Combining Fact and Document Retrieval with Spreading Activation for Semantic Desktop Search

Kinga Schumacher, Michael Sintek, and Leo Sauermann

Knowledge Management Department
German Research Center for Artificial Intelligence (DFKI) GmbH,
Kaiserslautern, Germany
{firstname.surname}@dfki.de

Abstract. The Semantic Desktop is a means to support users in Personal Information Management (PIM). It provides an excellent test bed for Semantic Web technology: resources (*e. g.*, persons, projects, messages, documents) are distributed amongst multiple systems, ontologies are used to link and annotate them. Finding information is a core element in PIM. For the end user, the search interface has to be intuitive to use, natural language queries provide a simple mean to express requests. State of the art semantic search engines focus on fact retrieval or on semantic document retrieval. We combine both approaches to search the Semantic Desktop exploiting all available information. Our semantic search engine, built on *semantic teleporting* and *spreading activation*, is able to answer natural language queries with facts, *e. g.*, a specific phone number, and/or relevant documents. We evaluated our approach on ESWC 2007 data in comparison with Google site search.

1 Introduction

The Semantic Desktop [20,7] is a means for personal knowledge management. It transfers the Semantic Web to desktop computers by consistent application of Semantic Web standards like the Resource Description Framework (RDF) and RDF Schema (RDFS). Documents, e-mails, contacts are identified by URIs, across application borders. The user is able to annotate, classify, and relate these resources, expressing his view in a *Personal Information Model* (PIMO) [21]. On a full-featured Semantic Desktop many data sources are integrated and searchable: a personal categorization system of topics, projects, contacts, *etc.* is established. The text and metadata of all documents is indexed and categorized. Together with the collected facts about people, organizations, events and processes, a critical amount of information is available to the user.

To use all information, we propose to use a Semantic Desktop search engine which is able to find structured information, to focus on unstructured information, and combine them providing a comprehensive search solution for the user.

S. Bechhofer et al.(Eds.): ESWC 2008, LNCS 5021, pp. 569–583, 2008.

To achieve this goal, the search engine automatically explores the knowledge base to find facts which answer the query and accomplish enhanced document retrieval embracing the found facts including the metadata of the documents. The engine should also facilitate not only the searching for documents but also *semantic teleporting*. Teleporting is a search strategy where the user tries to jump directly to the information target, *e. g.*, the query 'phone number of the DFKI KM-Group secretary' delivers the document which contains the wanted phone number [23]. Semantic teleporting does not deliver the document which contains the wanted phone number but the phone number itself. Another basic point is to enable *natural language queries* to keep knowledge overhead of special query languages away from the user. Free-text queries are usually dealt with NLP technology which, among other things, has to resolve syntactic ambiguity, *i. e.*, words with multiple meanings and structural ambiguity[1], *i. e.*, the ambiguity of the underlying structure of complex expressions [10]. For this reason we focus on semantic search approaches based on natural language queries which support the resolution of ambiguity.

Recently, several semantic search approaches with diverse application areas have been published. There are two main research thrusts: semantic search engines to retrieve documents which are enriched with semantics (*semantic document retrieval*) and engines which are deployed to search within ontologies (*fact retrieval*). Semantic document retrieval augments traditional keyword search with semantic techniques. Such search engines often use *thesauri* for query expansion and/or apply *graph traversal* algorithms to the available ontology for generalization, specification of the query terms, or to match predefined categories [10,15]. Fact retrieval approaches apply three kinds of core search techniques: *reasoning*, *triple based* (also referred to as statements based), *i. e.*, structural interpretation of the query guided by semantic relations, and *graph traversal* [10].

According to the requirements of searching the Semantic Desktop, our engine combines fact retrieval with semantic document retrieval using a triple-based algorithm and graph traversal. We selected these two approaches for the following reasons: Triple-based search provides the resolving of syntactic and structural ambiguity since the existing triples can constrict the possibilities of the interpretation of a concrete query (see Sect. 3.1). The reason for choosing a graph traversal algorithm, especially *spreading activation* [6], is that this approach enables an effective combination of fact retrieval and document retrieval (see Sect. 3.3). The goal of our approach is to provide a simple to use, yet powerful search functionality to users of the Semantic Desktop, through which all information from a PIMO can be retrieved, *i. e.*, both facts like a special phone number and relevant documents. We supports queries like 'literature by Sintek', 'semantic search papers', 'persons interested in document retrieval', 'phone number of Nepomuk's project manager', 'abstract and authors of papers about Semantic Desktop applications', 'when and where is the welcome reception of the ESWC 2007', *etc.*.

[1] For example, the sentence 'I saw the man with the telescope' has two underlying structures; it is not clear who is using the telescope [11].

The rest of this paper is organized as follows. The next section gives a state of the art overview on semantic search. Section 3 explains our approach, where Sect. 3.1 describes the fact retrieval, *i. e.*, the *semantic teleporting*, Sect. 3.2 the semantic document retrieval and Sect. 3.3 the way of their combination. The experimental evaluation including method, data, and results is dealt with in Sect. 4. Section 5 concludes the paper.

2 State of the Art

In recent years, several semantic search engines have been developed. There are some main publications which give a good overview on this research field.

Hildebrand *et al.* consider in [10] 35 existing systems and examine them according to the three main steps of the search process, *i. e.*, query construction, the core search process, and the representation of the results. They identify the general approaches for the core search process. C. Mangold focuses in [16] on semantic document retrieval. He introduces a classification scheme and compares 22 semantic document retrieval engines by means of the defined criteria.

Lei, Uren and Motta investigate how current semantic search approaches address user support without making restrictions on considered approaches [14]. This work points out the problem of knowledge overhead and the lack of support for answering complex queries. Search engines which allow the user to specify the query by choosing ontologies, classes, properties, values, and query language front-end engines require that the user possesses knowledge about the back-end knowledge base or copes with a special query language. In contrast, keyword-based semantic search is user-friendly but breeds the problem of supporting complex queries since allowing free-text queries causes syntactic and structural ambiguity [14,10].

Given that we aspire to provide natural language queries, we focus upon semantic search approaches which enable to resolve these ambiguities. Syntactic ambiguity can be solved by pre-query disambiguation, *e. g.*, in Squiggle [2], through user interaction, *e. g.*, MIT's Semantic Desktop Haystack [13]. A further possibility is to include the user context, *e. g.*, TAP [9]. In terms of resolving structural ambiguity, one such approach is a triple-based algorithm which defines and combines query templates to translate the free-text query into a formal query, introduced in [14]. Another triple-based approach uses keyword search to identify concepts of the ontology and forms RDF queries with one or two variables based on matched properties and non-properties, *i. e.*, subjects and objects of a triple [8].

F. Crestani gives in [6] a good introduction on the application of spreading activation techniques in semantic networks in information retrieval. Rocha *et al.* applies spreading activation as graph traversal algorithm for semantic document retrieval using weights assigned to links based on certain properties of the ontology [19]. Berger *et al.* [1] use spreading activation and take additionally the relatedness of terms into account.

Current approaches especially to Semantic Desktop Search concentrate on semantic document retrieval. Beagle^{++} [4,12,5] enhances document retrieval on the desktop by exploiting activity-based meta data, *e. g.*, a saved email attachment is annotated with the sender, date, subject, body text and status of the email. The engine executes a keyword search on the document index and on available meta data and ranks the found documents with an enhanced ranking system. The ranks are computed based on ontological relation weights and on ranks computed by considering external sources. Further approach for semantic document retrieval on the Semantic Desktop applies spreading activation to find resources which are sparsely annotated with semantic information [22]. The underlying semantic network (see Sect. 3.2) is initially set up with a path length based semantic similarity measure of concepts.

3 Searching the Semantic Desktop

Figure 1 depicts the architecture of our semantic search approach. The search engine is composed of a fact retrieval engine, a semantic document retrieval engine and a component which controls both in order to exploit all available data of the Semantic Desktop. The engines are linked among each other and they are connected to the knowledge base by the use of an interface which can be used to define and apply views and inferences. Our knowledge base, the PIMO, consists of the ontology and its instances. Native structures, *i. e.*, folders in file system and in email client are mapped to ontological concepts. Files and other information objects, *i. e.*, text documents, emails, address book entries are mapped to instances of the ontology. Documents also refer to the knowledge base, they are semantically tagged.

3.1 Fact Retrieval Approach

Our fact retrieval approach is based on the triple-based search algorithm by Goldschmidt and Krishnamoorthy [8]. Their semantic search engine processes natural language queries. Syntactic matching is done by string matching against the labels (`rdfs:label`), comments (`rdfs:comment`), and literals (`rdfs:literal`) in the knowledge base. Results are the URIs of found properties and non-properties (subjects and objects of a triple), weighted according to their frequency. The semantic matching is carried out by generating RDF queries as combinations of potential properties (p_i) and non-properties (n_j) with one or two variables, *e. g.*, ((n_1 p_1 ?), (? p_1 ?), (? p_1 n_2)). The process of creating and applying queries is repeated for each new resource detected but limited to only two hops in the knowledge base in order to avoid flooding with irrelevant inferences [8].

We adopted and extended this basic idea in order to enhance results by improving the syntactic matching and by providing more hops in the knowledge base conforming with the query. The *syntactic matching* of the query against the knowledge base step enacts, after removing stop words, a keyword search over the textual content of the knowledge base, *i. e.*, over all kinds of labels,

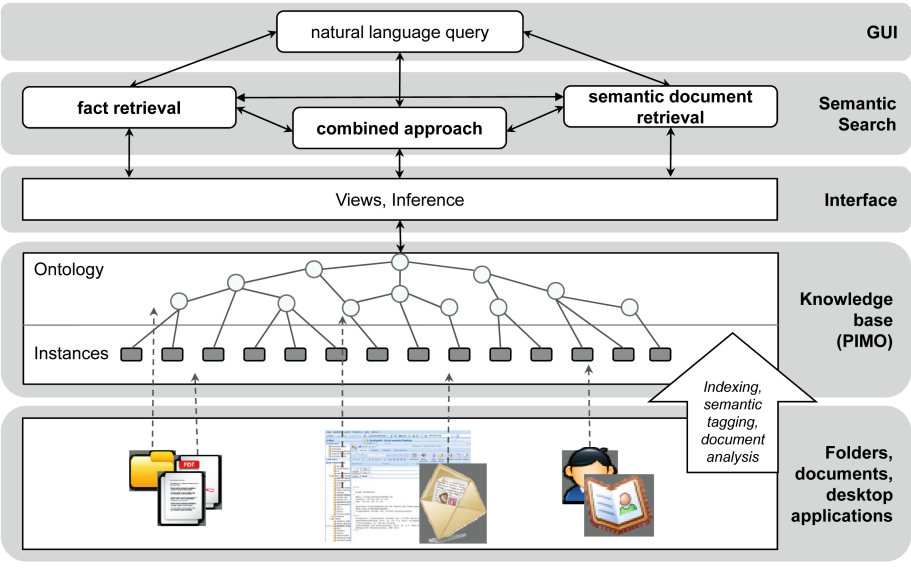

Fig. 1. Architecture

literals, local names of URIs and comments if available. Our knowledge base supports the search engine with synonyms/alternative names of the knowledge base elements (classes, properties, and instances). We use elements of the SKOS[2] formal language to define them, `<skos:prefLabel>` contains the labels which are used for visualizing and textual output, `<skos:altLabel>` stands for other alternative designators. Thus we perform query rewriting by augmentation [16], *i. e.*, we derive further terms from the ontological context of the query term. However, this part of the search engine focuses on the knowledge base in order to find the *perfect* answer and it is optimized for the Semantic Desktop where in addition to the use of thesauri the user is enabled to assign alternative names of ontological elements himself. Therefore, we only use the SKOS labels of the matched elements for this step, *i. e.*, without an explicit generalizing and specializing through hypernyms, meronyms, or adjacent concepts. This feature is customizable according to the application area. We use the n-gram method to compute the syntactic similarity of two strings[3]. The matches are also weighted with the n-gram value of the query term and label, literal or comment using

[2] Simple Knowledge Organisation Systems, `http://www.w3.org/2004/02/skos/`

[3] The N-gram method divides a sequence in subsequences of n items, *e. g.*, a string in subsequences of n characters. The similarity of two terms can then be determined by mapping the words to vectors containing the number of occurrences of several n-grams and computing the distance/similarity of them. More different ways are used for the n-gram decomposition and the computation of the similarity of two sequences. For example, the term 'basic' can be decomposed in the 2-grams ('ba', 'as', 'si', 'ic') and the Dice distance can be used as similarity measure [17]. We use 2-grams combined with 3-grams.

predefined thresholds for the selection of potential elements. Additionally we support phrase matching by determining possible phrases and comparing their n-gram weights with the weights of the single terms. For further steps, we differentiate between matched classes, properties and instances for each query term.

The *semantic matching* is simple in the case of one query term. When instances are matched, the engine returns them. For matched properties it delivers the associated triples of instances, in case of classes their instances. When the query is composed of multiple query terms we start a search process over three levels in order to detect relevant triples from the instance base.

1st level. At first, we consider all pairs of terms side by side, *e. g.*, if a query is composed of 3 terms we consider the pairs (t_1, t_2) and (t_2, t_3). We determine and execute the possible RDF queries based on the matched ontological elements of the terms and gathering the result bindings. For example, if the term t_1 matches the instance $inst_j$, t_2 matches the property $prop_i$ and the class $class_k$, the possible RDF queries are:

$$(inst_j\ ?\ class_k),\ (?\ prop_i\ inst_j),\ (inst_j\ prop_i\ ?) \tag{1}$$

For matched classes, we consider the class itself to find class-instance relations and if it returns no results with the adjacent terms, we replace the class by its instances. That means, for the example above, if neither the RDF query $(inst_j\ ?\ class_k)$ nor the RDF queries created based on t_2 and t_3 match existing triples, we replace $class_k$ by its instances and apply the possible RDF queries. If at least one of the executed RDF queries based on a term pair (t_a, t_b) results in existing triples from the knowledge base, t_a and t_b are marked as matched.

2nd level. At the second level, the found triples and until now unmatched query terms build the base for generating RDF queries along the same way as in the first level. This step is iterated until all new results and all still unmatched terms have been processed. Example: Assuming that the query $(inst_j\ prop_i\ ?)$ results in the triple $(inst_j\ prop_i\ inst_n)$ and the property $prop_l$ of t_3 is not marked as matched, the query $(inst_n\ prop_l\ ?)$ is executed. Embedded in this process, we discover and handle enumerations of instances, properties or classes, since we iterate over unmatched terms and found triples following the order of the query terms but considering the results of all terms in a set of found triples. This feature enables to answer queries like 'mail address, phone number and affiliation of the NEPOMUK project members.'

3rd level. The third level combines the found triples if possible. This includes as well the identification of connected triples, thus triples which build a coherent subgraph, as the identification of subgraphs which are seen as single results. In our example, we assume that the 1st and the 2nd level results are the triples:

$$(inst_j\ prop_i\ inst_n),\ (inst_n\ prop_l\ inst_m),\ (inst_j\ \texttt{rdf:type}\ class_k),$$

$$(inst_q\ prop_i\ inst_r),\ (inst_r\ prop_l\ inst_s),\ (inst_q\ \texttt{rdf:type}\ inst_r).$$

The first two triples have a joint instance, they belong together. The subject of the 3rd triple $inst_j$ is same as the subject of the 1st triple, *i. e.*, the three triples build a subgraph. The next three triples build a subgraph, too. There is a link between the two subgraphs since both instances $inst_j$ and $inst_q$ have the same class but we identify the both subgraphs as different results, the engine also outputs the two subgraphs.

Figure 2 demonstrates this process by means of the query 'phone number of the KM-Group secretary.' The numbers of edges state the order of executed steps.

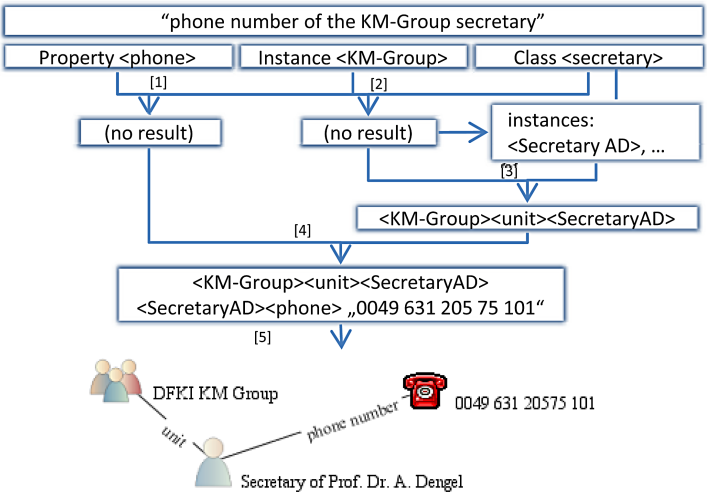

Fig. 2. Example for Semantic Teleporting

The process stops either when all query terms are matched or when there is no possibility to include all terms given that some term pairs do not lead to existing triples. The result consists of a set of instances and a set of triples which constitute the whole answer including semantic relations which explain why the result should be relevant to the query.

The ranking is based on the n-gram values, which result from the syntactic matching. Since our goal is to find the 'perfect answer', the number of involved query terms is used in the ranking function. Starting weight is the computed n-gram value of a query term and textual knowledge base content relating to the matched element. The weight of a matched triple is the sum of the n-gram weights of participating ontological elements. Each added triple increases the weight of the partial result by the appropriate n-gram value. Class-instance information is included by adding the weight of the class and assigning the appropriate query term to the partial result. Finally, the rank of a result is the sum of participating elements' n-gram weights divided by the number of query terms.

Resolving structural ambiguity is supported by the step by step, *i. e.*, triple to triple processing. Since we do not directly transform the natural language query to an RDF query, the triples found step by step lead to possible ways. Syntactic ambiguity is resolved in most cases by the same process. The ontological elements, which lead to existing triples, are higher ranked than elements that cannot be combined with other matched elements.

Compared to [8], our algorithm enables to make as many hops as required by the query and stops when all query terms are matched or there is no possibility to include further triples since no existing ones corresponds to the query. Furthermore, we include information about classes and involve their instances if the class itself does not lead to further results. The recognition of enumerations enables to answer queries which ask for multiple properties of multiple resources.

3.2 Semantic Document Retrieval Approach

The semantic document retrieval approach applies spreading activation to enhance the results by exploiting available domain knowledge. Before starting the description of our solution, it is helpful to explain spreading activation in general.

The basic idea behind spreading activation is to find more relevant information based on retrieved relevant information items (*e. g.*, results of traditional keyword search on the document index) by exploiting associations represented by semantic networks [6]. Spreading activation considers the knowledge base as a network structure, where the ontological concepts are the nodes and the properties are the edges. The edges are usually directed and weighted. The processing technique is an iteration of two main steps: pulses and termination check. A pulse is the incoming activation of a node, which propagates from a node to connected nodes along the edges by computing the outgoing activation, *i. e.*, the activation spreads through the network. The basic formula to compute the input I_j of the node j is:

$$I_j = \sum_i O_i w_{ij}$$

where O_i is the output of node i and w_{ij} is the assigned weight of the edge between node i and node j, thus the strength of the association between node i and node j. Usually, the activation level is the output of the node, determined by an output function of the input value:

$$O_i = f(I_j)$$

Commonly used functions for f are the threshold function, linear function, step function, or sigmoid function. The spreading activation results in the activation level of each node at the termination time. To avoid an uncontrolled flooding of the network, the application of constraints is needed, *e. g.*, distance constraint to limit the number of hops from the initially activated nodes or fan-out constraint to stop spreading at nodes with high connectivity since they often have a broad semantic meaning [6].

Spreading activation requires coupling the documents with the knowledge base, *i. e.*, the metadata of documents refers explicitly to instances of the knowledge base. Furthermore, the configuration of the network by choosing useful relations and corresponding weights (w_{ij}), defining adequate constraints, input and output function, requires domain knowledge.

We design our semantic network as follows: a default weight is assigned to each kind of relation w_{ij} and the graph is represented as a similarity matrix. Rows and columns are dedicated to the ontological concepts, w_{ij} to the weight of the connecting edge. The activation level of a node i is recorded in w_{ii}.

We use a Lucene[4] index for keyword matching on the document corpus. The query is expanded with the alternative names of knowledge base elements which are found by the meta data search. Alternatively, it is possible to index the textual content of the ontology with Lucene. In this case, the query modification is carried out implicitly during the spreading activation. Lucene delivers a ranked set of documents which match one or more terms of the expanded query. These documents are the initial activation points for spreading, inputs are the appropriate ranks. For this simple spreading activation, we extended the basic activation function with the factor α, so-called 'loss of energy' [19]:

$$I_j = \sum_i O_i w_{ij}(1 - \alpha)$$

Parameter α can be seen as an attenuation factor. It decreases the activation by factor α by each propagation from one node to a connected one. This feature is combined with an activation constraint, which stops spreading at a node when its activation level does not exceed a defined threshold. A fan-out constraint averts the danger of a too wide spreading through nodes with high connectivity, thus to become noise in results. At the same time, these constraints define the stop condition. It is important to assure that each edge is processed only once (in case of directed edges each direction is considered as one edge). The process stops when no more nodes have an activation level above the defined threshold or the nodes above the threshold have no pending edges. The activation level of nodes determines their ranks.

3.3 Combined Approach

In this section, we describe the combination of the fact retrieval and the document retrieval with spreading activation, shown in Fig. 3.

First, the fact retrieval is processed. If it results in nothing we process the semantic document retrieval (described in Sect. 3.2). If the user attempts teleporting and the engine delivers the 'perfect answer' no further steps are needed. We define the 'perfect answer' for a query with multiple terms/phrases as a set of results with a rank close to 1.0 where each result is composed of triples and the triples of one result match all query terms. In case of one term matching

[4] Lucene is a high-performance full text search engine library, *cf.* `http://lucene.apache.org/java/docs/`

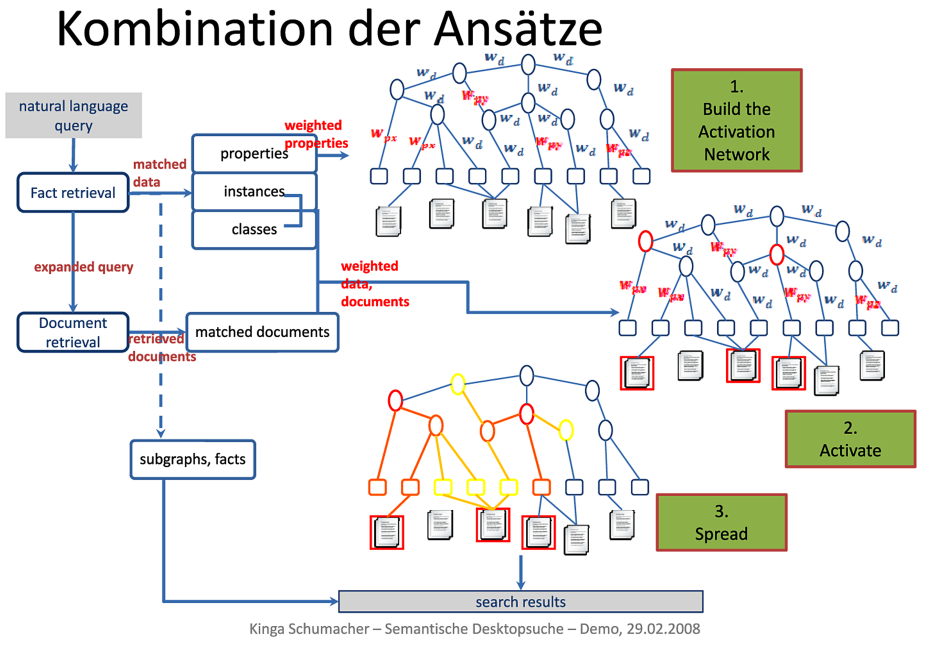

Fig. 3. Overview of the combined approach

a particular instance in the knowledge base, we assume that the spreading activation delivers helpful contextual information to the wanted resource. When no 'perfect answer' is found, we extract the following information from the fact retrieval results:

- instances, which are components of the results,
- alternative names, synonyms of the matched instances,
- matched properties.

The query is expanded with the alternative names of found instances. We chose this way of query expansion since it enables better to specify which documents are relevant. A graph-based query expansion would spread all resources associated with an activated instance which is suited for thesauri. Domain ontologies, which do not mainly describe linguistic relations between terms but support domain-specific relations between resources (instances, documents), require a very specific configuration of the spreading activation process. For example, the query 'spreading activation' on the Lucene index delivers the document 'Application of Spreading Activation Techniques in Information Retrieval' by Crestani. This document is categorized with the topics 'spreading activation', 'information retrieval' and 'semantic networks'. The activation of these topics results in documents which are on 'information retrieval' or on 'semantic networks', but not on 'spreading activation', the precision decreases. To avoid this noise, we should define additional constraints for categorization topics. Query expansion by found facts before spreading is also more effective.

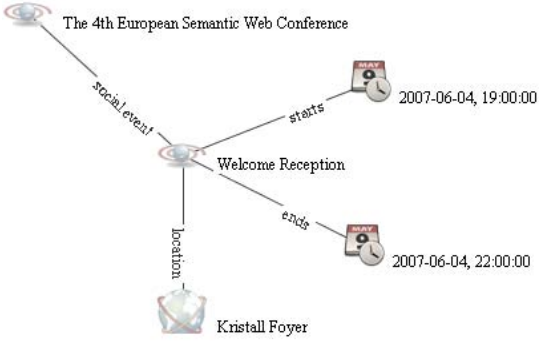

Fig. 4. Result graph of the query 'when and where is the welcome reception of the ESWC 2007'

Fig. 5. Presentation of a result composed of the first name, affiliation and paper of a person

After querying the document index we have all required information to create the network model and to start spreading activation. The network model includes all classes, instances and documents and involves all properties between instances, *i. e.*, all properties which link instances but not instances with literal values. These properties are marked as spreadable in the knowledge base and receive a low initial weight in the network model. We assign to the properties which are delivered by the fact retrieval their n-gram weight. We also assign to each edge the according weight, where some rules are carried out:

- directed relation from node i to node j, which has an inverse relation from node j to node i (*e. g.*, 'hasAuthor' – 'authorOf'): assigning the weight from i to j
- directed relation without an inverse: assign the weight in both directions
- instance-class relation (`rdf:type`): assign the weight from class to instance

The second point makes sure that relations without an inverse are involved if one of the connected instances is activated. The third condition helps to avoid spreading from one instance to all other instances of a particular type but enables to identify all instances of a class, if a class is initially activated.

The initial activation points are the instances, classes of the fact retrieval results and documents found by keyword search. The strength of activation

corresponds to the weights and ranks which the elements have. The spreading activation process complies with the one described in Sect. 3.2 using an activation constraint and a fan-out constraint. It results in a ranked set composed of instances of the knowledge base and related documents.

We visualize the 'perfect answer' as a graph[5] with associated symbols for persons, projects, email, phone, event, location, date *etc.*, in order to enable the user to recognize the result immediately. Furthermore, a graph shows at same time the explanation of the result, see Fig. 4. Other results are shown as items in a list according to their ranks and enriched with the requested information. Properties which are matched by the fact retrieval and which have a literal value are added to the appropriate results. The matched properties which connect two instances are used to group the instances and documents which belongs together. Figure 5 shows an example.

4 Evaluation

As there is no standardized and annotated test data set for Semantic Desktop evaluations yet [3], we used the ESWC 2007 [18] data for our evaluations which has a data structure similar to the PIMO. The ESWC conference knowledge base describes the conference including information about people, talks, papers, posters, conference events, it also involves abstract concepts and references to documents like the Semantic Desktop.

We extended the knowledge base with some synonyms of the ontological elements and instances and created the index of the document set using Lucene. The evaluation is based on 11 queries carried out with our semantic search engine and Google site search using '$<$ *query* $>$ site:`://www.eswc2007.org/`'. We computed the precision of the results, shown in Fig. 6. The last column states the results as 'perfect answer' or not, *i. e.*, subgraph(s) which accomplish the conditions of a 'perfect answer' (see Sect. 3.3) found by the fact retrieval engine. Since the semantic search engine applies additional information as ontological elements and instances it is not possible to build a ground truth for this evaluation. For instance, a session is included in the knowledge base but there is no document on the ESWC 2007 web sites about a single session. For this reason, recall, f-measure of the results cannot be determined.

The results demonstrate clearly the power of our combined approach, it returns precise results to extensive queries by exploiting additional information from the knowledge base. The additional results for 'RDF' compared with the Google site results are caused partly by the expansion of the query with 'Resource Description Framework' which enhance the results of the document retrieval and by the found instances through our combined approach, *e. g.*, sessions with topic 'RDF'. In case of simple queries like 'social networks', the precision of the semantic search engine is lower since the engine results documents which

[5] The graph visualization was developed by Björn Forcher (DFKI) and is based upon the Java Universal Network/Graph Framework and the Batik SVG Toolkit, `http://jung.sourceforge.net/` and `http://xmlgraphics.apache.org/batik/`

QUERY	Google site search		Semantic desktop search		
	No. of results	precision	No. of results	precision	Perfect answer
RDF	55	1.0	77	1.0	
Social networks	13	1.0	24	0.58	
Sintek	11	1.0	15	0.8	
System descriptions	0	0.0	10	1.0	yes
Organizer of workshop 5	0	0.0	4	1.0	yes
Invited talks	0	0.0	4	1.0	yes
Semantic search demos	13	0.077	2	1.0	yes
Papers from Voelker	2	0.5	2	1.0	yes
Authors and abstracts of reasoning papers	21	1.0	1	1.0	yes
DFKI members at the ESWC 2007	1	0.0	6	1.0	yes
When and where is the welcome reception	2	0.5	1	1.0	yes
Average precision		**0.4615**		**0.9436**	

Fig. 6. Table of results

were presented in the same session as the relevant documents. The additionally found documents are related since the sessions are organized based on topics, but not relevant to the query. This result demonstrates the failing of spreading activation: if there are no certain properties matched against the query and/or only documents are matched, the spreading activation propagates to all adjacent nodes with the same intensity. For some queries it is helpful for the user to have this additional information based on the ontological context of found resources. In all further evaluated cases, the semantic search engine delivered precise answers. The query 'authors and abstract of reasoning papers' is, compared to the Google site results, a special case. It results in one document with the keyword 'reasoning' incl. its abstract and authors. Since it is a 'perfect answer', no document retrieval is processed and so all other documents containing the term but not focused on 'reasoning' are not returned. Since the engine is configured to search the semantic desktop and to response as precise answers as possible, we consider this result as actually the 'perfect answer' but we enable the user to inquire more information, thus the results of the complete search process.

The benefit of our semantic search is that it supports the user with precise information (facts) to extensive queries. Furthermore, the user receives useful additional information to the found resources, *e. g.*, not only the title of a paper and the appropriate text snippet, but also the authors and keywords. Furthermore, the ranking which combines the ranks from the fact retrieval and the ranks from document retrieval delivers a good order of the results, since the found facts support more precise information about documents and instances of the knowledge base. The example query 'RDF' returns at the top of the result list papers and sessions with this topic, *i. e.*, with the keyword 'RDF', thus papers which focus on 'RDF'. A document retrieval just delivers documents that contain this term and ranks them based on the number of the query term's frequency.

5 Conclusions and Future Work

This paper describes a semantic search approach which combines fact retrieval and document retrieval with spreading activation in order to exploit all available information on a Semantic Desktop. The developed semantic search engine works with natural language queries and supports not only a semantic document search but also *semantic teleporting*. We evaluated our approach based on the ESWC 2007 knowledge base and documents comparing the results of our engine with the results of Google Site search since there is no standardized test data set for Semantic Desktop or Semantic Desktop search evaluations. The evaluation results demonstrate the power of our combined approach; the engine returns precise results to extensive queries by exploiting facts, metadata of documents from the knowledge base, *i. e.*, it is appropriate to search for information on the Semantic Desktop in a goal-directed way.

For future work, we plan to extend our approach with machine learning technology in order to learn the network model's edge-weights by exploiting user feedback. We assume that the user is often interested in specific information about an instance or in a specific aspect of a document. Using relevance feedback, both explicitly by the user and implicitly by user observation, allows the adaption of the weights according to the user's needs, *i. e.*, to personalize the search on the Semantic Desktop.

Acknowledgements. Part of this work has been supported by the Rheinland-Pfalz cluster of excellence "Dependable adaptive systems and mathematical modeling" DASMOD, project ADIB,[6] and by the European Union IST fund (Grant FP6-027705, Project NEPOMUK[7]). We also want to thank the various developers of Gnowsis[8], Aperture and Nepomuk.

References

1. Berger, H., Dittenbach, M., Merkl, D.: An adaptive information retrieval system based on associative networks. In: Proc. of the first Asian-Pacific Conference on Conceptual Modelling, pp. 27–36 (2004)
2. Celino, I., Turati, A., Valle, E.D., Cerizza, D.: Squiggle – a semantic search engine at work. In: Proc. of the 4th European Semantic Web Conference (2007)
3. Chernov, S., Serdyukov, P., Chirita, P.-A., Demartini, G., Nejdl, W.: Building a desktop search test-bed. In: Proc. of the 29th European Conference on Information Retrieval (2007)
4. Chirita, P.-A., Costache, S., Nejdl, W., Paiu, R.: Beagle^{++}: Semantically enhanced searching and ranking on the desktop. In: Proc. of the 3rd European Semantic Web Conference, pp. 348–362 (2006)
5. Chirita, P.-A., Gavriloaie, R., Ghita, S., Nejdl, W., Paiu, R.: Activity based metadata for semantic desktop search. In: Proc. of the 2nd European Semantic Web Conference, pp. 439–454 (2005)

[6] `http://www.dasmod.de/twiki/bin/view/DASMOD/ADIB`
[7] `http://nepomuk.semanticdesktop.org/`
[8] `http://www.gnowsis.org/`

6. Crestani, F.: Application of spreading activation techniques in information retrieval. Artificial Intelligence Review 11(6), 453–482 (1997)
7. Decker, S., Frank, M.R.: The networked semantic desktop. In: WWW Workshop on Application Design, Development and Implementation Issues in the Semantic Web (2004)
8. Goldschmidt, D.E., Krishnamoorthy, M.: Architecting a search engine for the semantic web. In: Proc. of the AAAI Workshop on Contexts and Ontologies: Theory, Practice and Applications (2005)
9. Guha, R., McCool, R., Miller, E.: Semantic search. In: Proc. of the 12th International Conference on Word Wide Web (2003)
10. Hildebrand, M., Ossenbruggen, J., van Hardman, L.: An analysis of search-based user interaction on the semantic web. Report, CWI, Amsterdam, Holland (2007)
11. Hindle, D., Rooth, M.: Structural ambiguity and lexical relations. Computational Linguistics 19(6), 103–120 (1993)
12. Iofciu, T., Kohlschütter, C., Nejdl, W., Paiu, R.: Keywords and rdf fragments: Integrating metadata and full-text search in beagle++. In: Proc. of Semantic Desktop Workshop at the International Semantic Web Conference, vol. 175 (2005)
13. Karger, D.R., Bakshi, K., Huynh, D., Quan, D., Sinha, V.: Haystack: A customizable general-purpose information management tool for end users of semistructured data. In: Proc. of the 2nd Conference on Innovative Data Systems Research (2005)
14. Lei, Y., Uren, V.S., Motta, E.: Semsearch: A search engine for the semantic web. In: Proc. of the 15th International Conference on Knowledge Engineering and Knowledge Management, pp. 238–245 (2006)
15. Mäkelä, E.: Survey of semantic search research. In: Proc. of the Seminar on Knowledge Management on the Semantic Web (2005)
16. Mangold, C.: A survey and classification of semantic search approaches. International Journal of Metadata, Semantics and Ontologies 2(1), 23–34 (2007)
17. Manning, C.D., Schütze, H.: Foundations of statistical natural language processing. MIT Press, Cambridge (1999)
18. Möller, K., Heath, T., Handschuh, S., Domingue, J.: Recipes for semantic web dog food - the eswc and iswc metadata projects. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ISWC 2007. LNCS, vol. 4825, pp. 802–815. Springer, Heidelberg (2007)
19. Rocha, C., Schwabe, D., Aragao, M.P.: A hybrid approach for searching in the semantic web. In: Proc. of the 13th International Conference on World Wide Web, pp. 374–383 (2004)
20. Sauermann, L., Bernardi, A., Dengel, A.: Overview and outlook on the semantic desktop. In: Proc. of the Semantic Desktop Workshop at the 4th International Semantic Web Conference, vol. 175 (2005)
21. Sauermann, L., van Elst, L., Dengel, A.: Pimo – a framework for representing personal information models. In: Proc. of the I-SEMANTICS 2007, pp. 270–277 (2007)
22. Scheir, P., Ghidini, C., Lindstaedt, S.N.: Improving search on the semantic desktop using associative retrieval techniques. In: Proc. of the I-Semantics 2007, pp. 415–422 (2007)
23. Teevan, J., Alvarado, C., Ackerman, M.S., Karger, D.R.: The perfect search engine is not enough: a study of orienteering behavior in directed search. In: Proc. of the SIGCHI conference on Human factors in computing systems, pp. 415–422 (2004)