Semantic Web & Ontologies

Jun Cai Vladimir Eske Xueqiang Wang

1. Semantic Web: Informational Retrieval System

1.1. Introduction

1.1.1. Motivation and concepts

What is The Semantic Web?

To define what is the Semantic Web is very difficult as well as Web itself.

"The Semantic Web is a mesh of information linked up in such a way as to be easily processable by machines, on a global scale."
"The Semantic Web approach develops languages for expressing information in a machine processable form. "

These two sentences define the essence of the SW:
It's information in machine processable form, however in the same time first one defines SW as a global scale information mesh and the second sentence defines it as Framework for expressing information.

Both citations demonstrate the main principle of the SW:
Information in Web should be more machine processable and understandable. In this case SW can be the goal (mesh of information) as well as a tool (language for expressing). It seems to be the main paradox of the SW: to be an egg and a chicken in the same time.

Why we should use SW?
We use Web as a global database first of all for search. Today's search engines can not search more precise that they do it now. May be the main reason is that the structure and size of current Web do not allow to make search more precise and efficient. The second reason can not be eliminated: Web contains now a huge number of documents and this number has a strong tendency to double each one or two years. The structure of documents and Web itself, probably, can be changed in "a better – machine processable way"

1.1.2. SW architecture

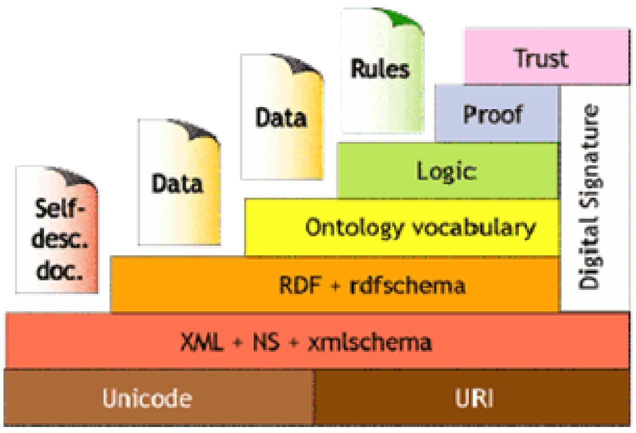


Figure 1. SW architecture

The SW architecture is presented on the figure 1 and includes the following main topics:

URI and Unicode.

The Semantic Web is generally built on syntaxes which use URIs to represent data, usually in triples based structures: i.e. many triples of URI data that can be held in databases, or interchanged on the world Wide Web using a set of particular syntaxes developed especially for the task. These syntaxes are called "Resource Description Framework" syntaxes. Unicode allows to support the international text style standard.

RDF and rdfschema.

Resource Description Framework (RDF) is:

- A general metadata format
- Used to represent information about Internet resources
- Semantic Web extends the expressive capability of the Web
- Augments human-readable web pages with machine-processable information

RDF's vocabulary description language, RDF Schema, is an extension of RDF. It provides mechanisms for describing groups of related resources and the relationships between these resources. RDF Schema vocabulary descriptions are written in RDF. The extra descriptive power of RDF Schema is carried in a collection of RDF resources described in this document.

Ontology

The term Ontology has been used in several disciplines, from philosophy, to knowledge engineering, where ontology is comprised of concepts, concept properties, relationships between concepts and constraints. Ontologies are defined independently from the actual data and reflect a common understanding of the semantics of the domain of discourse. Ontology is an explicit specification of a representational vocabulary for a domain; definitions of classes, relations, functions, constraints and other objects. Pragmatically, a common ontology defines the vocabulary with which queries and assertions are exchanged among software entities. Ontologies are not limited to conservative definitions, which in the traditional logic sense only introduce terminology and do not add any knowledge about the world. To specify a conceptualisation we need to state axioms that put constraints on the possible interpretations for the defined terms.

Inference

The principle of "inference" is quite a simple one: being able to derive new data from data that you already know. In a mathematical sense, querying is a form of inference (being able to infer some search results from a mass of data).

<MyCar de:Leistung "200kw"> and **<de:Leistung daml:equivalentTo en:Power>**
Here have been used the DAML "equivalentTo" property to say that "Leistung" in the German system is equivalent to "power" in the English system. Now, using an inference engine, a Semantic Web client could successfully determine that
<MyCar en:Power "200kw">

This is only a very simple example of inference, but you can see immediately how easily the system could scale up. Merging databases simply becomes a matter of recording in RDF somewhere that "Person Name" in your database is equivalent to "Name" in my database, and then throwing all of the information together and getting a processor to think about it.

1.1.3. SW technologies (DAML+OIL)

DAML+OIL is a description language which is a very important component of the SW. Many applications (e.g. OWLIR, which is described later) use this language for storing the semantic pages.

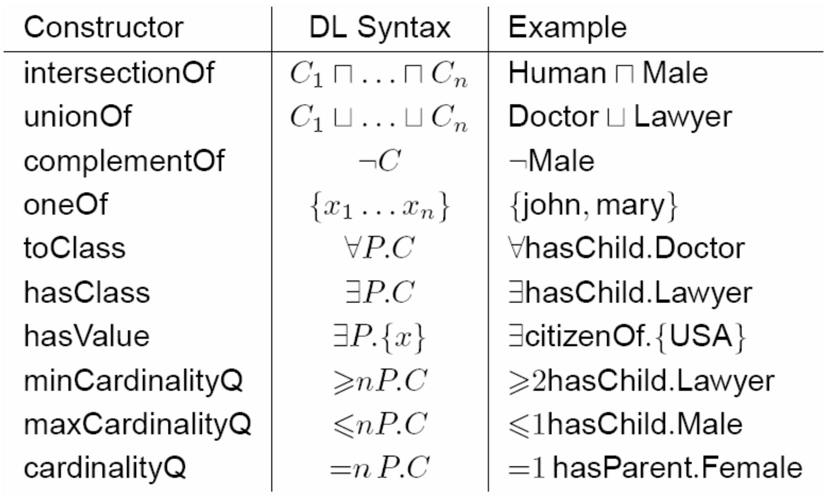
DAML+OIL is an ontology language which describes structure of the domain, RDF used in the same time to describe specific instances. Structure is described in terms of classes (concepts) and properties(roles). Ontology in DAML+OIL consists of set of axioms (E.g., asserting class subsumption/equivalence). Classes can be names or expressions and various constructors provided for building class expressions.
Expressive power of DAML+OIL is determined by

- Kinds of axiom supported
- Kinds of class (and property) constructor supported

DAML+OIL basic properties:

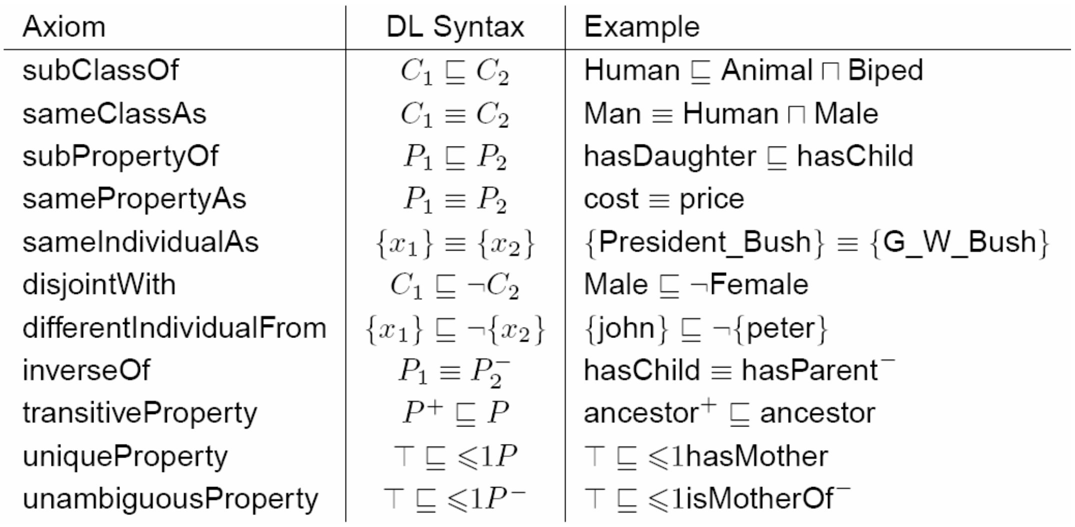
- DAML+OIL layered on top of RDFS
  - RDFS based syntax
  - Inherits RDFS ontological primitives (subclass, range, domain)
  - Provides much richer set of primitives (equality, cardinality, . . .)
- DAML+OIL designed to describe structure of domain (schema)
  - Object oriented: classes (concepts) and properties (roles)
  - DAML+OIL ontology consists of set of axioms asserting characteristics of classes and properties
  - E.g., Person is kind of Animal whose parents are Persons
- RDF used for class/property membership assertions (data)
  - E.g., John is an instance of Person; <John; Mary> is an instance of parent

DAML+OIL Class Constructors



Constructor	DL Syntax	Example
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human $\sqcap$ Male
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor $\sqcup$ Lawyer
complementOf	$\neg C$	$\neg$Male
oneOf	$\{x_1 \ldots x_n\}$	{john, mary}
toClass	$\forall P.C$	$\forall$hasChild.Doctor
hasClass	$\exists P.C$	$\exists$hasChild.Lawyer
hasValue	$\exists P.\{x\}$	$\exists$citizenOf.{USA}
minCardinalityQ	$\geqslant n P.C$	$\geqslant$2hasChild.Lawyer
maxCardinalityQ	$\leqslant n P.C$	$\leqslant$1hasChild.Male
cardinalityQ	$=n\, P.C$	$=$1 hasParent.Female

DAML+OIL Axioms



Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
sameClassAs	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
samePropertyAs	$P_1 \equiv P_2$	cost $\equiv$ price
sameIndividualAs	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
disjointWith	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq$ $\neg$Female
differentIndividualFrom	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq$ $\neg${peter}
inverseOf	$P_1 \equiv P_2^-$	hasChild $\equiv$ hasParent$^-$
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor
uniqueProperty	$\top \sqsubseteq \leqslant 1 P$	$\top \sqsubseteq \leqslant$1hasMother
unambiguousProperty	$\top \sqsubseteq \leqslant 1 P^-$	$\top \sqsubseteq \leqslant$1isMotherOf$^-$

1.2. Hybrid Informational Retrieval (HIR)

Semantic Web with all its features like semantic markups and ontologies allows to use additional search techniques. However, In the same time the standard IR approaches are still useful for making differences “semantically similar” (similarity only by additional features) to the query documents. SW IR system can use so called Hybrid Informational Retrieval.
HIR is a clever combination of semantic oriented search approaches with standard text retrieval techniques and can be splited logically into 2 parts:

Standard text IR

- Vector space model
- IDF_TF weighting
- Term indexing (n-grams)
- Similarity measuring (cosine measure)

“Semantic” IR

- Inference
- Markup similarity
- Markup/text relationships
- Ontology mapping
- “Transitive” querying

Document in this model is represented also in vector space model but in addition semantic markup or pair of markup-term can be defined as a term as well as words are occurred in document.

Markup/text relationship (statistics of occurring the pair markup/term in data collection) allows in the first step to convert the query, which is usually posted as simple text string into semantic (markup) form. Tags (markups) are associated with query term, if tag/term frequency exceeds some threshold.

Inference is a general preprocessing technique. Text data together with ontology and already existent semantic markup are used to infer additional semantic markups. These relations are used to decide the scope of the search and to provide more germane responses.

“Transitive” querying is an example of using “Inference” for search. Transitive query means that are you looking for information which does not present in a data collection directly. For example you ask: “parent Bob”, but in a data collection there is only a triple <Mary, Mother, Bob>. Today’s web search engines can not answer the query. But the inference machine even very primitive can easily transform the query <?, parent, Bob> into queries
<?, Mother, Bob> and <?, Father, Bob> or add a new triple into data collection <Mary, Parent, Bob>. Of course such action needs a special inference rule based machine which will contain the rules like <Mother, SubInstance, Parent>, <Father, SubInstance, Parent>.

Markup similarity allows to rank the result together with text term similarity. Query after it was processed in semantic form has nondeterministic form (with one text-term can be associated more then one markup). In this case query can be evaluated in two ways.

- The first one is essentially SW method: the structure of the query is sifted using ontology and the most relevant to the query concept is used to reduce search space.

- The second approach includes markup component in the ranking function. In this case similarity between query markup and concept taken from data collection has numerical equivalent.

Now all special semantic web techniques look like very clever engineer tricks. They have poor theoretical background, but in the same time they really make search more efficient.

1.3. OWLIR

OWLIR is an example of the Semantic Web IR system. It is an implemented system for retrieval of documents that contain both free text and semantic markup. OWLIR is only a framework, which was designed to work with almost any local information retrieval system.

1.3.1. OWLIR as HIR system

OWLIR (Ontology Web Language and Information Retrieval) focuses on addressing three scenarios that involve semantically marked up web pages and text documents:

- Information retrieval (IR) - e.g., identifies and rank relevant pages or documents for a query looking for detail descriptions concerning USA and Afghanistan leaders.
- Simple question answering (Q&A) - e.g., who is the president of the USA?
- Complex question answering - e.g., what is the current situation in Afghanistan?

OWLIR can be described in terms of two primary components: a set of ontologies and a hybrid information retrieval mechanism. OWLIR defines
ontologies encoded in DAML+OIL allowing users to specify their interests in different events.

There is a wide spectrum of techniques, which can be applied to address querying, and retrieval of semantically marked documents. OWLIR can help bypass some of the limitations of information access:

- Use semantic information for guiding the query answering process.
- Enable answers with a well-defined syntax and semantics that can directly be understood and further processed by automatic agents or other software tools.
- Provide information that is not directly represented as facts in the WWW, but which can be derived from other facts and some background knowledge.

Using these techniques together with inference and ontology mapping makes OWLIR Hybrid Information Retrieval system in content of part 2 of this report.

1.3.2. OWLIR architecture

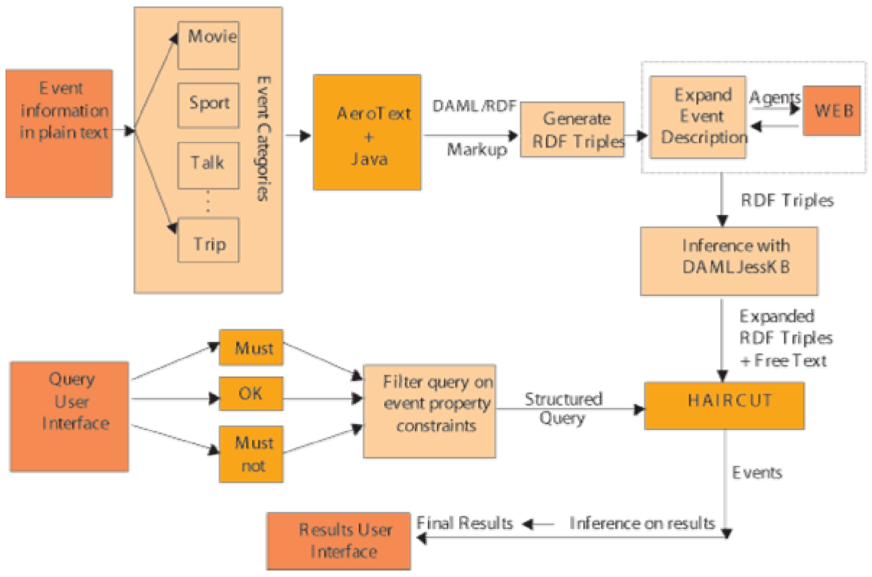


Figure 2. OWLIR dataflow diagram

OWLIR is only a framework, but a very important framework. Each component of this system can be replaced by another component with the same functionality.

OWLIR consists of the following parts:

Information extraction.

OWLIR is not only a semantic web retrieval system. Unfortunately, SW is only under construction now and the problem of markuping the documents in a data collection (to make them really SW's documents) can be solved either by human correcting (which requires much time and resources) or by using information extraction tools.

OWLIR uses AeroText system for text extraction of key phrases and elements from free text documents. Document structure analysis supports exploitation of tables, lists, and other elements and complex event extraction to provide more effective analysis. AeroText is used together with Event ontology and outputs the result into a corresponding RDF triple model that utilizes the DAML+OIL syntax. It is accomplished by referencing to the Event ontology that directly correlates to the linguistic knowledge base used in the extraction process.

Inference engine.

On the next step OWLIR uses the metadata information added during the text extraction process to infer additional semantic relations. The inference engine exploits two information sources for deriving an answer: Event Ontology and the facts in the knowledge base. DAMLJessKB facilitates reading DAML+OIL pages, interpreting the information as per the DAML+OIL language, and allowing the user to reason over that information.

DAMLJessKB is an inference system included into OWLIR. It provides basic facts and rules that facilitate drawing inferences on relationships such as Subclasses and Subproperties.

The combination of DAMLJessKB and domain specific rules has provided us with an effective inference engine.

Event Ontology.

The basic concept of the SW Ontology is described in part 1.2. OWLIR uses ontologies to annotate web pages with semantic information. The Event Ontology is built following the concept of \Natural Kinds OF". This ontology describes categories, relationship rules between categories or other data, which can be used later to describe relationships between pages and other data like numbers or dates. See figure 3.

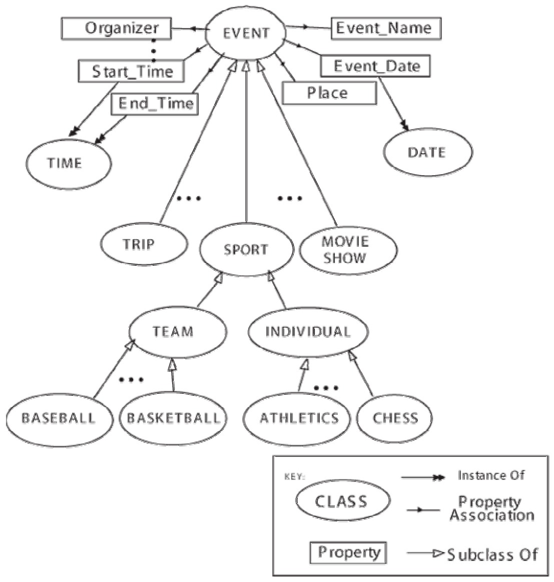


Figure 3. Event Ontology

A peculiar aspect of OWLIR is that the ontology not only aspects how germane knowledge is retrieved, but also influences system interaction, at the encoding and end-user levels. The ontology becomes a means of communication between the user and the system and helps overcome the
bottlenecks in information access, which is primarily based on keyword searches. It supports information retrieval based on the actual content of a page and helps navigate the information space based on semantic concepts. Ontologies enable advanced query answering and automatic information extraction.

Indexing and retrieval.

HAIRCUT is a IR system which is used for OWLIR search engine. HAIRCUT supports:

- reasoning document similarity
- Boolean or vector-space models
- variety of tokenization schemes including overlapping character n-grams
- novel term similarity measure

HAIRCUT allows the user to specify required, allowed and disallowed query terms. All these features allow to realize HIR's concepts in practice with markup similarity and indexing.

Query preprocessing.

There is a special query user interface in OWLIR which provides many data types for query term, including numeric attributes, geographic location, temporal values and other quantities whose semantics are difficult to capture with keyword search. Taking advantage of the HAIRCUT feature which allows the user to specify which terms in the query MUST, MUSTNOT and MAYBE considered, each query is expressed as a document consisting of triples and
free text.

Information Flow.

See Figure 2. Documents from basic collection together with ontology are processed by extraction tool. The output of this step is documents described in DAML+OIL. On the next step RDF triples are generated from DAML+OIL pages and additional rdf triples extracted from web, ontologies and current document collection using Inference engine. The IR engine preprocesses and indexes these documents. A query formulation mechanism translates natural language questions into queries for the IR engine in order to retrieve apposite documents from the collection, i.e., documents that can potentially answer the question.

1.3.3. Experimental results

The aim of the experiments is to verify that semantic markup within documents can be exploited to achieve better retrieval performance. Authors wanted to measure the extent to which Precision and Recall is improved with the use of semantic markup.

Experiments conditions:

- The baseline measurements are the P/R values obtained from the free text documents.
- Precision and Recall are measured for retrieval over three different types of document:
  - text only
  - text with semantic markup
  - text with semantic markup that has been augmented by inference.
- two types of inference were used to augment document markup:
  - reasoning over ontology instances (e.g., deriving the Date and Location of a Basketball Game)
  - reasoning over the ontology hierarchy (e.g., a Basketball Game Event is a subclass of Sport Event).

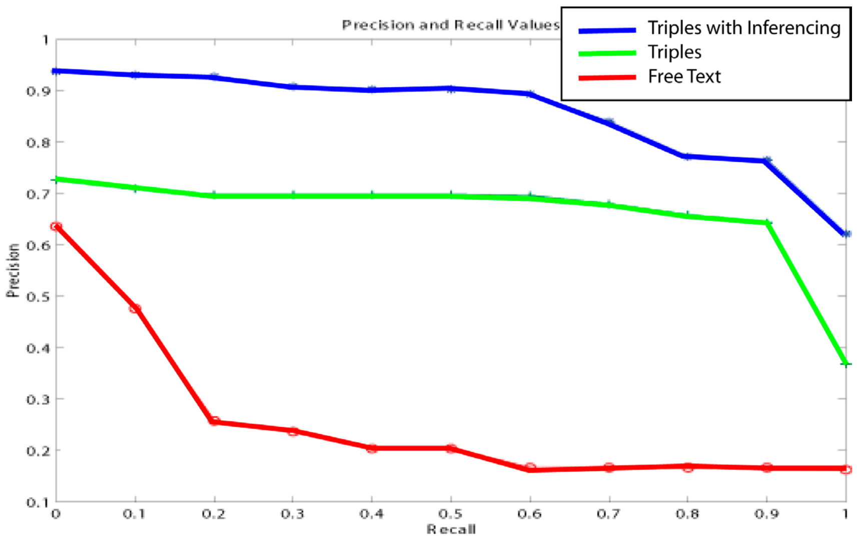


Figure 4. Precision and Recall Values.

You can see on the Figure 4 that looking for information in semantic data collection with inference gives the best combination of Precision/Recall values. This means that a greater number of relevant documents was retrieved and ranked higher than those with the text documents. This fact can be attribute to the inference capabilities and indexing of text and triples as a means of hybrid information retrieval.

1.4. Semantic Web discussions

1.4.1. Problems

Now Semantic Web is mostly a future concepts in spite of 4 years of intensive researches and developments. It is caused by many reasons:
Not all parts of the SW are developed yet. May be the most important parts like agent, trust and retrieval techniques are developed very poor that strongly restricted the real usability of the SW.

Technological base do not allow to use SW in good way. Technological base means the current level of hardware and software that can not support all SW features. (e.g. "We believe that performance in terms of speed is not as important in this case as performance in terms of what is retrieved." from OWLIR report) Software as well as hardware does not allow to easy develop a SW application and documents that brakes the mass SW's growing.

People have no resolution to change the Web and to spend a lot of money. After all IT crisis people are very careful with all new technologies which needs a lot of money. It is mostly a psychological problem, but may be it is the most important problem in the case of making the SW really a global scale system.

Many people do not believe in Semantic Web. Poor theoretical foundation and in the absence of good news and evaluation results make people and especially computer experts more and more pessimistic.

1.4.2. Solutions (It's not so bad, it's not so bad …)

The core of the SW are already developed (Description languages: RDF, DAML+OIL, ontology and inference, the basic concepts of agents and proofs are taken from AI)

There is a clear plan for future. It seems the group of developers has a clear plan how to make web semantic. It gives the hope that all other problem are only technical.

Industry makes computer faster and faster every minute. Thousand of unemployed but highly qualified programmers are ready to develop a new software agent for SW.

Wide using the XML created a good foundation for Semantic Web together with already developed software tools for creation SW application and documents (even poor) allow to reduce the cost of the SW building. That is a good reason to put up money right now.

Active position of Scientific Community made people believe in SW. The problem of the SW is it is not a technology but philosophy of the future web existing. You trust the technology, but you believe the philosophy. Technology gives the answer, philosophy teaches to find the answer.

1.5. References

1. Urvi Shah et al.: Information Retrieval on the Semantic Web. In: Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management, McLean, VA, USA, November 4-9, 2002
2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. Scientific American, May 2001.
3. Sean B. Palmer. The Semantic Web: An Introduction. 09-2001. http://infomesh.net/2001/swintro/
4. DAML+OIL: a Reason-able Web Ontology Language. Keynote talk given at EDBT 2002, Prague, March 26th, 2002.
5. James Mayfield, Tim Finin. Information retrieval on the Semantic Web: Integrating inference and retrieval. SIGIR 2003 Semantic Web Workshop, 1 August 2003, Toronto, Canada.

2. Building the Semantic Web

The Semantic Web is supposed to be an extension of the World-Wide Web. However, the current recommended Semantic Web languages, RDF and RDFS, are not built on top of the current World-Wide Web. In order to describe the semantics of the World-Wide Web, the Semantic Web must be based on XML, the data format of the World-Wide Web.
In the first section, we will give a very brief introduction to RDF; then, we will present an approach for building a new common semantic foundation for both the World Wide Web and the semantic Web, taking ideas from both. Finally, we will give a small example to summarize our discussion.

2.1. Introduction to RDF

The World Wide Web was originally built for human consumption, and although everything on it is *machine-readable*, this data is not *machine-understandable*. In order to make data on the Web be machine-understandable, we can use *metadata* to describe the data contained on the Web. Metadata is "data about data" (for example, a library catalog is metadata, since it describes publications) or specifically in the context of this specification "data describing Web resources".
Resource Description Framework (RDF) is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasizes facilities to enable automated processing of Web resources.

2.1.1. Basic RDF Model

The foundation of RDF is a model for representing named properties and property values.
The basic data model consists of three object types:

1. A **Resource** is anything that can have a URI; this includes all the Web's pages, as well as individual elements of an XML document.
2. A **Property** is a Resource that has a name and can be used as a property, for example Author or Title. In many cases, all we really care about is the name; but a Property needs to be a resource so that it can have its own properties.
3. A **Statement** consists of the combination of a *Resource*, a *Property*, and a *value*. These parts are known as the *'subject'*, *'predicate'* and *'object'* of a Statement.

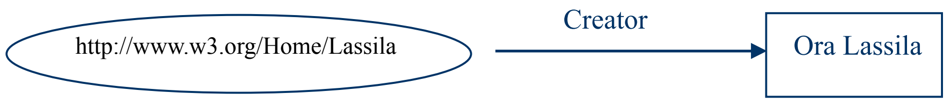
Resources are identified by a *resource identifier*. A resource identifier is a URI plus an optional anchor id.
Consider as a simple example the sentence:

Ora Lassila is the creator of the resource http://www.w3.org/Home/Lassila.

This sentence has the following parts:

Subject (Resource)	http://www.w3.org/Home/Lassila
Predicate (Property)	Creator
Object (literal)	"Ora Lassila"

This RDF can be illustrated using directed labeled graphs. The nodes (drawn as ovals) represent resources and arcs represent labeled properties. String literals will be drawn as rectangles.



The graph consists of two nodes connected by a single arc. One node (represented as an oval) contains the URI text http://www.w3.org/Home/Lassila and the other (represented as a rectangle) contains the text "Ora Lassila". The arc connecting these nodes bears the label Creator.

2.2. Building the Semantic Web

2.2.1 Requirement

The Semantic Web aims at machine-processible information. It will only be possible once further levels of interoperability have been established. Standards must be defined not only for the syntactic form of documents, but also for their semantic content. The Web is the first widely exploited many-to-many data-interchange medium, and it has new requirements for any exchange format:

Universal expressive power.

Because it is not possible to anticipate all potential uses, a Web-based exchange format must be able to express any form of data.

Syntactic interoperability.

Applications must be able to read the data and get a representation that can be exploited. e.g.: software components like query APIs, should be as reusable as possible among different applications.

Semantic interoperability.

One of the most important requirements for an exchange format is that data be understandable. Whereas syntactic interoperability is about parsing data, semantic interoperability is about defining mappings between terms within the data, which requires content analysis.

Using XML

XML fulfills the **universal expressive power** requirement because anything for which a grammar can be defined can be encoded in XML. It also fulfills the **syntactic interoperability requirement** because an XML parser can parse any XML data, and is usually a reusable component. When it comes to **semantic interoperability**, however, XML has disadvantages. XML's major limitation is that it just describes grammars. There is no way to recognize a semantic unit from a particular domain because XML aims at document structure and imposes no common interpretation of the data contained in the document.

Using RDF
RDF's nested *object-attribute-value* structure satisfies our **universal expressive power** requirement for an exchange format. Application-independent RDF parsers are also available, so RDF fulfills our **syntactic interoperability** requirement as well. When it comes to **semantic interoperability**, RDF has significant advantages over XML: The *object-attribute structure* provides natural semantic units because all objects are independent entities.

2.2.2. Challenge:

The current recommended Semantic Web languages RDF/S is not built on top of the current World-Wide Web. RDF does use the syntax of XML. However, it does not give this syntax the same meaning that it is given in XML. Thus there is a **semantic discontinuity** at the very bottom of the Semantic Web.
The reason is that RDF and XML have different modeling foundations. XML is based on a tree model where edges have no labels and outgoing edges from a node have a total order. This kind of model has its roots in semi-structured data and databases. RDF and RDFS are based on a directed graph model where edges do have labels, but are unordered. This kind of model has its roots in the **model theory** for standard logics. The data model underlying RDF is a directed graph. Nodes of the graph are labeled with identifiers, or literals (i.e., text), or are unlabeled. Edges of the graph are also labeled with identifiers. Some edges of the graph, those labeled with rdf:type, are have a built-in meaning. These edges link nodes to their types.

Our goal is to create a *common semantic foundation* for both the World-Wide-Web and the Semantic Web, taking ideas from both. No change to XML, only minor changes to RDF.

2.2.3. A new foundation

The difference between trees and directed graphs is easily reconciled. We use the XML data model directly, not allowing **labels on edges**. The result is that labeled edges in RDF graphs correspond to two unlabeled edges, with the label ending up on the node in the middle.
XML and RDF take different views of **node labels** as well. In XML some node labels are element names, which correspond to RDF types, not RDF labels. Other node labels are attribute names, which correspond to RDF edge labels. Yet other node labels are text or typed values, which correspond to RDF literals. RDF identifier labels have no corresponding label in the XML data model.

In our merge, nodes are of one of two kinds. One kind of node corresponds to text or value nodes from XML or literal nodes from RDF. These nodes are given **labels** that are their text or their typed value, as appropriate. The other kind of node corresponds to element or attribute nodes from XML or non-literal nodes or edge labels from RDF.These nodes are given two optional labels, one of which is the RDF identifier and the other of which contains the element or attribute names from XML or the rdf:type(s) or edge label from RDF. Two of these nodes in a graph cannot have the same RDF identifier. To handle the edge ordering of XML, our data models partially order the outgoing edges from each node. A final reconciliation that is needed is the difference in naming conventions between XML and RDF. We propose to go with

qualified names, abandoning direct use of URIs.
So we end up with a directed graph data model where edges have no label, text or value nodes have their text or value as their label, other nodes have two optional labels, both of which use Qnames, and there is a partial order on edges.

2.2.4. A New Semantic Web

So what will this new Semantic Web look like? Again, there would not be much difference from the current World-Wide Web, at least as far as XML and XML Schema are concerned. Documents would still be parsed and validated, much as they are now. The only change here would be that the meaning provided by XML and XML Schema would end up being the Semantic Web meaning of XML documents, instead of being supplanted by the RDF meaning. The situation is somewhat different from the RDF and RDFS side. RDF is reduced from the main language of the Semantic Web to an extension of XML that really only provides the identifiers (rdf:ID) that tie the XML data model nodes together to make a graph. RDFS is completely gone, being replaced by our as-yet-unspecified Semantic Web Ontology Language, SWOL.

SWOL would roughly fill the same role as currently filled by DAML+OIL, but would have a much closer relationship to XML and XML Schema. This SWOL document fragment contains the definition of a class, named Organization. Elements of this class, organizations, have a name, in the form of a string. Elements of the class also can have purchases, each of which must belong to the PurchaseOrderType.

```
<swol:class name="Organization" defined="no">
   <swol:exists>
      <swol:class name="name"/>
      <swol:class name="xsd:String"/>
   </swol:exists>
   <swol:all>
      <swol:class name="purchase"/>
      <swol:class name="PurchaseOrderType"/>
   </swol:all>
</swol:class>
```

For now, we can think of a SWOL document as a collection of several definitions of this sort, each with a name. There are actually many possibilities for SWOL, just as there are many description logics, varying in expressive power from frame-like formalisms up to very powerful ontology-definition formalisms.

2.2.5. Semantic Foundations for the New Semantic Web

The machinery we have introduced so far is not sufficient to provide a semantic foundation for this new Semantic Web. Data models, only work well for providing meaning for simple, inexpressive languages. The meaning of documents that contain disjunctive or vague information, such as saying that *either John or Jim is the author of a web page*,

where the exact state of affairs of the world is not known, cannot be captured in a single data model. The usual solution to this problem is to have a document correspond to any one of several interpretations, each of which corresponds to one of the ways the vague information can be resolved. So the above disjunction would give rise to two classes of interpretations, one class where John is the author of the web page and one where Jim is.

Model Theory:

This way of giving meaning to documents (usually referred to as collections of statements) is called model theory. It's a formal semantic theory that relates expressions to interpretations.

The name 'model theory' arises from the usage, traditional in logical semantics, in which a satisfying interpretation is called a "model".

The meaning of a document is different in Data Model and in model theory. In data models, the meaning of a document is a single a single data model, which corresponds to the portion of the world being considered. In model theory the meaning of a document is a collection of interpretations. Each of these interpretations corresponds to one of possible ways the world could be, given the information under consideration.

2.2.6. Interpretation

Interpretations are the essential component of our model theory. An interpretation corresponds to one possible way the world can be, hence encoding a certain meaning for the information manipulated by an application. Interpretations give information about resources and related resources through relationships and semantic constraints. We define a notion of interpretation that is suitable for both XML and RDF documents, through the XQuery data model.

An *interpretation* is a six-tuple, $\langle R,E,EXT,CEXT,O,S \rangle$, where:

R is a set of resources,

E is a set of relationships,

$EXT : E \rightarrow R \times (R \cup DV)$ *maps relationships to the resources they relate,*

$CEXT : U \rightarrow P(R \cup DV)$ maps classes to their extensions,

$O : R \rightarrow P(E \times E)$ *provides a local order on the relationships, and*

$S : U \rightarrow R$ is a partial map from *QNames* to resources.

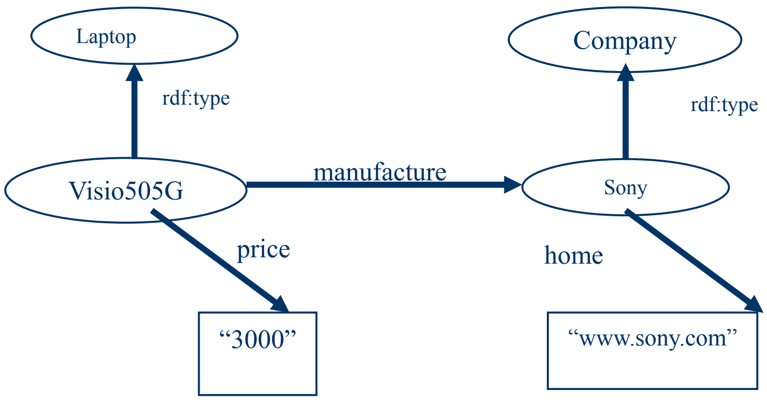
U the value space of QNames, i.e., pairs of URIs and local parts.

DV the union of the value spaces of the XML Schema primitive datatypes.
In RDF elements of DV are generally called *literals*.

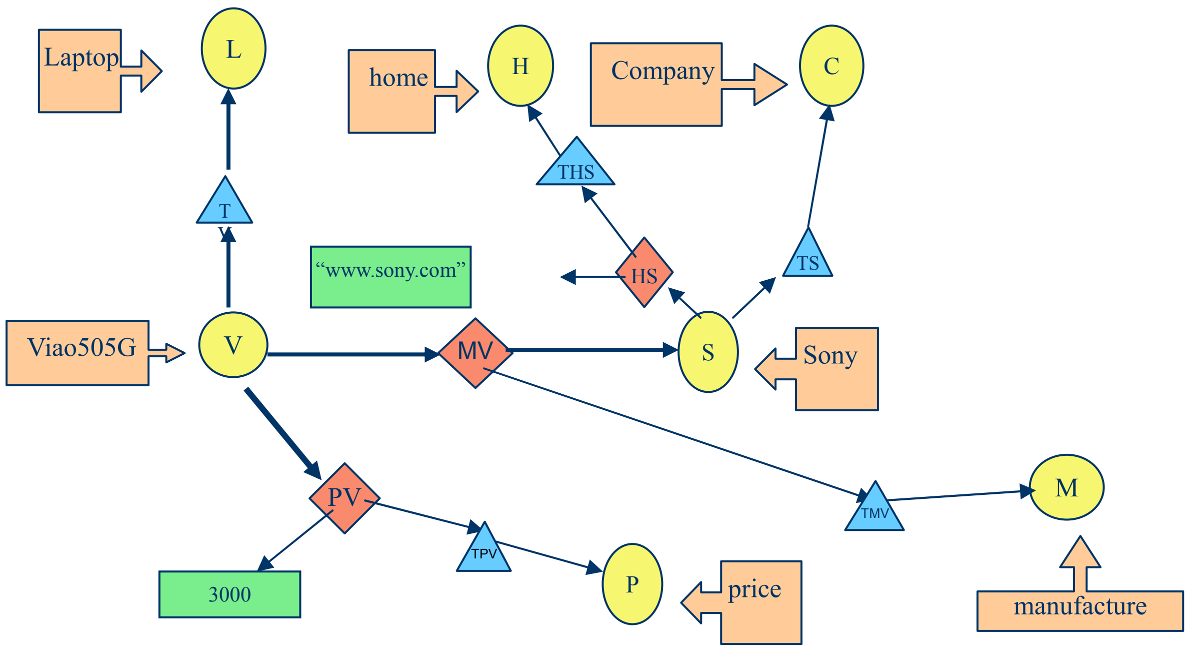
EXT maps relationships to the resources they relate, For instance, a relationship $e1 \in E$ with

$EXT(e1)$ = *John, p* indicates that the resource *John* is related to the resource *p*. *S* provides a mapping between syntax (QNames) and their denotation (resources). *S* gives a means to identify these entities using QNames. *CEXT* provides typing information for resources. For instance, if Sony is in CEXT(Company) then the resource *Sony* is of type Company. Finally, *O* provides ordering information between the relationships that are related to a common resource.

Interpretation as graph



We can present part of an interpretation of the example showed in the above RDF graph. Most nodes are **resources**, i.e., elements of R; the node with “3000” next to it is the integer 3000 and the node with “www.sony.com” next to it is a string, both **data values**. The **mapping S** is given by the arrows from QNames to the nodes. **Relationships** in the interpretation (E and EXT) are shown as links between the resources. Finally **CEXT** provides typing information for resources.



Get Information of Documents

Now that we have defined our notion of interpretation, we need to explain how documents correspond to sets of interpretations. Intuitively, each node in the data model graph resulting from document is mapped to a resource in the interpretation, and *EXT* relationships are built according to the edges of the data model graph.

Documents no longer correspond to single data models, but instead correspond to a set of interpretations, the models of the document. Thus it is no longer possible to just get the information contained in a document by performing operations on the data model. Instead, we

have to find out this information indirectly. The usual method for finding out this information in model theory is via an entailment relationship. Entailment captures what information is implicit in a document.
Definition of Entailment:

Given 2 document, D and E. D ⊨ E if all models of D are also models of E.

2.3. Example

A simple example that shows how to describe collections of XML documents in differing formats. We assume the existence of a collection of different *purchaseOrders* and *PurchaseOrderTypes* each defined in a different XML Schema document, with different URLs. We will assume that each of these documents have a namespace, *pos-i*. Without loss of generality, we will assume that the different XML Schema documents use the same internal name for their top-level components.
We can use SWOL to define the *Organization* class, containing resources that have *purchases* that belong to the *PurchaseOrderType*.

```
<swol:class name="Organization" defined="no">
  <swol:all>
                <swol:class name="purchase">
                <swol:class name="PurchaseOrderType">
          </swol:all>
          ...
</swol:class>
```

This *PurchaseOrderType* is then defined as a **generalization** of the various *PurchaseOrderTypes*.

```
<swol:class name="pos-i:PurchaseOrderType" defined="no">
          <swol:class name="PurchaseOrderType"/>
</swol:class>
```

We can then create a document that ties together various purchase orders, each in its own document with its own name, here given as *pos-i*.

```
<Organization rdf:ID="foo">
    <purchase rdf:ID="po-1:">
    <purchase rdf:ID="po-2:">
    ...
</Organization>
```

However, all we have so far is a collection of purchase orders with no combined way of accessing the information in them, because they each have different elements names (because of their differing namespaces). To unify these elements, we have to provide a generalization of the different element names,

```
<swol:class name="pos-i:shipTo” defined="no">
    <swol:class name="shipTo" />
</swol:class>
<swol:class name="pos-i:items" defined="no">
```

```
    <swol:class name="items" />
</swol:class>
...
```
Using this, and other, facilities from SWOL, we can take information from disparate XML documents, using disparate XML Schema types, and access it in a uniform manner, resulting in a Semantic Web version of the World-Wide Web.

2.4. Conclusion

So what have we achieved? We have created a semantic foundation for the Semantic Web that unifies it with the semantic foundation for the World-Wide Web. This semantic foundation takes the semantic foundation of XML, node-labelled ordered trees, and adds in semantic notions from RDF, including node identifiers and graphs. We then moved from data models, where a document corresponds to a single data structure, to model theory, where a document singles out a collection of interpretations, so as to allow for disjuntive or vague information, as needed in ontologies. In the process we eliminated some of the semantic notions from RDF, like edge labels, to achieve a better relationship between the XML expression of RDF and XML itself.

The **result** of the new semantic foundation is a new vision of the Semantic Web as a natural extension of the World-Wide Web. In this new Semantic Web, XML is no longer just "the universal format for structured documents and data on the Web", but instead is the major source of semantic information for the Semantic Web. XML Schema documents still play their current role of constraining and typing XML documents, but, since XML plays a larger role in this vision of the Semantic Web, even this use of XML Schema has more utility. XML Schema documents can also be used as global definitions of types, a new role for XML Schema. Ontology information, that cannot be represented by XML Schema, is carried by a new ontology language SWOL.

3. Semantic Mapping on Ontologies

3.1. Introduction

Content on the web today (over 1.5 billion pages) is designed for human consumption, and not conducive for automatic information processing by software agents. Semantic Web is set out to change this situation. Content in the web of tomorrow will be marked up with the help of ontologies. This will make information processing easier for software agents, opening the door for a slew of new web-based applications. Personal Software agents will browse the entire breadth of the web keeping track information relevant to users. We shall be able to go beyond the keyword searches. Queries can be answered by gathering information from multiple web-pages, something that is not possible on the Google of today.

So what is ontology?

3.1.1. Ontology

The best characterizes the essence of ontology is based on the related definitions by Gruber (1993): An **ontology** is a formal, explicit specification of a shared conceptualization. A "conceptualization" refers to an abstract model of some phenomenon in the world that identifies the relevant concepts of that phenomenon. "Explicit" means that the type of concepts used and the constraints on their use are explicitly defined. "Formal" refers to the fact that the ontology should be machine understandable. "Shared" reflects the notion that an ontology captures consensual knowledge, that is, it is not restricted to some individual but accepted by a group. Large ontology like WordNet provide a thesaurus for over 100,000 terms explained in natural language.

Ontologies allow users to organize information into taxonomies of concepts. Each with their attributes, and describe relationships between concepts. More recently, the notion of ontology has also become widespread in fields such as intelligent information integration, cooperative information systems, information retrieval, electronic commerce and knowledge management.

3.1.2. Why ontology mapping

Here is the simply reason why ontology becomes crucial in future web technology. Ontology promises: a shared and common understanding of some domain that can be communicated among people and application systems

But there definitely will be multiple ontologies within a same domain. This is inevitable considering the distributed and autonomous nature of development of web content. Semantic Mappings are essential to realize the full potential of the Semantic Web vision and allow information processing across ontologies

Consider one such scenario.

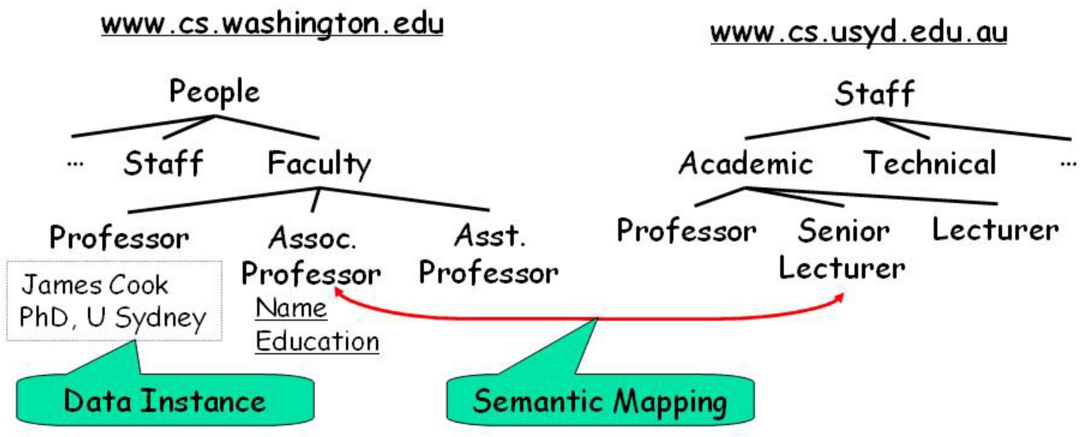


Figure 1

Consider two computer science department web-pages – university of Washington and the university of Sydney. Suppose the two departments decide to semantically enable their web-content. The do so by marking up their web-pages with ontology. Here is how a small subset of their ontology might look like - one that marks up content related to people in the department

An ontology has concepts that identify the data entities of interest. These concepts are organized in a hierarchy like the ones above. Such a hierarchy is called a **taxonomy**. Concepts might have **attributes**, and also **relationships**. A data item that has been marked up with a label corresponding to a concept is called a **data instance**.

For example, there might be one Prof. James Cook like figure 1. We can now answer a query like "Find Prof. Cook, a professor in a Seattle college, earlier an assoc. professor at his alma mater in Australia" – We know that James Cook has a PhD from Australia, and we also know that he was a Senior Lecturer there. We will know that he does satisfy our requirement if we knew this critical piece of information Associate Profs and Senior Lecturers are equivalent concepts in the US and Australia. Such a correspondence is called a **Semantic Mapping**.

There has been research in the SW community on languages for ontologies – RDF and description logics such as DAML+OIL. And there has been some work on design tools for domain ontologies and on ontology learning like Protégé, Ontolingua. However there has been only some work that relates to automatically generating semantic mappings. As showed in above example, such mapping is essential in the presence of multiple ontologies. With each web-page speaking its own language, automated reasoning will not be possible. Without semantic mappings, the semantic web will be nothing more than an electronic version of the Tower of Babel.

The report is based on the paper "Learning to Map between Ontologies on Semantic Web" (VLDB 2003) and describes a system which makes ontologies mapping be possible,

3.2. Semantic Mapping Challenges

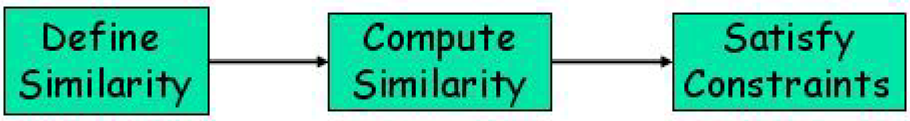
Ontologies may use different vocabularies, different design methodologies, might overlap but not coincide. While doing in manually might be possible for small examples, it becomes in feasible and time consuming on the Web-scale – large number of ontologies, and large-sized ontologies.

But there are sources of information that can be leveraged to learn semantic mappings. For example, there are data instances that are available from the web content and will be marked up with ontologies, that can be used to learn mappings. There might be constraints and heuristics on the mapping – ones that are available from the structure and relationships in the ontologies and from our prior knowledge of the domain of the ontologies. There are many heuristics that might be able to use different types of such information and we need a single framework that will put them all together.

3.2.1. Semantic mapping overview

There are 4 contributions to semantic mapping described in the paper:

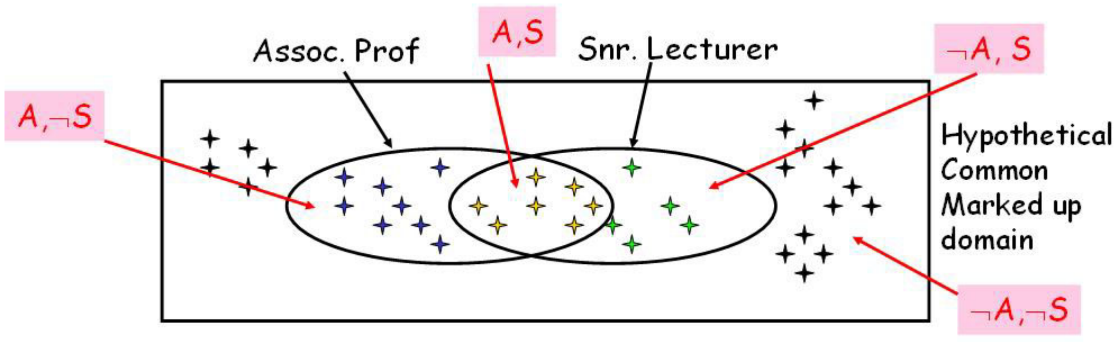
- Proposed an automatic solution, system GLUE, to matching taxonomies.
- The system works with multiple notions of similarity, and uses machine learning to incorporate varied information from data instances and taxonomic structure.
- Extend a constraint satisfaction technique, Relaxation Labeling, to the ontology matching context, and use it to incorporate diverse generic and domain-specific constraints on the mappings.
- Experimental results from matching real-world taxonomies on the web, and demonstrate the accuracy and utility of our system. (High accuracy (68-98%) on large taxonomies (100-330 concepts)



Here is a high-level overview of semantic mapping. Two taxonomies are given and it is desired to obtain mappings between them. In particular for each concept in the left, we want to map it to the most similar concept on the right. Here most similar is something that is application specific, and depends on the purpose of our mappings.

The first step in a mapping process is to define a meaning for similarity. The next step is to compute the similarity with concepts in the other taxonomy as defined by our measure. The best mapping might be chosen in a straightforward way by picking the one with the highest similarity value. However before confirming the mapping, we check if the mapping satisfies different constraints that might be relevant to our context – possibly altering and fixing the mapping

3.2.2. Defining Similarity



In above figure, it shows a hypothetical domain of instances that has been marked up with both our ontologies of interest. We can extract data instances for each concept in both the ontologies. Consider the Venn diagram representation for the set of instances of Associate Profs and the set of Senior Lecturers. The area in between contains instances that are both Assoc. Profs and Senior Lecturers.

One measure of Similarity could be the ratio of these instances in the intersection to those that are either Profs or Lectures

$$\textit{Jaccard- sim}(A, B) = P(A \cap B) / P(A \cup B)$$
$$= \frac{P(A,B)}{P(A,B) + P(A,\overline{B}) + P(\overline{A},B)}$$

This is equivalent to ratio of the probability of the intersection set to the union.
This measure has some desirable properties – 0 when disjoint, 1 when exact.
Note that this expression can be broken down into the following expression.
Here:
P(A,B) = Probability of a random instance being both an Associate Prof. And a Senior Lecturer. (the intersection area)
P(A, not B) = Probability of a random instance being a Assoc. Prof. but not a Senior Lecturers.
Given two concepts, the domain can be divided into these 4 regions

P(A, B)
P(A, not B)
P(not A, B)
P(not A, not B)

The probabilities corresponding to these 4 regions are called the **Joint Probability Distribution (JPD)**. Note that these measures are not related in any way to the syntax of the data instances. Instances can be web-pages, plain text, multi-media clips, etc.

Note there are some other candidates for similarity measure:

Dice-coefficient	(2*P(A,B)/(P(A)+P(B)))
Overlap-coefficient	(P(A,B)/min{P(A), P(B)})

So the system can change to any similarity formula as user wish. This reflects the robust of the system.

Now that similarity measure is defined, the next step is to compute it.

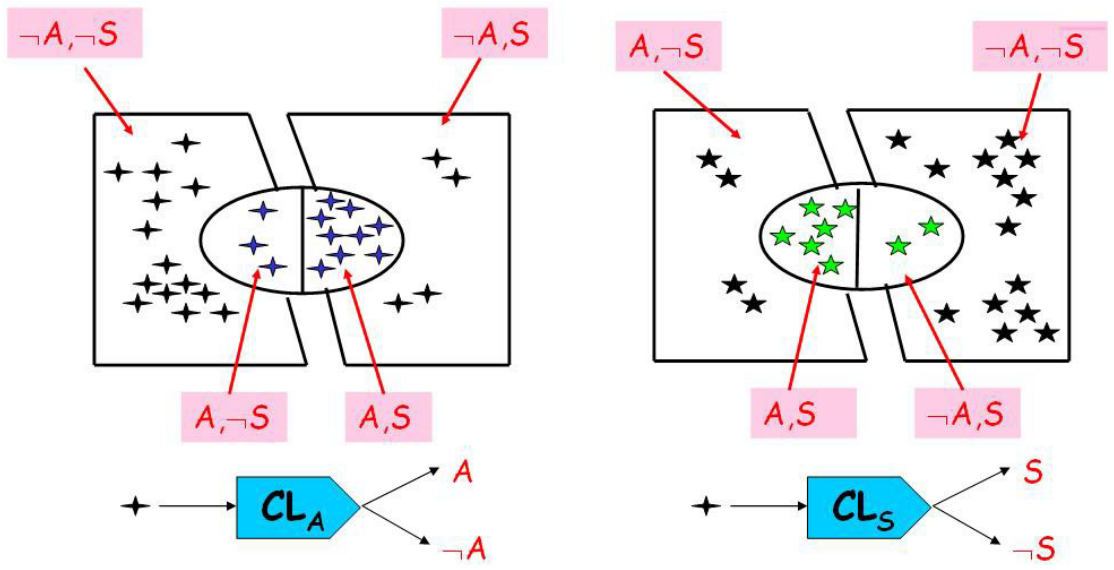
3.2.3. Compute similarity

The above definition of similarity measure assumed the existence of a common set of instances marked up with both the taxonomies. But this is most often not the case. What we have is data instances for concepts in the United States computer science department taxonomy, and data instances in CS department in Sydney.

Given the two concepts Assoc. Prof. and Senior Lecturer, all that we can do is partition the instances on the left into two sets – A and not A, similarly on the right into S and not S. But our goal is to compute similarity as defined earlier that relies of dividing the data instances into 4 partitions. So Machine Learning comes into play. Machine Learning algorithms can generalize from data instances and learn identifiers for concepts.

(see the following figure) A classifier takes as input a data instance and returns a prediction of whether the instance belongs to the concept learnt by the classifier.

A classifier is learnt by giving positive and negative examples of a concept of interest. In this case the positive examples are data instances in A and others are negative examples. The learnt classifier will take as input any instances, even unseen ones, and predict if it is an A or not A. Similarly a classifier for S can be trained (learned).



We can now apply A's classifier on the data instances on the right. This will let us separate them into 4 partitions. The A,S partition has instances that are known to be in S, and that our classifier identified as being possible examples of A. Analogously we can partition the examples on the left using the other classifier. The probabilities in the JPD can be estimated by counting the sizes of the partitions and taking an average from the left and right.

We have talked using classifier to decide which instance belongs to which concept in different domain. But what is exactly the "classifier"? And we shall see later using a single classifier doesn't do the job, because single classifiers cannot exploit all types of information.

In the paper, a so called "multi-strategy learning" is used – the basic idea being to combine the predictions of different learning algorithms. Instead of learning a single classifier for A, many classifiers are trained (learned). These classifiers use different information available in data instances (e.g. the text descriptions, or the names of the concepts), or might be learnt by a different process (decision tree, Bayes classifiers).

For each instance in the domain of S, we apply each of the classifiers and combine the results, to make a single prediction. The learner which combines the results is called a meta-learner.

Two base learners are introduced in the paper.

- Content learner uses frequency of words occurring in the data instances. An instance typically has a name and a set of attributes together with their values. So the name and attribute values are treated as data instance (textual content).

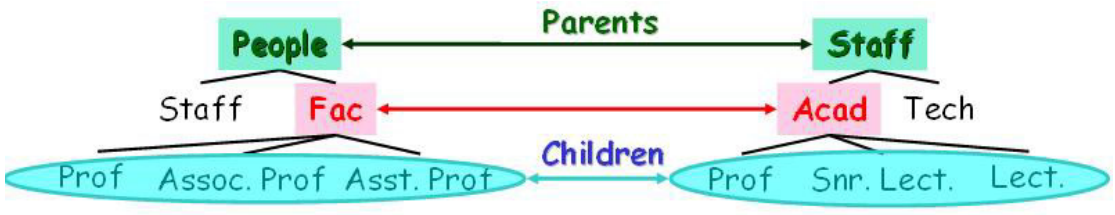
- Name learner uses the words that occur in the descriptive names of concepts. The full name of an instance is the concatenation of concept names leading from the root of taxonomy to that instance.

Finally, meta learner assigns to each base learner a learner weight that indicates how much it trusts that learner's predictions. Then it computes the base learners' prediction via a weighted sum.

Now the similarity matrix between the concepts in the two taxonomies can be computed as per the definition.

3.2.4. Exploit Constraints

It is often the case that we have some prior knowledge about properties of the mappings we are trying to obtain. Some of this knowledge can be formulated into constraints that can be used to guide our search for the correct mappings.
Consider the following examples of constraints derived from the structure of the taxonomies.



Concepts match if their parents match, or if a large fraction of their children match. These constraints are in a sense generic as they hold across all mappings. There may also be domain-specific constraints, for example, there exists a unique department chair. In addition there may be heuristics that are better captured as constraints rather than machine learning algorithms.

Further these constraints can be numerous and of many different types.
So a single framework in which we can incorporate all such available information is required. And the framework should also be tractable.

Relaxation labeling is adapted, a well-known local optimization technique, to obtain such a framework.

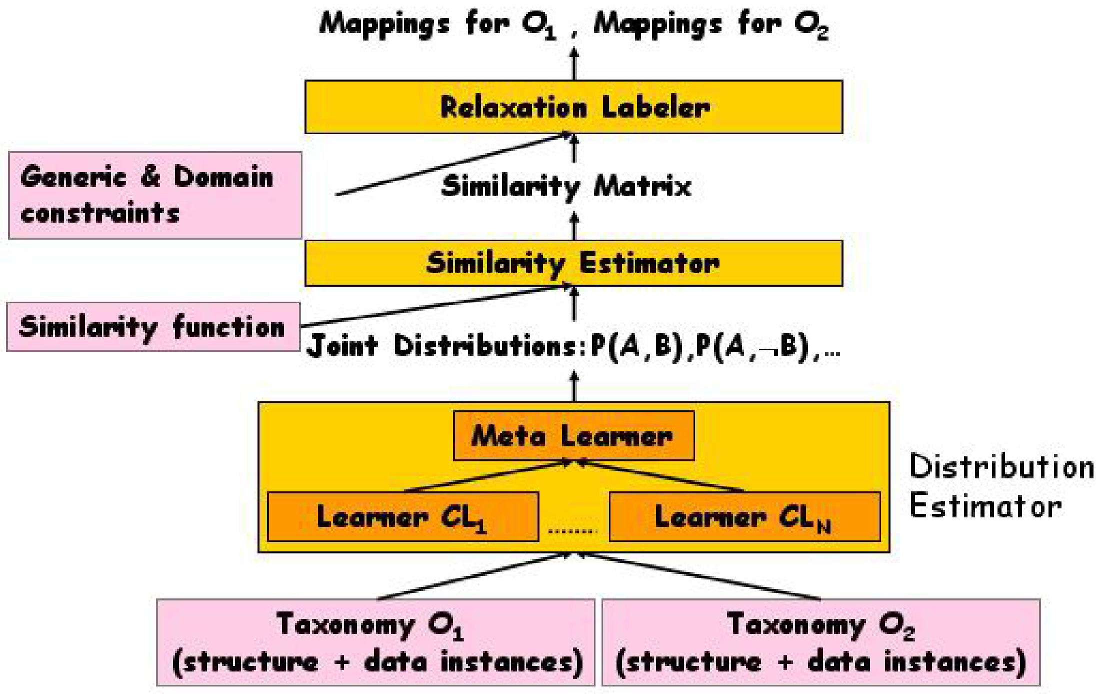
3.2.5. Relaxation labeling

Relaxation labeling is an efficient technique to solve the problem of assigning labels to nodes of a graph, given a set of constraints. The key idea is that the label of a node is typically influenced by the features of the node's neighborhood in the graph. Examples of such features are the labels of the neighborhood nodes, the percentage of nodes in the neighborhood that satisfy a certain criterion, and the fact that a certain constraint is satisfied or not.

Relaxation labeling assigns initial labels to nodes based solely on intrinsic properties of the nodes. Then it performs iterative local optimization. It uses a probability formula to change the label of a node based on the features of its neighborhood. The algorithm stops when all labels of nodes are fixed.

3.4. System Architecture

After examing all components of the system, It is time to put them altogether. It starts out with our two taxonomies. Estimate the joint probability distribution between pairs of concepts. Using Multi-strategy learner on the data instances. The Similarity matrix is computed given some definition of similarity. Relaxation labeling is used to incorporate the effects of multiple generic and domain specific constraints. The result is a pair of mappings. This is whole GLUE system described in the paper.


Mappings for O_1 , Mappings for O_2
Relaxation Labeler
Generic & Domain constraints
Similarity Matrix
Similarity Estimator
Similarity function
Joint Distributions: $P(A,B), P(A,\neg B), \ldots$
Meta Learner
Learner CL_1
....
Learner CL_N
Distribution Estimator
Taxonomy O_1 (structure + data instances)
Taxonomy O_2 (structure + data instances)

3.5. Experiment

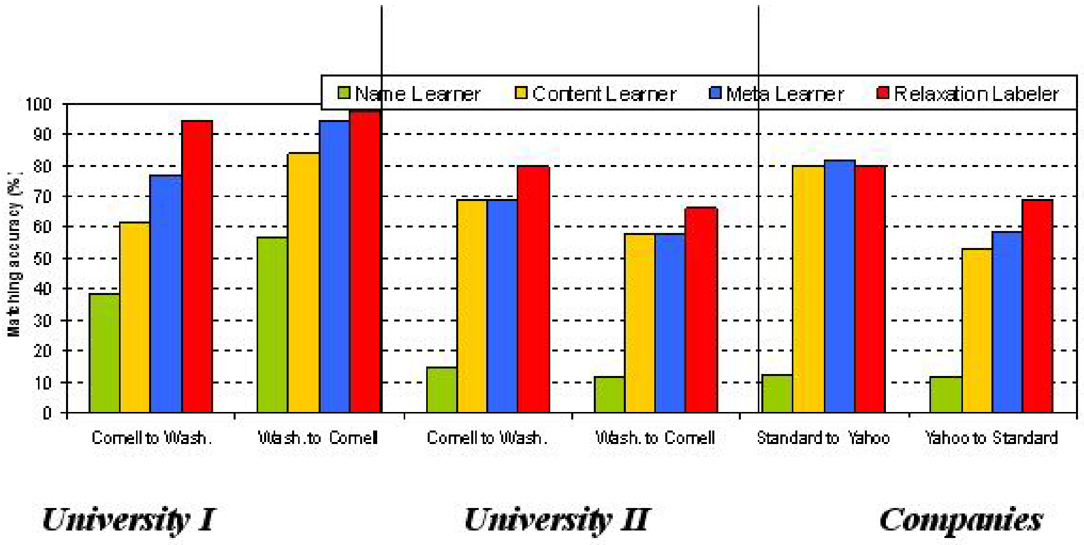
The experiments show promising results: system was tested on real-world taxonomies. The taxonomies include two domains.

The first one relates to classes in universities – from the univ of washington and cornell university. The classes were organized by schools within the university, departments, schools and possibly sub-fields within them.

In the second one, companies were organized by industry and sectors. The two taxonomies were those available at Yahoo and TheStandard

Each taxonomy and related data instances were obtained by crawling the web-sites. The data instances were class descriptions or company web-pages.
Some trivial data cleaning is done – eliminating instances with less than 5 words.

The following figure shows the experiments result with different learner applied. As you can see, meta learner gains finer result comparing with base learner and sometimes even better with relaxation labeller.


Name Learner
Content Learner
Meta Learner
Relaxation Labeler
Matching accuracy (%)
Cornell to Wash.
Wash. to Cornell
Cornell to Wash.
Wash. to Cornell
Standard to Yahoo
Yahoo to Standard
University I
University II
Companies