Tutorial: User Model Enrichment using the Social and Semantic Web

Federica Cena, University of Torino;
Eelco Herder, L3S Hannover

ESWC
30 May 2016

1 INTRODUCTION

2 PERSONALISATION

3 USER MODEL

4 USER MODELING

5 RECOMMENDATION

6 OVERVIEW

7 SOURCE OF WEB USAGE DATA

8 USER MODEL STRUCTURE AND INTEROPERABILITY

9 ENRICHING USER PROFILES WITH THE WEB OF DATA

10 ENRICHING USER MODELS WITH THE SOCIAL WEB

Tutorial goals

The goals of the tutorial are to understand

- User Modeling process, the methods, techniques and approaches involved and how they changed along with the changes in the Web
- the role of the Social and Semantic Web in the enrichment of user model.

We are both experts on user modeling, not on semantic techniques, thus we will mainly present applications of semantics to UM.

Tutorial outline

1. a brief historical overview on the research in the area of User Modeling, focusing on the traditional methods and techniques for representing, reasoning:
2. deeper insights on data-oriented aspects of the User Modeling process;
3. hands-on exercise

User Modeling: definition

- User Modeling is the process of getting to know the user.
- User Modeling is a cross-disciplinary research topic that can be studied from different perspectives and disciplines: from human-computer interaction to artificial intelligence, from psychology to philosophy of mind.

UM in Human-Computer Interaction

Cognitive Models:

- used by researchers for describing different typologies of users
- exploited by the interface designers to design effective user interfaces

UM in Artificial Intelligence

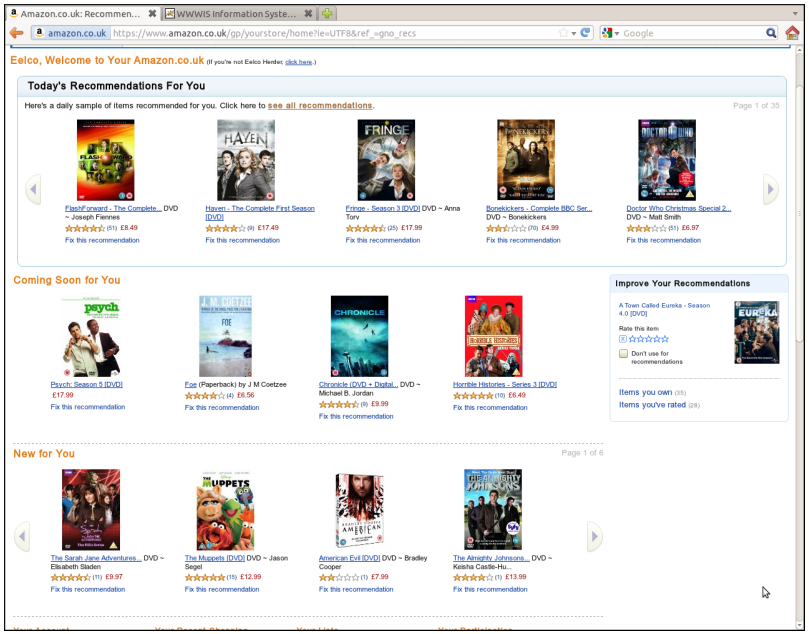
Algorithms and methods for:

- creating digital representations of users,
- inferring knowledge about the user based on past and present interaction,
- using these models for adapting the interface or the content

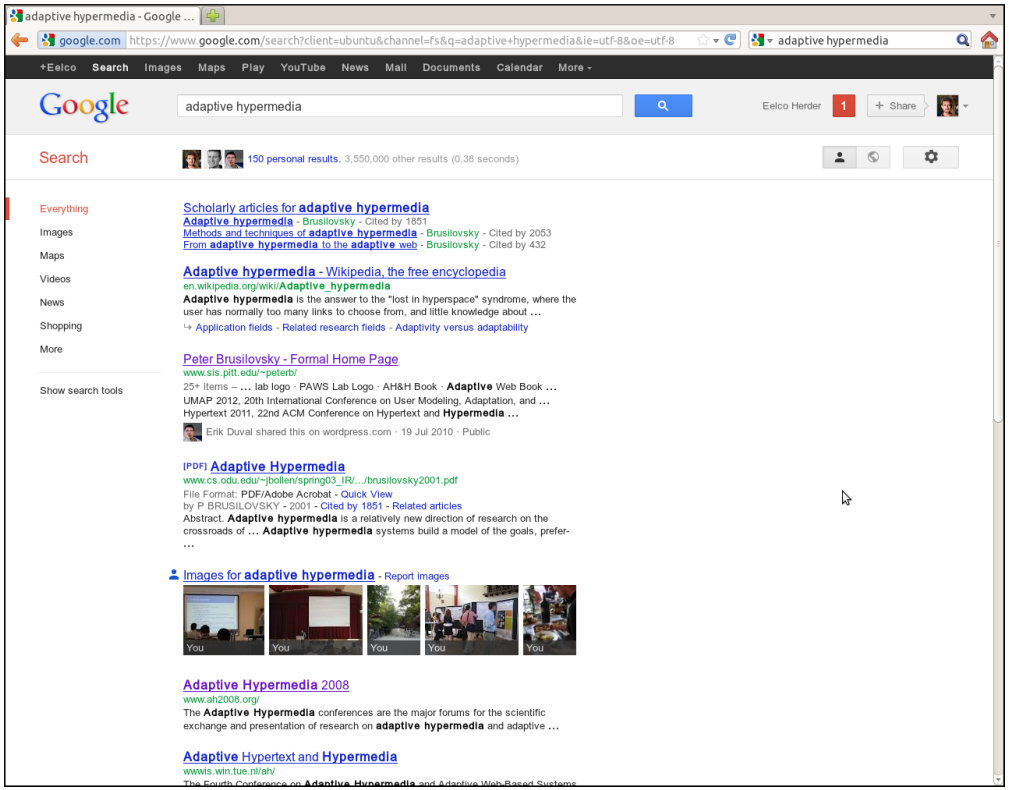
Personalized systems maintain a model of the user and then use it for adapting themselves to the user. Recommender systems are the most well-known type of such systems.

USER MODELING FOR PERSONALISATION

User modeling for personalization



Product recommendations in Amazon. These recommendations are based on past purchases and past browsing behavior. The user can improve the recommendations by editing his or her user profile.



Google search results are personalized, based on past searches, current location, language settings (apparently 57 features in total).

A formal definition

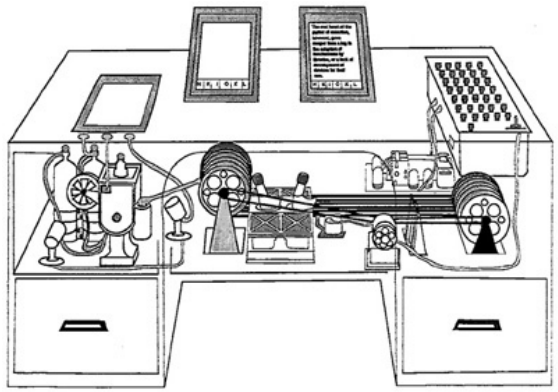
Adaptive Hypermedia

By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in a user model and apply this model to adapt various visible aspects of the system to the user.

Peter Brusilovsky: Methods and Techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction 6 (2-3), 1996

Adaptive hyperwhat?

In 1945, Vanevar Bush envisioned a machine, the *memex*. By consulting several sources consecutively, a user builds an associative *trail* of related documents, which can be labeled and annotated with notes and comments.



Similar to this idea, hypertext is a collection of documents that is connected by (associative) links.
The World Wide Web is the most common form of hypertext.

Does one size fit all?

In a library, a person looks for some books on China. What will the librarian recommend?

- Is the person a *small child* who saw a TV show about China and wants to learn about this exotic country?
- Or a *high school student* working on a paper?
- Perhaps a *prospective tourist*?
- A scholar interested in *Eastern philosophy*?
- Someone who can *read Chinese*?

Elaine Rich: User Modeling via Stereotypes. Cognitive Science 3, 329-354 (1979)

Most likely the librarian will make an educated guess, based on the person's appearance:

- age, style of clothing, accent, choice of words, ...

This initial guess might be confirmed or refuted by observations.

- It is assumed that a European cannot read Chinese, unless said otherwise
- Children are generally not (yet) interested in Eastern philosophy, but there are exceptions
- ....

The educated guess, a *stereotype* can be refined with follow-up questions.

Persons expect a *personalized* advice, even though the librarian does not know them.

And the same seems to yield for Web stores.

Jeff Bezos, amazon.com

If I have 3 million customers on the Web, I should have 3 million stores on the Web

When is personalization useful?

My supermarket is not personalized. Still, I can find all products that I need. Probably just because my needs are similar to everyone else's needs.

Personalization is deemed useful when:

- there are so many things to choose from that there is a need for guidance or recommendations
- the system is used by people with different goals and backgrounds

The ideal recommender

Your partner, your best friend or your mother probably knows a lot about you:

- the food you like, the books you read, the movies you watch
- things that interest you or that upset you
- your current needs, aspirations and goals
- dates of your birthday, your kids' birthdays, and holidays
- secret desires and phantasies

Still, this does not guarantee that your mother will buy you a present that you like.

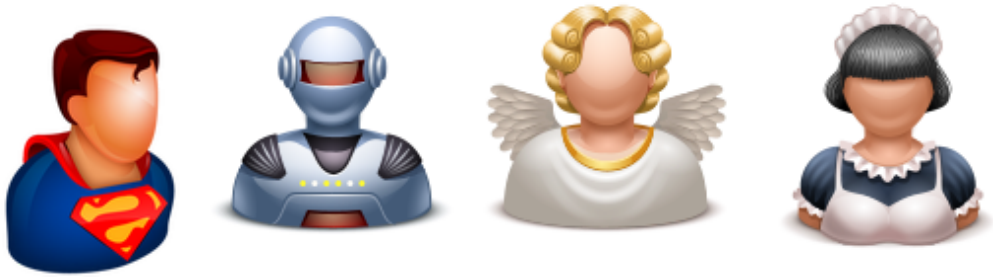
It can be something that

- you already have
- you hate for some reason only known to you
- she bought to surprize you (sometimes this works out perfectly fine, though)

Goals of personalization

In the literature, the following goals are often mentioned:

- helping users to find information they need
- presenting information in the language of choice
- recommending products
- supporting collaboration
- taking over parts of routine tasks



How does personalization work?

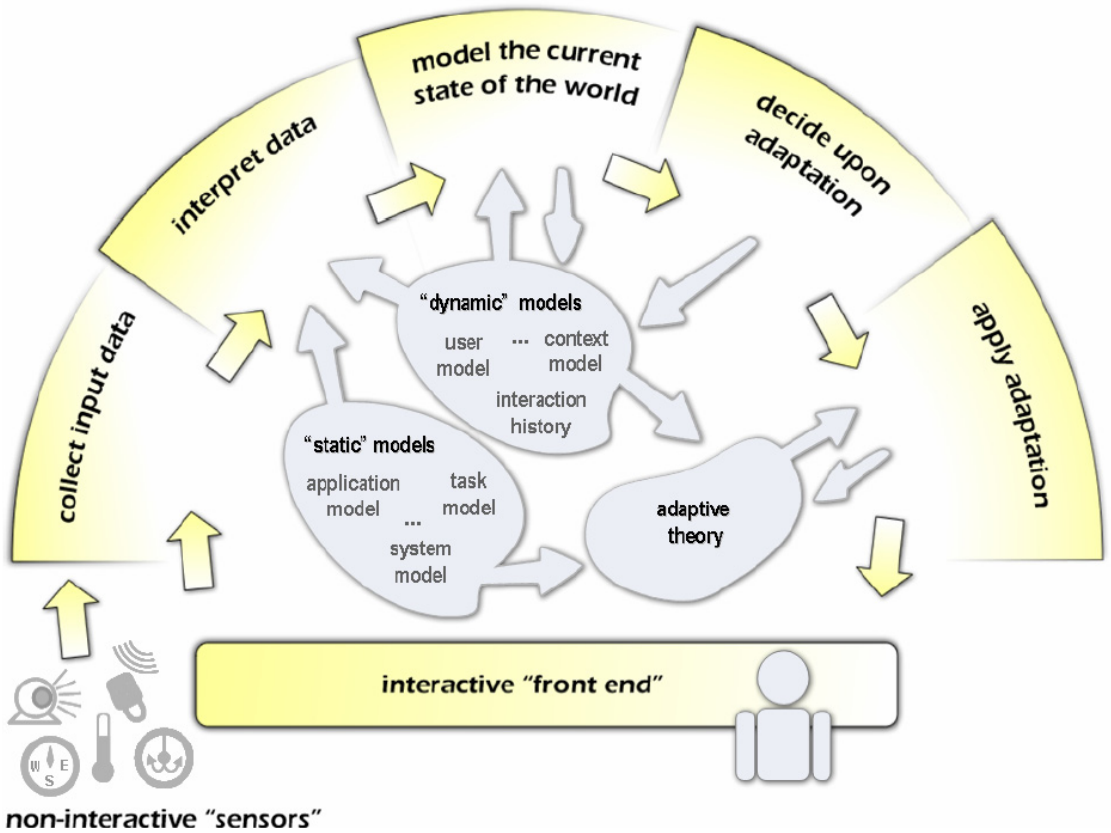
In a nutshell, a personalized system tries to understand the user using

- information that he or she provides
- activities that a user carries out

And tries to guess what you currently want to do, by comparing

- you with people that are similar to you
- things that you own or like with similar things

Steps in the personalization process



Alexandros Paramythis, Stephan Weibelzahl, Judith Masthoff. Layered evaluation of interactive adaptive systems: framework and formative methods. User Model. User-Adapt. Interact. 20(5): 383-453 (2010)

WHAT IS A USER MODEL

User model: definition

USER MODEL (UM): a UM is a **data structure** that describes a user U in a certain moment in time.
At time t, the UM contains a snapshot of the characteristics of the user U, as collected, inferred and stored by the system S.

User data

They are are data about the user and observations about the user interaction with the system:

- that can be directly used for adaptation,
- need to be elaborated.

Which User Data

Domain independent:

- *Demographic information*: age, gender, profession, etc.
- *Contextual information*: location
- *User goals*: long and short term user objectives
- *User habits*: user recurrent actions

Which User data

- *User skills*: user's familiarity with the system
- *User traits*: personality factors, cognitive factors, learning styles
- *User mood*: happy, stressed, relaxed, tense, afraid, motivated, bored, engaged, frustrated, . . .

Domain dependent

- *User knowledge*: which concepts a user is familiar with, and which need additional explanation
- *User interests*: in some domain concepts

Some data are easier to obtain or to infer than others (demographic data easier to obtain than traits).

Implicit vs. Explicit model

The UM can explicitly represent user features (**explicit UM**) or can be a function obtained by an inductive learning process (**implicit UM**).

Explicit user model

The user model explicitly represents the relevant aspects of the user as closely as possible (heuristics-based approach). Explicit user profile can be represented as a set of feature-value pairs or as vectors of terms.

- Pros: the process is intuitive and the models are interpretable and reproducible
- Cons: limitations in scalability and extendability

Implicit user model

Implicit, statistical models and machine learning techniques. Model-based approaches can learn a regression or classification model starting from a collection of items rated by users.

- Pros: more flexible and better suitable for dealing with huge quantities of data
- Cons: less human-readable

Lifetime and scope

Lifetime:

- short-term UMs that are valid for a specific session/task
- long-term UMs that store knowledge, interests, demographics etc. valid for longer time periods

Scope:

- individual UMs store information about single users
- group models represent groups of users (e.g. a class of learners)

WHAT IS USER MODELING

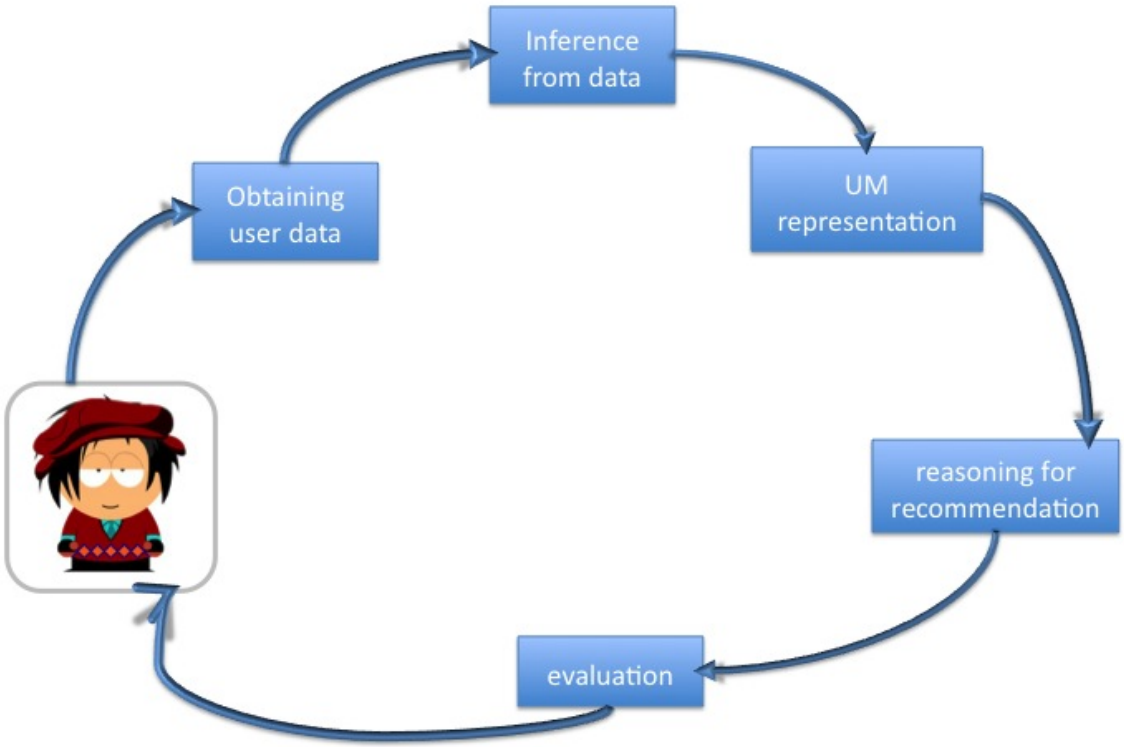
User Modeling: definition

User Modeling is the process of creating and updating a user model, by deriving user characteristics from user data.

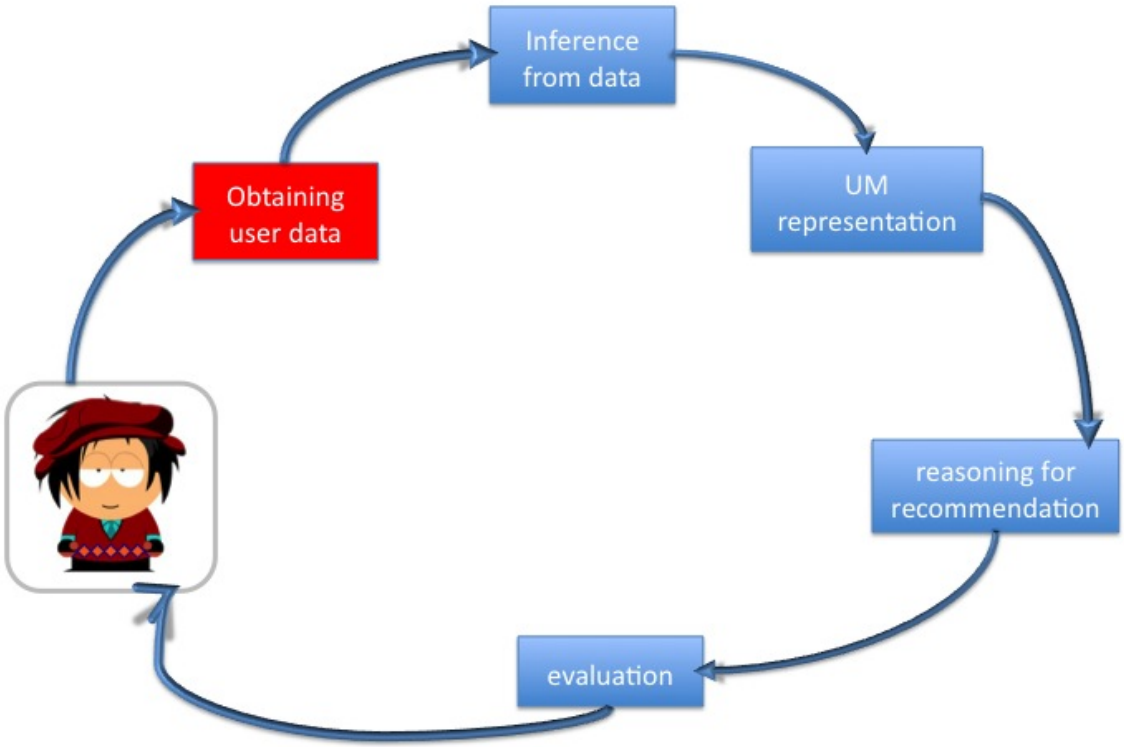
User Modeling process

- **obtaining user data** (explicit or implicit methods)
- **inference of knowledge from data** (e.g., usage pattern, user classification, inference of new user features...)
- **representing UMs**, (e.g., flat, hierarchical, overlay model...)
- **reasoning on data**, (e.g., methods for adaptation and recommendation)
- **evaluating UMs**, (e.g., user centered and/or dataset-based methods)

User Modeling process



User Modeling process



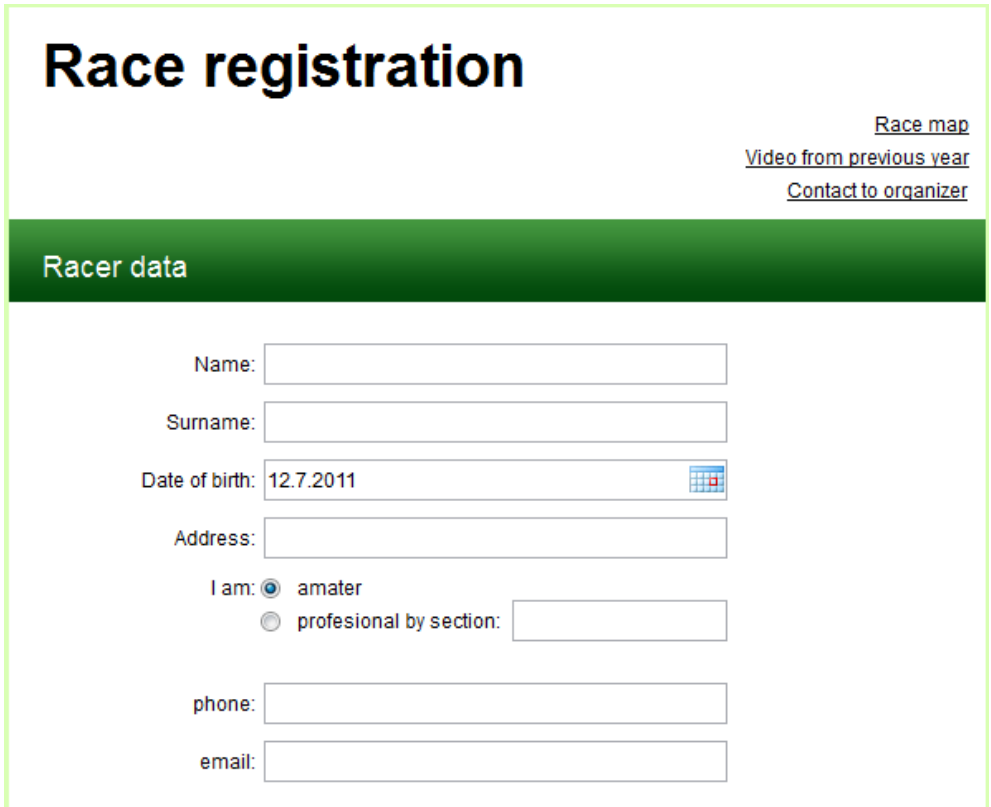
Obtaining user data

User data can be:

- explicitly provided by the user
- implicitly inferred from raw data (by observation)

Direct input from user

User input is often gathered during the first usage of the system using forms or questionnaires.



Direct input from user

User input can be gathered while the user interacts with the system. The user can give relevance feedback by means of **rating scales**. For example, in Movielens, the recommendation process exploits user ratings of movies.

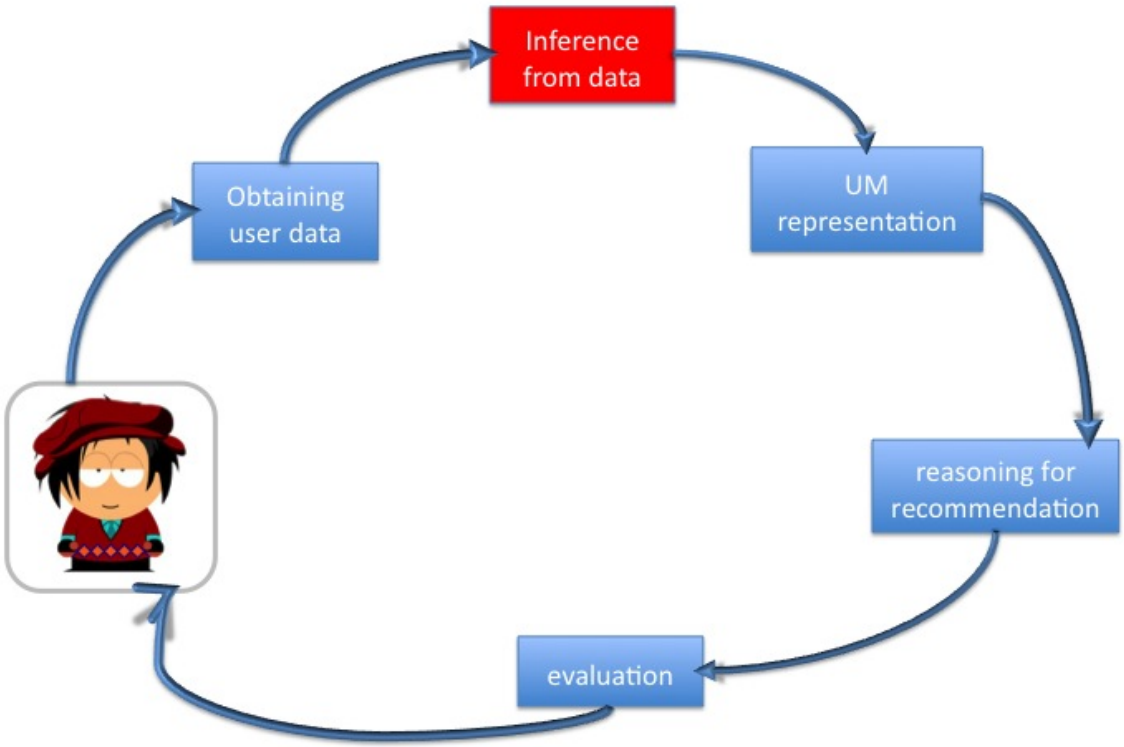
Observing the user

Users do not want to fill out forms or follow an introductory tour.
Many adaptive systems try to infer knowledge directly by unobtrusively monitoring the user interactions with the system:

Observing the user

- browsing history (bookmark folder, search history)
- device information (display resolution, network speed and bandwidth, software)
- location (position, direction...)
- social network data (group membership...)

User Modeling process



Inference from data

The process of interpreting the observations about the user U, using conditions, rules or other forms of reasoning, and the storage of the inferred knowledge in the UM.

Inference from data

Many interactions contain meaning themselves, such as page visits, bookmarking or saving actions, queries issued by the user and items inspected or bought from an e-commerce Web site.

Other interactions need to be interpreted in order to become meaningful, such as key strokes, mouse clicks and eye gaze behavior.

Inference from data

Examples:

- detecting patterns in user behavior (to infer items that may be of interest for users);
- matching user behavior with the behavior of other users;
- classifying a user based on her behavior (stereotyping and the modeling of user interests).

Inference from data

For example incremental update of the UM learning from user actions.

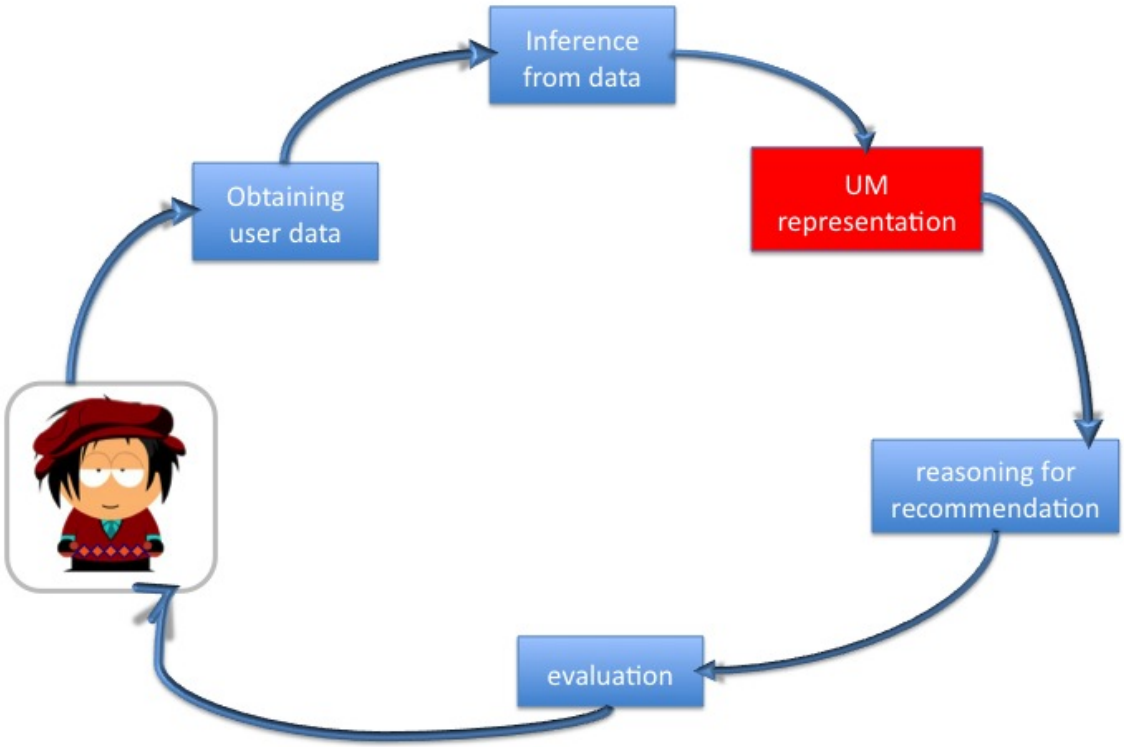
- which users actions to monitor?;
- how such actions impact on user model?;
- how often it should be updated?

Inference from data

$$UpdatedModel = k1 * oldModel + k2 * newModel$$

where
k1 + k2 = 1
at the beginning, k1 near to 0 (0.2) and k2 near to 1 (0.8)
then, k1 increases and k2 decreases (k1 0.8 k2 0.2)

User Modeling process



User Model structures

A User Model is a data structure that characterizes the user U at a certain moment in time.

As seen before, UM features can be **domain independent** (user's demographics) or **domain dependent** (knowledge or preference on certain topics). The latter requires a domain representation as well.

User Model structures

It is possible to use different knowledge structure for UM, from simpler to more complex:

- Flat Models
- Hierarchical models
- Overlay models
- Stereotypes

User model representation

In each structure, domain independent and dependent user features can then be represented in any of the following ways:

- Attribute-Value Pairs
- Booleans
- Probability distributions, Fuzzy Intervals
- Vectors of terms, possibly including weights
- Logic Based representation
- Triples, semantic representations

Flat model

The most simple way to represent user model is a collection of variables and associated values.
An example of *attribute-value pairs* can be:
{age, 15}
{sex, male}
{profession, student}

They may be combined as basic rules to provide adaptivity. A rule might indicate that if a user's age < 18 and user gender=female, select news items interesting to young females. It is hard to make more complex deductions.

Flat model

An example of attribute-value with *probability distribution* can be:

UM's feature distribution

Art-Architecture/Design 0.023181

Art-Multimedia/Performances 0.013145

Art-Museums 0.001416

Cinema-Movies 0.004132

Flat model

Another example of a flat model is *bags-of-words* model, where using Information Retrieval techniques we construct a set of concepts which describe the user's interest. For example, in a movie domain, the following concepts describe the categories of movies that the user might be interested in:

{ sport
rock
music
gym
cartoon }

Flat model

An example of **vector space model** where items and user's profile are represented as a weighted vector computed using TF-IDF formula:

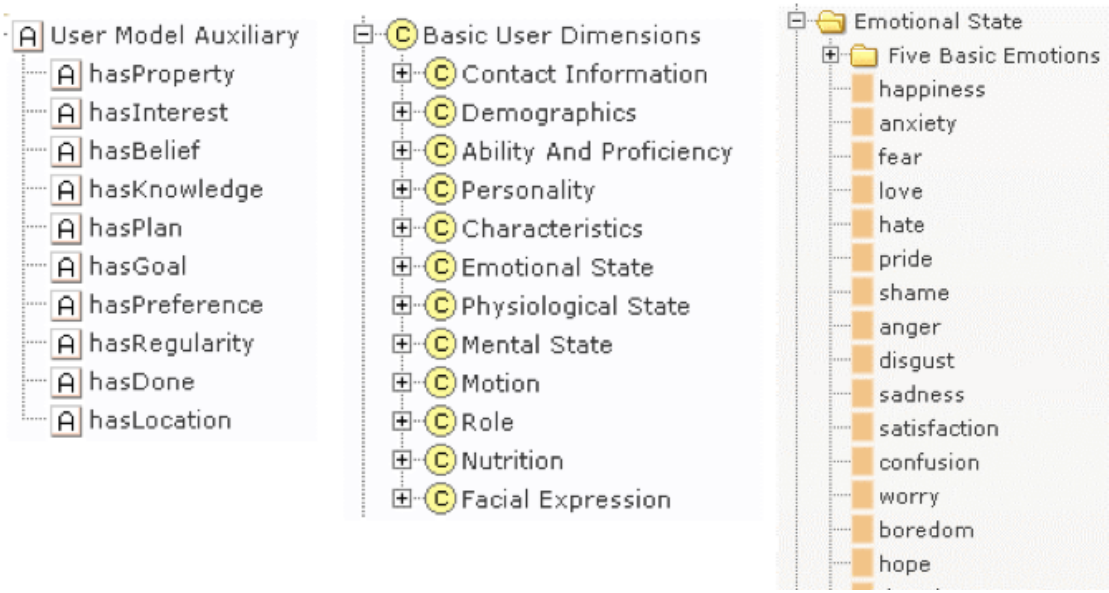
$$\begin{pmatrix} & T_1 & T_2 & .... & T_t \\ D_1 & w_{11} & w_{21} & \dots & w_{t1} \\ D_2 & w_{12} & w_{22} & \dots & w_{t2} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ D_n & w_{1n} & w_{2n} & \dots & w_{tn} \end{pmatrix}$$

Hierarchical model

Some aspects of the UM are higher level and more general than others, representing relations between user characteristics. A common hierarchical structure is a tree or a directed acyclic graph. They are hand-crafted based on the domain knowledge of the designer.

Hierarchical model

An example of a hierarchical model for domain-independent user features

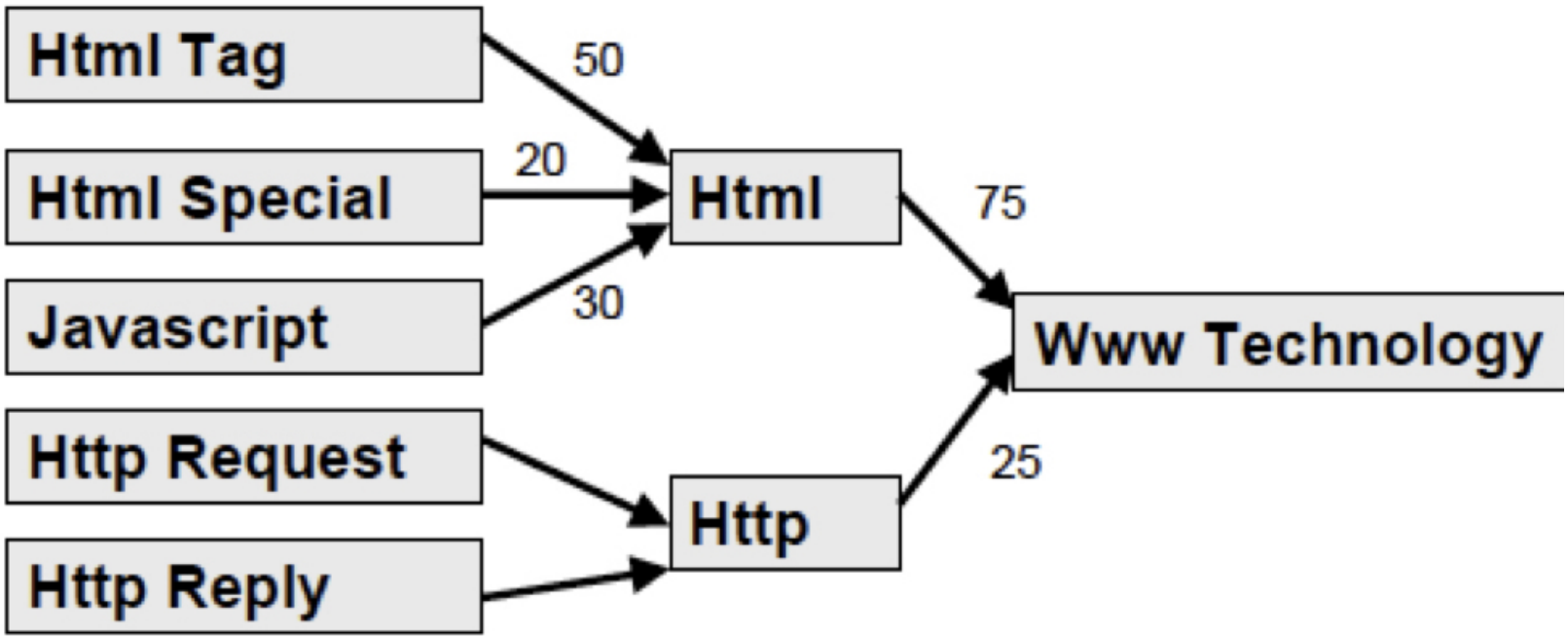


Overlay model

Domain-dependent user features (interest or knowledge) are usually represented as an overlay of the domain structure. For each item in the domain, the user's current state (e.g. interest or knowledge) with respect to the item is recorded.

Overlay model

An example for user knowledge:



Stereotypes

People often make assumptions about other people, often based on fairly simple observations. Stereotypes contain:

- a classification part (with domain-independent features) to classify a user in a category;
- a predictive part (with domain-dependent features) to make predictions based on the category, i.e. standard features associated.

Stereotypes

A stereotype is composed of two parts [Rich, 1989]:

- a set of triggers which can apply the stereotype to the user.
- a body, with information that is typically true for the member of the stereotype;

Reasoning by stereotypes means to evaluate the triggers for the specific users, and when activated, to insert in the UM the body content as assumption on user behaviour.

Stereotypes

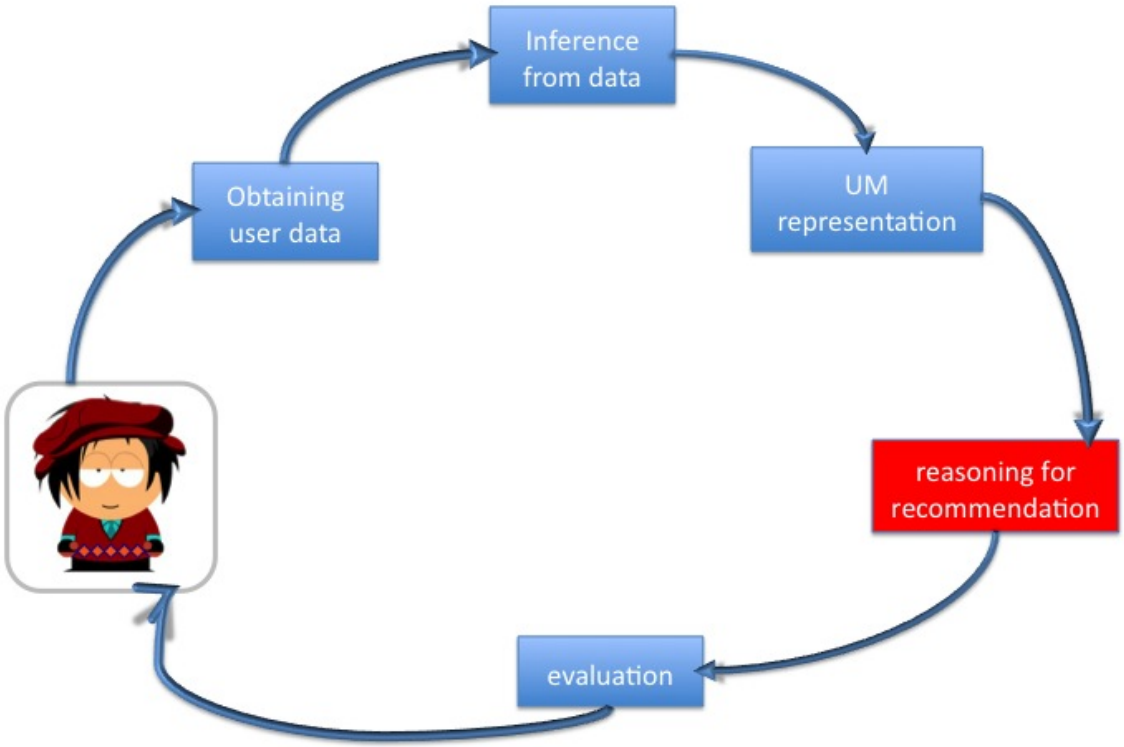
Each stereotype UM is a combination of attributes and their values that belong to the activated stereotypes and consists of:

- Attributes (facets): user characteristics derived from the activated stereotypes
- Values: the score associated with the characteristics
- Rating: degree of certainty
- Justification: the stereotypes that contributed to this attribute

Stereotypes

FACET	*VALUE*	*RATING*	*JUSTIFICATIONS*
Gender	female	1000	Inference-female name WOMAN
Nationality	USA	100	ANY-PERSON
Education	5	900	INTELLECTUAL
Seriousness	5	800	INTELLECTUAL
Piety	-3	423	WOMAN FEMINIST INTELLECTUAL
Politics	Liberal	910	FEMINIST INTELLECTUAL
Tolerate-sex	5	700	FEMINIST
Tolerate-violence	-5	597	WOMAN
Tolerate-suffering	-5	597	WOMAN
Sex-open	5	960	FEMINIST INTELLECTUAL
Personalities	4	646	WOMAN
Opt-pes	0	100	ANY-PERSON
Plot-intr	0	100	ANY-PERSON

User Modeling process: recommendations



User Modeling process: recommendations

The goal of a UM is to create a digital representation of users to be used to adapt interface and content.
In this tutorial, we focus on *content adaptation*, describing the **recommendation process**.

Recommender systems

RSs: a family of information filtering tools providing suggestions for items.
Differently from *search systems*, they allow users to discover new resources that they may have not initially thought about.
"Search is what you do when you are looking for something. Discovery is when something wonderful that you didn't know existed finds you"

Recommender systems

The recommendation problem:

$$f : UXI \rightarrow R$$

The utility function f measures the usefulness of item $i \in I$ for user $u \in U$.
The recommendation problem consists of finding for each user U an item $i \in I$ maximizing the utility function f.

$$\forall u \in U, \ i^{max,u} = arg \ max_{i \in I} f(u, i)$$

Recommender systems

Typically, the utility of an item is represented by a rating. Only ratings for a subset of items are available.
The main task of a RS is to estimate the utility function (rating) from the available data (**predictive system**).

Prediction tasks

- **Rating prediction task**: to accurately predict ratings (metrics: MAE, RMSE)
- **Top-n recommendation task**: to find new specific items supposed to be the most appealing (metrics: precision, recall)

Recommender systems

Basic elements of a RS are:

- **Items**: represented as numeric id in a data base, or bag of keywords or set of attribute/values, or ontology based descriptions
- **Users**: represented explicitly or implicitly depending on the approach (heuristics vs. model based)
- **Ratings**: can be explicitly gathered by different rating scales - numerical (1-5 stars) or ordinal (strongly agree, agree...), or binary (like-dislike) - or implicitly inferred from user behavior.

Recommendation techniques

Main families of recommender systems:

- content-based
- collaborative filtering

on the basis of:

- the way the utility function is estimated
- the way the users and the items are represented

Content-based RSs

CB-RSs recommend an item to a user based on a description of the item and a profile of the user's interests. A CB-RS matches up the attributes of a user profile in which preferences an interests are stored, with the attributes of an object (item). It depends on the availability of content features describing the items. They can be extracted from unstructured items description using NLP, or can be obtained from structured data as database.

Content-based RSs

(source: Di Noia and Ostuni, 2015)

Content-based RSs

In this approach movies are provided with attributes, such as actors, genres, etc. Only the target user is considered in the recommendation process. The basic intuition behind this approach is the following: since Alice likes Argo she may like Heat because they both belong to the Drama genre.

Content-based RSs

There are two main CB-RS approaches:

- **heuristics-based**: explicit representation of user features in the model: items are recommended based on a comparison between their content and the user model
- **model-based**: no need of explicit representation of user features in the model: first a model is created starting from examples and then used to predict unknown ratings.

Heuristics-based approach

The UM is represented using typical **Information Retrieval** techniques, for analysing the items the users liked. A typical approach is to use a **Vectors of Terms, Bags of Words** or a more sophisticated **Vector Space Model (VSM)** (where items and user profiles can be represented as a weighted vectors computed using the tf-idf formula).
The match between items and user profile can be computed using *similarity metrics* (e.g., cosine similarity) and the most similar items to the user profile are recommended.

Model-based approach

Machine Learning techniques are used to learn a model (regression or a classification model) of the user preferences by analyzing the content of the items the user rated (Bayesian Network, SVM - Support Vector Machine).
The training set consists of item feature vectors labeled with ratings. Such learnt user model can be used for estimating the unknown ratings.
The process is usually done for each user separately.
Limitation: a large number of examples is necessary.

Model based approach

An example of model-based CB-recommender:

Training Set

Drama	Crime	class
1	0	*like*
0	1	*dislike*

Test Set

Drama	Crime	class
1	0	*?*

(source: Di Noia and Ostuni, 2015)

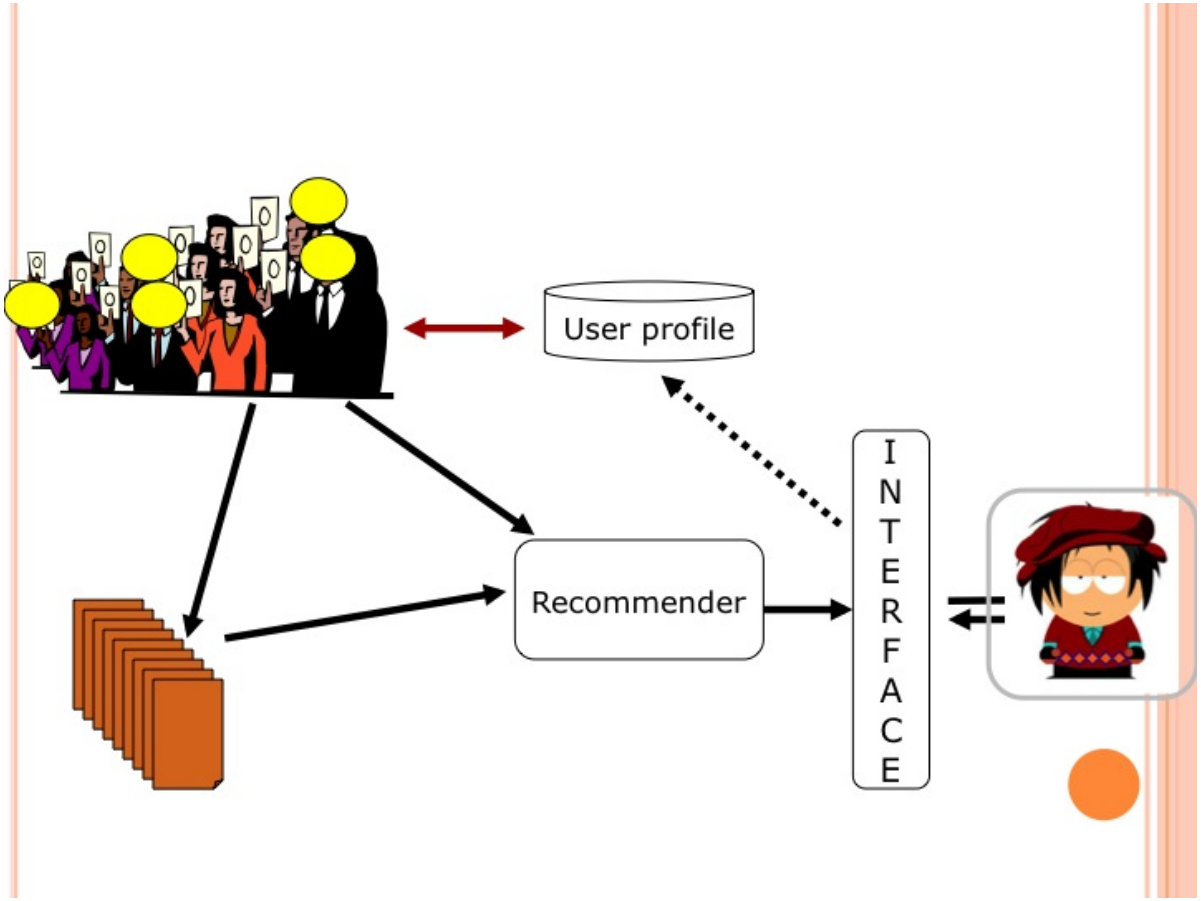
Limitations of content-based

- *content overspecialisation*, i.e. the incapability of the RS to recommend relevant items different from the ones already known
- *portfolio effect*, i.e. redundancy and low diversity in the recommendation list
- *limited content analysis*, i.e. the quality of CB recommendations depends on the quality of features extracted from items

Collaborative filtering (CF)

CF is the the process of filtering items using the opinions of other users having similar tastes.

Collaborative filtering



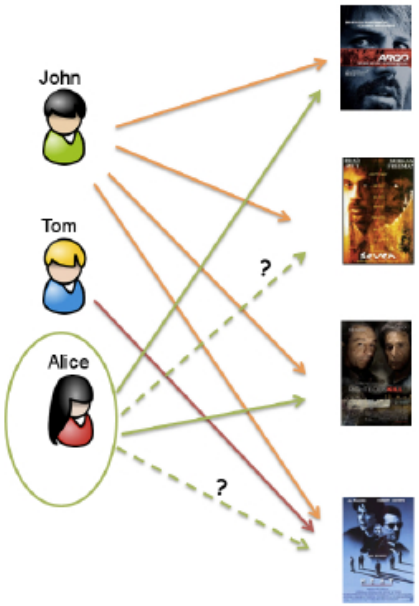
Collaborative filtering

Differently from CB technique, the only input data that CF-RSs need is the user-item ratings matrix.

(source: Di Noia and Ostuni, 2015)

Collaborative filtering

The CF recommendation process:



(source: Di Noia and Ostuni, 2015)

Collaborative filtering

Recommendations are generated considering the ratings given by other users with similar tastes. In this case, both John and Alice have similar tastes because they both rated similarly Argo and Righteous Kill. The system can exploit John's ratings for estimating Alice's unknown ratings.
The basic intuition behind this method is that since John really likes Heat then also Alice may like it.

Collaborative filtering approaches

Two typologies of CF exist:

- **heuristics-based**: based on neighborhood models (nearest neighborhood (k-NN) algorithm), which uses similarity metrics (Pearson, cosine similarity) for finding similar users and items. It does not require any preliminary model building phase.
- **model-based**: first learn a predictive model which is used to make predictions.

Collaborative filtering: user to user

It is based on, very reasonable, heuristic that a person will like the same item with users with similar tastes (user-to user) or like the items that are similar to previously liked items (item-to-item).
The user-to-user approach consists of predicting the relevance of an item for the target user by a linear combination of her neighbour's ratings, weighted by the similarity between the target user and such neighbours.

Collaborative filtering: user to user

STEP 1: INPUT DATA

	u_1	u_2	u_3	u_4	u_5	u_6
$item_1$	5	1	5	4	0	3
$item_2$	3	3	1	1	5	1
$item_3$	0	1	0	2	1	4
$item_4$	1	1	4	1	1	2
$item_5$	3	2	5	0	0	3
$item_6$	4	3	0	0	4	0
$item_7$	0	1	5	1	1	1

Collaborative filtering

STEP 2: CALCULATE SIMILARITY

	u_1	u_2	u_3	u_4	u_5	U_6
$item_1$	5	1	5	4	0	3
$item_2$	3	3	1	1	5	1
$item_3$	0	1	?	2	1	4
$item_4$	1	1	4	1	1	2
$item_5$	3	2	5	0	0	3
$item_6$	4	3	0	0	4	0
$item_7$	0	1	5	1	1	1
Similarity measure	**0.63**	**0.56**		**0.71**	**0.22**	**0.93**

Collaborative filtering

STEP 3. DEFINE NEIGHBOURHOOD

	u_1	u_2	u_3	u_4	u_5	U_6
$item_1$	5	1	5	4	0	3
$item_2$	3	3	1	1	5	1
$item_3$	0	1	?	2	1	4
$item_4$	1	1	4	1	1	2
$item_5$	3	2	5	0	0	3
$item_6$	4	3	0	0	4	0
$item_7$	0	1	5	1	1	1

Similarity measure 0.63 0.56 0.71 0.22 0.93

Collaborative filtering

Step 3. Predictions/Recommendations

Weighted sum
$\frac{(0x0.63+2x0.71+4x0.93)}{(0.63+0.71+0.93)}$
= 2.26

			u_3	u_4	u_5	U_6
			5	4	0	3
			1	1	5	1
$item_3$	**0**	1	?	**2**	1	**4**
$item_4$	1	1	4	1	1	2
$item_5$	3	2	5	0	0	3
$item_6$	4	3	0	0	4	0
$item_7$	0	1	5	1	1	1
Similarity measure	**0.63**	**0.56**		**0.71**	**0.22**	**0.93**

Collaborative filtering

Recommendation can be computed as a weighted sum

$$r_{u,i} = \frac{\sum_{j=1}^{K} r_{u_j,i} \cdot w_{u,u_j}}{\sum_{j=1}^{K} w_{u,u_j}}$$

Collaborative filtering: item to item

The item-based CF approach is based on the usage of the same correlation-based or cosine-based techniques to compute similarities between items instead of users. The idea is to derive a notion of item similarity from user rating and recommend items similar to those the user has already liked.

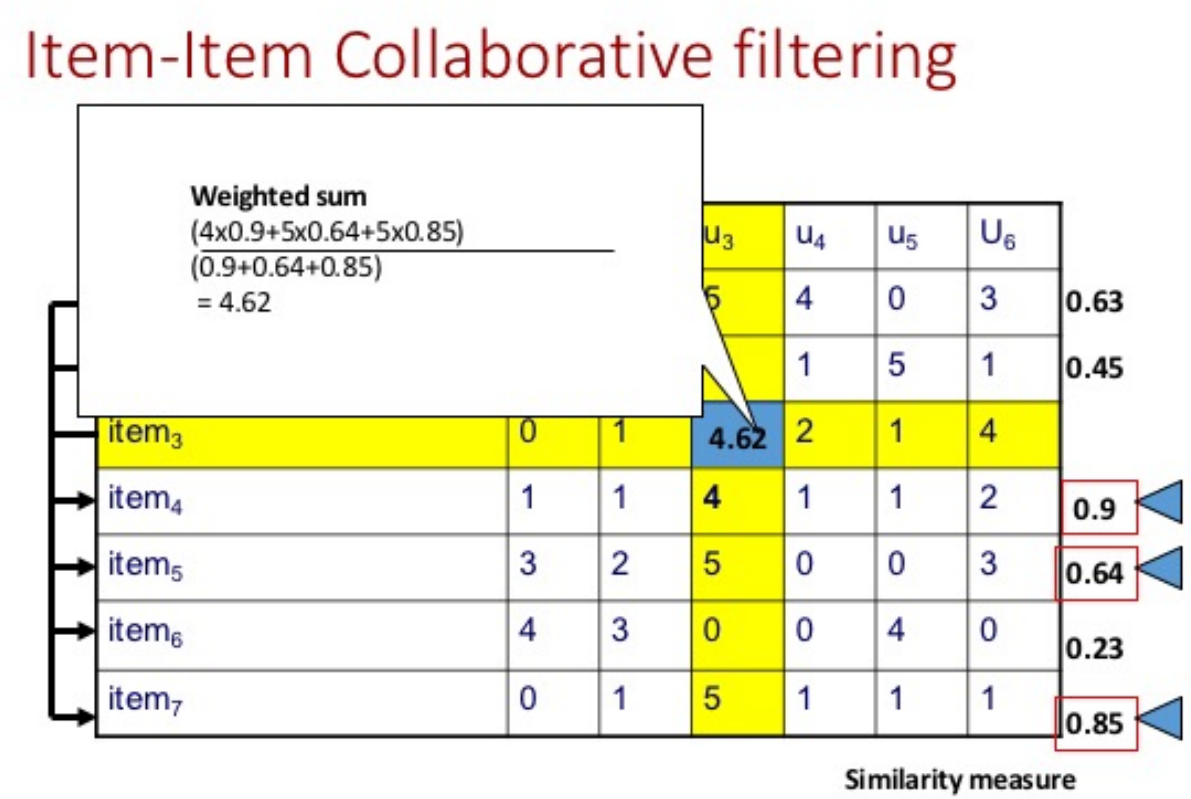
Collaborative filtering: item to item

ITEM-ITEM COLLABORATIVE FILTERING

	u_1	u_2	u_3	u_4	u_5	U_6	
$item_1$	5	1	5	4	0	3	0.62
$item_2$	3	3	1	1	5	1	0.44
$item_3$	0	1	?	2	1	4	
$item_4$	1	1	4	1	1	2	0.9
$item_5$	3	2	5	0	0	3	0.64
$item_6$	4	3	0	0	4	0	0.23
$item_7$	0	1	5	1	1	1	0.85

Similarity measure

Collaborative filtering: item to item



Limitations of collaborative-filtering

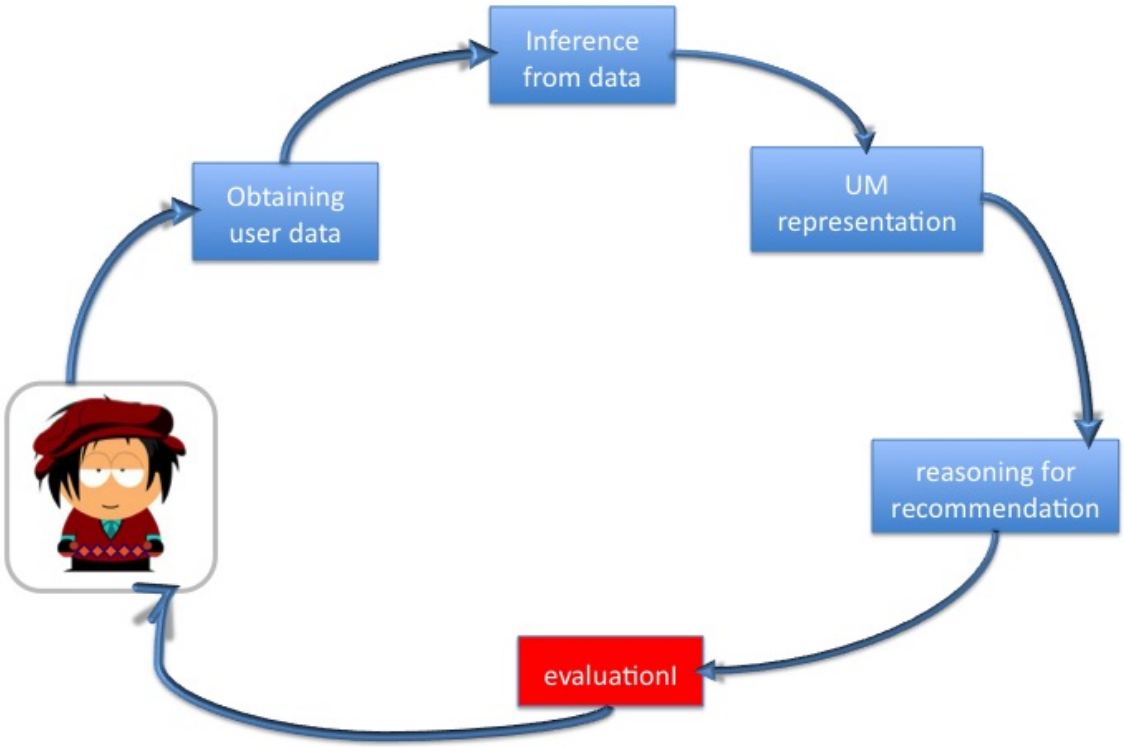
- *cold start or sparsity problem*, i.e. need many ratings data to work
- *diversity problem*, i.e. lack of diversity in the results
- *grey sheep problem*, i.e. the inability of the system to consider user with very unusual preferences (not able to find similar users)

Hybrid recommender systems

They combine different recommender techniques (content and collaborative) to mitigate the weakness of the individual approaches. Several ways to merge:

- mixed, recommendation generated from several RSs are presented together
- switching, one RS is turned on and the other is turned off
- features combination, the features used by different recommender are integrated and combined in a single data source

User Modeling process: evaluation



Evaluation

Need for a layered-evaluation

- user-based: formative and summative evaluation with users
- data-set based: metrics and statistics

User-based evaluation

The evaluation of the UM may be twofold:

- during the user model construction to assess the coverage of user model (formative evaluation phase)
- measuring the quality of recommendation based on UM (summative evaluation phase)

Techniques: controlled experiment, observations, focus groups, interviews...

Formative evaluation

Formative evaluation is a method of evaluating a model during its construction. It is aimed at checking the first design choices before actual implementation and getting the clues for revising the design in an iterative design-re-design process. It should assess, for instance, if the UM contains features that are relevant for the final recommendation.

Techniques for formative evaluation

- heuristic evaluation
- expert review
- card sorting
- Wizard of Oz prototyping
- first prototyping
- qualitative participative evaluation

Summative evaluation

Summative evaluation is a method of judging the worth of something that has been completed at the end of the interaction with it.
It should assess, for instance, the accuracy, the final users' opinions and satisfaction, and the coverage of the multi-source user model

Techniques for Summative Evaluation

- usability testing
- observational methods
- controlled experiments
  - Between-groups design uses separate groups of participants for each of the different conditions in the experiment.
  - Within-groups design exposes each participant to all of the conditions of the experiment
- collection of user opinion (questionnaire, interview)

Dataset-based Evaluation

The most common aspect of recommendation quality measured is **accuracy**, i.e. how much the recommendation fits user interest.
Accuracy Metrics:

- **rating prediction task:** Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE)
- **top-n recommendation task:** Precision, Recall

Dataset-based evaluation

Mean Absolute Error: difference between predictive rating and user ratings $MAE = \frac{1}{|TS|} \sum_{(u,i) \in TS} |\hat{r_{u,i}} - r_{u,i}|$

Root Mean Squared Error $RMSE = \sqrt{\frac{1}{|TS|} \sum_{(u,i) \in TS} (\hat{r_{u,i}} - r_{u,i})^2}$

Dataset-based evaluation

Precision: measure of accuracy $P = \frac{Relevant \quad Items \quad Found}{Total \quad Items}$

Recall: measure of completeness $R = \frac{Relevant \quad Items \quad Found}{Total \quad Relevant \quad Items}$

Dataset-based evaluation

Another aspects of recommendation quality measured is **diversity**: how different the recommended items are w.r.t. "what has been previously seen".
Another aspect of recommendation quality measured is **novelty**, i.e. to recommend not only popular item .

HISTORICAL OVERVIEW ON USER MODELING

First generation of user modeling systems

User modeling is usually traced back to the works of Allen, Cohen and Perrault (e.g., Perrault et al., 1978) and Rich (Rich, 1979a, 1979b). For a ten-year period following this seminal research, numerous systems were developed that exhibited different kinds of adaptation.

In this early work, no clear distinction could be made between system components that served user modeling purposes and components that performed other tasks. **Stereotype user modeling** was one of the earliest approaches to user modeling (Rich, 1979).

Second generation of user modeling

Many efforts are reported on rendering the user modeling component reusable for the development of user-adaptive systems.

- **General user modeling systems** (Finin and Drager, 1986) provide user modeling services at runtime. When filled by the developer with application-specific user modeling knowledge, these systems would serve as separate user modeling components.
- **User modeling servers** (Fink, 2003) work as an application-external knowledge base. In this way, the knowledge about the user is made available to more than one application at the same time.

Third generation of user modeling

While in the centralized approach, a single user model is shared by the interacting systems, in decentralized settings each system maintains a small user model. There were attempts to make such models interoperable. **Decentralized user modeling** investigates how to combine partial user data and make sense of it in a specific context (Dolog and Vassileva 2005) (semantic techniques are used).

Current trend in user modeling

We have assisted to the shift from:

- **centralized** to **distributed** approach
- **explicit user modeling approaches** (based on heuristics) to **model-driven, implicit approaches** (based on statistics and machine learning), made possible by the large data sets of semantic and social web.

.

SOURCE OF WEB USAGE DATA

Types of Web Data and Web Usage Data

Four general categories of data sources for Web usage mining are distinguished:

Content data: the real data in Web pages, as presented to the end user.

- The data can be simple text, images or structured data, such as information retrieved from data repositories.
- Applications of content analysis include finding Web pages that contain similar content and determining which Web pages match a user query or user preferences;

Structure data: data that describes the organization of the content.
The data can be either data entities used within a Web page (*the content*), or data entities used to connect pages (*the navigation structure*) - most notably hyperlinks.

- Statistics on the content - such as the number of words, images or hyperlinks on a page - reflects the function of a Web page (e.g. does it mainly provide content or is it used for navigation).
- The navigation structure, as defined by connecting entities - such as hyperlinks - determines to a large extent user navigation patterns.

Usage data: data that describes the pattern of usage of Web pages, such as IP addresses, access date and time, referring pages and other attributes that can be retrieved

- From the raw Web usage data user groups may be distinguished, as well as groups of pages that are often visited together, popular pages and frequently followed paths.

User profile data: explicitly gathered data that provides demographic information about users of the Web site.

- This includes registration data and customer profile information.
- User profile data may help in explaining peculiarities in the Web usage data, but it can also be used as an additional source of information for personalization.

Sources of Web Usage Data

Web Server

- Data from multiple users on one site
- In general limited to click-behavior

Web Client (i.e. Browser)

- Data from one user on multiple sites
- More sophisticated tracking (e.g. mouse movements)
- Requires user cooperation (e.g. installation of a program)

Proxy Server

- Data from multiple users on multiple sites
- Sits between server and client
- Analyses (and possibly modifies) the data stream

Server-Side Data Collection

The most widely used source of navigation data is the Web server. Most servers store all transactions in a *Web server log*. Whereas the Web Consortium has defined a standardized log format, several Web servers have their own proprietary format. Typical fields included in the log file are:

- The *requested URL*;
- The *remote IP* of the machine from which the page request originated. This may be a user's computer, a provider's proxy server or any other machine from which a page request can be made;
- A *timestamp* of the date and time of the page request;

- The *method* used for requesting the page;
- The *status code*, which indicates whether the page was retrieved successfully;
- The *number of bytes sent* to answer the request;
- The *referrer*: if the user followed a link, the originating url is listed here;
- The *user agent*, which indicates the browser used, or which robot requested the page.

Example server log entries:

(Already somewhat older)

- 213.6.31.68 - - [01/May/2004:22:38:32 +0200] "GET /forsale.html HTTP/1.1" 200 14956 "http://www.fortepiano.nl/indexforsale.html" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)ï¿½
- 213.6.31.68 - - [01/May/2004:22:38:34 +0200] "GET /pictures/forsale/bertsche1835-small.jpg HTTP/1.1" 200 5753 "http://www.fortepiano.nl/forsale.html" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"

Preprocessing web server data

Several issues need to be dealt with before the data from the Web server log can be analyzed:

Data cleaning. This step consists of removing all the requests that are not explicitly initiated by the user, such as additional files that are embedded in the originally requested page.

User identification. Whereas each computer has its own unique IP address, several providers use a proxy server to balance Web traffic. Heuristics or technical tricks are needed to separate users behind the proxy.
Further, sites such as search engines use *robots* for building and updating their database. The common politeness of having robots identifying themselves is not followed by all developers.

Session identification. It may be possible that users visit the site more than once. The goal of session identification is to divide the page accesses of each user into individual sessions.

Path completion. In practice, not all page requests reach the Web server. Browsers may store a page in their local cache for improving response time.
In addition, several providers use a proxy server that stores popular pages for improving response time as well as for reducing Web traffic.

Data cleaning

This step consists of removing all the requests that are not explicitly initiated by the user, such as additional files that are embedded in the originally requested page.

Typically, in this step graphical content such as jpg and gif images are removed.

User identification

One computer may be used by multiple users. One user may use multiple computers. ISPs may use a proxy server, so that multiple users have one IP address. IP addresses may be dynamic. So how to recognize a user?

- The easiest solution is to require users to **log in**. This might put users off, though.
- Another solution is to send a **cookie** to the user computer, or to use a unique session identifier.
- Should this not be possible, it might still be possible to distinguish users based on their **browser identifiers**.
- Should none of these methods be of any avail, one can analyze whether a requested page is reachable from the last requested page.

Robots

Most robots self-identify themselves in the server logs. If not, one can exploit the fact that robots explore the site more systematically and faster than users do, and return to the site at regular time intervals.
Robots explore the entire website in breadth first fashion

- Periodic Spikes (can overload a server)
- Lower-level constant stream of requests

Humans access web-pages in depth first fashion

- Daily pattern: Monday to Friday
- Hourly pattern: peak around midday, low traffic during nights

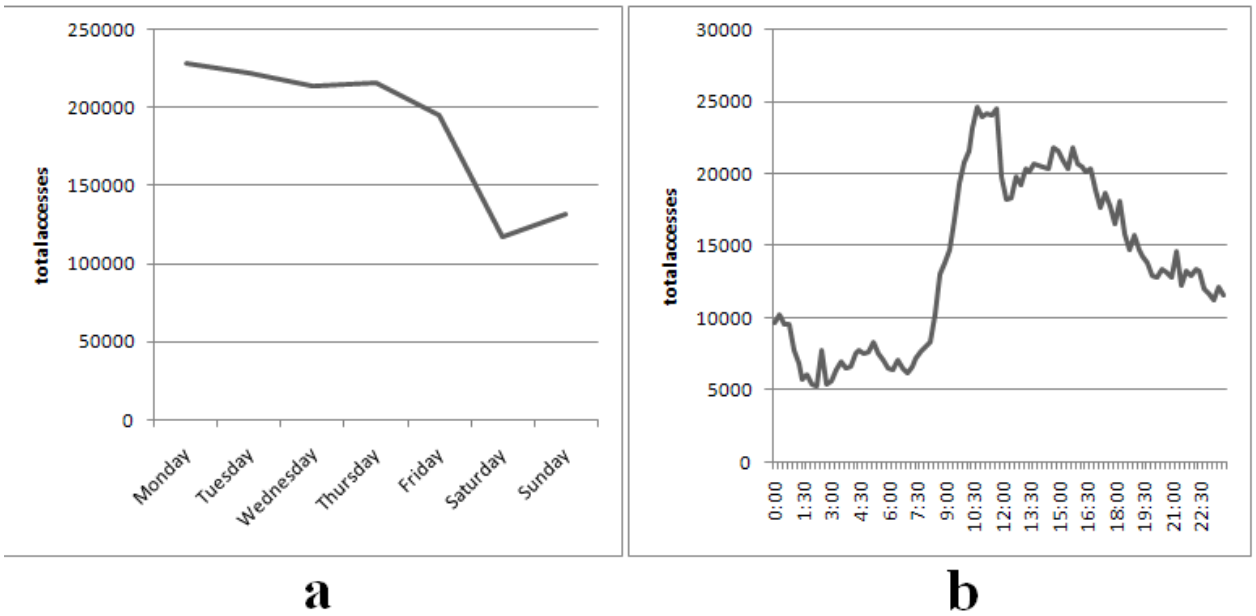


Figure: Typical human patterns of Web site usage

Session identification

Users do not access the Web continuously, but rather have several periods of increased navigation activity and periods in which they do not access the Web at all. These periods of increased navigation activity are assumed to represent a user task, or a set of user tasks.

Session identification - or rather session reconstruction - attempts to split the Web log into meaningful chunks, which are called sessions.

Whereas these sessions are related to the user activity, they are merely a technical concept, defined by the heuristics used during the reconstruction process.

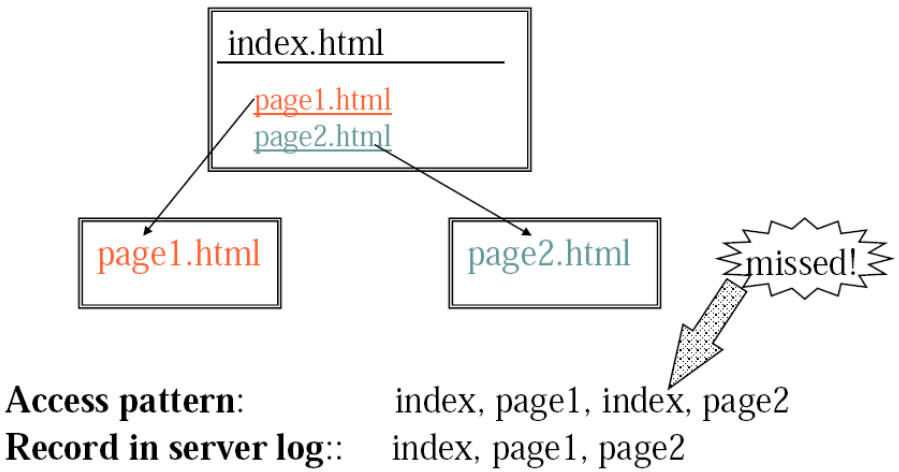
The most common heuristic used for session reconstruction is a time-out mechanism; if the time between two subsequent page requests exceeds a certain time limit, it is assumed that the user is starting a new session.

Path completion

In practice, not all page requests reach the Web server. Browsers may store a page in their local cache for improving response time.

In addition, several providers use a proxy server that stores popular pages for improving response time as well as for reducing Web traffic.

A common approach in situations when two subsequent page visits to *a* and *b* are recorded, while no link between *a* and *b* exists, is to assume that a user backtracked from *a* to some page *c*, from which a link (b, c) exists.



Proxies / Client Cache, from "Web Mining: Accomplishments & Future Directions" Jaideep Srivastava, University of Minnesota, pp 63-90. `http://www.ieee.org.ar/downloads/Srivastava-tut-pres.pdf`

Client-Side Data Collection

In contrast to Web server logs, *client side logging* provides data on user navigation on multiple servers. As the data is collected on the user computer, problems associated with user recognition and path completion are eliminated or reduced.

A popular early technique for client-side logging is the use of *instrumented browsers*. However, browser modification is not a trivial task for modern browsers.

The adoption of *browser plugins* (or browser extensions) is a recent alternative for instrumented browsers.

An alternative approach is the use of agents that are embedded in Web pages, for example as Java applets, which are used to collect information directly from the client.

A more intrusive method for client-side logging is the use of *spyware*.

Potentially, a large number of events can be recorded on client-side, from low-level events such as keystrokes and mouse clicks to higher-level events such as page requests.

There are several limitations to client-side logging:

- client-side modifications might alter the user's normal behavior
- users may be unable or unwilling to install special software that might cause problems or that might form a security thread
- some mechanism is needed to retrieve the data from the client computer

Basic structure of a statement about a user

A statement in a user model should contain at least the first three statements of the main part (subject-predicate-object). Metadata is optional, but recommended. The more complete, the more useful.

Main Part

- *Subject*: whom or what is this statement about (the user)
- *Predicate*: the (user) characteristic represented in the statement (e.g. Interest)
- *Object*: what is the target or object of this characteristic
- *Level*: Qualification/level (if applicable)
- *Origin*: The statement in its original form (if applicable)

Meta Part

- *ID*: Globally unique
- *Creator*: Entity that created the statement
- *Created*: Time of creation/submission of statement
- *Access*: Data for any kind of access control mechanism
- *Temporal*: Constraints on temporal validity of statement
- *Spatial*: In which contexts is the statement valid
- *Evidence*: Refers to or embodies formal evidence
- *Rating*:Level of trust

An example statement

“Peter is interested in Sweden”

gc = http://www.grapple-project.org/grapple-core/
foaf = http://xmlns.com/foaf/0.1/

gc.Statement (
gc:id gc:statement-peter-2009-01-01-3234190;
gc:user http://www.peter.de/foaf.rdf#me;
gc:predicate foaf:interest;
gc:object http://en.wikipedia.org/wiki/Sweden;
)

(Metadata omitted for simplicity)

User Model Interoperability

Interoperability

The ability to cooperate and exchange data despite differences in languages, interface and execution platform

The main advantages of interoperable user models are:

- acquiring more data and more accurate data about users
- acquiring functionalities (both user model and user modeling functionalities) that systems do not themselves implement

Interoperability for better user models

Cross-application user model interoperability is a strategy to cope with the *cold start problem*, which refers to the difficulty for applications to provide suitable adaptations for new users.

Interoperability may also relieve users from the pain of training new systems or wasting time filling in their profile for every new application.

In general, it is believed that interoperable user models lead to more user data and more accurate models.

Dimensions of interoperability

In order to successfully exchange and (re-)use user profile data, requirements on several levels need to be met:

- *Structural level*: Protocols for communication (i.e. interfaces)
- *Syntactic level*: Languages for the exchange of user model data
- *Semantic level*: Agreement on definitions what certain concepts (such as knowledge), terms and expressions, exactly mean

Interoperability on a structural level

Protocols for communication are abundant:

- Remote calls (remote procedure calls, Java remote service invocation, other client-server approaches)
- Web-based protocols (HTTP POST/GET, WSDL/SOAP, XML, JSON, OAI, RESTFUL interfaces)

Syntactic and semantic interoperability is much harder.

Two basic approaches

In essence, there are two ways to ensure interoperability between two (or more) adaptive systems and their user models:

- **Lingua franca:** an agreement between all parties on a common representation and semantics
  This is the philosophy underlying the generic (centralized) user model server approach
- **Conversion** of user model information between the different systems.

Lingua Franca Approaches

Generic User Modeling Ontology

GUMO is developed at the DFKI German Institute for Artificial Intelligence and is divided in four parts:

- *GUMO-Basic*: models user dimensions like personality traits and user characteristics
- *GUMO-Context*: models the world directly around the user, including the events and the environmental context
- *GUMO-Domain*: includes a general interest (domain overlay) model
- *GUMO-Extended*: ranges, rating dimensions and predicates that address other attributes

By adding new parts to GUMO, other ontologies (for example elearning-related) can be integrated.

GUMO

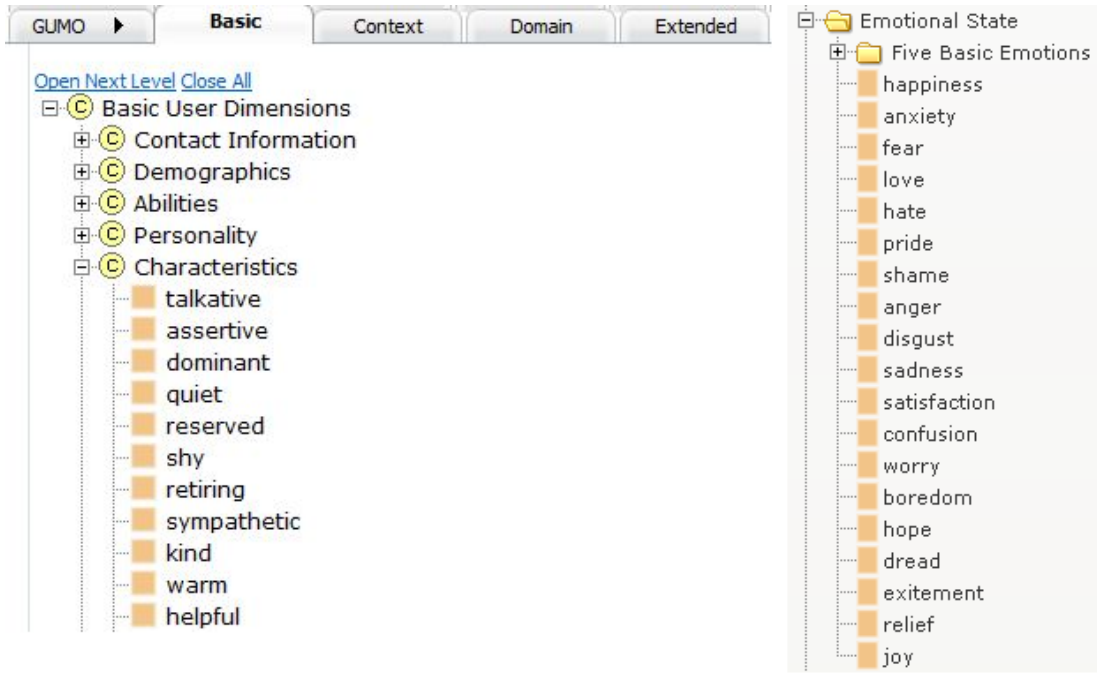


Figure: Screenshot of GUMOs user interface

Conversion Approaches

Conversion allows for flexible and extensible user models, at the price of possible loss of information in the conversion process:

- models may be simply incompatible (there is no suitable mapping)
- mappings may be incomplete (information required in one model is not available in the other)
- the observations in the different systems may lead to contradictions

Conversion: A rule-based example (1/2)

We illustrate conversion approaches by means of a Grapple Derivation Rule that convert data from one format into another.

This simple rule maps test scores of the elearning system CLIX to a more generic external knowledge format by simply multiplying the score by 10:

Conversion: A rule-based example (2/2)

```
<gdr:rule
  xmlns:gdr="http://www.grapple-project.org/grapple-derivation-rule/"
  xmlns:gc="http://www.grapple-project.org/grapple-core/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\#"

  id="1"
  name="Quiz Level to External Knowledge (CLIX)"
  description="Maps the quiz/test result of a specific CLIX score to external knowledge"
  creator="http://pcwin530.win.tue.nl:8080/grapple-umf/client/1">
   <gdr:premise dataspace="1">
      <gc:subject>?user</gc:subject>
      <gc:predicate rdf:resource="http://www.grapple-project.org/ims-lip/completedTest"/>
      <gc:object>solar-system-quiz-id</gc:object>
      <gc:level>?level</gc:level>
   </gdr:premise>
   <gdr:consequent dataspace="1">
      <gc:subject>?user</gc:subject>
      <gc:predicate rdf:resource="http://gale.tue.nl/predicate/knowledge"/>
      <gc:object>gale://gale.tue.nl/cam/DavidTestCAM/SolarSystem</gc:object>
      <gc:level>op:multiply(?level,10)</gc:level>
   </gdr:consequent>

</gdr:rule>
```

Enriching user profiles with the Semantic Web

Tim Berners-Lee

The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

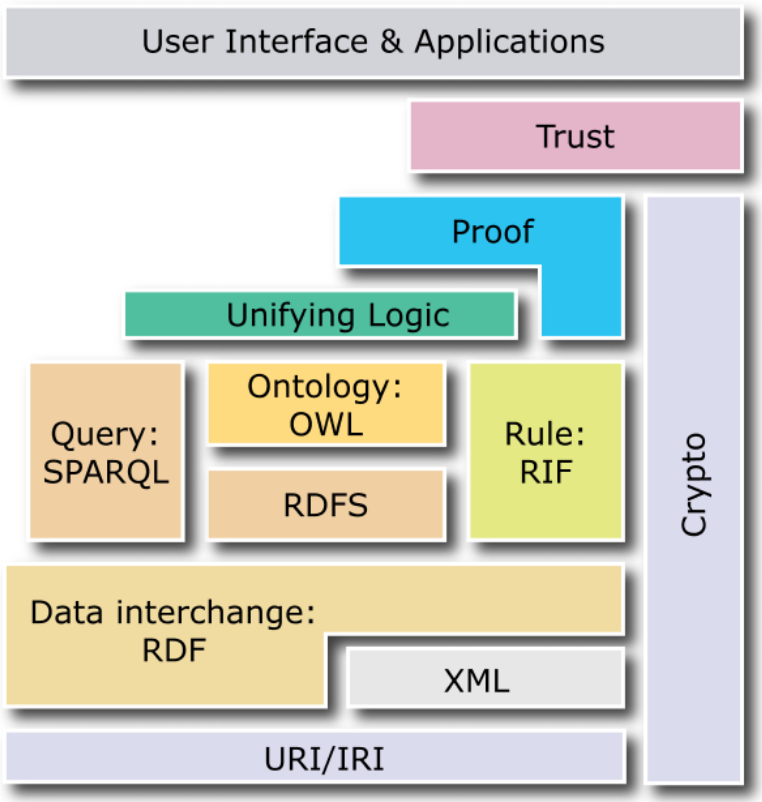
The Semantic Web

The Semantic Web provides a common framework that allows data to be shared and reused accross application, enerprise and community boundaries.

It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners.

It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming.

Semantic Web Layers



RDF

RDF is a model for representing **information about resources** in the WWW.

In particular: metadata about Web resources (e.g.,title, author, version, publication date, ...)

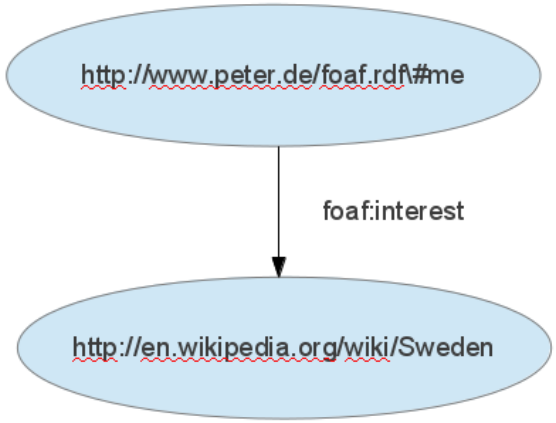
RDF can also be used as a model for representing **information about things** that can be identified in the WWW.

In particular: metadata about items (e.g., prices, specifications, availability information, etc.)

RDF statement: ‘Peter is interested in Sweden’

subject: http://www.peter.de/foaf.rdf#me;
predicate: foaf:interest;
object: http://en.wikipedia.org/wiki/Sweden;

This statement can be represented as a graph.



RDF Schema and ontologies

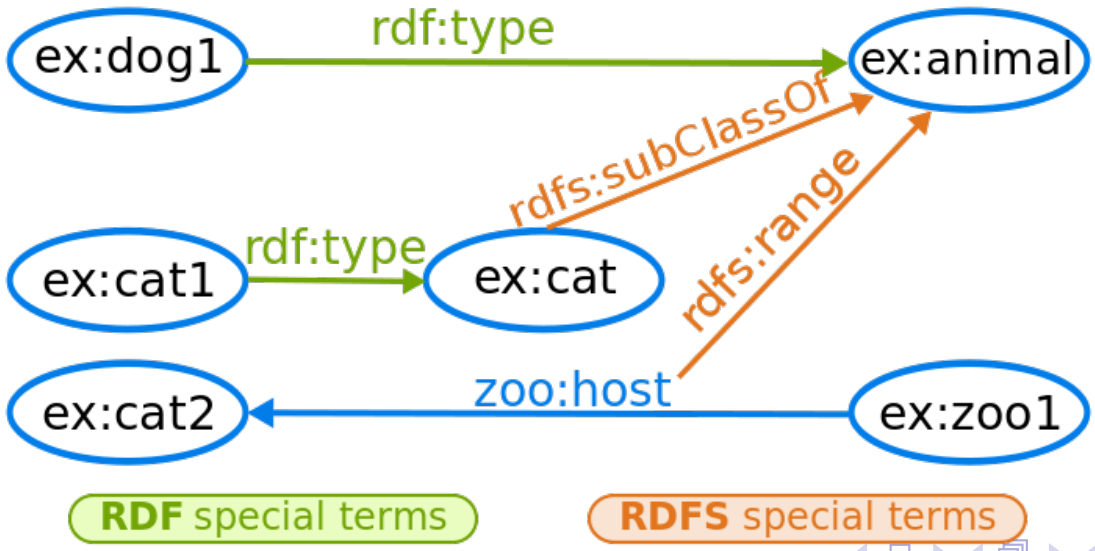
RDF Schema (RDFS) is a set of classes with certain properties using the RDF extensible knowledge representation language, providing basic elements for the description of ontologies.

An **ontology** formally represents knowledge as a set of concepts within a domain, and the relationships between pairs of concepts. It can be used to model a domain and support reasoning about entities.

For example, if we want to represent the following statements:

- Dog1 is an animal
- Cat1 is a cat
- Cats are animals
- Zoos host animals
- Zoo1 hosts the Cat2

This would result in the following RDF graph:



Ontology-based user models

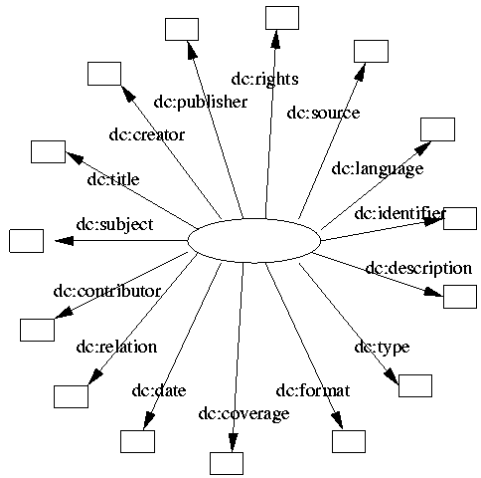
An ontology-based user model is devised as an overlay model over the domain ontology. Using ontologies as the basis of the user profile allows the initial user behavior to be matched with existing concepts in the domain ontology.

Using ontology structure to propagate user interest values starting from a small number of initial concepts to other related concepts in the domain has proven to be a valuable tool in resolving the **cold-start problem** in recommender systems.

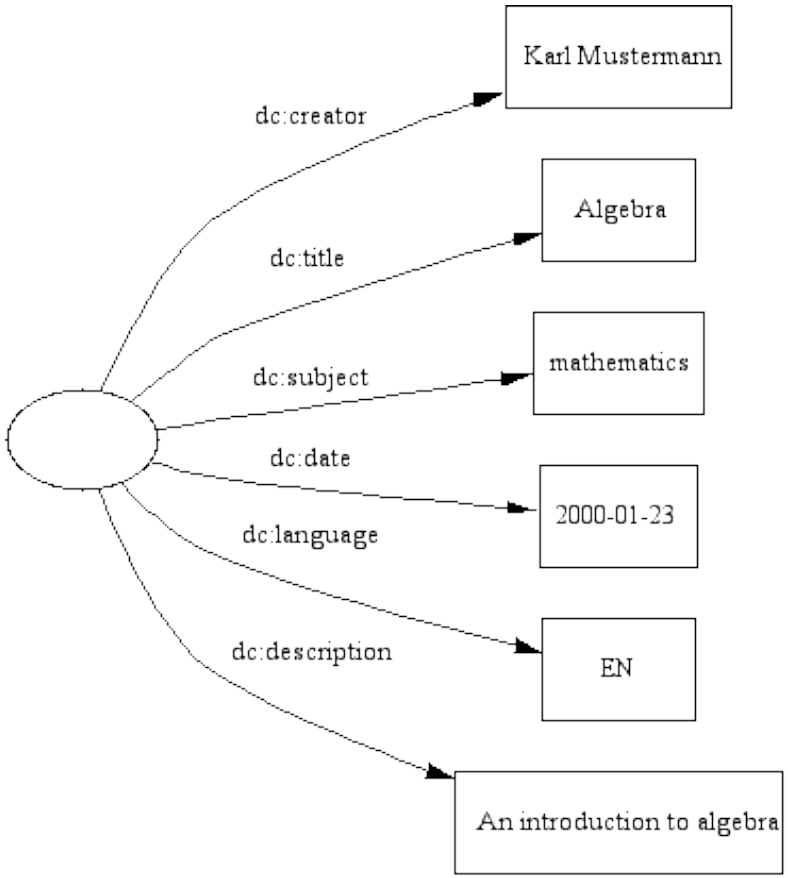
(Brusilovsky and Millan, 2007)

Dublin Core

The Dublin Core metadata terms are a set of vocabulary terms which can be used to describe resources for the purposes of discovery.
The terms can be used to describe a full range of web resources (video, images, web pages, etc.), physical resources such as books and objects like artworks.



An example Dublin Core description of a book



Friend Of A Friend (FOAF)

FOAF is a language to describe people and resources in the Web.

FOAF Basics

- Agent
- Person
- name
- nick
- title
- homepage
- mbox
- mbox_sha1sum
- img
- depiction (depicts)
- surname
- family_name
- givenname
- firstName

Personal Info

- weblog
- knows
- interest
- currentProject
- pastProject
- plan
- based_near
- workplaceHomepage
- workInfoHomepage
- schoolHomepage
- topic_interest
- publications
- geekcode
- myersBriggs
- dnaChecksum

Online Accounts / IM

- OnlineAccount
- OnlineChatAccount
- OnlineEcommerceAccount
- OnlineGamingAccount
- holdsAccount
- accountServiceHomepage
- accountName
- icqChatID
- msnChatID
- aimChatID
- jabberID
- yahooChatID

Projects and Groups

- Project
- Organization
- Group
- member
- membershipClass
- fundedBy
- theme

Documents and Images

- Document
- Image
- PersonalProfileDocument
- topic (page)
- primaryTopic
- tipjar
- sha1
- made (maker)
- thumbnail
- logo

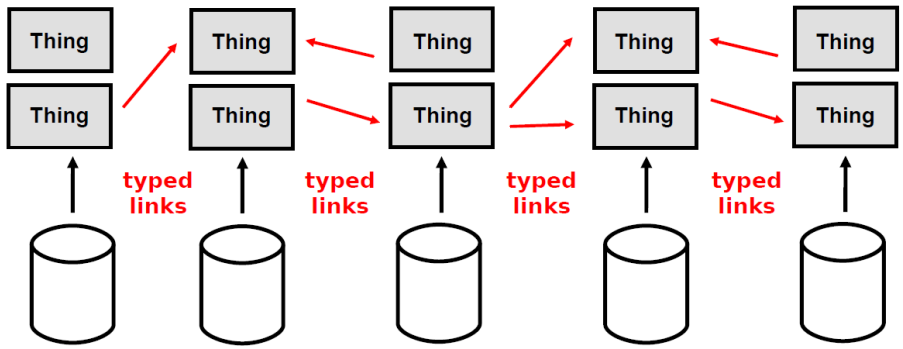
Linked data

The Web enables us to link related documents. Similarly it enables us to link related data.

Linked Data refers to a set of best practices for publishing and connecting structured data on the Web.

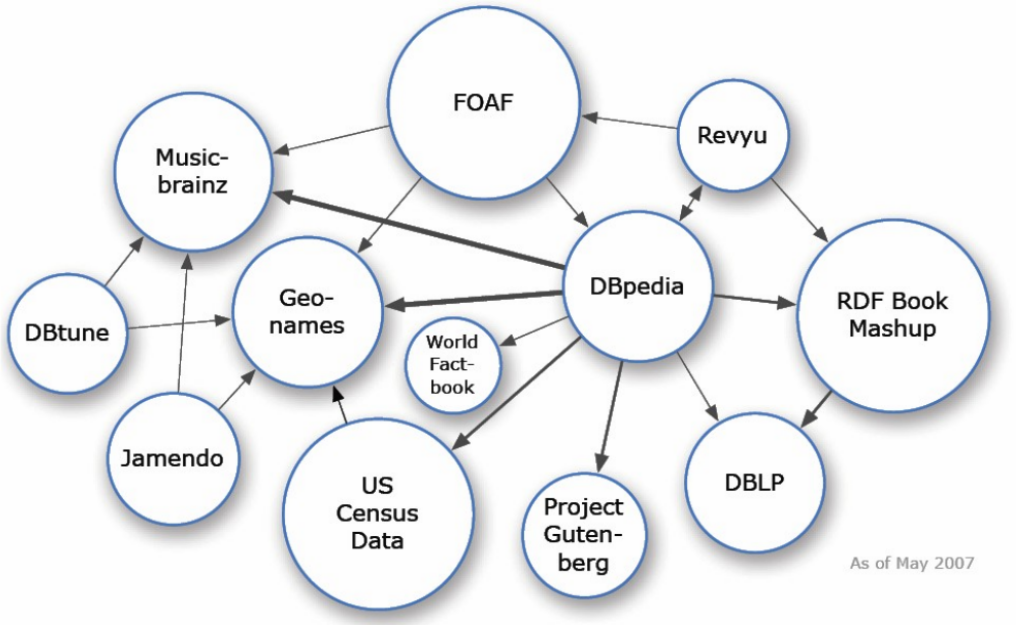
Key technologies that support Linked Data are

- **URIs** - a generic means to identify entities or concepts in the world
- **HTTP** - a simple yet universal mechanism for retrieving resources, or descriptions of resources
- **RDF** - a generic graph-based data model with which to structure and link data that describes things in the world



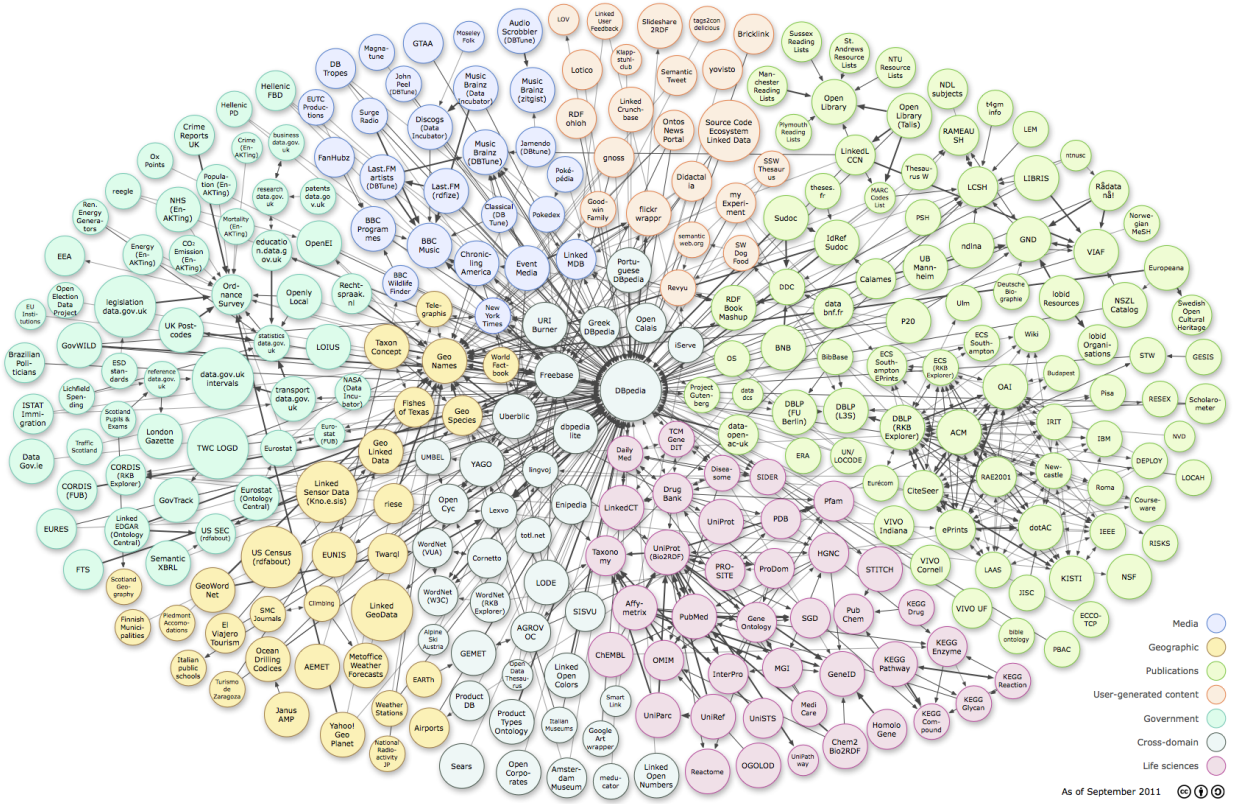
The LOD cloud

The Linking Open Data (LOD) project takes existing open data sets, makes them available on the Web in RDF and interlink them with other data sets.



2007

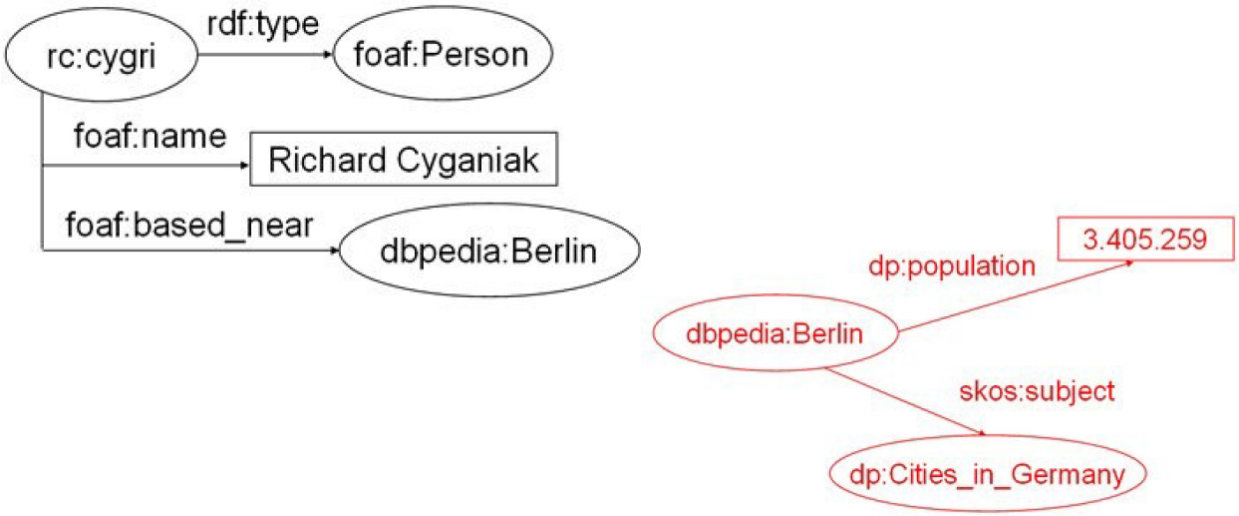
The LOD cloud expands quickly



2011

How to make use of Linked Data

Making use of linked data principles and the “LOD Cloud”, several pieces of knowledge can be connected.
For example, making use of DBPedia, we can infer that Richard Cyganiak lives in a city with a population of 3.405.259.



Representing the User: A Moving Target

Several ontologies and interchangeable standards have been proposed for representing user models.

GUMO includes basic user dimensions, such as demographics, user knowledge, emotional state and personality aspects, skills and capabilities, interests and preferences, goals and plans, etc.

Moreover, GUMO also models the environment by representing data like location, time, device, etc.

However, the current version lacks of modeling social data, even if the authors have started to work on it.

Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., von Wilamowitz-Moellendorff, M. (2005). Gumo - the general user model ontology. In User modeling 2005 (pp. 428-432). Springer Berlin Heidelberg.

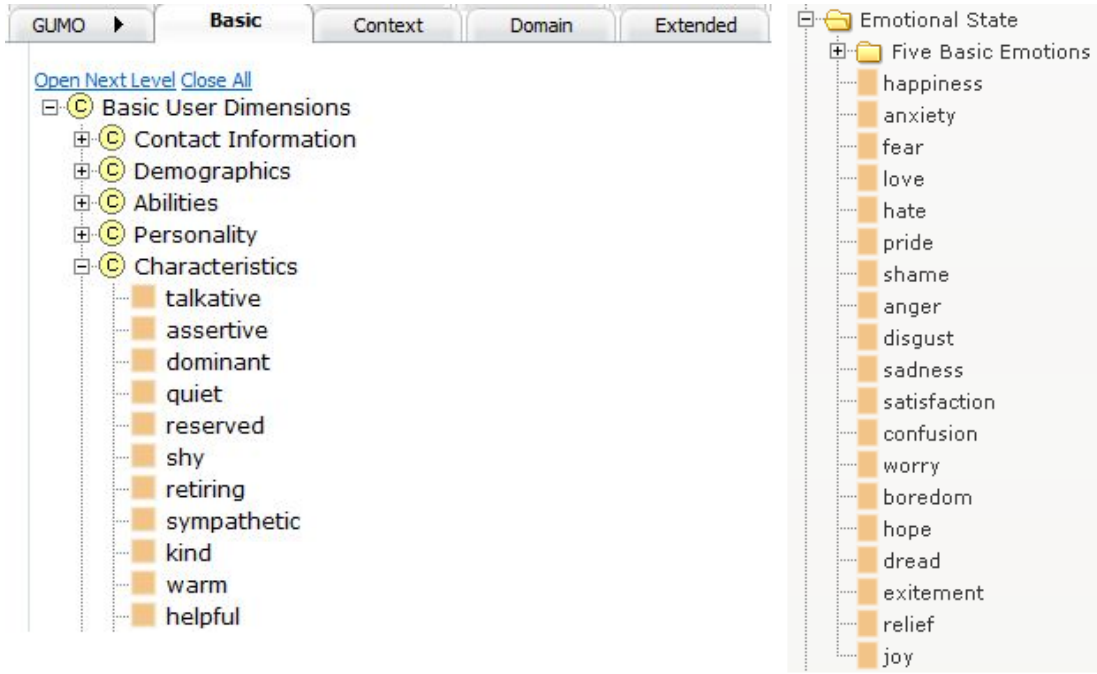


Figure: Screenshot of GUMOs user interface

UUCM - Unified User Context Model

UUCM [Mehta et al., 2008] models several features of the user and her situation:

- cognitive characteristics (areas of interests, competence, preferences),
- usage data (current task, task role, task history),
- social data (online relationships the user is involved in),
- environment data (device, current time, language, location)..

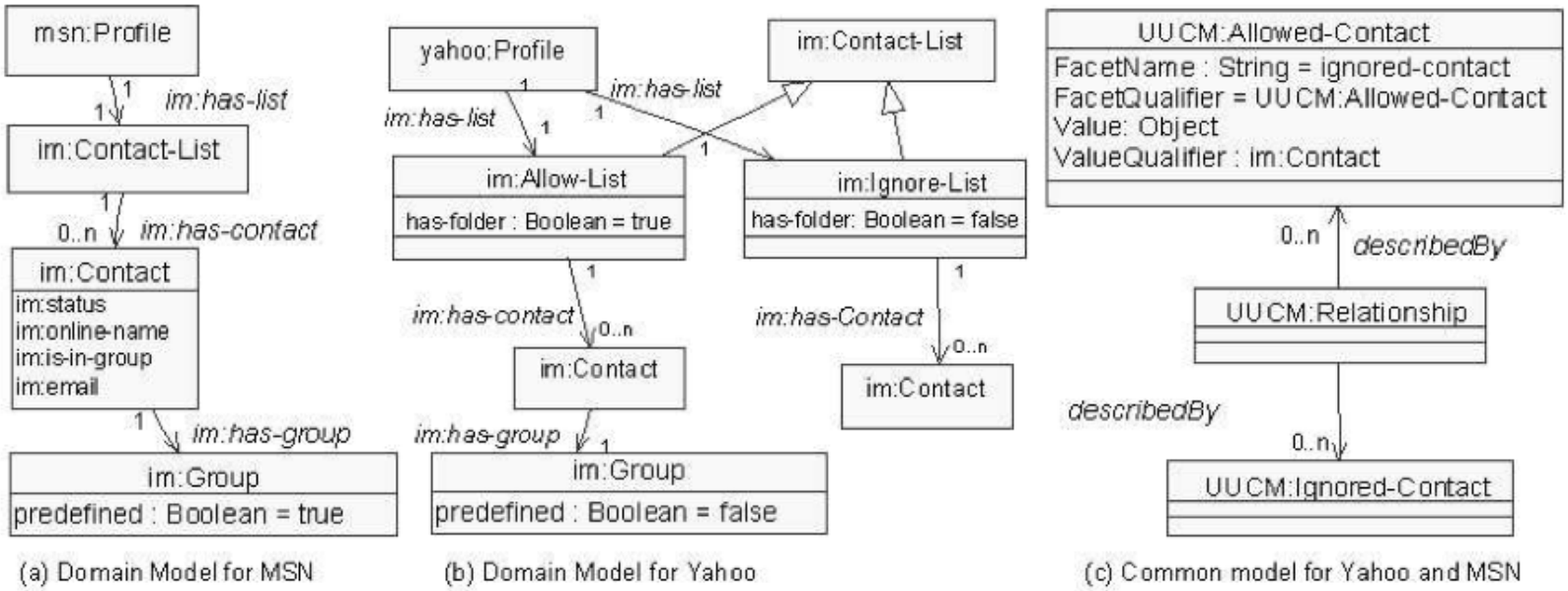


Figure: Part of the UUCM model

The Semantic Web User Model

"[Earlier approaches toward holistic user models] could not succeed because of their static, predefined user models while application-based user models strongly differ in the information they need to know about a user."

"As shown in GUMO, a lot of dimensions exist, but not all of them are required in the context of the Social Web."

"The SWUM is a collection of different linked entities that give a complete picture of the features needed to cover all aspects of the Semantic Web."

Till Plumbaum (2015). User Modeling in the Social Semantic Web. Ph.D. Thesis at the Technical University of Berlin.

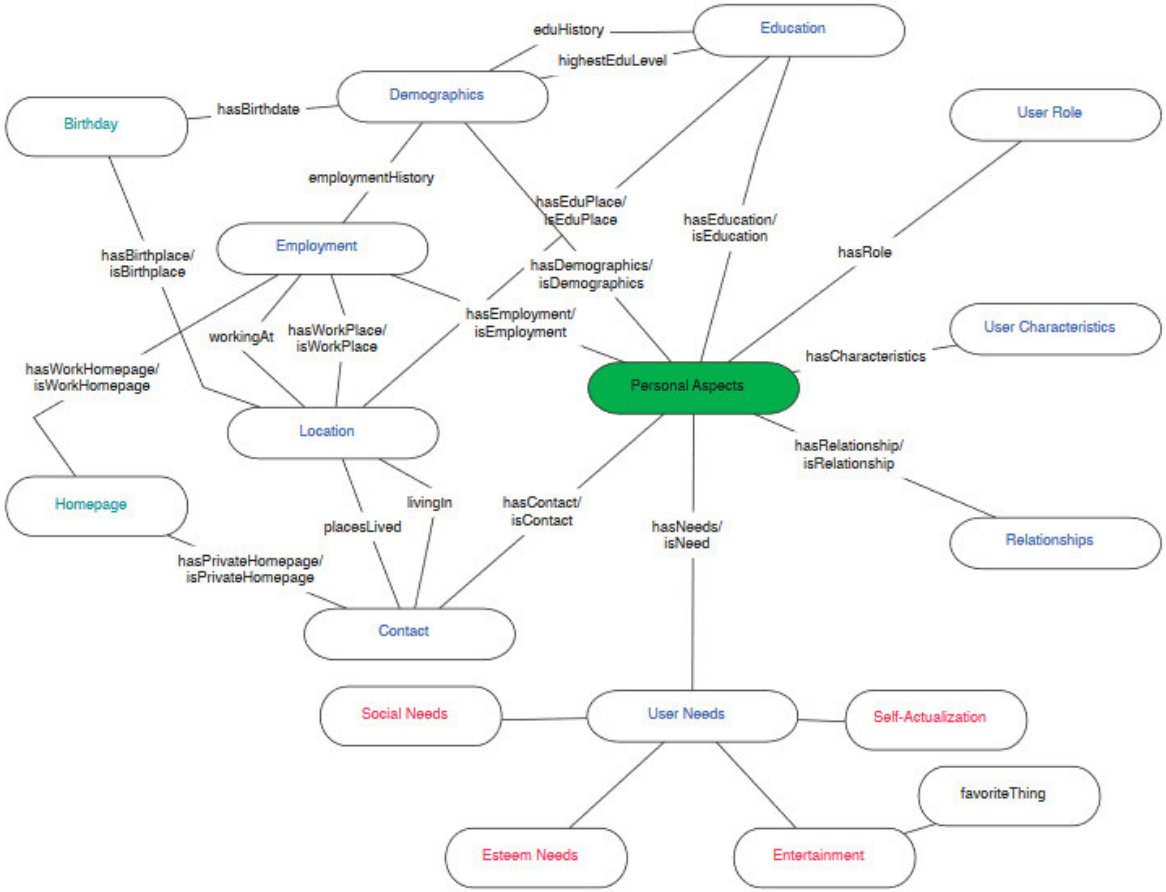


Figure: The SWUMO ontology

Open Standards for User Profiling

Individual user profiling

- Microformats: vCARD, hCARD
- Ontologies: GUMO, UUCM
- XML-based languages: APML, Weighted Interest Vocabulary
- E-learning vocabularies: IEEE PAPI, IMS LIPS

Social user profiling

- Microformats: XMS
- Ontologies: FOAF, SIOC

Enriching User Models with the Social Web

Users leave different types of profile traces on the Social Web.

People fill out their profile attributes, such as name, affiliations, etcetera.

Social tagging systems capture tagging activities of the users to create *tag-based profiles*.

Form-based profiles

The form-based profile of a user u is a set of attribute-value pairs.

$$UM(u) = \{\{a, v\} | a \in A_{UM} \text{ and } v \text{ is in the range of } a\}$$

A_{UM} defines the vocabulary of attributes that can be applied to describe characteristics of the user u. The value v associated with an attribute a must be in the range of a.

Traditional attributes might be name or email address:

- UM(u_1) = {(name, 'Bob'), (email, `bob@mail.com`)}

Tag-based profiles

The *tag-based profile* of a user u is a set of weighted tags where the weight of a tag t is computed by a certain strategy w with respect to the given user u.

$$P(u) = \{\{t, w(u,t)\} | t \in T_{source}, u \in U\}$$

$w(u,t)$ is the weight that is associated with tag t for a given user u. T_{source} is the source set of tags from which tags are extracted for the tag-based profile $P(u)$.

For example, P(u_1) = {(research, 0.65), (semantic web, 0.2), (jazz, 0.15)}

Aggregation of form-based profiles

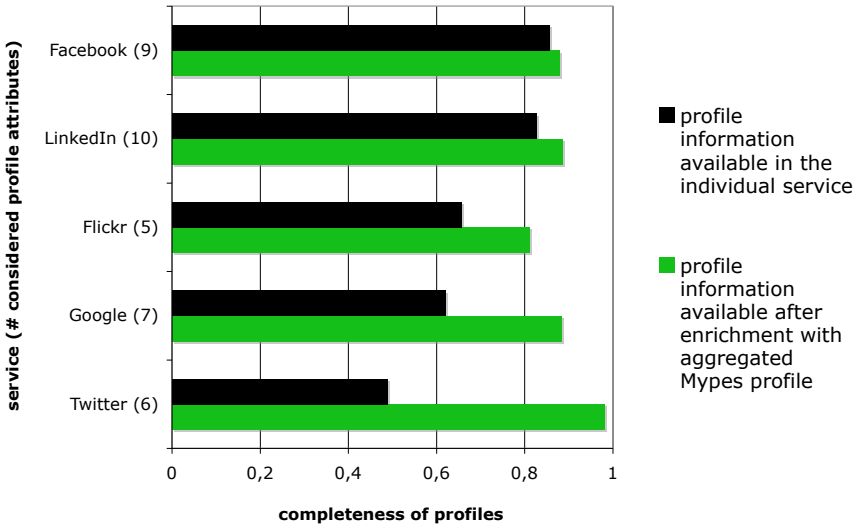
As they serve different purposes, all Social Web Platforms collect their own sets of user data.

form-based profile attributes	Face-book	LinkedIn	Twitter	Blog-Spot	Flickr	Delicious	Stumble Upon	Last.fm	Google
nickname	x	x	x	x	x	x	x	x	x
first name	x	x							
last name	x	x							
full name	x	x	x		x				x
profile photo	x		x		x				x
about		x							x
email (hash)	x				x				
homepage	x	x	x						x
blog/feed			x	x	x	x	x	x	
location		x	x		x				x
locale	x								
interests		x							
education		x							
affiliations	x	x							
industry		x							

Fabian Abel, Eelco Herder, Geert-Jan Houben, Nicola Henze, Daniel Krause. Cross-system User Modeling and Personalization on the Social Web. UMUAI (2-3), 2013, pp 169-209

Completeness of user profiles

Users often do not fill out their profiles completely. For example, Twitter only asks 6 attributes, but these profiles are only completed up to 49%.
Aggregating data from different sources leads to more complete user profiles.



Fabian Abel, Eelco Herder, Geert-Jan Houben, Nicola Henze, Daniel Krause. Cross-system User Modeling and Personalization on the Social Web. UMUAI (2-3), 2013, pp 169-209

Aggregation of tag-based profiles

How useful is a tag-based profile? Intuitively, if the profile contains just one tag, this profile is not very useful.

The profile becomes more varied if other tags are included - in particular if there is not one tag that appear 8 out of 10 times in the profile.

One way to measure the usefulness of a tag-based profile is the *entropy* of the collection.

Entropy

Entropy is a measure of disorder, or more precisely unpredictability.

For example, a series of coin tosses with a fair coin has maximum entropy, since there is no way to predict what will come next.

A string of coin tosses with a two-headed coin has zero entropy, since the coin will always come up heads.

Most collections of data in the real world lie somewhere in between.

Calculating Entropy

The entropy of a tag-based profile T, which contains of a set of tags t, is computed as follows:

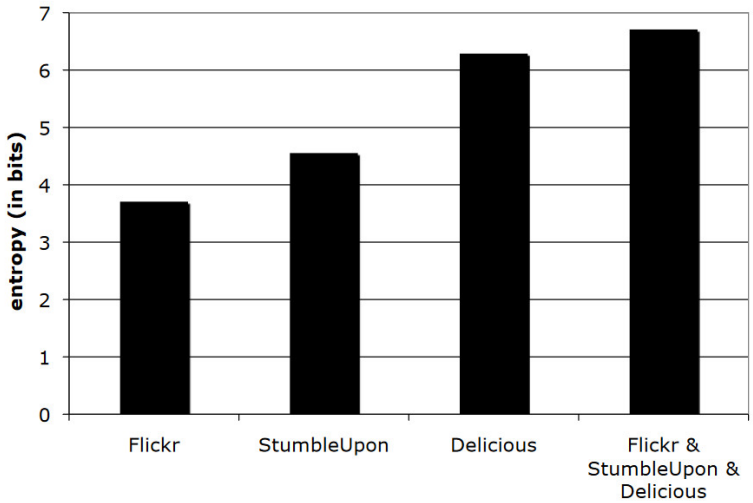
$$entropy(T) = \sum_{t \in T} p(t) \cdot self\text{-}information(t) \quad (1)$$

In Equation 1, $p(t)$ denotes the probability that the tag t was used by the corresponding user. Self-information is the logarithm of $p(t)$ multiplied by -1:

$$self\text{-}information(t) = -log_2(p(t)) \quad (2)$$

User data used for tag-based entropy

For modeling the probability $p(t)$ that a tag t appears in a given user profile, we apply the individual usage frequencies of the tags: for a specific user u the usage frequency of tag t is the fraction of u's tag assignments where u referred to t.



Tag entropy in Delicious, StumbleUpon and Flickr. In the photo sharing service Flickr, entropy is lowest.

Overlap between user profiles

Aggregation of tag-based profiles is meant to reveal more information about the users than the profiles available in some specific service.

If you combine two more or less identical profiles, the added value is rather low. If you combine two completely different profiles (e.g. your favorite music with your research interests), the entropy will increase, but the aggregated profile is probably less useful.

Therefore, there needs to be at least *some* overlap between the profiles that you combine.

Calculating overlap between profiles

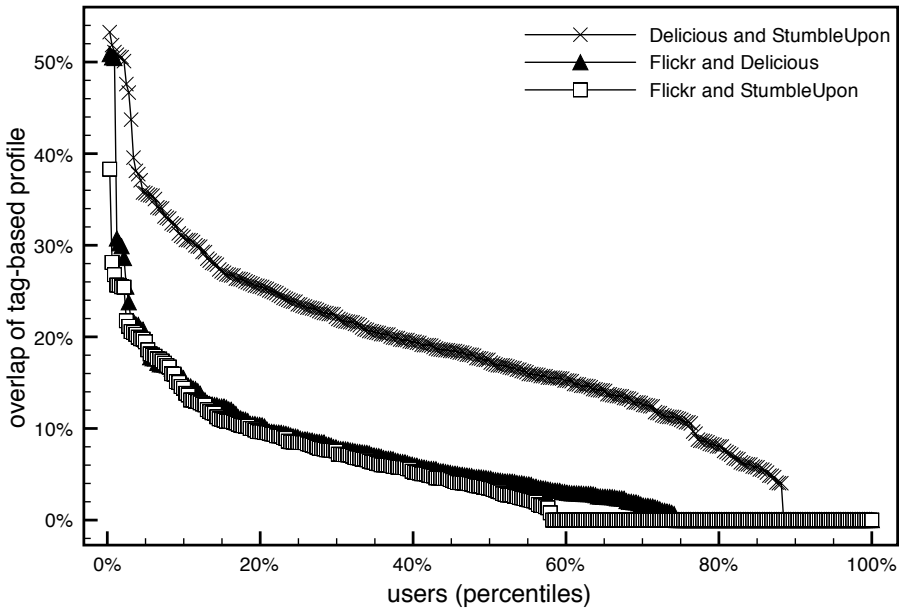
For each user u and each pair of services A and B, we compute the overlap as specified in Definition 3.

$$overlap(u_A, u_B) = \frac{1}{2} \cdot \left(\frac{|T_{u,A} \cap T_{u,B}|}{|T_{u,A}|} + \frac{|T_{u,A} \cap T_{u,B}|}{|T_{u,B}|}\right) \quad (3)$$

$T_{u,A}$ and $T_{u,B}$ denote the set of (distinct!) tags that occur in the tag-based profile of user u in service A and B respectively. Hence, $|T_{u,A} \cap T_{u,B}|$ is the number of (distinct) tags that occur in both profiles, u_A and u_B.

Calculating overlap in this way compensates for differences in profile size of services A and B.

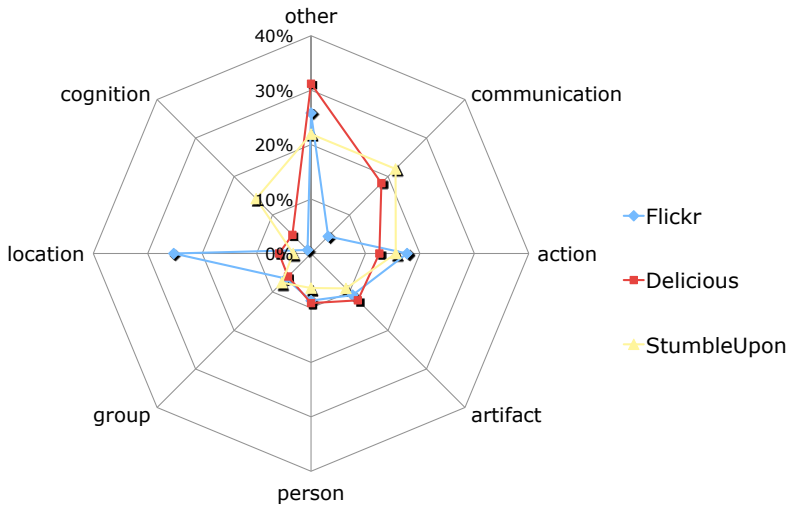
Example profile overlaps



The social bookmarking sites Delicious and StumbleUpon have the biggest overlap, but still rather small (for more than 50% of the users less than 20%). The photo sharing site Flickr is different from the other two.

Differences in types of tags between systems

Based on a mapping of tags to Wordnet categories.

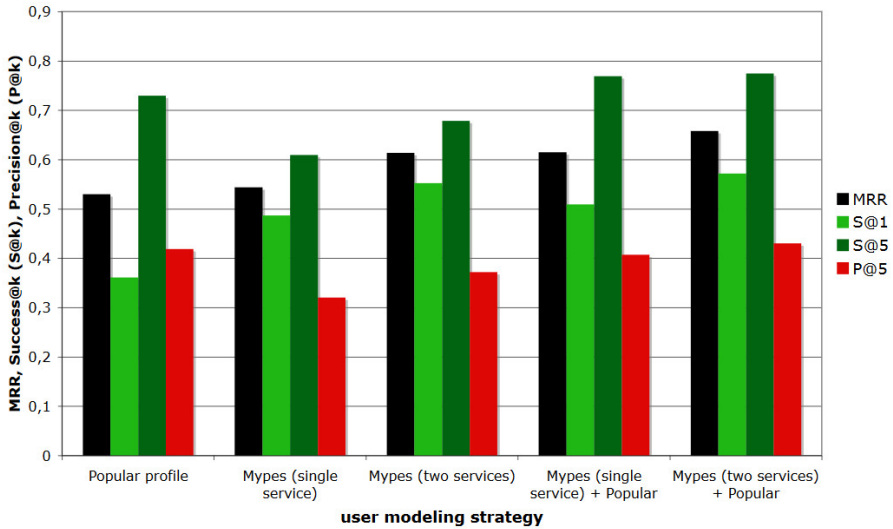


Most tags on Delicious and StumbleUpon are about *communication*. In Flickr, most tags denote *locations*.

Benefits of aggregated profiles for tag recommendation

In an experiment, we showed that using the user's own tags from other services performs significantly better than using the most popular tags of the target system.

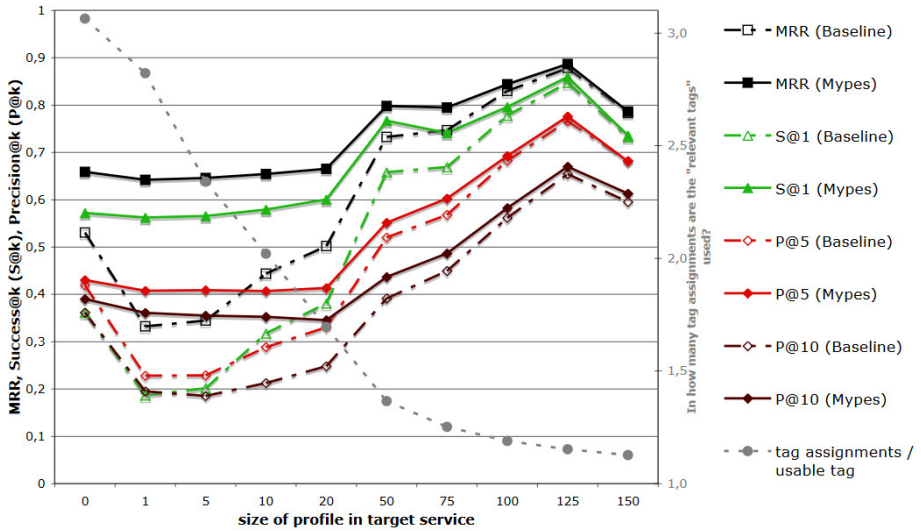
A mixture of the user's own tags and the most popular tags performs even better.



Limitations of aggregated profiles

Using the user's own tags from other systems is particularly useful for solving the *cold start* problem.

When the target profile size exceeds 100 tags, the performance differences are no longer significant.



PROPAGATION OF USER INTERESTS

Propagation of user interests

On ontology-based user model, it is possible to perform two types of propagation (i.e. spreading activation):

- edge-based propagation: propagation of user interests following the IS-A relations in the ontology [Cena, Likavec, Osborne 2013]
- property-based propagation: propagation of user interests to nodes with properties similar to the starting node [Likavec, Osborne, Cena, 2015]

Propagation process

1. Precompute the values needed for propagation (distance or similarity among objects);
2. Capture the user interest for some concepts in the ontology (direct or indirect feedback);
3. For each object *N* receiving feedback from user *U*:
   - calculate the sensed interest value for *N* and store it in the user model of U.
   - calculate the propagated interest value for *N*.

Values needed for user interest propagation

- edge-based propagation: the conceptual distance between each pair of concepts in the conceptual hierarchy;
- property-based propagation: the similarity among all the objects in the ontology, based on number of property in common.

Sensed interest

A measure of how much of the direct user feedback each node can sense. It depends on the direct user feedback for the object and its level in the taxonomy. The idea is that low level concepts weight more since more specific.

$$\mathcal{I}_S(N) = \frac{\ell(N)}{max} sig(F(N) + f(N))$$

- $\ell(N)$ - level of the node receiving the feedback, max - level of the deepest node;
- $sig(x)$ - variant of the sigmoid function $sig(x) = \frac{e^x - 1}{e^x + 1}$,
- $F(N)$ - sum of the prev. feedback from the user for N,
- $f(N)$ - current (new) feedback from the user for N.

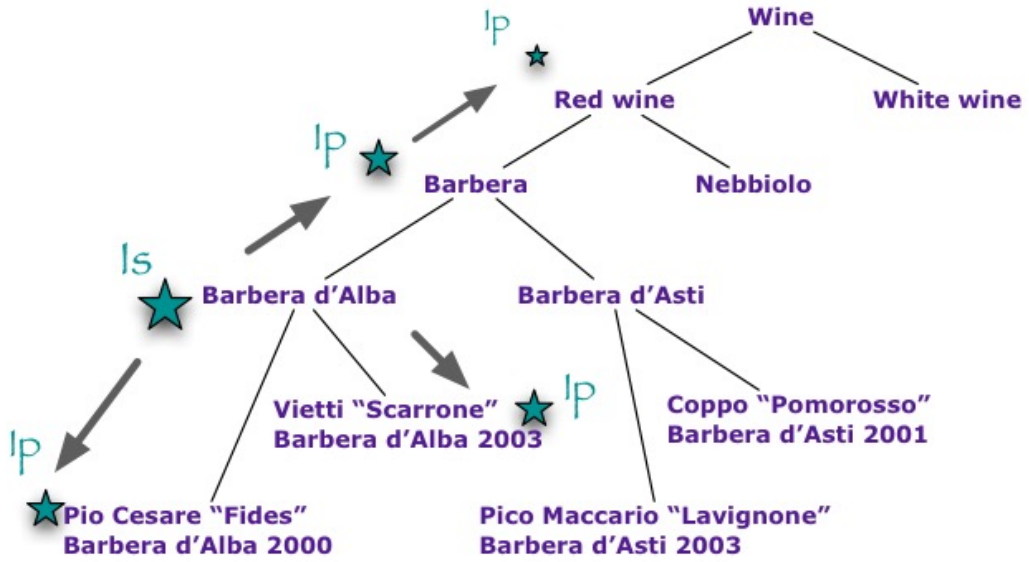
Propagated interest - edge-based

Propagate vertically in the taxonomy based on the distance between nodes. Propagated interest values depend on the original sensed interest of N and on the conceptual distance between the starting node and the node receiving the propagated value.

$$\mathcal{I}^0_{\mathcal{P}}(M,N) = \mathcal{I}_{\mathcal{S}}(M)e^{-k\cdot dist(M,N)}$$

- $\mathcal{I}_{\mathcal{S}}(M)$ - sensed interest of the node M receiving the feedback,
- $dist(M,N)$ is the *conceptual* distance (i.e. exponentially decreasing distance) between the node M receiving the feedback and the node N receiving the propagated interest,
- $k \in \mathbb{R}$ is a constant, called *attenuation coefficient*.

Edge-based propagation



Edge-based propagation

Other examples of Edge-based propagation:

- Sieg et al. (2007, 2010): consider the ontology as as a semantic network and the propagation (bottom-up) depends on the weight of the relation.
- Middleton et al. (2004): the propagation (bottom-up) of interest values in the taxonomy is based on is-a similarity and static (50percent of this value is spread to its super class.)

Propagated interest - property-based

Propagate to all the objects similar to N. Propagated interest depends on the sensed interest of N and on the similarity between N and the object receiving the propagated value.

$$\mathcal{I}_{\mathsf{P}}^{sim,0}(N,M) = \frac{e^{2\mathrm{SIM}(N,M)} - 1}{e^{2\mathrm{SIM}(N,M)} + 1}\mathcal{I}_{\mathcal{S}}(N)$$

- $\mathcal{I}_{\mathcal{S}}(N)$ - sensed interest of the object N receiving the feedback;
- $\mathrm{SIM}(N,M)$ - *similarity* between the object N receiving the feedback and the object P receiving the propagated interest.

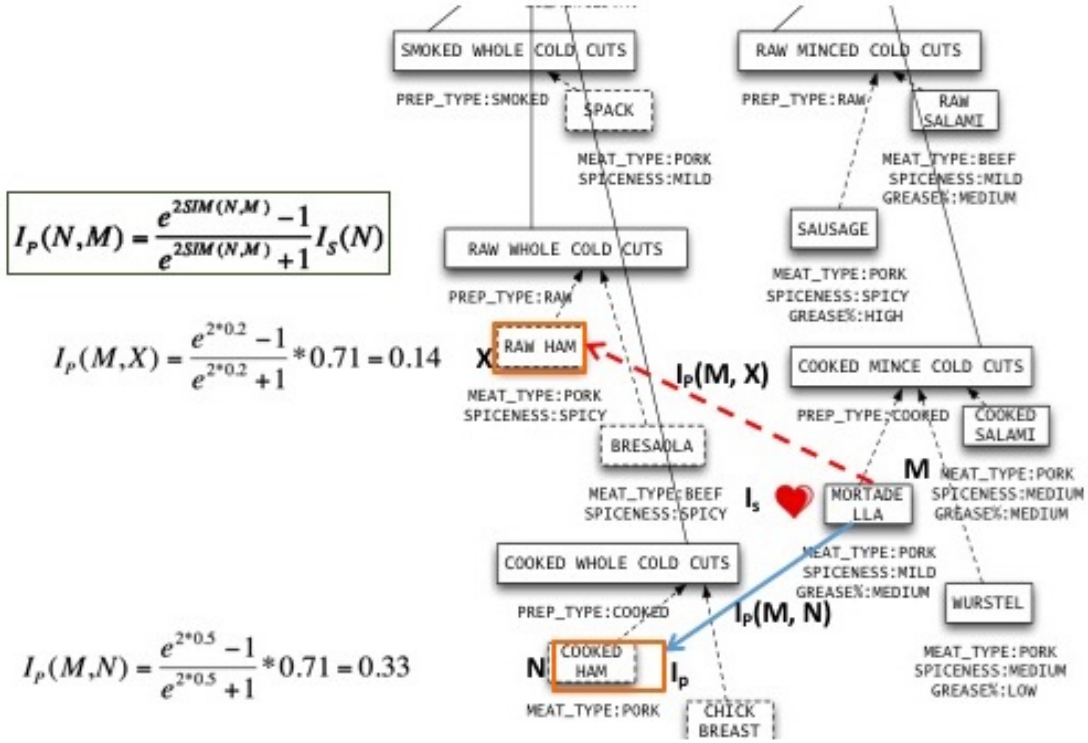
Similar for $\mathcal{I}_{\mathsf{P}}^{rel,0}(N,P)$.

Property-based propagation

$$I_P(N,M) = \frac{e^{2SIM(N,M)} - 1}{e^{2SIM(N,M)} + 1} I_S(N)$$

$$I_P(M,X) = \frac{e^{2*0.2} - 1}{e^{2*0.2} + 1} * 0.71 = 0.14$$

$$I_P(M,N) = \frac{e^{2*0.5} - 1}{e^{2*0.5} + 1} * 0.71 = 0.33$$



Property-based propagation

Other examples of property-based propagation are:

- Thiagarajan et al. (2008) represent user profiles as bags-of-words (BOW) where a weight is assigned to each term describing user interests. Then, they consider ontological relationships between the terms in BOW to propagate the values using graph spreading activation techniques.
- Heitmann and Hayes (2014), constrained spreading activation is used among RDF graph

Limitations

- edge-based propagation: need for a deep taxonomy (not flat), rigid;
- property-based propagation: need for an ontology with explicitly defined properties, need to consider relevance of properties.