Towards a Domain Oriented and Independent Semantic Search Model

Zhixian Yan, Ying Ding, Emilia Cimpian

Digital Enterprise Research Institute (DERI) Innsbruck,
Innsbruck University, Austria
{firstname.secondname}@deri.org

Abstract. Approaches towards semantic search can be mainly divided into two kinds: one is augmenting traditional keyword-based search engines with semantic techniques such as ontology, semantic inference; the other is proposing methods for directly searching on semantic repositories as RDF or OWL files. However, compared with those theoretics-driven approaches, semantic search still lacks applications with real life cases at the moment. By analyzing the scenario of domain resources, this paper proposes a domain-oriented semantic search model, which provides both entity search (*concepts* and *instances*) and relationship retrieval (*concept2concpet*, *concept2instance* and *instance2instance*). The search model can be further refined as a generalized and domain independent architecture with inference supports. The supporting semantic inference is gradually achieved by reasoning-rules, formulas and algorithms. With such semantic search model, more exact and meaningful information hidden in the resource repository can be extracted.

1 Introduction

Semantic web focuses on semantics rather than syntactics. Furthermore, semantic search uses semantics to discover more meaningful query results. Current semantic search studies have two strategies: the first strategy is using semantic techniques to strengthen traditional keyword-based search. We can extend queries by light-weight metadata annotation, with the help of generalized ontologies (e.g. Dublin Core, WordNet) or specialized ontologies (e.g. DOAP for open source projects, FOAF for social network). Besides query extensions, search results can be semantically repacked. Stanford TAP[1] is one of the earliest works as this strategy, and many others come forth subsequently like [2][3]. The second strategy focuses on directly searching on semantic web repositories like RDF/OWL files. Swoolge and OntoSearch are the two representatives, and there are also many other detailed algorithms for indexing and retrieving RDF/OWL [4][5]. The standardization of query language, SPARQL, further congregates the studies on RDF/OWL retrievals. In addition, there are some domain-oriented semantic works, such as audio archives (SIMAC and EASAIER), open source projects (DOAP), digital library[6], social network (SIOC and FOAF), e-Learning[7] etc.

The stress of this paper is to analyze common semantic requirements for various specific domains, in the aspects of resources management and retrieval. We proposes a generalized and integrated semantic search model, supporting fully-fledged semantic search, not only for single entity (i.e. concepts or instances) but also for relationships (i.e. *concept2concpet*, *concept2instance* and *instance2instance*). The remainder of this paper proceeds as follows: Section 2 proposes the semantic search model for domain resources; Section 3 discusses semantically enhanced search portal; Section 4 provides implementation; and finally the conclusion is shown in Section 5.

2 Semantic Search for Domain Resources

To create a domain-oriented semantic model, domain scenarios need to be provided analyzed. We further refine and formalize it with semantic definitions.

2.1 Scenarios for Domain Resources

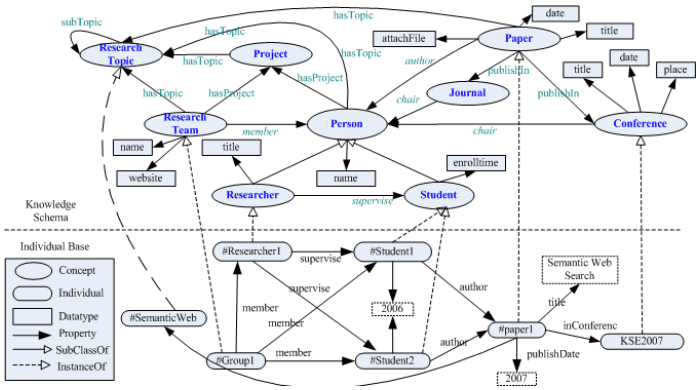

Fig. 1. The semantic scenario for Research Community

An example domain scenario is about *Research Community* shown in Fig. 1 with two layers. The upper layer contains nine main concepts, namely *Person*, *Researcher*, *ResearchGroup* etc.; while the lower layer is about concrete instances. Besides the two kinds of entities, affluent relationships can be detected in three categories, namely *Concept2Concept*, *Concept2Instance* and *Instance2Instance*. This scenario is similar with some ontology construction works[8][9], but our emphasis is the division of the two typical entities and the three kind relationships.

2.2 Domain-Oriented Semantic Search Model

To realize the semantic scenario, search portal is the core issue. However, there are some supporting components as illustrated in Fig. 2.

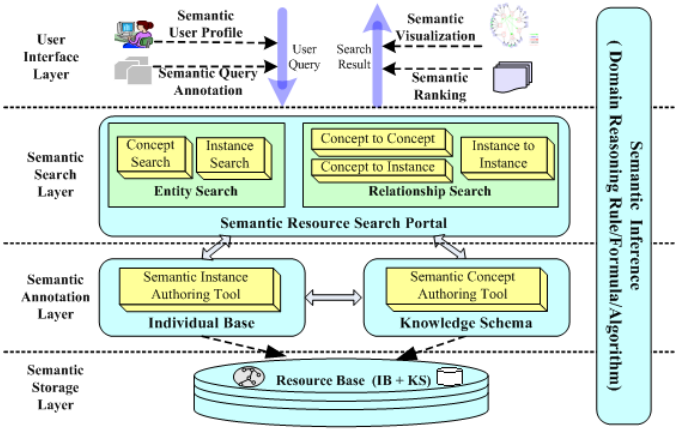

Fig. 2. The Semantic Search Model

User Interface Layer: User interface involves query-submission and results-return. The former can be improved by *semantic user profile* or *semantic query annotation*; whereas the result aspect ought to be promoted by *semantic ranking* and *semantic visualization*.

Semantic Search Layer: The portal in this layer supports semantic search for two type entities and three kind relationships in domain resources.

Semantic Annotation Layer: We provide authoring tools to realize semi-automatic semantic annotation for domain resources .

Search Storage Layer: In our implementation, both traditional database (MySQL for instances) and semantic web languages (OWL/RDF files for concepts) are applied, because the former has higher retrieval efficiency whilst the later supports richer semantics.

Search Inference Vertical: The vertical semantic inference is gradually realized via domain-oriented reasoning rules, formulas and relevant algorithms.

2.3 Semantics for Domain Resources

To formalize the search model, we provide semantic definitions for domain resources, which is also the preprimary step for the following sections.

Definition 1 (Domain Resources Semantics) *e, c, i, p, r, v,dp and op respectively denotes an entity, a concept, an instance, a property, a relation, a datatype value, a datatype property and an object property; E, C, I, P, R, V, DP and OP are their corresponding sets respectively. Herein, $E = C \cup I$ and $P = DP \cup OP$; op is a special r and $OP \subset R$.*

1.1 Concept Semantics *$< c, hasProperty, p >$ means concept c has a property p; hereinto p can be dp or op, thus two sub-definitions come forth:*

1). $< c, dp, V >$ means concept c has datatype property dp, its value range is V;

2). $< c_1, op, c_2 >$ *means concept* c_1 *has object property op to concept* c_2*; hereby it equals* $< c_1, r, c_2 >$ *in logic, since r is a kind of special op.*
1.2 Instance Semantics $< c, hasInstance, i >$ *means concept c has an instance i, and vise verse* $< i, belongTo, c >$*; sub-definitions exist:*
1). $< i, dp, v >$ *means the concrete value of property dp in instance i is v;*
2). $< i_1, op, i_2 >$ *or* $< i_1, r, i_2 >$ *means instance* i_1 *has object property op or relationship r with instance* i_2
1.3. Relationship Semantics *the semantics of relationship can be sub-defined according to the three types division:*
1). $< c_1, op, c_2 >$ *in Definition 1.1 describes the relationship between two concepts, hereby,* c_1 *and* c_2 *can be the same concept* ($c_1 = c_2$) *or not* ($c_1 \neq c_2$)*;*
2). $< c, hasInstance, i >$ *describes the relationship between concept and instance and vise verse, formalized as* $< i, belongTo, c >$*;*
3). $< i_1, op, i_2 >$ *and* $< i_1, r, i_2 >$ *in Definition 1.2 describes the relationship between two instances.*

3 Semantic Search Portal for Domain Resources

In this section, we discuss the detailed search portal supporting entity and relationship retrievals. Entity search portal can retrieve the single resources as concepts or instances. The formal semantics about entity search is following,

Definition 2 (Entity Search) *Given an entity e (c or i), the search portal extracts all the semantic relevant information about e as a result set RS; there are two sub-definitions:*
1). ***Concept Search*** *Given c, the search portal return its datatype properties DP, object properties OP and related instances I, as* $RS(i) = \{DP, OP, I\}$*;*
2). ***Instance Search*** *Given i, the search portal returns the concept set C it belongs to, the datatype properties with exact values DP_V, and the object properties with exact objective instances OP_I, thus* $RS(i) = \{C, DP\_V, OP\_I\}$*.*

3.1 Semantic Concept Search

For concept search, the portal should capture the related semantic information comprising its datatype properties, object properties as well as relationships with other concepts, and relevant instances. To completely extract semantic related information for the given concept, inference mechanism plays a significant role, which defines many reasoning rules, provides relevant formulas as mathematic models, and finally implements them via algorithms. Hereby, for concept layer, we have the following reasoning obvious rules.

Rule 1 *the transitivity of sub-concept relationship; furthermore, the sub-concept can inherit its super-concepts' properties, whilst the super-concept can get its sub-concepts' instances:*
$< c_1, subConcept, c_2 >, < c_2, subConcept, c_3 > \Rightarrow < c_1, subConcept, c_3 >$;
$< c, subConcept, c_{sub} >, < c, hasProperty, p > \Rightarrow < c_{sub}, hasProperty, p >$;
$< c, subConcept, c_{sub} >, < c_{sub}, hasInstance, i > \Rightarrow < c, hasInstance, i >$.

With the previous rules, some formulas are provided and also with relevant implemental algorithms. We list the following representative ones.

Formula 1. calculate a concept's instances

$$getInstances(c) = determinedInstances(c) + inheritedInstances(c)$$

$$determinedInstances(c) = \{i_k\}, \forall k < c, hasInstance, i_k >$$

$$inheritedInstances(c) = \begin{cases} \cup getInstance(c_k), \forall k < c, subconcept, c_k > \\ \phi, \qquad \text{with no tuple above} \end{cases}$$

Algorithm 1. Semantic Concept Search for the *queryKey*

```
1   Set resultSet = new Set ();
2   Set concepts = searchByStringKey(queryKey);
3   if (concepts == NULL) return NULL;
4   for (each concept_i in concepts) do
5     // get all properties belongs to concept_i
6     Set DP = getDatatypeProperties(concept_i);  // datatype: basicInfo
7     Set OP = getObjectProperties(concept_i);    // object: relations
8     // get all instances belongs to concept_i
9     Set I = getConceptInstances(concept_i);
10    Set conceptResult = {DP,OP,I};    // complete concept result
11    resultSet = resultSet + conceptResult;
12  end for
13  resultSet = sort(resultSet);
14  return resultSet;
```

3.2 Semantic Instance Search

For instance search, the semantic portal ought to return the belonging concepts, the datatype values and relevant relationships with other instances. Formula 2 calculates relevant concepts for instance i. It's the opposite operation to Formula 1. Its detailed algorithm is shown subsequently.

Formula 2. calculate the concepts that a given instance belongs to

$$getConcepts(i) = c_{declaredConcepts} + inheritConcepts(i)$$

$$inheritConcepts(i) = \begin{cases} \cup getConcept(c_k), \forall k < c_k, subconcept, c > \\ \phi, \qquad \text{with no tuple above} \end{cases}$$

Algorithm 2. Semantic Instance Search for the *queryKey* with *semanticTag*

```
1   Set resultSet, searchConcepts = new Set ();
2   if (semanticTag != NULL)   // only search tagged concepts
3     Set searchConcepts = getDenoteConcepts(semanticTag );
4   else                       // search all concepts
5     Set searchConcepts = getAllConcepts();
6   end if
7   for (each concept_i in concepts) do
8     Set instances = getConceptInstance(concept_i , queryKey);
9       for (each instance_j in instances) do
10        get <C,DP_V,OP_I> for instance_j
11        replace instance_j with <C,DP_V,OP_I> in instances set
12      end for
13    // sort instances belonging to the same concept
14    instances = sort (instances);
15    Element resultElement = <concept_i, instances>
16    resultSet = resultSet + resultElement
17  end for
18  return resultSet;
```

Some domain specific reasoning rules can be generalized based on the characteristics of their properties. For instance, three typical special characteristics amongst object properties are prevalent, i.e. transitive, symmetric and functional. Their three corresponding general rules are shown in Rule 2,

Rule 2 *op_{trans}, op_{symm} and op_{func} are respectively denoting transitive, symmetric and functional object properties.*
$< i_1, op_{trans}, i_2 >, < i_2, op_{trans}, i_3 > \Rightarrow < i_1, op_{trans}, i_3 >$;
$< i_1, op_{symm}, i_2 > \Rightarrow < i_2, op_{symm}, i_1 >$;
$< i_1, op_{func}, i_2 >, < i_1, op_{func}, i_3 > \Rightarrow < i_2, equals, i_3 >$.

3.3 Semantic Relationship Search

Besides single entities, their relationships are more interesting and challenging. Three typical relationships further embody semantic features. Intuitively, the semantic relationship can be seen as the semantic path or reachability between two semantic entities. With such analogy, related graphic theories and algorithms can be reused and semantically enhanced.

Definition 3 (Relationship Search) *Given e_1 and e_2, the search portal extracts relevant semantic relationships between e_1 and e_2 as search result set RS; according to the three relationship divisions among concepts and instances, there are three sub definitions on this:*
*1, **Concept2Concept Search**: Given c_1 and c_2, the portal extracts all the r between them, hereby r can be op or complex relationship as a op series (denoted as op_s), thus $RS(c_1, c_2)$=$\{r\}$,r=$op|op_s$*
*2, **Concept2Instance Search**: Given c and i, the portal extracts only one exact relation like 0/1 boolean, i.e. $RS(c, i) = hasInstance$ or $RS(c, i) = \emptyset$*
*3, **Instance2Instance Search**: Given i_1 and i_2, the portal extracts relevant relationships between them.* Instance2Instance *involves more about op_s than* Concept2Concept *has.*

The following algorithm provides a detailed procedure for searching relationship between two concepts $c1$ and $c2$. Some reasoning rules mentioned previously are concerned, such as Rule 1. A good case in point is that the concept "*Paper*" attains the "*hasAuthor*" relationship to "*Student*" as a heritage from its super concept "*Person*". Basically, the algorithm only focuses on the relationship *op* not op_s between concepts.

Algorithm 3. Semantic Relationship between two Concepts $c1$ and $c2$

```
1  Set resultSet = new Set ();
2  //get relation from c1 to c2
3  Set c2Ancestor = c2.getSuperConcepts();
4  for ( each op in c1.getObjectProperties()) do
5    if (op.getRange c2Ancestor || op.getRange== c2)
6      resultSet = resultSet + { <c1, op, c2>}
7  end for
8  // get indirect relation from c1 to c2
```

```
9   Set c1Ancestor = c1.getSuperConcepts();
10  for ( each op in c2.getObjectProperties()) do
11    if (op.getRange c1Ancestor || op.getRange== c1)
12      resultSet = resultSet + { <c2, op, c1>}
13  end for
14  return resultSet;
```

About *Concept2Insatnce*, there are only two exact ones, namely "*hasInstance*" or "*NULL*". The Rule 1 also plays an important role in this algorithm.

Algorithm 4. Semantic Relationship between Concept c and Instance i

```
1   // judge isBelongTo relation by direct determine
2   if (i difinedAs c )
3     return {hasInstance};
4   // judge isBelongTo relation by indirect determine
5   Concept[] cOffspring = c.getSubConcepts();
6   for (each concept _i in cOffspring)
7     if (i difinedAs concept _i )
8         return {hasInstance};
9   endfor
10  return NULL;
```

Compared with the former two kinds of relationships, *Instance2Instance* is more challenging because of its abundance. Some domain-rules are involved like Rule 3, and Rule 4 is a more complicated one, requiring logic foundation such as the quantifier EXISTS and FORALL in first-order logic.

Rule 3 *two persons authoring the same paper can be seen as "coauthors",*
$< i_{pape}, hasAuth, i_{pers1} >, < i_{pape}, hasAuth, i_{pers2} > \Rightarrow < i_{pers1}, coauthor, i_{pers2} >$

Rule 4 *the person who publish the first paper on a given topic is the topic's inventor. Two sub-rules achieve it: the first attains the topic's first paper, and the second concludes the topic's inventor.*
1). $< \forall i_{pape\_k}, hasTopic, i_{topi} >, < i_{pape\_k}, pubTime, i_{time\_k} >,$
$< i_{pape}, pubTime, i_{time} >, < i_{time\_k}, notPrev, i_{time} > \Rightarrow < i_{topi}, firstPaper, i_{pape} >$
2). $< i_{topi}, firstPaper, i_{pape} >, < i_{pape}, hasAuthor, i_{pers} > \Rightarrow < i_{topi}, inventor, i_{pers} >$

4 Implementation and Evaluation

The domain oriented and independent semantic resources search model has been implemented in *Research Community* and other scenarios like *Culture Archives* [15]. In our experiences, OWL is processed by Jena, whist database application is enhanced by J2EE lightweight frameworks such as Struts and Hibernate, which contribute a lot to the final system performance. Compared with traditional resource search, our semantic search for domain resources in this paper can provide more integrated and reasonable search results. The reason is that the model has solid semantic foundation and clear division of two typical entities and three categorial relationships. The detailed valuable metrics are: for the entity search, the results are presented as semantic entities list not simple text

list, and each entity has the individualized semantic exhibition according to the inward schema; for the relationship search, more implicit semantic relationships can be extracted with inference supports including domain-oriented reasoning rules, formulas and algorithms; furthermore, the exhibition of search results can be semantically repacked in more intuitive way with rich semantics.

5 Conclusion

In this paper, we have proposed a domain-oriented semantic search model, which is fully-fledged and domain-independent. The model provides semantic search portal to support semantic resource search, involving two typical entities (*concepts* and *instances*) and three kinds of relationships (*Concept2Concept*, *Concept2Instance* and *Instance2Instance*). To realize the model, we have refined a four-layered semantic architecture with the vertical support of semantic inference, which is gradually achieved by domain-oriented reasoning rules, formulas and algorithms. We have mainly validated the semantic search model in *research community* and *culture archives*. Our future work is to further refine and verify the model, especially to enhance the query results on two aspects: one is semantic ranking for relationships; the other aspect is to enhance semantic visualization for search results.

References

1. R. Guha, R. McCool, E. Miller, Semantic Search, WWW03, Page 700-709. 2003.
2. C.Rocha, D.Schwabe, E. Arnal, A hybrid approach for searching in the semantic web. WWW04, Page 374-383, 2004.
3. A. Calsavara, G. Schmidt, Semantic Search Engines, ISSADS04, 2004.
4. L.Zhang, Y. Yu, J. Zhou, Y. Yang, An enhanced model for searching in semantic portals, WWW05, Page453 - 462, Chiba, Japan, 2005.
5. Y. Lei, V. Uren, E. Motta , SemSearch: A Search Engine for the Semantic Web, EKAW'06, Page 238-245, Podebrady, Czech Republic, 2006.
6. S.Kruk, H.Krawczyk, Intelligent Resources Search in Virtual Libraries, Page 439-443, Intelligent Information Systems, 2004.
7. M. Taibi et cl, A Semantic Search Engine for Learning Resources, ICTE2005, Spain.
8. S. Staab, R. Studer, Handbook on Ontologies, Springer, January 2004.
9. Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, D. Oberle, The SWRC Ontology - Semantic Web for Research Communities, EPIA05, Covilha, Portugal, 2005.
10. J. Francisco et cl., HyCo - An Authoring Tool to Create Semantic Learning Objects for Web-Based E-learning Systems, Springer Volume 3140, Page 344-348, 2004
11. P. Dolog, W. Nejdl, Challenges and Benefits of the Semantic Web for User Modelling., AH'03, Adaptive Hypermedia, 2003
12. K. Anyanwu et cl.: Ranking Complex Relationship Search Results on the Semantic Web, WWW05, Chiba, Japan, 2005.
13. H.Zhuge, P.Zheng, Ranking Semantic-linked Network, WWW03, Hungary, 2003
14. F. Harmelen et cl., Ontology-based Information Visualization, 5th International Conference on Information Visualisation, Conference, Page 546-554, 2001
15. Z.Yan, F. Scharffe,Y.Ding, Semantic Search on Cross-media Cultural Archives, 5th Atlantic Web Intelligence Conference, France, 2007