A Relation-Based Page Rank Algorithm for Semantic Web Search Engines

Fabrizio Lamberti, *Member*, *IEEE*, Andrea Sanna, and Claudio Demartini, *Member*, *IEEE*

Abstract—With the tremendous growth of information available to end users through the Web, search engines come to play ever a more critical role. Nevertheless, because of their general-purpose approach, it is always less uncommon that obtained result sets provide a burden of useless pages. The next-generation Web architecture, represented by the Semantic Web, provides the layered architecture possibly allowing overcoming this limitation. Several search engines have been proposed, which allow increasing information retrieval accuracy by exploiting a key content of Semantic Web resources, that is, relations. However, in order to rank results, most of the existing solutions need to work on the whole annotated knowledge base. In this paper, we propose a relation-based page rank algorithm to be used in conjunction with Semantic Web search engines that simply relies on information that could be extracted from user queries and on annotated resources. Relevance is measured as the probability that a retrieved resource actually contains those relations whose existence was assumed by the user at the time of query definition.

Index Terms—Semantic Web, knowledge retrieval, search process, query formulation.

1 Introduction

In the last years, with the massive growth of the Web, we assisted to an explosion of information accessible to Internet users. Nevertheless, at the same time, it has become ever more critical for end users to explore this huge repository and find needed resources by simply following the hyperlink network as foreseen by Berners-Lee and Fischetti in 1999 [4]. Today, search engines constitute the most helpful tools for organizing information and extracting knowledge from the Web [9]. However, it is not uncommon that even the most renowned search engines return result sets including many pages that are definitely useless for the user [18]. This is mainly due to the fact that the very basic relevance criterions underlying their information retrieval strategies rely on the presence of query keywords within the returned pages. It is worth observing that statistical algorithms are applied to "tune" the result and, more importantly, approaches based on the concept of relevance feedback are used in order to maximize the satisfaction of user's needs. Nevertheless, in some cases, this does not suffice.

In order to show this odd effect, let us see what happens when a user enters a query composed by the following keywords "hotel," "Rome," and "historical center" (or "hotel," "Roma," and "centro storico") in the Italian version of the well-known Google search engine.[1] He or she would not be astonished probably by finding that the result set actually includes several hotels located in the historical center of Rome, as expected. Another hotel located in a small town at some distance from the Rome city center is also included. However, two hotels located in the historical center of other main Italian cities are also displayed. Finally, three hotels named Roma are included among the 10 most relevant results even if they have nothing to do with the selected city. Only 4 out the 10 results presented to the user satisfy user needs (even if they seem to satisfy the user query, based on the strategy adopted to process it). There is no doubt that the user would be able to easily decide which results are really of interest by looking, for example, at the two-line excerpt of the Web page presented in the displayed list or by quickly examining each page. Anyway, the presence of unwanted pages in the result set would force him or her to perform a postprocessing on retrieved information to discard unneeded ones. Even though several automatic techniques have been recently proposed [32], result refinement remains a time-waste and click-expensive process, which is even more critical when the result set has to be processed by automatic software agents. Let us try to analyze more in detail the reason why "out-of-scope" pages are inserted in the result set. When the user entered the query "hotel," "Rome," and "historical center," he or she was assuming the existence of some relations among those terms, such as "hotel" *located in the* "historical center" *of* "Rome." However, when the query was sent to the search engine logic, these hidden details were lost. The search logic usually tries to recover this information by exploiting many text-matching techniques (such as the number of occurrences and distance among terms). Nevertheless, traditional search engines do not have the necessary infrastructure for exploiting relation-based information that belongs to the semantic annotations for a Web page.

The Semantic Web [5] will offer the way for solving this problem at the architecture level. In fact, in the Semantic Web, each page possesses semantic metadata that record additional details concerning the Web page itself. Annotations are based on classes of concepts and relations among them. The "vocabulary" for the annotation is usually expressed by means of an ontology that

1. Discussion based on result set obtained on 4 August 2007.

• *The authors are with the Dipartimento di Automatica ed Informatica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129 Torino, Italy. E-mail: {lamberti, sanna, demartini}@polito.it.*

Manuscript received 9 Aug. 2007; revised 26 Feb. 2008; accepted 12 May 2008; published online 2 June 2008.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-08-0412. Digital Object Identifier no. 10.1109/TKDE.2008.113.

provides a common understanding of terms within a given domain.

In this paper, we will prove that relations among concepts embedded into semantic annotations can be effectively exploited to define a *ranking strategy* for Semantic Web search engines. This sort of ranking behaves at an inner level (that is, it exploits more precise information that can be made available within a Web page) and can be used in conjunction with other established ranking strategies to further improve the accuracy of query results. With respect to other ranking strategies for the Semantic Web, our approach only relies on the knowledge of the user query, the Web pages to be ranked, and the underlying ontology. Thus, it allows us to effectively manage the search space and to reduce the complexity associated with the ranking task.

The organization of this paper is given as follows: In Section 2, we provide an overview of existing strategies for Semantic Web search. In Section 3, the basic idea behind the proposed approach is presented by resorting to practical examples, while in Section 4, a formal methodology for deriving the general rule is illustrated. In Section 5, details concerning the implementation are provided. An analysis of the algorithm complexity is given in Section 6, while experimental results are discussed in Section 7.

2 Related Works in Semantic Web Search

The aim of this paper is to show how to make use of relations in Semantic Web page annotations with the aim of generating an ordered result set, where pages that best fit the user query are displayed first. The idea of exploiting ontology-based annotations for information retrieval is not new [7], [8], [11], [25]. Nevertheless, these first works did not focus on semantic relations, which are considered (and expected) to play a key role in the Semantic Web [16], [24]. In fact, it has been recently outlined that in order to fully benefit on semantic contents, a way for achieving relation-based ranking has to be found [2], [16], [19], [26].

One of the first attempts to enhance Semantic Web search engines with ranking capabilities is reported in [19]. The authors define a similarity score measuring the distance between the systematic descriptions of both query and retrieved resources. They first explode an initial set of relations (properties) by adding hidden relations, which can be inferred from the query. Similarity is then computed as the ratio between relation instances linking concepts specified in the user query and actual multiplicities of relation instances in the semantic knowledge base. This method is applied on each property individually and requires exploring all the Semantic Web instances. Moreover, the user is requested to specify all the relations of interest. Thus, since it is predictable that the number of relations will largely exceed the number of concepts [1], its applicability in real contexts is severely compromised. A similar approach, aimed at measuring the relevance of a semantic association (that is, a path traversing several concepts linked by semantic relations) is illustrated in [26]. The authors provide an interesting definition of relevance as the reciprocal of the ambiguity of the association itself. However, this approach suffers from the same limitations of [19], since queries have to be specified by entering both concepts and relations, and ambiguity is measured over each relation instance.

Nevertheless, the idea of exploring the set of relations that are implicit in the user's mind (but which are not made explicit in defining the query) has been pursued in many works. In [18], ontology-based lexical relations like synonyms, antonyms, and homonyms between keywords (but not concepts) have been used to "expand" query results. In this case, search is targeted to the Web, rather than to the Semantic Web. In [27], a similar approach has been integrated into artificial intelligence methodologies to address the problem of query answering. In [3], query logs are used to construct a user profile to be later used to improve the accuracy of Web search. Semantic Web search from the point of view of the user's intent has been addressed also in [15] and [28], where the authors present two methodologies for capturing the user's information need by trying to formalize its mental model. They analyze keywords provided during query definition, automatically associate related concepts, and exploit the semantic knowledge base to automatically formulate formal queries.

A slightly different methodology has been exploited in SemRank [2]. Here, the basic idea is still to rank results based on how predictable a result might be for the user but based on how much information is conveyed by a result, thereby giving a sense of how much information a user would gain by being informed about the existence of the result itself. To achieve their goal, the authors define two measures, named "uniqueness" and "discrepancy," which allow accounting for the specificity or deviation of a particular result with respect to instances stored in the database. An additional added value of SemRank is that in the computation of the ranking, it exploits a so-called "modulative relevance model" that is capable of taking into account the particular context/purpose in/for which a query has been submitted (conventional or discovery search). Even if the authors do not provide any analysis of the computational cost of their approach, it is reasonable to infer that since to rank a single page information related to the annotations of all the remaining pages is needed, the performance of the proposed solution would hardly scale for huge Semantic Web environments.

An approach also based on the context and partially solving the problems above is taken in [26]. Here, the context (defined in this case as a subset of concepts and relations of the whole Semantic Web environment) of interest to the user, rather than specific concepts or relations, can be specified together with the query using an ad hoc language. The authors assign "universal" and "user-defined" weights to each semantic relation/association, taking into account the context, as well as other parameters like specificity and path length. These weights are combined into a global formula where multiplying constants are specified by the user (or by an expert) and are strictly query dependent. Thus, in order to get accurate results, an intensive manual configuration step has to be performed, and this step cannot be valid for answering heterogeneous queries. A solution capable of partially overcoming the limitations above is presented in [22]. Here, a strategy for clustering concepts based on query keywords provided by the user is proposed. A spread activation process is applied to navigate the whole relation set and discover related nodes that could be of interest. This process is only partially

automated since it has to be guided by the information on knowledge provided by a domain expert.

A totally different solution is represented by OntoLook [16]. The basic idea is that if a graph-based representation of a Web page annotation can be provided, where concepts and relations (together with their multiplicities) are modeled as vertices and weighted edges, respectively, it becomes possible to define a series of cuts removing less relevant concepts from the graph. This allows for the generation of a so-called candidate relation-keyword set (CRKS) to be submitted to the annotated database, which can significantly reduce the presence of uninteresting pages in the result set. It is worth observing that the strategy behind OntoLook only allows us to empirically identify relations among concepts that should be less relevant with respect to the user query. This information is used to reformulate the user query by including only a subset of all the possible relations among concepts, which is later used to retrieve web pages from the annotated database. The user is not requested to specify relations of interest during query definition. However, the effectiveness of the approach is strongly limited by the fact that there does not exist any ranking strategy. Even if the authors claim that any of the existing page ranking algorithms can be used to order the obtained result set, it is worth remarking that this is not completely true. In fact, a ranking strategy like the PageRank [13], [17] used by Google [6] is only one of the ranking algorithms used to organize results to be displayed to the user. Many other statistical and text-matching techniques are used together with PageRank. Of course, PageRank can be used in conjunction with [16] to exploit relevance feedback and postprocess the result set. But the use of the remaining techniques is not feasible since they cannot be reasonably applied into a concept-relation-based framework where ontology is predominant on pure text. The authors themselves state that what is really needed is a relation-based page rank algorithm.

Our work moves from the considerations above and relies on the assumption that for providing effective ranking, the search engine logic should only need to know the structure of the underlying ontology and of the Web page to be ranked in order to compute the corresponding relevance score. In this way, effective performance can be achieved in heterogeneous real frameworks. It is worth observing that the proposed approach could be easily seen as an extension of [16]. Moreover, it does not represent an alternative to any of the approaches above, but rather, they can be regarded as complementary to our solution (and vice versa). For instance, the availability of an ad hoc language allowing the user to preprocess the graph and reduce the region of interest [26] could be integrated in our approach as a preprocessing step. Similarly, the availability of instruments for inferring concepts of interest starting from a pure keyword-based query [22] can be helpful to limit the amount of knowledge of the underlying ontology requested to the user. Finally, the proposed technique is not intended to replace the ranking strategies of actual search engines. In fact, it relies on relevance information that is totally different from that exploited, for example, in algorithms like SemRank, Pagerank, and others. Rather, it should be understood as a preprocessing step to produce a semantic-aware ordered result set to be later (or simultaneously) treated with existing (popular) techniques in order to come to an increased hit ratio in user query processing.

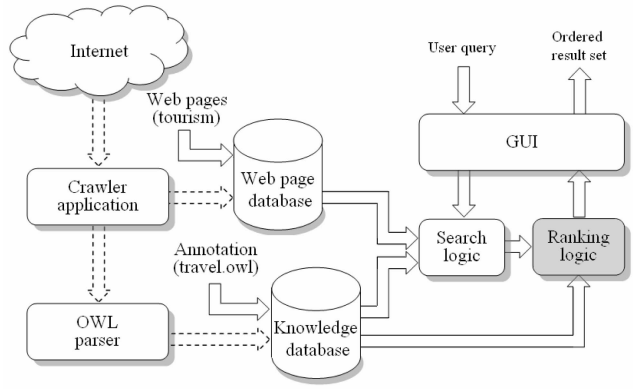

Fig. 1. Semantic Web infrastructure (prototype architecture).

3 Overview of the Ranking Strategy

In this section, the basic idea behind our ranking strategy is discussed. In order to introduce the readers to its formalism and let them foresee its applicability in real scenarios, the overall architecture of a prototypal search environment developed in our laboratory is presented first.

3.1 Prototype of a Relation-Based Search Engine

To evaluate the feasibility of the proposed approach, we first constructed a controlled Semantic Web environment. To do this, we selected the well-known *travel.owl* ontology [20] written in the OWL language [29], and we modified it by adding new relations in order to make it more suitable for demonstrating system functionality. We then created a knowledge base by either downloading or automatically generating a set of web pages in the field of tourism, and we embedded into them RDF [21] semantic annotations based on the ontology above. Finally, we designed the remaining modules of the architecture, including a Web page database, a crawler application, a knowledge database, an OWL parser (OwlDotNetApi), a query interface, and the true search engine module embedding the proposed ranking logic (Fig. 1). The crawler application collects annotated Web pages from the Semantic Web (in this case, represented by the controlled environment and its Web page collection) including RDF metadata and originating OWL ontology. RDF metadata are interpreted by the OWL parser and stored in the knowledge database. A graphics user interface allows for the definition of a query, which is passed on to the relation-based search logic. The ordered result set generated by this latter module is finally presented to the user. The details of the system workflow will be provided in the following sections, starting with the query definition process, since it was through the analysis of its dynamics that we came to the identification of our ranking strategy.

3.2 Starting Point: The Query Definition Process

In a traditional search engine like Google [6], a query is specified by giving a set of keywords, possibly linked through logic operators and enriched with additional constraints (i.e., document type, language, etc.). On the

other hand, semantic search engines are capable of exploiting concepts (and relations) hidden behind each keyword together with natural language interpretation techniques to further refine the result set. The core step that consists of identifying the mapping between keywords and concepts can be performed in a (semi)automated way [15], [22], [23], [28]. Otherwise, in order to avoid ambiguities, the user can be requested, during query definition, to specify the concept a keyword refers to [2], [16], [19], [26]. Given the fact that the query interpretation step is out of the scope of this paper, the proposed methodology relies on the second approach. That is, like in [16], the user specifies a query by entering a keyword and selecting a concept from a pull-down menu containing ontology classes of the *travel.owl* ontology organized in a hierarchical fashion.

It is worth observing that the current implementation is not able to handle multiple ontologies describing the same domain. From the point of view of the search logic, this would require the integration of one of the existing techniques for mapping or merging/translating the heterogeneous ontologies [10], which would result in the definition of a set of mapping rules or in the creation of a novel (possibly extended) ontology, respectively. From the point of view of user interaction, having an extended ontology would increase the need for a preprocessing step enabling automatic identification of keyword-concept pairs. On the other hand, mapping rules would have to be only implemented in the search logic; the user could continue to use the same interface (possibly allowing to choose the ontology best suited for the query).

3.3 Introduction to Relation-Based Ranking

Let us assume now that the user specifies the keyword "Rome," and he or she then selects from the pull-down menu one of the possible concepts such as *Destination* or *City*. A second keyword "hotel" is then added, choosing *Accomodation* as the associated concept. In general, there is no way to state which was the relation in the user's mind between those two concepts (even if in this case, it seems to be obvious). But what can be certainly said is that the user was assuming the existence of at least one relation between the two terms (and concepts as well) or between these terms and the following ones (if this is the case). Now, let us consider a set of annotated pages containing keywords "Rome" and "hotel" and associated concepts *Destination* and *Accomodation*. A traditional search engine like Google would return both pages without considering the information provided by the semantic mark. On the other hand, a semantic search engine would take into account keyword-concept associations and would return a page only if both keywords (or synonyms, homonyms, etc.) are present within the page and they are related to associated concepts. Finally, a relation-based search engine like the one presented in [16] would go beyond pure "keyword isolated" search and would include these pages in the result set only if there exist enough relations linking considered concepts. However, pages included in the result set would have the same "weight."

3.4 Basic Idea

Let us try to see if there is a way for presenting these pages in order of importance to the user. We continue with query definition, and we assume now that the user enters the last two keywords of his or her query, for example, "museums" and "historical center," associated to concepts *Activity* and *Destination*, respectively. Let us assume also that according to the ontology, these concepts are linked to both the previous concepts through a certain number of relations. There is again no way to infer either to which concept/s and by means of which relation/s the newly added concepts are related to. However, we can certainly say that since these are the last concepts, they should be related to each other or to at least one of the previously entered concepts. In general, what we can always say is that *each concept specified within the query should have to be characterized by relations with at least another concept*. This consideration can be of great help when trying to define a way for providing a ranking among semantic annotated pages. In fact, the larger is the number of relations linking each concept with each other concept given the total number of relations among those concepts in the ontology, the higher is the probability that this page contains exactly those relations that are of interest to the user and, as a consequence, that this page is actually the most relevant with respect to user query. Thus, the idea is to define a "ranking criterion" based on *an estimate of the probability that keywords/concepts within an annotated page are linked one to the other in a way that is the same (or at least that is similar) to the one in the user's mind at the time of query definition*. As will be shown in the following, this probability measure can be effectively computed by defining a graph-based description of the ontology (*ontology graph*), of the user query (*query subgraph*), and of each annotated page containing queried concepts/keywords (both in terms of *annotation graph* and *page subgraph*). In the following, the ontology graph, query subgraph, annotation graph, and page subgraph notions will be presented through the use of intuitive examples.

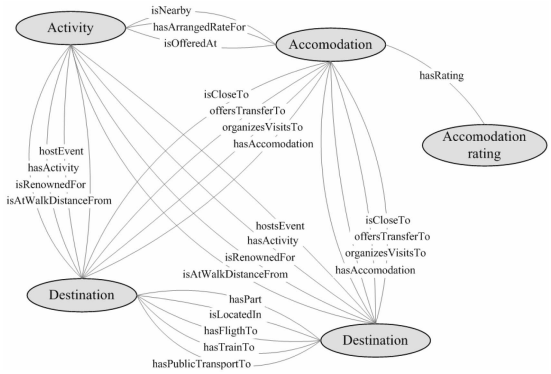

Fig. 2. A portion of the graph-based representation for travel.owl ontology (ontology graph).

3.5 Graph-Based Notation and Methodology

In the ontology and annotation graphs, concepts and relations are translated into graph nodes and edges, respectively. A portion of the ontology graph for the *travel.owl* ontology is reported in Fig. 2, while two examples of annotation graphs built upon as many annotated Web pages are shown in Fig. 3. It is worth observing that by taking into account the considerations in Section 3.4, a ranking for Web pages in Fig. 3 can be easily found: in fact, according to the query, the user was probably looking for a hotel located in the historical center of Rome and (maybe)

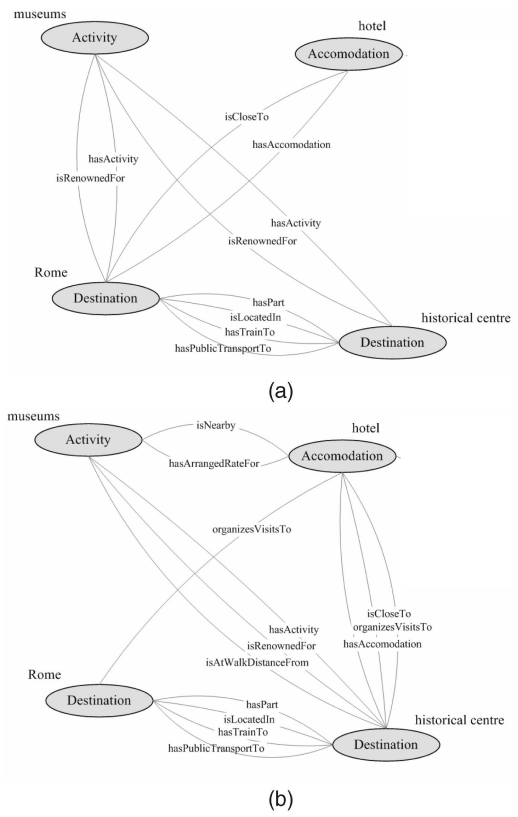

Fig. 3. Example of an annotation graph for two Web pages. (a) Activities, accommodations, and sightseeing places in Rome. (b) Hotel in the historical center of Rome, close to museums.

close to museums. However, even if this ranking can be proved intuitively by looking at the actual relations, a way for instructing the logic of the search engine is still needed. To do this, the notions of query subgraph and page subgraph have to be introduced.

In a query subgraph, nodes are represented by concepts that have been specified within the query. Nodes/concepts are linked by an (weighted) edge only if there exists at least one relation between those concepts in the ontology. The weight is represented by the actual number of relations. Similarly, a page subgraph is built based on the annotation associated to the page itself.

The methodology we propose in this paper starts from a page subgraph computed over an annotated page and generates all the possible combinations of the edges belonging to the subgraph itself not including cycles. Since there could exist pages in which there are concepts that do not show any relations with other concepts but that could still be of interest to the user, the methodology progressively reduces the number of edges in the page subgraph and computes the probability that each of the resulting subgraphs obtained by a combination of the remaining edges is the one that matches the user's intention. Edge removal could lead to having concepts without any relation with other concepts. Thus, several relevance classes are defined, each characterized by a certain number of connected concepts. Within each class, pages are ordered depending on the probability measure above and presented to the user.

4 Relation-Based Ranking Formal Model

In this section, a formal model for the proposed ranking strategy will be provided, by taking into account all the critical situations that could be envisioned.

4.1 Graph-Based Formalization

Starting from the ontology defined for a domain, a graph-based representation can be designed where OWL classes are mapped into graph vertices and OWL relation properties are mapped into graph edges. Thus, the existing relations between couples of concepts in the domain are depicted by means of connected vertices in the graph. We call it the *ontology graph* G. According to graph theory, the undirected graph G can be defined as $G(C, R)$, where $C = \{c_1, c_2, \ldots, c_n\}$ is the set of concepts that can be identified in the ontology, $|C| = n$ is the total number of concepts available, $R = \{R_{ij} | i = 1, \ldots, n, j = 1, \ldots, n, j > i\}$ is the set of edges in the graph, and more specifically, $R_{ij} = \{r_{ij}^1, r_{ij}^2, \ldots, r_{ij}^m, m < n\}$ is the set of edges between concepts i and j. An example of an ontology graph (based on the formal notation summarized in Table 1) is illustrated in Fig. 4a. Since queries are specified by the user by providing a collection of keywords and associated concepts, a single query can be formally expressed as $Q = \{(k_t, c_t)\}$.

Given a particular query containing a specific set of keywords related to a subset of ontology concepts, it is possible to construct a *query subgraph* G_Q. The query subgraph is an undirected weighted graph derived from G where vertices not belonging to C_Q are deleted. Moreover, in the query subgraph, vertices i and j are linked by an edge only if there exists at least one relation between the corresponding concepts in the ontology graph G. By referring to the same notation used for the ontology graph, G_Q can be expressed as $G_Q(C_Q, R_Q)$, where $C_Q = \{c_t | (k_t, c_t) \in Q\} \subseteq C$ is the subset of concepts mentioned in the query, $R_Q = \{\bar{R}_{ij} | 1 \leq i \leq n, 1 \leq j \leq n, j > i\}$, and $\bar{R}_{ij} = \{\bar{r}_{ij} | c_i, c_j \in C_Q, |R_{ij}| \geq 1\}$. Each edge $\bar{r}_{ij}$ in the query subgraph is assigned a weight η_{ij} that corresponds to the number of relations between concepts i and j in the ontology graph. Thus, it is $\eta_{ij} = |\bar{R}_{ij}|$. The query subgraph that can be obtained from the ontology graph in Fig. 4a for a query $Q = \{(k_1, c_1), (k_2, c_2), (k_3, c_3)\}$ is shown in Fig. 4b.

The aim of this paper is to demonstrate that, given an ontology graph G and a query subgraph G_Q, it is possible to define a ranking strategy capable of assigning each page including queried concepts a relevance score based on the semantic relations available among concepts within the page itself (thus neglecting the contribution of the remaining Web pages). The proposed ranking strategy assumes that given a query Q, for each page p, it is possible to build a *page subgraph* $G_{Q,p}$ using a methodology that is similar to the one used for G and G_Q and exploiting the information available in page annotation A. By expressing page annotation A as a graph, we have $A = (AC, AR)$, where AC and AR are the sets of annotated concepts and relations, respectively. Page subgraph $G_{Q,p}$ contains only those concepts included both in C_Q and in page annotation AC. Concerning graph edges, all the edges $\bar{r}_{ij}$ in R_Q are maintained, if the related concepts belong

TABLE 1
Definition of Symbols

Symbol	Definition
$G(C,R)$	Ontology graph
$C=\{c_1,c_2,\ldots,c_n\}$	Set of concepts constituting the vertices of the ontology graph
$R=\{R_{ij} \mid i=1,\ldots,n,\ j=1,\ldots,n,\ j>i\}$	Set of relations constituting the edges of the ontology graph
$R_{ij}=\{r_{ij}^1,r_{ij}^2,\ldots,r_{ij}^m,\ m<n\}$	Set of relations between concepts c_i and c_j in $G(C,R)$
$Q=\{(k_t,c_t)\}$	Query as a collection of pairs (keyword, concept)
$G_Q(C_Q,R_Q)$	Query subgraph for query Q
$C_Q=\{c_t \mid (k_t,c_t)\in Q\}$	Set of concepts constituting the query subgraph
$R_Q=\{\bar{R}_{ij} \mid 1\le i\le n,\ 1\le j\le n,\ j>i\}$	Set of relations constituting the query subgraph
$\bar{R}_{ij}=\{\bar{r}_{ij} \mid c_i,c_j\in C_Q, \lvert R_{ij}\rvert\ge 1\}$	Set of relations between c_i and c_j in the query subgraph
$\eta_{ij}=\lvert\bar{R}_{ij}\rvert$	Number of relations between c_i and c_j in $G_Q(C_Q,R_Q)$
$A=(AC,AR)$	Graph-based annotation (AC and AR are the sets of concepts and relations)
$AR_{ij}=\{r_{ij}^d \mid r_{ij}^d\in AR,\ 1\le d\le m\}$	Set of relations between c_i and c_j in AR
$G_{Q,p}(C_{Q,p},R_{Q,p})$	Page subgraph for page p given the query Q
$C_{Q,p}=\{c_t \mid c_t\in C_Q\cap AC\}$	Set of concept of page subgraph $G_{Q,p}(C_{Q,p},R_{Q,p})$
$R_{Q,p}=\{\bar{r}_{ij} \mid c_i,c_j\in G_{Q,p}\}$	Set of relations of page subgraph $G_{Q,p}(C_{Q,p},R_{Q,p})$
$\delta_{ij}=\lvert AR_{ij}\rvert$	Number of relations c_i and c_j in $G_Q(C_Q,R_Q)$
$\tau_{ij}=P(\bar{r}_{ij},p)=\delta_{ij}/\eta_{ij}$	Relation probability for $\bar{r}_{ij}$ in page p given the query Q
$SF_{Q,p}(l)$	Set of spanning forests (l edges) for page p and query Q
$SF_{Q,p}^f(l)$	f-th spanning forest (l edges) for page p and query Q
$\sigma_{Q,p}(l)=\lvert SF_{Q,p}(l)\rvert$	Number of spanning forests (l edges) for page p and query Q
$P(Q,p,l)$	Constrained relevance score for page p, query Q, relevance class l
$ps_{Q,p}$	Relevance score of page p for a given query Q

to $G_{Q,p}$. Weights η_{ij} specified for R_Q are inherited also by edges in $G_{Q,p}$. However, an additional weight δ_{ij} is associated to each edge to take into account the number of relations actually linking concepts i and j in the selected page (on the basis of the set of annotated relations, AR). According to the notation above, the page subgraph for page p can be defined as $G_{Q,p}(C_{Q,p},R_{Q,p})$, where $C_{Q,p}=\{c_t|c_t\in C_Q\cap AC\}$, and $R_{Q,p}=\{\bar{r}_{ij}|c_i,c_j\in G_{Q,p}\}$. We have also $\delta_{ij}=|AR_{ij}|$, where $AR_{ij}=\{r_{ij}^d|r_{ij}^d\in AR,1\le d\le m\}$.

4.2 Relevance and Semantic Relations

Let us pass now at considering how to apply the methodology above for the computation of a page relevance score. We start again by analyzing (now from

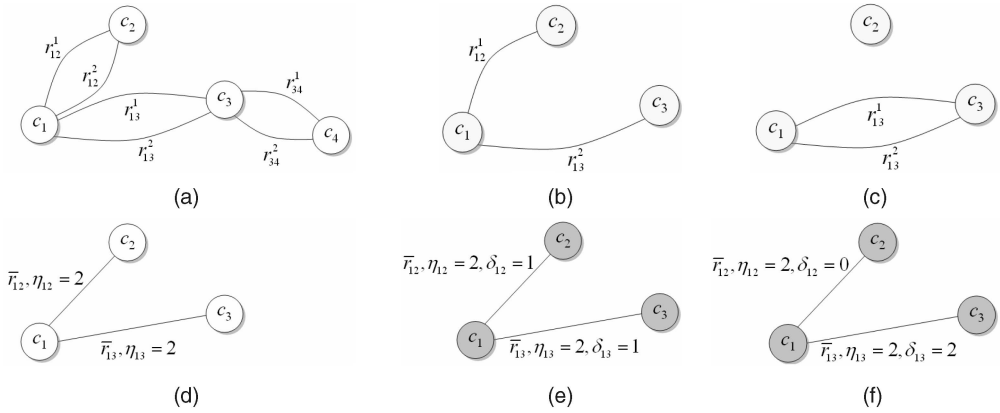

Fig. 4. (a) An ontology graph. (b) Query subgraph obtained for a given query specifying concepts c_1, c_2, and c_3. (c) and (d) A first example of page annotation p_1 and the related page subgraph. (e) and (f) A second example of page annotation p_2 and the related page subgraph.

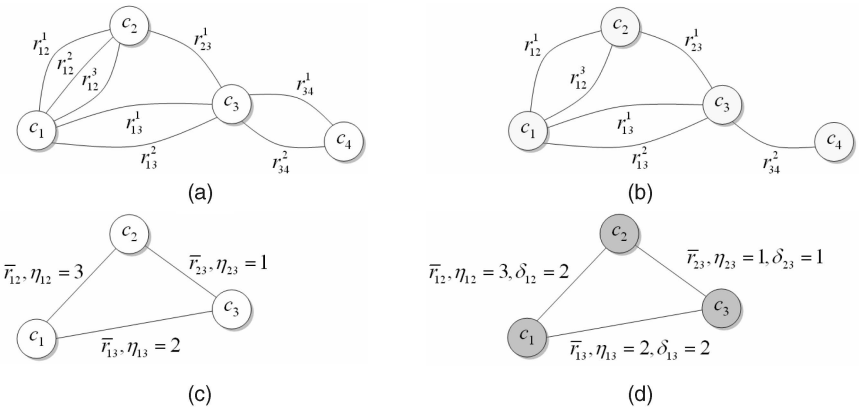

Fig. 5. (a) An ontology graph. (b) Query subgraph. (c) An example of an annotated page. (d) Page subgraph built upon the given ontology/query.

a formal point of view) the steps followed by a user during the process of query definition. Let us imagine that a user is interested in pages containing three generic keywords k_1, k_2, and k_3 (associated to as many generic concepts c_1, c_2, and c_3). The user begins query definition by specifying a pair including a keyword and its related concept. Let us assume that he or she starts with k_1 and c_1. It is reasonable to assume that after specifying keyword k_1, the user inserts a second keyword (for example, k_2, together with concept c_2) expecting either to find pages where k_1 and k_2 (that is, c_1 and c_2) are related in some way or to find pages where k_1 is linked to some other keywords/concepts that will be specified later. In a similar way, when he or she specifies k_3 and c_3, he or she would be expecting to further adjust the result set in order to find pages showing also relations between k_3 and k_1 (not k_2 since in the ontology, there is no relation linking c_3 with c_2). Let us consider a very trivial example assuming that there exists only two pages p_1 and p_2 containing all the keywords (and associated concepts) specified by the user. This represents the (initial) result set for the given query. We want to rank those pages in order to present to the user first the page that best fits his or her query. The semantic annotations and page subgraphs for these pages are illustrated in Figs. 4c, 4d, 4e, and 4f. In the first page, both c_2 and c_3 are linked to c_1 through a single relation (Fig. 4c), while in the second page there exists two relations linking c_3 to c_1. However, c_2 is not linked in any way to c_1 (Fig. 4f). Since we cannot assume which could be the concepts or the relations more important with respect to user query, we can provide a significant measure of page relevance by computing the probability that a page is the one of interest to the user (that is, its relevance) by calculating the probability that c_2 is linked to c_1 and c_3 is linked to c_1 through the relations in the user's mind (either r_{12}^1 or r_{12}^2 and r_{13}^1 or r_{13}^2, respectively). Let us compute $P(\bar{r}_{ij}, Q, p)$, which is the probability of finding in a particular page p a relation $\bar{r}_{ij}$ between concepts i and j that could be the one of interest to the user (because of query Q). According to the probability theory, this can be defined as $P(\bar{r}_{ij}, p) = \delta_{ij}/\eta_{ij} = \tau_{ij}$ (note that it does not depend on Q). We call it the *relation probability*. Thus, for the first page, we have $P(\bar{r}_{12}, p_1) = \delta_{12}/\eta_{12} = \tau_{12} = 1/2$ and $P(\bar{r}_{13}, p_1) = \delta_{13}/\eta_{13} = \tau_{13} = 1/2$. For the second page, we have $P(\bar{r}_{12}, p_2) = \delta_{12}/\eta_{12} = \tau_{12} = 0$ and $P(\bar{r}_{13}, p_2) = \delta_{13}/\eta_{13} = \tau_{13} = 1$. Based on the considerations above, we can compute the joint probability $P(Q, p) = P((\bar{r}_{12}, p) \cap (\bar{r}_{13}, p))$. The dependency on Q is due to the fact that only concepts given in Q are taken into account. Since the events $(\bar{r}_{12}, p)$ and $(\bar{r}_{13}, p)$ are not correlated, $P(Q, p)$ can be rewritten as $P(Q, p) = P(\bar{r}_{12}, p) \cdot P(\bar{r}_{13}, p)$. Thus, for the specific example being considered, it is $P(Q, p_1) = 1/4$ and $P(Q, p_2) = 0$, respectively, for the first and second page. This allows placing the first page before the second one in the ordered result set. However, to preserve the behavior of common search strategies, a way for assigning a score different than zero to pages in which there exists concepts not related to other concepts will have to be identified.

Another critical situation is illustrated in Fig. 5. In this case, the user specifies a query composed by concepts c_1, c_2, and c_3 over a novel ontology. Based on the considerations above, a measure of page relevance can be computed by estimating, for each concept, the probability of having a relation between that concept and another concept and that such relation is exactly the one in the user's mind. However, it can be demonstrated that this probability can be expressed also in different terms, capable of taking into account situations in which a particular concept can be related to more than one concept (that is, the case of the specific example being considered, as well as of common situations in any concrete search scenario). Specifically, the probability that each concept is related to other concepts is given by the probability of having c_1 linked to c_2 and c_2 linked to c_3 or c_1 linked to c_2 and c_1 linked to c_3 or c_2 linked to c_3 and c_1 linked to c_3. The situations above can be modeled again by using graph theory. In fact, having each concept related to at least another concept in the query is equivalent to considering all the possible spanning forests (a collection of spanning trees, one for each connected component in the graph) for page subgraph $G_{Q,p}$ given the query Q. In Fig. 6, all the possible spanning forests (trees, in this case) of the page subgraph in Fig. 5d are shown. We call $SF_{Q,p}^f$ the fth *page spanning forest* computed over $G_{Q,p}$. We define $P(SF_{Q,p}^f)$ as the probability that $SF_{Q,p}^f$ is the spanning forest of interest to the user. By simplifying the notation and

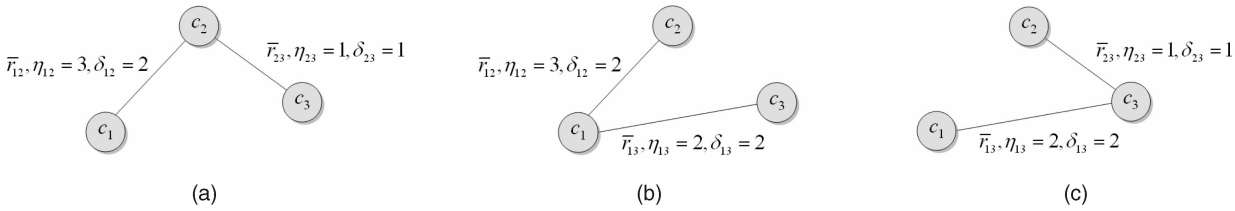

Fig. 6. All the possible spanning forests (trees) that could be obtained from $G_{Q,p}$ in Fig. 5d.

replacing $\bar{r}_{ij}$, p with $\bar{r}^p_{ij}$, the probability for page p can be computed as

$$P(Q,p) = P\Big(\Big((\bar{r}^p_{12} \cap \bar{r}^p_{23}) \cap SF^1_{Q,p}\Big) \cup \Big((\bar{r}^p_{12} \cap \bar{r}^p_{13}) \cap SF^2_{Q,p}\Big) \cup \Big((\bar{r}^p_{23} \cap \bar{r}^p_{13}) \cap SF^3_{Q,p}\Big)\Big). \tag{1}$$

Since the events are not correlated, it is also

$$\begin{aligned} P(Q,p) &= P(\bar{r}^p_{12} \cap \bar{r}^p_{23}) \cdot P\Big(SF^1_{Q,p}\Big) + P(\bar{r}^p_{12} \cap \bar{r}^p_{13}) \cdot P\Big(SF^2_{Q,p}\Big) \\ &\quad + P(\bar{r}^p_{23} \cap \bar{r}^p_{13}) \cdot P\Big(SF^3_{Q,p}\Big) \\ &= P(\bar{r}^p_{12}) \cdot P(\bar{r}^p_{23}) \cdot P\Big(SF^1_{Q,p}\Big) + P(\bar{r}^p_{12}) \cdot P(\bar{r}^p_{13}) \cdot P\Big(SF^2_{Q,p}\Big) \\ &\quad + P(\bar{r}^p_{23}) \cdot P(\bar{r}^p_{13}) \cdot P\Big(SF^3_{Q,p}\Big), \end{aligned} \tag{2}$$

where $P(\bar{r}_{ij,p})$ can be replaced with $\tau_{ij} = \delta_{ij}/\eta_{ij}$.

Since the probability for a single page spanning forest to be the one of interest to the user is the same with respect to the remaining ones, if we define $\sigma_{Q,p}$ as the number of spanning forests for $G_{Q,p}$, we have $P(SF^1_{Q,p}) = P(SF^2_{Q,p}) = P(SF^3_{Q,p}) = 1/\sigma_{Q,p}$. Thus, the expression for $P(Q,p)$ can be rewritten again as

$$P(Q,p) = \frac{P(\bar{r}^p_{12}) \cdot P(\bar{r}^p_{23}) + P(\bar{r}^p_{12}) \cdot P(\bar{r}^p_{13}) + P(\bar{r}^p_{23}) \cdot P(\bar{r}^p_{13})}{\sigma_{Q,p}}, \tag{3}$$

and according to the definition of relation probability, it is

$$P(Q,p) = [\tau_{12} \cdot \tau_{23} + \tau_{12} \cdot \tau_{13} + \tau_{23} \cdot \tau_{13}]/\sigma_{Q,p}. \tag{4}$$

Given the ontology and the query selected for the considered example, (4) can be used to compute a relevance score for each page in the result set and to provide a ranking within the result set itself. As expected, (4) works well also for the example in Fig. 4, where $\sigma_{Q,p} = 1$ (since the page subgraph already constitutes the only spanning forest). Nevertheless, $P(Q,p)$ can still assume a value equal of zero for all those pages in which there exists concepts that do not show any relation with other concepts but is still present, as a keyword, in the annotated page. In the following, we will analyze this issue in detail, and we will show how to extend the methodology above in order to come to a general rule for ranking all the pages in the (initial) result set.

We consider again an example represented by two pages (depicted in Fig. 7 and based on the same ontology as in Fig. 5a), where concept c_4 (in the first page) and concept c_2 (in the second page) do not show any relations with the remaining concepts. If we compute $P(Q,p_1)$ and $P(Q,p_2)$ using (4) (which is still valid since the page annotation refers to the same ontology), we get a relevance score equal to zero. Based on the definition of relevance score provided above, in order to find a score different than zero allowing each page to be ranked with respect to other pages, we have to *relax the condition of having each concept related to each other concept*. Since by definition, in a spanning forest, there does not exist any cycles, removing one edge means removing a link between a couple of concepts. That is, edges from all the page spanning forests have to be progressively removed, thus obtaining *constrained page spanning forests* composed by a decreasing number of edges (and, equivalently, of connected concepts). We maintain the term "spanning" in order to recall that each constrained page spanning forest originates from a true spanning forest in which for all the connected components of the graph, all the

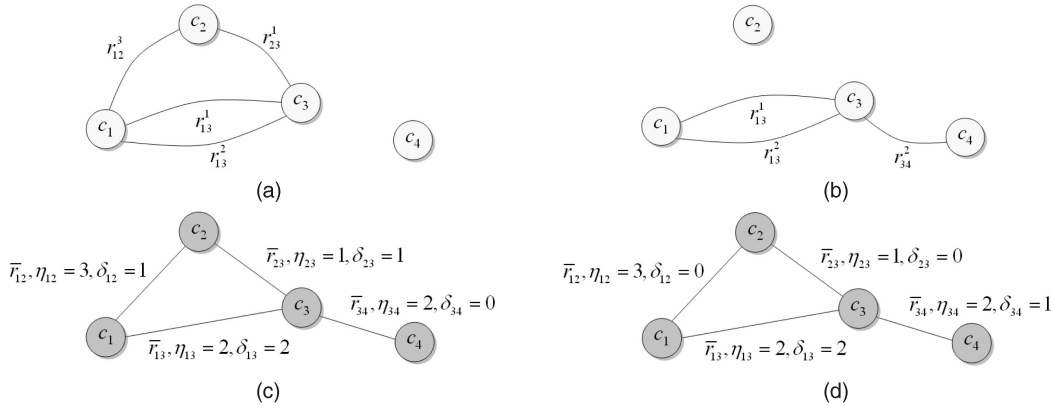

Fig. 7. (a) An annotated page p_1 where concept c_4 is not linked to any other concepts. (b) Page subgraph for a query Q specifying c_1, c_2, c_3, and c_4. (c) Annotation of a second page p_2, where c_2 is not linked to any other concepts. (d) Page subgraph for the same query.

vertices are linked by exactly one edge. However, we introduce the term "constrained" to recall that there exists a constraint on the number of edges of the forest allowing for the existence of not connected vertices in the graph. Since there is no way to infer which was the link between concepts more relevant to the user at the time of defining the query, constrained page spanning forests characterized by the same number of edges can be considered as comparable in terms of relevance with respect to the user query. All the constrained page spanning forests composed by the same number of edges represent a possible (even if less relevant) answer to the user query. Based on the number of constrained page spanning forests that can be generated from the page subgraph for a given number of edges, the probability of that page can be calculated as the sum of the probabilities computed for each constrained page spanning forest of a given length divided by the total number of constrained page spanning forests of that length that can be originated by the page subgraph. In the following, this latter consideration, together with the empirical results presented in this section, will be exploited in order to provide a general rule for relation-based ranking of semantic annotated Web pages.

4.3 Page Relevance Score and Ranking

Let us consider an ontology graph G and a query subgraph G_Q. Let us consider a page p and let us derive its page subgraph $G_{Q,p}$. We now define $SF_{Q,p}(l)$, which is the set including all the constrained spanning forests for a given number of edges l $(1 \leq l < |C_{Q,p}|)$. The cardinality of this set is $\sigma_{Q,p}(l) = |SF_{Q,p}(l)|$. Finally, let us define $SF^f_{Q,p}(l)$ as the fth spanning forest originated from the page subgraph for the given query Q and page p and a specific number of edges l. When l is equal to the maximum length of a spanning forest of the page subgraph, this correspond to a page spanning forest. Otherwise, it corresponds to a constrained page spanning forest. To simplify the explanation, we will sometimes refer to both these forests as page forests (except when this can cause ambiguities). The probability that a page forest $SF^f_{Q,p}(l)$ is the one of interest to the user can be written as $P(SF^f_{Q,p}(l))$. Based on the considerations above, it is possible to define a constrained relevance score for page p as

$$P(Q,p,l) = P\left(\bigcup_{f=1}^{|SF_{Q,p}(l)|}\left(\bigcap\left\{\bar{r}_{ij,p}|\bar{r}_{ij,p} \in SF^f_{Q,p}(l)\right\} \cap SF^f_{Q,p}(l)\right)\right) = \sum_{f=1}^{|SF_{Q,p}(l)|}\prod_{\bar{r}_{ij,p}\in SF^f_{Q,p}(l)} P(\bar{r}_{ij,p}) \cdot P\left(SF^f_{Q,p}(l)\right), \tag{5}$$

where $P(SF^f_{Q,p}(l)) = 1/\sigma_{Q,p}(l)$. We call it a *constrained page relevance score* since its value depends on the value of l.

By iteratively considering all the constrained spanning forests characterized by the same length, we are progressively relaxing the constraint of having all the concepts related in some way to other concepts within the page. As soon as a value different than zero is obtained for $P(Q,p,l)$, we assume that this corresponds to a "final" relevance score for that page.

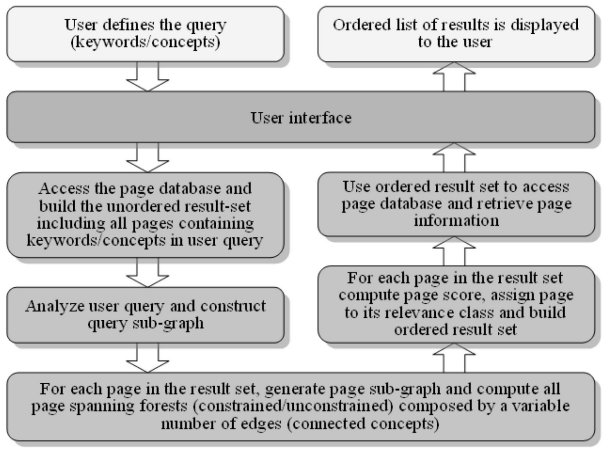

Fig. 8. Workflow from query definition to the presentation of results.

However, since $P(Q,p,l)$ is computed as a probability, we have $0 \leq P(Q,p,l) \leq 1$. Thus, $P(Q,p,l)$ cannot be directly used to compare one page in the result set with the remaining ones. Nevertheless, we can exploit the information on l to create several relevance classes in a straightforward way. In fact, reducing the value of l, as soon as we find a value different than zero for $P(Q,p,l)$, we compute the *page relevance score* (or *page score*) as

$$ps_{Q,p} = P(Q,p,\max(l)) + \max(l)|P(Q,p,l) \neq 0. \tag{6}$$

In this way, each relevance class contains pages with a score in the range $]l, l+1]$, and pages within the same class are directly comparable, and the (final) result set can be ordered by decreasing values of the page score.

5 Implementation of the Ranking Algorithm

5.1 Overall Procedure

We now assemble the various steps illustrated in the previous sections to present the overall ranking methodology (whose workflow is depicted in Fig. 8). The user starts defining query keywords and concepts. The search engine logic accesses the Web page database, constructs the initial result set including all those pages that contain queried keywords and concepts, and computes the query subgraph. Then, for each page in the result set, the page subgraph is computed. Starting from each subgraph, all page spanning forests (both constrained and unconstrained) are generated and used to compute the page score based on (6). Web pages are associated to relevance classes, and the final (ordered) result set is constructed.

5.2 Spanning Forest Generation Algorithm

According to (6), calculating the relevance score for a single page requires considering all the page forests and, for each forest, computing the constrained page relevance score. This requires finding an efficient way for both enumerating all the page forests for a given query and computing the page probability.

Two strategies are feasible in order to approach the problem above. The first strategy could be to consider all the possible page spanning forests (page forests including a

number of edges equal to the number of nodes minus one) of the page subgraph and progressively remove their edges generating constrained page spanning forests by taking care to avoid duplicate configurations. In the worst case, all the edges have to be recursively removed until page forests with a single edge are generated. It is worth observing that from a computational point of view, avoiding the production of duplicate configurations is an extremely expensive task. A second strategy could start by considering all the page forests of length one and generate all the possible page forests of increasing length by recursively adding a new edge until a page spanning forest is obtained. With respect to the previous approach, this second method shows several advantages. First, by properly selecting the edge to be added in the recursive process, it is possible to implicitly obtain a set of page forests without duplicates. Moreover, the iterative approach allows us to exploit the results achieved in previous steps in order to speed up the time requested for computation. In fact, the probability associated with a particular forest made up of a given number of edges can be obtained by simply taking into account the contribution of the newly added edge. That is, computing the probability of a forest composed by n edges simply requires multiplying the probability obtained for the page forest with $n-1$ edges by the relation probability associated to the additional edge.

Thus, in this work, we chose the second approach. Unfortunately, even if many algorithms have been proposed in the literature for addressing the task of finding all the spanning forests (or trees) in a graph [14], [30], none of them is capable of taking into account forests with a variable number of edges derived from originating spanning forests. Thus, an ad hoc algorithm has been designed (whose pseudocode is reported in Fig. 9). A detailed analysis of its complexity is provided in Section 6. It is worth observing that the incremental approach adopted in this algorithm shows an additional benefit with respect to the decremental one. In fact, it becomes possible to impose an upper bound to the growth of page forests in terms of the number of edges. Since a larger number of edges means a higher accuracy in the estimation of page relevance accompanied by a larger computational cost, the possibility of introducing a threshold to the widest page forest to be considered could allow us to achieve a trade-off between ranking precision and complexity.

```
Label the edges in G_{Q,p} with an index ranging from 1 to R_{Q,p}
Define variables e and a to index graph edges
Set η_e = η_ij      //number of relations linking concepts
                    // i and j in the ontology graph (edge e ∈ R_{Q,p})
Set δ_e = δ_ij      //number of relations linking concepts
                    // i and j in the page subgraph (edge e ∈ R_{Q,p})
Set τ_e = η_ij / δ_ij   //relation probability for edge e
Mark all the edges in G_{Q,p} as not visited
Allocate weight vector W of size |C_{Q,p}|-1
                    // W[l] stores the accumulated constrained
                    //probabilities for page forests of length l
Allocate vector Σ of size |C_{Q,p}|-1
                    // Σ[l] stores the number of page forests
                    //for a given length l
Initialize W and Σ to zero

for e=1, e≤|R_{Q,p}|, e=e+1
  mark edge e as visited
  visit(e,e,1,τ_e)
  W[1]=W[1]+τ_e
  Σ[1]=Σ[1]+1

function visit(o,e,l,s)
  a=e+1
  while a≤|R_{Q,p}| and l≤|C_{Q,p}|-1
    if a is not visited and a is safe
       //(does not introduce cycles, checked through DFS)
       mark edge a as visited
       visit(o,a,l+1,s×τ)
       W[l+1]=W[l+1]+s
       Σ[l+1]=Σ[l+1]+1
       set edge a as not visited
    else
       a=a+1
```

Fig. 9. Pseudocode of the algorithm for generating all the page spanning forests of variable length (incremental approach).

6 An Estimate of Algorithm Complexity

6.1 Overview of the Evaluation Method

According to the ranking method presented above, the relevance score for a particular page is given by the first nonzero constrained relevance score obtained by varying the value of l. The number of times that the length has to be varied depends on the annotation of the considered page. Moreover, the number of page forests for a given length depends on the topology of the page subgraph. Thus, in order to provide an estimate of algorithm complexity, we have to consider the worst case, represented by a page based on an ontology with a complete graph and whose annotation includes only one relation. In this case, all the possible lengths have to be considered, and the maximum number of spanning forests has to be taken into account.

6.2 Worst-Case Analysis

Let us consider an ontology characterized by a complete graph G. Let us assume that a query subgraph G_Q has been defined and let us consider a page subgraph $G_{Q,p}$, where $N_V = |C_{Q,p}|$ is the number of vertices, and $N_E = |R_{Q,p}|$ is the number of edges. By definition, given the number of edges of a complete page subgraph, the number of l-length spanning forests $\sigma_{Q,p}(l)$ is given by the number of subgraphs composed by l edges not including cycles. In other words, $\sigma_{Q,p}(l)$ can be computed as the overall number of possible combinations (without repetitions) of the N_E edges minus the number of subgraphs composed by l edges and including a cycle. The number of l-combinations of N_E edges can be expressed as

$$\kappa(l) = bin(N_E, l), \tag{7}$$

while the number $\chi_\lambda(l)$ of l-length subgraphs including an λ-cycle (with $3 \leq \lambda \leq l$) and having $l-\lambda$ additional "floating" (free) edges can be expressed as in (8). This expression takes into account the number $bin(N_V, \lambda)$ of combinations of λ vertices over the N_V vertices of the graphs, as well as the possible $(\lambda-1)!/2$ configurations for each combination. For each combination and for each configuration, the presence of $l-\lambda$ free edges, which can be combined in $bin(N_E-\lambda, l-\lambda)$ ways, is considered. Since the presence of free edges could lead to the generation of cycles longer than λ, a correction factor is applied. Another correction factor is applied to deal with duplicate configurations possibly resulting into shorter

TABLE 2
Number of Page Forests for Complete Page Subgraphs with Increasing Number of Nodes

	$N_V=2$	$N_V=3$	$N_V=4$	$N_V=5$	$N_V=6$	$N_V=7$
$l=1$	1	3	6	10	15	21
$l=2$		3	15	45	105	210
$l=3$			16	110	435	1295
$l=4$				125	1080	5250
$l=5$					1296	13377
$l=6$						16807

cycles. Because of the novelty of this formulation, the proofs for (8) will be provided in a specific paper.[2]

$$\chi_\lambda(l)=\begin{cases}\binom{N_V}{\lambda}\cdot\frac{(\lambda-1)!}{2} & l=\lambda,\\ \binom{N_V}{\lambda}\cdot\frac{(\lambda-1)!}{2}\cdot\left\{\binom{N_E-\lambda}{l-\lambda}-\left[\frac{\lambda\cdot(\lambda-1)}{2}-\lambda\right]\cdot\binom{N_E-\lambda-1}{l-\lambda-1}\right\}\\ -\sum_{i=2}^{l-\lambda}\binom{N_V}{\lambda}\frac{(\lambda-1)!}{2}\frac{\lambda}{i}^{(N_V-\lambda)}\cdot\binom{N_E-\lambda-i}{l-\lambda-i}\\ -\lfloor(l-\lambda)/3\rfloor\cdot\binom{N_V}{\lambda}\cdot\left(\frac{\lambda\cdot(N_V-\lambda)}{1+\lfloor(l-\lambda)/3\rfloor}+\frac{(N_V-\lambda)}{\lfloor(l-\lambda)/3\rfloor}\right)\cdot\binom{N_E-\lambda-\lfloor(l-\lambda)/3\rfloor\cdot\lambda}{l-\lambda-\lfloor(l-\lambda)/3\rfloor\cdot\lambda} & l>\lambda\geq 3.\end{cases}\tag{8}$$

In conclusion, expression for $\sigma_{Q,p}(l)$ can be written as

$$\sigma_{Q,p}(l)=\kappa(l)-\sum_{\lambda=3}^{l}\chi_\lambda(l).\tag{9}$$

Values for $\sigma_{Q,p}(l)$ computed using (9) over complete page subgraphs including a number of concepts N_V ranging from two to seven are reported in Table 2. It is worth observing that according to the methodology presented in the previous section, computing the constrained relevance score requires $\sigma_{Q,p}(l)$ accumulations plus one division for $l=1$ and $\sigma_{Q,p}(l)$ multiplications, $\sigma_{Q,p}(l)$ accumulations, and one division for $1<l\leq N_V-1$. In the worst case, ranking one page requires $\sum_{l=1}^{N_V-1}\sigma_{Q,p}(l)$ accumulations, $1+\sum_{l=2}^{N_V-1}\sigma_{Q,p}(l)$ multiplications, and N_V-1 divisions.

7 Experimental Results and Comparisons

In this section, the applicability of our technique into real scenarios will be analyzed by conducting two types of evaluations aimed at measuring the performance in terms of both time complexity and accuracy. The time complexity will be compared with that of [16], since our technique could be easily seen as an extension of it. Nevertheless, since the methodology in [16] is not targeted at ranking the result set, the accuracy of results will be compared with that of a traditional search engine like Google.

7.1 Time Complexity

The computation of fair results concerning time complexity requires a sufficiently large repository with a significant number of annotated pages. Because of the difficulty of integrating the proposed technique within today's search engines like Google, in which a native semantic layer is actually missing, we chose to estimate the computation time over a synthetic Semantic Web environment. The positive effect of this choice is twofold. On one hand, it is possible to work on as many pages as needed, thus effectively simulating the next-generation Semantic Web repositories. On the other hand, by statistically annotating Web pages, we do not incur in the risk of biasing the result. In order to compare our measures with those of [16], we worked with the same ontology (*travel.owl*), and we selected the same query (in the query, illustrated in Fig. 10, specific keywords and concepts defined in [16] have been replaced with numeric indexes). We automatically generated a Web page database with one million pages, each page containing all the keywords specified in the query. For each page, we constructed a semantic annotation based on the concepts defined in the selected ontology, randomly associating to each keyword one of the concepts in the ontology. We adjusted the statistical parameters so as to obtain a set of approximately 100,000 pages (precisely, 96,843 pages) including at least one of the keywords associated exactly to the concept specified in the query. Finally, we added semantic relations between concepts by uniformly distributing them across pages. In this way, each pair of concepts was linked by a variable subset of the relations associated to that pair in the ontology (each page containing approximately 10 relations). The distribution of concepts and relations in the Web page database is summarized in Table 3. For each concept $c_i|c_i\in C_Q$, Table 3 reports the number of pages containing exactly the association keyword/concept (k_i,c_i) defined in the query. Moreover, for each couple of concepts $c_i,c_j|c_i,c_j\in C_Q$, it reports the number of pages in which both the concepts are associated to the keywords specified in the user's query. Finally, for each couple of concepts c_i and c_j, Table 3 gives the number of pages that contain at least one of the relations $\bar{r}_{ij}^d|\bar{r}_{ij}^d\in\bar{R}_{ij}$, $1\leq d\leq\eta_{ij}$ defined in the ontology, as well as the exact number of pages existing in the Web page database for each of the possible relations. Both the approach in [16] and the methodology presented in this paper have been applied onto the experimental environment above using an Intel Core 2 6400 CPU at 2.13 GHz with 2 Gbytes of RAM. Results are shown in Table 4.

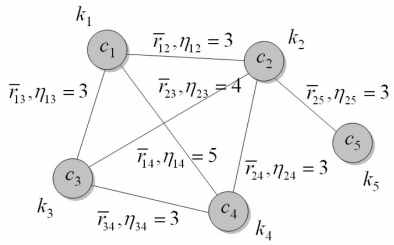

Fig. 10. Ontology from [16] used for measuring time complexity.

As illustrated in Section 2, in [16], a query graph is constructed over the ontology starting from concepts and keywords passed by the user. Then, the query graph is progressively reduced, thus obtaining several query subgraphs (Table 4, column 3). By taking for each edge in the query subgraph one of the possible relations associated to that edge, several property-keyword pairs are generated (column 4). The collection of these pairs constitutes a CRKS, which is submitted to the knowledge database for retrieving the page result set. With respect to [16], results related to CRKS generation present a speedup due to the newer hardware used for the experiments and to an optimized procedure for combining contributing edges. However, the overall delay (column 6) is worse than that in [16], since the

2. An application to a concrete example is shown in the Appendix, which is available online: http://gohan.polito.it:8080/tkde/appendix.pdf.

TABLE 3
Statistics Related to the Experimental Database Used for Evaluating Performance (Time Complexity)

	c_j (j=1)	c_j (j=2)	c_j (j=3)	c_j (j=4)	c_j (j=5)	
c_i (i=1)	-	25131* 16811$^{\pm}$ 8305/8373/8482$^{\square}$	25231* 20201$^{\pm}$ 10132/9988/9918/10013/10112$^{\square}$	25097* 16793$^{\pm}$ 8416/8456/8346$^{\square}$	25153* 0$^{\pm}$ 0$^{\square}$	50186$^{\#}$
c_i (i=2)	-	-	25118* 18855$^{\pm}$ 9251/9436/9412/9324$^{\square}$	25119* 18940$^{\pm}$ 9519/9464/9475/9566$^{\square}$	25062* 16750$^{\pm}$ 8351/8320/8256$^{\square}$	50146$^{\#}$
c_i (i=3)	-	-	-	25143* 16764$^{\pm}$ 8383/8297/8353$^{\square}$	25081* 0$^{\pm}$ 0$^{\square}$	49978$^{\#}$
c_i (i=4)	-	-	-	-	25041* 0$^{\pm}$ 0$^{\square}$	50181$^{\#}$
c_i (i=5)	-	-	-	-	-	49990$^{\#}$

Pages: $^{\#}$ *with at least one pair* (c_i, k_j), * *with both pairs* (c_i, k_j) *and* (c_j, k_j), $^{\pm}$ *including also at least one of the relations* $\bar{r}_{ij}^{d}$, *and* $^{\square}$ *with specific relation* $\bar{r}_{ij}^{d}$.

time for submitting CRKSs to the database and for intersecting the results is also taken into account (column 5).

The results obtained using our methodology are tabulated in columns 7-9. In particular, in column 7, the average number of page spanning forests for increasing length given the number of query concepts/keywords is reported. Moreover, column 8 gives the time requested for extracting from the database the annotation of pages to be processed and for generating the associated page subgraphs, together with the time requested for running the proposed algorithms and getting page relevance scores. It is worth observing that even if a final ordering is needed to sort the results, this delay has not been considered in Table 4 in order to let the user evaluate the time complexity of comparable result sets.

Experimental results show that the methodology in [16] is severely affected by the costs associated with the submission of CRKSs to the database and with the intersection of results, while our spanning-forest-based approach allows us to effectively manage the search space and to reduce the time complexity associated with the search task. The additional advantage of the proposed approach is that it incorporates the computation of a probability measure that can be effectively used to produce an ordered result set. Even if promising results over one million pages demonstrate the feasibility of the approach, at the same time, they anticipate the need for further research activities aimed at ensuring scalability with the next-generation Semantic Web repositories. In Section 5.2, we propose a practicable approach based on a threshold over the computed spanning forests' width. Nevertheless, we expect to further investigate this issue by analyzing the effect of the adoption of parallel and distributed computing paradigms, as well as of solutions for storing precomputed (and updated) digests of page subgraphs.

7.2 Accuracy

The accuracy of the proposed technique has been evaluated against the result set generated by running the query "hotel," "Rome," "four stars," "gym," and "tennis" (or "hotel," "Roma," "quattro stelle," "palestra," and "tennis") over the Italian version of Google on 6 January 2008. Web pages returned by Google are reported in their original order in Table 5. As remarked in Section 1, it can be observed how there exist possibly out-of-scope pages that have been ranked as very relevant (a four-star hotel without tennis facilities located in Abano Terme, 500 km from Rome, is on the top of the result set), while potentially interesting pages (like the Rome Hilton Cavalieri) are positioned at the end of the list.

In order to apply our ranking methodology and show how existing search engines could benefit from its application, we manually annotated each page using concepts *Accomodation*, *Destination*, *Accomodation rating*, and *Activity* in the *travel.owl* ontology. Relations were specified by following a fair approach relying only on the information contained in the Web page (fourth column, keywords/concepts numbered progressively from one to five). The constrained page relevance score is reported in column 5. From column 2, it can be easily observed that the ranking is significantly improved. For example, the first four entries now refer to Web pages that completely satisfy the user query; entries 5 and 6 refer to hotels with all the requested characteristics, located in the vicinity of Rome and providing transfers to it. Another interesting example is provided by the fifth entry (now 15th), whose ranking was boosted through hidden text Web spam [12]. Even if the Semantic Web will require the development of ad hoc techniques for "semantic" antispam [31], our solution proved to be able to cope with the presence of malicious information in today's Web pages.

TABLE 4
Experimental Results of Our Approach Compared with Those of [16] (Time Complexity)

Query concepts/ keywords	Relations among concepts	Subgraphs processed [16]	CRKSs processed [16]	Gen. CRKSs/ Subm. to DB and inters. [16] (ms)	Total delay of [16] (ms)	Average num. of PFs (l=1/2/3/4)	Query the DB and gen. subgraphs/ Proc. PFs (ms)	Total delay [our] (ms)
1	0	0	0	-/-	-	-/-/-/-	-/-	-
2	1	2	3	0.87/0.81	**1.68**	1/-/-/-	7.10/2.45	**9.55**
3	2	4	15	7.45/175.67	**183.12**	2/1/-/-	31.32/47.63	**78.95**
4	4	16	383	41.27/1251.37	**1292.64**	4/5/2/-	62.44/156.74	**219.18**
5	7	128	38399	632.41/2815.71	**3448.12**	7/12/11/9	84.69/688.16	**772.85**

TABLE 5
Accuracy of Our Ranking Algorithm over the First 20 Entries of a Result Set Generated by Google

Google ranking	Our ranking	Web page highlights, URL and content summary (some of the reasons for inclusion in the Google result set and guidelines used for identifying relations).	Rel. in page subgraph	Contrained relev. score
1	13	**Hotel** 4 **stelle** Abano Terme - **Hotel quattro stelle** Abano Terme www.abanoprenotazioni.it/**hotel**-4-**stelle**-Abano-terme.asp *Web page of a four-star hotel named "Hotel Terme Roma", with gym, in Abano Terme, five hundred kilometers from Rome. Another hotel in the Web page has gym and tennis courts.*	$\delta_{13}=1$ $\delta_{14}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.000000$ $P(Q,p,2)=0.333333$ $P(Q,p,1)=0.428571$
2	14	**Hotel** 4 **stelle** Abano Terme - **Hotel quattro stelle** Abano Terme www.abanoprenotazioni.it/**hotel**-4-**stelle**/**hotel**-**quattro**-**stelle**-abano-terme.asp *Web page of a four-star hotel named "Hotel Terme Roma", with gym, in Abano Terme, five hundred kilometers from Rome. Another hotel in the Web page has gym and tennis courts.*	$\delta_{13}=1$ $\delta_{14}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.000000$ $P(Q,p,2)=0.333333$ $P(Q,p,1)=0.428571$
3	11	**Hotel** 4 **stelle** Ischia - **Hotel quattro stelle** Ischia www.ischiaprenotazioni.it/**hotel**-4-**stelle**-ischia.asp *Four-star hotel located in Ischia, renowned isle in front of Naples. Hotel facilities include gym and tennis courts.*	$\delta_{13}=1$ $\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.166667$ $P(Q,p,2)=0.333333$ $P(Q,p,1)=0.547619$
4	4	**Hotel** Petra & Residence - **Hotel Roma** www.initalia.it/**hotel**/**hotel**petraresidence.htm *Four-star hotel located in a residential area south-west of Rome, with gym and tennis courts. Easy connections to the city centre through public transports.*	$\delta_{12}=1$ $\delta_{13}=1$ $\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.041667$ $P(Q,p,3)=0.108974$ $P(Q,p,2)=0.255556$ $P(Q,p,1)=0.511905$
5	15	**Roma** vacanze on line viaggi città arte alberghi hotels bed **...** www.eurovacanza.com/strutture_html/idDest/9_lin/ita *Four-star hotel located in Rome. Web page describes other hotels with different accomodation ratings. Web page has hidden spam inlcuding generic "gym" and "tennis" keywords.*	$\delta_{12}=1$ $\delta_{13}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.000000$ $P(Q,p,2)=0.250000$ $P(Q,p,1)=0.892857$
6	5	Grand **Hotel** Duca d'Este Tivoli **Roma** *annunci.repubblica.it/**roma**/turismo/**hotel**/grand-**hotel**-duca-d-este-tivoli-pj-798427.html* *Four-star hotel located in Tivoli, a small town approximately twenty kilometers from Rome. Hotel facilities include gym and tennis courts.*	$\delta_{12}=1$ $\delta_{13}=1$ $\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.041667$ $P(Q,p,3)=0.108974$ $P(Q,p,2)=0.255556$ $P(Q,p,1)=0.511905$
7	8	Alberghi per sportivi a S'ORU E MARI, **hotel** con **palestra**, piscina **...** www.superdossier.com/attrezzati_per_lo_sport/SARDEGNA/CAGLIARI/SORU_E_MARI/ *Four-star hotel with gym and tennis courts located in Sardegna. In the Web page there are links allowing to seach for hotels in all the main Italian cities (Rome, and others).*	$\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.166667$ $P(Q,p,2)=0.333333$ $P(Q,p,1)=0.547619$
8	6	Piste sci trekking scuola **hotel** albergo **quattro stelle** Monte **...** www.**hotel**cristalloterminillo.it/sport.asp *Four-star hotel with gym and tennis courts on the Monte Terminillo, a renowned winter location a few kilometers from Rome.*	$\delta_{12}=1$ $\delta_{13}=1$ $\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.041667$ $P(Q,p,3)=0.108974$ $P(Q,p,2)=0.255556$ $P(Q,p,1)=0.511905$
9	1	CROWNE PLAZA **ROMA** - ST. PETER'S **HOTEL** SPA - **Hotel** Invest Italiana www.**hotel**-invest.com/index.asp?id=385 *Four-star hotel located in the city centre of Rome, with gym and tennis courts. The hotel is close to major attractions and offers a shuttle bus to main locations in the city.*	$\delta_{12}=3$ $\delta_{13}=1$ $\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.125000$ $P(Q,p,3)=0.224359$ $P(Q,p,2)=0.366667$ $P(Q,p,1)=0.583333$
10	12	Pagine gialle aziendali Viaggi turismo e tempo libero per l'import **...** viaggi-affari.europages.it/epq/dmc/l-it/did-21/hc-21510/**Hotel**s_Italia.html *Web page describing a four-star hotel in Rome, as well as other hotels located in different Italian cities with gym or tennis courts characterized by various accomodation ratings.*	$\delta_{12}=2$ $\delta_{13}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.000000$ $P(Q,p,2)=0.500000$ $P(Q,p,1)=0.928571$
11	9	**Hotel** St. Gregory Park - Albergo **quattro stelle** San Giuliano Mare **...** www.abcfiere.com/dettaglio_**hotel**_st_gregory_park_rimini_219-1.php *Four-star hotel with gym and tennis courts located in Rimini (four hundred kilometers from Rome). Web page advertises several fashion events in Rome.*	$\delta_{13}=1$ $\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.166667$ $P(Q,p,2)=0.333333$ $P(Q,p,1)=0.547619$
12	3	Alberghi 4 **stelle Roma** - **Hotel** Aldobrandeschi - **hotel Roma quattro ...** www.**hotel**aldobrandeschi.it/travel/it/alberghi_4_**stelle_roma**.htm *Four-star hotel in Rome with easy connections to the city and arranged rates for a gym in the vicinity. Hotel is close to a sporting complex with tennis courts.*	$\delta_{12}=2$ $\delta_{13}=1$ $\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.083333$ $P(Q,p,3)=0.166667$ $P(Q,p,2)=0.311111$ $P(Q,p,1)=0.547619$
13	2	**hotel** lusso **roma** www.rome-luxury-**hotel**.com/**hotel**-lusso-**roma**.htm *Four-star hotel located few minutes from the historical centre of Rome, with free shuttle bus to downtown. Hotel facilities inlcude gym and tennis court.*	$\delta_{12}=2$ $\delta_{13}=1$ $\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.083333$ $P(Q,p,3)=0.166667$ $P(Q,p,2)=0.311111$ $P(Q,p,1)=0.547619$
14	10	Grand **Hotel** Palazzo della Fonte Fiuggi www.iper**hotel**.com/index.cfm?Fuseaction=**Hotel**.Info**Hotel**&Id**Hotel**=325 *Five-star hotel located in Fiuggi, a hundred kilometers from Rome, with gym and tennis courts (with instructors). Free shuttle bus to Rome city centre.*	$\delta_{12}=2$ $\delta_{14}=1$ $\delta_{15}=2$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.166667$ $P(Q,p,2)=0.393939$ $P(Q,p,1)=0.833333$
15	18	GOLF TOSCANA - **HOTEL** - GOLF TOSCANA - CAMPI DA GOLF **...** www.**hotel**benessere.it/golf-toscana.htm *Golf clubs in Toscana with gym and tennis courts. Web page provide links to hotels in the vicinity and driving directions from the main Italian cities (Rome, and others).*	$\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.000000$ $P(Q,p,2)=0.166667$ $P(Q,p,1)=0.476191$
16	17	**Hotel** Cervia, Albergo Cervia, Hotels Cervia, Alberghi Cervia **...** it-**hotel**.7mates.com/italia/emilia-**roma**gna/cervia.htm *Three-star hotel with gym named "Hotel Roma" located in Cervia, on the Riviera Romagnola. Web page describes other hotels (three and four stars) in Cervia with tennis courts.*	$\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.000000$ $P(Q,p,2)=0.166667$ $P(Q,p,1)=0.476191$
17	19	Siria - Viaggi, valigie e salute viaggiare.mevigi.it/articoli_viaggiare/200701181736151743625 8.lasso *Travel to Siria departing from Rome and organized by a travel agency in Rome. Accomodation in a five-star hotel with gym and tennis courts (or in a four star hotel).*	$\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.000000$ $P(Q,p,2)=0.166667$ $P(Q,p,1)=0.476191$
18	20	Mauritius, MAURITIUS : BLUE LAGOON BEACH 3 **Stelle** da 1viaggi **...** www.1viaggi.com/pacchetto/tour_fuga_nel-60.asp *Travel to the Mauritius with accomodation in a three-star hotel with gym and tennis courts. Flight departing from Rome FCO airport.*	$\delta_{14}=1$ $\delta_{15}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.000000$ $P(Q,p,2)=0.166667$ $P(Q,p,1)=0.476191$
19	7	Offerte - Hilton Cavalieri - Rome - **Roma hotel** - Italia - hotels.com www.**hotel**.it/albergo-italia/albergo-**roma**/hilton-cavalieri-rome/ *Five-star hotel located in the city centre of Rome, with free shuttle service to main city locations. Hotel facilities include gym and tennis courts.*	$\delta_{12}=2$ $\delta_{14}=2$ $\delta_{15}=2$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.333333$ $P(Q,p,2)=0.560606$ $P(Q,p,1)=0.880952$
20	16	**Roma** vacanze on line viaggi città arte alberghi hotels bed **...** www.eurovacanza.com/strutture_html/idProvincia/94_lin/ita *Four-star hotel located in Rome. Web page describes other hotels with different accomodation ratings. Web page has hidden spam inlcuding generic "gym" and "tennis" keywords.*	$\delta_{12}=1$ $\delta_{13}=1$	$P(Q,p,4)=0.000000$ $P(Q,p,3)=0.000000$ $P(Q,p,2)=0.250000$ $P(Q,p,1)=0.892857$

8 Conclusion

The next-generation Web architecture represented by the Semantic Web will provide adequate instruments for improving search strategies and enhance the probability of seeing the user query satisfied without requiring tiresome manual refinement. However, actual methods for ranking the returned result set will have to be adjusted to fully exploit additional contents characterized by semantic annotations including ontology-based concepts and relations. Several ranking algorithms for the Semantic Web exploiting relation-based metadata have been proposed. Nevertheless, they mainly use page relevance criteria based

on information that has to be derived from the whole knowledge base, making their application often unfeasible in huge semantic environments. In this work, we propose a novel ranking strategy that is capable of providing a relevance score for a Web page into an annotated result set by simply considering the user query, the page annotation, and the underlying ontology. Page relevance is measured through a probability-aware approach that relies on several graph-based representations of the involved entities. By neglecting the contribution of the remaining annotated resources, a reduction in the cost of the query answering phase could be expected. Despite the promising results in terms of both time complexity and accuracy, further efforts will be requested to foster scalability into future Semantic Web repositories based on multiple ontologies, characterized by billions of pages, and possibly altered through next-generation "semantic" spam techniques.

REFERENCES

[1] B. Aleman-Meza, C. Halaschek, I. Arpinar, and A. Sheth, "A Context-Aware Semantic Association Ranking," *Proc. First Int'l Workshop Semantic Web and Databases (SWDB '03),* pp. 33-50, 2003.

[2] K. Anyanwu, A. Maduko, and A. Sheth, "SemRank: Ranking Complex Relation Search Results on the Semantic Web," *Proc. 14th Int'l Conf. World Wide Web (WWW '05),* pp. 117-127, 2005.

[3] R. Baeza-Yates, L. Calderón-Benavides, and C. González-Caro, "The Intention behind Web Queries," *Proc. 13th Int'l Conf. String Processing and Information Retrieval (SPIRE '06),* pp. 98-109, 2006.

[4] T. Berners-Lee and M. Fischetti, *Weaving the Web.* Harper Audio, 1999.

[5] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific Am.,* 2001.

[6] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Proc. Seventh Int'l Conf. World Wide Web (WWW '98),* pp. 107-117, 1998.

[7] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, "XSEarch: A Semantic Search Engine for XML," *Proc. 29th Int'l Conf. Very Large Data Bases,* pp. 45-56, 2003.

[8] L. Ding, T. Finin, A. Joshi, R. Pan, R.S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: A Search and Metadata Engine for the Semantic Web," *Proc. 13th ACM Int'l Conf. Information and Knowledge Management (CIKM '04),* pp. 652-659, 2004.

[9] L. Ding, T. Finin, A. Joshi, Y. Peng, R. Pan, and P. Reddivari, "Search on the Semantic Web," *Computer,* vol. 38, no. 10, pp. 62-69, Oct. 2005.

[10] L. Ding, P. Kolari, Z. Ding, and S. Avancha, "Using Ontologies in the Semantic Web: A Survey," *Ontologies,* pp. 79-113, Springer, 2007.

[11] R. Guha, R. McCool, and E. Miller, "Semantic Search," *Proc. 12th Int'l Conf. World Wide Web (WWW '03),* pp. 700-709, 2003.

[12] Z. Gyongyi and H. Garcia-Molina, "Spam: It's Not Just for Inboxes Anymore," *Computer,* vol. 38, no. 10, pp. 28-34, Oct. 2005.

[13] C. Junghoo, H. Garcia-Molina, and L. Page, "Efficient Crawling through URL Ordering," *Computer Networks and ISDN Systems,* vol. 30, no. 1, pp. 161-172, 1998.

[14] S. Kapoor and H. Ramesh, "Algorithms for Enumerating All Spanning Trees of Undirected and Weighted Graphs," *SIAM J. Computing,* vol. 24, pp. 247-265, 1995.

[15] Y. Lei, V. Uren, and E. Motta, "SemSearch: A Search Engine for the Semantic Web," *Proc. 15th Int'l Conf. Managing Knowledge in a World of Networks (EKAW '06),* pp. 238-245, 2006.

[16] Y. Li, Y. Wang, and X. Huang, "A Relation-Based Search Engine in Semantic Web," *IEEE Trans. Knowledge and Data Eng.,* vol. 19, no. 2, pp. 273-282, Feb. 2007.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Stanford Digital Library Technologies Project, 1998.

[18] A. Pisharody and H.E. Michel, "Search Engine Technique Using Keyword Relations," *Proc. Int'l Conf. Artificial Intelligence (ICAI '05),* pp. 300-306, 2005.

[19] T. Priebe, C. Schlager, and G. Pernul, "A Search Engine for RDF Metadata," *Proc. 15th Int'l Workshop Database and Expert Systems Applications (DEXA '04),* pp. 168-172, 2004.

[20] H. Knublauch, *Protégé,* Stanford Medical Informatics, http://protege.cim3.net/file/pub/ontologies/travel/, 2002.

[21] *Resource Description Framework (RDF) Model and Syntax Specification,* http://www.w3.org/TR/rdf-primer, 2004.

[22] C. Rocha, D. Schwabe, and M.P. Aragao, "A Hybrid Approach for Searching in the Semantic Web," *Proc. 13th Int'l Conf. World Wide Web (WWW '04),* pp. 374-383, 2004.

[23] A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut, and Y. Warke, "Managing Semantic Content for the Web," *IEEE Internet Computing,* pp. 80-87, 2002.

[24] A. Sheth, B. Aleman-Meza, I.B. Arpinar, C. Bertram, Y. Warke, C. Ramakrishanan, C. Halaschek, K. Anyanwu, D. Avant, F.S. Arpinar, and K. Kochut, "Semantic Association Identification and Knowledge Discovery for National Security Applications," *J. Database Management,* vol. 16, no. 1, pp. 33-53, 2005.

[25] N. Stojanovic, "An Explanation-Based Ranking Approach for Ontology-Based Querying," *Proc. 14th Int'l Workshop Database and Expert Systems Applications,* pp. 167-175, 2003.

[26] N. Stojanovic, R. Studer, and L. Stojanovic, "An Approach for the Ranking of Query Results in the Semantic Web," *Proc. Second Int'l Semantic Web Conf. (ISWC '03),* pp. 500-516, 2003.

[27] R. Sun, H. Cui, K. Li, M.Y. Kan, and T.S. Chua, "Dependency Relation Matching for Answer Selection," *Proc. ACM SIGIR '05,* pp. 651-652, 2005.

[28] T. Tran, P. Cimiano, S. Rudolph, and R. Studer, "Ontology-Based Interpretation of Keywords for Semantic Search," *Proc. Sixth Int'l Semantic Web Conf.,* pp. 523-536, 2007.

[29] *Web Ontology Language,* http://www.w3.org/2004/OWL/, 2004.

[30] B.Y. Wu and K.M. Chao, *Spanning Trees and Optimization Problems.* CRC Press, 2004.

[31] H. Yang, I. King, and M.R. Lyu, "DiffusionRank: A Possible Penicillin for Web Spamming," *Proc. ACM SIGIR '07,* pp. 431-438, 2007.

[32] Y.J. Zhang and Z.Q. Liu, "Refining Web Search Engine Results Using Incremental Clustering," *Int'l J. Intelligent Systems,* vol. 19, no. 1, pp. 191-199, 2004.

Fabrizio Lamberti received the degree in computer engineering and the PhD degree in computer engineering from the Politecnico di Torino, Torino, Italy, in 2000 and 2005, respectively. He is currently with the Dipartimento di Automatica ed Informatica, Politecnico di Torino. He has published a number of technical papers in international journals and conferences in the areas of distributed computing and information retrieval. He has served as a reviewer and program committee member for several international conferences and journals. He is a member of the IEEE and the IEEE Computer Society.

Andrea Sanna received the degree in electronic engineering and the PhD degree in computer engineering from Politecnico di Torino, Torino, Italy, in 1993 and 1997, respectively. He is currently with the Dipartimento di Automatica ed Informatica, Politecnico di Torino. He has authored and coauthored several papers in the areas of computer graphics, scientific visualization, and parallel and distributed computing. He serves as a reviewer for a number of international conferences and journals.

Claudio Demartini received the degree in computer engineering and the PhD degree in computer engineering from the Politecnico di Torino, Italy, in 1980 and 1987, respectively. He is currently with the Dipartimento di Automatica ed Informatica and the vice dean of the Industrial Engineering and Management School at Politecnico di Torino. He has published a number of technical papers in international journals and conference proceedings in the areas of distributed computing. He is a member of the IEEE.