SemSearchPro – Using semantics throughout the search process

Thanh Tran*, Daniel M. Herzig, Günter Ladwig

Institute AIFB, Karlsruhe Institute of Technology, D-76128 Karlsruhe, Germany

1. Introduction

Since its start in the early 1990s, the World Wide Web has exponentially grown to more than 100 million active websites and continues to grow further. Recently, also the amount of semantic data is rapidly increasing. Billion of triples1 can be found on the Web today, mostly as RDFa data associated with Web pages or as RDF data published and maintained by the Linked Open Data project.

Keyword search engines became a popular way for searching information on the Web, with Google being the dominant player among the search engine providers. However, there is much more potential to exploit than what is harnessed by Web search engines today, especially because of the developing Web of data. Motivated by economic incentives, start-ups like Cuii,2 Hakia,3 and the bigger rivals like Microsoft and Yahoo challenge Google with new solutions to semantic search, and get wide spread attention even in mainstream media. All challengers emphasize one claim: that they are able to deliver more precise search results.

Delivering better results and addressing more complex information needs have also been the main objectives of the semantic search research community, which has grown substantially in the past years. Resulting from this trend is a large body of research on different concepts and techniques for semantic search.

[13,28,39]. In the information retrieval (IR) community, semantic models have been proposed to obtain richer representations of queries and resources [7,4]. This line of research is mainly focused on proving the hypothesis that semantically richer models will ultimately allow for a more precise matching of queries against resources. While this research is focused on documents, many approaches also considered as semantic search are concerned with the matching of queries against semantic data [39].

To collate these various approaches and to better understand what the concept of semantic search entails, we study semantic search under a general model. Extending this model, we introduce the notion of process-based semantic search, where semantics is exploited not only for query processing, but might be involved in all steps of the search process. We propose a particular approach that instantiates this process-based model. The usefulness of using semantics throughout the search process is finally assessed via a task-based evaluation performed in a real world scenario.

© 2011 Elsevier B.V. All rights reserved.

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Semantic search
Semantic search process
Ontology-based document retrieval
Semantic data retrieval

ABSTRACT

Semantic search attempts to go beyond the current state of the art in information access by addressing information needs on the semantic level, i.e. considering the meaning of users’ queries and the available resources. In recent years, there have been significant advances in developing and applying semantic technologies to the problem of semantic search. To collate these various approaches and to better understand what the concept of semantic search entails, we study semantic search under a general model. Extending this model, we introduce the notion of process-based semantic search, where semantics is exploited not only for query processing, but might be involved in all steps of the search process. We propose a particular approach that instantiates this process-based model. The usefulness of using semantics throughout the search process is finally assessed via a task-based evaluation performed in a real world scenario.

© 2011 Elsevier B.V. All rights reserved.
refinement. Namely, it enables efficient translation of keywords to structured queries, thus allowing users to construct complex queries using simple list of keywords. It is used to guide the search during the matching to focus on relevant candidates and ultimately, to improve efficiency. Further, this model is the basis for designing accessible presentation elements, and for the automatic selection of appropriate ones, given the current results. Also for query refinement, it is used to derive facets.

We evaluated the use of semantics throughout this process, both separately, considering the performance of the individual steps, and holistically, considering the process as a whole. We showed that using semantics to guide and to reduce the search space of the translation and the matching tasks substantially increase performance. We achieved performance comparable to the state of the art on keyword search, while reducing the amount of paths that have to be materialized and indexed. For matching, 5–7 times faster performance was achieved. The task-based user evaluation targeted at the search process also yielded promising results, showing that most users could solve the majority of the evaluation tasks. The performance of the task-based user state of the art on keyword search, while reducing the amount of paths that have to be materialized and indexed. For matching, 5–7 times faster performance was achieved. The task-based user evaluation targeted at the search process also yielded promising results, showing that most users could solve the majority of the tasks in reasonable time.

Outline: This paper is organized as follows. In Section 2, we define the general semantic model with all its components involved in the semantic search process. Moreover, we describe the state of the art and give our view on process-based semantic search. In Section 3 the semantics of queries and resources as well as the semantic model are described. The concepts and techniques applied in our approach are presented in Section 4. Implementations realizing the process are described in Section 5. In Section 6, we present the evaluation and finally we conclude in Section 7.

2. Semantic search

The term “semantic” has the tendency to become a buzzword as many applications claim to be “semantic” or to feature “semantic search”. There exist many different conceptions and definitions for semantic search [13,45,7]. Semantic search from the information retrieval (IR) point of view [4] is different from the understanding in the Semantic Web community.

Central to all semantic search approaches proposed so far is the use of a semantic model. A basic retrieval model that captures the main idea of existing approaches can be formalized as follows:

Definition 1 (Basic semantic search model). The basic semantic search model is a quadruple \((R, Q, S, M((Q, R, S)))\) where

(i) \(R\) is the resource model, which is a set of syntactic representations for the underlying resources.

(ii) \(Q\) is the query model, which is a set of syntactic elements representing the user information needs.

(iii) \(S\) is the semantic model capturing the meaning of the represented information needs and resources.

(iv) \(M((Q, R, S))\) is the semantic matching framework which models the relationship between resource representation and query representation. It specifies the notion of matching, which incorporates the associated semantic model to compute whether the resource representation contains an answer to the query representation. In particular, there is a matching function \(M: Q \times R \times S \rightarrow \{0, 1\}\), which for a query representation \(q \in Q\), a resource representation \(r \in R\) as well as the associated semantic model \(S\), outputs the degree to which \(r\) is a result to \(q\).

In the following, we discuss several kinds of models that capture semantics in different ways. Based on that, we discuss the main directions of semantic search with respect to the semantic models being used. Finally, we discuss the notion of semantic search suggested by Hildebrand [21], which is the one we follow in this paper. In particular, we provide a formal model of process-based semantic search where semantics is not only used for matching but throughout the retrieval process.

2.1. Semantic models

Generally, semantics is concerned with the meaning of things. Meaning is established through a semantic model, which commonly captures interrelationships between elements and their interpretations. Various semantic models have been proposed and used in different research communities. There are linguistic models such as thesaurus that capture relations between syntactical elements. In the database community, conceptual models such as Entity Relationship diagrams are used to capture relations between entities [5]. Thus, while linguistic models are concerned with meanings at the level of words, conceptual models more specifically deal with meanings at the level of real-world entities denoted by words. That is, conceptual models deal with interpretation of words in terms of real world entities the words refer to. There are formal (logic-based) conceptual models where interpretations are precise and computable [35].

In the Semantic Web community, ontologies have received widespread acceptance. The notion of ontologies employed by this community is very general. Ontologies constitute a family of models, which might differ in the degree of expressivity and formality, ranging from simple taxonomies and shallow conceptual models represented in RDF(S) to expressive formal models represented in Description Logics [35]. Commonly, ontologies comprise not only the conceptual part but also instances. They are more similar to databases (and knowledge bases) where the conceptual part corresponds to what is referred to as the schema (terminological knowledge) and instances correspond to the actual data of the database (assertional knowledge of the knowledge base).

In this paper, semantic models refer to the family of models capturing the conceptualization but not the actual instantiation. This family of model includes linguistic models, conceptual models as well as formal models varying in the degree of expressivity and formality. We now provide a general working definition:

Definition 2 (Semantic models). Semantic models represent the family of models providing a conceptualization of the universe of discourse. A semantic model is a tuple \((\mathcal{E}, \mathcal{P})\), where \(\mathcal{E}\) and \(\mathcal{P}\) are syntactic elements denoting real-world entities and relations between them, respectively. The semantics of a model is established through interpretation, which is a mapping of syntactic elements to real world entities and their relations. A semantic model is express when there are many modeling constructs to specify syntactic elements. A semantic model is formal when there is a defined interpretation function \(I\) based on which interpretations of syntactic elements become computable.

Note that according to the definition, semantics is not always computable but might have to be established by hand, i.e. interpretation of the model might have to be performed by the human worker. Also, computability here is not to be confused with decidability. In particular, there might be no effective method for computing interpretations of a formal semantic model, i.e. it is undecidable.

However, we stated that, if a formal model is present, in the sense that there is a well defined interpretation function, then it is possible to compute the interpretation of syntactic constructs.
2.2. State of the art

There exist many approaches that comply to the basic notion of semantic search discussed previously. They can be categorized according to the types of results returned to the users, namely semantic data and semantic document retrieval.

Semantic data retrieval: In the Semantic Web community, data in RDF, which might be associated with conceptualization available as RDFS or OWL ontologies, is commonly referred to as semantic data. Approaches which deal with search and retrieval of this data are considered semantic search. More specifically, $\mathcal{R}$ is RDF data, $\mathcal{Q}$ is often SPARQL or a subset of this W3C RDF query language, and $\mathcal{S}$ is an RDFS or OWL ontology. An overview of this type of semantic search can be found in the survey conducted by Uren et al. [39]. Recently, a number of Semantic Web search engines such as Hermes [37], Watson [8], SWSE [18], FalconS [6] and Sindice [30] have been developed for retrieving semantic data on the Web. These engines primarily rely on keywords as a means for specifying queries. The exceptions are Hermes and SWSE, which support more advanced querying capabilities, including basic SPARQL graph patterns. The semantic matching framework $\mathcal{M}$ implemented by these systems amounts to matching graph patterns against RDF data and, when applicable, to use the formal semantics captured by the associated RDFS/OWL ontologies for reasoning (to infer further statements). This kind of semantic matching is also implemented by RDF stores such as Virtuoso, Sesame and RDF-3X. An important part of the matching framework is ranking, i.e. to output not only the matches but also the degree of matching. While RDF stores primarily focus on data storage and efficient querying, Semantic Web search engines strongly rely on ranking. Since these engines are built for dealing with large amounts of Web data that currently, are often of low quality, it is essential to have a mechanism in place that helps to identify and focus on the part of the data that is relevant. Since many search engines have been built on top of an IR system (such as Lucene), the score returned by the underlying keyword search engine is often used as the basis for ranking [30,6,37]). Depending on the IR model in place, the computation of this score might incorporate different aspects of the query and resources. One popular aspect that is typically supported is the discriminative quality of terms measured using TF-IDF. Recently, also the authority of resources, a concept well-studied in IR research (e.g. HITS [26], PageRank [3]) and successfully applied by commercial Web search engines, is adopted for ranking Semantic Web data [17].

Semantic document retrieval: Different to this paradigm, the IR community views semantic search under the perspective of document retrieval. Here, $\mathcal{R}$ is typically the bag-of-word model (i.e. a list of terms contained in the documents), $\mathcal{Q}$ is also term-based (i.e. a list of keywords) and $\mathcal{S}$ includes thesauri, taxonomies and formal ontologies. The resource representation is typically a bag of words model. Semantic models are used here to interpret the meaning of words. Traditionally, linguistic models such as thesauri and taxonomies have been used. For instance, WordNet has been used for the disambiguation of terms [41], and Roget’s thesaurus is used for query construction and expansion [23]. Also, conceptual models in the form of semantic networks have been suggested for relevance propagation, assisted navigation strategies, and query formulation [34].

The need for going beyond the matching purely based on words and statistical dependencies that exist between them has been clearly expressed by leading researchers in the field. In the early years of IR research, Riksbergen already argued for the use of semantics. In order to return precise results to the user, it is essential to compute whether the resources semantically entail what the user asks for [40]. For this, expressive formal semantic models are needed. Such a model is hard to construct and maintain, which so far, seems to be the major drawback of this kind of approaches. However, the IR community seems to embrace the vast body of explicit knowledge that increasingly, is made publicly available. There are approaches which use ontologies [4] and large amount of explicit knowledge to improve the precision of search [7].

Process-based semantic search: From a process point of view, the different approaches to semantic search discussed previously focus on matching, whereas the other steps play less prominent roles. In semantic data retrieval, the semantic model provides the theoretical framework for defining and computing matches (i.e. to produce answers that match the semantics of the query), and also, might be used to infer further results. In document retrieval, semantics has been mainly used to disambiguate terms, and ultimately, to improve the quality of matching. In this paper, we follow the model of process-oriented semantic search, which goes beyond matching to exploit semantics also for the additional steps of query construction and result presentation part of the retrieval process:

Definition 3 (Process-oriented semantic search model). The process-oriented semantic search model is a tuple $(\mathcal{R}_s, \mathcal{R}_p, \mathcal{Q}_s, \mathcal{Q}_p, \mathcal{S}, \mathcal{M}(\mathcal{Q}_s, \mathcal{R}_s, \mathcal{S}), \mathcal{T}(\mathcal{Q}_s, \mathcal{Q}_p, \mathcal{S}), \mathcal{P}_s(\mathcal{R}_p, \mathcal{R}_p), \mathcal{P}_p(\mathcal{Q}_s, \mathcal{Q}_p))$ where

(i) $\mathcal{R}_s$ is the system-resource model, which is a set of syntactic elements constituting the internal representation of the system’s resources.
(ii) $\mathcal{R}_p$ is the presentation-resource model, which is a set of presentation elements constituting the external representation of the system’s resources.
(iii) $\mathcal{Q}_s$ is the user-query model, which is a set of elements representing the information needs as provided by the user.
(iv) $\mathcal{Q}_p$ is the system-query model, which is a set of elements constituting the internal representation of the user information need employed by the system.
(v) $\mathcal{Q}_p$ is the presentation-query model, which is a set of elements constituting the external representation of the information need provided by the system.
(vi) Just like in the basic model, $\mathcal{S}$ is the semantic model capturing the meaning of query and resources. $\mathcal{M}(\mathcal{Q}_s, \mathcal{R}_s, \mathcal{S})$ is the semantic matching framework that models the relationship between resource representation $\mathcal{R}_s$ and query representation $\mathcal{Q}_s$ and specifically, specifies the notion of matching under consideration of the semantic model $\mathcal{S}$.
(vii) $\mathcal{T}(\mathcal{Q}_s, \mathcal{Q}_p, \mathcal{S})$ is the semantic translation framework which models the relationship between user-query representation $\mathcal{Q}_s$ and system-query representation $\mathcal{Q}_p$. In particular, there is a translation function $\mathcal{T} : \mathcal{Q}_s \times \mathcal{Q}_p \times \mathcal{S} \rightarrow \{0, 1\}$, which for the given query $\mathcal{q}_s \in \mathcal{Q}_s$ provided by the user, an internal representation in $\mathcal{q}_s \in \mathcal{Q}_s$ employed by the system, as well as the associated semantic model $\mathcal{S}$, outputs the degree to which $\mathcal{q}_s$ is a translation of $\mathcal{q}_p$.
(viii) $\mathcal{P}_s, \mathcal{P}_p$ are the semantic presentation frameworks which model the relationship between internal representation and external representation of queries and resources, i.e. $\mathcal{Q}_s, \mathcal{Q}_p$ and $\mathcal{R}_s, \mathcal{R}_p$. Under consideration of the semantic model $\mathcal{S}$, it specifies the mapping of syntactic elements in $\mathcal{R}_s$, $\mathcal{Q}_s$ to presentation elements in $\mathcal{R}_p, \mathcal{Q}_p$. 

---

4 http://virtuoso.openlinksw.com/
6 http://www.mpi-inf.mpg.de/neumann/rdf3x/.
7 Please refer to literature on the Pedantic Web (http://pedantic-web.org/ September 08, 2011) to find more details about this quality issue of Web data.
Besides the main approaches for semantic document and data retrieval discussed previously, there are more focused approaches specifically dedicated to some steps involved in the search process. For supporting query construction, there is a large body of work on Natural Language (NL) search interfaces [2]. A popular example of question answering systems, which provides a NL interface (in the sense that the system can process NL inputs) is IBM’s Watson.8 The database community has been investigating the use of keywords for search [38,24,22,1]. Unlike keyword search used on the Web, which focuses on simple needs, the keyword search elaborated here is used to obtain more complex results. Instead of a single set of resources, the goal is to compute complex sets of resources and their relations. Rather than computing results directly, candidate interpretations in the form of structured queries have been proposed [38]. Given the provided keywords, completions in the form of queries are presented from which the user chooses the intended one to obtain the results in a subsequent step. Since both the NL and keyword approaches involve the transformation of user provided inputs to an internal representation, they can be seen as implementations of the translation framework presented above. In particular, since these approaches leverage the semantics of the underlying resources for computing meaningful queries, they in fact implement the concept of semantic translation.

Typically, the interface for inputting the query is the same for presenting the query ($Q_U = Q_P$). Thus, all popular search interfaces are also candidate presentation interfaces. NL and graphical interfaces [12] have proven to be useful in specific domains. More widely used are keyword and form-based interfaces such as the ones used in commercial websites like Amazon and eBay. We note that in today’s search engines, $Q_U = Q_P$, also holds also for approaches such as NL or keyword search where the internal representation of the user need is clearly different from the user query model ($Q_U \neq Q_P$). Providing the translation from $Q_U$ to $Q_P$ facilitates query construction. However, the gap between these two models might be large, which might result in translation results that are suboptimal. The use of an additional presentation model that is intuitively assessable, yet more similar to the system query model, can address this issue. Different than the user query model, this one does not directly address the construction, but focus on the aspects of presentation and refinement. Thus, it might be harder to construct, but is more appropriate for facilitating comprehension and refinement of results. A paradigm implementing this idea which gains momentum is faceted search [9,33]. For a quick start, users enter keywords to obtain the initial set of results. Facets are then computed, based on which the user can explore and iteratively refine the results. Facets are in fact description of the current result set. Thus, the computation of facets have to take the semantics of the resources into account in order to produce meaningful descriptions. Typically, they are derived from the data schema. Popular implementations are Exhibit910 and Parallax.11

Less research work can be found on using semantics for the presentation of results. In practice, presentation components are always built for some specific types of results. In doing so, the engineers clearly have to take the semantics of these results into account. For instance, specific presentation elements have to be designed, given that results to be displayed are “people”. However, there are no generic approaches, which leverage the semantics explicitly given for resources to manage the presentation of results in a more mechanized fashion. One direction towards this is Frensel [32], which is a vocabulary that can be used to define presentation related aspects of RDF resources.

2.3. SemSearchPro – process-based semantic search

So far, we have discussed approaches that deal with the main problem of semantic matching (for data and document retrieval). Also, we elaborated on specialized systems that focus on query construction and result presentation. Different to all these approaches, the approach we present now targets the use of semantics throughout the entire process. It fully implements the process-based semantic search model. In doing so, work presented previously at conferences and specifically focused on different steps of the process is brought together. Central to the integration, and to the core functionality of these pieces of work is a lightweight semantic model. It is used to implement the translation approach proposed in [38]. For implementing the matching framework, work presented in [36] is used. The semantic model is used to improve SPARQL graph pattern matching. Essentially, semantics is exploited to guide the pruning of the answer search space. Finally, the same semantic model is used for defining different types of presentation mappings, based on which appropriate presentation modules can be automatically determined for the given results.

3. Semantics of queries and resources

In this section, we present the models underlying our semantic search approach called SemSearchPro. We will first discuss the internal representation of the underlying resources. Then, we introduce the formal query model used to represent information needs. Finally, we elaborate on the semantics of queries and resources.

3.1. System resource model

Resources are represented using a general graph-structured data model:

**Definition 4 (System resource model).** A system resource model is a graph \( R_d(V^R, E^R, V^E) \) where

- \( V^R \) is a finite set of vertices. Thereby, \( V^R \) is conceived as the disjoint union \( V^R_1 \cup V^R_2 \) with E-vertices \( V^R_1 \) (representing entities) and V-vertices \( V^R_2 \) (data values).
- \( E^R \) is a finite set of edge labels, subdivided by \( L_e = L_e \cup L_a \), where \( L_e \) are relation labels and \( L_a \) are attribute labels.
- \( E^R \) is a finite set of edges of the form \( e(v_1, v_2) \) with \( v_1, v_2 \in V \) and \( e \in L_e \). Moreover, the following types are distinguished:
  - \( e \in L_e \) (A-edge) if and only if \( v_1 \in V^R_1 \) and \( v_2 \in V^R_2 \) and
  - \( e \in L_a \) (R-edge) if and only if \( v_1, v_2 \in V^R_2 \).

**Example 1.** An example resource graph describing several entities is illustrated in Fig. 1. In particular, it says that per2 with name John McCarthy works at Stanford University, is author of artic2 and won the Turing Award.

![Fig. 1. A graph structured resource model.](image-url)
The presented graph-structured model is similar to that of RDF. The intuitive mapping from RDF to this is, resources correspond to entities, properties to either relations or attributes and literals to data values.

A general graph-structured model like this one captures not only RDF but also Web documents, XML as well as relational data. In particular, this model subsumes the tree-structured XML data model. Relational data can be represented as a graph-structured data, e.g. by mapping relations to vertices and foreign keys to edges. Likewise, Web documents can be represented as vertices and hyperlinks connecting them can be modeled as edges. Note that in fact, resources denoted by \( V_i \) might be any real world entities. Documents are not explicitly distinguished from other types of entities. Just like other entities, documents might have relations to other entities such as author, publisher and special attributes such as title and abstract. This is illustrated in the example where \( \text{artic2} \) is also represented as an E-vertex. Thus, documents are indexed and searched in the same way like other types of resources such that retrieval of documents as well as information about other types of entities boils down to one and the same task, namely the one of entity retrieval.

Note that this resource model is a conceptual representation of the data internally used by the system. In the physical implementation, there are different ways to manage and store graph-structured data like this. Databases (for managing graphs) such as RDF extensions to Oracle and IBM DB2, stores specifically built for RDF such as RDF-3X [29] and YARS [19], as well as IR-based technologies have been suggested [44]. For fast data access and for supporting search and ranking, specific indexes might be employed. Indexes might be created to support different lookup patterns [19,43]. For entities representing documents (or any types of entities that are associated with text), an inverted index might be used for supporting keyword lookup and search [44].

### 3.2 System query model

Internally, the information need of the users are represented as a particular type of conjunctive queries:

**Definition 5 (Conjunctive query).** A conjunctive query \( Q_2 \) is an expression of the form \( \exists x_1, \ldots, x_n. \bigwedge_{i=1}^{m} P_i \wedge \bigwedge_{j=1}^{n} \forall x_j, \bigwedge_{k=1}^{m} \neg P_k \), where \( x_1, \ldots, x_n \) are called distinguished variables, \( x_{n+1}, \ldots, x_{m} \) are undistinguished variables and \( P_1, \ldots, P_m \) are query atoms. These atoms are of the form \( \langle p, v_1, v_2 \rangle \), where \( p \) is a predicate symbol, \( v_1, v_2 \) are variables or, otherwise, are called constants.

**Example 2.** An example query is \( (x,y,z,u). \text{prizes}(y,z) \land \text{label}(z, \text{Turing Award}) \land \text{author}(y,u) \land \text{type}(u, \text{Article}) \land \text{employment}(y,x) \land \text{name}(x, \text{Stanford University}) \). This query is illustrated in Fig. 2. It asks for \( y \) working at place \( x \) called \text{Stanford University}, which author articles \( u \) and have won a \text{Turing Award}. All variables are distinguished such that all bindings to \( x, y, z, u \) will be returned as answers.

![Fig. 2. A graph-structured query model.](image)

Note that this query model is a restricted type of general conjunctive queries, a form of first-order queries. As opposed to the general formalism, query atoms are atomic formulas that draw exclusively from the set of predicates symbols. More complex formulas are allowed as query atoms in the general model, which might be construct using symbols such as equality, negation and quantifiers. The general model is well-studied in the literature, as it is capable of expressing a large portion of relational queries (relational algebra). The vast majority of query languages used in practice fall into this fragment, including large parts of SQL. The specific model used here has practical relevance because it is computationally more tractable. At the same time, it covers a frequently used feature of SPARQL, i.e. the basic graph pattern fragment of this query language. We will now discuss the three common information needs that can be represented with this model:

- **Entity search:** In the IR community, this is also commonly known as navigational search. It is typically used as a point to the system, which is an entity such as a product, a Web page or a document in the collection. The user already knows the existence of the entity and uses the search function as a shortcut to this. Since the result is more or less known, the information need is expressed rather precisely, often with terms particular to the entity. The part \( (x). \exists x. \text{name}(x, \text{Stanford University}) \) of the example query for instance, represents by itself a query referring specific to an entity that has name \text{Stanford University}.

- **Fact search:** This refers to situations where the user is interested in a certain fact, like a phone number of a friend or the current temperature in San Francisco. While entity search involves one or several entities as results (E-vertices), this kind of search produces facts in the form of specific attribute values (V-vertices). Also, it is different to entity search in that it is not a navigational search for a known item, but rather, the purpose is to find unknown information. Thus, it is also referred to as informational search.

- **Relation queries:** This is another type of information search, where the goal is to gather not only information about a specific entity, but to find out complex set of entities, and especially, how they are related. The query example discussed previously belongs to this type, asking for relations between \( x, y, z, \) and \( u \).

### 3.3 Semantic model

We will now discuss the semantics of queries and resources. The proposed query model is a fragment of conjunctive queries, which in general belongs to the class of first order queries. Thus, the queries have standard first order semantics. Precisely, every query is interpreted as a first order formulae that is constructed from atomic formulae using conjunction and existential quantification. The formal semantics of resources can be established by mapping elements of the resource model to first order logic. Since the semantic model we are interested in refers to the conceptual part, rather than the instances, we will now focus on the conceptualization of resources and discuss its formal first order semantics.

The conceptualization of resources is built upon the basic notion of classes, class attributes and class relations. Basically, a class denotes a group of instances that commonly exhibit the same types of relations and attributes. This conceptual knowledge about entity types can be explicitly defined in the following semantic model:

**Definition 6 (Semantic model).** A semantic model is a graph \( S(V, E, P) \) where

Example 3. The semantic model for the resource graph of our example is illustrated in Fig. 3. It captures persons, universities, articles and relations between them. Further examples of the semantic model are discussed in the subsequent sections. Fig. 5 shows the same model that is adopted for query translation and Fig. 9 shows a model used for query matching.

The formal semantics of this model is established by an explicit mapping to first-order logic. So far, edges \( e(t_1, t_2) \) are only syntactic entities. They are given meaning by declaring that \( e(t_1, t_2) \) holds exactly when this, taken as first-order sentence, is evaluated as true. Thus, we give the proposed model a first-order semantic by interpreting edges as first-order sentences:

Definition 7 (Semantics). A model \( S(V, L, E) \) is given first-order semantics though the mapping of edges \( e(t_1, t_2) \) to atomic first order formulae \( p(t_1, t_2) \), where \( t_1, t_2 \) are terms and \( p \) is a binary predicate symbol.

Note that with respect to the different types of semantic models discussed in the previous section (and the definition of semantic models in Section 2), the one presented here corresponds to a lightweight ontology. Its degree of expressiveness is less than RDFS. In particular, class vertex corresponds to rdfs:Class, data type vertex corresponds to rdfs:Datatype, relation and attribute vertices capture the notion of rdfs:Property, and domain and range are same as rdfs:domain and rdfs:range, respectively. Additional RDFS features such as rdfs:subClassOf and rdfs:subPropertyOf are however not considered, and may be included in further extensions of this model. It has been chosen as the central model for semantic search for the following reasons:

- Tractability: Compared to the more expressive languages such as many description logics fragments defined for OWL2 and full first order logic, the model presented here is rather “lightweight”. While it has been associated with formal semantics, the kind of reasoning that can be performed on it is limited, i.e. there is no implicit knowledge that can be inferred. However, we exploit the formal semantics of explicitly asserted knowledge. In particular, we will discuss how the semantics of explicitly given information is used for interpreting keywords and for matching computed interpretations to resources. While reasoning capabilities that infer implicit knowledge can help increasing the expressiveness of search (and thus, help satisfying more complex information needs), it comes at the cost of higher computational complexity. Since the kind of large scale and domain independent type of search we are concerned with typically involves a large amount of resources, this kind of reasoning is expensive. For online tasks and especially for search, this is still a problem because timely response is critical for user acceptance.

- Generality and extensibility: It is a basic model that is sufficiently general to conceptualize different kinds of resources, including documents of different types (Web, XML) and real world entities. Besides acting as a conceptual model, the basic first order semantics defined for it is also compatible to linguistic models, i.e. words can be interpreted as terms and linguistic relations between them mapped to predicates. Further, since this model and the semantic web languages RDF(S) and OWL rest on the same foundation, i.e. the one of first order logic, it is straightforward to extend it with additional modeling constructs. For instance, there are applications where knowledge about class hierarchies is available and can be effectively exploited for semantic search. In this case, formal conceptualization of the RDF(S) vocabularies subClassOf and subPropertyOf might be added.

- Manageability: Most importantly, we note that for search, it is a difficult task to obtain conceptual knowledge. For instance, there is a large amount of RDFa data available, which represents useful information about the documents it is associated with. However, this data comes rarely with a schema. Also, for a large number of ontologies currently indexed by current Semantic Web search engines, the associated schema is incomplete or even does not exist. Further, resources under consideration are heterogeneous and evolve dynamically over time. Thus, the manual maintenance of the conceptual knowledge poses another problem. Using a more lightweight semantic model solves this problem. This is because there are techniques (which we will discuss in next section), for which such a semantic model can be automatically computed from the data.

4. Supporting the semantic search process

In this section, we discuss the idea of process-based semantic search behind our approach called SemSearchPro. This approach instantiates the process-based semantic search model introduced previously. The high-level overview on the individual steps involved in the search process as well as main the models employed for this are illustrated in Fig. 4. Central to this approach is the graph-based semantic model. This conceptualization of resources and information needs might be established by hand, i.e. developed by knowledge engineers and made available in the form of a schema. As an alternative (or in combination with this traditional knowledge engineering approach), the semantic model we focus on can also be automatically derived from the data. For this, we run a bisimulation on the data graph (offline processing). Intuitively speaking, applying this notion originating from state-based dynamic systems amounts to partitioning the data into classes of structurally similar elements, and relations between them. Computing the semantic model from data this way is important especially on the Web, where schema information does not always exist for the available data, is incomplete or generally speaking, is costly to develop and to maintain.

The construction of the semantic model is performed offline. We will now discuss the online search process and the specific steps supported by SemSearchPro.

(i) Keyword query: The process starts with the information needs of users being represented in terms of keywords. The use of keywords as the user query model is driven by
its popularity and widespread adoption. Past experiences with Web search engines have shown that the keyword search paradigm is a simple and intuitive paradigm for expressing information needs.

(ii) **Keyword translation:** However, the query model used internally is more expressive than that. In order to facilitate the construction of expressive information needs, the system automatically translates the keyword query entered by the user to a list of candidate conjunctive queries.

(iii) **Query visualization:** These queries, representing possible interpretations of the keywords, are presented using different visualization modules. For obtaining groups of instances that commonly exhibit the same types of relations and attributes. Typically, knowledge engineers define such classes based on their knowledge about the domain. In fact, such a semantic model is designed beforehand in the form of a data schema. Then, data is put into the system to populate the schema. While this traditional workflow is commonly used for applications that are based on well-defined domain specific data, it is no longer applicable to large scale Web applications that have to deal with dynamically evolving generic data. In such scenarios, a schema cannot be defined completely a priori but must also evolve with changes in usage requirements, and with changes in the underlying data.

Instead of defining the conceptualization of resources a priori, we propose to compute it automatically from data to support a more affordable approach to semantic search. According to the resource model, we note that relations and attributes of instances correspond to edges of a data graph. For obtaining groups of instances that are similar w.r.t. to the types of edges they exhibit, we apply the well-known notion of bisimulation originating from the theoretical analysis of state-based dynamic systems. In particular, we consider graph vertices \(v_1, v_2\) as bisimilar (i.e. \(v_1 \sim v_2\)), if they cannot be distinguished by looking at their “neighborhood of edge-labels”. We consider the general case of complete bisimilarity, where the neighborhood to be considered is simply the entire graph, i.e. all incoming and outgoing edge-labeled paths of arbitrary lengths.

Definition 8 (Bisimulation). Given a data graph \(G = (V, E)\), a (back-and-forth) bisimulation on \(G\) is a binary relation \(R \subseteq V \times V\) on the vertices of \(G\) such that for \(v, w \in V\) and \(l \in L\):

\[-vRw \land \ell(v, v') \in E\] implies that there is a \(w' \in V\) with \(\ell(w, w') \in E\) and \(v'Rw'\),

\[-vRw \land \ell(w, w') \in E\] implies that there is a \(v' \in V\) with \(\ell(v, v') \in E\) and \(v'Rw'\),

\[-vRw \land \ell(v, v') \in E\] implies that there is a \(w' \in V\) with \(\ell(w, w') \in E\) and \(v'Rw'\),

\[-vRw \land \ell(w', w) \in E\] implies that there is a \(v' \in V\) with \(\ell(v', v) \in E\) and \(v'Rw'\).
Two vertices \( v, w \) will be called bisimilar (written \( v \sim w \)), if there exists a bisimulation \( R \) with \( v R w \).

Intuitively speaking, two vertices are bisimilar when they share the same structure found in the data graph. Based on this notion of bisimilarity, classes of a semantic model can be computed by considering pairwise bisimilar elements. In particular, we refer to a class of all pairwise bisimilar elements as an extension. These extensions can be computed by applying bisimulation on a data graph.

For this, we use an adapted version of the algorithm for forward bisimulation equivalence presented in [11] which in turn is an extension of Paige and Tarjan’s algorithm [31] for determining the coarsest stable refinement of a partitioning \( P \). This algorithm starts with a partition consisting of a single extension that contains all nodes from the data graph. This extension is successively split into smaller blocks until the partition \( P \) is stable (i.e. no more splitting is possible on the remaining extensions).

After having determined the bisimulation, the resulting extensions from the partition \( P \) of the bisimulation are used to form classes of the semantic model \( S \). An edge with label \( 1 \) between two extensions \( E_1, E_2 \) (two classes in \( S \)) is created, if there is at least one pair of vertices \( v_1 \in E_1, v_2 \in E_2 \) such that the input data graph has an edge with label \( 1 \) from \( v_1 \) to \( v_2 \), i.e. \( \{ v_1, v_2 \} \in E \).

Note that this way, the semantic model \( S \) is derived from structural patterns found in the data.

4.2. Translation

In our approach, users articulate their information needs using keyword queries. We believe this is an adequate interaction paradigm for the Web, because for searching, users do not need to know the formal query language, underlying data or the schema. They can simply use their “own words” to express information needs of different types.

Thus, the specific problem we deal with here is to translate the given user keyword query \( Q_u = \{ k_1, \ldots, k_n \} \) to a list of candidate interpretations in the form of conjunctive queries \( Q_S = \{ \alpha_1, \ldots, \alpha_m \} \). The following implementation of the semantic translation framework can be used for this.

The translation basically involves three main tasks: namely (1) construction of the query space, (2) top-k query graph exploration and (3) query graph ranking. This procedure is similar to the techniques used for keyword search in databases. Typically, the search space employed for keyword search is the data graph [20,25].

Keywords entered by the user are matched against elements of this query space. It is also used for the exploration of subgraphs, which connect the keyword matching elements. Such an exploration might be very expensive when the data graph is large. While these approaches compute the actual answers, our approach focuses on the translation and thus, is concerned with computing queries only. For this purpose, we leverage the semantic model as the space of possible interpretations of the keywords, i.e. the query space. Clearly, this model is typically much smaller than the actual data graph.

Construction of the query search space: In fact, the construction of the query space is performed in two separate phases, resulting in two main parts:

- **Keyword matching elements**: in the first step during the online search process, elements which possibly correspond to the keywords \( Q_u = \{ k_1, \ldots, k_n \} \) entered by the user are computed. This is performed via \( f(k) \rightarrow N_k \), i.e. a matching function that maps keywords to sets of graph elements referred to as the keyword matching elements \( N_k \), where \( N_k \subseteq V^R \cup L^E \). In other words, keywords are interpreted as constants represented by some data graph vertices \( V^R \) or as predicates drawn from the set of edge labels \( L^E \).
  - **Semantic model**: the goal of translation is not only finding keyword matching elements but moreover, to discover what they actually mean in combination. Technically speaking, the aim is to find out how constants and predicates found in the first step are connected. Since the underlying semantic model captures the different ways resources are related, it is used to explore the different interpretations of the computed elements.

**Definition 9** (Query space). A query space \( S_Q(S, N_k) \) comprises the keyword matching elements \( N_k \) computed for a query \( Q_u \), which when not already contained, are connected with elements of a semantic model \( S(V^Q, L^Q, E^Q) \).

**Example 4.** Fig. 5 illustrates the semantic model, omitting attribute edges, computed from the data graph in Example 1 for the purpose of translation. This one is extended to obtain the query space shown in Fig. 6. The elements denoting keywords in the query \( Q_u \) = “Article Stanford Turing Award” have to be taken into account. Keyword elements obtained through matching the user keywords against resource labels are Article, Stanford University and Turing Award. Article is already part of the semantic model. Stanford University and Turing Award have to be added to obtain the complete query space for \( Q_u \).

**Exploration and ranking**: Given the query space, query interpretation amounts to searching for the minimal query graphs, defined as follows:

**Definition 10** (Query graph). Let \( Q_S = K \) be the user query, \( S_Q(S, N_k) \) be the query space, a query graph \( Q_S = (V^Q, L^Q, E^Q) \) with \( V^Q, L^Q \) and \( E^Q \) being elements of \( S_Q \) and

- for every \( k \in K, f(k) \cap V^Q \neq \emptyset \), i.e. \( Q_S \) contains at least one representative keyword matching element for every keyword from \( K \), and
- \( Q_S \) is connected, i.e. there exists a path from every graph element of \( Q_S \) to every other graph element of \( Q_S \).

A matching graph \( Q_S \) is minimal if there exists no other \( Q_S \) in \( S_Q \) such that \( \text{Score}(Q_S) < \text{Score}(Q_S) \), where \( \text{Score} : Q_S \mapsto [0,1] \).

![Fig. 5. A semantic model used for translation.](image1.png)

![Fig. 6. A query space.](image2.png)
We employ a top-k graph exploration procedure to find such query graphs (see [38] for details). It starts from the keyword elements $N_k$ and iteratively explores the query space $SQ$ for all distinct paths beginning from these elements. During this procedure, the path with the highest score so far is selected for further exploration. For scoring paths we incorporate (1) the popularity of graph elements (obtained via an adapted version of TF-IDF called EF-IDF), (2) the matching score $Sm$ of keyword elements (obtained via the imprecise matching of keywords against graph element labels) and (3) the length of the path. Basically, the EF-IDF measures the popularity of an element by how often it relatively occurs in the dataset [38].

At some point, an element might be discovered to be a connecting element, i.e. there is a path from that element to at least one keyword element, for every keyword in $K$. The paths between the keyword elements and the connecting element are merged to form a query graph. The graphs explored this way are added to the candidate list. The process continues until the upper bound score for the query graphs yet to be explored is lower than the score of the k-ranked result. Thus, addressing complex information needs of the user amounts to the problem of graph pattern matching. Clearly, $\mu$ is a certain type of homomorphism (i.e. a structure preserving mapping\(^{12}\)) from the query graph to the data graph. We will use this perspective of considering answers to a query as homomorphisms. In the following, we will elaborate on how the semantic model can also be leveraged for this matching problem.

Recall that the semantic model is actually derived from structural patterns found in the data. It contains the different structures exhibited by data graph elements. In other words, it preserves the structure of the data graph. Thus, it can be seen as a compact representation of candidate answer graphs, where instead of single data elements, vertices represent set of elements (i.e. extensions).

Example 6. Fig. 8 shows an extended example for the data graph. By means of bisimulation, the semantic model shown in Fig. 9 can be computed from it. Note that data graph elements that share the same structure are grouped and represented as extensions in the semantic model. For instance the extension $E_2$ comprises $un1$ and $un2$, which as illustrated in the data graph in Fig. 8, are similar w.r.t. to all the incoming and outgoing edges. Clearly, all distinct paths in the data graph in Fig. 8 are captured by this semantic model.

Note that this example of a semantic model is conceptually not different from the ones shown in Figs. 3 and 5. Extensions correspond to classes, which are connected by relations. However, to illustrate the correspondence between the semantic model and the data from which it has been derived, and its correspondence to the query, we render the relations as edges. That is, instead of using a vertex and the two edges domain and range to capture a relation, we replace them by an edge.

Similar to the concept of query space used for translation, the semantic model is employed for the purpose of matching, and is referred to as the “answer space”. We will now characterize the properties of the answer space which justify its usage (see formal proofs in [36]).
Proposition 1. Let \( g \) be a data graph with the associated answer space \( g^\ast \) and let \( g' \) be another graph such that there is a homomorphism \( h \) from \( g' \) into \( g \). Then \( h^\ast \) with \( h^\ast(v) := [h(v)]_r \) is a homomorphism from \( g' \) into \( g^\ast \).

Roughly speaking, this proposition ensures that, whenever there is a match of a query graph on a data graph, the query also matches on the answer space. Moreover, classes that are vertices in the extension of a query graph on a data graph, the query also matches.

Example 7. Fig. 10 depicts a query, which asks for authors working at Stanford University that have won a Turing Award. Further, \( y \) should supervise some \( u \) that is author of some \( v \). The matching of the query graph in Fig. 10 on the answer space in Fig. 10 results in one single match \( h = \{ x \mapsto E1, y \mapsto E4, z \mapsto E7, u \mapsto E3, v \mapsto E5, Stanford University \mapsto E6, Turing Award \mapsto E8 \} \). Through this matching, we know that elements in \( E4 \) work at some places \( x \), have some prizes \( z \) and supervise \( u \). Further, we also know that \( u \) is author of some \( v \). Next, we check whether elements in \( E4 \) match the elements mentioned in the query, i.e. they really work at Stanford University, and have won a Turing Award. For this, we retrieve data contained in the extensions \( E6, E1, E4, E7 \) and \( E8 \) and join them along the edges \( \{ y \) employment \( x \}, \{ y \) name Stanford University \}, \{ y prize \}, \{ z label Turing Award \}.

Note that using the answer space, we retrieve and join data only for a certain part of the query, i.e. the rest of the query can be pruned away after processing step one. We will give another proposition that more precisely defines the part that can be pruned away.

Proposition 2. Let \( g \) be a data graph with the associated answer space \( g^\ast \) and let \( g' \) be a tree-shaped graph, where all nodes except possibly the root \( r \) are non-distinguished variables. Let \( h' \) be a homomorphism from \( g' \) to \( g^\ast \). Then for every node \( v \in h'(r) \), there is a homomorphism \( h \) from \( g' \) to \( g \) with \( h(r) = v \).

Informally, this proposition ensures the following: Suppose there is an accordingly tree-shaped query graph \( g' \) corresponding to \( \{ or \mid \text{is part of} \} \) the query posed against the data graph \( g \). The proposition now states that for any match \( h' \) of \( g' \) against the answer space \( g^\ast \), every data element \( v \) in the extension(s) assigned to the query node \( r \) (i.e. \( h'(r) \)), represents a data graph match (i.e. matches both the structure and the query elements). In other words, for this special type of tree-like query parts, no verification step will be necessary. Data elements are retrieved only for the root node \( r \) of the query \( g' \), while the rest of \( g' \) can be pruned away.

Example 8. Continuing with our previous example, we can see that there are two tree-like parts that contain no distinguished variables, i.e. the paths \( \{ x \) employment \( u \} \{ y \) author \( u \} \) and \( \{ y \) supervises \( u \} \{ y \) author \( u \} \). These parts can be pruned away after step one as all data elements contained in answer space matches are already known to satisfy these structural constraints, i.e. elements in \( E4 \) are already known to supervise some \( u \) that are authors of \( v \), and elements in \( E1 \) are known to employ some \( u \) that are authors of \( v \), respectively.

4.4. Presentation and refinement

The formal models of resources and queries are sufficiently expressive to address different kinds of information needs. The goal of our approach is to enable users harnessing this expressiveness, without having to deal directly with the underlying formal models – at anytime during the search process. So far, we have shown that the underlying semantic model can be leveraged for supporting a more lightweight query interface based on keywords, and for a more efficient matching of queries against resources. In this section, we will show that also the semantics of queries and resources can be exploited for enabling users to interact with more intuitive presentations of queries and results.

Result presentation: In particular, the semantics of resources are used to map the internal representation of resources to presentation modules of different kinds. Recall that results to a query are of the types \{1\} facts, \{2\} entities and \{3\} tuples of entities (i.e. entities, their attributes and relations between them). These types of results are presented using generic and entity and fact specific presentation libraries. The mapping of internal representation of results \( R_\delta \) to presentation libraries \( R_P \) are as follows:

Generic presentation interfaces: For generic results of types facts, entities and entity tuples, we use the generic interfaces fact box, list and table. In particular, factual results containing several data values (V-vertices \( V^\delta \)) are mapped to a fact box. Results containing several entities (E-vertices \( V^\delta \)) are mapped to a list where every entity description is rendered using a separate row in the list. A table is used to present entity tuples where entities’ relations (\( V^\delta \)) and attribute values (\( V^\delta \)) are mapped to table cells, and attribute and relation names (\( L_\delta \) and \( L_\delta \)) are mapped to column labels. Note that unlike tables in databases, the ones used here might contain information about entities of
several classes, and different relations between them. For instance, one table might contain people, places, events, some of their attributes, and relations between them. Therefore, it might be unclear to which entities some other entities or attribute values refer to. In addition to the column labels, one another layer is thus used to depict the connections between them, i.e. the connections between the entities and data values contained in the columns.

- **Fact-specific presentation interfaces:** For single facts of certain types, there exist specific presentation modules. For instance, facts of the types location and time are mapped to visualization modules specifically designed for rendering locations (on a map) and times (on a timeline). Presentation modules are not limited to the purpose of presentation but might contain elements supporting further interactions. For instance, facts of the type telephone number are mapped to widgets that display the number and also, support actions such as “store number” and “call” (e.g. using Skype).

- **Entity-specific presentation interfaces:** For single entities of certain types, i.e. E-vertices of some classes $V^E$, there exists also specific presentation modules. Note that for entities, the units of presentation comprise not only of the entity identifier but its entire description, i.e. a set of relations and attributes value pairs $(L^E_V v^E_i)$ and $(L^E_R v^E_i)$. Accordingly, entity presentation modules are also composites, which might be constructed using fact-specific presentation modules. There are specific modules for rendering entities of types people, place, event, publication and projects. For instance, attributes of persons are presented using fact-specific presentation modules as obtained via the mapping of facts, e.g. location, telephone number etc.

**Query presentation:** In the traditional search process, users typically issue a query, obtain results, and if these do not satisfy the information need, start over with issuing a new query. The approach we propose emphasizes the steps of supporting users during their entire description, i.e. a set of relations and attributes value pairs $(L^E_V v^E_i)$ and $(L^E_R v^E_i)$. Accordingly, entity presentation modules are also composites, which might be constructed using fact-specific presentation modules. These operations are transparently converted to changes on the underlying query. The query refined this way is immediately evaluated, and new results are presented without the user having to explicitly issue a new query.

5. Implementation

In this section we present three systems that have been developed: (1) AskTheWiki, (2) Hermes and (3) the Information Workbench. These systems serve as demonstrators of process-based semantic search. That is, they instantiate the process-based semantic search model presented in Section 2 and implement our process-based semantic approach presented in Section 4. All systems support the entire search process, but focus on different aspects. We refer the interested reader to the more complete descriptions of these systems available in the respective conference papers and technical reports [15,42,14].

The main goal of AskTheWiki [15] is to study search in the context of a specific information portal. The search system is developed on top of a system called Semantic MediaWiki [27]. While it operates currently only on data provided by this system, it is in fact able to search on any graph structured data.

The Hermes system [42] targets semantic search in a multi-dataset scenario. It has been built with many datasets available for the Billion Triple Challenge. The system supports querying and combining results from different datasets using keywords, and the subsequent exploration and refinement using faceted search. Especially for this setting, the use of a compact semantic model has proved essential for efficient query translation and matching. Besides the concepts presented in this paper, specialized techniques and indexes for computing and managing mappings have been used for this system. These mappings establish links between datasets, which are used for federated query processing (schema level mappings) as well as for the combination of results that come from different sources (data level mappings).

Refinement: Refinements to the query may be needed for several reasons. The computed interpretations may not exactly match the information need. Also, the user may start out with a vague information need, not knowing exactly what he is searching for. For these cases, a presentation query model which intuitively reflects the semantics of the result set, facilitates the modification of results and the comprehension of the resulting changes. In particular, a facet-based presentation model provides the means for the user to narrow down or expand the resources of interest according to their information need in an iterative way. In particular, the user can add, remove or edit the facets (i.e. change the predicate of the constant). These operations are transparently converted to changes on the underlying query. The query refined this way is immediately evaluated, and new results are presented without the user having to explicitly issue a new query.

The Information Workbench [14] targets not only search, but the entire process of interacting with the Web of data. Users can import data, then search, explore, analyze as well as refine it, and finally, publish it back to the Web. Besides supporting the search features that are similar to the other two implementations, this system features a concept called “Living UI”, which basically implements the concepts for result presentation discussed in this paper. It offers an adaptive user interface, where different results are presented using type-specific widgets.

Fig. 11 shows the interfaces that can be used for the main steps of the search process, illustrating how it is supported by these systems. In particular, it shows the presentations of queries and results, illustrating the concepts discussed in Section 4.4. Note that since the systems operate on different datasets, the queries and results shown in different screenshots were actually taken from different examples. The keyword interface shown in Fig. 11(1) is well known and similar to those of Web search engines. Queries computed from the user’s keywords are presented using graph- or table-based visualization, as shown in Fig. 11(2,3). Query refinement is achieved through a faceted search interface, which is another form of query presentation. Fig. 11(4) shows a facet menu that allows relations to be added to or removed from the query. Presentations specific for entities show the relations and attributes of an entity as a graph, structured data combined with the textual description of the entity, see Fig. 11(5,6). Maps and diagrams, as shown in Fig. 11(7,8), are examples of fact-specific presentations that display facts of specific types.

6. Evaluation

The work presented here brings together pieces of work targeting different aspects of the semantic search process. In particular, we have reported and evaluated the work on query translation, showing that it outperforms the state-of-the-art baseline by at least one order of magnitudes [38]. While achieving performance comparable to the baseline, our approach of using a compact query search space does not require large indexes and thus can reduce space requirements. Also for matching, we showed that our approach of using the semantic model for pruning the answer search space can reduce IO costs as well as the number of unions and joins. Performance on a mix of different query types can be 5–7 times faster than the state of the art [36]. More details on the evaluation results and the comparative studies of our translation and matching approaches can be found in [38,36].

We have also conducted an experiment considering the semantic process as a whole, which is presented in this section. The goal for this is to assess the applicability of our process-based approach in a real-life scenario. We performed a user study to evaluate the system primarily in terms of effectiveness and efficiency, and also derived some initial results concerning user satisfaction and usability.

6.1. Evaluating the entire process

Since our approach takes the entire process into account, the standard IR evaluation based on the technical measures of precision and recall does not apply. This paradigm is too limited to assess the overall effectiveness and efficiency of the overall search...
process. For example, using standard document retrieval it is not possible to directly retrieve facts, but instead only documents containing these facts. For answering a fact or relation query a user might have to inspect a large number of documents whereas our system can return results directly. The evaluation was conducted with AskTheWiki.

6.1.1. Evaluation setting

Thus, we follow the task-based evaluation design that has gained acceptance in the IR community for evaluating interactive systems [10].

In particular, we chose a task-based user evaluation where each task represents an information need that typically occurs when using the portal. After the task evaluation, we asked the participants to answer a multiple choice questionnaire about their experience. The questions concerned their technical background and their experience and satisfaction with certain aspects of the supported search process.

Participants of the user study were 14 volunteers from four different organizations active in the Semantic Web community. The users know the semanticweb.org platform. While they partially know the kind of data that can be found there, they do not know the schema (as it is not explicitly available). All participants are familiar with keyword search. Some users (25 percent) do not know how to use SPARQL, the system query language $Q_5$, used in our implementation of process-based search.

Each participant was given five tasks and had up to three minutes to solve each task. A task could be skipped, if participants felt that they could not solve the task. The participants received limited information about the search interface upfront, namely that they do not need to know the schema and a formal query language, that the search process consists of three steps, that the system will not return a list of documents like common search engines but interpretations of their keywords, and that they have to choose an interpretation, and that the results can be modified in the third step. All actions taken by the participants and the system responses were logged. In particular, we logged the users’ steps, the keyword inputs as well as the system responses. We measured how often users could solve the task and how much time it took them. The evaluation was performed based on both the analysis of the log files as well as the questionnaire.

Additional information such as the questionnaire and the handouts provided to participants can be found at [16].

6.1.2. Tasks

Designing the tasks is crucial for the success of an evaluation. The tasks were constructed to assess the quality of the individual steps and more importantly the search process as a whole. They cover different levels of difficulty. In particular, we created tasks corresponding to queries that fall into the search categories introduced in Section 3.2, i.e. entity queries, fact queries and relation queries.

We created two task sets as shown in Tables 1 and 2. Both sets have the same structure and cover the same query types.

6.1.3. Data

We performed the evaluation within the community portal semanticweb.org, a wiki-based platform serving the Semantic Web community. The wiki contains information about the Semantic Web such as events, publications, tools, and people. This data comprises a total of 55,365 triples (as of December 4, 2008). The data was created by the users of the wiki over the last 3 years. Since the nature of a wiki is to provide unconstrained user editing, the data does not follow a predefined vocabulary or strict schema. Rather, it evolves with changes in the data. For the evaluation, we have computed a semantic model from the data. It has a size of 7101 triples.

6.1.4. Effectiveness

To measure the overall effectiveness, we analyzed the ratio of tasks that have been successfully completed. Overall, 6 out of 14 participants were able to fulfill all five tasks, 12 of the 14 were able to fulfill 60% or more. The other two users quickly gave up after the first or second task stating that they found the system too complicated (see Fig. 13). For the simple tasks (entity queries), the success rate was 100%, the more complex tasks result in lower success rates: 79% for fact queries and 64% for relation queries. These results are illustrated in Fig. 12loat>.

Fig. 14 shows the success rate for each individual task. There is a notable difference between the success rates of task 4e and 4d, as well as between 5e and 5d. For tasks 4e the success rate is comparatively low, because four of the seven participants did not enter the keyword “homepage”. One reason for this difference might that participants misunderstood the term “homepage” in the query 4e, and thought that the links to wiki pages of the reasoners with GPL licenses are already the results. For 4d, only one participant

<table>
<thead>
<tr>
<th>Task</th>
<th>Task description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1d</td>
<td>Find the wiki page of AIFB</td>
<td>Entity query</td>
</tr>
<tr>
<td>2d</td>
<td>When is the paper deadline for the ASWC2008?</td>
<td>Fact query</td>
</tr>
<tr>
<td>3d</td>
<td>What is the email of Holger Lewen?</td>
<td>Fact query</td>
</tr>
<tr>
<td>4d</td>
<td>Find exporters with GPL license and their homepages</td>
<td>Relation query</td>
</tr>
<tr>
<td>5d</td>
<td>Find the capitals of countries in Europe and the population of these cities</td>
<td>Relation query</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Task description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1e</td>
<td>Find the wiki page of Stanford University</td>
<td>Entity query</td>
</tr>
<tr>
<td>2e</td>
<td>What is the homepage of the ISWC2008 conference?</td>
<td>Fact query</td>
</tr>
<tr>
<td>3e</td>
<td>What is the email of Thanh Tran?</td>
<td>Fact query</td>
</tr>
<tr>
<td>4e</td>
<td>Find reasoners with GPL license and their homepages</td>
<td>Relation query</td>
</tr>
<tr>
<td>5e</td>
<td>Who was the local chair of the conferences located in Karlsruhe in 2008?</td>
<td>Relation query</td>
</tr>
</tbody>
</table>

Fig. 12. Effectiveness of search by query type.

---

15 The data is available at [http://semanticweb.org/RDF](http://semanticweb.org/RDF).
did not use the keyword “homepage” and therefore most users found the actual results, namely the homepage of the exporters instead of the wiki page. Task 5e has a lower success rate than 5d because many participants used the keyword “2008” stated in the task description. This keyword is problematic because it matches many elements in the data, i.e. many attribute values contain “2008”. Due to this kind of ambiguity, the quality of the generated interpretations were not high, and therefore this task yielded a lower success rate.

In conclusion, the search process was effective in supporting the tasks, as most of them could have been completed by all users. The problematic cases are when users enter too few keywords, resulting in incomplete interpretations or when keywords are too ambiguous, resulting in too many matches and interpretations, respectively.

6.1.5. Efficiency

To assess the efficiency, we measured how many search iterations were performed by counting the number of keyword queries a user had to reissue per task. The results are shown in Fig. 15 grouped by the types of queries and for each task separately in Fig. 16. We see that on average the users needed to issue between 1.6 and 2 keyword queries to fulfill a task, depending on the query type. Expectedly, the value is larger for the more complex relation queries (2 keyword queries per task) than for the simple types of queries. As shown in Fig. 16, there is large difference between the number of queries issued for tasks 5d and 5e. This is directly correlated to the low success rate of task 5e, which we just discussed. The participants tried harder to solve 5e and thus issued more queries, whereas task 5d was apparently easier to solve.

Beside the total number of process iterations per task, we also measured the total process cost in terms of the steps taken and the time consumption. Note that users do not have to work through the entire search process. At any step, users might decide to stop when the information need is found to be fulfilled. For example, when searching for an entity, the returned interpretations might already contain the entity, and thus the answer. The actual process steps taken by the users are shown for tasks of different types in Fig. 17. We see that only 21% of the 24 search processes executed for entity search tasks continued to the second step, whereas in 40% and 61% of the fact and relation search processes, respectively, users had to chose an interpretation. Refining the search result was performed in 17%, 11%, and 20% of the search processes, respectively, as shown in Fig. 17.

We measured the time spent for a task by summing up the duration of all steps performed by the user for each task. The total duration for a search is the time difference between the keyword query arrival at the server and the last action performed by the user. The time needed by the user to actually type the keywords is not included in the measurement.

For the entity search tasks, the users needed on average 9.3 s with a median of 0.5 s. The median is significantly lower than the average, since 8 of the 14 participants needed just one keyword query to complete the task and did not take any further steps, as shown in Fig. 17. The fact search tasks took on average 12.8 s with a median of 9.6 s. The participants spent on average 52 s with a median of 44.5 s on the relation search tasks.

Further, we recorded the time needed for two main parts of this process: the time up to choosing an interpretation and the time needed for every subsequent refinement of the result set. Part 1 took the participants 14.7 s (median). Between refinement operations they participants spent 14.0 s (median). Note that part 1 includes the time needed by the system to actually compute the interpretations. We already discussed the performance results that can be achieved for this kind of processing. It actually makes up only a small share, while the rest of the time is needed by the user to understand and to choose the intended interpretation. The second part contains the time the system needed for query processing but also here, most of the time is actually spent by the user. More
than 90% of this time is needed to understand the results and to assess whether they fit the information need.

The overall results are encouraging, showing that even tasks which involve complex structured queries and results, require no more than two process iterations, and less than one minute in total. Also, the overall success rate is encouraging, considering that the participants have varying technical backgrounds, do not know the underlying data schema, and used the system, which was new to them, without detailed usage instructions.

### 6.1.6. Usefulness

We now discuss the usefulness of different aspects of the implemented search process. This discussion is based on the results of the questionnaire. The responses to these questions are shown in Fig. 18.

**Articulation of the information needs**: The first question asked how difficult the users found it to express the information need in keywords. As expected, the users found it rather easy to do so, as all of them are familiar with keyword-based search interfaces.

**Efficiency measured by search-process steps taken.**

![Fig. 17.](image)

**Usability study (n=14)**

<table>
<thead>
<tr>
<th>How easy was it for you to express your information need in keywords in step 1?</th>
<th>How easy did you find it to choose an interpretation in step 2?</th>
<th>Is the presentation of the interpretations comprehensible in step 2?</th>
<th>How easy was it for you to understand the presentation of the results in step 3?</th>
<th>Did you find it useful to modify the interpretation in step 3?</th>
</tr>
</thead>
<tbody>
<tr>
<td>easy</td>
<td>7</td>
<td>easy</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>rather easy</td>
<td>6</td>
<td>rather difficult</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>difficult</td>
<td>1</td>
<td>difficult</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>other</td>
<td>0</td>
<td>not comprehensible</td>
<td>0</td>
<td>not comprehensible</td>
</tr>
<tr>
<td>how to do it</td>
<td>0</td>
<td>rather not comprehensible</td>
<td>0</td>
<td>rather not comprehensible</td>
</tr>
<tr>
<td>didn’t know how to do it</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Fig. 18.](image)
The results of the questionnaire tell that all participants use Google and some have tried other search engines with more advanced features such as Powerset and Cuil.

**Translation:** Overall, the users found the representation of interpretations easily comprehensible, and it was easy for them to choose the right interpretation. Two users had difficulties (see questions regarding step 2 in Fig. 18). One reason was that in some cases the interpretations were so similar that the users could not easily tell the differences.

**Presentation and refinement:** The majority of the users found the presentation of the results understandable. However, only seven users made use of faceted search to refine a query. As answers to the question about how useful the feature was to the users, most participants found it very useful or useful to modify the interpretations, whereas three participants stated that they did not know how to do it (see questions regarding step 3 in Fig. 18). This suggests that when users know how, faceted search is useful in this process. Unlike keyword search, faceted search is however not yet a commonly used paradigm. Effective use might require more detailed instructions, which were (deliberately) not given in our experiment. Interestingly, the use of the faceted search was particularly effective for the more complex tasks. On average, 29.6% of the successfully completed tasks involved refinements using faceted search. For the most complex tasks involving relation search, 38.8% of the successfully completed tasks involved the use of faceted search. We thus have reason to believe that the overall success rate would have been higher, if all users had known how to utilize faceted search.

7. Conclusions

We discussed different approaches to semantic search under a general model. Based on this, we proposed an extension, which takes the entire search process into account. We discussed a compilation of our work called SemSearchPro, which implements this process-based process model using a lightweight semantic model that can be automatically derived from the underlying data. Several systems implementing SemSearchPro were discussed to demonstrate how the search process can be supported in real-world scenarios. We evaluated different aspects of SemSearchPro. The results for the individual steps suggest that the use of semantics can lead to an increase in performance, for both the translation as well as the matching tasks. The task-based evaluation, which considers the process as a whole suggests that it is efficient and effective. Most of the tasks were completed in reasonable time.

References


