Ontology Discovery for the Semantic Web Using Hierarchical Clustering

Patrick Clerkin, Pádraig Cunningham, Conor Hayes

Department of Computer Science
Trinity College Dublin
Patrick.Clerkin@cs.tcd.ie
Padraig.Cunningham@cs.tcd.ie
Conor.Hayes@cs.tcd.ie

Abstract. According to a proposal by Tim Berners-Lee, the World Wide Web should be extended to make a Semantic Web where human understandable content is structured in such a way as to make it machine processable. Central to this conception is the establishment of shared ontologies, which specify the fundamental objects and relations important to particular online communities. Normally, such ontologies are hand crafted by domain experts. In this paper we propose that certain techniques employed in data mining tasks can be adopted to automatically discover and generate ontologies. In particular, we focus on the conceptual clustering algorithm, COBWEB, and show that it can be used to generate class hierarchies expressible in RDF Schema. We consider applications of this approach to online communities where recommendation of assets on the basis of user behaviour is the goal, illustrating our arguments with reference to the Smart Radio online song recommendation application.

1 Introduction

Tim Berners-Lee has proposed and extension to the existing World Wide Web known as the Semantic Web (Berners-Lee, 1998, Berners-Lee et al, 2001). Most of the Web's existing content is designed to be read and understood by humans, and cannot readily be parsed and processed by software agents. The central idea behind the Semantic Web is to develop and use machine-understandable languages for the expression of the semantic content of Web pages. This promises to enhance the ability of software agents to navigate the Web's information space and carry out tasks for humans, without the need for sophisticated artificial intelligence.

Central to the Semantic Web project is the concept of an ontology. Web pages are conceived of as being composed of statements relating objects. The denotations of the terms making up the statements need to be fixed relative to a particular universe of discourse, which is represented in an ontology. The ontology codifies a shared and common understanding of some domain. An ontology is usually constructed by domain experts. In this paper, we examine the possibility of generating ontologies automatically using hierarchical conceptual clustering, and consider certain online

communities where such methods are highly appropriate, since there is no existing conceptualisation of the site resources. It is important to emphasis at this point that we are concerned with generating ontologies from behavioural and usage data relating to resources of interest, rather than from the free text data that might be found on web pages.

Since we aim to demonstrate how this technique may be practically implemented, we first present an overview of some of the technologies being used to build the Semantic Web, focusing in particular on how basic ontologies can be represented using RDF Schema (Brickley and Guha, 2000). In subsequent sections we discuss the concept formation system, COBWEB (Fisher, 1987), and demonstrate how the concept hierarchies discovered by this algorithm can be represented as ontologies with RDF Schema. We conclude with a discussion of the application of this approach to online communities - dealing in particular with the Smart Radio system (Hayes and Cunningham, 2000), developed as a test bed for our ideas - and point to some further research directions.

2 Implementing the Semantic Web

The Uniform Resource Identifier (URI)[1] provides the foundation for the Web, since it allows us to give any object or concept a uniquely identifying name. URIs are decentralized, in the sense that no one person or organisation controls their defintion and use. Since anyone can create a URI, we inevitably end up with multiple URIs representing the same thing, so it is important for the Semantic Web to provide a means for resolving names correctly.

This is provided for in the use of an ontology, which usually takes the form of a taxonomy defining classes and relations among them. The meaning of terms can now be resolved if they point to particular ontologies, and if equivalence relationships are defined between ontologies.

The Resource Description Framework (RDF)[2] provides a means for software agents to exchange information on the Web. It defines a simple model for describing relationships between Web resources in terms of properties and their values. However, RDF itself provides no means of declaring such properties. This task is left to RDF Schema, which can be used to represent simple ontologies.

RDF can be written using XML tags, but it is important to note that XML is not sufficient for building the Semantic Web. XML facilitates the arbitrary creation of tags that can be used to annotate Web pages. If a programmer knows in advance what these tags signify, then it is possible for her to write software to process these Web pages automatically. However, in the absence of such knowledge, it is not possible to write such programs, since XML builds no semantics into its structures. RDF, on the other hand, encodes machine-processable structures into its statements. An RDF

[1] See http://www.w3.org/Addressing/ for an overview of naming and addressing schemes used on the World Wide Web.

[2] See http://www.w3.org/RDF/.

statement consists of a triplet, which asserts that a particular thing has a certain property with a certain value. For example, the sentence

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

can be represented in RDF by

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://description.org/schema/">
  <rdf:Description
about="http://www.w3.org/Home/Lassila">
    <s:Creator>Ora Lassila</s:Creator>
  </rdf:Description>
</rdf:RDF>
```

The RDF Schema defines a collection of RDF resources that can be used to describe properties of other RDF resources that define application-specific RDF vocabularies. The RDF Schema type system is similar to the type systems of object-oriented programming languages such as Java. For the purposes of this paper, it is sufficient to present this system in the context of an example. Consider the following class hierarchy. We first define a class MotorVehicle. We then define three subclasses of MotorVehicle, namely PassengerVehicle, Truck and Van. We then define a class Minivan which is a subclass of both Van and PassengerVehicle. In representing this hierarchy we must make use of some core classes and properties defined by RDF Schema. In particular: all things being described by RDF expressions are called *resources*, and are considered to be instances of the class rdfs:Resource; when a resource has an rdf:type property whose value is some specific class, we say that the resource is an *instance of* the specified class; when a schema defines a new class, the resource representing that class must have an rdf:type property whose value is the resource rdfs:Class; the rdfs:subClassOf property specifies a subset/superset relation between classes. Our example class hierarchy is therefore represented by the following diagram:

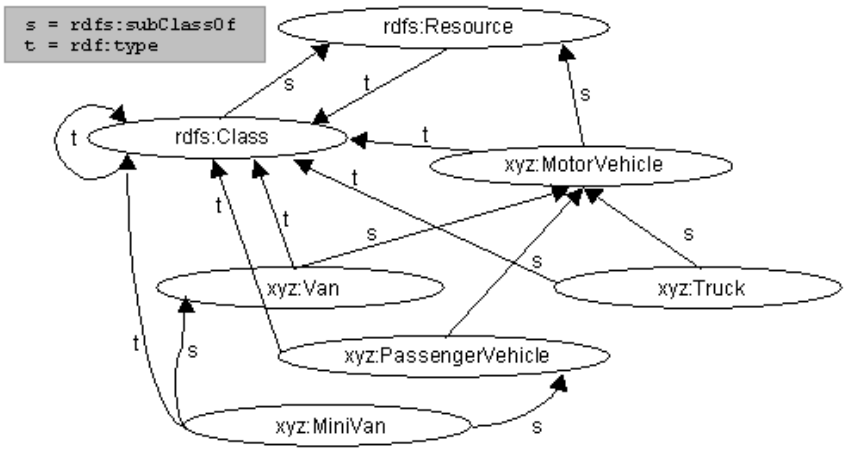

Fig. 1 Class Hierarchy for MotorVehicle class and its subsets (Brickley and Guha, 2000)

This model can be rendered in XML as follows:

```
<rdf:RDF xml:lang="en"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<rdf:Description ID="MotorVehicle">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf
  rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Resource"/>
</rdf:Description>

<rdf:Description ID="PassengerVehicle">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description ID="Truck">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

```
<rdf:Description ID="Van">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description ID="MiniVan">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#Van"/>
 <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>

</rdf:RDF>
```

3. Hierarchical Conceptual Clustering

Data mining is the extraction of implicit, previously unknown, and potentially useful information from data. The objective is to build computer programs that automatically detect regularities or patterns in databases. Useful patterns, if found, should generalise to make accurate predictions on future data. Thus, the final objective of data mining activity is knowledge discovery.

Machine learning provides the technical basis of data mining. It is used to extract information from the raw data in databases. The process is one of abstraction in order to find patterns. Usually, we also require that the system should provide us with an explicit structural description, so as to provide the observer with an explanation of what has been learned and an explanation of the basis for new predictions.

Clustering is a data-mining task that has at its goal the unsupervised classification of a set of objects. Classification is unsupervised in the sense that there are no *a priori* target classes used during training. Clustering techniques rely on the existence of some suitable similarity metric for objects. Clustering algorithms may be classified according to a number of criteria. Some are distance-based and describe clusters purely by enumerating their members, while others represent the clusters by means of a description. This description may take the form of a set of necessary and sufficient conditions for membership of a given cluster, or it may be a probabilistic description where no such set of conditions is tenable. Furthermore, a set of clusters may be "flat" in the sense that no cluster is "contained" in any other cluster, or it may be hierarchical, providing a taxonomy of clusters with definite relationships between them.

COBWEB is an incremental conceptual clustering algorithm with represents concepts probabilistically. It was initially inspired by research on *basic level* effects. For example, humans can typically verify that an item is a bird more quickly than they can verify the same item is an animal, vertebrate, or robin. Thus, the concept of birds is said to reside at the basic level. COBWEB's design assumes that principles which dictate basic concepts in humans are good heuristics for machine concept formation as well.

COBWEB is designed to produce a hierarchical classification scheme. It carries out a hill-climbing search - which consists of taking the current state of the search, expanding it, evaluating the children, selecting the best child for further expansion, etc, and halting when no child is better than its parent – through a space of schemes, and this search is guided by an heuristic measure called *category utility*.
The category utility metric was originally developed by Gluck and Corter (1985) to predict the basic level in human classification categories. In adopting it as a criterion for evaluating concept quality in AI systems, Fisher notes that it can be viewed as a function that rewards traditional virtues held in clustering generally – similarity of objects within the same class, and dissimilarity of objects in different classes.

$$CU(\{C_1, C_2, ..., C_n\}) = \frac{\sum_k \left((P(C_k) \left[P(A_i = V_{ij} \mid C_k)^2 - \sum_i \sum_j P(A_i = V_j)^2 \right] \right)}{n} \quad \textbf{(1)}$$

COBWEB performs its hill-climbing search of the space of possible taxonomies and uses category utility to evaluate and select possible categorisations. It initialises the taxonomy to a single category whose features are those of the first instance. For each subsequent instance, the algorithm begins with the root category and moves through the tree. At each level it uses CU to evaluate the taxonomies resulting from:

1. Classifying the object with respect to an existing class.
2. Creating a new class.
3. Merging: combining two classes into a single class.
4. Splitting: dividing a class into several classes.

4. Ontology generation using COBWEB

We propose that COBWEB may be used to automatically generate ontologies. Let us consider the following artificial and simple example. We have a domain consisting of only four resources, namely, the following four cells:

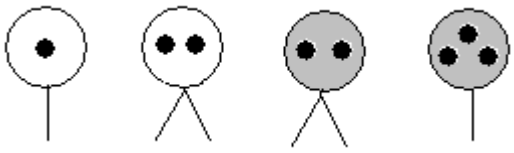

Fig. 2 The four cells to be clustered by COBWEB. (Gennari et al, 1989, Luger and Stubblefield, 1998)

COBWEB generates the following hierarchy of concepts:

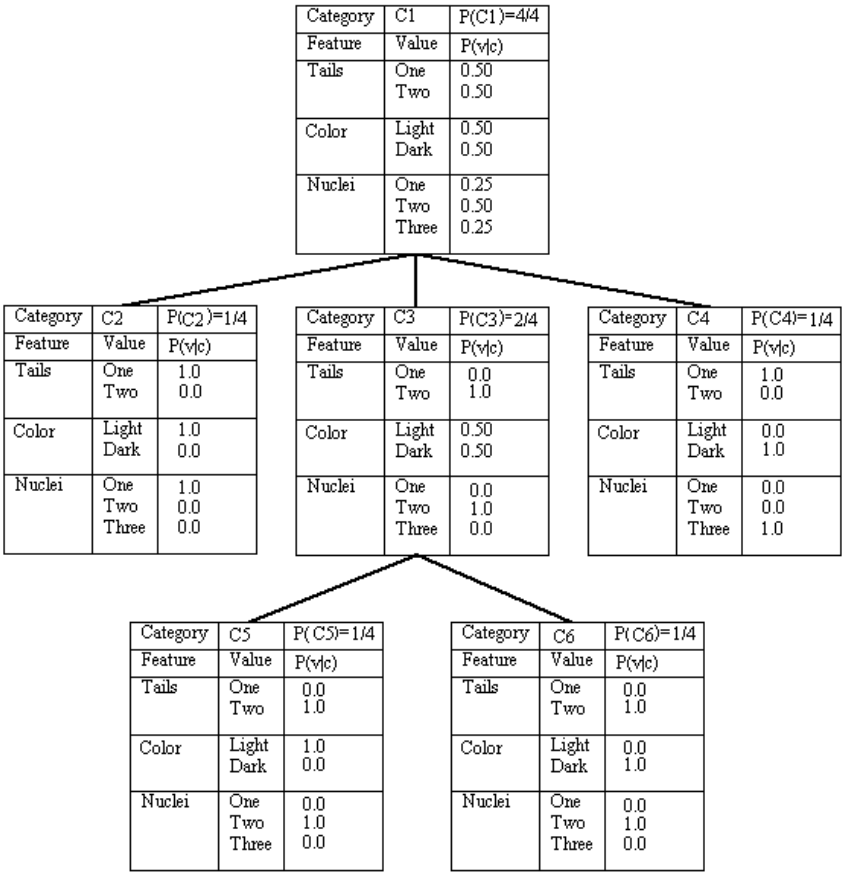

Fig. 3 The concept taxonomy produced by COBWEB. (Gennari et al, 1989, Luger and Stubblefield, 1998)

Just as in our previous example, we can represent this hierarchy in RDF Schema with a diagram:

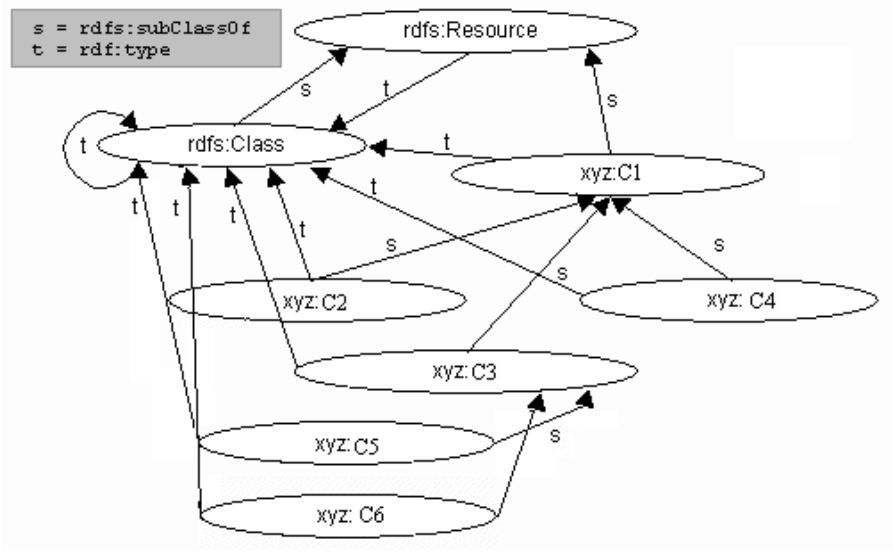

Fig. 4 The Class Hierarchy corresponding to the COBWEB concept taxonomy

The actual XML looks like this:

```
<rdf:RDF xml:lang="en"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
<rdf:Description ID="C1">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf
  rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Resource"/>
</rdf:Description>
<rdf:Description ID="C2">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#C1"/>
</rdf:Description>
<rdf:Description ID="C3">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#C1"/>
</rdf:Description>
<rdf:Description ID="C4">
```

```
 <rdf:type resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#C1"/>
</rdf:Description>
<rdf:Description ID="C5">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#C3"/>
</rdf:Description>
<rdf:Description ID="C6">
 <rdf:type resource="http://www.w3.org/2000/01/rdf-
schema#Class"/>
 <rdfs:subClassOf rdf:resource="#C3"/>
</rdf:Description>
</rdf:RDF>
```

5. Applications

Now we turn to the question of an application for this approach to ontology generation. We will discuss this in the context of Smart Radio, which is a web-based song recommendation system that relies on users' ratings of songs. The user builds playlists consisting of ten songs that they choose from the database. They may then listen to the songs and rate them on a scale of one to five, where one indicates a strong dislike for song, and five indicates a strong liking.

Track	Artist	Your Rating (1=bad,5=fab)
Wednesday Night Prayer Meeting	Mingus, Charles	1 2 3 4 X
My Jelly Roll Soul	Mingus, Charles	1 2 3 4 X
Everytime We Say Goodbye	Coltrane, John	1 2 3 X 5
My Funny Valentine	Baker, Chet	1 X 3 4 5
Someone To Watch Over Me	Baker, Chet	X 2 3 4 5
It Ain't Necessarily So	Hancock, Herbie	1 2 3 4 5
St. Thomas	Rollins, Sonny	1 2 3 4 5
Satin Doll	Duke Ellington	1 2 3 X 5
Take the "A" Train	Fitzgerald, Ella	1 2 3 4 5
Freddie Freeloader	Davis, Miles	1 2 3 4 X

Fig. 5 An example Smart Radio playlist showing the songs rated by the user.

This results in something like the following matrix of values:

Table 1. **Smart Radio data showing how users have rated songs.**

	Song 1	Song 2	Song 3	Song 4	Song 5	Song 6
User 1	5	2		4	2	
User 2	2	5	1		2	5
User 3	1	1	5			3
User 4					4	2
User 5	3	1	5	5		
User 6		4	1	1	5	3
User 7	1			1		
User 8		3	1	4	1	5

The Smart Radio system currently relies solely on Automated Collaborative Filtering (ACF) to make recommendations. We are experimenting with using knowledge discovery techniques to enhance the quality of these recommendations. In particular, we have employed the COBWEB algorithm to generate a hierarchy of concepts. To do this, we define a song to be *good* for a user if and only if she gives that song a rating of 4/5 or more; below this, the song is *bad* for the user. Then, we characterise each song in the database as an object with the same number of attributes as there are users in the database. Each attribute can take on the value *good* or *bad* according to how the user rated the song. The following is an example song object, where question marks symbolise missing values:

```
good bad ? ? good ? good ? ? ? ? ? ? good ? ? ? ? ? ?
? ? good ? ? ? ? ? good bad ? ? ? ? good good ? ? ? bad
? ? bad ? bad good ? ? ? bad ? bad ?
```

When COBWEB is run on the complete set of these objects, we acquire a hierarchy of clusters and associated probabilistic descriptions. An example cluster, arbitrarily named ‘C112’, is presented below:

```
['Break On Through (To The Other Side)', 'Mr Tambourine
man', 'Say hello wave goodbye', 'Hallelujah', 'Unfin-
ished Sympathy', 'Where The Wild Roses Grow', 'Please
Forgive Me', 'The Girl From Ipanema', 'Gin Soaked Boy',
'Street Spirit (Fade Out)', "La Femme D'Argent", 'Right
Here, Right Now', 'Redemption Song']
```

We can then go on to translate the output of COBWEB into a class hierarchy that can be rendered in RDF Schema as outlined above.

The advantages of this approach to this sort of domain is best discussed in light of the fundamental goal of the Semantic Web project, which is to create ‘an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users’ (Berners-Lee et al, 2001). In the context of an online music community, one of these tasks is going to be the recommendation of songs to users. A user might, for example, request that an agent finds her new songs similar to those she has liked in the past. Or she might request that new songs be provided which are some-

what dissimilar to those that she has previously encountered, but which she may nevertheless like. An ontology facilitates the fulfilment of these requirements, because similar songs will fall under the same concept, and degrees of similarity/dissimilarity will hopefully be captured in the relationships between concepts.

The usual way of creating an ontology is for domain experts to establish the fundamental concepts, objects, relations, etc, which exist for a given community. This presupposes that these ontological elements can be uncovered *a priori*. However, in domains such as that of Smart Radio, it is not at all clear that any *a priori* analysis by a team of experts could yield the sort of concepts important to recommendation tasks. While songs may be categorised according to artist, and while to a much lesser extent, genres and sub-genres may be employed, this approach is inadequate, since it does not account for the fact that many people like songs that are widely divergent according to the artist and genre criteria. By using such algorithms as COBWEB to cluster songs based on user ratings, we hope to discover structures more truly reflective of the similarities and dissimilarities between songs. We need only evaluate the resulting conceptual structures in terms of their impact on recommendations, and we need not worry that users may be unable to articulate the hypothesised perceived similarities and dissimilarities between songs. Furthermore, we do not expect that the discovered conceptual hierarchy will map onto any existing and already familiar network of human concepts. Rather, we expect to discover structures that it was never feasible for human experts to detect.

A further advantage conferred by the automatic generation of ontologies using COBWEB and related systems is that such concept formation algorithms are *incremental*, in the sense that observations are not processed *en masse*. There is a stream of objects, which is processed over time. In the case of Smart Radio, this means that the conceptual hierarchy can automatically evolve over time as new songs are added to the database, and as new users join the system. This is something that would be very costly if human experts were involved, and yet such a capacity to evolve over time is essential to a constantly expanding online community resource.

Finally, we may wonder at how such automatically generated ontologies, which do not map onto any existing human understandable ontologies, can fulfil the requirement for interoperability across web sites. So far, we have considered how one online community can be held together and enhanced by such ontologies, but we now turn to a consideration of how other online communities with similar assets (in this case, songs) could exchange information, via agents, with our site. In traditional ontology engineering, collaboration is required between the people who run Web sites and online communities. It is no different in the case where we are employing automatic ontology generation techniques. The only difference is that it is not human beings who collaborate, but, rather, machines. If the Smart Radio database is accessible to a COBWEB-based agent, and another, different, database of songs and user ratings from a hypothetical Smart Radio II, is also accessible to the same agent, then there would be no problem in that agent constructing, maintaining, and evolving a shared ontology for both sites. The only limitation is that the agent must be able to understand the structure of all such databases. Collaboration requires a set of standards and conventions for the construction or description of such databases, and such collaboration is entirely within the spirit of the Semantic Web project.

Summary and conclusion

We have discussed how hierarchical clustering algorithms may be employed to automatically construct basic ontologies, and illustrated this in the context of COBWEB and RDF Schema. We have argued that such an approach is highly appropriate to domains where no expert knowledge exists, or where it proves inadequate, and have gone on to propose how we might employ software agents to collaborate, in place of human beings, on the construction of shared ontologies. This benefits recommendation tasks, in particular, by allowing for the evolution of concept hierarchies which do not match any articulated human conceptual structures, but which are, hopefully, closely reflective of the criteria that people employ in rating online assets. If the task of Semantic Web project is to render human understandable resources processable by machines, we might say that the task envisaged here is to extend the resources processable by machines beyond the domain of human understanding – but always with a view to helping humans carry out their online tasks.

References

1. Berners-Lee, T. (1998). Semantic Web Road map. Available online at http://www.w3.org/DesignIssues/Semantic.html.
2. Berners-Lee, T., Hendler, J., Lassila, O. (2001). The Semantic Web. *Scientific American* feature article, May 2001. Available online at http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html.
3. Brickley, D., Guha, R.V., (Eds.) (2000). Resource Description Framework (RDF) Schema Specification 1.0. W3C Candidate Recommendation 27 March 2000. Available online at http://www.w3.org/TR/rdf-schema/.
4. Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. Machine Learning, 2, 139-172.
5. Gennari, J.H., Langley, P., Fisher, D. (1998). Models of incremental concept formation. Artificial Intelligence, 40(1-3):11-62.
6. Gluck, M.A., Corter, J.E. (1985). Information, uncertainty, and the utility of categories. *Proceedings of the Seventh Annual Conference on Artificial Intelligence*, pp. 831-836, Detroit, MI: Morgan Kaufmann.
7. Hayes, C., Cunningham, P. (2000) *Smart Radio: Building Music Radio on the Fly*. Expert Systems 2000, Cambridge, UK, December 2000.
8. Luger, G.F., Stubblefield, W.A., (1998). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Third Edition. Addison-Wesley, 1998.