Personalized information filtering for mobile applications

Christian Reuschling and Sandra Zilles
DFKI GmbH
Erwin-Schrödinger-Str. 57
67663 Kaiserslautern, Germany
{christian.reuschling,sandra.zilles}@dfki.de

Abstract

Recent technological development has enhanced research in the field of pervasive computing for mobile applications. In particular, topics like ad-hoc community building and personalized contextual product offers are of relevance for cellular radio providers. In this context, we propose a generic approach to personalized information filtering.

This concerns first an appropriate representation scheme for the application domain, the user, as well as the information to be filtered. Here we propose to model the domain in an ontology with special weight attributes in RDFS, such that personal interests or resources (e. g., product descriptions) can be represented as RDF instances of this ontology.

Using case based reasoning techniques, we second propose an implementation of a similarity measure between such instances. On the one hand, given a special domain model, this similarity measure allows for filtering a list of resources according to a person's interests in a way immediately suitable for the intended applications; on the other hand, this similarity measure is defined generally enough to allow for the comparison of RDF instances in general, with different specialized similarity measures depending on the intended semantics of similarity.

Third, in addition, the problem of how to maintain representations of user interests and resource descriptions in a dynamic domain is addressed briefly.

1. Introduction

Recent technological development, such as concerning UMTS and smartphones, has enhanced research in the field of pervasive computing for mobile applications. In particular, topics like socializing, ad-hoc community building, entertainment on demand, and personalized contextual product offers are of relevance for cellular radio providers and for the corresponding third party providers. What many products cellular radio providers aim at have in common is the fact that they implement

- a model for representation of the user,
- a model for representation of a user's context[1],
- a model for representation of available information or resources (e. g., products, entertainment offers, other users[2]),
- a scenario for filtering and recommendation of information or resources,
- a process for content acquisition (including maintenance of the information represented).

Here the main difference compared to classical, non-mobile applications is the focus on the user's context, in particular, e. g., her current location, the current daytime, current and/or previous tasks, active applications and services, etc., cf. [15, 9, 14]. Roughly

[1] i. e., additional constraints describing the current situation of a user's fluently changing environment

[2] Note that users themselves can be resources, for instance, in applications focusing on socializing scenarios.

speaking, the user's context is defined by the current status of her fluently changing environment, see Section 2.3 for more details.

However, the actual implementations of these models, scenarios, and processes differ in terms of methods used. The variety of the approaches of course results from different applications and thus different intended semantics of user/resource representations and different intended recommendation features. State of the art approaches to modeling personal taste for recommendation tasks are for instance discussed in [11], cf. also the references therein. Ideas especially aiming at mobile applications often concern socializing based on, for instance, music recommenders, cf. [2, 3].

The aim of this paper is to propose a unified framework allowing for a generic implementation of the required models, scenarios, and processes – independent from the actual application domain. This framework is based on previous work concerning

- Semantic Web technologies (here used for the representation model); see [6, 12] and
- case based reasoning methods (here used for the recommendation model); see [4, 5], as well as [1] for a fundamental overview.

The fundamental approaches combined are not new; however, the combination of Semantic Web technology with case-based reasoning methods in a unified framework as proposed below hopefully provides a fundament for fruitful application-oriented research.

Figure 1 sketches the underlying scenario in the scope of the applications our approach addresses. Note that the term 'case base' here represents a database containing all profiles – the latter being regarded as cases in a case-based reasoning approach. We will return to this point of view in Section 3. Basically, the idea is that a user will receive personalized, filtered information via a mobile device (e. g., cell phone). The recommendation process is to first use context information for filtering the case base, and then select appropriate 'cases' (i. e., profiles) depending on a similarity match with the given user profile. The system has to be maintained in a clearly defined workflow, for instance by (but not restricted to), using external databases.

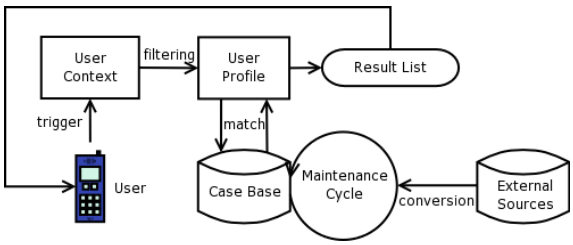

Figure 1: The general scenario.

The paper is organized as follows: we first motivate and describe our chosen model for representing user and resource profiles, i. e., descriptions of their interests or features, (Section 2), then, in the main section (Section 3) we discuss our generic recommender approach, followed by a brief overview over possible approaches to content acquisition (Section 4). In the concluding section, we address some possible issues for future work in this context.

2. Representation model: the RDF 'boost factor' framework

2.1. Basic requirements

Let us first motivate our choice of the model for representing user interests (i. e., user profiles) and resource properties (i. e., resource profiles).

First of all, note that, particularly for mobile applications, a user should never be described by her profile alone, but also by her current context. Substracting context from a user description, it is straightforward to use only a single model both for user profiles and for resource profiles. Requirements for a corresponding representation scheme should be:

1. The representation scheme must be general enough to express different

kinds of real-world concepts and their relations. In particular it must provide options for

   (a) representing taxonomies of concepts (in order to allow for proper recommendations of specialized to generalized profiles and vice versa),

   (b) representing the amount of user interest in each of the real-world concepts (in order to allow for more specially personalized recommendations),

   (c) transferring user or resource profiles from (publicly) available databases into the predefined representation scheme (in order to allow for content acquisition in a recommender system),

   (d) modifying a given domain model as well as the profiles (in order to allow for reflecting the changes in a dynamic application domain).

2. The representation scheme must be specialized enough to be suitable for efficient content acquisition and recommendation. In particular, it must allow for

   (a) an *efficient* extraction and transformation of external data,

   (b) an *efficient* computation of similarities between profiles (in order to determine recommendations),

   (c) *efficient* profile generation and update tools.

In particular, the need for feasible content acquisition methods suggests using standards in the basic representation scheme. Moreover, basing on the requirement for representing taxonomies (profiles express some special interests or features of a collection of – maybe inter-related – concepts in the application domain), we chose an ontology-based approach for representation. That means, concepts of the domain are modeled in an ontology, using a representation scheme expressive enough for representing taxonomies, whereas user and resource profiles are instances of this ontology (respecting some particular features, as will be explained in Subsection 2.2).

Currently, there are several standards for ontology representation schemes, e. g., RDFS, OWL (OWL Full and restrictions OWL DL and OWL Lite), etc., see also [17, 16]. On the one hand, a language like OWL Full is very expressive, but on the other hand, this expressiveness entails high costs in the tasks resulting from our requirements. Therefore a first reasonable step towards a generic framework for recommender systems is to choose RDFS/RDF for modeling the domain ontology and the profiles (as instances).

Note that an additional benefit from using RDF might be that RDF is the language of the Semantic Web, where many users express their personal interests within a net of linked RDF resources. This may entail further options for extracting contents for a recommender system.

Moreover, the quite simple structure of RDFS may have usability benefits; for real-world applications it would be rather inconvenient to model in OWL Full.

2.2. From general RDF to interest profiles

So we propose to model the application domain in an RDFS ontology; profiles are instances of the latter. However, some detailed requirements have to be taken into account, in order to use RDF specifically for representing interest profiles. These detailed requirements concern

1. the attribute types considered,

2. attributes for user rating in interest profiles,

3. degrees of relationships between concepts in the ontology.

Attribute types

Up to a certain degree of granularity, a simple metadata scheme may be sufficient for describing all relevant information about a person or a resource. However, one immediately recognizes limitations of such ontology-based metadata schemes. For instance, music is very hard to describe using metadata only, state-of-the-art music mining and retrieval techniques prove much more information in an audio file itself than can be expressed with simple metadata. Similar statements hold for text documents.

Thus including attributes of type mp3 or string in the ontology allows for integrating audio files or any kind of textual reviews/summaries (of music, literature, any kind of products, etc.) into a profile. Hence profiles can be enriched by additional information.

By the way, this also has a positive effect in easing profile generation, as will be discussed in Section 4.

Attributes for user rating

As already indicated, it should be possible to instantiate a concept in an ontology such that a person can express liking or disliking the class of objects associated to that concept. The motivation behind is that – under an open-world assumption – in general we cannot conclude a person dislikes a concept, if it is not instantiated in the person's profile.

Our simple approach here is to demand that each concept in the ontology can have a special 'weighting' attribute, a float attribute with range $[-1, +1]$ reserved for expressing the so-called 'boost factor'. In a person's interest profile:

- a positive boost factor for some concept c indicates that the person rates c positively; the absolute value is interpreted as a weight in this rating,
- a negative boost factor for some concept c indicates that the person rates c negatively; the absolute value is interpreted as a weight in this rating,
- a boost factor of 0 for some concept c indicates that the person is indifferent concerning c.

In a resource profile, the same kind of boost factor is reserved.

Note that the open-world assumption is reflected in our approach as follows: If, in a profile, a concept c is not instantiated, the intended semantics would not be equal to the case when c is instantiated with a boost factor of 0. The open-world assumption forbids us to interpret the rating of c as 'indifferent' (for persons) or 'non-existing' (for resources).

Degrees of concept relationships

Assume a taxonomy of concepts is modeled in an ontology. Then the intended semantics probably cannot always be reflected in the taxonomic structure only, as the following example shows:

Assume the ontology contains concepts c, c_1, and c_2, where c_1 and c_2 are subconcepts of c, see Figure 2.

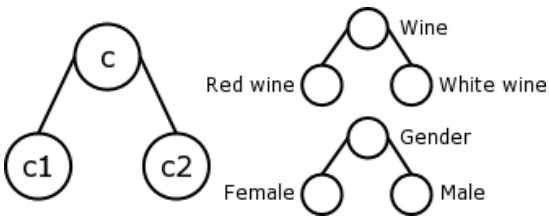

Figure 2: A simple example for taxonomies.

Then c_1 and c_2 are 'brother' concepts and could be interpreted as related. But the degree of their relation is not clear without any additional specification which might depend on the intended application. Focusing on recommender systems, a strong relation would mean that profiles instantiating c_1 could be recommended to profiles instantiating c_2. Obviously, the intended recommender functionality depends on some kind

of *degree* of relationship you would like to assign to c_1 and c_2. If the intended semantics would say that 'red wine' is closely related to 'white wine', but 'female' is not (closely) related to 'male' (which should definitely have an impact on how to compute recommendations), this requires some additional representation within the ontology. In particular, the concept c should be annotated with some value expressing the degree of relationship of its child concepts.

Thus, in the proposed generic model, each concept in the ontology has a special numerical attribute (e. g. with range $[0, 1]$) used for representing the degree of relationship of its child concepts. Extreme values then are interpreted as the maximum or minimum possible degree of relationship, i. e. if the range is $[0, 1]$, then the value 0 for c means the sons of the concept c are not related at all; the value 1 means that the sons of c are just representations of only one subconcept of c.

2.3. Context in addition to a user profile

A generic representation model for user context is not so easy to define. In general, there is no universal definition of the term context, cf. [7, 8], though this topic has gained a lot of interest in the scientific community, see also [9, 14]. In Section 1, we explained the user's context as the current state of her fluently changing environment – but this is far from being a clear definition. Obviously, in mobile applications, location and time should be part of a user's context, but it is unclear, how these should in general be evaluated by a recommender system. A generic approach would have to allow for reflecting any particular application-specific definition of context and its semantics.

Up to now, we have not yet elaborated a generic context model for mobile applications, though in several realizations of our proposed approach (recommender systems for leisure time, for shopping scenarios, and for media recommendations) the user's context has been integrated.

Currently, we define context from a more functionality-driven point of view: instead of asking first what context is, it is reasonable to ask first what context should be used for in the application. In our current model, we define the user's context by all constraints that are used for pre-filtering information (in a database containing all profiles) before computing recommendations via similarities or for weighting the overall similarity values between profiles, which are determined for computing recommendations.

Note that it may be worth analyzing to what extent a user's context could also be modeled as part of her interest profile, using the RDFS framework as explained above.

3. Recommender model: a generic similarity measure

From now on, suppose an ontology and a list of profiles have been defined according to our special RDFS/RDF representation approach. Additionally, assume some context information is attached to the profiles,

Given a profile p, how should recommendations for p be computed?

Let us first assume the existence of algorithms for computing

1. a similarity $\mathrm{sim}(\mathrm{p}, \mathrm{p}')$ between the profile p and any other profile p' (according to some similarity measure),
2. a context relevance $\mathrm{con}(\mathrm{p}, \mathrm{p}')$ of any other profile p' to profile p.

Then the recommendation process could be sketched as follows:

parameters: thresholds θ_s, θ_c, θ, integer k
input: list L of all profiles,
query profile $p \in L$
output: list of profiles in L
(recommendations for p),
with similarity values in $[0, 1]$

1. (* optional *) Let $L' \subseteq L$ be the list of all profiles $p' \in L$, for which $\mathrm{con}(\mathrm{p}, \mathrm{p}') < \theta_c$; $L := L \setminus L'$;

2. (* options: a, b, c, d *)

   (a) For all profiles $p' \in L$ with $\mathrm{sim}(\mathrm{p}, \mathrm{p}') \geq \theta_{\mathrm{s}}$, return $(p', \mathrm{sim}(\mathrm{p}, \mathrm{p}')$.

   (b) For all profiles $p' \in L$ with $\mathrm{sim}(\mathrm{p}, \mathrm{p}') \cdot \mathrm{con}(\mathrm{p}, \mathrm{p}') \geq \theta$, return $(p', \mathrm{sim}(\mathrm{p}, \mathrm{p}') \cdot \mathrm{con}(\mathrm{p}, \mathrm{p}')$.

   (c) Return $(p_1, \mathrm{sim}(\mathrm{p}, \mathrm{p}_1))$, ..., $(p_k, \mathrm{sim}(\mathrm{p}, \mathrm{p}_\mathrm{k}))$, for profiles $p_1, \ldots, p_k \in L$, such that $\mathrm{sim}(\mathrm{p}, \mathrm{p}_\mathrm{i}) \geq \mathrm{sim}(\mathrm{p}, \mathrm{p}')$ for all $i \in \{1, \ldots, k\}$ and all profiles $p' \in L$.

   (d) Return $(p_1, \mathrm{sim}(\mathrm{p}, \mathrm{p}_1) \cdot \mathrm{con}(\mathrm{p}, \mathrm{p}_1))$, ..., $(p_k, \mathrm{sim}(\mathrm{p}, \mathrm{p}_\mathrm{k}) \cdot \mathrm{con}(\mathrm{p}, \mathrm{p}_\mathrm{k}))$, for profiles $p_1, \ldots, p_k \in L$, such that $\mathrm{sim}(\mathrm{p}, \mathrm{p}_\mathrm{i}) \cdot \mathrm{con}(\mathrm{p}, \mathrm{p}_\mathrm{i}) \geq \mathrm{sim}(\mathrm{p}, \mathrm{p}') \cdot \mathrm{con}(\mathrm{p}, \mathrm{p}')$ for all $i \in \{1, \ldots, k\}$ and all profiles $p' \in L$.

In other words, one usually recommends either all resources with a minimum similarity or just k resources with highest similarity. The context relevance may be used for pre-filtering and/or as a weight for the overall similarity.

So a generic framework requires a definition of a suitable similarity measure, and thus a generic algorithm for matching RDF files. We propose a matching algorithm which should be a fixed component in any concrete application task, such that, with each new application, it remains only to define a new ontology and a new context model (the latter including definitions of how to compute context relevance values). That means, the matching algorithm proposed is generic; its parameters are the ontology and the algorithm for computing the context relevance. Basically, the desired similarity measure is implemented in this matching algorithm in the sense that the latter compares two profiles and returns a similarity value in $[0, 1]$.

Requirements for such a similarity measure should concern local similarities, which must be defined for all relevant attributes, as well as the global (overall) similarity.

Local similarity

Concerning local similarities, there is a need for different type-specific similarity measures, e. g., for audio files of the types occurring in the ontology, for html documents, for general string values, and for numerical values. Here different similarities must be defined for different intended semantics of certain attributes:

Two strings s, s' can have a similarity defined by

1. 0, if $s \neq s'$ and 1, if $s = s'$; or
2. $1 - l(s, s')$, where $l(s, s')$ is the Levenshtein distance of s and s' (normalized in the interval $[0, 1]$), cf. [10]; or
3. ...

Two numerical values r, r' can have a similarity defined by

1. $1 - d$, where d is their absolute distance (normalized in the interval $[0, 1]$); or
2. 0, if $r \neq r'$ and 1, if $r = r'$; or
3. 0, if $r < r'$ and 1, if $r \geq r'$; or
4. 0, if $r > r'$ and 1, if $r \leq r'$; or
5. ...

Similar variants are conceivable for different types of attributes.

Note that each variant of a local similarity measure goes along with different intended semantics of the corresponding attribute. Consequently, this has to be expressed by formally different types of string attributes or different types of numerical attributes in the ontology.

The need for these different semantics is obvious: If, for instance, a numerical attribute represents a production year of a movie a user is interested in, then the local similarity for this attribute should be defined using their absolute distance, since the user may presumably like movies from that time, but not only from that year. The intended local

similarities would of course be different, if the attribute should represent some maximal allowed price for recommended products.

Global similarity

Note that, as for the case of local similarities, we assume $\mathrm{sim}(\mathrm{p}, \mathrm{p}') \in [0, 1]$ for all profiles p, p'.

The overall similarity of two profiles should take into account the local similarities concerning different attribute values, as well as the degree of relationships between the instantiated concepts. Moreover, the user's interests expressed in the boost factors must define 'weights' in computing a similarity. Note that the boost factor is a special kind of attribute, the value of which is not included into the computation of similarities in the usual way. The boost factor merely determines the amount of how strongly the local similarity will contribute to the global similarity. Thus it works like a weight, though it is not really a weight, due to the fact that the boost factors can have arbitrary values in $[0, 1]$, without requiring that they sum up to 1, as would be usual for weights.

Here different requirements concerning a similarity measure are conceivable:[3]

1. Reflexivity
   One might or might not require that $\mathrm{sim}(\mathrm{p}, \mathrm{p}) = 1$ for all profiles p.

2. Symmetry
   One might or might not require that $\mathrm{sim}(\mathrm{p}, \mathrm{p}') = \mathrm{sim}(\mathrm{p}', \mathrm{p})$ for all profiles p, p'.

3. Triangle inequality
   One should in general not require that $\mathrm{sim}(\mathrm{p}, \mathrm{p}') \geq \mathrm{sim}(\mathrm{p}, \mathrm{p}'') + \mathrm{sim}(\mathrm{p}'', \mathrm{p}') - 1$ for all profiles p, p', p''. The reason is, roughly speaking, that a user's boost factors can determine how much special local similarities influence the global similarity value. Thus, even if all local similarities fulfill the triangle inequality, this does not hold for the global similarity.

So, in what follows, we should concentrate on reflexivity and symmetry constraints.

For instance, in a shopping scenario, if a user has rated her favorite movie with title t, a resource (e. g., a DVD) with the same title t would maybe not be the top recommendation, assuming the user already possesses such a product. Hence you would require a low similarity of a profile to itself, i. e., the corresponding distance measure would be non-reflexive. On the other hand, retrieval scenarios are conceivable, where each profile has maximum similarity to itself, i. e., where the global distance measure is reflexive. Similar scenarios are conceivable disallowing or requiring symmetry.

Now, if our global similarity measure is supposed to be generic, this means that whether or not reflexivity, symmetry, or the triangle inequality hold, must depend on the local similarities exclusively. Only in this case we can guarantee that the ontology model alone (and in particular the choice of special attributes for which local similarity measures are defined) determines the properties of the similarity measure.

Consequently, the main requirements concerning the global similarity measure can be summarized as follows:

1. The global similarity of two profiles must depend on their local similarities (in particular also concerning the boost factors) and the relationship degrees defined in the ontology, only.

2. Given a local similarity or a relationship degree as a parameter, the global similarity must be monotonically increasing in this parameter.

3. The global similarity function is reflexive, if and only if all given local similarity functions are reflexive.

[3] The requirements reflexivity and triangle inequality for a distance measure d are usually $d(p, p') = 0$ and $d(p, p') \leq d(p, p'') + d(p'', p')$. We obtain the given formulations regarding $\mathrm{sim} = 1 - \mathrm{d}$ for a $[0, 1]$-valued distance measure d.

4. The global similarity function is symmetric, if and only if all given local similarity functions are symmetric.

The easiest way to achieve this is to define a global similarity by a weighted sum of all local similarities.

For this purpose we follow the case-based reasoning method proposed in [5]. Here the case base is just the profile database; each profile corresponds to a case. The query profile p is then understood as a new case, such that for each case p' (existing profile) in the case base, a similarity value for p and p' has to be computed.

The method explained in [5] can be sketched as follows:[4]

input: query profile p,
case profile p'
output: similarity value $\mathrm{sim}(\mathrm{p}, \mathrm{p}') \in [0, 1]$

1. Determine the set A of all attributes instantiated both in p and in p' (except for boost factor attributes).

2. For all attributes $a \in A$ compute a (local) similarity value $\mathrm{lsim}(\mathrm{p}, \mathrm{p}', \mathrm{a})$ as follows:

   - If a is a complex attribute, then let $\mathrm{lsim}(\mathrm{p}, \mathrm{p}', \mathrm{a}) = \mathrm{sim}(\mathrm{p_a}, \mathrm{p'_a})$, where p_a and p'_a are the parts of the instances p and p' concerning a (* recursion *).
   - Else let $\mathrm{lsim}(\mathrm{p}, \mathrm{p}', \mathrm{a})$ be the local similarity of p and p' concerning a.

3. For all attributes $a \in A$ compute a boosted local similarity value $\mathrm{blsim}(\mathrm{p}, \mathrm{p}', \mathrm{a}) = \frac{1}{2} \cdot (1 + \mathrm{lsim}(\mathrm{p}, \mathrm{p}', \mathrm{a})(1 - |\mathrm{bf}(\mathrm{p}, \mathrm{a}) - \mathrm{bf}(\mathrm{p}', \mathrm{a})|))$, where $\mathrm{bf}(\mathrm{q}, \mathrm{a})$ denotes the boost factor assigned to the attribute a in some profile q.[5]

4. Return the global similarity $\mathrm{sim}(\mathrm{p}, \mathrm{p}')$ as the weighted mean of all values $\mathrm{blsim}(\mathrm{p}, \mathrm{p}', \mathrm{a})$ for $a \in A$, where the weights are given by the degree of relationship between the concepts instantiated by p and p'.

4. Content acquisition model: generating and updating ontologies and profiles

Concerning the content in a recommender system in the proposed framework, both the ontology and the profiles have to be discussed. However, we only briefly sketch some well-known basic approaches here.

4.1. Ontology generation and update

How can an initial ontology be obtained and how can it be updated? In most cases, presumably, an initial ontology must be designed by an administrator – as usual. However, there are in fact approaches allowing for automatically computing suggestions for ontology updates, which can be used in the context of semi-automated ontology maintenance.

Non-automated workflow

Non-automated ontology modification here simply means manual editing. This only requires the provision of an interface allowing for ontology upload and download, with connection to an ontology editing tool.[6]

Semi-automated workflow

Though machine learning tools can be used for inferring new relations for the ontology

[4] Here we deal only with a simplified case, assuming that no attribute in the ontology is of a list type. The algorithm can be extended to matching profiles in which lists occur as attribute types, but we omit the relevant details.

[5] The factor $\frac{1}{2}$ is needed for normalizing onto the interval $[0, 1]$.

[6] Note that some ontology editing tools will not store the ontology in the format required in our proposal. Here parsers must be used for transformation.

or for detecting superfluous/missing concepts or attributes, ontology changes should definitely always be executed only upon the decision of an administrator.

Suggestions for an ontology update can be obtained by automatically analyzing all profiles in the database or logging user actions – for instance, using association rule mining or clustering techniques, see [18]. Such techniques are well-known from market basket analysis.

4.2. Profile generation and update

How can profiles be generated and updated? Again we only sketch some basic ideas.

Non-automated workflow

Of course, it is possible to provide user interfaces for manual editing of profiles. However, this may cause problems concerning the usability as well as because of the often observed phenomenon of users incapable of describing their own interests in terms of concepts. Still manual editing might be reasonable for creating resource profiles or modifying them.

Semi-automated workflow

In semi-automated workflows, a human administrator can be assisted by machine learning systems inferring new knowledge from user and/or resource data.

Here we distinguish between 'local' approaches using single user data and 'global' ones using a larger part of the system content for learning.

1. Local approach
   The interest profile of a user can for instance be obtained/updated by an automatic analysis from relevance feedback, see [13]. If the user rates resources, which have profiles, positively (or negatively), they can be merged into the user's profile with boost factors as given in the resource profile (or boost factors as given in the resource profile, but multiplied by -1). If the user rates resources without profiles, e. g., audio files or text documents, suitable mining methods may be used for extracting new profile features out of these.

   This local approach works for acquisition of resource profiles as well, since, in general, features of resources can also be described by relevance feedback.

2. Global approach
   User or resource profiles can be updated by an automatic global analysis of all profiles in the database (or of all user profiles of certain clusters/communities). If association rule mining or clustering methods, cf. [18], yield new relations between concepts, these can be used for predicting unknown values in single profiles.

Fully automated workflow

The most obvious approach for a fully automated workflow, as already mentioned before, is profile extraction from external databases – at least resource profiles can be automatically extracted from available databases. For instance, movie profiles could be obtained from the internet movie database (http://www.imdb.com).

Still note that some administrative tasks may be required concerning a comparison of the vocabulary (the metadata values themselves) used in existing profiles and those in the external resource descriptions.

5. Conclusions

We have proposed and implemented a generic framework for personalized information filtering, basing on a special RDFS/RDF representation model and a uniform algorithm computing a similarity measure for RDF instances.

The idea is that, for each new application, only a new ontology (and, if required, new local similarity measures) has to be defined.

The global similarity measure can be computed using a fixed uniform tool following our proposal.

However, there are still several aspects in which further details in our proposal have to be elaborated.

Efficiency/performance issues

Up to now we have not analyzed the efficiency of the proposed matching algorithm formally. A uniform approach however requires reliable statements concerning the conditions under which this algorithm works well. This should be one point of focus in future work.

Framework documentation

Even if a framework proves uniform for a class of applications, this does not immediately imply that it is uniformly usable in practice. Here a representation language is required which clearly explains the effect of all kinds of parameters in the ontology upon the resulting global similarity measure. In particular, if a designer of some new application has an intended similarity measure in mind, then it should be clear how to define an ontology appropriately. This is a further aspect which must definitely be addressed. Moreover, this aspect also includes the question of how the generic tools in our framework can be best embedded into a concrete application.

Context representation

As already indicated in Section 2.3, a suitable framework for context representation has to be defined; in particular when aiming at mobile applications this is of great importance.

Content acquisition and maintenance issues

Furthermore, the methods sketched in Section 4 have to be conceptualized in a way such that our generic framework is complemented by a toolkit easing content acquisition and update in concrete applications. We are currently developing a toolkit for merge functions allowing for profile update basing on relevance feedback. Adapting standard market basket analysis approaches to our framework will be one of the next steps.

Possible extensions

Because of its simple structure and the resulting efficiency benefits, we had decided to use RDFS/RDF for ontology modeling. However, one future task might be to extend our framework by designing and implementing a matching algorithm which defines a similarity measure for OWL instances.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Commununications*, 7:39–59, 1994.

[2] A. Bassoli and S. Baumann. BluetunA: music sharing through mobile phones. In *Proceedings of the Third International Workshop on Mobile Music Technology*, 2006.

[3] S. Baumann. Smartmobs and music: Ad-hoc socializing by portable music profiles. In *Inspirational Ideas at International Computer Music Conference (ICMC 2005)*, 2005.

[4] R. Bergmann. On the use of taxonomies for representing case features and local similarity measures. In L. Gierl and M. Lenz, editors, *Proceedings of the 6th German Workshop on CBR*. Universität Rostock, 1998. IMIB Series Vol. 7.

[5] R. Bergmann and A. Stahl. Similarity measures for object-oriented case representations. In B. Smyth and P. Cunningham, editors, *Advances in Case-Based Reasoning, 4th European Workshop, EWCBR-98, Dublin, Ireland, September 1998, Proceedings*, volume 1488 of *Lecture Notes in Computer Science*. Springer, 1998.

[6] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 89, 2001.

[7] Patrick Brezillon. Context in Artificial Intelligence: I. A survey of the literature. *Computer and AI*, 18(4):321–340, 1999.

[8] Patrick Brezillon. Context in Artificial Intelligence: II. Key elements of contexts. *Computer and AI*, 18(5):425–446, 1999.

[9] A. Dey, B. Kokinov, D. Leake, and R. Turner, editors. *Modeling and Using Context, 5th International and Interdisciplinary Conference, CONTEXT 2005, Paris, France, July 5-8, 2005, Proceedings*, volume 3554 of *Lecture Notes in Computer Science*. Springer, 2005.

[10] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR*, 163:845–848, 1965. English translation in Soviet Physics Doklady 10, 707–710, 1966.

[11] H. Liu, P. Maes, and G. Davenport. Unraveling the taste fabric of social networks. *International Journal on Semantic Web and Information Systems*, 2:42–71, 2006.

[12] F. Manola and E. Miller (Eds.). RDF Primer. W3C recommendation, W3C, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/, 2004.

[13] J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, 1971.

[14] T. Roth-Berghofer, S. Schulz, and D. Leake, editors. *Modeling and Retrieval of Context, Second International Workshop, MRC 2005, Edinburgh, UK, July 31 - August 1, 2005, Revised Selected Papers*, volume 3946 of *Lecture Notes in Computer Science*. Springer, 2006.

[15] Sven Schwarz. A context model for personal knowledge management. In Roth-Berghofer et al. [14].

[16] W3C. OWL web ontology language, overview. Technical report, 2004. D. McGuinness and F. van Harmelen (Eds.), W3C Recommendation, http://www.w3.org/TR/owl-features/.

[17] W3C. RDF vocabulary description language 1.0: RDF Schema. Technical report, 2004. D. Brickley, R. Guha, and B. McBride (Eds.), W3C Recommendation, http://www.w3.org/TR/rdf-schema/.

[18] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.