Semantic Association Identification and Knowledge Discovery for National Security Applications[1]

Annual Progress Report (Year 1)

Report Date: April 30, 2003

Contributors:
Faculty: Prof. Amit Sheth (PI), Prof. I. Budak Arpinar (co-PI), Prof. Krys Kochut (co-PI)
Research Assistants: B. Aleman, K. Anyanwu, C. Ramakrishnan, B. Reed

1. Introduction

National security application, such as aviation security and terrorist threat assessments, represent significant unmet challenges, and provide excellent source for further research in developing next generation IT solutions. The emerging vision of Semantic Web provides an excellent multidisciplinary context for this research. It involves ability to semantically annotate information that formally describing information such that it is machine processable, and then use techniques for better, more automated utilization of information leading to more actionable and timely decision making.

With this overall context, and starting with an existing ability to automatically create semantic metadata, two important challenges our project seeks to address are (a) rapid identification of semantic associations involving entities (such as a passenger or a group of passengers on a flight), and (b) knowledge discovery that identify semantic associations of interest (such as those that may pose a risk). This report is the first annual report of the three year project. More project related material, including publications, is available from the project page: http://lsdis.cs.uga.edu/proj/SAI/

2. Progress Overview

We have met all the objectives and milestones of the first year of this year project to date. In particular, we have completed the following:

- formal description of semantic association involving different types of associations,
- naïve algorithms design and prototype implementation to support path discovery,
- design of a testbed for their evaluation, including an ontology in RDFS targeting certain aspects of national security domain, and a large knowledgebase in RDF,
- sample scenarios within the context of a terrorism analysis application that matches the testbed ontology and knowledgebase.

[1] This material is based upon work supported by the National Science Foundation under Grant No. **0219649**. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Three to four graduate students have been funded as research assistants and two of them are pursuing their dissertation in the area of this research. Two publications have resulted so far for this research, including one in the highly competitive WWW2003 (W3C's flagship conference), and more are in preparation. Additionally, the PI gave two keynotes and one invited talk at conferences and workshops during this year on the topics related to this project (details are provided at the end of the report).

Work in progress and planned includes the critical scalability issues for discovering semantic associations in very large knowledge bases. Context-aware ranking techniques are also under investigation. Future research items include development of an ontology-driven document classification technique.

The overall progress is summarized in the following table.

#	Description	Milestone (month ending)	Status
1	Application requirements with emphasis on understanding types/classification of semantic associations, and types of knowledge discovery which is possible using semantic metadata	Initial study (12); revised study (24)	Completed; publications describing semantic associations
2	Testbed for using evaluating techniques with open source data, involving metadata extraction, knowledge base and InfoQuilt tools for ontology creation	Testbed for 1-st round evaluation (12); testbed for 2-nd round evaluation (24)	Prototype ontology defined for national security domain in RDFS; knowledgebase built and being extended; path and isomorphic path discovery techniques implemented in testbed
3	Create scenarios in accordance to requirements (1) and set up the evaluation using our testbed (2)	Initial evaluation (18); final evaluation (36)	Sample scenarios captured through extracting relevant data from Web sources
4	Research 1: Ontology-based Metadata Extraction • Develop an automatic metadata extractor and document annotator based on ontological document classification • Develop a lazy annotation mechanism where a document will be anchored to the target ontology, and the full meta-data will be dynamically materialized at the time it is needed	Model development (12); initial implementation and evaluation (18); extensions and final evaluation (36)	Use partner technology for automatic metadata extraction based on ontology developed; Future research on lazy annotation-- approach outlined in this report
5	Research 2: Knowledge Discovery • Develop association discovery techniques on an interconnected set of concepts and relations (i.e., ontologies) and their instances in the knowledge base • Define and formalize target semantic association types; develop ranking schemes to designate most interesting associations among many discovered associations	Model development (12); implementation and initial evaluation (18); extensions final evaluation (30)	Basic discovery techniques implemented; scalability issues under consideration; context-aware ranking techniques being developed

The rest of this report consists of the following sections. Section 3 discusses formal description of Semantic Associations. Section 4 discusses our implementation of three of the semantic association operators. Section 5 discusses the testbed. Section 6 discusses ongoing and future work.

3. Semantic Association Specification

We will now illustrate Semantic Associations by way of a simple example shown in Figure 1. The figure shows an RDF model base containing information to be used in the development of a cultural portal, given from two perspectives, reflected in two different schemas (the top part of the figure). The top left section of the picture is a schema that reflects a museum specialist's perspective of the domains using concepts like `Museum`, `Artist`, `Artifact`, etc. The top right section is a schema that reflects a Portal administrator's perspective of the domains using administrative metadata concepts like `file-size`, `mime-type`, etc. to describe resources. The lower part of the figure is the `model base` (or `description base` in the lingo of [9]), that has descriptions about some Web resources, e.g., museum websites (`&r3`, `&r8`), images of artifacts (`&r2`, `&r5`, `&r7`) and for resources that are not directly present on the Web, e.g., people, nodes representing electronic surrogates are created (`&r1`, `&r4`, `&r6` for the artists Pablo Picasso, Rembrandt, and Rodin August respectively).

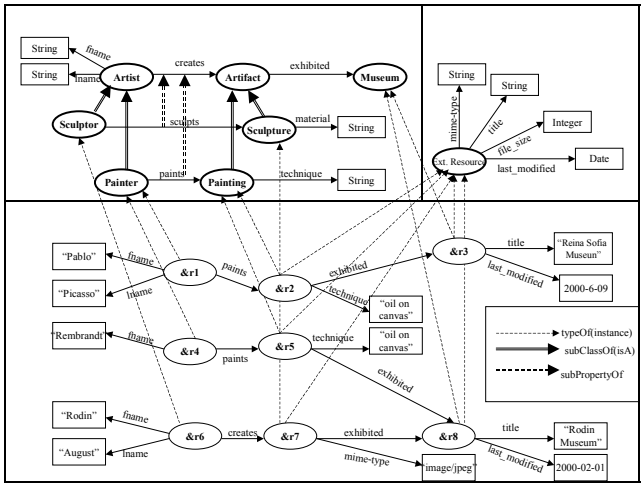

Figure 1: Cultural Portal Information in RDF

Typically, a query language allows you to find all entities that are related by a specific relationship. For example, we may ask a query to retrieve all resources related to resource `&r1` via a `paints` relationship, or via a `paints.exhibited` relationship, and get &r2 as a result for the first query and &r3 as the answer for the second query. However, we are unable to ask queries such as "`How` are resources &r1 and &r3 related? Such a query should return for example that "&r1 `paints` &r2 which is `exhibited` in &r3", indicating a path connecting the two entities. With a query such as this one, the user is trying to determine `if` there is a relationship between entities, and `what` the nature of

the relationship(s) is(are). It should be possible to ask such a query without any type of specification as to the nature of the relationship, such as using a path expression to give information about the structure of the relationship. For example, the following example RQL query

```
select * from {;Artist}@P{X}.{;Sculpture}@Q{Y}.@R{Z}
```

finds all data paths that traverse the class hierarchies `Artist` and `Sculpture`, containing three schema properties, one for each property variable (@variable). However, we notice that the query requires that a property variable be added for every edge in the required path. That is, the user is required to have some idea of at least the structure e.g. length, of the relationship. One approach that some of these systems offer to alleviate this problem is that they provide mechanisms for browsing or querying schemas to allow users to get the information they need. While this may be a reasonable requirement when querying specific domains with a few schemas involved, on the Semantic Web, many schemas may be involved in a query, and requiring a user to browse them all would be a daunting task for the user. In fact, in some cases, such information may not be available to all users (e.g., classified information) even though the data may be used indirectly to answer queries. Furthermore, browsing schemas do not always give the complete picture, especially in the case of RDFS schemas, because, entities may belong to different schemas, creating links between entities that are not obvious from just looking at the schemas. For example in Figure 1, the relationship `paints.exhibited.title` connecting &r1 to "Reina Soifa Museum", is not apparent by just looking at either schema.

So far, we have talked about relationships in terms of a directed path connecting two entities. However, there are some other interesting types of relationships. Let us take for example, resources &r4 and &r6. Both resources could be said to be related because they have both created artifacts (&r5, and &r7) that are exhibited at the *same* museum (&r8). In this case, having some relationship to the *same* museum associates both resources. This kind of connectivity is an undirected path between the entities. Another closely related kind of association is class membership. For example, &r1 and &r6 are both Artists, even though of a different kind, and therefore are somewhat associated. Also, &r1 and &r6 could be said to be associated because they both have creations (&r2, and &r7) that are exhibited by a Museum (&r3 and &r8 respectively). In this case, the association is that of a *similarity*. So, in the first three associations the relationships capture some kind of connectivity between entities, while the last association captures a similarity between entities. Note that the notion of similarity used here is not just a structural similarity, but a semantic similarity of paths (nodes and edges) that the entities are involved in. Nodes are considered similar, if they have a common ancestor class. For example in the relationship involving &r1 and &r6, although one case involves a painting and the other a sculpture, we consider them similar because sculptures and paintings are kinds of Artifacts and sculpting and painting are both kinds of creative activities (the notion of similarity is extended to properties as well).

The Semantic Associations shown in this example are fairly simple involving only short paths and are useful only for the purpose of illustration. However, in environments that support information analytics and knowledge discovery involve longer paths, especially

undirected paths, which are not easily detectable by users in fast-paced environments. For example at airport security portals, agents may want to quickly determine if a passenger has any kind of link to terrorist organizations or activities. This work is discussed in detail in [5] and [6].

4. Implementation of ρ Operators

In this section we describe and discuss the implementation of the three ρ operators on the RDF data. We use the domain of terrorism with the analysis (knowledge discovery) application for our test-bed. We have identified and extracted two test suites. The ontology designed for national security domain in RDFS is shown Figure 2.

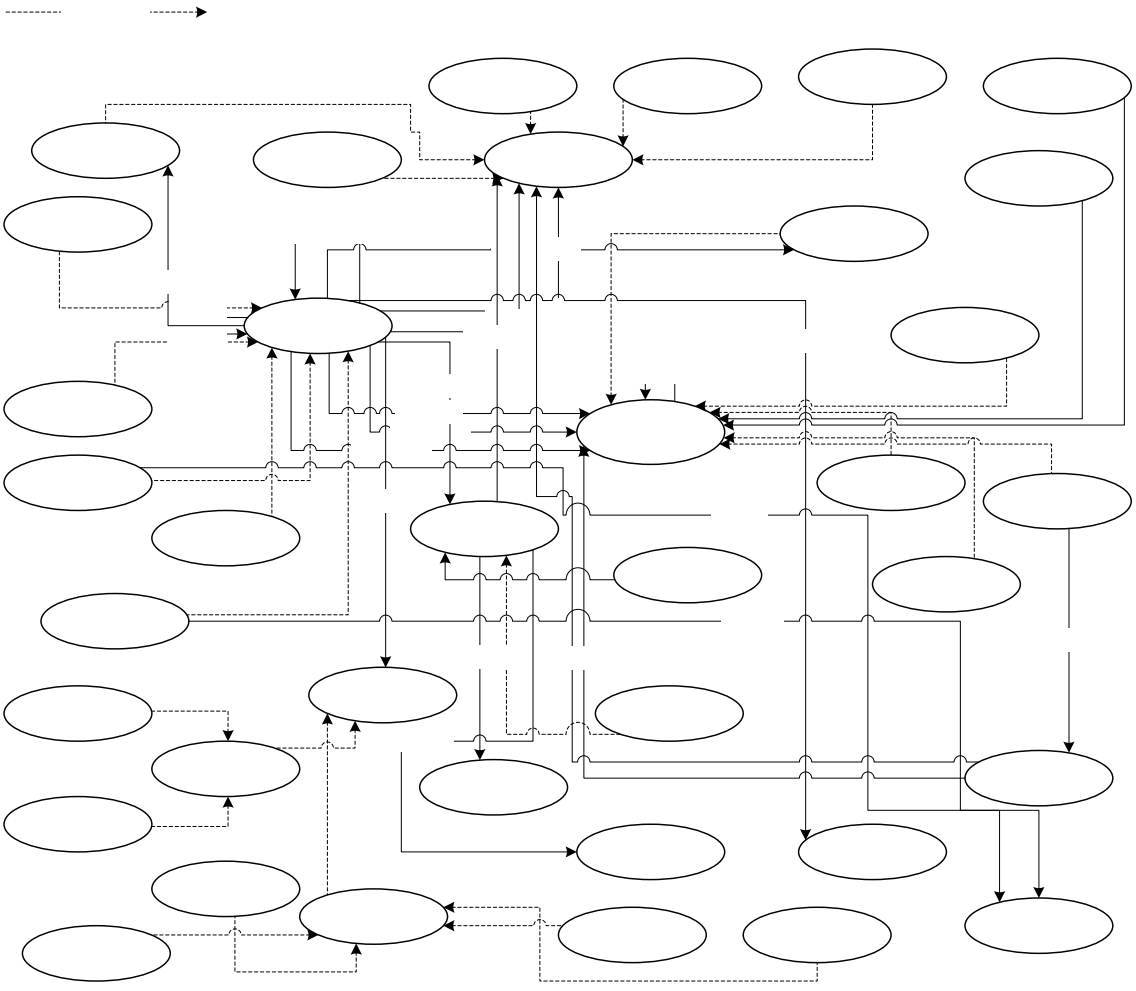

Figure 2. Ontology

Heuristic based search

We have implemented the simple search algorithms for each of the following operators. These algorithms use the schema information in conjunction with the RDF data that find path sequences that represent the relationships between any two resources. The figure below shows the (partial) visualization of the instance data with respect to which the results of the 3 operators are shown.

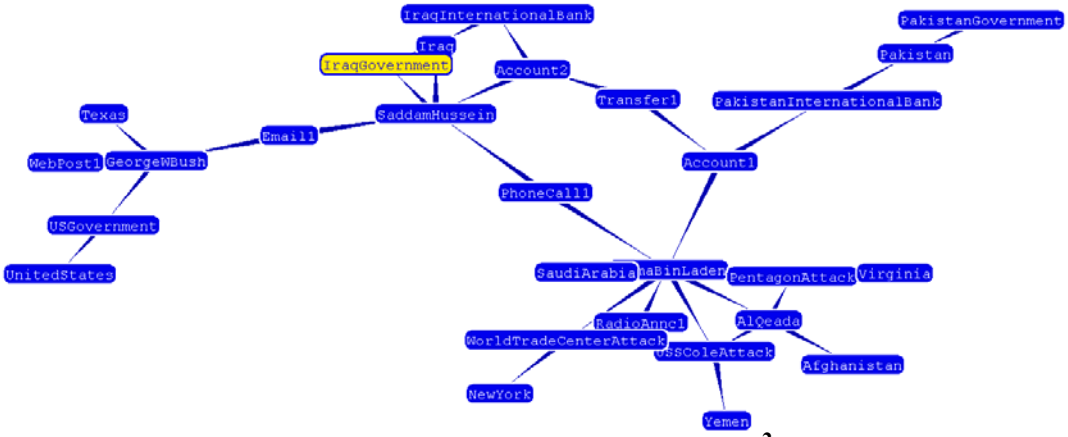

Figure 3. (Partial) Knowledgebase[2]

ρ-path

The naïve algorithm to find all paths between 2 nodes in a directed graph [1] shown below is a recursive implementation of a depth-first search. Our first implementation of the `ρ-path` operator is based on this algorithm. The statement `path = path + [start]` is not equivalent to the append operation; instead it creates a new list.

```
def find_all_paths(graph, start, end, path=[]):

        path = path + [start]

        if start == end:
            return [path]

        if not graph.has_key(start):
            return []

        paths = []

        for node in graph[start]:
            if node not in path:
                newpaths = find_all_paths(graph, node, end, path)
                for newpath in newpaths:
                    paths.append(newpath)

        return paths
```

[2] The arcs connecting the nodes shown in all the graphs of this type, are directional. These arcs taper along the intended direction of the association.

The test suite has at least one RDF schema and RDF data based on this schema. The basic idea in reducing the complexity of the above algorithm is to use the information from the schema level to prune the search in at the data level. The nodes at the schema level are far fewer than those at the data level. Hence the any search running at the schema level will take less time than the at the data level. We represent both the RDF schema and the RDF data as main memory directed graphs based on the JENA [2] model.

Concept 1: When looking for all paths between resources $\mathbf{r_1}$ and $\mathbf{r_2}$ in the graph representing the RDF data, $\mathbf{G_{data}}$, check if the classes $\mathbf{c_1}$ to which $\mathbf{r_1}$ belongs and $\mathbf{c_2}$ to which $\mathbf{r_2}$ belongs have a path between them in the schema graph $\mathbf{G_{schema}}$. If there is such a path then find all such paths first. These schema path expressions will then be used to prune the list of successor nodes for every state in the search through $\mathbf{G_{data}}$.

Based on Concept 1 we have implemented the ρ-path operator. Figure 4 shown below shows the visualization of such a set of paths that our algorithm finds.

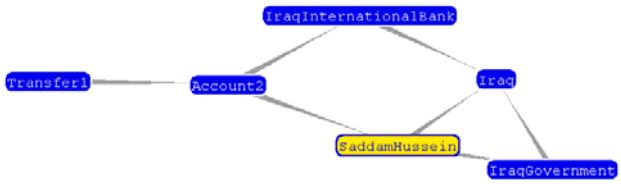

Figure 4. ρ-Path between Transfer1 and Iraq

Transfer1 → Account2 → IraqInternationalBank → Iraq
Transfer1 → Account2 → SaddamHussein → Iraq
Transfer1 → Account2 → SaddamHussein → IraqGovernment → Iraq

ρ-Intersect

Our initial implementation of the ρ-Intersect operator is based on the ρ-path operator. It searches for nodes where two ρ-paths intersect (see Figure 5). We recognize the fact that there could be multiple intersection points for the ρ-paths. Hence our implementation returns the sequence of nodes that are common between 2 ρ-paths.

Our implementation of the ρ-Intersect operator iterates over all the resources in the given model and attempts to find paths (using ρ-path) between the two input resources and every resource in the model. Thus we get a pair of paths, such that each path in the pair starts at one of the two input resources. If there is an intersection then these two paths will end at the same resource in the model. Although this is not the most efficient approach we plan to use it in a benchmark to compare it with future techniques. Our later, more efficient implementations of this algorithm will be compared to this naïve implementation with respect to the complexity (time and space).

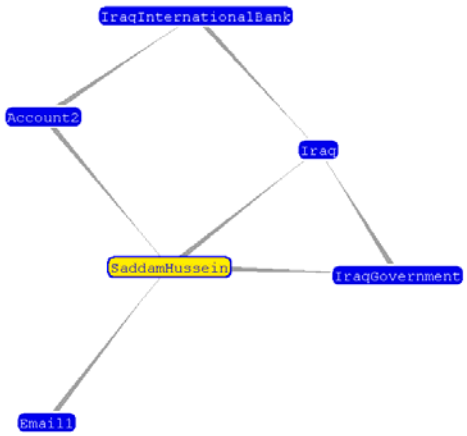

Figure 5. ρ-Intersect originating at Account2 & Email1

Account2 → IraqInternationalBank → Iraq Email1 → SaddamHussein → Iraq
Account2 → IraqInternationalBank → Iraq Email1 → SaddamHussein → IraqGovernment → Iraq
Account2 → SaddamHussein Email1 → SaddamHussein

ρ-Iso

The goal of ρ-Iso is to take two resources as input and discover all paths that are "isomorphic" in both resources (see Figure 6). A path discovered by ρ-Iso is a path that is similar in both resources. By similar we mean that a labeled edge (representing a property) exists in both resources. For example, the property "participatesIn" appears in two "Person" entities. However, there is more flexibility in how two properties are similar. In the RDF model, it is possible to have a *hierarchy* of properties. An example is a property "memberOf" which has a sub-property "leaderOf". The property "leaderOf" is a specialization of "memberOf", that is, a leader of an organization is as well a member of the organization. The resources connecting properties may also be "similar". If two resources are of the same class they are considered similar. However, we also consider as similar two resources that belong to different classes as long as the subclasses to which they belong share a common parent. From a *hierarchy* perspective this means that two resources that are "siblings" are similar in our implementation. The third and last similarity occurs when a resource belongs to a subclass of the other resource. That is, they both have a common parent class.

An example of a ρ-Iso path relating two persons is two persons that received training considered as possible threat, one took fire arms training courses and the other took flight courses. ρ-Iso, however, discovers paths that may span several relations and resources. Thus, ρ-Iso finds that two persons are related to a terrorist organization through a series of associations that span *similar* relations and resources.

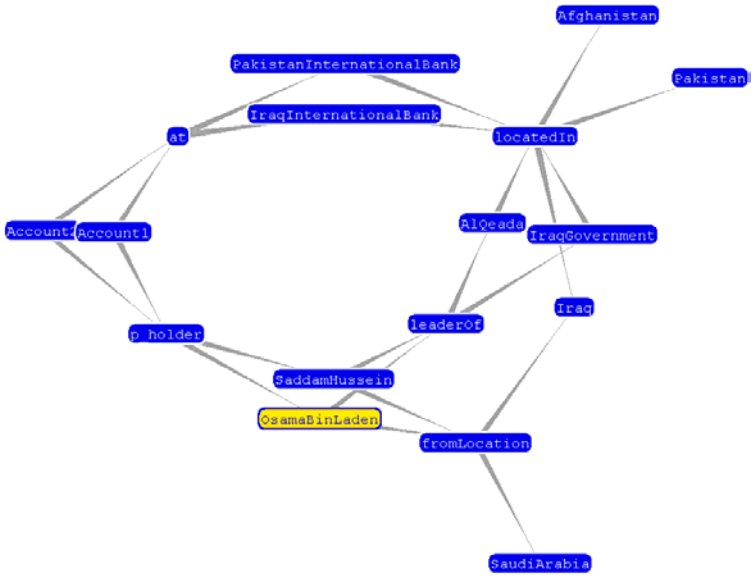

Figure 6. ρ-Iso between Account2 & Account1

Account2 → at → IraqInternationalBank → locatedIn → Iraq Account1 → at → PakistanInternationalBank → locatedIn → Pakistan
Account2 → p_holder → SaddamHussein → fromLocation → Iraq Account1 → p_holder → OsamaBinLaden → fromLocation → SaudiArabia
Account2 → p_holder → SaddamHussein → leaderOf → IraqGovernment → locatedIn → Iraq Account1 → p_holder → OsamaBinLaden → leaderOf → AlQeada → locatedIn → Afghanistan

5. Testbed

A prototype implementation of the proposed system (PISTA) and a testbed are completed. The testbed enables testing of the algorithms using massive amounts of entities and relationships between them in the ontology and knowledgebase. The PISTA architecture is outlined in Figure 7.

Ontology

The first step was to develop an ontology that captures the different classes of the entities. The initial development of the ontology included enough classes and relationships to capture a set of example scenarios. This was further refined and by using extraction technology of Freedom product from Semagix (which commercialized earlier SCORE technology related research at the LSDIS lab) [8] it was possible to define the classes and the relationships between them in a timely manner. The Freedom software from Semagix [8] includes a set of tools for extraction of entities from (semi)-structured sources. This toolkit allows extraction of entities in Web pages and establishes relationships between them. This extraction is based on the ontology thereby placing an extracted entity in its appropriate place in the hierarchy of classes.

The software has its own internal representation of the information and the ontology. It provides however, an interface that allows external programs to query it. In order to satisfy our needs, we developed algorithms that capture the internal representation of

Semagix's Freedom product and then transform it to the RDF model. Our algorithms are designed to process data represented using the RDF model. This allowed for verification of the validity of our schema with well-known RDF parsers in the Semantic Web community. Figure 1 shows the ontology in a graphical representation obtained by using *RDF Validator* from the W3C website.

The ontology so far contains 70 classes and 20 properties relating them. Note that from among those properties, most of them apply to classes located high in the ontology. Therefore, the properties will apply to all sub-classes of those classes. The ontology alone, when written to disk in RDF/XML serialized syntax has a file size of 18 KB of data.

Information Sources

The sources from which data were extracted were selected to populate the ontology with entities related to national security. Several terrorist databases were extracted as well as well known sources providing information about locations (cities, countries, etc.).

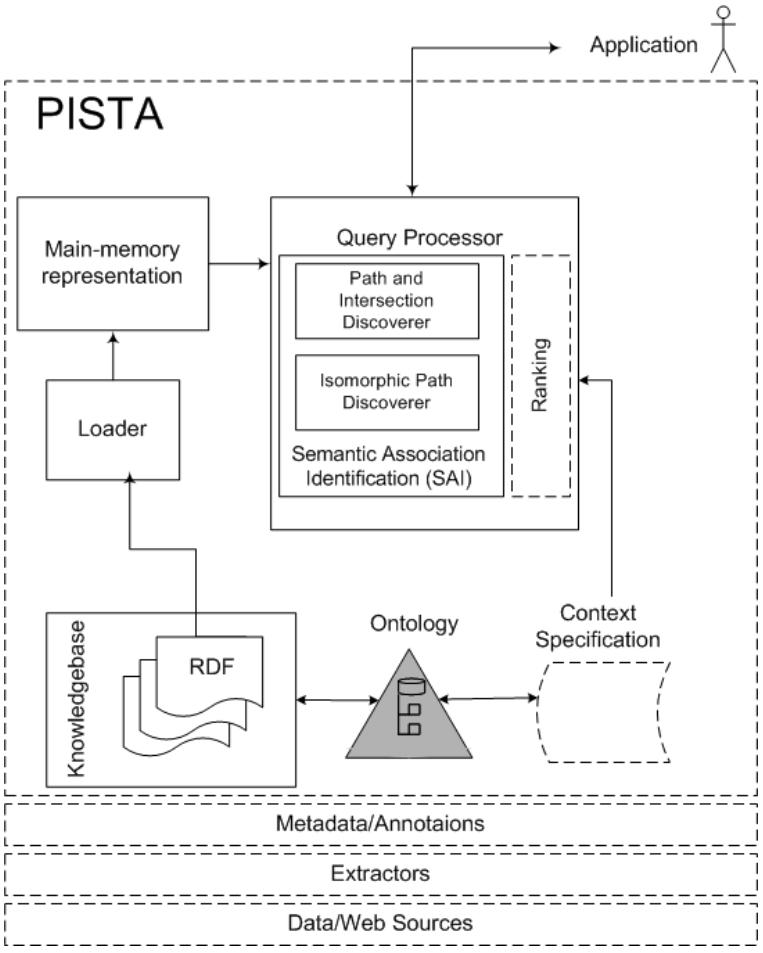

Figure 7. PISTA Architecture

Instances in the ontology

As mentioned above, entities were extracted from Web sources. Many entities appear multiple times in different sources. A process identifies that an entity already exists in order to avoid duplicates. The entities (RDF Data) are stored in serialized RDF/XML syntax in a separate file than that of the ontology (RDF Schema). Thus far, the size of the file with the RDF data is of 1.6 MB. There are 6,000 entities. These entities have relationships between them; the number of explicit relationships among the entities is over 11,000. This does not account for the relationships that are implicit in the RDF Schema describing the ontology.

In summary, during the initial period semantic associations are defined and a testbed involving a comprehensive ontology and a large knowledgebase is built. Algorithms for finding different type of associations including isomorphic paths are implemented.

6. Ongoing and Future Work

The emphasis of the first phase of our work was to gain a better understanding of the issues that arise in developing algorithms for ρ-query processing. We achieved this through prototyping a preliminary version of our Semantic Association Identification system. The algorithms used in this prototype are based on graph traversal algorithms with slight optimizations using heuristics based on data schemas. It should not be surprising however, that any strategies for computing *all* paths, and consequently Semantic Associations, that depend on traditional graph traversal algorithms (depth-first search or breadth-first search) will tend to have to exponential time complexity in the worst case. While this may be sufficient for small applications where data representation results in relatively small and sparse graphs, it will not scale well in very large scale environments such as the Web. Consequently, the major emphasis of the next phase of our work will be on performance enhancement. That is, the development of novel data structures and algorithms that will yield efficient processing of ρ-queries.

We propose a two-pronged approach to performance enhancement. First, is the development of data structures and algorithms for efficient ρ-query processing. Second, is the introduction of an explicit notion of context that will allow searches to be delineated to a specific set of data that is relevant for the user, thereby reducing the search space and discovering only context-specific Semantic Associations.

Another important issue we investigate currently is that of ranking. Just as search engines have ranking strategies that enable them determine which documents are most relevant and should be returned first, we develop ranking strategies for determining which Semantic Associations are most relevant. We use context as a vital element in determining relevance rankings of results.

Development of Data Structures and Algorithms for Efficient ρ-Query Processing

Algorithms

The algorithms we are currently investigating will build upon the work reported in Tarjan [7] where fast algorithms (near linear time) for solving path problems are presented. The approach discussed solves path problems using a two-phase approach. The first stage is the computation of a *Path Sequence* of the input graph, a notion introduced in the paper. Then, computations may be performed on the Path Sequence to answer path related questions, such as finding all paths between nodes in a graph. The output of the algorithm is a *Path Expression* of all the paths found. Essentially, a Path Expression is a regular expression over the alphabet $\sum$ of edge labels in the graph, where a string in the language of the path expression yields a path in the graph.

Our attempt to adapt this algorithm for ρ-query processing has uncovered two main challenges, which we are now focusing on. First, is an effective means of representation for graph and Path Expressions. Second, is the processing of the resultant Path Expressions- its expansion, ranking and presentation to the user. Finally, Tarjan [7] discusses some options for optimization; it is likely that further optimizations may be possible if we exploit the semantics of nodes and edges in the graph. We will now discuss these issues in some more detail.

Data Structures for Path Expressions

Regular expressions are useful in that they provide compact representations of large amounts of information. For example, an infinite number of paths can easily represented in a Path Expression using the closure ($*$) operator. However, we need to develop representations for these structures in both in memory and on disk. Furthermore, the representations should be amenable to efficient computation of regular expression operations ($\cup$, $*$, **.**), and other application-specific computations such as result pruning, (discussed later in section 2.1). We are considering some implicit representations (e.g. a graph) such as is used in compiler construction [Aho98], or complex programming language data structures built using primitive data types. Of-course, it is entirely possible to avoid explicitly constructing such a representation but rather to interpret the regular expression operations within the path finding algorithm. However, the explicit construction will allow for reuse of results. For example, the results of a query that finds all paths involving an entity A could be used to find paths between entity A and another entity B. In particular, we could reuse the results of the first stage (Path Sequence generation) of the first query to answer the second. This also opens up the possibility of indexing these Path Expressions for quick access at reuse time.

We envisage the use of Path Indexes that play an analogous role to those used by semi-structured data query, e.g. the Dataguides [1] used in the LORE project, which provide structural summaries of data graphs for retrieving entities. Because, the focus of these query systems is to retrieve entities in contrast to ours which is to retrieve complex relationships, such indexes are not appropriate for our purposes. Consequently, we plan to develop novel index structures that are appropriate for supporting ρ-query processing.

These index structures will allow us to retrieve quickly paths that exist either between entities classes or between specific entities. Constant time access to path information will remove most of the overhead for query processing making the discovery of the more complex Semantic Associations such as ρ-joinAssociativity and ρ-Isomorphic Associations much more efficient.

The Role of Context

For a given pair of entities, there are potentially a large number of Semantic Associations that link them, however not all of them would be considered important or relevant for a specific context, and returning all of the results will lead to an information overload problem similar to what happens as a result of a Web search. Therefore, we need a context-sensitive approach for query-processing and post-processing (pruning) of results. The resulting path expressions provide a compact means for representing what could potentially be an exponential number of paths, which must be pruned to contain only those results that are relevant in the query context.

To achieve this, we try to formalize the notion of context as well as its representation that will support such processing. In Information Retrieval, the context of a query amounts to the set of keywords in the query, and any document that contains a subset of the keywords is considered relevant. In our approach, there are several other factors that contribute to the context of a query, e.g., the kind of component relationships (i.e. rdf:properties) of the Semantic Association, the task which is being supported by the query results, domain-specific knowledge such as that used by experts for determination of relevance. For example, in the context of assessing flight security, information about which country the passenger is a citizen of, whether the passengers have flight training, any relationship to countries or people on the government's watchlist will be considered relevant. On the other hand, what meal the passenger had in the hours preceding the flight will not be considered relevant. However, if a passenger should suddenly become ill, a medical staff evaluating that passenger will likely find such information of prime importance.

Ranking

Another important related issue is the issue ranking results when several results are found, which is likely to be a common scenario. This issue is apart from the issue discussed in the preceding section, which talked about what kinds of Semantic Associations should be members of the result set. In situations where many Semantic Associations are found, we need a means for determining the relative importance of each member of the result set.

We believe that ranking strategies must incorporate contextual knowledge in order to yield meaningful results. This means that the notion of context we use should capture adequate information for determination of relevance. Ranking schemes may also incorporate other general factors such as the length of the Semantic Association. For example, in the absence of contextual information to the contrary, shorter paths may be generally assigned more relevance than longer paths, based on the intuition that the strength of a relationship or Semantic Associations wanes with increased distance.

Additionally, ranking strategies should be adaptive, unlike the fixed strategies used by search engines where ranks are pre-assigned to documents based on a fixed notion of importance. Because the context of a query can change, the same data may be considered very important in one context, and much less relevant in another.

Lazy Annotation

Ontologies provide the context for metadata extraction and annotation. The first step is to identify relevant ontologies, namely classification. The second step is to perform contextual processing of data/content to extract ontology driven semantic metadata, which can then be expressed in a suitable representation. This approach helps in eliminating the data and media heterogeneity because the heterogeneous content is classified using a common set of (yet user selectable) ontologies and meta-data is extracted accordingly. Furthermore, it is possible to annotate these documents using different ontologies in a *lazy* fashion, which provides a greater flexibility in analyzing the documents.

We intend to use existing work on classifier committee based automatic classification available to us from our partner Semagix [8]. After classification, creating document annotations is the next important task in enabling sharing of knowledge, both for integration and querying of distributed knowledge sources. A given document may present different information when it is analyzed with respect to different, but potentially related ontologies. For example, a meaningful annotation of the business-related financial document may be created, which would include the known facts (entities and relationships among them) using the business ontology. Furthermore, a completely different annotation may be created using the ontology dealing with terrorism, which would include and highlight various uncovered relationships between companies and terrorist organizations, their members and possibly financial dependencies connecting them.

Since ontologies may be created and/or modified frequently (as new information on terrorism is uncovered), it is impossible to create all interesting annotations at the time of document creation. We intend to use some of the existsing classification methods (Score) to create several meaningful annotations for the document. Furthermore, we intend to view an annotation of the document as *lazy annotation* in the sense that a document's meta-data will not be created and attached to the document. Rather, a document will be anchored to the target ontology, and the full meta-data will be materialized at the time it is needed. This will have the advantage of using the provided reasoning mechanism in order to extract other meta-data about the document at the time the data is needed. Any changes made to the ontology relationships, or updates to extensional part of the ontology will be materialized as well. This concept of lazy annotation can be extended to a WWW URL that represents a frequently modified content (e.g. a news page).

We intend to investigate lazy as well as eager annotations of documents, as well as investigate the effectiveness of this technique particularly on dynamic content sources.

REFERENCES

[1] J. McHugh, J. Widom, S. Abiteboul, Q. Luo, and A. Rajamaran. Indexing Semistructured Data. Technical report, Stanford University, Computer Science Department, 1998. http://citeseer.nj.nec.com/mchugh98indexing.html

[2] A. Aho, R. Sethi, J. Ullman. Compiler. Principles, Techniques and Tools. Addison Wesley Longman. 1988.

[3] http://www.python.org/doc/essays/graphs.html

[4] http://www.hpl.hp.com/semweb/doc/tutorial/index.html

[5] K. Anyanwu and A. Sheth. The ρ Operator: Discovering and Ranking Associations on the Semantic Web, SIGMOD Record, Vol. 31, No. 4, December 2002, pp. 42-47.

[6] K. Anyanwu and A. Sheth. "The ρ Operator: Discovering and Ranking Associations on the Semantic Web," The Twelfth International World Wide Web Conference, Budapest, Hungary, May 2003.

[7] R. Tarjan. Fast Algorithms for Solving Path Problems. J. ACM Vol. 28, No. 3, July 1981, pp.594-614.

[8] A. Sheth, C. Bertram, D. Avant, B. Hammond, K. Kochut, and Y. Warke. Semantic Content Management for Enterprises and the Web, IEEE Internet Computing, July/August 2002, pp. 80-87.

[9] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis and K. Tolle. The RDFSuite: Managing Voluminous RDF Description Bases. In: Proc. of the 2nd Int. Workshop on the Semantic Web, Hong-Kong, 2001.