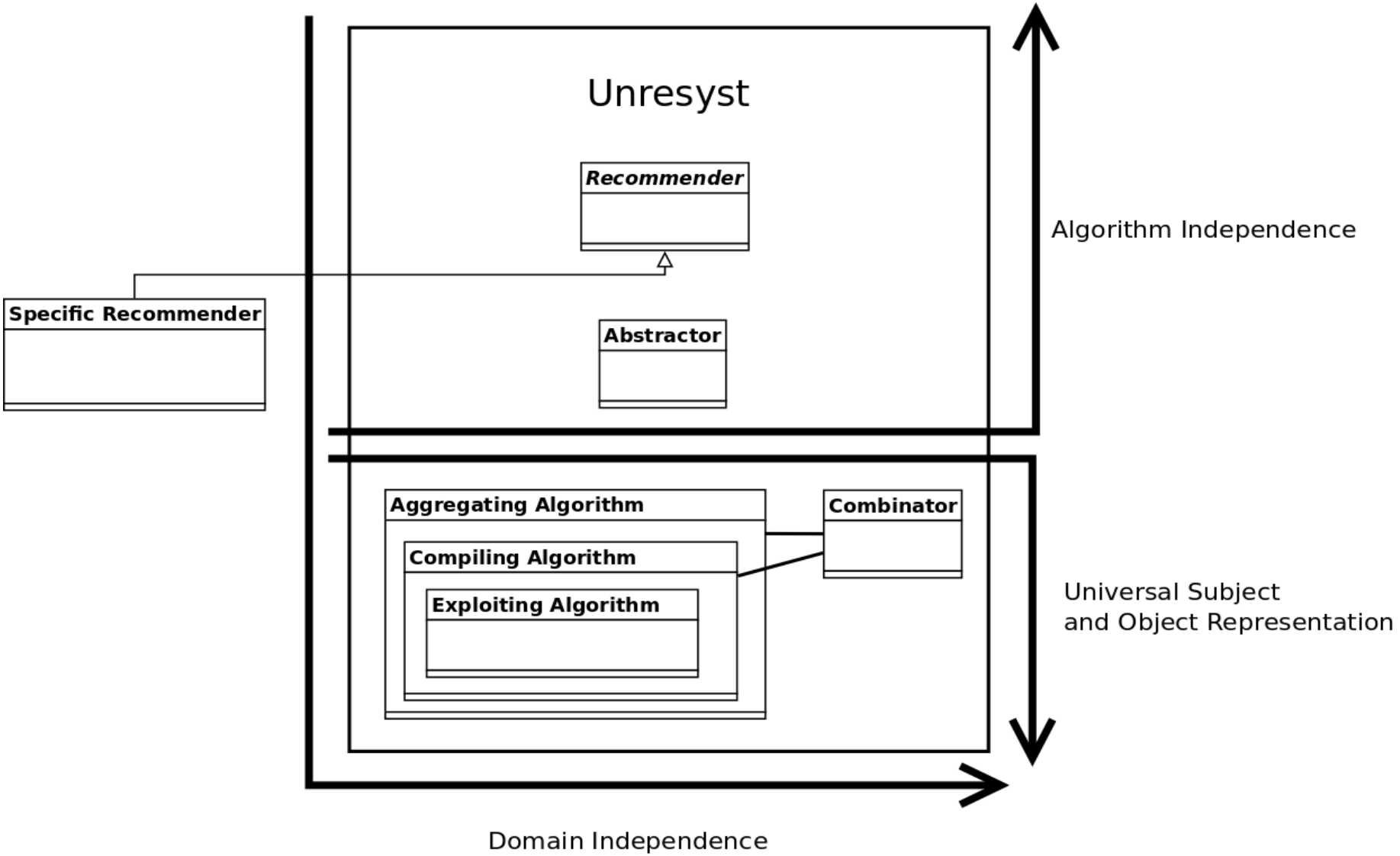
3. Design
Unresyst

Architecture concepts:

- outside the client system
- easy-to-use interface
- can share parent system DB
- can use external algorithm implementation
- needs setup



3. Design
Architecture - Layers

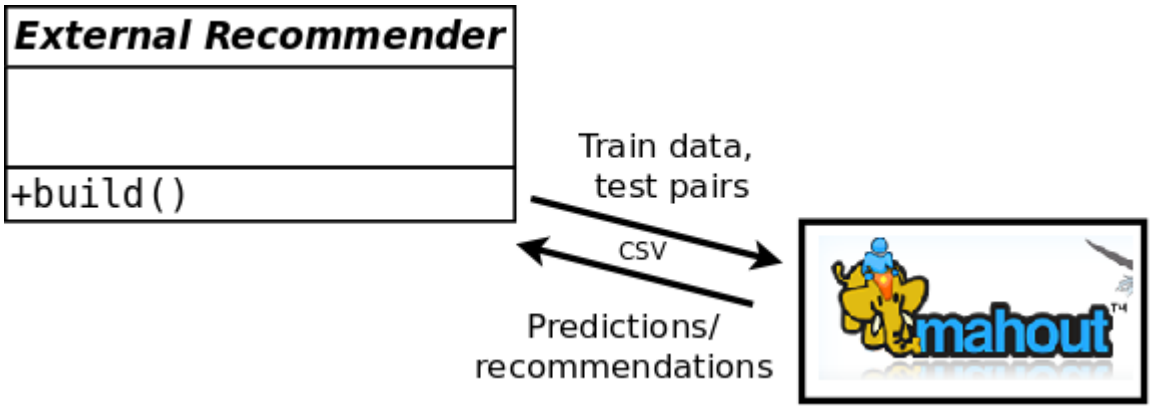


4. Adaptation to Datasets, Comparison

Offline accuracy analysis:

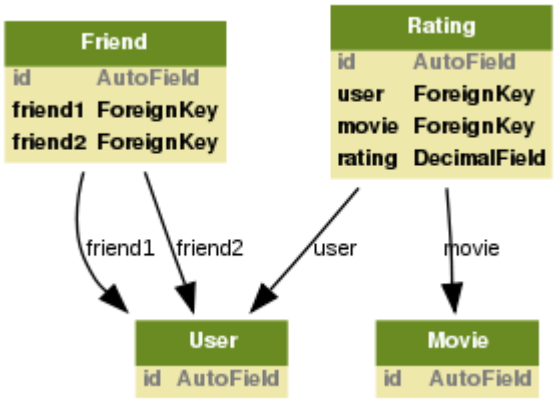
- Divide data to train and test set (timestamp).
- Run recommender on train set.
- Validate the recommendations against test set values, using RMSE and Rank metrics

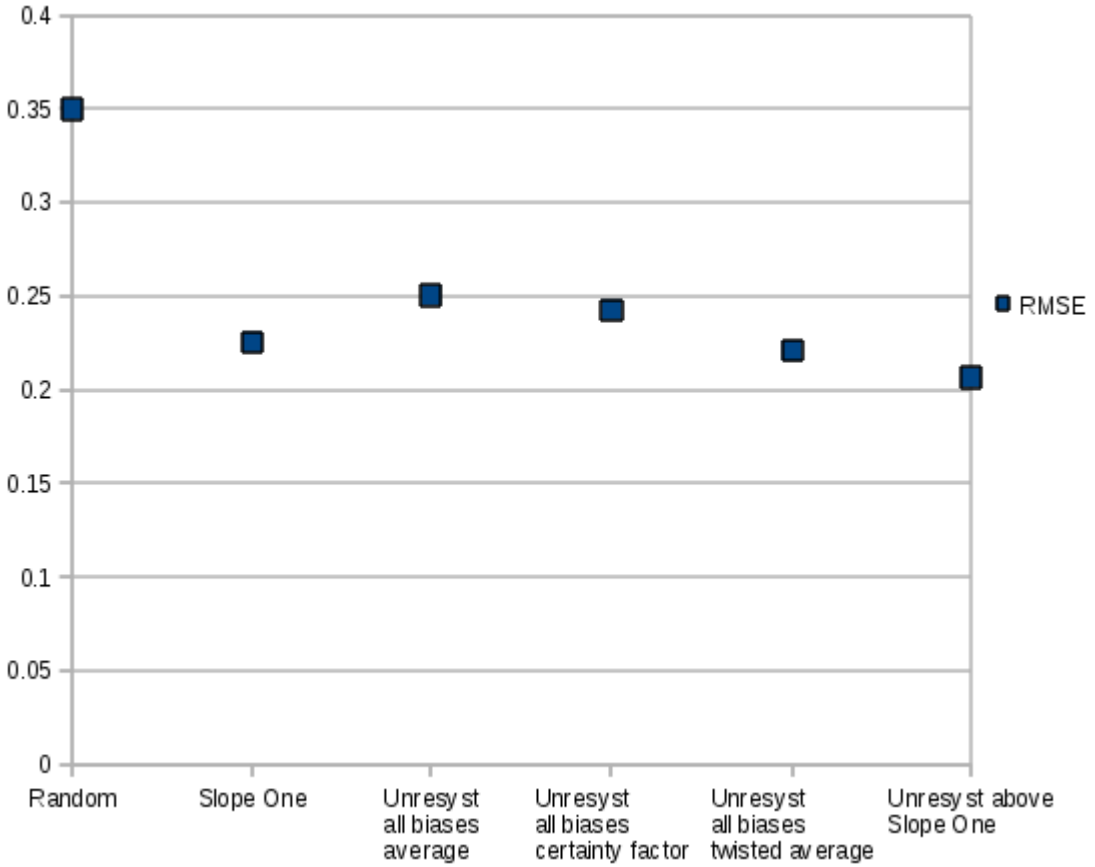
Comparison to a collaborative filtering algorithm (slope one).



4. Adaptation to Datasets, Comparison Flixster Data Set

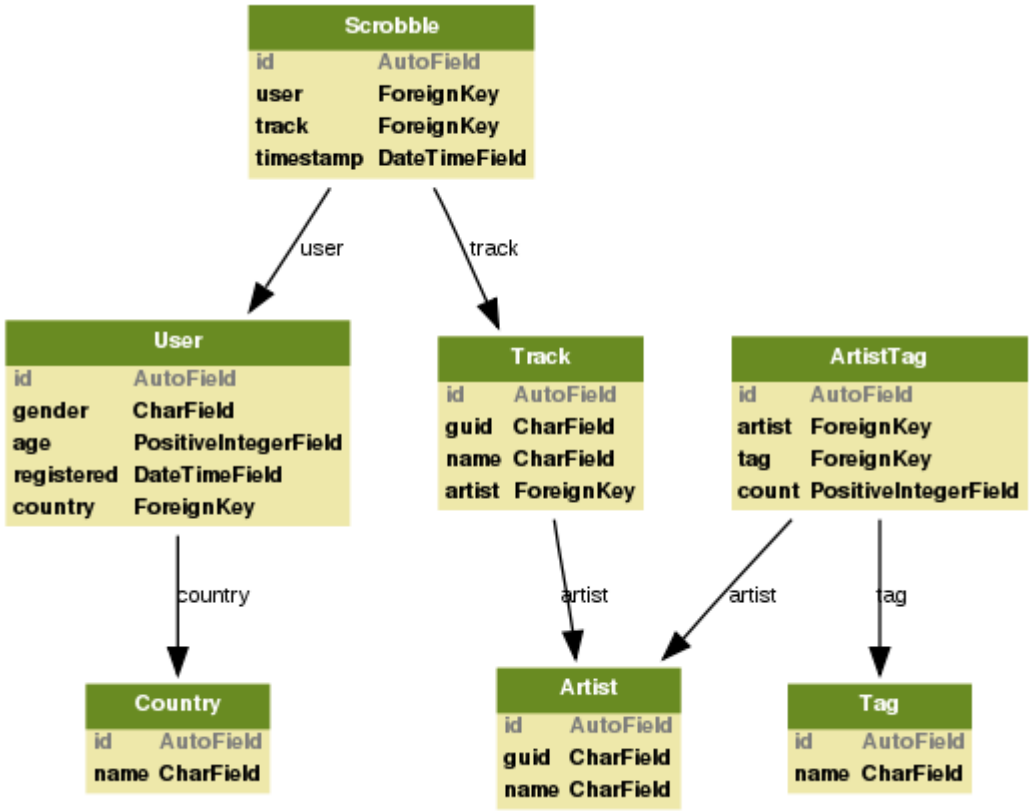
- Users rating movies
- Social links between users

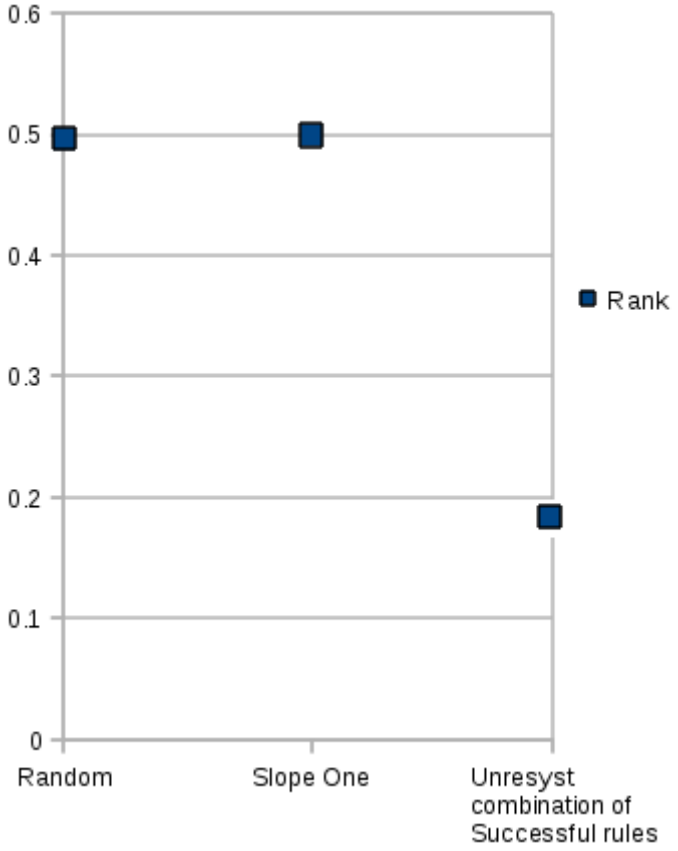




4. Adaptation to Datasets, Comparison
Last.fm Data Set

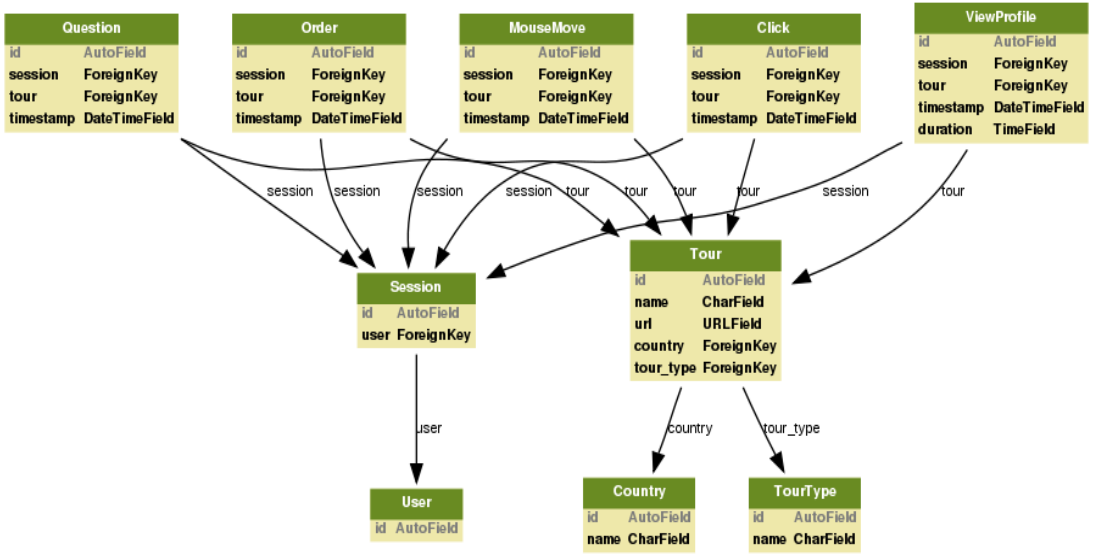
- Users listening to Tracks by Artists
- Recommending Artists to Users

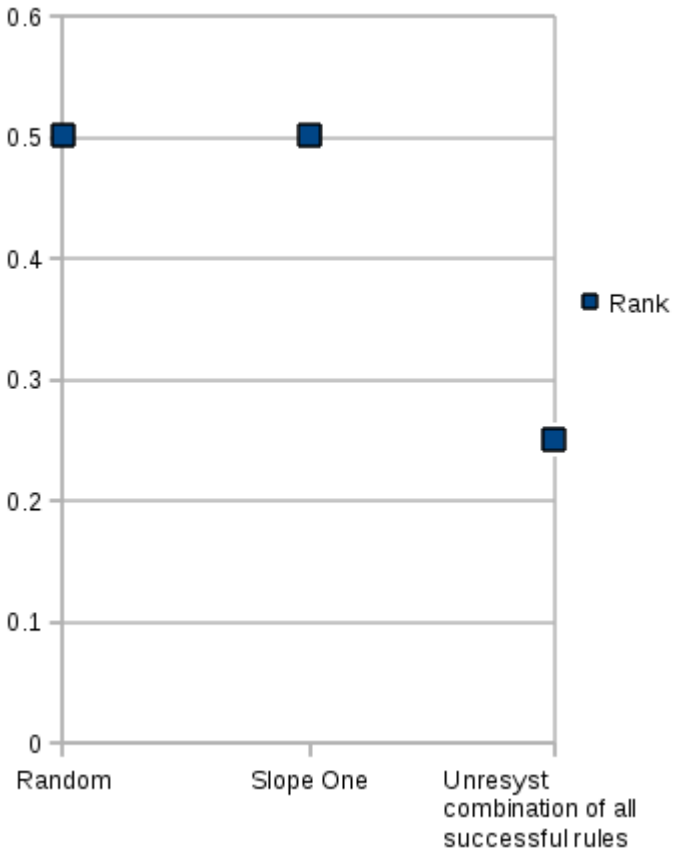




4. Adaptation to Datasets, Comparison
Travel Agency Data Set

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





5. Example Use
Unresyst Setup (Adaptation)

- Creating a recommender

```
class ArtistRecommender(Recommender):
    """A recommender recommending artists (musicians) that
    the user can like.
    """

    name = " Artist Recommender"
    """The name"""

    subjects = User.objects
    """The subjects to who the recommender will recommend."""

    objects = Artist.objects
    """The objects that will be recommended."""

    predicted_relationship = PredictedRelationship(
        name="User listens to artist's tracks.",

        # gives true for user, artist pairs where the user have listened
        # to the artist
        condition=lambda user, artist: \
            user.scrobble_set.filter(track__artist=artist).exists(),

        description="""User %(subject)s listens to the %(object)s's tracks."""
    )
    """The relationship that will be predicted"""
```

5. Example Use
Unresyst Setup (Adaptation)

- Adding rules and relationships

```
class ArtistRecommender(Recommender):
    # ...

    # the class contains definitions for business rules
    rules = (

        # don't recommend artists with male-specific tags to females
        SubjectObjectRule(
            name="Don't recommend male music to female users.",

            # the user is a female and the artist was tagged by
            # a male-specific tag
            condition=lambda user, artist: user.gender == 'f' and \
                artist.artisttag_set.filter(tag__gender_specific='m').exists(),

            # it's a negative rule
            is_positive=False,

            weight=0.5,

            # the more male-specific tags the artist has, the higher is
            # the rule confidence. Normalized by the artist tag count
            confidence=lambda user, artist: float(
                artist.artisttag_set.filter(tag__gender_specific='m').count())/ \
                    artist.artisttag_set.count(),

            description="Artist %(object)s isn't recommended to %(subject)s, " +
                "because the artist is considered male-specific."
        ),
```

5. Example Use
Using the Runtime Recommender API

- Building a recommender and getting a recommendation

```
>>> from lastfm.models import *
>>> from lastfm.recommender import *
>>>
>>> NovelArtistRecommender.build()
>>> u = User.objects.get(id=44)
>>> NovelArtistRecommender.get_recommendations(u,1)
[< user_44 <- Cat Stevens: 0.750000,
Similarity: user_44's gender is f. user_2's gender is f.
And: User user_2 listens to the Cat Stevens's tracks. >]
```

- Getting a relationship prediction

```
>>> u2 = User.objects.get(id=78)
>>> a = Artist.objects.get(id=32)
>>> NovelArtistRecommender.predict_relationship(u2, a)
< user_78 <- Boards Of Canada: 1.000000, User user_78 listens
to the Boards Of Canada's tracks. >
```

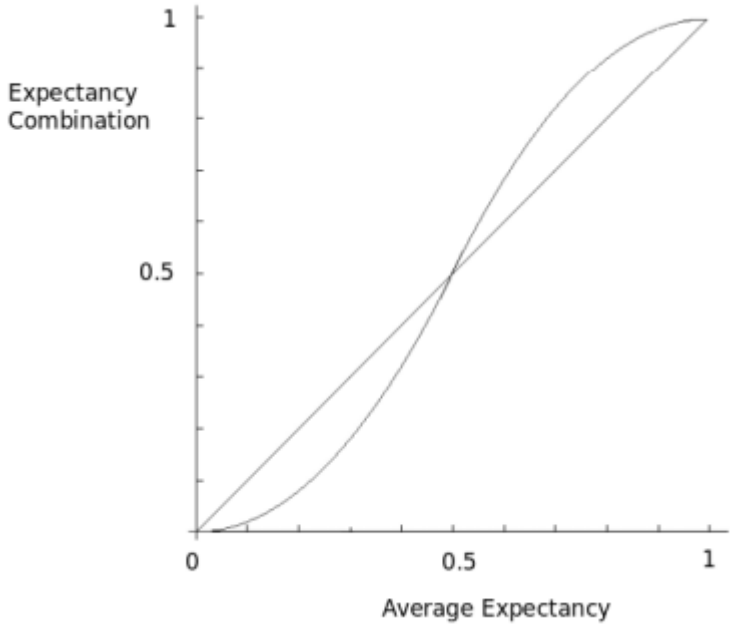
Full example source code available at:
http://code.google.com/p/unresyst/source/browse/trunk/code/adapter/recommender_example.py

6. Finale

Thanks for your attention.

Unresyst project homepage: http://code.google.com/p/unresyst/

BACKUP
Combination – Twisted average



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

BACKUP
Combination – Mycin

$$t(x) = 2x - 1$$

$$t^{-1}(x) = \frac{x+1}{2}$$

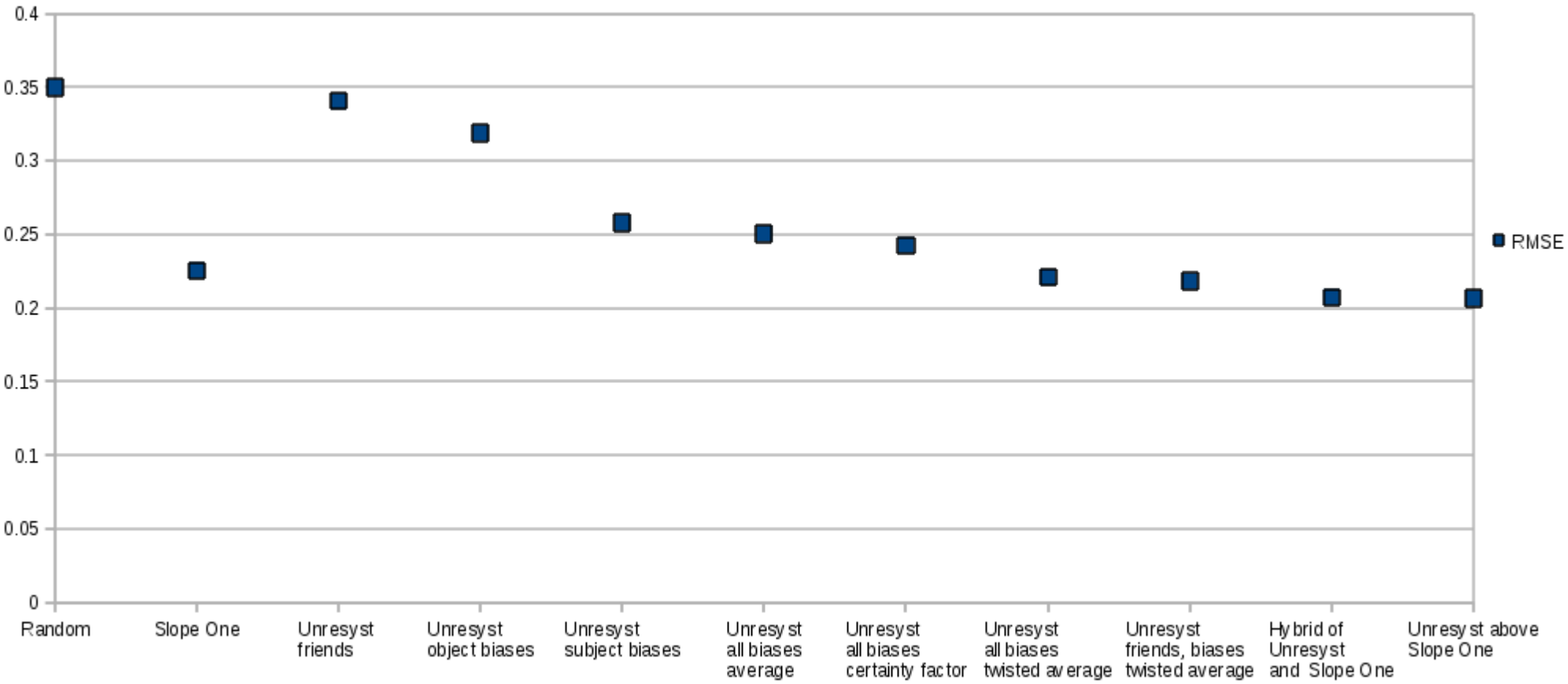
$$CFC(cf_1, cf_2) = \begin{cases} cf_1 + cf_2(1 - cf_1) & \text{if } cf_1, cf_2 > 0 \\ \frac{cf_1 \ cf_2}{1 - min(|cf_1|, |cf_2|)} & \text{if } -1 < cf_1 cf_2 \leq 0 \\ cf_1 + cf_2(1 + cf_1) & \text{if } cf_1, cf_2 < 0 \end{cases}$$

BACKUP
Measuring Efficiency – Average Rank

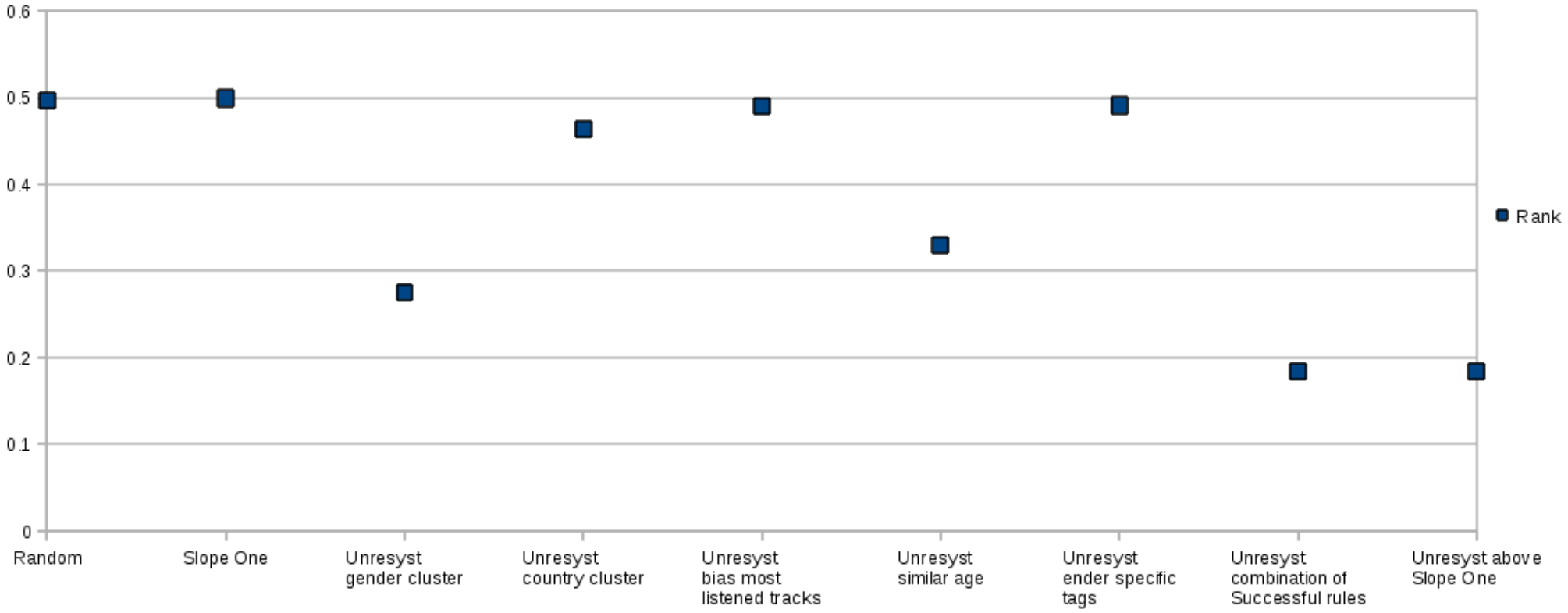
$$\overline{rank} = \frac{\sum_{(s,o)\in T} rank_{so}}{|T|}$$

$$rank_{so} = \frac{i_{so}}{m-1}$$

BACKUP
Flixster



BACKUP
Last.fm



BACKUP
Travel Agency