Data & Knowledge Engineering 79–80 (2012) 62–86

Contents lists available at SciVerse ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

A multi-layer framework for personalized social tag-based applications ☆

Barbara Carminati [a], Elena Ferrari [a], Andrea Perego [b,*]

[a] Dipartimento di Scienze Teoriche ed Applicate, Università degli Studi dell'Insubria, Varese, Italy
[b] European Commission — Joint Research Centre, Ispra, Italy

ARTICLE INFO

Article history:
Received 31 May 2010
Received in revised form 25 June 2012
Accepted 25 June 2012
Available online 4 July 2012

Keywords:
Web-based information systems
Social Web
Semantic Web
Social tagging
Personalization

ABSTRACT

Recent years have seen an increasing diffusion of online communities giving their members the ability of specifying and sharing metadata concerning online resources. Such practice, also known as *social* or *collaborative tagging*, has the purpose of collecting and sharing opinions about Web resources and simplifying their retrieval. In this paper, we go one step further and show how tags can have more enhanced applications to be exploited for customizing Web content fruition. More precisely, we propose a multi-layer framework where data collected by social tagging communities are complemented with additional services. Such services provide users the ability of expressing their dis/agreement with existing tags, denoting the members they trust based on their characteristics and relationships, or specifying policies on which "quality" assessment of resources should be returned. Besides providing the formal specification of the proposed framework, we illustrate two case studies we have implemented and the experiments we have carried out in order to verify the feasibility of our approach.

1. Introduction

In recent years, the widespread adoption of Web 2.0 related technologies has greatly facilitated user collaboration and knowledge sharing, bringing several benefits to the field of Web metadata generation and management. Notable examples of Web 2.0 technologies applied to Web metadata are those related to online communities, whose members have the ability of specifying and sharing metadata (referred to as *tags*). These are the cases, for example, of Delicious (http://delicious.com), RawSugar (http://rawsugar.com), Flickr (http://flickr.com), and Last.fm (http://last.fm). Such practice, also known as *social* or *collaborative tagging* [1,2], has the purpose of collecting and sharing opinions about Web resources, and simplifying resource retrieval by organizing them according to a tag-based browsing criterion.

The huge availability of social tagging systems has pushed the development of several applications exploiting these metadata, hereafter called *social tag-based applications*. Recommender systems [3] are notable examples of social-tag based applications. Here, users of online communities share resources that they consider relevant, and express personal opinions on them with the purpose of making resource retrieval easier.

In general, social tag-based applications gather metadata associated with resources, elaborate them and exploit the obtained results to trigger some actions (for example, resource recommendation, classification or filtering). Up to now, research on social tag-based applications has mainly focused on personalized recommendations of tags (e.g., [4–6]) or resources (e.g., [7–10]), which make use of techniques derived from the data mining area to predict which tags/resources might be relevant, and how much.

However, we believe that social tag-based applications can be further improved following several directions. First of all, existing social tag-based applications do not consider much the issue of metadata trustworthiness. As metadata evaluation might trigger crucial actions (for example, those related to resource filtering or access control), it is important to have an accurate estimation

☆ The work reported in this paper is partially funded by the European Community under the QUATRO Plus project (SIP-2006-211001) and by the Italian Ministry of University, Education and Research under the ANONIMO project (PRIN-2007F9437X_004).

* Corresponding author at: European Commission DG JRC, Via E. Fermi, 2749–TP 262, 21027 Ispra, Italy. Tel.: +39 0332786423; fax: +39 0332786325.
E-mail addresses: barbara.carminati@uninsubria.it (B. Carminati), elena.ferrari@uninsubria.it (E. Ferrari), andrea.perego@jrc.ec.europa.eu (A. Perego).

0169-023X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2012.06.002

of metadata trustworthiness, so to exclude untrustworthy metadata. As such, a *mechanism able to assess the trustworthiness of social metadata* is needed. More precisely, we think that collaborative environments and Semantic Web technologies can help. In fact, the availability of online communities consisting of thousands of users would help not only in increasing the number of labeled/tagged resources, but also in assessing their trustworthiness [11].

So far, this issue has been addressed by providing a measure/definition of trust based on some statistics on tags' frequency (see, e.g., [9]). As a naïve example, trustworthiness of a tag associated with a resource could be defined based on the percentage of tags providing identical descriptions for that resource. However, in designing mechanisms to assess social metadata trustworthiness, we think that it is fundamental to take into account also the *explicit* users' opinions on the metadata itself. Indeed, if users could express their *agreement/disagreement with the descriptions* provided by metadata, this would further help in determining metadata trustworthiness.

Further, we believe that several scenarios exist where taking into account user preferences during metadata selection could bring to more meaningful resource descriptions. This is inspired by our normal behaviors in real life. Let us consider, for example, the case of a person, say Kate, who is looking for some science-fiction books for a teenager. In real life, Kate might ask recommendations and suggestions only to those of her friends that are considered expert in science-fiction books. Also, Kate could restrict her selection only to recommendations of those experts that are teens. Applying the same approach in social tag-based applications implies *to make users able to denote who he/she considers trustworthy in describing resources*, that is, who are the users whose metadata have to be elaborated during resource evaluation. In contrast, existing social tag-based applications process resources by elaborating the whole set of metadata associated with it (see for instance [12]).

Further, the tags associated with a resource and their trust values can be used by users to decide how a given resource has to be processed, that is, they can be used to *personalize application behaviors according to user preferences*. For instance, a user might prefer that a recommender system recommends only those books whose metadata state that their content is related to science-fiction with trust value at least equal to 80%.

To cope with the above-discussed requirements, in this paper we propose a framework to support *personalized social tag-based applications based on trust policies and user preferences*. Trust policies allow one to identify trusted users – i.e., users whose metadata have to be considered – according to a variety of criteria (i.e., users' profiles, users' relationships, specific topics). In contrast, user preferences allow users to specify one or more conditions on resources' descriptors and corresponding trust values, and to state which "quality" assessment must be returned (e.g., "the resource is safe for children") and, possibly, which action has to be performed (e.g., recommend, filter, classify) in case at least one of the specified conditions is satisfied.

More precisely, we propose a *multi-layer framework*, where each layer is designed as a black box, providing basic services to the upper layers. The framework supports a *data layer*, gathering metadata from social tagging systems, a *rule layer*, that enforces trust policies and user preferences, and an *application layer*, which elaborates the metadata, filtered according to trust policies, by returning the notices and actions stated by user preferences. As it will be discussed in Section 2, the literature offers several proposals for the *data layer* and the *application layer*, but, to the best of our knowledge, nothing equivalent to our rule layer. More precisely, the main difference of our proposal with regard to existing personalization approaches is that they address a specific issue only, namely, tag/resource recommendation, whereas our framework is designed to be as flexible as possible with regard to the purposes social tags are used for.

The novelty of this framework is in the supported features and in its modular architecture, thanks to which it is possible to tailorize, or even disable, one or more components depending on the different contexts and requirements. We would like to note, however, that, although in this framework trust computation plays an important role, it is just one of the components of our framework. Also, it is not our purpose to propose a new trust system. Actually, our framework does not rely upon a specific method for trust computation, but it is designed to support different methods depending on the considered application scenario.

The work reported in this paper is an extension of [13], where we proposed a system for collaborative resource labeling and label rating, showing how this can be exploited for Web access personalization. In this paper, we significantly extend [13] in that we make it independent from which purpose and by what end user applications it is used. This is achieved by the introduction of the application layer in architecture proposed in [13], obtaining thus a multi-layer framework. With respect to [13], in this paper we have verified the feasibility of the multi-layer approach by designing and developing a prototype system implementing the proposed framework, and addressing a real world scenario. More precisely, we have tested the ability of our framework to (a) reuse datasets of existing Web-based communities, (b) enhance them by providing support to trust policies and user preferences, and (c) return information which can be exploited by end user applications for a variety of purposes. For this purpose, we have used the dataset provided by the Delicious online community (i.e., tags and users' relationships) to implement two distinct case studies, namely, personalized Web search and Web access personalization.

We would like to note that, in this paper, we focus on the technical feasibility of the framework we propose, in order to demonstrate that it is technically possible to enhance existing social media by providing personalization features currently not supported, and by exploiting the potential of user-generated content, as social tags are. Nonetheless, the paper includes a preliminary evaluation of the usability and effectiveness of our framework, which gave us an important feedback on the issues at stake, which we plan to address in future work.

The rest of the paper is organized as follows. Related work is discussed in Section 2. Section 3 provides an introduction to the proposed multi-layer framework. Data layer is described in Section 4,whereas Section 5 presents the rule layer. We prove the feasibility of the proposed framework by showing two case studies in Section 6. Section 7 concludes the paper and outlines future research directions. Finally, Appendix A contains the main algorithms underlying our framework.

2. Related work

The social tagging phenomenon and its advantages in terms of knowledge acquisition and enhancement of Web users' experience have been early recognized by scientists as a challenging research topic. Although work in this area spans over several issues, in this section we will discuss the ones that we think are most relevant to our approach.

A first research direction concerns the mechanisms that can be used to increase social tags' effectiveness. In fact, since tags are not terms from a predefined vocabulary, but keywords freely specified by end users, we may have different tags with the same meaning (synonymy), same tags with different meaning (homonymy), tags with multiple meanings (polysemy). This negatively affects the effectiveness of tag-based search, and leads to a low precision and recall of the obtained search results.

In order to address such issue, several solutions have been proposed, aiming at enforcing semantic interoperability (for the state of the art in this area, we refer the reader to [12]). In particular, some works have built on the notion of *emergent semantics*, i.e., the idea that, in widely distributed systems, semantics emerges from the agreement of the involved agents [14]. As an example, [15] proposes an approach to derive emergent semantics in a social bookmarking service by using statistical analysis. On the other side, [16] proposes to model a *folksonomy* – i.e., the set of social tags specified by users in an online community – as a tripartite graph derived from the information concerning users, tags, and bookmarked resources. Graph transformations are then used to derive lightweight ontologies of concepts and users.

The other main research direction focuses on using social tagging to enforce personalized recommendations. Basically, work carried out in this area is based on the adaptation to social tagging of techniques already established in recommender systems [3]. One of the addressed issues is *social tag prediction*, i.e., the possibility of predicting the set of tags applying to a not bookmarked resource. As an example, in [4] resources are classified based on their textual content, anchor text, and the set of websites the resources are linked from and to. Tags are then derived based on the similarity between the resource to be annotated and the set of bookmarked resources. By contrast, in [5], social tags are used to build user profiles, which are then exploited to cluster users with similar interests. Tag prediction is then carried out by suggesting to a given user the set of tags specified by other users with a similar profile, i.e., in the same cluster. An alternative approach is proposed in [6], where tags are predicted by extracting relevant terms from a resource, and then by computing their similarity with keywords representing the documents' corpus of the personal desktop of a given user. Other related works have been carried out in the context of Web services (see e.g., [17,18]).

User profiling is a technique which is used also in works aiming to recommend to a given user a set of bookmarked resources relevant for his/her interests. For instance, in [7] a set of algorithms is proposed, extending Information Retrieval techniques to build profiles and recommendations for Last.fm users. By contrast, [8] proposes to build user profiles by mining personal tags through formal concept analysis, then obtaining a hierarchy of user's interests. A different strategy is proposed in [9], where tags are hierarchically clustered by using measures like tag's frequency and distinctiveness. The system then computes how much a given cluster is relevant for the interests of a given user based on the tags he/she specified. As a last example we can cite [10], where, after having identified twelve tasks relevant to social tagging personalization, a probabilistic approach is adopted to model three of them, namely, collaborative tagging, browsing, and search.

The personalization approaches discussed above might be seen as an alternative solution to our framework, as far as the rule and application layers are concerned. However, they can be used to address a specific issue, namely, tag/resource recommendation, whereas our framework is designed to be as flexible as possible with regard to the purposes social tags are used for. From our perspective, the most relevant difference with our framework concerns how personalization is enforced. It is important to note that all these approaches have a common denominator in that they *statistically derive* new information from existing one, without requiring end users' opinion. This has great advantages in terms of usability, but has a major drawback in the probabilistic nature of the obtained results. As a consequence, it cannot be used for purposes where false positives might determine harmful consequences on the side of end users. For instance, they cannot be used to grant that a resource is safe for children, or that it satisfies the privacy requirements of an end user. For these purposes, the explicit specification of end users' preferences and requirements must be required, and this is exactly the issue addressed by the rule layer of framework.

For these reasons, we see such approaches as services which can be incorporated into the data layer, where they can play a crucial role in refining social tags. Semantic interoperability is actually a key issue which must be addressed in order to increase as much as possible the effectiveness of social tagging. Moreover, besides simplifying to end users the tag specification task, social tag prediction can help as well in building a common folksonomy. As far as personalized recommendation is concerned, this can be a service useful as alternative to the rule layer of our framework, depending on users' purposes. In fact, we remind that the application layer can interact with either the data or rule layers, thus deciding whether or not to use trust policies and user preferences.

The rule layer of our framework can be seen as an enhancement to social tagging services by the integration of some of the features provided by Semantic Web rule languages (e.g., [19]) and Semantic Web policy frameworks — an issue that, to the best of our knowledge, has not been addressed so far. Basically, Semantic Web policy frameworks are meant to support the specification and enforcement of policies expressed in terms of constraints on the machine understandable resource descriptions provided by Semantic Web languages. As such, they are one of the possible solutions to the issue of policy enforcement in highly distributed systems, having a huge number of agents, with heterogeneous and dynamically evolving characteristics. Examples of such frameworks are KAoS [20], REI [21], and Ponder [22,23], focusing mainly on access control, Protune [24,25], which provides support also to trust negotiation and privacy policies, and WIQA [26], which gives end users the ability of using filtering policies in order to denote given “quality” requirements that resources must satisfy.

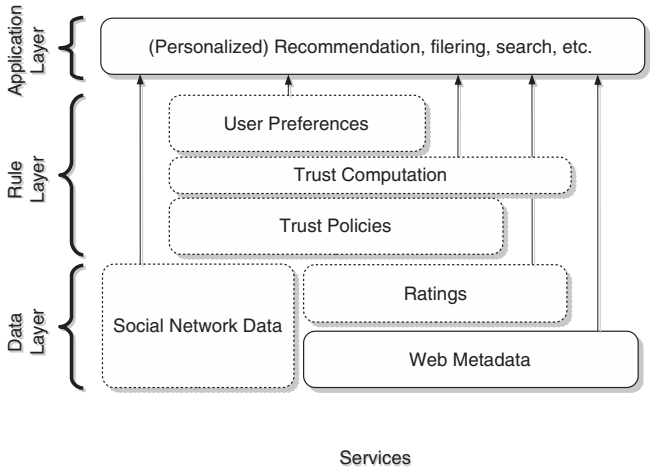

Fig. 1. Architectural components of the proposed framework. Services delimited by dotted lines are optional.

We would like to note, however, that our rule layer is not meant to be a new policy framework, specifically designed for social tagging services, and developed from scratch due to the fact that existing policy frameworks cannot be applied or adapted to such an application domain. By contrast, the rule layer just defines the minimum requirements to be satisfied in order to give end users the ability to denote trustworthy users and to express their preferences. Therefore, since we do not put any constraint on how the services supported by our rule layer are implemented, any framework/system can be a possible candidate, provided that the basic features of the rule layer are supported. As far as the policy frameworks mentioned above are concerned, we think that the best candidate might be the WIQA framework, since it is meant to be used by end users in order to filter Web resources which do not satisfy their requirements.

3. A multi-layer framework for personalized social tag-based applications

As represented in Fig. 1, the proposed framework consists of three main layers. The *data layer* groups all services in charge of managing personal data of online communities' members and services generating the metadata. To make the framework able to cooperate with different Web metadata services, we adopt the specification of the POWDER (Protocol for Web Description Resources) W3C Working Group,[1] as interchange format for Web metadata [27]. POWDER can be used to associate any type of description with a group of resources, and, additionally, to provide meta information about such descriptors (such as, who has specified them, when have they been issued, what is their validity period). As it will be discussed in Section 4.2, the proposed multi-layer framework makes use of the notion of *descriptor*, instead of the one of tag, to denote a keyword, or attribute–value pair, denoting a specific resource's characteristic. Moreover, we use the notion of *label* to denote a set of descriptors applying to a given resource and specified by a given author.

In the proposed framework, besides labeling resources, social network members can express their dis/agreement about existing labels, by specifying ratings on the contained descriptors. Therefore, the *data layer* also includes a *rating service* (see Fig. 1). All data collected by services in the *data layer* are provided to the upper layers, that is, the *rule layer* and the *application layer*. The first provides the services to support trust policies and user preferences, that is, to enforce rules for determining, respectively, the trustworthiness of Web metadata and the quality of Web resources. Although there currently exist several languages which can be used to enforce trust policies and user preferences (see e.g., Protune [24] and WIQA [26]), in this paper we investigate Semantic Web technologies as the basis of the standard interchange format among our layers. More precisely, we use RDF/OWL [28,29] to encode the data layer's components, whereas trust policies and user preferences are represented and enforced through N3Logic rules [30]. We have adopted N3Logic, although it is not a standard Semantic Web language, not only because of its support for rules, but also because it supports an effective mechanisms, i.e., *quoted formulae*, to express *statements about statements* — a feature that we need, for example, to specify the author of a label. The alternative would have been the technologies designed by the Rule Interchange Format (RIF) of the W3C (http://www.w3.org/2005/rules). However, such technologies have reached only recently the status of a Semantic Web standard, and they are not fully implemented.

On the top of the proposed framework there is the *application layer*. This encloses those services (typically, user agents) that elaborate the output of the data or rule layer, in order to address specific purposes (e.g., resource classifications, filtering). As depicted in Fig. 1, the output is previously elaborated so as to compute the aggregate trust values and quality measures of Web resources according to the specified trust policies and user preferences and the available ratings and Web metadata. This computation is provided by the *trust computation* service in the *rule layer*. The application layer will also provide end users an abstract view of our framework, hiding the complexity of its architecture. This is an important issue, as far as usability is concerned, which is however out the scope of this paper, where we focus on system design, implementation, and evaluation.

[1] Working Group page: http://www.w3.org/2007/powder.

The proposed multi-layer framework has the main advantage of being modular and therefore can complement existing services without the need of their re-design. The use of Semantic Web technologies as the interchange format among layers makes such task easier. Indeed, as reported in Fig. 1, with the exception of the Web metadata service, all the other services in the data and rule layers are optional. For instance, a typical social tagging service supports just the Web metadata service and, possibly, the one concerning social networks. Ratings, trust policies, and user preference services can then be added to enhance its features by exploiting information already stored by the service.

In what follows, for the sake of brevity, we will refer to our framework using the acronym WPF (Web Personalization Framework).

4. Data layer

In this section, we illustrate the first component of our framework, i.e., the *data layer*, which consists of the services in charge of managing social network data, Web metadata, and ratings.

4.1. Social network data

Each WPF user is associated with a *profile*, corresponding to a set of personal data (such as first and last name, email address, personal interests, nationality, job, expertise), encoded by one or more *credentials*. Besides the certified properties, a credential contains the IDs of the Certification Authority (CA) issuing it, and of the member whom the credential refers to. Finally, the credential is signed by the CA releasing it (see Example 1). Note that, there exist some attributes (e.g., those describing the interests of a given user) which do not need to be certified. However, for simplicity, in this paper, we assume that all the credential attributes are certified.

Example 1. The following are examples of credentials released by CAs CA_1 and CA_2: $C_1 = (CA_1, \text{Betsy}, \{\text{profession} = \text{teacher}\}) \,||\, DSig_1$; $C_2 = (CA_2, \text{Henry}, \{\text{profession} = \text{physician}\}) \,||\, DSig_2$, where $DSig_i$, $1 \le i \le 2$, denotes the digital signature of credential C_i.

In order to support an exchange format for users' credentials, their formal representation, reported by Example 1, is converted into FOAF profiles [31]. FOAF (Friend of a Friend) is a widely used Semantic Web technology that allows the specification of personal information.

In addition, we need to model the relationships established by WPF users in the community with the purpose of specifying and sharing tags. Please note that relationships play a key role in our framework as they are used in trust policies to identify trustworthy users, that is, users whose labels/ratings should be considered for trust computation. Indeed, according to the proposed model such users can be specified by putting constraints on a user social graph (e.g., only my friends and the friends of my friends are considered trustworthy).

Formally, we model such social network $\mathcal{SN}$ as a tuple $\left(M_{\mathcal{SN}}, E_{\mathcal{SN}}, RT_{\mathcal{SN}}, \phi_{E_{\mathcal{SN}}}\right)$, where $M_{\mathcal{SN}}$ and $E_{\mathcal{SN}}$ are, respectively, the nodes and edges of a directed graph $(M_{\mathcal{SN}}, E_{\mathcal{SN}})$, $RT_{\mathcal{SN}}$ is the set of supported relationship types, whereas $\phi_{E_{\mathcal{SN}}} : E_{\mathcal{SN}} \rightarrow RT_{\mathcal{SN}}$ is a function assigning to each edge $e \in E_{\mathcal{SN}}$ a relationship of type $rt \in RT_{\mathcal{SN}}$.

We say that two WPF users participate in a relationship of a given type *rt*, if there exists a path connecting them consisting only of edges labeled with relationship type *rt*. The length of the path is the *depth d* of the corresponding relationship. If $d = 1$, the relationship is *direct*; if $d > 1$, we say that the relationship is *indirect*. To model such notion of relationship, we use the REL-X ontology,[2] which defines an OWL class (*relx:Relationship*), and properties denoting the members (*relx:hasMember*), type (*relx:type*), and depth (*relx:depth*) of a relationship.[3]

Example 2. Fig. 2 depicts a small portion of a social network. In the figure, the arrows at both ends of an edge are a shortcut to denote mutual relationships, i.e., the existence of two edges between the same pair of nodes, associated with the same label, and having opposite directions. More precisely, in Fig. 2 there exist direct relationships of type *friendOf* between Kate (*K*) and Phil (*P*), Kate and Betsy (*B*), Phil and Henry (*H*), and of type *colleagueOf* between Kate and Betsy, Phil and Betsy, and Phil and Henry. Moreover, there exist four indirect relationships of depth 2: one of type *friendOf* between Kate and Henry (corresponding to path *KPH*) and Phil and Betsy (corresponding to path *PKB*); two of type *collegueOf* between Kate and Phil (corresponding to path *KBP*) and Betsy and Henry (corresponding to path *BPH*). Finally, there exist two indirect relationships of depth 3: one of type *friendOf* between Betsy and Henry (*BKPH*) and the other of type *colleagueOf* between Kate and Henry (*KBPH*).

It is interesting to note that up to now, existing social networks do not provide such functionality and as such our framework cannot be directly applied on them. However, thanks to FOAF and REL-X ontologies, profile and relationship information can be exported from existing social networks so as to be used in WPF. As an example, the framework could be integrated with Facebook by developing a third-party Facebook application for those WPF users having a Facebook account. By means of this application, all

[2] Namespace URI: http://www.dicom.uninsubria.it/dawsec/vocs/relx.

[3] There exist other ontologies, like the one described in [32], which can be used for the same purpose. However, different from them, the REL-X ontology has the advantage of allowing one to associate with a relationship not only a type, but also other information (in our case, its depth), without the need of making use of RDF reification, which would make relationship specifications unnecessarily complex [33].

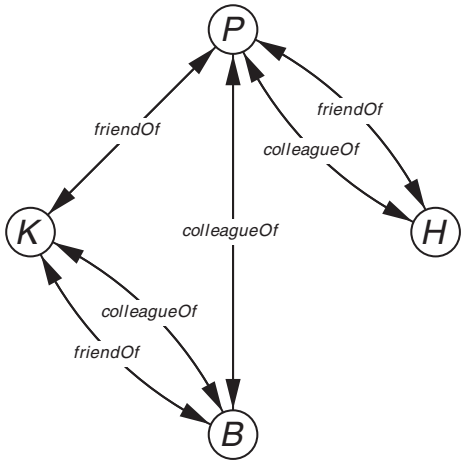

Fig. 2. A small portion of a social network.

the relevant information (i.e., profiles, relationships) could be easily extracted and used to generate the WPF social network. The Facebook application could also be used to gather from WPF users their ratings and labels.

4.2. Web metadata

By abstracting from the various representations, Web metadata can be modeled as *labels*, which in turn consist of a set of descriptors. Labels describe the content and/or characteristics of a (set of) resource and can be specified by resource owners, or by WPF users. They are identified by a URI, and contain a set of *resource descriptors* $rd_1,\ldots,rd_n$, which may be of two different types, namely, *resource property descriptors* and *resource content descriptors*. Resource property descriptors are used to model specific characteristics of the resource (such as the author's name, its title, the language used). They are modeled as pairs $pn=pv$, where pn denotes the name of a resource property and pv denotes the value of pn. In contrast, resource content descriptors are used to convey different information on resource content. This can be a description of the resource content itself, as well as represent opinions on it. They are expressed as pairs $t=\rho$, where $\rho\in[0,1]$ denotes the relevance of t in describing the considered resource. The set of resources to which a label refers to is denoted by a *URI pattern*, by which it is possible to express statements like "all the resources hosted by www.example.org, where the URI path component starts with `foo`".[4] Besides resource descriptors, a label contains the ID of the WPF user who created it, a timestamp, and, optionally, the validity period for the label.

Example 3. In the running example we will use to illustrate our framework, we suppose that Kate runs a blog – http://about-gmos.net – about Genetically Modified Organisms (GMOs), where she publishes her personal posts as well as contributions from experts in the field. Table 1 presents examples of resource labels, where, for simplicity, the timestamp and validity period have been omitted. Moreover, we denote by LB_n the URI of label n. Labels LB_1 and LB_3 describe all the resources hosted by about-gmos.net. Label LB_1 has been specified by Kate, and it states that she is the author of the resources hosted by about-gmos.net, that the used language is English, and that topic *gmos* has a relevance equal to 80% in describing their content. Label LB_3 has been specified by Phil, and it states that topic *biology* has a relevance equal to 40% for the description of the resources content. In contrast, labels LB_2 and LB_4 describe all the resources having a URI starting with http://about-gmos.net/children. Label LB_2, specified by Kate, states that such resources are authored by Henry, that their title is "GMOs and Child Nutrition", and that the topic *childCare* has a relevance equal to 100%, whereas topic *nutrition* has a relevance equal to 60%. Finally, label LB_4 is specified by Phil, and it states that the corresponding resources are authored by Kate, and topic *health* has a relevance equal to 20%, whereas topic *nutrition* has a relevance equal to 40.

As mentioned in Section 3, we adopt POWDER as the interchange format for Web metadata [27], to make easier processing and aggregation of metadata originating from heterogeneous sources. In what follows, we therefore use the traditional term of *label* to denote a set of descriptors encoded according to the POWDER syntax.

Example 4. Fig. 3 shows the RDF/OWL encoding of LB_1 in Table 1 according to POWDER specifications [34] and using the N3 syntax.[5] The ontology header at line 8 encodes the information about who issued the label, when it has been issued, and its

[4] URI patterns are specified by using a simplified regular expression syntax, where the wildcard (*) matches a string of $0,\ldots,n$ URI characters.

[5] Describing the POWDER format would require a discussion which is out of the scope of this paper. We refer the interested reader to [27,35,34]. For a better understanding of our example, suffice it to say that an RDF/OWL *POWDER document* – i.e., a label, according to our terminology – is an OWL ontology including a class description I, denoting the set of resources having a URI/IRI matching a given pattern, a set $D_1,\ldots,D_n$ of class descriptions denoting the set of resources having given characteristics, and a statement S denoting class I as a subclass of $D_1,\ldots,D_n$. Finally, the ontology header is used to denote the author, issue date, and validity period of the POWDER document.

Table 1
Examples of labels. Labels' issue date and validity period have been omitted in the table due to space constraints.

Example 3				
URI	Author	URI Pattern	Property descriptors	Content descriptors
LB_1	Kate	http://about-gmos.net*	*author* = Kate, *lang* = en	*gmos* = 0.8
LB_2	Kate	http://about-gmos.net/children*	*author* = Henry, *title* = GMOs and Child Nutrition	*childCare* = 1.0, *nutrition* = 0.6
LB_3	Phil	http://about-gmos.net*	∅	*biology* = 0.4
LB_4	Phil	http://about-gmos.net/children*	*author* = Kate	*health* = 0.2, *nutrition* = 0.4

validity period (we suppose here that LB_1 has been issued on 12 December 2008, and that it is valid for a whole year after that date). By contrast, the class description at line 10 denotes all the resources having a URI starting with http://about-gmos.net, whereas the class descriptions at lines 12–14 denote the resources having Kate as author (line 12), those written in English (line 13), and those where the relevance of topic *gmos* is equal to 80% (line 14). Finally, line 16 implements the rule according to which all the resources having a URI starting with http://about-gmos.net are (a subset of those) authored by Kate, written in English, and have a content where the relevance of topic *gmos* is equal to 80%. This is expressed by stating that class *Iriset* is a subclass of classes *D1*, *D2*, and *D3*.

4.3. Ratings

To make their evaluation easier, ratings applying to the same label are grouped into a *label rating*. The structure of a label rating is similar to the one of a label in that it contains ratings for property descriptors ($pn = pv$) and content descriptors ($t = \rho$), both modeled as pairs ($pn = pv$, rating) and ($t = \rho$, rating), respectively. The difference is that label rating URI pattern always corresponds to a single resource – i.e., the label being rated – , and no validity period is specified since a rating is valid only during the lifetime of the corresponding label.

Example 5. Table 2 reports examples of ratings for the labels in Table 1, where the timestamp component of each rating has been omitted for brevity. As in Example 3, we assume that $LB_1,\ldots,LB_4$ correspond to the URIs of the labels in Table 1. Ratings RT_1 and RT_2 have been specified by Betsy and they apply to labels LB_3 and LB_4, respectively. In RT_1, Betsy agrees that the topic *biology* has a relevance equal to 40% – i.e., (*biology* = 0.4,1) – for the description of the corresponding resources, whereas in RT_2 she disagrees on the fact that the topic *health* has a relevance equal to 20% – i.e., (*health* = 0.2,0) – , but she agrees that the topic *nutrition* has a relevance equal to 40% — i.e., (*nutrition* = 0.4,1). Rating RT_3 applies to LB_1: it has been specified by Henry, who agrees that the topic *gmos* has a relevance equal to 80% — i.e., (*gmos* = 0.8,1). Henry has specified also ratings RT_4 and RT_5 concerning labels LB_2 and LB_4, respectively. In RT_4, he disagrees on the fact that he has been claimed to be the author of the labeled resources, and that the relevance of topic *nutrition* is equal to 60% – i.e., (*author* = Henry,0) and (*nutrition* = 0.6,0) – , but he agrees that topic *childCare* has a relevance equal to 100% — i.e., (*childCare* = 1.0,1). In contrast, in RT_5, Henry agrees that Kate is the author of the labeled resources, and on the fact that topic *nutrition* has a relevance equal to 40% — i.e., (*author* = *Kate*,1) and (*nutrition* = 0.4,1). Finally, rating RT_6 is specified by Phil on label LB_1, and it expresses Phil's agreement about the descriptors stating that Kate is the resource's author – i.e., (*author* = *Kate*,1) – , and that the topic *gmos* has a relevance equal to 80% — i.e., (*gmos* = 0.8,1).

Example 6. Fig. 4 shows the N3 encoding of RT_6 in Table 2. In order to associate a rating with the statements in label LB_1, they are enclosed into *quoted formulae*,[6] and then the rating (*voc:rating*)[7] is specified on them. Thus, lines 12–15 specify the rating about the statement according to which Kate is the author of the resources having a URI starting with http://about-gmos.net, whereas lines 16–19 specify the rating about the relevance of topic *gmos* for the same set of resources. Quoted formulae are then used also to specify when the ratings have been issued (line 20), and who issued them (line 21).

5. Rule layer

In this section, we illustrate the main components of the rule layer, that is, trust policies and user preferences, describing also how they can be enforced. It is important to recall that trust policies and user preferences play two different roles. As such they can be seen as two complementary yet orthogonal components of the rule layer. Trust policies are used to select only some labels/ratings to be considered for descriptor trust computation, by leveraging trust relationships among WPF users. In contrast, user preferences are more similar to traditional access control policies, although they broaden their scope, in that they allow a user to

[6] In N3, quoted formulae are statements delimited by curly brackets, used to represent multiple, and possibly nested, RDF graphs into the same document. As already mentioned in Section 3, this allows one to specify "statements describing other statements", thus providing an effective alternative to RDF reification, which would dramatically increase the complexity of rating specification.

[7] Here and in the remainder of the paper, we use the *voc* namespace prefix for properties and classes needed to model the notions of our approach.

```
@prefix      rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix       owl: <http://www.w3.org/2002/07/owl#> .
@prefix     wdrs: <http://www.w3.org/2007/05/powder-s#> .
@prefix property: <http://www.example.com/property#> .
@prefix      base: <http://mynet.net/labels/lb1#> .
@prefix         : <http://mynet.net/members/> .

base: a owl:Ontology; wdrs:issuedby :Kate; wdrs:issued "2008-02-12"; wdrs:validFrom
    "2008-02-12"; wdrs:validUntil "2009-02-12" .

_:Iriset a owl:Class; rdfs:subClassOf [ a owl:Restriction; owl:onProperty
    wdrs:matchesregex; owl:hasValue "http:\/\/about\-gmos\.net.*" ] .

_:D1 a owl:Class; rdfs:subClassOf [ a owl:Restriction; owl:onProperty
    property:author; owl:hasValue :Kate ] .
_:D2 a owl:Class; rdfs:subClassOf [ a owl:Restriction; owl:onProperty property:lang;
    owl:hasValue "en" ] .
_:D3 a owl:Class; rdfs:subClassOf [ a owl:Restriction; owl:onProperty property:gmos;
    owl:hasValue "0.8" ] .

_:Iriset rdfs:subClassOf _:D1, _:D2, _:D3 .
```

Fig. 3. OWL-encoding of LB_1 in Table 1.

state which quality assessments should be associated with a resource satisfying them (e.g., safe for children, relevant) on the basis of several different dimensions (e.g., associated descriptors and/or their trust values). Therefore they are a means to associate a sort of user-defined quality label to resources. Moreover, they allow to associate different actions to be performed when the corresponding resource is accessed, based on the associated label.

5.1. Trust policies

Labels associated with a given resource, and the corresponding ratings, are used to determine the trustworthiness of the corresponding descriptors, according to a trust computation algorithm that may vary depending on the considered application scenario. Independent from the adopted trust computation method, it may be often the case that WPF users want to base trust computation only on selected labels/ratings, specified by authors they consider trustworthy. For instance, they may trust only the labels/ratings created by their friends or, depending on a specific topic (e.g., sport, books, movies) and/or specific resources, a WPF user can consider relevant the opinions of some members (e.g., because they are expert in the field) and not relevant those of others. It is then important that WPF users are provided the possibility of specifying which labels/ratings should be taken into account when computing descriptors' trust values.

In our approach, such preferences can be expressed through *trust policies*. A trust policy specifies which WPF users are considered trustworthy in labeling/rating a given set of resources. Therefore, while evaluating descriptors' trustworthiness, only the descriptors and/or ratings authored by them are taken into account. Descriptors/ratings' authors can be identified either by their IDs, or by specifying a set of constraints concerning their credentials and/or the relationships they participate in. Moreover, it is possible to specify that a user is trustworthy only with regard to a given set of topics and/or resource properties. More precisely, the notion of trust policy is formally defined as follows, where $\mathcal{AN}$, $\mathcal{T}$, and $\mathcal{PN}$ denote, respectively, the set of attribute names, topics, and resource property names in the system.

Definition 1. A trust policy TP is a tuple (*author*, *uriPattern*, *trustedM*, *T*, *PN*), where:

- *author*$\in M_{\mathcal{SN}}$ is the WPF user who specifies the policy;
- *uriPattern* is a URI pattern denoting the set of resources TP applies to;
- *trustedM* denotes the set of WPF users whose opinion is considered trustworthy with regard to resources denoted by *uriPattern*. It can have one of the following forms, possibly combined:
  1. a set *M* $M_{\mathcal{SN}}$ of WPF users;
  2. a set of attribute constraints of the form *an* OP *av*, where *an*$\in\mathcal{AN}$ is the name of an attribute in a user credential, *av* is an attribute value, whereas OP is a comparison operator compatible with *an*'s domain; the semantics of a set *AC* of attribute constraints is equivalent to a conjunction;
  3. a set of relationship constraints of the form (m,rt,max d), denoting all the WPF users participating with member $m\in M_{\mathcal{SN}}$ in a relationship of type $rt\in RT_{\mathcal{SN}}$, having a depth $d\leq$max d; the semantics of a set *RC* of relationship constraints is equivalent to a conjunction;
- $T\subseteq\mathcal{T}$ is a set of topics, possibly empty, for which the opinions of the WPF users denoted by *trustedM* are considered trustworthy in describing the resources denoted by *uriPattern*; in case $T=\emptyset$, the opinions of the WPF users denoted by *trustedM* are considered trustworthy for any topic;

Table 2
Examples of label ratings.

ID	Author	Label's URI	Ratings on property descriptors	Ratings on content descriptors
RT_1	Betsy	LB_3	∅	(*biology* = 0.4,1)
RT_2	Betsy	LB_4	∅	(*health* = 0.2, 0), (*nutrition* = 0.4, 1)
RT_3	Henry	LB_1	∅	(*gmos* = 0.8, 1)
RT_4	Henry	LB_2	(*author* = Henry, 0)	(*childCare* = 1.0, 1), (*nutrition* = 0.6, 0)
RT_5	Henry	LB_4	(*author* = Kate, 1)	(*nutrition* = 0.4, 1)
RT_6	Phil	LB_1	(*author* = Kate, 1)	(*gmos* = 0.8, 1)

- $PN \subseteq \mathcal{PN}$ is a set of property names, possibly empty, for which the opinions of the WPF users denoted by *trustedM* are considered trustworthy in describing the resources denoted by *uriPattern*; in case $PN = \emptyset$, the opinions of the WPF users denoted by *trustedM* are considered trustworthy for any property name.

Example 7. Table 3 reports examples of trust policies, all applying to the resources hosted by about-gmos.net. TP_1 states that Betsy considers trustworthy for topic 'nutrition' and for any resource property only those WPF users who are experts in dietology — i.e., Phil, according to Example 1. TP_2 specifies that Henry considers trustworthy for any topic and resource property only the WPF users who are, at the same time, Henry's direct friends, and Henry's colleagues, with a maximum depth equal to 2 (i.e., only Phil, according to Fig. 2). TP_3 states that Kate considers trustworthy her direct friends for any topic and for resource property *author*. Finally, TP_4 states that Phil considers Kate trustworthy for topic *gmos* and for resource property *author*.

Suppose now that each WPF user *m* in Fig. 2 has specified, in addition to the trust policies in Table 3, a trust policy of the form $(m,*,m,\emptyset,\emptyset)$, stating that *m* considers him/herself trustworthy for any resource, topic, and resource property.

When a user requires to access a resource *rsc* having a URI matching http://about-gmos.net*, the system verifies which descriptors and ratings match the policies (see Table 4). Then, it computes the trust of the descriptors concerning resource *rsc* by taking into account only the matching descriptors and ratings.

Example 8. As an example of how Semantic Web technologies can be used to represent trust policies, Fig. 5 shows the encoding of TP_3 in Table 3 into an N3 rule. More precisely, lines 14–16 correspond to the antecedent of the rule, stating the constraints on the URI pattern (line 14), trustworthy members (line 15), and property/content descriptors (line 16). If such constraints are satisfied, then a label or rating is marked as trustworthy (line 18). Finally, line 20 states that the author of such trust policy is Kate.

Trust policies may be of two different types, depending on the member who specifies them. More precisely, given a trust policy $TP = (m, uriPattern, trusteM, T, PN)$, if *m* is the owner of the resource(s) denoted by *uriPattern*, we call TP an *owner-defined trust policy* (denoted as OTP); otherwise, TP is called *user-defined trust policy* (denoted as UTP). This distinction is introduced because somehow the owner of a resource would like to state which are the users he/she considers trustworthy with respect to the description of his/her resources (e.g., because they have the required expertise), and such policies may be taken into account by other users when

```
1  @prefix      rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
2  @prefix       owl: <http://www.w3.org/2002/07/owl#> .
3  @prefix      foaf: <http://xmlns.com/foaf/0.1/> .
4  @prefix  dcterms: <http://purl.org/dc/terms/> .
5  @prefix      wdrs: <http://www.w3.org/2007/05/powder-s#> .
6  @prefix  property: <http://www.example.com/property#> .
7  @prefix       voc: <http://mynet.net/voc#> .
8  @prefix          : <http://mynet.net/members/> .

10 {
11   {
12     {[] a owl:Class; rdfs:subClassOf
13       [ a owl:Restriction; owl:onProperty wdrs:matchesregex; owl:hasValue
             "http:\/\/about\-gmos\.net.*" ],
14       [ a owl:Restriction; owl:onProperty property:author; owl:hasValue :Kate ]
15     } voc:rating "1" .
16     {[] a owl:Class; rdfs:subClassOf
17       [ a owl:Restriction; owl:onProperty wdrs:matchesregex; owl:hasValue
             "http:\/\/about\-gmos\.net.*" ],
18       [ a owl:Restriction; owl:onProperty property:gmos; owl:hasValue "0.8" ]
19     } voc:rating "1" .
20   } dcterms:issued "2008-07-11" .
21 } wdrs:issuedby :Phil .
```

Fig. 4. OWL-encoding of RT_6 in Table 2.

Table 3
Examples of trust policies concerning resources having a URI matching pattern http://about-gmos.net*.

ID	Author	URI Pattern	Trustworthy members	Topics	Properties
TP_1	Betsy	http://about-gmos.net*	Expertise = dietology	*nutrition*	∅
TP_2	Henry	http://about-gmos.net*	(Henry, *friendOf*, 1), (Henry, *colleagueOf*, 2)	∅	∅
TP_3	Kate	http://about-gmos.net*	(Kate, *friendOf*, 1)	∅	*author*
TP_4	Phil	http://about-gmos.net*	Kate	*gmos*	*author*

they want to specify their own trust policies or when computing descriptors' trust values. As an example, suppose that a WPF user owns a Web site dealing with medicine. As resource owner, he/she may trust only experts to specify labels for such Web site, and/or rate the associated labels. How owner-defined policies are enforced depends on the application domain. For instance, they can be used to prevent inexpert members from specifying labels and ratings. Alternatively, WPF users might be allowed to freely create labels/ratings, but to use owner-defined trust policies in combination with user-defined trust policies, when evaluating descriptors' trustworthiness. In our approach, to be as flexible as possible, we allow WPF users to specify through their user preferences whether and how user- and owner-defined trust policies should be taken into account (see Definition 2).

5.2. User preferences

Labels and ratings give end users the ability of being aware of the content/characteristics of the resources they access, whereas trust policies allow users to state which descriptors, and associated ratings, must be considered to compute descriptors' scores. Once returned to an end user, such information can be exploited by him/her as a basis to assess the "quality" of a resource against his/her personal opinions and beliefs. For instance, if a resource is associated with a set of descriptors stating that it provides accurate and reliable medical information, and which have a trust value greater than 80%, an end user might decide that such resource is "safe," and he/she can trust the provided medical information. In general, the assessed quality of a resource depends on two main factors (a) the descriptors associated with it and their associated trust values and (b) for which purpose(s) an end user is accessing such resource.

We can therefore encode the criteria adopted by an end user to decide the quality of a resource to automate this process. More precisely, in our framework, the automatic assessment of resources' quality is achieved by allowing a user to specify a further type of policy, referred to as *user preferences*. User preferences state the conditions to be satisfied in order to return a given quality assessment for a set of resources. These conditions are expressed as constraints on the associated descriptors and their trust values. Moreover, user preferences give the possibility of stating whether user- and/or owner-defined trust policies specified for the requested resource must be taken into account, and how they must be combined. More precisely, a user preference allows one to state whether only user-defined or only owner-defined policies, or both/neither of them must be considered to select the labels and ratings used to evaluate descriptors' trustworthiness. If both user- and owner-defined trust policies must be used, it is also possible to specify whether the descriptors and ratings denoted by user- and owner-defined trust policies must be combined by using a union or intersection operator. Finally, we give the end user the possibility of deciding whether all or only some of the owner-defined trust policies must be taken into account.

As we have mentioned above, the notion of "quality" may vary depending on the end user's purpose(s), which might be quite heterogeneous. In other words, this is an issue to be addressed by the application layer. Consequently, quality assessment at the rule layer is expressed by a set of not predefined, system specific, statements denoting whether a given resource can be used for given purposes. Finally, a user preference includes a set of options that are then used by the user agent to perform specific tasks. For instance, considering a parental control scenario, such options might state whether access to a resource satisfying a given user preference must be blocked or not.

The notion of user preference is therefore formally defined as follows.

Definition 2. A user preference UP is a tuple (*author*, *uriPattern*, *PC*, *CC*, *settings*, *Q*, *options*), where:

- $author \in M_{\mathcal{SN}}$ is the WPF user who specifies UP;
- *uriPattern* is a URI pattern, denoting the set of resources UP applies to;
- *PC* is a set of triples (*pc*,*tc*,*dc*), where:
  - *pc* is a property constraint of the form *pn* OP *pv*, where *pn* is a property name, *pv* is a property value, whereas OP is a comparison operator compatible with *pn*'s domain;

Table 4
Labels and ratings in Tables 1 and 2 satisfying (Y) or not satisfying (N) the trust policies in Table 3.

	LB_1	LB_2	LB_3	LB_4	RT_1	RT_2	RT_3	RT_4	RT_5	RT_6
Betsy	N	N	N	Y	Y	Y	N	N	N	Y
Henry	N	N	Y	Y	N	N	Y	Y	Y	Y
Kate	Y	Y	Y	Y	Y	Y	N	N	N	Y
Phil	Y	N	Y	Y	N	N	N	N	N	Y

```
@prefix       voc: <http://mynet.net/voc/#> .                                          1
@prefix      relx: <http://www.dicom.uninsubria.it/dawsec/vocs/relx#> .                2
@prefix       owl: <http://xmlns.com/foaf/0.1/> .                                     3
@prefix      foaf: <http://xmlns.com/foaf/0.1/> .                                     4
@prefix       log: <http://www.w3.org/2000/10/swap/log#> .                            5
@prefix    string: <http://www.w3.org/2000/10/swap/string#> .                         6
@prefix      math: <http://www.w3.org/2000/10/swap/math#> .                           7
@prefix      wdrs: <http://www.w3.org/2007/05/powder-s#> .                            8
@prefix  property: <http://www.example.com/property#> .                               9
@prefix         : <http://mynet.net/members/> .                                       10

{                                                                                     12
  {                                                                                   13
    ?Resource a foaf:Document; log:uri [ string:startsWith "http://about-gmos.net" ] . 14
    ?Relationship a relx:Relationship; relx:hasMember ?Author, :Kate; relx:type       15
        relx:FriendOf; relx:depth [ math:notGreaterThan "1"] .
    ?LabelOrRating a [ owl:unionOf ( voc:Label voc:Rating ) ]; wdrs:issuedby ?Author; 16
        log:includes {[] a owl:Restriction; owl:onProperty property:author} .
  } log:implies {                                                                     17
    ?LabelOrRating voc:isTrustworthy "true"                                           18
  }                                                                                   19
} wdrs:issuedby :Kate .                                                               20
```

Fig. 5. N3-encoding of TP_3 in Table 3.

- *tc* is a trust constraint of the form tv OP τ, where $\tau \in [0,1]$ is a trust value, and OP$\in\{=,<,>,\leq,\geq\}$;
- *dc* is a distribution constraint of the form dv OP δ, where $\delta \in [0,1]$ denotes the required percentage of descriptors satisfying *pc*, and OP$\in\{=,<,>,\leq,\geq\}$;

- *CC* is a set of pairs (*cc*,*tc*), where *cc* is a resource content constraint of the form t OP ρ, where *t* is a topic, $\rho \in [0,1]$ denotes the relevance of topic *t*, OP$\in\{=,<,>,\leq,\geq\}$, whereas *tc*is a trust constraint on *cc*;
- *settings* is a triple (*checkUTP*, *checkOTP*, *comb*), where:
  - *checkUTP*$\in\{$`all`, `none`$\}$ denotes whether user-defined trust policies must (all) or must not (none) be taken into account;
  - *checkOTP*$\in\{$`all`, `some`, `none`$\}$ denotes whether owner-defined trust policies must (all), must not (none) be taken into account, or if only those selected by the end user (some) must be considered;
  - *comb*$\in\{\cap,\cup\}$ denotes whether the sets of descriptors and ratings denoted by the user- and owner-defined trust policies must be combined by using the union ($\cap$) or intersection ($\cup$) operator; in case *checkUTP* and/or *checkOTP* are set to `none`, *comb* is set to $\cup$, by default.
- *Q* is a set of attribute–value pairs assessing the quality of the resources satisfying UP;
- *options* is a set of attribute–value pairs, possibly empty, denoting the set of actions to be performed by a user agent when accessing a resource satisfying UP.

It is important to note that the *Q* and *options* components of user preferences should be initialized with values from vocabularies defined according to the reference application scenario. For instance, if the user agent aims at personalizing access to Web resources, possible attribute–value pairs for *options* is 'block = yes' or 'block = no', whereas pairs for *Q* could be 'safe-info' and 'relevance' as attributes and 'yes/no' as possible values. However, to make meaningful examples, hereafter all examples will refer to user preferences defined for the Web resource personalization application scenario.

Example 9. Table 5 reports examples of user preferences, all applying to the resources hosted by about-gmos.net. Preference UP_1, specified by Henry, denotes the resources hosted by about-gmos.net as carrying information which can be safely used (i.e., *safe-info* = *yes*), if (a) at least 50% ($dv \geq 0.5$) of the associated descriptors concerning property *author*, and having a trust value greater than 50% ($tv > 0.5$), have a value equal to *Kate* (*author* = Kate); (b) the content descriptors stating that topics *gmos* and *childCare* have a relevance greater than 50% (*gmos* > 0.5, *childCare* > 0.5), with a trust value greater than 60% ($tv > 0.6$). Preference UP_1 also states that the descriptors and ratings to be considered when computing descriptors trustworthiness are only those satisfying at least one among the owner- and user-defined trust policies — i.e., (`all`, *all* $\cup$). Preference UP_2, specified by Kate, denotes as relevant all the resources whose author is Phil, provided that the descriptor *author* = *Phil* has a trust value and distribution value equal to 100%. Note that, different from UP_1, preference UP_2 does not include content constraints, and it states that, when evaluating descriptors' trustworthiness, the descriptors and ratings to be selected are those satisfying at least one among (a) the user-defined policies or (b) those owner-defined policies selected at run-time by Kate (this is denoted by (`all`, `some`, $\cup$)). Preference UP_3, specified by Phil, includes both property and content constraints, and it states that, when evaluating descriptors trustworthiness, only the descriptors and ratings satisfying (a) at least one among the owner-defined trust policies and (b) at least one among the user-defined trust policies must be considered — i.e., (`all`, `all`, $\cap$). Preference UP_3 asks to block access to resources hosted by about-gmos.net (i.e., *block* = *yes*), when (a) at least 50% of the associated descriptors concerning property *author*, and having a trust value greater than 50%, have a value different from *Kate*, and (b) the content descriptors having a e greater than 40% state that topics *gmos* and *biology* have a relevance greater than 80%. Finally, preference UP_4, specified by Betsy, includes just content constraints, and it denotes the resources satisfying them as carrying relevant information. For evaluating descriptors' trustworthiness, UP_4 states that only user-defined trust policies must be considered — i.e., (`all`, `none`, $\cup$).

Table 5
Examples of user preferences concerning resources having a URI matching pattern http://about-gmos.net*. The URI pattern component has been omitted in the table due to space constraints.

ID	Author	Property constraints	Content constraints	Settings	Assessment	Options
UP_1	Henry	(*author* = Kate, *tv* > 0.5, *dv* ≥ 0.5)	(*gmos* > 0.5, *tv* > 0.6), (*childCare* > 0.5, *tv* > 0.6)	(all, all, ∪)	Safe-info = yes	Block = no
UP_2	Kate	(*author* = Phil, *tv* = 1.0, *dv* = 1.0)	∅	(all, some, ∪)	Relevant = yes	Block = no
UP_3	Phil	(*author* ≠ Kate, *tv* > 0.5, *dv* > 0.5)	(*gmos* > 0.8, *tv* > 0.4), (*biology* > *tv* > 0.4)	(all, all, ∩)	Safe-info = no	Block = yes
UP_4	Betsy	∅	(*nutrition* > 0.8, *tv* > 0.2)	(all, none, ∪)	Relevant = yes	Block = no

Example 10. As an example of how Semantic Web technologies can be used to represent user preferences, Fig. 6 shows the encoding of UP_1 in Table 5 into an N3 rule. More precisely, line 12 denotes the trust policies to be considered (in this case both owner- and user-defined, combined by using OR, as required by UP_1). If, after having evaluated the rules corresponding to the selected trust policies, their conclusions (i.e., the statements inferred from the rules) satisfy the constraints on property/content descriptors in the user preference (lines 14–17), then the resource is marked as carrying safe information, and access to it must be granted (line 20). Finally, line 22 states that the author of such user preference is Henry.

5.3. Trust policies and user preferences enforcement

Whenever a user *m* requests access to a resource *rsc*, the system verifies whether the resource satisfies one or more of the user preferences specified by *m*, and then returns the specified quality assessments and options, if any. To perform this task, the enforcement mechanism (whose main algorithms are reported in Appendix A) takes as input the set of *m*'s user preferences, the set of *m*'s trust policies, the set of owner-defined trust policies associated with *rsc*, the set of labels applying to *rsc*, and the set of the corresponding ratings.

Fig. 7 depicts the main steps performed by the enforcement mechanism, considering, for simplicity, the case in which only a single user preference must be processed. First, the set of labels received as input are processed in order to extract only the descriptors relevant for the considered user preference. Then, from such set of descriptors the mechanism filters out those not satisfying user and/or owner defined preferences, according to what is stated in the considered user preference. If no trust policies apply to the considered descriptors, the set of descriptors computed in the previous step is not modified. Then, the trust value of each descriptor is computed according to the selected trust computation algorithm. Then, the enforcement mechanism checks if the considered descriptors match the user preference. More precisely, property descriptors are evaluated against the *dc* constraints in the considered user preference. Suppose, for instance, that a user preference has the following constraint: *author* = *Kate*, *tv* > 0.4, *dv* > 0.8. This constraint is satisfied if more than 80% (*dv* > 0.8) of the descriptors concerning property *author*, and having a trust value > 0.4, have a property value equal to Kate. By contrast, content descriptors are analyzed in order to compute the average relevance of a given topic and then to compare it with the content constraints specified in the considered user preference. Suppose, for instance, that a user

```
@prefix       voc: <http://mynet.net/voc/#> .                                    1
@prefix       owl: <http://xmlns.com/foaf/0.1/> .                                2
@prefix      foaf: <http://xmlns.com/foaf/0.1/> .                                3
@prefix       log: <http://www.w3.org/2000/10/swap/log#> .                       4
@prefix    string: <http://www.w3.org/2000/10/swap/string#> .                    5
@prefix      math: <http://www.w3.org/2000/10/swap/math#> .                       6
@prefix      wdrs: <http://www.w3.org/2007/05/powder-s#> .                        7
@prefix  property: <http://www.example.com/property#> .                         8
@prefix          : <http://mynet.net/members/> .                                  9

{                                                                                11
  { ?TrustPolicy a [ owl:unionOf (voc:UserDefinedTrustPolicy                     12
      voc:OwnerDefinedTrustPolicy) ];
    log:supports {                                                               13
      ?Resource a foaf:Document; log:uri [ string:startsWith "http://about-gmos.net"] 14
        .
      { ?Resource a foaf:Document; property:author :Kate . } voc:trustLevel [    15
          math:greaterThan "0.5"]; voc:distribution [ math:notLessThan "0.5"] .
      { ?Resource a foaf:Document; property:gmos [ math:greaterThan "0.5"] . }   16
          voc:trustLevel [ math:greaterThan "0.6"] .
      { ?Resource a foaf:Document; property:childCare [ math:greaterThan "0.5"] } 17
          voc:trustLevel [ math:greaterThan "0.6"] .
    } .                                                                          18
  } log:implies {                                                                19
    ?Resource voc:safe-info true; voc:blocked false                              20
  } .                                                                            21
} wdrs:issuedby :Henry .                                                         22
```

Fig. 6. N3-encoding of UP_1 in Table 5.

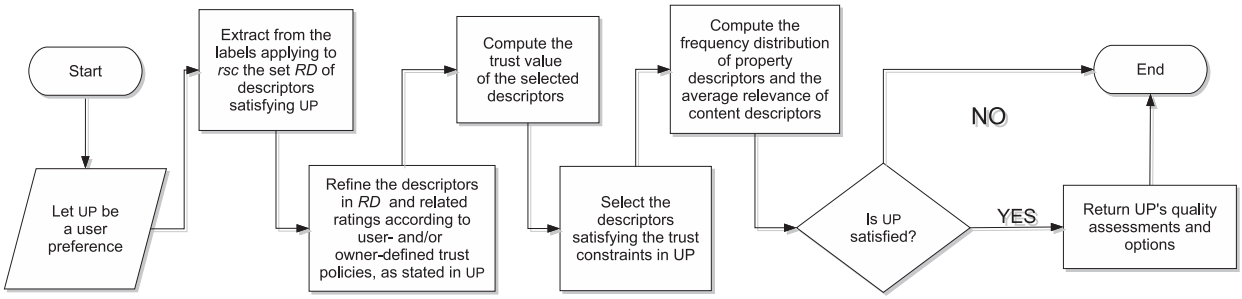

Fig. 7. Main steps of user preference enforcement.

preference has the following constraint: *nutrition* > 0.2, *tv* > 0.4. In order to verify whether it is satisfied, the average relevance value $\bar{\rho}$ of the set of descriptors on topic *nutrition*, having a value > 0.4 is computed. If $\bar{\rho} > 0.2$, the resource matches the user preference. If all the constraints in the user preference are satisfied, then the associated quality assessments and options are returned.

Regarding the trust computation required in step 4 (cfr. Fig. 7), it is interesting to note that several algorithms (e.g., Eigentrust [36] and PeerTrust [37]) may be used for trust computation, which might depend also on the possible values that ratings can assume (e.g., ratings can be binary or scalar, using either discrete or continuous values). Additionally, each user could be associated with a reputation score, which can then be used to assign a specific weight to his/her labels and ratings. This is an issue that has been thoroughly investigated by recommender systems [3], among which there exist examples of online communities, such as MovieLens [38] and MyWOT (http://mywot.com). However, in our framework we do not want to rely on a specific method for trust computation. This because our framework is designed to be applied to different application domains, and thus it has to be able to work with any trust computation algorithm, since it is well known that the notion of trust varies depending on the context and the purposes for which it is used (see [39] for a discussion on this topic). For these reasons, in what follows we denote as $\psi(rd, rsc)$, the trust value of descriptor *rd* of resource *rsc*, computed by the selected trust computation algorithm. Also, for simplicity, we assume that descriptors' trust is specified as a value in [0,1]. However, values from other domains, like strings (e.g., {not trusted, trusted, very trusted}), can be adopted as a front end for the user, and then normalized to [0,1] (e.g., {0,0.5,1}) for internal processing.

The output of the enforcement mechanism is used by the application layer in order to perform a set of operations which depend on the specific agent. As an example, the returned quality assessments can be displayed as graphical and/or textual notifications (like a message window carrying a green icon and stating that the current resource can be safely used, since its content is trustworthy). As far as user preference options are concerned, supposing, for instance, a *block* directive, the user agent will deny access to the resource.

Example 11. Consider user preference UP_1 in Table 5, and suppose that Henry requests access to a resource *rsc* having URI http://about-gmos.net/children/, owned by Kate. In order to determine the action to be performed on *rsc*, the system first retrieves the associated labels, and then extracts from them the set of descriptors concerning property *author* or topics *gmos* and *childCare*.

Then, according to UP_1 *settings*, it selects only the descriptors and associated ratings satisfying one among Henry's trust policies (i.e., TP_2) or the owner-defined trust policies (i.e., TP_3) — see Table 3. In this case, all the descriptors selected in the previous phase satisfy TP_2 or TP_3, whereas the selected ratings are RT_3, RT_4, RT_4, and RT_6 (see Table 2).

Finally, the system computes (a) the trust value $tv_{rd,rsc}$ of each selected descriptor *rd* for resource *rsc*, (b) the average relevance $\bar{\rho}_{cc}$ of the topic in each selected content descriptor *cc*, and (c) the percentage $\delta_{author=\text{Kate}}$ of the set of property descriptors having a trust value greater than 50% and satisfying *author* = Kate. Suppose that $tv_{rd,rsc} = 1$, if *rd* corresponds to *gmos* = 0.8, *childCare* = 1.0, or *author* = *Kate*, whereas $tv_{rd,rsc} = 0$, if *rd* corresponds to *author* = Henry. In our case, we have: $\bar{\rho}_{cc} = 0.8$, if *cc* concerns topic *gmos*; $\bar{\rho}_{cc}$1.0, if *cc* concerns topic *childCare*; $\delta_{author=\text{Kate}} = 1.0$. Since UP_1 is satisfied, the system grants access to resource *rsc*, and marks it as carrying information which can be safely used.

It may happen that no labels are associated with the considered resource or no user preferences are satisfied or that the satisfied user preferences specify different, conflicting assessments and options. For instance, according to a user preference, a given resource is safe for children and access should be granted, whereas another one claims the opposite. How such situations are treated may vary depending on the purpose user agents are designed for, and thus these are issues addressed by the application layer. As an example, when no user preferences are satisfied or no labels are available, the user agent might notify nothing to the end user, or, alternatively, it might return a *default assessment*, which is set by the end user in the configuration parameters of the user agent itself. In contrast, in case of conflicting assessments and options, it is first important to make the end user aware of such conflicts, so that he/she will be able to revise the relevant user preferences accordingly. In addition to this, a conflict resolution mechanism might be supported in order to automatically determine the prevailing user preference, based on a given set of directives specified by the end user. More precisely, the end user should be able to configure the user agent in order to

determine the prevailing quality assessments (e.g., referring to our example, the end user might decide that, between "safe for children" and "not safe for children", the latter must prevail).

6. Application layer: the delicious case studies

The output of the data and rule layers can be used by a set of agents belonging to the application layer for a variety of purposes (cfr. Fig. 1). For instance, social network's data, Web metadata, and ratings can be exploited by semantic search engines in order to find users and resources matching given queries. It is also possible to aggregate such metadata in order to have a general measure of their trustworthiness, which is however independent from the user submitting the query. In contrast, the rule layer can be used to refine the information provided by the data layer by taking into account subjective trust (trust policies) and to denote the quality, relevance, etc. of a resource for a specific user (user preferences).

This requires, however, the availability of a set of metadata huge enough to perform trust computation, and covering a relevant subset of Web resources. In other words, we need to deal with the knowledge acquisition issue, typical of all knowledge representation systems. However, as we have mentioned earlier in this paper, the success of social networking can be the solution. More precisely, online communities supporting social tagging of Web resources and allowing their members to establish relationships, provide a dataset which roughly corresponds to our data layer. Examples of such communities are Delicious (http://delicious.com), Digg (http://digg.com), Reddit (http://www.reddit.com), StumbleUpon (http://www.stumbleupon.com). Therefore, their datasets can be effectively reused by our framework.

Therefore, to provide an example of how our framework can be used in a real world scenario, we have developed a prototype system where the datasets of social tagging online communities are used for two possible application layer services, namely (a) personalized Web search, and (b) personalized access to Web resources. As it is explained in more detail in Section 6.1, the former service is a search engine which associates search results with the corresponding tags, ratings, and trust values, whereas the latter is in charge of enforcing user preferences on the resources visited by a user, and of returning their quality assessment. As a starting point, we have used the dataset of just one of such communities, i.e., Delicious. However, our approach can be applied also to multiple social tagging communities, by extending the modules in charge of retrieving the online datasets.

Delicious is a widely used social bookmarking service. Besides saving and sharing their lists of favorite Web sites, members are also given the ability of tagging them, thus making the Delicious community able to browse the Web according to a content-based criterion. Moreover, Delicious members can also add contacts, thus setting up a social network. In November 2008, Delicious claimed to have 5.3 million registered users and 180 million bookmarked URLs.[8] As such, it provides a huge dataset which fits our purposes. More precisely, Delicious tags and contacts correspond to the Web metadata and users' relationships contained in the data layer of our framework (see Fig. 1). Therefore, it is possible to build on them the remaining services of the data layer, as well as the rule and application layers. To achieve this, we have built a social networking system, referred to as *WPF Social Network* (WSN, for short), which runs on top of the Delicious dataset. Besides implementing the social network and rating services, as well as those services concerning the rule and application layers, the WSN can be used to specify the type of relationships existing among its members, whereas Delicious supports just a generic relationship type.

Fig. 8 provides a graphical representation of how our framework has been built upon Delicious.

In the following sections, we provide the details about the WSN system architecture and implementation, and illustrate the experiments that we have carried out in order to verify the feasibility of our approach.

6.1. System architecture

The main component of our prototype, depicted in Fig. 9, is the WSN Management System (WMS), which is in charge of storing and managing data collected by the WSN, and to enforce trust policies and user preferences.

The WMS is a typical social networking service, providing a Web User Interface through which users can register and log in into the system. Such interface allows members to create a profile, establish relationships, rate existing labels, specify trust policies and user preferences. It can be used also to create labels which will be transparently uploaded to Delicious by using its API.[9]

The WMS is complemented by a WMS User Agent, which is implemented as a browser extension for Firefox. Such application has been developed since the actions specified in the user preferences must be necessarily carried out on the accessed resource by a client-side agent, integrated into the user's browser.

Moreover, it provides direct access to WMS services, like the Semantic, and Trust-aware Search Engine, described later in this section. The WMS also includes a set of bases which correspond to the data managed by the different layers in Fig. 1. Each of such bases is accessible through an API, which can be used to query and manage the stored data.

Thanks to them, it is possible to give access to WMS data to all authorized users and agents, even those external to the WMS. For this purpose, such APIs support requests in multiple protocols, namely, REST [40], SOAP [41], and SPARQL [42].

To retrieve bookmarks and contacts from Delicious, we have designed two modules, namely the Label and Relationship Extractors, which store the retrieved data into the label and relationship bases, respectively. We have decided to adopt this solution instead of retrieving at runtime data from Delicious because this grants a better performance, as it will be explained in Section 6.1.2.

[8] Source: "delicious blog" (http://blog.delicious.com/blog/2008/11/delicious-is-5.html).

[9] The specification of the Delicious API is available at: http://delicious.com/help/api. Note that Delicious, while owned by Yahoo!, also supported an OAuth API, which has been dismissed in April 2011 after the service has been acquired by AVOS Systems. At the time of writing this paper, the Delicious OAuth API was not yet restored.

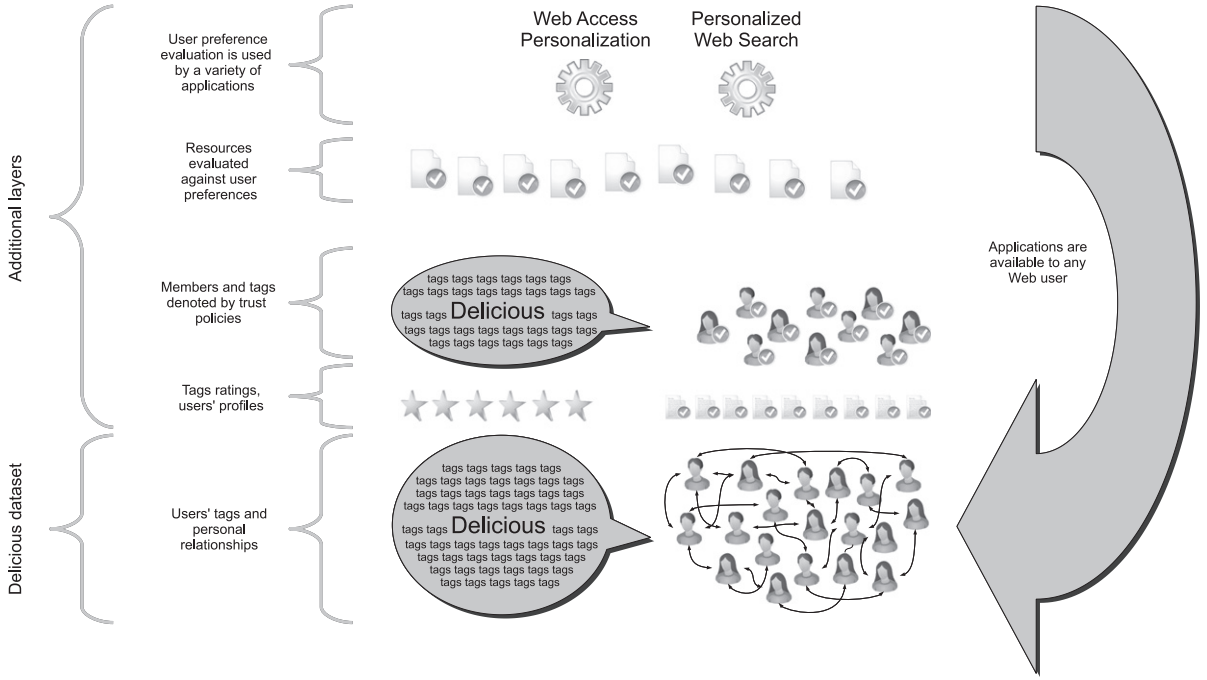

Fig. 8. Integrating Delicious with the proposed personalization framework.

All the queries performed on the WMS bases are managed by the Request Manager.

The Request Manager is also invoked by the User Preference Engine (UPE), and the Semantic and Trust-aware Search Engine (STSE), which are the two application layer services we have developed. The UPE is in charge of enforcing user preferences upon submission of an access request to a Web resource. Once a user preference has been evaluated, the UPE returns the quality assessment of the requested resource, as well as the set of options, if any, specified in the preference (e.g., whether to block or not the access to the resource). This information is returned to the agent installed by the user submitting the access request (i.e., in his/her browser). The STSE is a service which the WMS makes available to any user and agent, even those outside the WMS itself. Basically, the STSE works as a search engine, with the only difference that it returns search results by querying Web metadata stored by the WMS. Each search result is associated with the corresponding aggregated tags and ratings, as well as the computed trust value. Typically, the STSE returns search results only based on the data stored into the Label and Rating Bases. However, if the requesting user is recognized as a member of the WMS, he/she can decide whether they can be evaluated also with respect to the existing trust policies and user preferences. Search results are returned by default in RDF, but they can be converted into other formats by applying specific XSL transformations (XSLTs) [43]. At the moment, XSLTs have been built to return search results in (X)HTML, RSS, and Atom formats.

6.1.1. The delicious dataset

As mentioned earlier in this section, Delicious provides a dataset consisting of two types of information: a set of bookmarks, possibly associated with a set of tags, and the list of contacts of Delicious members. By contrast, Delicious does not support ratings or anything similar to our trust policies and user preferences, whereas the profile of a Delicious member consists only of his/her username, real name, email address, and, possibly, the URL of his/her homepage.

The Delicious dataset can be accessed through: the Delicious Web site and the Delicious API. For our purposes, it would be preferable to use the API, since it returns exactly the requested information, in a format which is not supposed to vary over time, and which can be extracted with a minimal processing overhead. However, the Delicious API has some disadvantages. First, it is still under development and, as stated in its specification, it can be frequently subject to changes. Second, such API does not provide access to all the data available from the Delicious Web site. Actually, the API is meant to give Delicious members the ability of building applications accessing their profiles. Consequently, the API supports not only just a subset of the services provided by Delicious, but the supported services are not flexible as on the Delicious Web site. A relevant example is the number of bookmarks that can be retrieved. The API allows you to retrieve at most the 100 most recent bookmarks about a URL, whereas through the Delicious Web site it is possible to access all the existing bookmarks about a URL (see Section 6.1.2 for more details).

Besides this, there exists another issue concerning the protection of private data. In order to use the API, the user must provide his/her username and password. This means that, in order to exploit the API, the WMS has to be set up with usernames and passwords of all WMS members, which is definitely a strong assumption. For all these reasons, we decided to retrieve the Delicious dataset directly from the Delicious Web site.

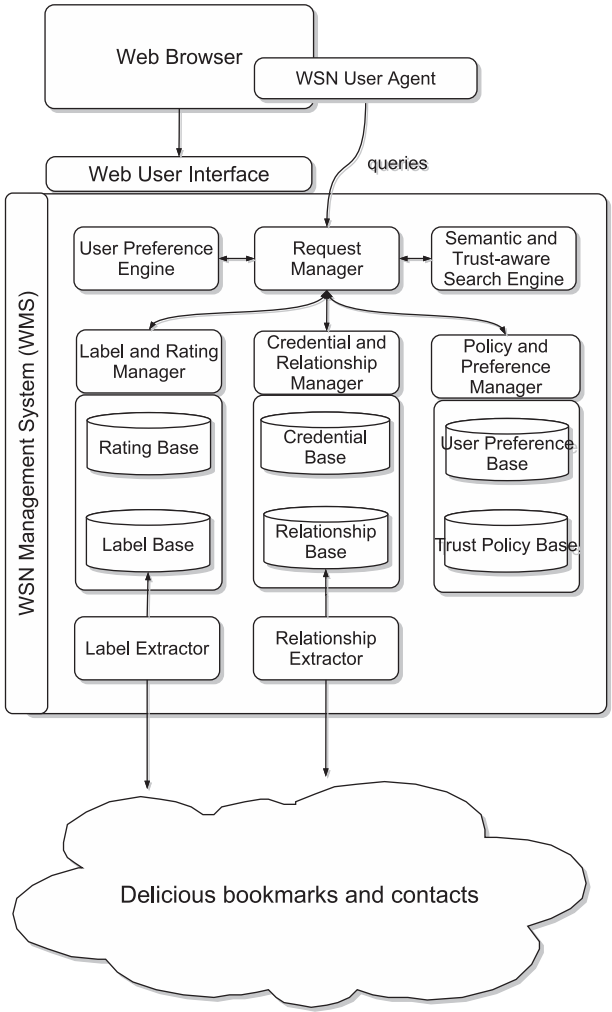

Fig. 9. Architecture of the Delicious prototype.

6.1.2. Implementation details and issues

The WMS is built on top of the PostgreSQL DBMS, by using the PHP language. Although the data collected by the WMS are served in RDF, we have decided to use a relational DBMS instead of an RDF-based one (as Sesame), because at the moment the former still has better performance. The PostgreSQL DBMS has been used not only to manage the WMS's bases, but also to perform the most costly operations, such as labels' and ratings' aggregation, and the enforcement of trust policies and user preferences. We have adopted this strategy, instead of developing specific applications, since we have to process a great amount of data, and a DBMS grants better efficiency for such kind of tasks. By contrast, the PHP language has been used mainly to develop the APIs of the WMS, the modules in charge of returning data in different formats, and those in charge of extracting the Delicious dataset.

The main issue we had to deal with has been how to extract the Delicious dataset. For our purposes, it would have been preferable to extract the dataset at runtime, in order to have the most updated set of tags and contacts. However, this has major drawbacks in terms of performance. This is only partially related to the time required to extract the dataset from the Delicious website, store it into the WMS, and process it. In fact, the major drawback in terms of performance is determined by the time required by the Delicious Web site to return the results of a query, and on how results are displayed. In the experiments we have carried out, Delicious requires an average of ~1.6 s to respond to a request, whereas it returns a maximum of 100 search results per page. This means that, if N is the number of bookmarks applying to a resource, extracting them from the Delicious website requires an average of $1.6 \cdot \frac{N}{100}$ s. Moreover, this implies that the procedure in charge of extracting the relevant information must be performed recursively on each of the pages.

For these reasons, we have decided to avoid retrieving the Delicious's dataset for each request, and to implement synchronization strategies, according to which the set of tags concerning a resource is periodically updated offline. The synchronization depends on a set of parameters, including the number of bookmarks associated with the resource, how frequently the resource is accessed by WMS members, and how old are the retrieved tags.

6.2. Experiments

In order to verify the feasibility of our approach, we have carried out a set of experiments to measure system performance when enforcing user preferences. In this section, we first analyze time complexity, and then illustrate the results of the experiments we have conducted.

As illustrated in Section 5.3, user preference enforcement requires the evaluation of labels and ratings, filtered based on the constraints specified by the applicable trust policies. Let us then first consider the complexity of trust policy enforcement. Trust policy enforcement is the most costly task, since it requires one to evaluate the whole information in the system. More precisely, trustworthy members are selected by evaluating the users attributes (attribute constraints) or their relationships (relationship constraints), whereas retrieving the relevant descriptors, and the corresponding trust values and distribution requires the evaluation of the available Web metadata and the corresponding ratings.

According to our implementation strategies (Section 6.1.2), we have used a relational database to store all the system information. Therefore, users' attributes, users' relationships, resource descriptors, and descriptor ratings are stored into specific tables. As a consequence, the retrieval of this information is performed by SQL queries, which require in the worst case $O(n)$ time complexity, where n denotes the number of rows of the table the query is performed on. Since the tables storing users' attributes, resource descriptors, and descriptor ratings use a different row to encode information of each single user attribute, resource descriptor, and descriptor rating, the total number of rows in such tables is obviously given, respectively, by the sum of the number of attributes, descriptors, and ratings of each user. By contrast, users' relationships correspond to the total number of shortest paths connecting WPF users, computed by performing a breadth-first search (BFS) on the social network graph. The BFS is performed by querying the table storing the direct contacts of WPF members, that is, storing the number of edges of the graph, i.e. $|E_{SN}|$. As such, the complexity is given by $|E_{SN}|$, whose estimation is difficult to have since the number of contacts per user may greatly vary. However, we can estimate the worst case, that is, when each user is connected with all the others: in such a scenario, we have $|E_{SN}| = |M_{SN}| \times (|M_{SN}|-1)$. Moreover, since the edges of the social network graph are labeled with relationship types, the same pair of users can be connected by multiple edges. Consequently, we can use as an upper bound $|E_{SN}| = |M_{SN}| \times (|M_{SN}|-1) \times |RT_{SN}|$.

In order to evaluate user preferences, it is necessary to compute descriptors' distribution and trust values (see Definition 2). The former computation is performed by a query that, for each URL and associated descriptor, returns the number of occurrences of such descriptor on that URL, divided by the total number of descriptors specified for that URL. Consequently, such task requires evaluating a number of rows equal to the total number of descriptors and ratings in the system.

As far as trust computation is concerned, its complexity depends on the adopted algorithm (see [39]). For testing purposes, the descriptors' trust values are computed as follows:

1. let *RD* be a set of descriptors, specified by a given WPF user *m*, and associated with a given resource *rsc*;
2. for each $rd \in RD$, the sum of the ratings on *rd* is divided by the total number of users who created a rating for any of the descriptors applying to *rsc*, thus obtaining a *preliminary* trust value $t_{rd,rsc}$;
3. the *reputation score* $r_{m,rsc}$ of user *m* for resource *rsc* is computed as the average of the preliminary trust values of the descriptors in *RD* specified by *m*;
4. the final trust value of *rd* is given by the product of $t_{rd,rsc}$ and $r_{m,rsc}$.

As far as step 2 is concerned, we have used such strategy in order to give existing ratings a weight which may vary depending on how many of the users who have rated descriptors applying to *rsc* have also rated descriptor *rd*. Such weight *w* will then be equal to 1 when all the users who have rated descriptors applying to *rsc* have also rated descriptor *rd*; $0<w<1$, otherwise. The obtained weighed rating value of *rd* is a sort of *global* trust value of *rd*, i.e., a trust value which reflects the global opinion of the community. However, such value can be refined by taking into account also how much trustworthy are the users who have labeled resource *rsc*, that is, by taking into account what, in this context, we call the *reputation* of a user. This is achieved in two steps. First (step 2) we compute the reputation score of a given user *m*, which depends on how much the other users agree on the labels he/she specified. Then (step 2) such reputation score is used to weigh the trust value obtained in step 2, thus obtaining the final trust value of *rd*.

In order to have an estimation of the time complexity in real world scenarios, we have performed several experiments to compute the time required by trust policy enforcement, by varying the number of selected users, descriptors, and ratings.

The experiments were conducted on a 3.60 GHz Dual-Core Intel Xeon GNU/Linux machine, with 4 GB RAM. We would like to note that this hardware configuration is far from being suitable for a typical Web community, which might collect data from millions of users, and thus this must be taken into account in evaluating the results of our experiments.

As far as trustworthy users are concerned, we have considered two different cases, depending on whether, in a trust policy, they are denoted by attributes or relationship constraints. In order to test system performance when trustworthy users are denoted by attribute constraints, we have varied the total number of users from 1000 to 10,000, and the average number of attributes per user from 10 to 20. Fig. 10 reports the results of our experiments for a total number of users equal to 1000, 5000, and 10,000, and by varying the percentage of trustworthy users selected by a trust policy. As can be seen, response times scale linearly as expected with the total number of rows in the table storing users' profiles.

As we have illustrated earlier in this section, relationship constraints require performing a BFS on the network graph. We have consequently carried out several experiments on the BFS by varying the graph order and degree (i.e., the total number of users and the average number of contacts per user, respectively).

Note that, in the experiments, we have considered the worst case — i.e., when to retrieve trustworthy users denoted by relationship constraints in the trust policy requires exploring the whole network graph.

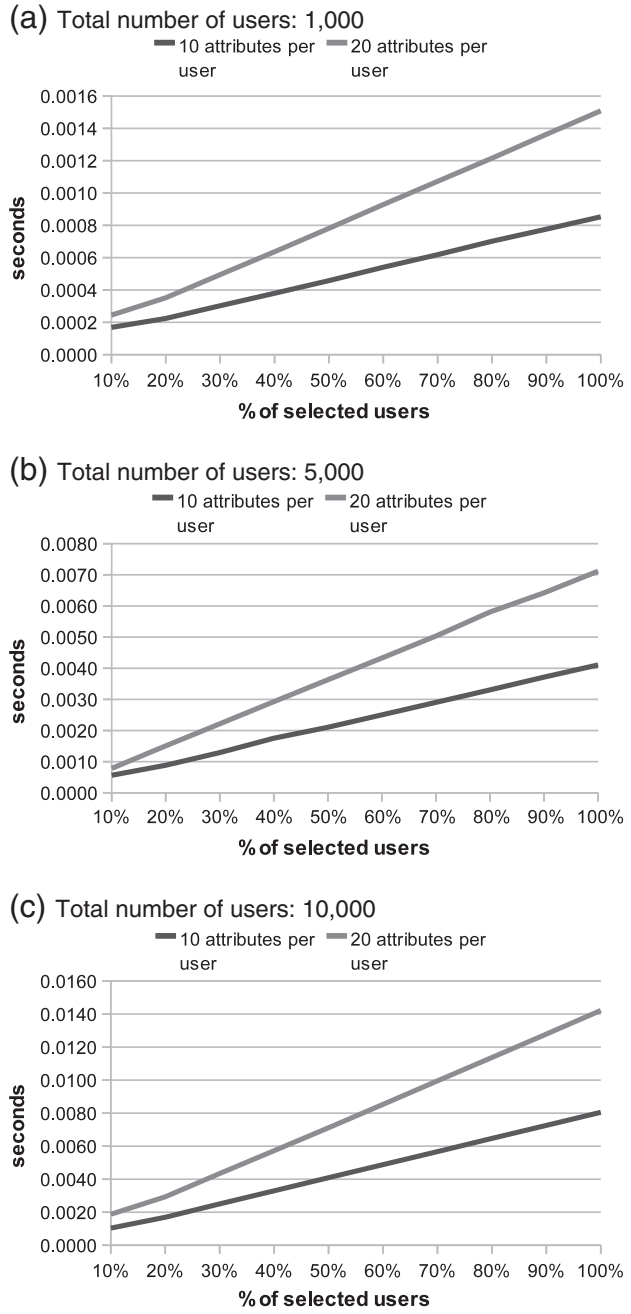

Fig. 10. Time required to select the users denoted by attribute constraints, considering a total number of users equal to 1,000 (a), 5,000 (b), and 10,000 (c).

Fig. 11 summarizes the results of our experiments considering a total number of users ranging from 1000 to 6000 and a number of contacts per user ranging from 10 to 100. As can been seen by comparing the response times reported in Figs. 10 and 11, enforcing relationship constraints is far less efficient than evaluating attribute constraints.

Note, however, that the worst case considered in our experiments on relationship constraints is not likely in a real world scenario. In fact, social networks are topologically similar to small world networks, which are characterized by a small diameter that grows logarithmically with the size of the graph [44–46]. Moreover, the constraints on the relationship type and depth are used to limit the BFS on a subgraph of the social network, thus reducing the number of nodes and edges to be explored.

The evaluation of attribute and relationship constraints affects, however, only part of response times. In fact, after having identified the set of trustworthy users denoted by a trust policy TP, the corresponding set of descriptors and ratings is retrieved and aggregated in order to compute their trust value.

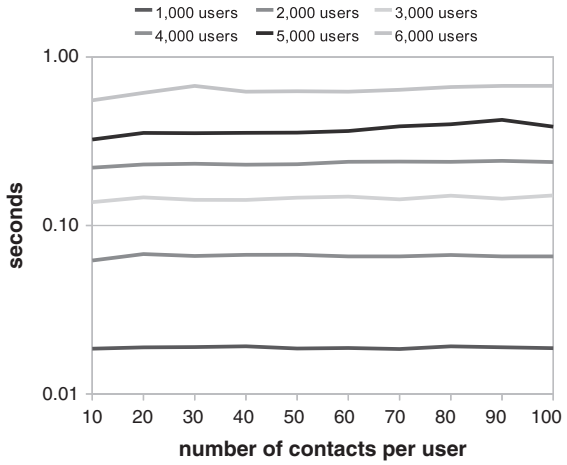

Fig. 11. Time required to select the set of users denoted by relationship constraints. Note that *y*-axis uses a logarithmic scale.

Fig. 12 shows the results of our experiments about the time required to retrieve the set of descriptors and ratings specified by the trustworthy users denoted by a trust policy TP (black area), by varying the number of descriptors and ratings from 1000 to 100,000. Moreover, Fig. 12 shows the time required to compute the trust value and distribution of the selected descriptors (gray area). The reason why we do not distinguish descriptors from ratings in the figure is due to the fact that, as mentioned earlier in this section, we have used the same table to store both descriptors and ratings.

Finally, Fig. 13 merges the results of the experiments reported in the previous figures, by showing the overall time required to evaluate a trust policy, and assuming 10 descriptors and ratings for each of the selected trustworthy users, over a total of 1000 users. More precisely, Fig. 13(a) concerns a trust policy denoting users through attribute constraints, whereas Fig. 13(b) concerns a trust policy denoting users through relationship constraints. As can be seen, in both cases, the maximum response time is around 0.1 s.

By taking into account the hardware configuration we have used for our experiments, which is far from being powerful enough for a typical Web community, we think that the results we have obtained are satisfactory and demonstrate the feasibility of our approach.

6.3. Usability

Traditionally, system personalization in enforced by inferring users' preferences from their data and behaviors. In such a scenario, testing system effectiveness means verifying whether and how much the system is able to guess users' interests, tastes, opinions, etc.

The framework proposed in our paper adopts a different approach, since users' preferences are not inferred by the system, but explicitly specified by end users. Consequently, precision and recall here measure whether and how much a given preference is effective. Basically, this means verifying whether the specified preference and its enforcement reflects the users' intentions and expectations. This issue is very similar to the one concerning the usability of policy specification and enforcement, typical of data security — in particular, access control (see, e.g., [47]). In our specific case, the factors which may affect preference effectiveness can be summarized as follows:

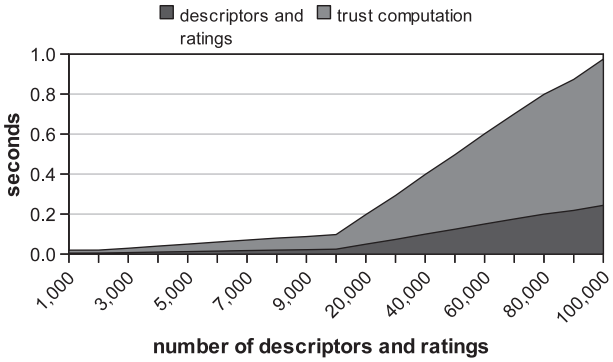

Fig. 12. Time required to select the set of descriptors and ratings specified by the users denoted by a trust policy TP, and to compute descriptors' trust value and distribution.

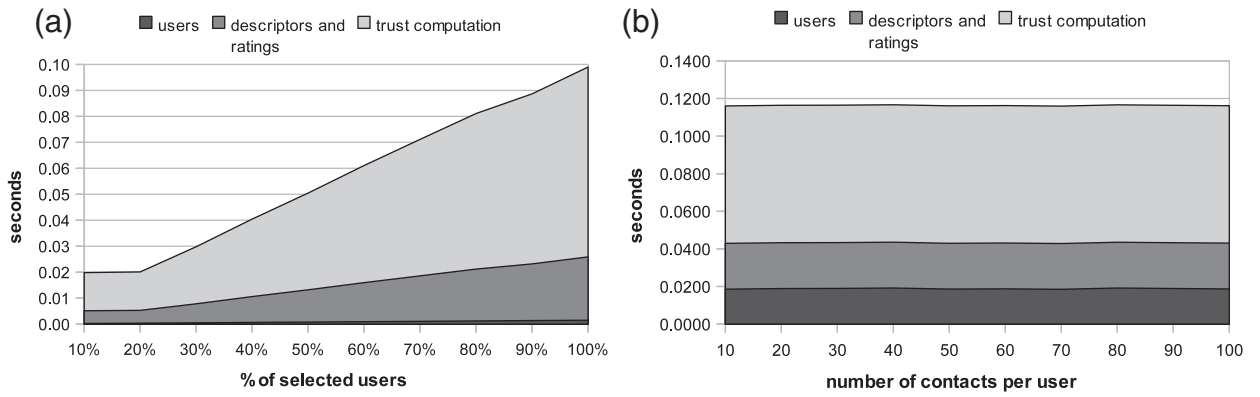

Fig. 13. Time required to evaluate a trust policy TP, supposing a total number of users equal to 1000, and considering both the cases when users are denoted by attribute constraints (a) and relationship constraints (b). The black area denotes the time needed to select the set of users satisfying TP, the dark gray area the time required to select the corresponding set of descriptors and ratings, whereas the light gray area denotes the time required to compute the trust value and distribution of the selected descriptors.

- *Preference specification usability*. This is strictly related to the design of the preference specification interface; besides being intuitive, it must help the user in specifying a preference reflecting as much as possible his/her intentions. Moreover, it must help the user in understanding the effects of a policy specification. For instance, if users are allowed to specify conflicting preferences, as in our framework, it is important to make users aware of the consequences of their decisions, and to verify whether this is what they actually intended (e.g., a user may unintentionally specify conflicting policies);
- *Preference expressiveness*. Ideally, preferences should be able to express exactly what the user intends; in the real word, it is necessary to find a trade-off among usability, expressiveness and efficiency of the enforcement, taking however into account that, if too much expressive preferences may result in low usability, it is also true that a too low degree of expressiveness may result in preferences which do not correctly reflect the users' intentions;

Moreover, even though all the requirements listed above are met, it may be often the case that the specified preferences do not return the expected results. In fact, users may realize that the preferences they specified are either too loose or too restrictive for their purposes. Also, preferences which have correctly worked until a given moment, may be less effective later on. This may happen when preferences denote a set of resources whose content has changed during time, but also when a preference denotes a set of un/trusted users based on their characteristics — i.e., a dynamic group of users which may change its overall trustworthiness depending on the members who join or leave it. It is then necessary to give users the ability of verifying the effectiveness of their preferences and to revise them, if necessary. Note that revising a preference may mean changing the original one and/or specifying exceptions to it.

It is worth noting that all the issues discussed above are mainly related to the usability of the system's front end, and they must be addressed accordingly. However, two issues are also strictly related to the characteristics of the underlying system, namely, preference expressiveness and update. About the former, it is necessary that the semantics of the preferences supported by the system is expressive enough to correctly represent users' intentions. About policy updates, the system must somehow support the specification of exceptions to existing preferences which allow users to obtain the intended results. An example of how to support exceptions is provided by our framework, which allows the specification of conflicting preferences and enforces conflict resolution criteria.

As we have made clear throughout the paper, our contribution focuses on the technical feasibility of the framework we propose, in order to demonstrate that it is technically possible to enhance existing social media by providing personalization features currently not supported, and by exploiting the potential of user-generated content, as social tags are. We do think that it is equally important to verify whether and how the complexity of our framework can be hidden to end users in order to make it usable. However, this would require a separate study, which cannot be included in this paper.

Nonetheless, we have tried to obtain a preliminary evaluation about the usability issues of our framework by setting up an experiment, involving a limited number of users, and testing a given set of features. Our main purpose was to verify whether our framework actually supports the technical requirements concerning preference expressiveness and update, but we have also partially verified other usability issues.

In order to carry out the experiment, we have involved 10 undergraduate students of the Computer Science course at the University of Insubria. We have prepared a testing environment in one of our laboratories, configuring a desktop computer for each one of the participants.

The test has been carried out on an instance of our system, where the front-end included a preference specification interface designed specifically for the experiment. As it can be seen in Fig. 14, depicting a screenshot of the preference specification interface, in the same page are provided both the list of existing preferences, and the preference specification form. If one of the existing preferences is selected, the same form displays its parameters, and it can be used to update or delete the preference itself. The different fields in the form correspond to the components of a user preference (see Definition 2, Section 5.2). For the

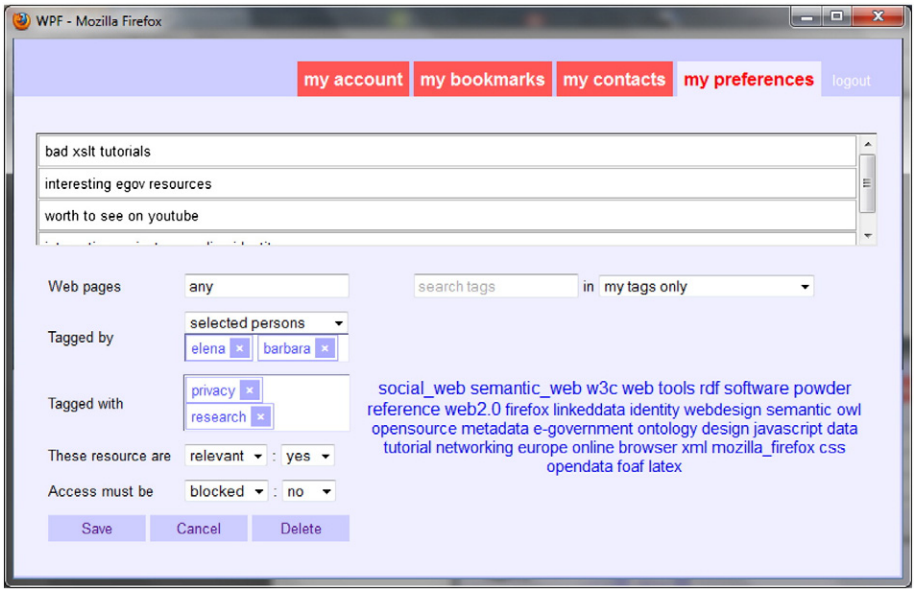

Fig. 14. A screenshot of the preference specification interface.

experiment, we have enabled neither property constraints nor the (all, some, ∪) component, and we simplified the specification of content constraint. Moreover, the form includes also a field to denote the trusted users. This is meant to incorporate in the same page also the specification of trust policies, in a transparent way. Finally, when specifying content constraints, the user can either type a tag, search it among the existing ones, or simply click on the relevant item in the tag cloud. It is also possible to decide which set of tags should be displayed in the cloud (most popular ones, tags specified by contacts, my tags).

The experiment has been designed and carried out according to the following phases:

Preparatory phase. From Delicious, we have collected the URLs of 50 resources annotated with the most popular tags, and we have then partitioned them into 10 groups of 5 resources each. We have then created 10 user accounts on Delicious, and established mutual relationships of type 'colleague' among all them. We have then setup 10 web pages, each one relative to a single user and listing a single group of resources, along the corresponding links. All these pages were linked from a page including the numbered list of user.

First phase. Users, after entering the testing environment, have been asked to access their page, log in into the system, and create bookmarks for the 5 resources assigned to them. When creating a bookmark, users were asked to choose from a minimum of 2 to a maximum of 5 tags among those suggested by Delicious.

Second phase. We have collected the set of tags chosen by the users, and we have then selected those more frequently specified. Then, we have asked the users to access the preference specification page, and to create 4 preferences each using a single tag among those selected before. Users have then been asked to access the resources bookmarked by the other users, check the response given by the system, based on the specified preferences and on the content of the page, and then rate such response to denote whether and how much it had met the expected results.

Third phase. We have then collected the evaluations carried out by the users about the effectiveness of the preferences they have specified. Users have then been asked to revise the specified preferences, if necessary, in order to obtain the expected result. They have been given the options to revise the preference specification itself and/or to create exceptions. Users have then re-tested the updated preferences.

Fourth phase. Users were given a questionnaire, and asked to rate the included questions by choosing one of the following options: "no", "neutral", "yes", "don't know", and to motivate them. The included questions were the following:

- Would you use this application in your everyday life?
- Have you find it useful?
- Have you found it easy to specify preferences?
- Is the logic underlying preference enforcement clear?

We have then collected the questionnaires and then started a discussion with the users by reviewing their answers. The results of such discussion can be summarized as follows:

- The majority of the users were not sure they would have used the application regularly in their everyday life, but they all shared the view that the application would be useful for some contents. Also, most of the users said that they have been positively impressed by the accurate evaluation of the quality of some of the resources they accessed.

- Nearly all the users said the preference specification interface needs improvements, to be more intuitive, and to reduce as much as possible the time and effort needed to specify a preference. A suggestion was to provide preference templates, which can then be customized by user by changing only the relevant settings.
- None of the users said that he/she was not able to specify the intended preferences — i.e., the preference semantics was found to be expressive enough for their purposes.
- Most of the users found the preference revision mechanism helpful to correct the unexpected results. However, they also said that when specifying exceptions more complex than those concerning specific websites, the expected result was often different from the expected one.
- Most of the users said that the application did not significantly increase the response delay of the overall system.

From the outcome of this experiment, we can say that the problems concerning effectiveness are mainly related to usability issues, whereas we had, in general, a positive feedback on the efficiency of the system and the flexibility of our preferences. Such results are far from being conclusive, since more accurate feedback requires setting up an experiment involving a higher number of users with heterogeneous profiles, and testing all the features in the system. Anyway, they already give important information about our framework and its front end, and also provide a first set of suggestions on how to improve usability.

7. Conclusions and future work

In this paper, we have presented a multi-layer framework, based on Semantic Web technologies, able to enhance the services provided by today social tag-based applications. To this purpose, the framework exploits social network facilities and their ability to support collaborative labeling and rating. Key features of our framework are the support for customizable trust policies and user preferences. To test the developed framework, we have implemented two distinct case studies, both based on the Delicious dataset, where the services provided by the framework are used for personalized Web search and Web access personalization.

Besides integrating additional datasets from further social tagging communities, future work will include, on one side, investigating optimization techniques and implementation solutions, different from the ones we have adopted, in order to improve system performance and scalability, and, on the other side, studying how our framework can be extended in order to apply to other application domains. Another important research issue concerns the design principles to be used in the application layer in order to make end users able to effectively and transparently exploit the services supported by our framework. A further relevant future work will be devoted to address the usability issues of the proposed framework. In particular, we believe that the multi-layer framework could be greatly improved by providing users a set of tools in support of labels, trust policies and user preferences specification. However, we are aware that a usable GUI could not be enough, since the proposed system may suffer problems similar to those encountered in the specification of privacy settings in social networks. In this context, many empirical studies (see, e.g., [48]) have shown that average users have difficulties in understanding also the simple privacy settings provided by today's online social networks. To overcome this problem, a promising trend is to exploit data mining techniques to infer the best privacy preferences for social network users, on the basis of the available social network data [49]. As future work, we intend to exploit similar techniques to infer trust policies and users preferences.

Appendix A. Algorithms for trust policies and user preferences enforcement

User preferences are enforced by function EnfUP, illustrated by Algorithm 1. For simplicity, in the algorithm, it is described only the case when the *PN* and *T* components of trust policies (i.e., the set of properties and topics, respectively), and the *PC* and *CC* components of user preferences (i.e., the set of property and content constraints, respectively) are different from the empty set. The algorithm can be easily extended to relax such assumption. Here and in what follows, given a tuple *t*, we denote by *t.comp* the value of the component *comp* of tuple *t*. Therefore, we denote by up.*author*, up.*uriPattern*, up.*PC*, up.*CC*, up.*settings*, up.*Q*, and up.*options* the different components of a user preference up.

Function EnfUP takes as arguments the set *UP* of *m*'s user preferences, the set *UTP* of *m*'s trust policies, the set *OTP* of owner-defined policies associated with *rsc*, the set *LB* of labels applying to *rsc*, and set *RT* of the corresponding ratings. It returns a set of quality assessments, resulting from the analysis of the labels/ratings associated with *rsc*, on the basis of the input user preferences and trust policies. As first step (lines 2–5), function EnfUP sets variables *Assessments* and *Options* to ∅, and performs a first selection of the user preferences to be considered, based on the URI pattern constraint. The resulting user preferences are collected into variable *Prefs*. Then, the function selects the set $\overline{LB} \in LB$ of not expired labels.

Then, (lines 6–10), for each preference $up \in Prefs$, the labels in $\overline{LB}$ are processed in order to extract only the descriptors relevant for up. More precisely, given a label $lb \in \overline{LB}$, the function stores into variable *PDes* the descriptors in lb concerning properties on which the resource property constraints in up are specified. Whereas those descriptors concerning the topics on which the resource content constraints in up are specified are stored into variable *CDes*.

Then, the function analyzes the *settings* component of the considered user preference (lines 11–23). First (lines 12–14) it verifies whether the currently considered user preference requires that user-defined policies must be taken into account, and, in such a case, it invokes function EnfTP (see Algorithm 2). Function EnfTP takes as arguments a set of trust policies (in this case *UTP*), the sets of descriptors *PDes* and *CDes*, and the set *RT* of ratings, and returns the set of descriptors and ratings satisfying at least one of the policies in *UTP*.

A similar procedure is performed on owner-defined policies, if this is required by the currently considered user preference (lines 15–23). The set of owner-defined policies to be evaluated is stored into variable $\overline{OTP}$ which, if the *checkOTP* flag in the current user preference is set to `some`, will correspond to the set of policies selected by the user (line 18); otherwise, $\overline{OTP}$ corresponds to all the owner-defined policies applying to the considered resource (line 20). Variable $\overline{OTP}$ is then passed as argument to function EnfTP, which returns the set of descriptors and ratings satisfying at least one of the policies in $\overline{OTP}$. In the next phase (lines 24-27), the set of descriptors and ratings satisfying user- and owner-defined trust policies are combined according to the *comb* flag in the considered user preference.

The subsequent part of the algorithm (lines 28–31) computes the trust value of each property and content descriptor in *PDes* and *CDes*.

Then (lines 32–49), the descriptors in *PDes* and *CDes* matching the property names and topics in the currently considered user preference are processed. Property descriptors in *PDes* are evaluated against the *dc* constraints expressed in the currently considered user preference (lines 34–39). In contrast, content descriptors in *CDes* are analyzed in order to compute the average relevance of a given topic and then to compare it with the content constraints specified in the currently considered user preference (lines 43–48).

Algorithm 1. User preference enforcement

1: **function** EnfUP(UP, UTP, OTP, LB, RT)
2: $Assessments \leftarrow \emptyset$
3: $Options \leftarrow \emptyset$
4: $Prefs \leftarrow \{$UP $\in UP |$ rsc's URI matches UP.$uriPattern\}$
5: Let $\overline{LB} \subseteq LB$ be the set of not expired labels
6: **for each** UP $\in Prefs$ **do**
7: $PN \leftarrow \{pn \in \mathcal{PN} | \exists (pc, tc, dc) \in$ UP.PC such that pc refers to $pn\}$
8: $PDes \leftarrow \{pd \in \bigcup_{\text{LB} \in \overline{LB}}$ LB.$PD | pd$ concerns a property name in $PN\}$
9: $T \leftarrow \{t \in \mathcal{T} | \exists (cc, tc) \in$ UP.CC such that cc refers to $t\}$
10: $CDes \leftarrow \{cd \in \bigcup_{\text{LB} \in \overline{LB}}$ LB.$CD | cd$ concerns a topic in $T\}$
11: Set $PDes_{UTP}$, $CDes_{UTP}$, $PRat_{UTP}$, $CRat_{UTP}$ to $\emptyset$
12: **if** UP.$checkUTP =$ `all` **then**
13: $(PDes_{UTP}, CDes_{UTP}, PRat_{UTP}, CRat_{UTP}) \leftarrow$ EnfTP(UTP, $PDes$, $CDes$, RT)
14: **end if**
15: **if** UP.$checkOTP \neq$ `none` **then**
16: Set $PDes_{OTP}$, $CDes_{OTP}$, $PRat_{OTP}$, $CRat_{OTP}$ to $\emptyset$
17: **if** UP.$checkOTP =$ `some` **then**
18: Ask m to select a subset $\overline{OTP} \subseteq OTP$ of owner-defined trust policies
19: **else**
20: $\overline{OTP} \leftarrow OTP$
21: **end if**
22: $(PDes_{OTP}, CDes_{OTP}, PRat_{OTP}, CRat_{OTP}) \leftarrow$ EnfTP($\overline{OTP}$, $PDes$, $CDes$, RT)
23: **end if**
24: $PDes \leftarrow PDes_{UTP}$ UP.$comb$ $PDes_{OTP}$
25: $CDes \leftarrow CDes_{UTP}$ UP.$comb$ $PDes_{OTP}$
26: $PRat \leftarrow PRat_{UTP}$ UP.$comb$ $PRat_{OTP}$
27: $CRat \leftarrow CRat_{UTP}$ UP.$comb$ $CRat_{OTP}$
28: **for each** $rd \in PDes \cup CDes$ **do**
29: Let $DR_{rd} \subseteq PRat \cup CRat$ be the set of ratings on rd
30: Let $tv_{rd,rsc} \leftarrow \Psi(rd, rsc)$ be the trust value of rd w.r.t. rsc
31: **end for**
32: $check \leftarrow 1$
33: **for each** $(pc, tc, dc) \in$ UP.PC **do**
34: $ValidPD \leftarrow \{rd \in PDes | tv_{rd,rsc}$ satisfies tc and $rd.pn = pc.pn\}$
35: Let δ_{pc} be the percentage of descriptors in $ValidPD$ satisfying pc
36: **if** δ_{pc} does not satisfy dc **then**
37: $check \leftarrow 0$
38: **break**
39: **end if**
40: **end for**
41: **if** $check = 1$ **then**
42: **for each** $(cc, tc) \in$ UP.CC **do**
43: $ValidCD \leftarrow \{rd \in CDes | tv_{rd,rsc}$ satisfies tc and $rd.t = cc.t\}$
44: Let $\overline{\rho}_{cc}$ be the average relevance of topic in cc
45: **if** $\overline{\rho}_{cc}$ does not satisfy cc **then**
46: $check \leftarrow 0$
47: **break**
48: **end if**
49: **end for**
50: **if** $check = 1$ **then**
51: Add UP.Q to $Assessments$
52: Add UP.$options$ to $Options$
53: **end if**
54: **end if**
55: **end for**
56: **return** $Assessments$, $Options$
57: **end function**

Algorithm 2. Trust policy enforcement

1: **function** ENFTP(TP, $PDes$, $CDes$, RT)
2: $PDes_{TP} \leftarrow \{pd \in PDes | \text{TP} \in TP$, the author of pd satifies TP.$trustedM$, and pd concerns a property $pn \in$ TP.$PN\}$
3: $PRat \leftarrow \{\text{PDR} \in \bigcup_{\text{RT} \in RT} \text{RT}.PDR | \text{PDR is a rating on a descriptor } pd \in PDes_{TP}\}$
4: $PRat_{TP} \leftarrow \{\text{PDR} \in PRat | \text{TP} \in TP$, the author of PDR satifies TP.$trustedM$, PDR.$pd \in PDes_{TP}$, and $\text{RT}_{\text{PDR}}.ts > \text{LB}_{pd}.ts$, where RT_{PDR} is the label rating containing PDR, and LB_{pd} is the label containing descriptor pd, to which PDR applies$\}$
5: $CDes_{TP} \leftarrow \{cd \in CDes | \text{TP} \in TP$, the author of cd satifies TP.$trustedM$ and cd concerns a topic $t \in$ TP.$T\}$
6: $CRat \leftarrow \{\text{CDR} \in \bigcup_{\text{RT} \in RT} \text{RT}.CDR | \text{CDR is a rating on a descriptor } cd \in CDes_{TP}\}$
7: $CRat_{TP} \leftarrow \{\text{CDR} \in CRat | \text{TP} \in TP$, the author of CDR satifies TP.$trustedM$, CDR.$cd \in CDes_{TP}$, and $\text{RT}_{\text{CDR}}.ts > \text{LB}_{cd}.ts$, where RT_{CDR} is the label rating containing CDR, and LB_{cd} is the label containing descriptor cd, to which CDR applies$\}$
8: **return** ($PDes_{TP}$, $CDes_{TP}$, $PRat_{TP}$, $CRat_{TP}$)
9: **end function**

Therefore, the algorithm iteratively considers each constraint (*pc*, *tc*, *dc*) in UP.*PC*, and selects the descriptors concerning the property in *pc*, that satisfy *tc*. Such descriptors are stored into variable *ValidPD* (line 34). Then, the algorithm verifies whether the selected descriptors satisfy the distribution constraint *dc* (lines 35–39). If this is the case, it goes on to perform the same verification with the next constraint in UP.*PC*; otherwise, it sets variable *check* to 0, and terminates the loop on the constraints in UP.*PC*.

If all the constraints in UP.*PC* are satisfied (line 41), the algorithm verifies whether the content constraints in UP.*CC* are satisfied or not (lines 42–49). The algorithm iteratively considers each constraint (*cc*, *tc*) in UP.*CC*, and selects those descriptors concerning the topic *t* contained into *cc*, that satisfy the trust constraint *tc*. Such descriptors are stored into variable *ValidCD* (line 43). Then (lines 44–48), the algorithm computes the average relevance of topic *t*, and verifies whether it satisfies *cc*. If this is the case, it goes on to perform the same verification with the next constraint in UP.*CC*; otherwise, it sets variable *check* to 0, and terminates the loop on the constraints in UP.*CC*. If all the constraints in UP.*CC* are satisfied, the quality assessment specified in UP is added to variable *Assessments* and UP's options, if any, are added to variable *Options* (lines 50–53).

When all the user preferences have been analyzed, the function returns the quality assessments and the options included in the satisfied user preferences (line 56).

References

[1] S.A. Golder, B.A. Huberman, The structure of collaborative tagging systems, Computing Research Repository abs/cs/0508082. URL, http://arxiv.org/abs/cs/0508082, 2005.
[2] J. Voß, Tagging, Folksonomy & Co - Renaissance of Manual Indexing?, Computing Research Repository abs/cs/0508082. URL, http://arxiv.org/abs/cs/0701072, 2007.
[3] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (6) (2005) 734–749, http://dx.doi.org/10.1109/TKDE.2005.99.
[4] P. Heymann, D. Ramage, H. Garcia-Molina, Social tag prediction, In: SIGIR 2008, ACM Press, 2008, pp. 531–538, http://dx.doi.org/10.1145/1390334.1390425.
[5] E. Frías-Martinez, M. Cebrián, A. Jaimes, A study on the granularity of user modeling for tag prediction, In: IEEE/WIC/ACM WIIAT 2008, IEEE CS, 2008, pp. 828–831, http://dx.doi.org/10.1109/WIIAT.2008.67.
[6] P. Chirita, S. Costache, S. Handschuh, W. Nejdl, P-TAG: Large scale automatic generation of personalized annotation tags for the Web, In: WWW 2007, IEEE CS, 2007, pp. 845–854, http://dx.doi.org/10.1145/1242572.1242686.
[7] C.S. Firan, W. Nejdl, R. Paiu, The benefit of using tag-based profiles, In: LA-WEB 2007, IEEE CS, 2007, pp. 32–41, http://dx.doi.org/10.1109/LA-WEB.2007.24.
[8] Z. Yun, F. Boqin, Tag-based user modeling using formal concept analysis, In: CIT 2008, IEEE CS, 2008, pp. 485–490, http://dx.doi.org/10.1109/CIT.2008.4594723.
[9] A. Shepitsen, J. Gemmell, B. Mobasher, R. Burke, Personalized recommendation in social tagging systems using hierarchical clustering, In: RecSys 2008, ACM Press, 2008, pp. 259–266, http://dx.doi.org/10.1145/1454008.1454048.
[10] J. Wang, M. Clements, J. Yang, A.P. de Vries, M.J. Reinders, Personalization of tagging systems, Information Processing and Management 46 (1) (2010) 58–70, http://dx.doi.org/10.1016/j.ipm.2009.06.002.
[11] E. Bertino, C. Dai, M. Kantarcioglu, The challenge of assuring data trustworthiness, In: DASFAA 2009, Vol. 5463 of LNCS, Springer, 2009, pp. 22–33, http://dx.doi.org/10.1007/978-3-642-00887-0_2.
[12] H.L. Kim, A. Passant, J.G. Breslin, S. Scerri, S. Decker, Review and alignment of tag ontologies for semantically-linked data in collaborative tagging spaces, In: ICSC 2008, IEEE CS, 2008, pp. 315–322, http://dx.doi.org/10.1109/ICSC.2008.79.
[13] B. Carminati, E. Ferrari, A. Perego, Combining social networks and Semantic Web technologies for personalizing Web access, In: CollaborateCom 2008. Revised Selected Papers, Vol. 10 of LNICST, Springer, 2009, pp. 126–144, http://dx.doi.org/10.1007/978-3-642-03354-4_11.
[14] S. Staab, Emergent semantics, IEEE Intelligent Systems 17 (1) (2002) 78–86, http://dx.doi.org/10.1109/5254.988491.
[15] X. Wu, L. Zhang, Y. Yu, Exploring social annotations for the Semantic Web, In: WWW 2006, ACM Press, 2006, pp. 417–426, http://dx.doi.org/10.1145/1135777.1135839.
[16] P. Mika, Ontologies are us: a unified model of social networks and semantics, Journal of Web Semantics 5 (1) (2007) 5–15, http://dx.doi.org/10.1016/j.websem.2006.11.002.
[17] J. He, Y. Zhang, G. Huang, J. Cao, A smart Web service based on the context of things, ACM Transactions on Internet Technology 11 (3) (2012) 13:1–13:23, http://dx.doi.org/10.1145/2078316.2078321.
[18] J. Yu, Q.Z. Sheng, J. Han, Y. Wu, C. Liu, A semantically enhanced service repository for user-centric service discovery and management, Data & Knowledge Engineering 72 (2012) 202–218, http://dx.doi.org/10.1016/j.datak.2011.10.005.

[19] E. Kontopoulos, N. Bassiliades, G. Antoniou, Deploying defeasible logic rule bases for the Semantic Web, Data & Knowledge Engineering 66 (1) (2008) 116–146, http://dx.doi.org/10.1016/j.datak.2008.02.005.
[20] A. Uszok, J.M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, S. Aitken, KAoS policy management for semantic Web services, IEEE Intelligent Systems 19 (4) (2004) 32–41, http://dx.doi.org/10.1109/MIS.2004.31.
[21] L. Kagal, M. Paolucci, N. Srinivasan, G. Denker, T.W. Finin, K.P. Sycara, Authorization and privacy for semantic Web services, IEEE Intelligent Systems 19 (4) (2004) 50–56, http://dx.doi.org/10.1109/MIS.2004.23.
[22] N. Damianou, N. Dulay, E. Lupu, M. Sloman, The Ponder policy specification language, In: POLICY 2001, Vol. 1995 of LNCS, Springer, 2001, pp. 18–38, http://dx.doi.org/10.1007/3-540-44569-2_2.
[23] K.P. Twidle, E. Lupu, N. Dulay, M. Sloman, Ponder2 — a policy environment for autonomous pervasive systems, In: POLICY 2008, IEEE CS, 2008, pp. 245–246, http://dx.doi.org/10.1109/POLICY.2008.10.
[24] P.A. Bonatti, D. Olmedilla, Driving and monitoring provisional trust negotiation with metapolicies, In: POLICY 2005, IEEE CS, 2005, pp. 14–23, http://dx.doi.org/10.1109/POLICY.2005.13.
[25] P.A. Bonatti, D. Olmedilla, J. Peer, Advanced policy explanations on the Web, In: ECAI 2006, Vol. 141 of FAIA, IOS Press, 2006, pp. 200–204, URL http://www.booksonline.iospress.nl/Content/View.aspx?piid=1675.
[26] C. Bizer, R. Cyganiak, Quality-driven information filtering using the WIQA policy framework, Journal of Web Semantics 7 (1) (2009) 1–10, http://dx.doi.org/10.1016/j.websem.2008.02.005.
[27] P. Archer, K. Smith, A. Perego, Protocol for Web description resources (POWDER): description resources, W3C Recommendation, World Wide Web Consortium. URL, http://www.w3.org/TR/powder-dr/ Sep. 2009.
[28] G. Klyne, J.J. Carroll, B. McBride, Resource description framework (RDF): concepts and abstract syntax, W3C Recommendation, World Wide Web Consortium. URL, http://www.w3.org/TR/rdf-concepts/ Feb. 2004.
[29] P.F. Patel-Schneider, P. Hayes, I. Horrocks, OWL Web ontology language: semantics and abstract syntax, W3C Recommendation, World Wide Web Consortium. URL, http://www.w3.org/TR/owl-semantics/ Feb. 2004.
[30] T. Berners-Lee, D. Connolly, L. Kagal, Y. Scharf, J. Hendler, N3Logic: a logical framework for the World Wide Web, Theory and Practice of Logic Programming 8 (3) (2008) 249–269, http://dx.doi.org/10.1017/S1471068407003213.
[31] D. Brickley, L. Miller, FOAF vocabulary specification v0.91, Namespace Document. URL, http://xmlns.com/foaf/spec/ Nov. 2007.
[32] I. Davis, E. Vitiello Jr., RELATIONSHIP: a vocabulary for describing relationships between people, Namespace Document. URL, http://vocab.org/relationship/ May 2009.
[33] J.J. Carroll, C. Bizer, P.J. Hayes, P. Stickler, Named graphs, Journal of Web Semantics 3 (4) (2005) 247–267, http://dx.doi.org/10.1016/j.websem.2005.09.001.
[34] S. Konstantopoulos, P. Archer, Protocol for Web description resources (POWDER): formal semantics, W3C Recommendation, World Wide Web Consortium. URL, http://www.w3.org/TR/powder-formal/ Sep. 2009.
[35] P. Archer, A. Perego, K. Smith, Protocol for Web description resources (POWDER): grouping of resources, W3C Recommendation, World Wide Web Consortium. URL, http://www.w3.org/TR/powder-grouping/ Sep. 2009.
[36] S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina, The Eigentrust algorithm for reputation management in P2P networks, In: WWW 2003, ACM Press, 2003, pp. 640–651, http://dx.doi.org/10.1145/775152.775242.
[37] L. Xiong, L. Liu, PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities, IEEE Transactions on Knowledge and Data Engineering 16 (7) (2004) 843–857, http://dx.doi.org/10.1109/TKDE.2004.1318566.
[38] S. Sen, S.K. Lam, A.M. Rashid, D. Cosley, D. Frankowski, J. Osterhouse, F.M. Harper, J. Riedl, Tagging, communities, vocabulary, evolution, In: CSCW 2006, ACM Press, 2006, pp. 181–190, http://dx.doi.org/10.1145/1180875.1180904.
[39] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, Decision Support Systems 43 (2) (2007) 618–644, http://dx.doi.org/10.1016/j.dss.2005.05.019.
[40] R. T. Fielding, Architectural styles and the design of network-based software architectures, Ph.D. thesis, University of California, Irvine (2000). URL http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation_2up.pdf
[41] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, H.F. Nielsen, A. Karmarkar, Y. Lafon, SOAP version 1.2 part 1: messaging framework (second edition), W3C Recommendation, World Wide Web Consortium. URL, http://www.w3.org/TR/soap12-part1/ Apr. 2007.
[42] K.G. Clark, L. Feigenbaum, E. Torres, SPARQL protocol for RDF, W3C Recommendation, World Wide Web Consortium. URL, http://www.w3.org/TR/rdf-sparql-protocol/ Jan. 2008.
[43] J. Clark, XSL transformations (XSLT) – version 1.0, W3C Recommendation, World Wide Web Consortium. URL, http://www.w3.org/TR/xslt Nov. 1999.
[44] D.J. Watts, Small Worlds: The Dynamics of Networks between Order and Randomness, Princeton University Press, Princeton, NJ, 2003.
[45] J. Kleinberg, The small-world phenomenon: an algorithmic perspective, In: STOC 2000, ACM Press, 2000, pp. 163–170, http://dx.doi.org/10.1145/335305.335325.
[46] C. Martel, V. Nguyen, Analyzing Kleinberg's (and other) small-world models, In: PODC 2004, ACM Press, 2004, pp. 179–188, http://dx.doi.org/10.1145/1011767.1011794.
[47] M. Hart, R. Johnson, A. Stent, More content — less control: access control in the Web 2.0, In: W2SP 2007, 2007, URL http://seclab.cs.rice.edu/w2sp/2007/papers/paper-193-z6706.pdf.
[48] K. Strater, H. Richter, Examining privacy and disclosure in a social networking community, In: SOUPS 2007, Vol. 229 of ICPS, ACM Press, 2007, pp. 157–158, http://dx.doi.org/10.1145/1280680.1280706.
[49] L. Fang, K. LeFevre, Privacy wizards for social networking sites, In: WWW 2010, ACM Press, 2010, pp. 351–360, http://dx.doi.org/10.1145/1772690.1772727.