Dealing with Semantic Heterogeneity for Improving Web Usage by Agents

F. Buccafurri, G. Lax, D. Rosaci and D. Ursino

DIMET, Università “Mediterranea” di Reggio Calabria, Italy
e-mail: {bucca,lax,domenico.rosaci,ursino}@unirc.it

Abstract

Designing applications for supporting the user activity on accessing Web information sources is one of the most appealing challenges for researchers in the area of Artificial Intelligence. In this paper we design an agent capable of both creating and managing the profile of a user as well as of exploiting it for discovering navigation paths potentially interesting for him. The agent, during the navigation of a site, provides the user with a set of recommendations and, at the same time, learns his preferences by updating its ontology, storing his profile. The power of the approach relies on the capability of the ontology model (called concept-graph) of representing user-behavior-dependent relationships among concepts and, importantly, dealing with structural and semantic heterogeneity of Web sources. In order to test the validity of our approach, we have implemented the proposed framework in a Java-based multi-agent system.

Key words: Conceptual Data Models, Knowledge Representation Techniques, Ontologies, Web-based Information Systems, Semantic Heterogeneity

* A short abridged version of this paper appeared in the Proceedings of the International Conference on Ontologies, Databases and Applications of Semantics (ODBASE 2002), pp. 1168–1185, Lecture Notes in Computer Science 2519, Springer [7].

1 NOTICE: this is the authors version of a work that was accepted for publication in Data and Knowledge Engineering. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Data and Knowledge Engineering, [VOL 58, ISSUE 3, (September 2006)] DOI: 10.1016/j.datak.2005.06.002

Preprint submitted to Elsevier Science 29 January 2012
1 Introduction

The exponential growth of accessible information sources, mainly due to the very fast expansion of the Web and its permeating information system architectures, has determined the necessity of adequately supporting the user activity for driving his actions in such a large and composite universe. E-commerce is certainly one of the most evident case in which such a need strongly arises, since a frequent reason of low profit is the customer confusion in front of a large offer landscape. In this case, automatic techniques, based on the construction of ontologies, handling personal profiles, may allow the reduction of the search space, by identifying the portion of the offers of interest for the customer. Similar considerations can be made in many other Web application contexts.

To face this problem, one immediately thinks to adopt agent-based solutions, since the above requirements lead to software systems that autonomously react to user actions and continuously learn, from his behavior, new knowledge which is then exploited for supporting his activity. However, one of the main difficulties we have to overcome concerns the inherently heterogeneous nature of the Web. Heterogeneity in the sense of multiplicity of data formats (HTML, XML, text, images, etc.), but also semantic, as, for instance, that originated by the presence of terminological, structural and semantic differences on terms [5]. As an example, consider an agent which should support the navigation, through an e-shop site, of the user John Smith, who is a sommelier. Even though the site contains a section named Ferrari, selling gadgets of the famous Italian car, it should be undesirable that the agent drives the user into this section, only because the word Ferrari appears in the profile of Smith. In fact, Ferrari is also the name of a famous Italian “spumante” wine.

There are many recent proposals in the context of user modeling (see, for example, [38,9,58,50,32,27,4,43,30]). However, conceptual models used in these cases take into account only lexical and syntactic (i.e., structural) information and, thus, they are not capable of capturing the semantics underlying concepts along with the semantic relationships among them.

The problem of dealing with heterogeneity, both syntactic and semantic, has been deeply investigated in the context of Cooperative Information Systems (CIS, for short) [6,12,18,24,11,52,28,22,55]. As a consequence, an interesting research direction is studying how the integration of CIS with agent-based solutions for user modeling can be profitably exploited for designing models and applications capable of supporting Web usage. Basically, the purpose of this approach is to gain the capability of representing user profiles which describe their preferences at conceptual level (that is, the interest of a user for a given concept, the relationships among concepts felt by users, and so on) and,
consequently, the capability of dealing with all the above mentioned forms of heterogeneity.

In this paper we give a contribution in such a context. Indeed, we propose a framework for supporting Web usage, which exploits recent results in the context of Cooperative Information Systems [52,37,45]. In particular, we design an agent capable of creating and managing a user profile and exploiting it for supporting the user activity. Construction and updating of the user profile is done by taking into account the structure of the visited information sources. The agent, during the navigation of a site, carries out, at the same time, two activities: the former consists of providing the user with a set of recommendations for supporting his navigation; the latter consists of monitoring the user behavior for both learning his preferences and encoding such a new knowledge into its ontology, that stores the profile of the associated user. A key issue which has to be faced concerns the model exploited for representing the user profile; this should be capable of embedding both relevant concepts and correlations among them. For this purpose, we define a conceptual model, which we call concept-graph (c-graph, for short). Since the ontology is built by the agent on the basis of user Web navigation, we exploit the results of [52,37] for providing the model with the capability of both representing structural properties of information sources characterized by different formats (i.e., HTML and XML documents, OEM graphs, E/R schemes) and dealing with inter-source heterogeneity. Moreover, c-graphs allow us also to construct a temporary representation of the site navigation, embedding both the structure of the site and the dynamics of the user behavior expressing his preferences. Such a temporary representation is exploited at the end of the visit, for updating the user profile.

This work is an extended version of [7]. Moreover, some preliminary ideas concerning the graph model can be found in [8,53], where the problem of handling e-commerce activities is faced. It is organized as follows. Section 2 describes the concept-graph model, and, together with Section 3, illustrating how inter-source heterogeneity is solved, gives the basis for the construction of the agent model. This issue is faced in Section 4, where the tasks performed by the agent are described. For updating the agent ontology an integration process between two c-graphs is needed: Section 5 illustrates technical details about this issue. In Section 6 we describe how the user ontology is pruned, in order to eliminate those instances and concepts which it is reasonable to consider little interesting for the user. In the Section 7 some experimental results validating our method are presented. Related work are discussed in Section 8. Finally, in Section 9, we draw our conclusions.
2 The Framework for Representing User Ontologies

In this section we describe the framework that will be exploited for representing user ontologies. It is based on a graph of concepts with labeled arcs. Labels encode knowledge about both structure and semantics of visited sites as well as past user behavior. A synthetic function elaborating all components (both structural and behavioral) provides a user-behavior dependent metrics for quantifying concept closeness.

2.1 The Concept-Graph

In this section we present the concept-graph model exploited in the following for representing the user ontology.

The atomic elements of our model are instances belonging to a given universe $\mathcal{U}$. Instances are used for representing different kinds of objects, depending on the data format we refer to (that is, they can be XML instances, or relational tuples, etc.). Since our model is Web oriented, we assume instances are accessible by a user by means of Web documents.

A set of instances, along with a name, represents a concept:

**Definition 1** A concept is a pair $\langle c, \text{name}(c) \rangle$ where $c$ is a set of instances and $\text{name}(c)$ is a string (with a prefixed maximum length).  

As an example, an element of the DTD of a XML site represents a concept whose name is the name of the element and whose instances are the instances of this element.

We denote by $\mathcal{C} = 2^{\mathcal{U}} \times \Sigma^*$ the set of all possible concepts named with strings in an alphabet $\Sigma$. Often, with a little abuse of notation, we refer to a concept just as a set of instances, not as a pair (according to definition above). So, we could say that a given set of instances $s$ is a concept with name $\text{name}(s)$.

The concept-graph, which we next formally introduce, is defined for a given user and a given set of concepts with different names. It contains an explicit representation of membership of instances to concepts, as well as the semantic relationships among them. Moreover, information about user accesses is included. From a physical point of view, user accesses only instances, since concepts do not correspond to physical objects. But we say that a user accesses a concept each time he accesses one of its instances. In order to discover useful information about the relationships among concepts along with the user preference for an instance w.r.t. the other ones of the same concept, we need to
distinguish *internal* accesses from the *external* ones. An access to an instance \( t \) of a concept \( s \) is internal if the user comes from another instance of the same concept \( s \). A user may access an instance \( t \) of a concept \( s \) by an external access if he comes from an instance of a concept \( s' \) distinct from \( s \).

We are now able to formally define the *concept-graph* model.

**Definition 2** Given a user \( u \) and a subset of concepts \( N \subseteq C \), such that there is no concept with same name of another one, a *c-graph* (for \( u \) on \( N \)) is a rooted labeled directed graph \( C_{Graph}(N, u) = (N, A) \), where \( N \) is the set of nodes and \( A \subseteq N \times N \) is the set of arcs.

Informally, \( N \) represents the set of concepts of interest for \( u \) and arcs encode the semantic relationships among concepts. Arc labels define a number of properties associated with the relationships of \( C_{Graph}(N, u) \) and contain also the dependence of the model on \( u \). More precisely, an arc \((s, t)\) is provided with a label \(
\text{label}(s, t) = (d_{st}, r_{st}, h_{st}, \tau_{st})\n\) where \(d_{st}\) and \(r_{st}\) are real numbers ranging from 0 to 1, \(h_{st}\) is a non negative integer, and \(\tau_{st}\) is a non negative real number. Such four *label coefficients* encode different properties. Their definition, which we next provide, clarifies why our graph is directed. In particular:

- \(d_{st}\) is the *(semantic) independence coefficient*. It is inversely related to the contribution given by the concept \( t \) in characterizing the concept \( s \). As an example, in an E/R scheme, for an attribute \( t \) of an entity \( s \), \( d_{st} \) will be smaller than \( d_{st'} \), where \( t' \) is another entity related to \( s \) by a relationship. Analogously, in a XML document, a pair \( \langle \text{element, sub-element} \rangle \) \((s, t)\) will have an independence coefficient \( d_{st} \) smaller than \( d_{st'} \), where \( t' \) is another element which \( s \) refers to through an IDREF attribute.

- \(r_{st}\) is the *(semantic) relevance coefficient*, indicating the fraction of instances of the concept \( s \) whose complete definition requires at least one instance of the concept \( t \). The relevance plays a role similar to the *support* defined for data mining association rules: it gives a measure (in terms of instances) of how much the concept \( t \) characterizes the concept \( s \). Such a coefficient is necessary since the c-graph is used for representing also semi-structured information.

- \(h_{st}\) is the *hit coefficient*, counting the number of hits which \( u \) carries out on \( t \) (i.e., on some instance of \( t \)) coming from \( s \) (i.e., coming from some instance of \( s \)).

- \(\tau_{st}\) is the *(no-idle) time coefficient*, defined as \(\sum_{i=1}^{h_{st}} \frac{c_i}{b_i} \), where \(c_i\) is the effective total time spent by \( u \) at the \( i \)-th hit for consulting the concept \( t \) (i.e., instances of \( t \)) coming from \( s \) (i.e., coming from some instance of \( s \)) and \(b_i = \frac{a_i}{a_{1K}}\) where \(a_i\) is the time spent for loading the relative accessed page and \(a_{1K}\) is the time required for loading a page of size 1K.\(^2\) Observe that

\(^2\) \(a_i\) and \(a_{1K}\) are computed by considering any fixed bit rate.
the role of $b_i$ is giving an adimensional measure of the accessed page size.

Similarly to the relationship between concepts, also the membership of instances to concepts is weighed by labeling. In particular, for each pair $(s, t)$ such that $s$ is a node of the graph and $t$ is an instance belonging to $s$, we define $label(s, t) = (h_{st}, \tau_{st})$, where $h_{st}$, called hit coefficient, and $\tau_{st}$, called (no-idle) time coefficient, are defined in obvious way, coherently with the corresponding coefficients presented above. The only difference here is that the instance which the user comes from is either another instance of the concept $s$ or an instance of another concept. In other words, while hit and time coefficients for pairs of concepts regard only external accesses, the corresponding coefficients in case of pairs $(\text{concept, instance})$ sum both internal and external accesses to instances. For taking into account the locality principle, we assume that, periodically, $h_{st}$ and $\tau_{st}$ are suitably automatically reduced (if greater than 0) in order to gives less importance to old accesses.

The $c$-graph is exploited for representing the user profile. The construction and the management of the user profile is a task of our agent and will be explained in detail in Section 4. As we shall see, the agent extracts, from visited sources, concepts and structural information and stores them in the independence and relevance coefficients. The user behavior is then monitored in order to capture user preferences: hit and time coefficients allow the agent to keep memory of user actions.

For extracting structural information, it is necessary to define how, given an information source, a static $c$-graph (that is, a $c$-graph with hit and time coefficients set to 0) can be automatically built. This is the first step which the agent performs at the beginning of each new visit in order to construct a temporary $c$-graph which the user behavior is traced on. At the end of the visit, such a temporary $c$-graph will be exploited for updating the user profile.

For facing the non-trivial problem of constructing the static $c$-graph from an information source (which might be represented by means of different data formats), we exploit the results obtained in the field of Cooperative Information Systems, and, in particular, the techniques defined in [52,37]. In this paper, a conceptual model called SDR network is proposed which basically corresponds to the notion of $c$-graph deprived of the instances associated with each node and the hit and time coefficients. Thus, for constructing a static $c$-graph, we use the algorithm defined for SDR networks in [52,37]. The algorithm is rather technical and elaborated, therefore, considering also the aim of the paper mainly focused on more abstract aspects, we do not report it in detail. The correspondence of static $c$-graphs with SDR networks, widely accepted in the scientific community of Cooperative Information Systems, allows us to validate the definition of independence and relevance coefficients given above. In particular, [37] includes a deep analysis about the soundness
of such parameters.

We conclude this section by showing an example of a simple user profile.

**Example 1** In Figure 1 a c-graph representing the profile of a user, named *John Smith*, is reported. Labels of arcs are 4-tuples listing, in order, independence, relevance, hit and time coefficients. For simplicity, we do not represent instances.

John Smith is a sommelier, very interested in wine as well as in optimal combination between food and wine. For simplicity, we assume that the navigation history of John Smith consists of just a number of visits to two XML sites, the former regards wine, the latter concerns gastronomy. The DTD of the first site contains the elements *oenology*, *wine* and *vines*. The element *oenology* has a sub-element *news*, and such a sub-element has one IDREFS attribute. An instance of *oenology* collects general news about wine; these may contain links to particular wines or to information about vines. An instance of *wine* is a particular wine, and it may contain links to external instances belonging to the second site concerning recipes.

The second site regards recipes, and its DTD contains the element *recipes*, which has the attributes *ingredients*, *type*, *nationality* and (preparation) *time*.

The topology of the c-graph representing the Smith’s ontology (see Figure 1), reflects the structure of the visited documents. This is true also for structural coefficients, that is, independence and relevance coefficients. For instance, consider the arc (*wine*, *recipes*): the independence coefficient is 1 (that is, the maximum value). This is rather intuitive, since an IDREF attribute just represents a link between the two concepts but not the composition of one concept in terms of the other one. Conversely, for the arc (*recipes*, *ingredients*) the independence coefficient is 0, since *ingredients* is an attribute of *recipes*, and it gives an high contribution in characterizing the concept *recipes*. The relevance coefficient measures how much the instances support relationships. For instance, for the arc (*recipes*, *time*), the relevance coefficient is 0.6, meaning that the 60% of recipes visited by Smith report the preparation time.
Behavioral coefficients (that is, hit and time coefficients), keep memory of actions performed by Smith. For instance, the label of the arc \(\text{(wine, recipes)}\) specifies that Smith has moved from an instance of wine to an instance of recipes for 100 many times, summarizing 800 seconds for consulting all these instances.

2.2 Measuring Concept Closeness

The four coefficients composing labels of a c-graph need to be elaborated in order to become really interpretable. For the rest of this section consider given a c-graph \(C_{\text{Graph}}(N, u)\).

Let \(s\) and \(t\) be two concepts occurring in \(C_{\text{Graph}}(N, u)\). First we examine independence and relevance coefficients \(d_{st}\) and \(r_{st}\). As remarked above, they express structural semantic closeness. In particular, having \((1 - d_{st})\) and \(r_{st}\) close to 1 means that \(t\) strongly characterizes \(s\). \(r_{st}\) has the role of support for such a relationship. Indeed, dealing with semi-structured information, it might happen that only a fraction of instances witnesses such a relationship. A reasonable way for merging the information arising from the two coefficients consists of exploiting the relevance coefficient as a reducing factor of \((1 - d_{st})\). Accordingly, we define the function \(\psi(s, t)\), called \(s\text{(structural)-closeness}\) for giving a synthetic measure of the structural dependence of the concept \(s\) from \(t\).

**Definition 3** Given two concepts \(s\) and \(t\) belonging to \(N\), the \(s\text{(structural)-closeness}\) of \(t\) w.r.t. \(s\) is

\[
\psi(s, t) = (1 - d_{st}) \cdot r_{st}
\]

where \(d_{st}\) and \(r_{st}\) are the independence and the relevance coefficients, respectively.

Also hit and time coefficients have to be elaborated in order to express, in a synthetic way, useful information about the user behavior. However, in this case, we have to consider both the case of a pair \((s, t)\) of concepts and the case of a pair \((s, t)\) where \(s\) is a concept and \(t\) is an instance of \(s\).

The first step is the definition of the function \(\theta\).

**Definition 4** Given a pair \((s, t)\) where \(s\) is a concept belonging to \(N\) and \(t\) is either a concept of \(N\) or an instance of \(s\), we define \(\theta(s, t)\) as:

\[
\theta(s, t) = h_{st} + \left\lceil \frac{r_{st}}{q} \right\rceil
\]
where \( q \) is a suitably set parameter.

Observe that the function \( \theta \), gives a measure of the “interest” the user \( u \) has for \( t \) whenever he accesses \( t \) through \( s \). Moreover, the parameter \( q \) modulates the importance of the effective access time w.r.t. the number of hits. Indeed, for high values of \( q \), \( \theta \) just measures how many times the user contacted \( t \). On the contrary, a small \( q \) cancels the effect of the hit number. The value \( q \) could be determined taking into account several variables, like the connection bit rate, the user expertise, and so on. As an example, in a search tree, an inexpert user typically remains in a failure node for a lot of time before realizing he is not interested in it. Thus, in this case, the effective time spent in each contact cannot give us information about the real interest of the user (i.e., \( q \) should be set to a high value). The contrary happens in case of an expert user.

Now we define the function \( \rho \) representing the preference the user \( u \) gives to \( t \) w.r.t. all the other concepts (in case \( t \) is a concept) or instances (in case \( t \) is an instance) reachable in just one step by \( s \). \( \rho \) is computed as a fraction of the overall interest. More precisely:

**Definition 5** Given a pair \((s, t)\) where \( s \) is a concept belonging to \( N \) and \( t \) is either a concept of \( N \) or an instance of \( s \), we define \( \rho(s, t) \) as:

\[
\rho(s, t) = \frac{\theta(s, t)}{\sum_{t^\prime \in A(s)} \theta(s, t^\prime)}
\]

where \( A(s) \) is either the set of nodes adjacent to \( s \), in case \( t \) is a concept, or \( s \) itself, in case \( t \) is an instance.

At this point we have two synthetic functions, that are \( \psi \) and \( \rho \), the former encoding the structural closeness, the latter the user preference. Observe that \( \rho \) works also on pairs \((s, t)\) where \( s \) is a concept and \( t \) is an instance of \( s \). In order to have a unique synthetic information which, in case of a pair of concepts, summaries both structural and behavioral components we define the function \( \gamma \), called \textit{u-closeness}, measuring the “subjective” semantic closeness of two concepts.

**Definition 6** Given a pair \((s, t)\) where \( s \) and \( t \) are concepts occurring in \( C_{\text{Graph}}(N, u) \) we define the \textit{u-closeness} \( \gamma(s, t) \) (between \( s \) and \( t \)) as:

\[
\gamma(s, t) = k \cdot \psi(s, t) + (1 - k) \cdot \rho(s, t)
\]

where \( 0 \leq k \leq 1 \) is a suitably set parameter.

Observe that the parameter \( k \) in the definition above modulates the importance of the behavioral information against the structural one (a small \( k \) reduces the importance of the structural component). Of course, by producing the above synthetic measure \( \gamma \), we introduce a dependency between structural
closeness and user preference in their usage. The purpose of such a notion is just this: In order to give a uniform measure, we do not allow the unrelated exploitation of the two different dimensions (i.e., the structural closeness and the preference ones), providing the framework with the possibility of setting the infinitely many ways of preferring one dimension w.r.t. the other.

In the following example we apply the above definition to our running case.

**Example 2** Consider the ontology of John Smith of Example 1. Again, consider the arc \((wine, recipes)\). The independence coefficient of the arc is 1. As observed in Example 1, this correctly reflects the structure of the sites visited by Smith. As a consequence, the function \(\psi(wine, recipes)\), measuring how much the concept \(recipes\) structurally characterizes the concept \(wine\), takes the value 0. However, under the perspective of Smith, who is interested in food and wine combination, the two concepts are strongly related each other and this is proved by the high values of behavioral coefficients of the arc \((wine, recipes)\). The function \(\gamma\) encodes this knowledge. Assuming the parameter \(k\) is set to 0.25 (i.e., that the behavioral component is considered more important than the structural one), we obtain that \(\gamma(wine, recipes) = 0.75\), denoting a high degree of semantic closeness.

The function \(\gamma\) allows the notion of *neighborhood* of a given concept \(s\) to be defined. Informally, it consists of the set of concepts that are sufficiently (i.e., up a suitable threshold) “close” (according to the function \(\gamma\)) to the concept \(s\). This is done by extending the function \(\gamma\) to the paths of the c-graph.

**Definition 7** Let \(s\) and \(t\) be a pair of nodes and let \(\pi\) be a path in the c-graph \(C_{Graph}(N, u)\) from \(s\) to \(t\). The \(p\)-closeness of \(t\) w.r.t. \(s\), denoted by \(\gamma_{\text{path}}(s, t, \pi)\), is the sum of the \(\gamma\) values associated with the arcs of \(\pi\).

The \(p\)-closeness allows us to define the notion of closeness between two concepts as the minimum of the \(p\)-closeness.

**Definition 8** Let \(s\) and \(t\) be two nodes such that there exists a path in \(C_{Graph}(N, u)\) from \(s\) to \(t\). The closeness of \(t\) from \(s\), denoted by \(\Gamma(s, t)\), is defined as \(\min\{\gamma_{\text{path}}(s, t, \pi) \mid \pi\text{ is a path in }C_{Graph}(N, u)\text{ from }s\text{ to }t\}\).

We are now prepared to define the concept of neighborhood.

**Definition 9** Given a positive integer number \(k\) and a concept \(t\) occurring in \(C_{Graph}(N, u)\), the \(k\)-neighborhood of \(t\) is the set of nodes \(\{s \in N \mid \Gamma(s, t) \leq k\}\).
Example 3 Consider the c-graph of Example 1 and its concept wine. The 1-neighborhood of wine is the set of concepts \{recipes, time\} since \(\Gamma(wine, recipes) = 0.5\) and \(\Gamma(wine, time) = 0.9\). Moreover, the 2-neighborhood of wine is \{recipes, time, ingredients, nationality, type\} since \(\Gamma(wine, nationality) = 1.01\) and \(\Gamma(wine, type) = \Gamma(wine, ingredients) = 1.14\).

3 Solving Semantic Heterogeneity

In order to use the c-graph model for representing the user ontology, and to design an agent capable of exploiting such an ontology for supporting the user, we have to deal with structural and semantic heterogeneity. Indeed, for instance, the agent must be able to discover that a given concept of the user profile coincides with a concept of a visited site, even though they have different names. In some cases, different names corresponding to two coinciding concepts are simply lexical synonyms and dictionaries available on the Web may be used for detecting them. But sometimes, synonymies can be detected only by means of techniques based on semantic approaches. Another possible case is that two concepts with the same name have actually a different meaning, that is, they are homonyms. We have to discover also such cases, since, clearly, they might cause erroneous agent actions.

We show now an example of non-detectable synonymy (through standard techniques).

Example 4 Consider the profile of John Smith introduced in Example 1. Suppose John Smith visits a XML e-shop site whose associated c-graph (built according to translation rules) is depicted in Figure 2. The site, among others, contains a category called food. On the basis of the interests of Smith, it would be desirable that the agent is able to conduct the user into the section of the site corresponding to food. But, in the Smith’s profile, it appears neither a concept with name food nor some lexical synonym. Indeed, it appears only the concept recipes. We can argue that such a synonym cannot be discovered only by means of dictionaries. This is a case of a semantic synonymy. 

Fig. 2. The e-shop site
As previously pointed out, the issue of detecting semantic heterogeneity has been deeply investigated in the field of Cooperative Information Systems [6,24,11]. Most of the approaches for solving synonymies and homonymies proceed by analyzing the context where the corresponding concepts are defined. Two concepts are detected as synonyms, if their contexts are sufficiently similar, while two concepts are homonymous if they have the same name but their contexts are sufficiently different. In our case, the semantics of a given concept \( c \) can be captured by analyzing concepts sufficiently close to \( c \). Closeness, in this case, is only structural, and thus it is computed on the basis of independence and relevance coefficients (that is, assuming that c-graphs are static). For such a reason, the approach we follow for detecting synonyms and homonyms is similar to that proposed in [52]. The effectiveness of such an approach is widely tested, as the reader may find in [45]. We give here only an informal description of how the technique works.

The context of a concept is captured by adapting the notion of neighborhood defined in Section 2.2 to the static case: this is obtained simply by giving to the function \( \psi \) the role played by the function \( \gamma \). We call this new notion structural neighborhood. Further, given two c-graphs \( CG_1 \) and \( CG_2 \), for any pair of concepts \( s \) and \( t \) belonging to \( CG_1 \) and \( CG_2 \), respectively, we determine a real coefficient, called similarity coefficient, ranging from 0 to 1, expressing how much the concepts \( s \) and \( t \) are similar. Once two suitable thresholds, say \( th_1 \) and \( th_2 \), have been dynamically computed, \( s \) and \( t \) are considered synonymous if their similarity coefficient is greater than \( th_1 \). Conversely, they are considered homonymous if they have the same name and their similar coefficient is smaller than \( th_2 \).

In the following example we show that our notion of similarity allows us to discover the synonymy illustrated in Example 4.

**Example 5** Consider the c-graphs \( CG_1 \) of Figure 1 and \( CG_2 \) of Figure 2. Consider the concept recipes of \( CG_1 \) and the concept food of \( CG_2 \). By applying rules described above, it is possible to verify that the similarity coefficient of the pair \( (\text{food}, \text{recipes}) \) is 0.74 showing that these concepts are (semantic) synonymous (for any reasonable fixed threshold).

4 The Agent Model

User activity is thought as an iterative process consisting of accesses to different sites. We design an agent for supporting such an activity based on handling a user profile stored in the ontology. Such a profile records preferences and semantic relationships learnt by monitoring the history of user activity. The more recent the history, the more important its role in the definition of the
user profile is. Such a behaviour takes into account the locality of user interests; it is obtained by suitably reducing the behavioral coefficients.

The agent carries out two tasks: the former consists of supporting the user during the navigation of each site, by providing him with a set of recommendations; the latter concerns the user profile management (that is, its construction and updating). User profile updating considers the new actions of the user; more specifically, the knowledge coming from new accesses to (possibly already exploited) information sources has to be incorporated into the profile. This is also periodically pruned, in order to eliminate that can be considered just noise. Pruning can be done on the basis of the function $\gamma$, by using a suitable thresholding.

The user profile, denoted by $UP$, is a c-graph, initially empty and updated after each new visit. Assume now the user visits a new site, say $IS$. First, the c-graph $S(IS)$, representing the structure of the site, is automatically built by the agent. Observe that this is a “static” c-graph in the sense that behavioral coefficients (that is, hit and time coefficients) are set to 0. From a practical point of view, the feasibility of such an initial step depends on the presence of some kind of cooperation between client and server sides. Cooperation could be implemented by using either mobile agents or agents working on both sides and exchanging data. Of course, such design decisions could be adopted only after that a precise application setting (like, for instance, e-commerce) has been chosen. However, we have intentionally presented the model in the most general context, with the purpose of focusing our attention on the description of its features. Thus, a deep analysis of the above considerations, as well as of other implementation issues, is outside of the scope of this paper.

Now we describe the two agent activities, that is user navigation support and user profile management.

### 4.1 Supporting User Navigation

By exploiting $S(IS)$ and $UP$, the system supports the user during his navigation by providing him with a set of recommended links (corresponding to collections of URLs) to the concepts of the site. For each visited page, the user receives new recommendations. These are obtained by identifying in $S(IS)$ those concepts which belong also to the user profile. Suggested links are obtained by considering relationships among concepts occurring in the user profile. Thus, recommendations take into account user preferences. As noted in the previous section, in order to discover concepts of $S(IS)$ belonging also to $UP$, semantic heterogeneity has to be solved. Indeed, a given concept of $S(IS)$ can appear in $UP$ as a lexical/semantic synonym, and this can be detected
by using the already described technique. Moreover, a concept of \( S(IS) \) can be a homonym of a concept of \( UP \), and thus could be erroneously considered belonging to \( UP \).

Once synonymies and homonymies have been detected and homonymous nodes in \( S(IS) \) have been renamed, the agent is able to select the concepts of the site (i.e., occurring in \( S(IS) \)) of interest for the user, by finding in \( S(IS) \) the concepts such that there exists at least a synonymous concept appearing in \( UP \).

Let \( C_I \) be such a set of concepts. For each concept \( c \) of \( C_I \) the agent builds the collection of the URLs of all the pages containing instances of \( c \) in the site \( IS \).

At this point, the agent builds a graph, say \( R(IS,UP) \), starting from the set \( C_I \). For each pair \( s \) and \( t \) of concepts in \( C_I \), an arc \((s,t)\) occurs in \( R(IS,UP) \) if there exists in \( UP \) an arc \((s',t')\) such that \( s' \) and \( s \) are synonymous and \( t' \) and \( t \) are synonymous. This graph, called the recommendation graph, is exploited for supporting the user visit in the following way: Initially the agent suggests to the user the page collections associated with \( C_I \), as starting points of the visit. Then, whenever the user visits a page corresponding to an instance of some concept in \( C_I \), say \( c \), the agent extracts from \( R(IS,UP) \) all adjacent concepts, and, for each of them, suggests to the user a link to the corresponding collection. Observe that, in a suggested URL collection associated with a concept \( c \) of \( UP \), the agent may identify the URLs leading to instances detected as sufficiently similar (by text matching techniques) with the \( k \)–first instances of \( c \) ordered by decreasing user preference, determined by means of the function \( \rho \) (see Definition 5), where \( k \) is a given fixed parameter. Note that suggested links might not be actual links of the visited site, but they could only derive from the past behavior of the user, encoded into his ontology, and representing the semantic closeness among concepts he perceives. Thus, the user can be driven by the agent through patterns not directly provided by the site but (probably) appearing as “natural” for the user.

The above mechanism is better explained in the following example.

**Example 6** Consider the user John Smith of Example 1 whose profile \( UP \) is reported in Figure 1. Suppose he visits the e-shop site of Example 2, here called \( IS \). Again, assume that Smith prefers red wines and, thus, he recently and frequently accessed instances of wine regarding red wines. Consequently, Smith often consulted meat-based recipes. Thus, it results that the 10-first instances of wine ordered by preference contain the word “red” and the 10-first instances of recipes ordered by preference contain the word “meat”.

Denote by \( S(IS) \) the e-graph of the site, reported in Figure 2. In this example we show how the agent supports the navigation of Smith in \( IS \). First, semantic heterogeneity is solved: The agent, by computing similarity coefficients, discovers that, besides lexical synonyms, concepts recipes of \( UP \) and food of \( S(IS) \) can be considered as synonyms (as shown in Example 5). Thus, the set
Fig. 3. The recommender graph built by the agent

$C_I$ of concepts of $S(IS)$ appearing also in $UP$ is \{wine, food\}. Therefore, for the concept wine, the agent builds a collection of URLs of pages containing the instances of this concept in IS; after this, it performs the same procedure for the concept food.

At this point the agent builds the recommendation graph $R(UP, IS)$ reported in Figure 3, where for the sake of presentation, we have shown, beside concepts of $R(UP, IS)$, also concepts belonging to $S(IS)$ (they are depicted by light lines). Note that (1) three new virtual links, represented by bold arrows, are added by the agent (namely, the arc from e-shop to food, the arc from e-shop to wine and the one from wine to food), and (2) the links between food and ingredients and between food and nationality are actual links of the site belonging to $R(UP, IS)$.

As a first suggestion the agent submits to the user the list of URL collections associated with the concepts wine and food. Due to the preferences of Smith, among all the URLs associated with wine, the agent selects red wines as favourite (because of the occurrence of the word “red” in the favourite instances of Smith).

Now, suppose that Smith chooses to visit the page Brunello di Montalcino (that is a famous Italian red wine) following one of the links appearing in the URL collection associated with the concept wine, suggested by the agent. On this page, the agent provides the user with a set of suggested links extracted from the recommendation graph $R(UP, IS)$, by considering all concepts adjacent to wine. In this case, there is only one concept adjacent to wine, that is food. Therefore, the agent shows to the user a link to the URL collection associated with the concept food. Moreover, exploiting the most user’s favourite instances of recipes (that is, the concept corresponding to food in the user profile), the agent selects, among the above URLs, those corresponding to the instances containing the word “meat”.

This way, the user, who is expert in combination between wine and food, is correctly driven to the section of the site regarding meat dishes, as the chosen
wine is red, even though the site is not provided with such a direct link, as it appears in Figure 2 (in fact, the two sections of the site are quite far each other).

We remark that such a useful recommendation can be generated only by exploiting the history of the user behavior (showing that he sees a strong relation between concepts wine and recipes), embedded in his profile, and, importantly, the capability of the model of dealing with semantic heterogeneity, allowing the agent to discover the semantic correspondence between the concept recipes of the user profile and the concept food of the site. Thus, this example shows how the (apparently too complex) features of the model are actually needed for effectively dealing with practical situations.

4.2 User Profile Management

The second task of the agent, transparent for the user, concerns the user profile management. Each new visit updates the user profile, in such a way that the behavior history of the user is recorded in it. Clearly, for privacy reasons, the user is allowed to disable the visit monitoring.

For updating the user profile, the agent dynamically builds a c-graph $CG(IS)$ during the visit and, at the end, incorporates the knowledge encoded in $CG(IS)$ into the user profile $UP$ by integrating the two c-graphs. At the beginning of the visit, $CG(IS)$ is empty. During the visit $CG(IS)$ changes, as all concepts accessed by $u$ along with their neighborhoods in $S(IS)$, are recorded in $CG(IS)$ by inserting new nodes and new arcs. Moreover, hit and time coefficients are re-computed at each step according to their definition. Independence and relevance coefficients are taken from the corresponding arcs in $S(IS)$. At the end of the visit, $CG(IS)$ is a representation of the portion of $IS$, visited by $u$ in this session, containing also information about his behavior, thanks to hit and time coefficients. By considering also neighborhoods (in $S(IS)$) of visited concepts, the agent autonomously discovers potentially interesting concepts for $u$ and includes them in $CG(IS)$.

More formally, given a concept $s$, we denote by $nbh(s)$ its $k$-neighborhood, for a given fixed $k$. In obvious way, we define as $arcs(nbh(s))$ the set of arcs induced by the $k$-neighborhood of a concept $s$. Thus, for each access $a$ to an instance $t$ of a concept $s$:

- if $a$ is the first access of the visit, then $s$ is inserted into $CG(IS)$ with only the instance $t$ belonging to it. $h_{st}$ and $\tau_{st}$ are updated (in this case $h_{st}$ is set to 1).
- if $s$ is accessed for the first time, then $a$ is not the first access, and, thus, the user comes from an instance of another concept, say $s'$, then $s$ is inserted
into $CG(IS)$ with only the instance $t$ belonging to it, and an arc $(s', s)$ is also added. $h_{st}$ and $h_{s't}$ are set to 1, $\tau_{st}$ and $\tau_{s't}$ are set to the same measured value. Independence and relevance coefficients of the arc $(s', s)$ are set to the corresponding values occurring in $S(IS)$.

• if $a$ is an external access coming from a concept $s'$, but $s$ was already accessed, then the arc $(s', s)$ is added to $CG(IS)$ if not already present. $h_{st}$ and $h_{s't}$ are increased by 1; $\tau_{st}$ and $\tau_{s't}$ are equally updated.

• if $a$ is an internal access and it is not the first one, then $h_{st}$ is increased and $\tau_{st}$ is updated.

• for every kind of access $a$, nodes of $nbh(s)$ and arcs of $arcs(nbh(s))$ are inserted into $CG(IS)$, if not already occurring in it. Independence and relevance coefficients of inserted arcs are derived from the corresponding arcs of $S(IS)$. Hit and time coefficients are set to 0.

At this point, before handling the choice of a new information source, the knowledge encoded in $CG(IS)$ has to be incorporated into the user profile $UP$, updating it. This is done by integrating the two c-graphs. Integration is described in the next section. After the update of $UP$, $CG(IS)$ is not useful anymore, since the memory of such a visit of $IS$ is kept into the profile, and then it is discarded. Now, $UP$ must be pruned, in order to eliminate all concepts and instances with low interest for $u$. After this last task, $UP$ can be exploited for supporting the next user visits. Pruning is described in Section 6.

5 Integration of two C-Graphs

In this section we describe how two c-graphs are merged into a global c-graph. Informally, this merge consists of the “union” of the two c-graphs executed after that synonymies and homonymies have been eliminated. By computing the similarity coefficients between all possible pairs of nodes (a node belonging to the first c-graph, the other belonging to the second c-graph), synonyms and homonyms are first detected; then, synonymous nodes are renamed in such a way that they have the same name and homonymous nodes are renamed in such a way that they have distinct names. The union of the two “normalized” c-graphs is done by suitably averaging values of arc labels.

Let $CG_1 = \langle N_1, A_1 \rangle$ and $CG_2 = \langle N_2, A_2 \rangle$ be two c-graphs. The union of $CG_1$ and $CG_2$, denoted by $U(CG_1, CG_2)$, is a directed labeled graph with the
following set of nodes:

\[ N = \{ s \in N_1 \mid \exists t \in N_2 \text{ s.t. } \text{name}(s) = \text{name}(t)\}\cup \{ s \in N_2 \mid \exists t \in N_1 \text{ s.t. } \text{name}(s) = \text{name}(t)\}\cup \{ x \in C \mid x = s \cup t, s \in N_1 \land t \in N_2 \land \text{name}(s) = \text{name}(t) = \text{name}(x)\} \]

where, we recall, \( C \) represents the universe of concepts \(^3\). The set of arcs is defined as:

\[ A = \{ (s, t) \mid \exists(s_1, t_1) \in A_1 \text{ s.t. } \text{name}(s) = \text{name}(s_1) \land \text{name}(t) = \text{name}(t_1)\}\cup \{ (s, t) \mid \exists(s_1, t_1) \in A_2 \text{ s.t. } \text{name}(s) = \text{name}(s_1) \land \text{name}(t) = \text{name}(t_1)\} \]

In words, nodes are obtained by copying those nodes of each c-graph whose name does not appear in the other c-graph and by merging nodes with common name into a single node with equal name including all the instances of the original nodes. Arcs are obtained in an obvious way.

We define now how labels are determined. Let \( (s, t) \) be an arc belonging to \( A \). In order to understand how the label \( \langle d_{st}, r_{st}, h_{st}, \tau_{st} \rangle \) of \( (s, t) \) is determined, we have to distinguish three cases:

(a) \( s \) is not the result of the merge of two nodes. Thus, the arc \( (s, t) \) comes from either \( CG_1 \) or \( CG_2 \) and, further, the number of instances of \( s \) is the same as the number of instances of the corresponding concept in the c-graph which it comes from. In this case, the only difference between the arc \( (s, t) \) and the corresponding arc of the c-graph it comes from is at most the number of instances of the concept \( t \). Since the only label element affected by the number of instances is the relevance coefficient, and since it depends only on the number of instances of the source concept \( (s, \text{ in this case}), the label of \( (s, t) \) is simply obtained by copying the label of the original arc.

More formally, \( d_{st} = d_{s_1t_1}, r_{st} = r_{s_1t_1}, h_{st} = h_{s_1t_1}, \tau_{st} = \tau_{s_1t_1}, \) if \( \exists(s_1, t_1) \in A_i \text{ s.t. } \text{name}(s) = \text{name}(s_1) \land \text{name}(t) = \text{name}(t_1) \land \land s_2 \in N_j s.t. \text{name}(s) = \text{name}(s_2), \) where \( i, j \in \{1, 2\} \) and \( i \neq j \).

(b) \( s \) is the result of the merge of two nodes and \( t \) is not the result of the merge of two nodes. As a consequence, the arc \( (s, t) \) comes from either \( CG_1 \) or \( CG_2 \) (also in this case the arc \( (s, t) \) has not been obtained as a merge of two arcs). However, since \( s \) has been obtained as a merge of two nodes sharing the name, one coming from \( CG_1 \) and the other from \( CG_2 \), and since the set

\(^3\) We recall that, although according to Definition 1 concept is a pair – set of instances and name –, we refer to a concept just as a set of instances. So, we say that a given set of instances \( s \) is a concept with name \( \text{name}(s) \).
of instances of $s$ has been computed as the union of the sets of instances of the two parent nodes, the relevance coefficient of the arc $(s, t)$ has to be re-computed, while the other coefficients of the label can be determined simply by copying the corresponding values of the original label.

More formally, $d_{st} = d_{s1t1}, r_{st} = f(r_{s1t1}, |s_2|), h_{st} = h_{s1t1}, \tau_{st} = \tau_{s1t1}$, if $\exists(s_1, t_1) \in A_i \ s.t. \ name(s) = name(s_1) \land name(t) = name(t_1) \land \exists s_2 \in N_j \ s.t. \ name(s) = name(s_2) \land \exists(s_2, t_2) \in A_j \ s.t. \ name(t_1) = name(t_2)$, where $i, j \in \{1, 2\}, i \neq j$ and we denote by $f(r_{s1t1}, |s_2|)$ the function for recomputing the relevance coefficient of the arc $(s_1, t_1)$ when the cardinality of the node $s_1$ is increased by $|s_2|$.

(c) both $s$ and $t$ are the result of a merge of two nodes. This means that the arc $(s, t)$ corresponds to two arcs, one occurring in $CG_1$, the other occurring in $CG_2$. Thus, it is necessary to merge also these two arcs in order to generate the arc $(s, t)$. This has a direct impact on how the label is calculated. Clearly hit and time coefficients are obtained as the sum of the corresponding values appearing in the two parent arcs. Since the relevance coefficient depends only on the number of instances of the source node, it can be obtained as in case (b). Finally, the independence coefficient is heuristically determined as a weighted mean (through the number of instances of the source node) between the two parent independence coefficients.

More formally, $d_{st} = \frac{|s_1|d_{s1t1} + |s_2|d_{s2t2}}{|s_1| + |s_2|}, r_{st} = f(r_{s1t1}, |s_2|), h_{st} = h_{s1t1} + h_{s2t2}, \tau_{st} = \tau_{s1t1} + \tau_{s2t2}$, if $\exists(s_1, t_1) \in A_i \ s.t. \ name(s) = name(s_1) \land name(t) = name(t_1) \land \exists s_2 \in N_j \ s.t. \ name(s) = name(s_2) \land \exists(s_2, t_2) \in A_j \ s.t. \ name(t_1) = name(t_2)$, where $i, j \in \{1, 2\}, i \neq j$ and $f(r_{s1t1}, |s_2|)$ is that defined in (b).

The algorithm for producing the union of two c-graphs is trivially induced by the above description.

With a little abuse, we assume that $U(CG_1, CG_2)$ so obtained is a c-graph. Note that this is not necessarily true since $U(CG_1, CG_2)$ might not to be rooted. However, in this case, a dummy root can be added to make $U(CG_1, CG_2)$ a c-graph.

In absence of lexical and semantic heterogeneities, the integration of two c-graphs would correspond to the union defined above. Unfortunately, the correspondence between concept names, which the union is based on, does not take into account the possible presence of synonymies and homonymies. Thus, before applying the union operation, we have to solve such a heterogeneity as shown in Section 3.

Let $T = \{(s, t) \mid s \in N_1 \land t \in N_2 \land name(s) \neq name(t) \land sim(s, t) \geq th_1\}$, where, $sim(s, t)$ is the similarity coefficient of the pair $(s, t)$ and $th_1$ is the threshold computed for detecting synonymies (see Section 3). $T$ represents the set of all pairs of candidate synonyms.
Any subset $\bar{T}$ of $T$, such that both (1) $(s, t) \in \bar{T}$ implies that there is no $(s', t) \in \bar{T}$ and $t \neq t'$ and (2) $(s, t) \in \bar{T}$ implies that there is no $(s', t) \in \bar{T}$ and $s \neq s'$, is an admissible synonym set, that is a set of pairs of candidate synonyms such that a concept of $N_1$ ($N_2$, resp.) is involved at most once. We require this last condition since we intend to solve synonyms by renaming candidates pairs of concepts in such a way that they have the same name. Thus, the above condition guarantees that, after this operation, there are not nodes in $CG_1$ (resp., in $CG_2$) sharing the name (according to the definition of c-graph).

Among all admissible synonym sets, we choose a set with the maximum value of the global similarity, where, for global similarity of an admissible set $\bar{T}$, we mean the sum of all the similarity coefficients of the pairs belonging to $\bar{T}$. Such a set, called $S$, represents the set of the detected synonyms.

Now, it is possible to determine the set of homonyms. It is defined as $H = \{(s, t) \mid s \in N_1 \land t \in N_2 \land name(s) = name(t) \land sim(s, t) \leq th_2\}$. Note that, a concept $s$ of $N_1$ (resp., $N_2$) can appear in $H$ at most once.

At this point $S$ and $H$ can be used for solving synonymies and homonymies in $CG_1$ and $CG_2$. In particular, for each pair $(s, t) \in S$, the concept $t$ is renamed in such a way that $name(t) = name(s)$; for each pair $(s, t) \in H$, the concept $t$ is renamed in such a way that $name(t) \neq name(s)$. Let $\bar{CG}_1$ and $\bar{CG}_2$ be the two c-graphs so obtained. The integration of $CG_1$ and $CG_2$ is then obtained by computing the c-graph $U(\bar{CG}_1, \bar{CG}_2)$.

As a final remark we observe that the integration strategy presented here has been designed on the trace of the integration algorithm defined for SDR networks (i.e., static c-graphs) presented in [45], adapted to the dynamic context. For this reason we do not produce here experiments for validating the effectiveness of such an integration procedure whose validity is widely tested in [45].

### 6 Pruning the User Profile

In this section we describe how the user profile $UP$, considered given in the whole section, is pruned in order to eliminate those instances and concepts which it is reasonable to consider little interesting for the user $u$. Obviously, such a process has to be executed periodically.

The policy we use for implementing this task exploits both the function $\rho$ (see Definition 5) and the independence coefficient. Pruning may involve both concepts and single instances. Consider the case of a concept and let $t$ be a
concept. The preference of the user w.r.t. $t$ can be measured by means of the function $\rho$. Thus, if for all the arcs $(s, t)$, the value of $\rho(s, t)$ is low (i.e., it is under a suitable threshold), we can conclude that the history of accesses of $u$ shows a poor interest of him w.r.t. $t$. But this is not enough for us in order to prune $t$. Indeed, $t$ could become interesting for $u$ in the future. What we can say about this? Following a semantic locality principle, it is reasonable to conclude that probably $u$ will not access $t$ if the semantic “independence” degree of $t$ (measured by using the independence coefficients) on all the other (sufficiently interesting) concepts is over a suitable threshold. In this case, $t$ is pruned by $UP$.

In case of instances, pruning is done as for concepts by considering only the function $\rho$.

To avoid that “newborn” concepts and instances are pruned, due to the low value of the function $\rho$, we require an additional condition in order to prune a concept/instance $s$, that is a function last_access computed on $s$ must return a time stamp not “too recent” (w.r.t a given threshold value).

The above policy can be formalized by introducing some notions.

**Definition 10** Let $t$ be either a concept or an instance of $UP$. We denote by $\rho_{\text{max}}(t)$ the maximum value of $\rho(s, t)$, for all the pairs $(s, t)$ which $\rho$ is defined on.

The above definition captures the current interest of the user w.r.t. the concept/instance $t$. In order to implement the locality principle, we need to exploit the independence notion, as follows:

**Definition 11** Let $t$ be a concept occurring in $UP$ and let $k$ be a positive real value. The $k$-minimum independence of $t$, denoted by $\delta_{\text{min}}(t, k)$, is defined as $\min\{d_{st} \mid s$ is a node of $UP$ and $\rho_{\text{max}}(s) > k\}$, where $d_{st}$ is the independence of $t$ on $s$.

Now we are ready to describe our pruning police.

Consider given four threshold values, namely $l_b$, $u_b$, $u'_b$ and $\tau$, and let $cts$ be the current time stamp. Pruning is done according to the following rules:

- Let $t$ be a node of $UP$. $t$ is eliminated from $UP$ iff both the following conditions are verified: (1) $\rho_{\text{max}}(t) < u_b$, (2) $\delta_{\text{min}}(t, u_b) > l_b$ and (3) $cts - \text{last_access}(t) > \tau$.
- Let $t$ be an instance appearing in $UP$. $t$ is eliminated from $UP$ iff both the following conditions hold: (1) $\rho_{\text{max}}(t) < u'_b$ and (2) $cts - \text{last_access}(t) > \tau$.

It is immediate to verify that the above rules implement the policy informally
described above. Finally, observe that all the above functions are polynomially computable.

7 Experimental Results

In this section we report the experiments we have conducted for testing the performance of our system. After a brief description of the data set and the sample of users involved in experiments, Section 7.1 presents the evaluation measures we have chosen for testing our approach. Finally, in Section 7.2, we illustrate the results of our tests. We have implemented a JAVA prototype for executing experiments. The experiments were conducted on a set $USet$ of 60 users. In all tests we have considered a pre-built set $WSet$ of Web pages belonging to two sets of domains, each organized into a specialization hierarchy: (1) "E-Shop", "Drinks", "Wine" and (2) "Books", "Computer Science Books", "Network Books".

7.1 Evaluation Measures

In our tests we have computed several widely accepted evaluation measures, described in the following three sections.

7.1.1 Precision, Recall, F-Measure and Overall

*Precision* is defined as the share of pages accepted by a user among those recommended by the system; *Recall* is the share of pages suggested by the system among those chosen by the user. *F-Measure* represents the harmonic mean between Precision and Recall. *Overall* measures the post-match effort needed for adding false negatives and removing false positives from the set of matchings returned by the system to evaluate (see [54] for details about these measures). In order to define how the above parameters have been computed, we give the following notations. Given a user $i$, we denote by $k$ the page accessed by $i$ and by $LP_k$ the set of pages reachable by links occurring in $k$. $UserRSet^k_i \subseteq LP_k$ represents the the subset of pages considered interesting by the user $i$ and $SystemRSet^k_i \subseteq LP_k$ is the set of pages recommended by our system.

The Precision $Pre^k_i$, the Recall $Rec^k_i$, the F-Measure $F^k_i$ and the Overall $Ove^k_i$, associated to the visit of $i$ to the page $k$, have been obtained by applying the
formulas (see [54]):

\[ 
Pre_i^k = \frac{|SystemRSet_i \cap UserRSet_i^k|}{|SystemRSet_i|} \quad \quad \quad \quad 
Rec_i^k = \frac{|SystemRSet_i \cap UserRSet_i^k|}{|UserRSet_i^k|} 
\]

\[ 
F_i^k = 2 \cdot \left( \frac{Pre_i^k \cdot Rec_i^k}{Pre_i^k + Rec_i^k} \right) \quad \quad \quad \quad 
Ove_i^k = Rec_i^k \cdot \left( 2 - \frac{1}{Pre_i^k} \right) 
\]

The Average Precision AvgPre, the Average Recall AvgRec, the Average F-Measure AvgF and the Average Overall AvgOve have been obtained by computing the mean over all Web pages and all users.

Observe that all the above average parameters but Average Overall belong to the real interval \([0, 1]\). Average Overall ranges between \(-\infty\) and 1; the higher the value of such parameters, the better the accuracy of the system is.

### 7.1.2 MAE and RMSE

We have adopted also MAE [48] and RMSE [47], that are two widely accepted metrics in recommender systems. We have proceeded as follows:

- \(i\) was asked to provide a subset \(WPSet_i^j \subseteq WPSet\) best matching his interests; moreover, for each page \(k \in WPSet_i^j\), he was required to provide a rank \(r_{i,k}^j\), belonging to the real interval \([0, 1]\), representing his interest for the page \(k\).
- Our prototype was run for determining a subset \(WPSet_i' \subseteq WPSet\) that it considers the most interesting for \(i\); for each page \(k \in WPSet_i'\), \(r_{i,k}''\) (belonging to the real interval \([0, 1]\)) represents the interest of \(i\) for \(k\) estimated by our system.
  \[ 
r_{i,k}'' = \max_{c \in k} \{ \text{avg}_{(q,c) \in c-graph_i} \{ \gamma(q,c) \} \} \]
  where \(c\) is a concept appearing in the page \(k\), \((q,c)\) is an arc belonging to the concept graph of the user \(i\), and \(\gamma(q,c)\) is the u-closeness between \(q\) and \(c\) in the concept graph of the user \(i\) (see Definition 6). Observe that \(r_{i,k}''\) so defined computes the maximum value among the average u-closeness values computed for every concept appearing in the page \(k\). The average u-closeness of a concept \(c\) is obtained by averaging the u-closeness values of all the incoming arcs \((q,c)\) occurring in the concept graph of the user.
- The parameter \(UInterest_i\), specifying the interest of \(i\) in the pages belonging to \(WPSet_i^j\), has been computed as \(UInterest_i = \sum_{k \in WPSet_i^j} r_{i,k}^j\).
- The parameter \(SInterest_i\) specifying the interest of \(i\) in the pages belonging to \(WPSet_i''\) measured by the system, has been computed as \(SInterest_i = \)
\[
\sum_{k \in WPSet} \frac{r''_{i,k}}{|WPSet|}.
\]

- \(MAE\) and \(RMSE\) have been computed for measuring the difference between the interest perceived by the user and the interest computed by the system. \(MAE\) and \(RMSE\) are defined as follows:

\[
MAE = \frac{\sum_{i=1}^{\text{|USet|}} |UInterest_i - SInterest_i|}{\text{|USet|}} \quad \text{and} \quad RMSE = \sqrt{\frac{\sum_{i=1}^{\text{|USet|}} (UInterest_i - SInterest_i)^2}{\text{|USet|}}}.
\]

Clearly, the lower \(MAE\) and \(RMSE\) are, the more accurate the system is.

### 7.1.3 Newell Distance

The third metrics we have adopted is the Newell Distance [36]. In this case each user \(i\) was required to provide a rank \(r'_{i,k}\) for each Web page \(k \in WPSet\). After this, the system was run in order to compute \(r''_{i,k}\), for each Web page \(k \in WPSet\). Newell Distance aims at computing the “distance” existing between the choices of a user and the corresponding ones of the system. In order to compute the Newell Distance, we need the following preliminary notions:

- \(U-WP_i\), the User Ordered Web Page List (resp., \(S-WP_i\), the System Ordered Web Page List), is the list obtained by ordering pages of \(WPSet\) by decreasing user-assigned (resp., system-assigned) rank.
- \(upos\) (resp., \(spos\))): \(WPSet \rightarrow \{1, \ldots, |WPSet|\}\) is a function that receives a Web page \(k \in WPSet\) and returns an integer \(l\) representing the position of \(k\) in \(U-WP_i\) (resp., \(S-WP_i\)).

We are now able to define the Newell Distance between \(U-WP_i\) and \(S-WP_i\). The Newell Distance associated with \(U-WP_i\) and \(S-WP_i\) is:

\[
N_i(U-WP_i, S-WP_i) = \sum_{k=1}^{\text{|U-WP_i|}} Z_k(U-WP_i, S-WP_i)
\]

where (1) \(Z_k = (U-WP_i, S-WP_i) = |weight(upos(WP_k), |U-WP_i|| \times upos(WP_k) - weight(xpos(WP_k), |S-WP_i|) \times xpos(WP_k)|\), (2) \(WP_k\) is the \(k\)-st Web page of \(U-WP_i\) and (3) \(weight\) is a function that associates a weight with each position; it is defined as \(weight(i, \text{dim}) = \left(\frac{\text{dim} - i}{\text{dim}}\right)^2\).

According to [42], the Newell Distance is a valid distance; moreover, the lesser the Newell Distance is, the better our system works. In this paper, we considered a “normalized” version of Newell Distance; it is defined as:

\[
N_i^{\text{norm}}(U-WP_i, S-WP_i) = \frac{N_i(U-WP_i, S-WP_i)}{\max_{1 \leq k \leq |USet|} N_i(U-WP_i, S-WP_i)}
\]

Clearly, \(0 \leq N_i^{\text{norm}}(U-WP_i, S-WP_i) \leq 1\), for all \(i\) such that \(1 \leq i \leq |USet|\).
7.2 Results of our experimental tests

Our tests were devoted to determine the accuracy of our system in proposing to a user the next Web pages to visit. In the following sections we report the results we have obtained w.r.t. the evaluation measures.

7.3 Results for Precision, Recall, F-Measure and Overall

We have computed the Average Precision, the Average Recall, the Average F-Measure and the Average Overall obtained by our system w.r.t. the number of accesses of the user. The results we have obtained are shown in Figures 4, 5, 6 and 7, respectively.

From the analysis of these figures, it is possible to observe that, in domains characterized by a low specialization level (“E-Shop” and “Books”), our system rapidly (i.e., after a few number of accesses) obtains good results; however, this performance slightly worsens when the number of Web accesses increases. In domains with a medium specialization level, our system takes more time to obtain good results; however, this performance does not worsen when the number of Web accesses increases. Finally, in domain characterized by a high specialization level, our system needs several Web accesses for achieving good results but, after this initial phase it maintains, and even improves, its performance.

This behaviour can be motivated by the following reasoning: during the first Web accesses, users interested in general domains do not have a precise idea about their needs and, consequently, many of the provided recommendations appear interesting to them. Vice versa, users interested in specific domains have a precise idea of their desires already during the initial phase; as a consequence, it is more difficult to satisfy them immediately. When the number of Web accesses increases, users of general domains “ripen” a more precise idea of their needs, but the application domain they are operating in is too generic to precisely satisfy their requirements; however, Figures 4, 5, 6 and 7 show that, even in these conditions, our system achieves quite good results. Vice versa, after many Web accesses, the profiles of the users of specific domains are quite rich and this allows our system to precisely identify their needs and to significantly reduce the search space.

7.4 Results for MAE and RMSE

We have computed MAE and RMSE for all the six domains considered versus the number of accesses. The most relevant results are shown in Figure 8. From
The performances of our system are very promising even if MAE and RMSE are never equal to 0. The main reason justifying the presence of (quite a little) MAE or RMSE is the “uncertainty” associated with coefficients composing user profiles, i.e., the fact that user profiles do not perfectly model the behaviour of real users.

The performances of our system w.r.t. the specialization level of the domain and the number of accesses confirm the results obtained in the previous
7.5 Results for Newell Distance

In Figure 9 we have plotted the percentage of users having a (normalized) Newell Distance belonging to the following intervals: (i) $[0,0.25)$, (ii) $[0.25, 0.5)$, (iii) $[0.5, 0.75]$ and (iv) $[0.75,1]$. Such values are the average of the (nor-
<table>
<thead>
<tr>
<th>Domain</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-Shop</td>
<td>0.15</td>
<td>0.11</td>
</tr>
<tr>
<td>Books</td>
<td>0.16</td>
<td>0.11</td>
</tr>
<tr>
<td>Drinks</td>
<td>0.18</td>
<td>0.15</td>
</tr>
<tr>
<td>Comp. Sc. Books</td>
<td>0.16</td>
<td>0.13</td>
</tr>
<tr>
<td>Wine</td>
<td>0.24</td>
<td>0.20</td>
</tr>
<tr>
<td>Network Books</td>
<td>0.28</td>
<td>0.31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Domain</th>
<th>MAE</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-Shop</td>
<td>0.25</td>
<td>0.20</td>
</tr>
<tr>
<td>Books</td>
<td>0.24</td>
<td>0.19</td>
</tr>
<tr>
<td>Drinks</td>
<td>0.15</td>
<td>0.11</td>
</tr>
<tr>
<td>Comp. Sc. Books</td>
<td>0.14</td>
<td>0.10</td>
</tr>
<tr>
<td>Wine</td>
<td>0.10</td>
<td>0.12</td>
</tr>
<tr>
<td>Network Books</td>
<td>0.15</td>
<td>0.13</td>
</tr>
</tbody>
</table>

Fig. 8. Values of $MAE$ and $RMSE$ after 20 and 80 user’s accesses, respectively

<table>
<thead>
<tr>
<th>Percentage of users</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
</tr>
<tr>
<td>0.05</td>
</tr>
<tr>
<td>0.10</td>
</tr>
<tr>
<td>0.15</td>
</tr>
<tr>
<td>0.20</td>
</tr>
<tr>
<td>0.25</td>
</tr>
<tr>
<td>0.30</td>
</tr>
<tr>
<td>0.35</td>
</tr>
<tr>
<td>0.40</td>
</tr>
<tr>
<td>0.45</td>
</tr>
<tr>
<td>0.50</td>
</tr>
</tbody>
</table>

Fig. 9. Distribution of users according to their Newell Distance

normalized) Newell Distance measured by considering 20, 40, 60, 80 and 100 user’s accesses and the different domains. From the analysis of this figure, we can conclude that the percentage of users having a small normalized Newell Distance (i.e. less than 0.5) is quite high (about 75%).

Such a result is a further confirmation that our system is particularly capable of both “understanding” user desires and adapting its behaviour to them.

8 Related Work

In this section we compare our approach with other ones already presented in the literature; such a comparison will be carried by taking into account different points of view. Specifically, in Section 8.1, we compare our approach with other ones conceived to handle the interschema property extraction task. Section 8.2 is devoted to highlight similarities and differences existing among our approach and some schema integration prototypes. Finally, in Section 8.4 we examine some systems conceived for supporting user navigation on the Web and make a comparison among them and our system.
8.1 Comparison with interschema property extraction approaches

Cupid In [33] Cupid, a system for deriving interschema properties among heterogeneous information sources, is presented. Property derivation is performed by carrying out two kinds of examinations, named linguistic and structure matchings. Some differences can be detected between Cupid and our approach. In particular, Cupid only derives interschema properties; in addition since the activities Cupid performs for extracting properties are numerous and sophisticated, the results it obtains could be more refined than those returned by our approach but the required time and user intervention are greater.

Kang and Naughton [29] In [29] the authors propose a technique, based on extensional data, for extracting interschema properties. Beside some similarities of this system with our proposal, (capability of capturing synonymies and homonimies and exploiting of graph matching techniques to derive interschema properties), a number of relevant differences exist. Indeed the applicability of [29] is restricted to flat (extensional data) tables whereas our approach manages generic data sources relying on intensional information.

SKAT [35] exploits first-order logic in order to express match and mismatch relationships as well as to derive new matches. The main difference with our system concerns the specification of relationships between concepts: since they can be represented by first-order logic, a heavy intervention of human experts is necessary. This is not the case of our model.

LSD (Learning Source Description) [18] is a machine learning approach, that has been also extended to ontologies in GLUE [19]. There is an important aspect which locates this system far from our proposal (even if they are related w.r.t. their purpose), that is LSD requires, during the initial phase, quite a heavy support of the user for carrying out training tasks.

He and Chang [26] is a statistical framework for performing schema matching. We observe that the approach of [26] creates a hidden Schema, which is capable of fully describing a domain; however, as claimed by the authors, its complexity is exponential and, consequently, is tolerable only if interschema property extraction activity is carried out off-line.

COMA (COmbining MAtch) is an interactive and iterative system for combining various interschema property extraction approaches, is proposed. The approach of COMA appears orthogonal to ours so that some idea of COMA could be incorporated in our model in order to improve its performances (like, for instance, the idea of operating iteratively). A relevant difference with COMA is that therein the user must specify the matching strategy (i.e., the algorithms to be used to detect semantic similarities and the modalities for combining their results).
8.2 Comparison with Schema integration approaches

In Passi et al. [40] an XML Schema integration framework, based on the extraction of interschema properties, is proposed. The approach exploits an object-oriented data model called XSDM (XML Schema Data Model) and is specific for XML Schema whereas our approach is capable of handling generic data sources.

In Yang et al. [57] an approach for the integration of XML data sources, based on the derivation of interschema properties, is proposed. Such an approach is computationally more expensive than our one, and is capable of managing only XML schemas.

XClust [31] is a system for XML data source integration. More specifically, it determines the similarity degrees of a group of DTD’s by considering not only the corresponding linguistic and structural information but also their semantics, derived by examining the neighborhoods of their elements. The main difference with respect to our approach is that XClust requires the support of a hierarchical clustering algorithm to perform the integration activity. In addition, XClust has been specifically conceived for operating on XML data sources.

Rondo [34] exploits a graph-based approach for modeling information sources and the Similarity Flooding Algorithm for performing schema matching. The techniques used are very sophisticated and, as a consequence, it obtains refined results but is time expensive and requires a heavy human feedback.

In R. dos Santos Mello et al. [20] an XML-based integration approach, capable of handling various source formats, is presented. It operates on DTD’s and requires to translate them in an appropriate formalism called ORM/NIAM [25]. Its main drawback is that it is quite complex to be applied when involved sources are numerous.

DIXSE [44] aims at supporting the integration of a set of XML documents. To this purpose it requires the support of the user, so that when the number of sources to integrate is high, the effort DIXSE requires to the user might be particularly heavy. In addition, DIXSE operates only on XML schemas.

In Castano et al. [13] an approach for carrying out the integration of XML sources with the support of interschema properties is proposed. We can observe that the intervention of the human expert it requires is heavier than that needed by our proposal and that it is specialized for operating on XML data sources.

8.3 Comparison with Web navigation support systems

Suggest [49] supports user navigation by dynamically generating links to pages that have not yet been visited by a user and might be potentially inter-
testing for him. In order to carry out its task, it builds and maintains historical information about user behaviour by means of an incremental graph partitioning algorithm. As far as the differences between Suggest and our systems are concerned, we point out that Suggest stores information about the user navigation in an undirected graph whose nodes represent Web page identifiers and whose arcs represent Web page relationships; vice versa, our approach exploits a user profile consisting of a graph whose nodes are associated with user interests and whose arcs represent relationships among interests; in addition Suggest performs its recommendation activity by means of a graph-partitioning algorithm; vice versa, our algorithm derives its recommendations by computing the semantic similarity between available Web pages and the user profile.

**DESIRE [39]** is a content-based Recommender System that combines a viewing time- and attribute-based preference inference algorithm with an attribute-based recommendation engine. In order to make its recommendations, DESIRE exploits only the current user navigational data, in conjunction with item property data. There are also some differences between our approach and DESIRE; more specifically, the user profile exploited by DESIRE is an item collection whereas that handled by our approach is a graph, and the algorithms that the two systems exploit for deriving their suggestions are very different.

**Alexa [1]** is a commercial system conceived for supporting Web navigation. It provides, for each site, its owner, its ratings, its reviews and its statistics. In order to derive ratings and reviews, Alexa requires the support of the users. Alexa and our approach are characterized by deep differences; (i) Alexa is utility-based whereas our approach shows a content-based behaviour; (ii) Alexa does not handle a user profile; (iii) the recommendation algorithms exploited by the two systems are completely different; (iv) Alexa is obtrusive whereas our approach is unobtrusive.

**Community Search Assistant [23]** is a system conceived for supporting users search engines to find information related to that they are currently accessing. In order to carry out its task, Community Search Assistant allows the users in a community to realize collaborative searches. The system exploits a support graph whose nodes represent queries submitted by the community users and whose arcs denote links between queries. In this way, the single user exploitation of information networks is transformed into a collaborative usage of it, since users can tap into the knowledge base of queries submitted by others. As far as the differences with our system are concerned, we observe that: (i) Community Search Assistant shows a collaborative filtering behaviour whereas our approach is content-based; (ii) the user profile exploited by Community Search Assistant stores user queries; vice versa the profile exploited by our approach stores user interests; (iii) the algorithms the two systems adopt for deriving recommendations are completely different.

The system **MEMOIR [46]** requires each user to manually enter URL trials and exploits this information for constructing the user profile. After this, it

31
monitors the browsing activity for each user and tries to find a correlation between a Web page he is currently accessing and a URL trial he previously provided. The main differences w.r.t. our approach are: that MEMOIR is slightly obtrusive whereas our approach is unobtrusive and that in MEMOIR the user profile stores URLs whereas in our approach it stores user interests. **METIOREW** [10] is an “objective-oriented” agent-based system and an objective represents an information need. Initially a user inserts his objectives by providing a list of keywords for each of them and the system constructs his profile as the set of his objectives. After this, the system associates two models with the new user, namely his own model and the most similar one already present in its knowledge base; this latter is associated with the new user until his own model becomes significant. The system provides its recommendations to a user on the basis of both his profile and the support one (if the user is new). As for differences between METIOREW and our approach, we observe the following: (i) our approach is content-based opposite to the collaborative filtering behaviour of METIOREW; (ii) our profile is graph-based whereas in [10] profiles are just collections of objectives; finally (iii) METIOREW is obtrusive whereas our approach is unobtrusive.

Similar consideration can be done for **SOAP** [56], a multi-agent system where a user can submit queries to an agent that calls a search engine for deriving Web pages answering these queries; these pages are proposed to the user who can rate them by means of both a 5-point scale and free-text annotations. User ratings are exploited by the system to filter known URLs and to provide its recommendations. Since annotations and ratings of each user are shared, any other user can inspect them.

**NAKIF** [21] acts as a content-based Recommender System. It associates a profile with each information item and defines a profile for each user. Then, it activates the matching algorithm for ranking the information items on the basis of their capability to match user preferences. This ordered list of items is supplied to the user who is required to provide an evaluation; the system exploits this evaluation as a feedback for updating the user profile. The main difference between NAKIF and our system is that the former is obtrusive whereas the latter is unobtrusive.

**Conceptual Graphs** [41] exploits a connected, bipartite graph characterized by two typologies of nodes, namely concept and relation nodes, representing both concepts and relationships among concepts. Such a graph is exploited for representing and handling a user profile that stores both the user content knowledge and his item ratings. The system provides its recommendations by comparing the graphs associated with the various user profiles. As for the differences between the Conceptual Graphs approach and ours, we observe that the former is obtrusive and shows a collaborative filtering behaviour whereas the latter is is unobtrusive and content-based.
8.4 Comparison with Semantic Web-Oriented Approaches

**OML (Ontology Markup Language) [2]** is an ontology specification language based on Conceptual Graphs [15]. It allows the representation of concepts organized in taxonomies, relations and axioms in first order logic.

**CKML (Conceptual Knowledge Markup Language) [14]** is an extension of OML, by the addition of theories, theory morphisms (which model concrete conceptual scales), and infomorphisms (which model realized conceptual scales). While OML defines the types for nouns and verbs, the theories in CKML offer a specification form for the modifiers, adjectives and adverbs. Moreover, the theories in CKML offer a means for the specification of ”controlled vocabularies.” Finally, CKML introduces the distinction between a theory (abstract conceptual scale) and a theory interpretation (concrete conceptual scale) that corresponds to the distinction between a terminological ontology and an axiomatic ontology.

**DAML+OIL [16]** provides modelling primitives commonly found in frame-based languages (such as an asserted subsumption hierarchy and the description or definition of classes).

**OWL (Ontology Web Language) [3]** has been conceived for defining and instantiating Web ontologies. An OWL ontology may include descriptions of classes, properties and their instances, and the OWL semantics specifies how to derive logical consequences from a given ontology.

**Semantic Web Rule Language (SWRL) [51]** is a combination of two OWL sub-languages with the Unary/Binary Datalog RuleML sub-languages of the Rule Markup Language. This approach extends the set of OWL axioms to include Horn-like rules, that model causal implications. Thus, in SWRL it is possible to combine Horn-like rules with an OWL knowledge base.

Similarly to these approaches, the c-graph model is able of representing semi-structured schemas, therefore it is suitable for representing the agent’s interests in an agent ontology. However, differently from the above models, the c-graph model does not use logic framework for representing agent behaviour, but it exploits hit and time coefficient for modelling the Web navigation actions. It is worth pointing out that the above models are conceived for modelling general-purpose agent ontologies, while our approach is specific for modelling Web navigation, thus it provides the advantage, with respect to the use of logic programs, of a more simple but very suitable representation of navigation actions.
9 Conclusions

This paper deals with the problem of modeling and supporting Web users by means of personal profiles. Our approach is based on a semantic representation of the user activity, which takes into account both the structure of visited sites and the way the user navigates them. The model appears interesting and exploitable in many application context, mainly because of its capability of dealing with semantic inconsistency occurring in the Web. This is shown by examples and experiments. In particular, a number of experiments performed by running a prototype validate our system on the basis of the customer satisfaction. The conclusion we may draw from this study is that taking account semantic heterogeneities leads to more effective results, since information occurring in actual sources are typically affected by such a phenomenon. This work gives thus the meaningful result that increasing the synergy between fields of Recommender Systems and Cooperative Information Systems might yields interesting issues and results. Among these, it would be interesting to investigate whether further semantic heterogeneities (beyond synonyms and homonyms) could be exploited in order to improve the effectiveness of Recommender Systems. Another interesting issue arising from our study is combining clustering techniques with our approach in order to obtain categories of both users and information sources. Finally, c-graphs may be used also for detecting useful cooperation among users on the basis of their closeness about interests and behavior.

References


38