# An Empirical Investigation of Learning from the Semantic Web

Peter Edwards, Gunnar AAstrand Grimnes*, and Alun Preece

Department of Computing Science
King's College
University of Aberdeen
Aberdeen, AB24 5UE
Scotland
+44 (0)1224 272270
{pedwards,ggrimnes,apreece}@csd.abdn.ac.uk

**Abstract.** The Semantic Web is a vision of a machine readable Web of resources, interlinked and connected through meta-data with common ontologies. In this paper we explore the impact such a Semantic Web would have on Machine Learning algorithms used for user profiling and personalisation. Our hypothesis is that learning from the Semantic Web should outperform traditional learning from today's World Wide Web for both performance and accuracy. In this paper we present results obtained with two different datasets marked-up with semantic meta-data; using these we have investigated different instance representations and various learning techniques. Our initial results with the Naïve Bayes and K-NN algorithms were disappointing, leading us to examine the use of the Progol algorithm. Using ILP techniques we were able to discover meaningful and we believe, potentially reusable knowledge.

* Author to whom all correspondence should be addressed.

# 1 Introduction

*The Semantic Web* [2][1] is a vision in which today's Web will be extended with machine readable content, and where every resource will be marked-up using machine readable meta-data. The intention is that documents on the Semantic Web will convey real meaning by using structured data-formats and by referring to common ontologies. We believe that initially the Semantic Web will consist of hand-crafted pages much like the Web we know today, providing the same information, but in machine readable form. For example, we could envisage semantic markup accompanying a conventional HTML page giving information such as: *This is the web page of Gunnar Grimnes, he works for Aberdeen University, his telephone number is 1224 630538 etc.* We believe that such static information demonstrates only part of the potential for Semantic Web technologies; their deployment should allow for advanced profiling methods capable of acquiring knowledge such as: *Gunnar likes bands who recorded most of their material from 1968 to 1975, as well as any band who uses a Moog Synthesiser.* When a true Semantic Web exists in this form it becomes useful, and should, for instance, allow for better matching of semantically enriched product descriptions with semantic user profiles.

*Machine learning* technologies have been applied in the context of today's World Wide Web to help users find their way through the unmanageable amount of information that exists. A typical scenario involves acquisition of a model of a user's interests which can then be used to make recommendations, e.g. *this link should be of interest* or *consider this product, it is similar to things you've bought previously* . A variety of approaches exist for learning from Web-content; these range from methods which choose to ignore all HTML markup and treat everything as plain text, to those which make use of the limited structure in HTML and treat the title, heading, link-texts differently. Once content has been extracted from documents, the next step is to apply information retrieval techniques, such as stopword removal, stemming, term weighting and so on. A bag-of-words representation is then used to form the training instances required by the learning algorithm. Figure 1 provides a schematic view of this process.

In the work presented here we wish to explore the impact of the Semantic Web on user profiling and personalisation; more specifically, we have investigated how machine-learning techniques could be used if we had access to semantic markup for every Web resource. Our hypothesis is that the Semantic Web should help solve fundamental problems that make machine learning from the Web today difficult, by providing structured information, reducing ambiguity, and providing useful references to background information in the form of ontologies. We suggest that learning from semantically marked-up data should outperform learning from unstructured or semi-structured text, with regard to increased accuracy, i.e. more meaningful and more usable results, as well as a decrease in the time and resources needed to execute the learning algorithm.

[1] World Wide Web Consortium Semantic Web Initiative, http://www.w3.org/2001/sw/

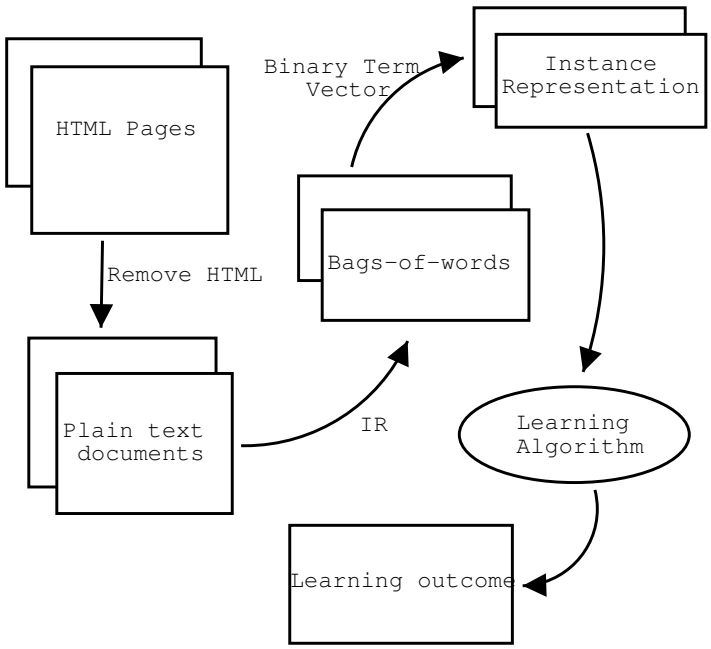


**Fig. 1.** Learning from the Web – Schematic View

### 1.1 Methodology

The approach we have taken is based on a typical machine learning application in the context of the World Wide Web: user-profiling. This scenario provides an opportunity to explore the behaviour of a number of learning algorithms. We assume that a user has interacted with a system on several occasions, perhaps by rating a Web page or making a purchase at an e-commerce site. For our purposes the exact scenario does not matter. Each of these interactions form a training instance, labelled with some class depending on the action performed by the user. For example (s)he might have rated a book "Very good", so the data about that book then becomes the instance and its class would be "Very good". The challenge is to use a set of such instances to acquire a classification model which can be used to predict class labels for future instances. For example, in an e-business context, such a model could then be used to recommend products to a user. To explore the impact of semantic markup, we require a number of datasets of interactions which exist in a semi-structured text format, as well as in a Semantic Web language, such as RDF[2]. We would then be able to compare the performance of learning from the plain text format with learning from semantic meta-data.

In the next section, we describe the datasets we have used for our experiments, then in section 3 we discuss our experiments with knowledge sparse learning, the algorithms used, instance representations and results. In section 4 we discuss knowledge intensive

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

learning, using the Inductive Logic Programming algorithm Progol, again discussing algorithms, document representation and results. Finally, we discuss some related work and present our conclusions.

## 2 Datasets

When commencing this work we knew that the Semantic Web was still very much in its infancy, but we still hoped that it would be possible to find semantically marked-up data upon which to base our experiments. Unfortunately, we have been unable to find any substantial amount of data containing such markup[3]. For this reason we were forced to generate our own (perhaps rather artificial) data. It is our hope, however, that this data will serve to illustrate the issues associated with learning from the Semantic Web.

### 2.1 The ITTalks Dataset

ITTalks[4] is an online portal for information about information technology seminars given at universities in the US. It was the only application of a Semantic Markup language we were able to find which had sufficient amounts of such data publicly available. It uses DAML+OIL[5] to describe talks, talkers, location and other concepts relating to seminars. The system is public and anyone is free to submit their own talk. The talks come in several formats, either as a plain HTML Web page or as DAML+OIL. Figure 2 shows an example of both formats.

Although these documents are generated from the same back-end database, there are some differences in content, e.g. the HTML version includes a biography of the author, while the DAML+OIL variant includes more information on time and place, etc. The ITTalks set contained descriptions of 64 talks, each of which formed an instance in our dataset. The 64 instances were manually classified by each of us (PE, GAG, ADP) into two classes, either *interesting* or *not interesting* based on the title, author and abstract. Three classified variants of the raw data were thus created. Figure 5 summarises the class distributions of each version.

### 2.2 The Citeseer Dataset

The NEC ResearchIndex[6] is a digital library of research papers within Computing Science. It does not provide documents with semantic markup, but the lack of any other sources of data with appropriate markup (other than ITTalks) forced us to look for different ways of acquiring such data. The ResearchIndex provides the full text of the papers, and an indexing system for citations; in addition, it provides the corresponding BibTex entries, see Figure 3.

BibTex is a highly structured format, and is therefore easily converted to an XML based format, such as RDF. We chose RDF, as it is the W3C's basic Semantic Web

[3] We would be delighted if anyone could point us at such a datasource!
[4] http://www.ittalks.org
[5] http://www.daml.org
[6] http://citeseer.nj.nec.com/cs

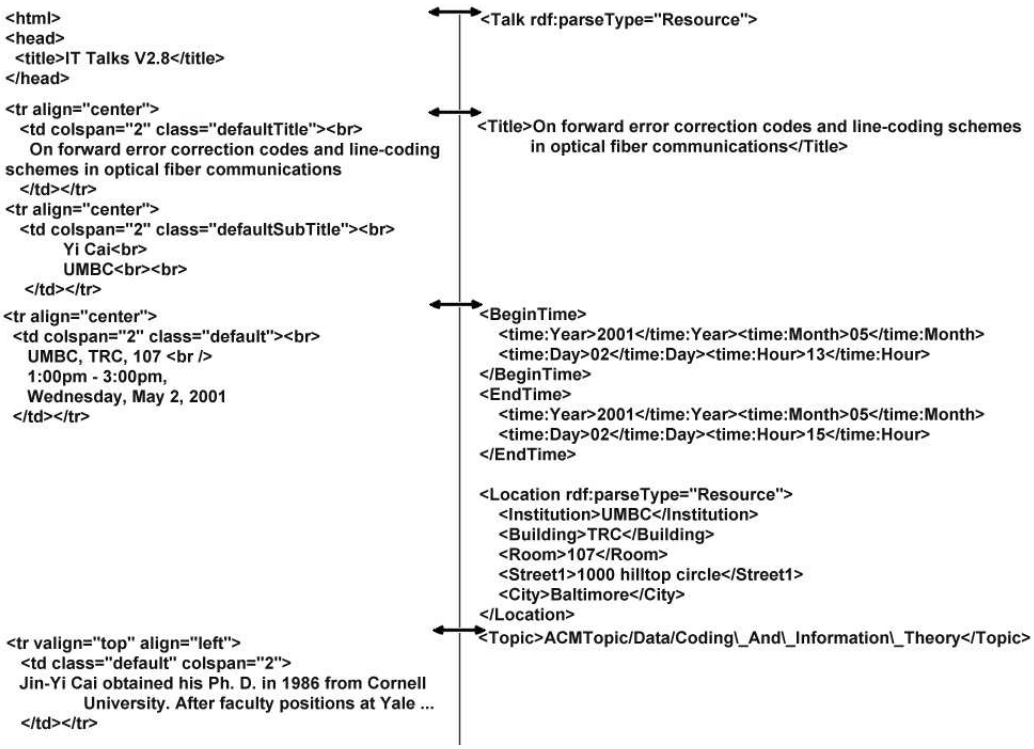


**Fig. 2.** HTML and DAML+OIL Examples from ITTalks.

```
@inproceedings{ zucker92performance,
    author = "R. Zucker and J.-L. Baer",
    title = "A Performance Study of Memory Consistency Models",
    booktitle = "Proceedings of the 19th International Symposium on Computer Ar-
chitecture",
    address = "Gold Coast, Australia",
    year = "1992",
    url = "citeseer.nj.nec.com/zucker92performance.html" }
```

**Fig. 3.** Example BibTex Entry from NEC ResearchIndex.

representation language. The conversion to RDF was performed by making the identifier of the paper (i.e. *zucker92performance*) the subject and each attribute-value line a predicate and object in an RDF triple. Figure 4 illustrates the RDF generated from the BibTex appearing in Figure 3.

```
<?xml version="1.0"?>
<rdf:RDF
        xmlns="http://www.csd.abdn.ac.uk/~ggrimnes/exp/\#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\#">
<inproceedings rdf:about="zucker92performance">
        <author>R. Zucker and J.-L. Baer</author>
        <title>A Performance Study of Memory Consistency Models</title>
        <booktitle>Proceedings of the 19th International Symposium on Computer Ar-
chitecture</booktitle>
        <address>Gold Coast, Australia</address>
        <year>1992</year>
        <url>citeseer.nj.nec.com/zucker92performance.html</url>
</inproceedings>
</rdf:RDF>
```

**Fig. 4.** RDF Example Generated from BibTex.

The meta-data generated from BibTex is very simple, and lacks many of the properties of a real Semantic Web resource, such as ontology references. One notable characteristic is that it is much shorter than the plain text representation of the same instance (typically ≈5000 words). In addition, the BibTex does include some information that is not to be found in the full text, such as *journal title* and *year of publication*.

The ResearchIndex provides a *Computer Science Directory*[7] listing the most cited papers in each of 17 subject categories. In these categories ResearchIndex lists 5066 papers. However, a number of these appear under more than one category and were removed, to leave 4220 instances in our dataset. One approach to learning from this data would be to attempt to learn a classifier capable of discriminating between the 17 categories. However, as a multi-class problem is substantially harder than a simple binary like/dislike classification, we also ran the experiment attempting a binary classification over each of the 17 categories, i.e. *is this paper an* Agents *paper, or is it any of the other 16 classes?* We chose the five categories with the most instances for binary classification, namely *Machine Learning, Artificial Intelligence, Information Retrieval, Human*

[7] http://citeseer.nj.nec.com/directory.html

*Computer Interaction* and *Databases*. We also selected the *Agents* as one of the smaller groups. Figure 6 presents the class distribution for this dataset. Note that the GAG variant has 10 more instances as more talks became available from ITTalks, however these were never classified by PE or ADP.

| | **Instances** | **Likes** | **Dislikes** |
|---|---|---|---|
| GAG | 64 | 25 | 39 |
| PE | 53 | 16 | 37 |
| ADP | 53 | 15 | 38 |

**Fig. 5.** ITTalks Dataset Profile.

| | **Dataset** | **ML** | **AI** | **IR** | **HCI** | **DB** | **Agents** |
|---|---|---|---|---|---|---|---|
| Number of Instances | 4220 | 385 | 367 | 337 | 284 | 285 | 218 |

**Fig. 6.** ResearchIndex Dataset Profile.

## 3 Knowledge Sparse Learning

### 3.1 Introduction

We define *Knowledge Sparse Learning* as the approach that is traditionally taken by statistical or probabilistic machine learning methods. In the Web context, the presence or absence of certain discriminating keywords within a set of training instances is used to produce a classification model. Such a model is then used to predict the class of future unseen instances. Such a model typically consists of a set of weights or a probabilistic model for terms occurring within each class. A model learned from such an approach is mainly useful within the given experiment, and will be useless for classifying instances from a different subject domain, where the discriminating keywords might be different, or even instances from the same domain which have been preprocessed in a different way.

### 3.2 Algorithms

For this part of our study, we used two well known machine learning algorithms, both of which operate using a feature vector representation of each instance. The features used to describe instances varied between our different approaches (see below). N-fold cross validation [21] was used when assessing the performance of the methods. The algorithms will now be discussed briefly.

*Naïve Bayes* [11] is a simplified version of the Bayes classifier, which takes a probabilistic approach to learning. Naïve Bayes reduces the complexity of the normal Bayes classifier by making the assumption that the features of an instance are conditionally independent. In practice, this assumption seldom holds true, but the algorithm still performs well on many real-world classification tasks, having been shown to be equal in accuracy to neural networks and decision tree learning [10].

*K-Nearest Neighbour* (KNN) [4] is an algorithm which will label unknown instances with the label of the majority of the K nearest neighbours in N-dimensional space, where N is the number of features used for describing each instance. KNN is a lazy algorithm, meaning that it will not generate a model based on the training instances, but only when asked to classify a new instance will it perform the computations needed to classify the given instance. The version of KNN used in our experiments is a variation of the standard algorithm, able to deal with symbolic inputs [3].

### 3.3 Instance Representations

We have investigated three different ways of representing instances in this study: one based on the text content of our documents, and two which make use (in some way) of the semantic meta-data. We will now describe each of these approaches.

**1. Plain Text** As a baseline approach we used a simple method to create training data [7, 1]. This was based upon the HTML version of the ITTalks data and the full text of the ResearchIndex articles. For each instance we removed all numbers and all words of length less than 3, before applying a stopword list which removes non-content words such as *it, the, their, etc.* We also explored application of a stemming algorithm [18], reducing words like *computer, computing, computers* to *comput*. The idea was to make generalising over classes easier, but we found stemming to make little or no difference in performance, and chose not to use it. Once this initial pre-processing had been completed, we had the option of either creating a binary vector with each element corresponding to a term in the document vocabulary, or some form of weighted vector (based on a subset of available terms). Due to the size of the vocabulary (150,000 terms for the ResearchIndex dataset) we decided to adopt the latter approach.

TF/IDF weights [6] were calculated for each of the terms and the 1500 with the highest TF/IDF ranking were selected. The presence or absence of each of these terms was then used to create a binary term vector. Figure 7 shows an example of the processing stages involved and the final representation.

**2. Treating RDF Tags as Plain Text** Our next approach was similar to the first, but instead of employing a plain text representation for each instance we make use of the marked-up data. This data was preprocessed in essentially the same manner as the plain text, with one important difference. In HTML documents the tags provide formatting information, i.e. $<b>$ tells us that this text should be printed in bold, while in the meta-data files the different XML/RDF tags represent some information about the meaning of the content, i.e. $<location>$ tells us that a talk has a location. We did not want to

Original HTML document:

```
<html>
<head><title>Machine Learning from the Semantic Web</title></head>
<body>
<h1>Machine Learning from the Semantic Web</h1>
<i>By Gunnar AAstrand Grimnes</i>
<p>In this seminar we give details on our recent experiments on learn-
ing from the semantic web

...
```

⇓

Removal of HTML markup, stopwords and numbers:

*machine learning semantic web machine learning semantic web gunnar aastrand grimnes seminar give details recent experiments learning semantic web ...*

⇓

Selection of most discriminating terms using TF/IDF:
*learning, semantic, ontology, agent, talk, experiments, url ...*

Binary term vector for this instance:
*1, 1, 0, 0, 0, 1, 0 ...*

**Fig. 7.** Instance Representation – Method 1.

ignore this information, so in this approach we treat the XML-tags as additional text content. As we use TF/IDF to select highly ranked terms to appear in the instance representation, commonly occurring tags will of of course be ignored. However, tags which occur infrequently will still find their way into the instance. Figure 8 shows an example of this instance representation.

**3. Using RDF with Tag ⇒ Feature Mapping** Our third approach was based on the observation that on the Semantic Web, markup provides structure, so instead of throwing away this structure by treating all the text as one unit we processed the content of each tag separately. Each element in our instance vector then became the set of words which occurred within a certain tag; the content of this tag was pre-processed in the same manner as for methods 1 and 2, i.e. applying a stopword list, ignoring short words, etc. The length of the instance vector then became the number of unique tags in the documents, not the number of unique words. This approach is possible because the variants of both Naïve Bayes and K-Nearest Neighbour we used supported sets of values as elements in the instance vectors.

While this approach made sense for most tags, some tags had a clearly defined internal structure where information would be lost if pre-processed. An example is the *ACMTopic* field of the ITTalks dataset, which gives the topic of the talk as a string such as *ACMTopic/Computer_Systems_Organization/Computer_Communication_Networks/-Internetworking*. If preprocessed normally this would be broken up into separate terms and the accurate meaning lost, so we did not preprocess this tag. Figure 9 shows an example of this instance representation.

Original RDF document:

```
<xml>
  <rdf>
    <talk id='mlsemweb1'>
      <title>Machine Learning from the Semantic Web</title>
      <speaker>
         <name>Gunnar AAstrand Grimnes</name>
         <url>http://www.csd.abdn.ac.uk/~ggrimnes</url>
         <faxnumber>+44 1224 273422</faxnumber>
      </speaker>

...
```

⇓

Removal of stopwords and numbers:
*xml rdf talk title machine learning semantic web speaker name gunnar aastrand grimnes name url csd abdn ggrimnes url faxnumber speaker ...*

⇓

Selection of most discriminating terms using TF/IDF:

*learning, semantic, ontology, faxnumber, agent, experiment ...*

Binary term vector for this instance:

*1, 1, 0, 1, 0, 0, ...*

**Fig. 8.** Instance Representation – Method 2. (Note difference from term vector in Figure 7)

### 3.4 Results

*The ITTalks results* are shown in Figures 10 and 11. From these results, it is immediately noticeable that the GAG variant of the classified ITTalks data led to poor results. We believe that this is an artifact due to the manual classification of the data; the GAG variant reflects a less clearly defined interest profile than the two other variants, both of which were created by academics researchers with specific interests. Another phenomenon we can observe from the results is the poor performance of Method 3 (mapping RDF tags to features). We believe this is caused by the large number of distinct tags which appear in the ITTalks meta-data, and the fact that the majority of the textual content is contained within very few tags, such as *Abstract, Bio-Sketch* and *Title*. This would cause the instance representation for Method 3 to have very sparse vectors with a few features containing large numbers of terms; there are thus many redundant features which do not provide any information that can be used for creating the model. Method 2 is not affected by this as the entire document is treated as one unit of text for instance generation purposes. Finally, we observe that Method 1 and Method 2 lead to very similar results. We believe that this is caused by another artifact of the dataset. All of the instances in the dataset contain the same set of DAML+OIL tags, even if these might be empty for certain instances. As we use TF/IDF to select highly ranked terms for inclusion in the instance vector, and tag names appear in all examples, they will never get into the final instance representation. Thus, the plain text and meta-data versions of this dataset are essentially the same. We would anticipate that in a true Semantic Web

Original RDF document:

```
<xml>
  <rdf>
    <talk id='mlsemweb1'>
      <title>Machine Learning from the Semantic Web</title>
      <speaker>
         <name>Gunnar AAstrand Grimnes</name>
         <url>http://www.csd.abdn.ac.uk/~ggrimnes</url>
      </speaker>

...
```

⇓

Removal of stopwords, numbers, etc. from tag content:

```
<xml>
  <rdf>
    <talk>
      <title>machine learning semantic web</title>
      <speaker>
         <name>gunnar aastrand grimnes</name>
         <url>csd abdn ggrimnes</url>
      </speaker>

...
```

⇓

Using the following tags as features:
*talk, title, speaker, name, url . . .*
Instance:

{}, { *machine, learning, semantic, web* }, {}, {*gunnar, aastrand, grimnes*}, {*csd, abdn, ggrimnes*} . . .

**Fig. 9.** Instance Representation – Method 3.

| | GAG | PE | ADP | Average |
|---|---|---|---|---|
| **1. Plain Text** | 48.27% | 69.38% | 65.30% | 60.98% |
| **2. RDF Tags as text** | 50.00% | 65.30% | 67.34% | 60.88% |
| **3. RDF Tags as Features** | 34.48% | 40.81% | 40.81% | 38.70% |

**Fig. 10.** Results for Naïve Bayes - ITTalks Dataset.

| | GAG | PE | ADP | Average |
|---|---|---|---|---|
| **Method 1: Plain Text** | 55.17% | 73.47% | 65.31% | 64.65% |
| **Method 2: RDF Tags as Text** | 56.90% | 69.39% | 59.18% | 61.82% |
| **Method 3: RDF Tags as Features** | 50.00% | 65.31% | 57.14% | 57.48% |

**Fig. 11.** Results for K-Nearest Neighbour - ITTalks Dataset.

| | Multi Class | ML | AI | IR | HCI | DB | Agents |
|---|---|---|---|---|---|---|---|
| **Method 1: Plain Tex** | 43.38% | 83.84% | 70.02% | 77.35% | 78.69% | 85.02% | 78.93% |
| **Method 2: RDF Tags as Text** | 47.53% | 91.71% | 89.76% | 91.06% | 93.67% | 94.43% | 95.23% |
| **Method 3: RDF Tags as Features** | 51.13% | 89.62% | 88.05% | 90.33% | 91.51% | 91.85% | 92.82% |

**Fig. 12.** Results for Naïve Bayes - ResearchIndex Dataset.

| | Multi Class | ML | AI | IR | HCI | DB | Agents |
|---|---|---|---|---|---|---|---|
| **Method 1: Plain Text** | 46.52% | 93.39% | 91.26% | 94.00% | 93.96% | 95.47% | 96.40% |
| **Method 2: RDF Tags as Text** | 26.47% | 91.73% | 90.73% | 92.39% | 93.41% | 94.00% | 94.95% |
| **Method 3. RDF Tags as Features** | 24.19% | 89.83% | 89.69% | 90.21% | 93.10% | 92.65% | 98.58% |

**Fig. 13.** Results for K-Nearest Neighbour - ResearchIndex Dataset.

dataset, the meta-data would be richer in nature and its presence would provide more information than the pure text instances.

*The ResearchIndex results* can be found in Figure 12 and 13. The first column gives results for the multi-category problem, and as expected, they are poor. K-Nearest neighbour performs much worse than Naïve Bayes at the multi-class classification, we believe this is caused by K-nearest neighbour being very susceptible to the inclusion of irrelevant or redundant attributes, as the distance metric combine measurements for all of the features [16].

## 4 Knowledge Intensive Learning

### 4.1 Introduction

On the Semantic Web content is represented via a logical language in which *meaning* is clearly defined. In our first group of experiments, while some use was made of the structure provided by markup, its logical nature was ignored. By exploiting the full potential of the Semantic Web we argue that it should be possible to learn rules and statements in a logical representation that is similar to that used for the content.

### 4.2 Inductive Logic Programming

We have chosen to use the Progol Inductive Logic Programming (ILP) system. Progol has been defined as "A standard Prolog interpreter with inductive capabilities" [15]. It is able to learn knowledge (expressed as Prolog predicates) from supplied example instances and supporting background information. The algorithm has been successfully used in experiments for analysis of mutagenic activity amongst nitroaromatic molecules [20], drug design [5] and protein shape prediction [14]. The theory behind ILP and the original Progol algorithm is described in [13]. For our experiments we used CProgol4.4.[8]

[8] Progol is freely available online from http://www.doc.ic.ac.uk/∼shm/Software/.

### 4.3 Methodology

We explored the application of Progol in the context of the NEC ResearchIndex Dataset, as it is the larger of our datasets and maps easily to a Prolog representation. The ITTalks dataset has a much richer set of meta-data which is more difficult to represent in Prolog. In our RDF→Prolog mapping, we have used the simple RDF data model of {*subject, predicate, object*} triples; the ITTalks dataset is encoded using DAML+OIL, which, when treated as plain RDF, generates many complex reification triples which would shroud the meaning of the documents, compared to the intuitive meaning of the simpler triples from the ResearchIndex, such as *the author of this article is Gunnar Grimnes*.

Progol was run on a randomly selected subset of 1000 of the total 4220 papers from the ResearchIndex, as running experiments with the full dataset would have taken too long, as we were continuously tweaking learning parameters and instance representations. As before, we ran binary experiments over the different classes, but with Progol we attempted to learn a classification for each of the 17 classes. We used a single Prolog predicate of the form *inClass( +article )* to represent class membership. This became the target clause which Progol would try to learn.

### 4.4 RDF as 1st Order Logic

Initially we chose a very simple approach to map RDF to Prolog. As the RDF data-model represents *(subject, predicate, object)* triples, we employed a single Prolog predicate called *triple*. Figure 14 illustrates our initial representation and corresponds to the RDF appearing in Figure 4. Note how the BibTex type maps to a RDF concept within the namespace of these experiments.

```
triple( url, zucker92performance,
    'citeseer.nj.nec.com/zucker92performance.html' ).
triple( booktitle, zucker92performance, 'Proceedings of the 19th
    International Symposium on Computer Architecture' ).
triple( type, zucker92performance,
    'http://www.csd.abdn.ac.uk/~ggrimnes/exp/#inproceedings' ).
triple( address, zucker92performance, 'Gold Coast, Australia' ).
triple( title, zucker92performance, 'A Performance Study of
    Memory Consistency Models' ).
triple( year, zucker92performance, '1992' ).
triple( author, zucker92performance, 'R. Zucker and J.-L. Baer' ).
```

**Fig. 14.** RDF Encoding – Initial Approach.

Perhaps as expected, this approach did not give very good results as the search space for Progol became extremely large. Due to Progol only having one predicate to use in the construction of the result clause, the algorithm would quickly get lost down a faulty path of the search-tree with incorrect constants or incorrect unifications, and never recover.

Our first improvement was to change the way we represented triples. Instead of casting them all to the same predicate, we created a Prolog predicate corresponding

to each RDF predicate, e.g. *triple( author, zucker92performance, 'J. Zucker')* became *author( zucker92performance, 'J. Zucker' )*. Secondly, we recognised that Progol operates on strings and if two literals are not *exactly* equal, Progol has no way of generalising over them. This led us to preprocess all strings so that each word became a separate Prolog fact. For example, instead of *title( learning02grimnes, 'Learning from the Semantic Web' )* we would have: *title( learning02grimnes, 'learning' ), title( learning02grimnes, 'semantic' ) and title( learning02grimnes, 'web' )* . In addition to this simple pre-processing we also applied a list of synonyms for commonly used abbreviations and misspellings, e.g. *proc, procs* and *proceeding* all map to *proceedings*, *sixth* maps to *6th*, etc. Finally, we standardised the representation of author names to first initial plus surname, as BibTex does not specify a standard. This means that *Alun Preece, Preece A.* and *Alun D. Preece* all map to *A. Preece*. As with title words we would created one Prolog fact for each author. An example of our final representation appears in Figure 15.

```
url( zucker92performance, 'citeseer.nj.nec.com/zucker92performance.html' ).
booktitleword( zucker92performance, 'proceedings' ).
booktitleword( zucker92performance, '19th' ).
booktitleword( zucker92performance, 'international' ).
booktitleword( zucker92performance, 'symposium' ).
booktitleword( zucker92performance, 'computer' ).
booktitleword( zucker92performance, 'architecture' ).
type( zucker92performance, 'http://www.csd.abdn.ac.uk/~ggrimnes/exp/#inproceedings' ).
address( zucker92performance, 'Gold Coast, Australia' ).
titleword( zucker92performance, 'performance' ).
titleword( zucker92performance, 'study' ).
titleword( zucker92performance, 'memory' ).
titleword( zucker92performance, 'consistency' ).
titleword( zucker92performance, 'models' ).
year( zucker92performance, '1992' ).
author( zucker92performance, 'R. Zucker' ).
author( zucker92performance, 'J. Baer' ).
```

**Fig. 15.** RDF Encoding – Second Approach.

### 4.5 Results

Lack of space prevents us from presenting the Progol results in full. However, we will discuss some features of the results and will provide illustrative examples. For most classes the majority of the rules discovered by Progol are of the form:
*inClass( zucker92performance )*, meaning that Progol was unable to find any common features between instances of the given class, and simply returned an *inClass* clause that lists all the instances declared to be in that class. This problem is almost certainly caused by the small number of features used to describe each instance, and overlap between some of the classes, making it difficult for the algorithm to identify discriminatory generalisations.

Despite this problem, some rules were discovered that covered more than a single instance. For example, Figure 16 shows the rules generated from the *Agents* class; the

first five rules are straightforward, and encapsulate obvious facts about publications in the area of *agents* technologies. However, the sixth rule, *inClass(A) :- titleword(A,bdi).*, is more interesting. BDI is an abbreviation for *beliefs, desires and intentions*, a common paradigm within agents research [19]. An active researcher in the agents field would find this almost as obvious as the other rules, based on their knowledge and experience of the field. This Progol result is thus a piece of general knowledge, which is not only usable in trying to classify new research papers from the ResearchIndex, but could also potentially be applied outside this experiment. Several of the other classifications also generated rules of a similar type. We find these results from our Progol experiments very exciting, and present a selection in Figure 17. The rules range from the slightly bizarre, such as *all papers published in volume 18 are Theory*, to rules containing interesting and potentially reusable knowledge, such as *Papers published by Morgan Kaufmann appearing in books with learning in the title are in the field of Machine Learning*.

```
Agents:

inClass(A) :- author(A,'A. Rao').
inClass(A) :- author(A,'D. Lambrinos').
inClass(A) :- titleword(A,agent), titleword(A,mobile).
inClass(A) :- type(A,'http://www.csd.abdn.ac.uk/~ggrimnes/exp/#misc'),
textword(A,agent), titleword(A,agent).
inClass(A) :- year(A,1999), titleword(A,agents).
inClass(A) :- titleword(A,bdi).
```

**Fig. 16.** Excerpt of Progol Results - Agents Experiment

Examination of the Progol results indicate that Progol is overfitting to the problem, by creating a large number of *inClass* rules. This is because it is impossible for Progol to generalise any further without creating rules that are not correct for 100% of all the instances. It would be desirable to allow rules that would correctly classify, say, 99% of the instances, thus allowing more generalisations and pruning the set of resulting rules. By doing this we would hopefully get a smaller set of rules for each class, and a much smaller set of *inClass* statements. However, as Progol has no built in method for doing this, we chose to take a very simple approach as follows: all the *inClass* rules were discarded and only the more knowledge-rich rules retained for each class. When this reduced set is used for classification, the precision of the rules is still 100% as no articles will ever be incorrectly classified, however recall will no longer be perfect. The percentage of recalled instances for each class are presented in Figure 18.

## 5 Related Work

We are aware of little work concerned with application of machine learning to Semantic Web data. This is in contrast to applications to the Web, of which there have been many.

```
Artificial Intelligence:
inClass(A) :- journal(A,'SIAM Journal on Control and Optimiza-
tion').
inClass(A) :- journal(A,'Computational Linguistics').

Databases:
inClass(A) :- titleword(A,warehousing).
inClass(A) :- titleword(A,deductive).
inClass(A) :- titleword(A,aggregate).
inClass(A) :- titleword(A,transactions).

Machine Learning:
inClass(A) :- publisher(A,'Morgan Kaufmann'), booktitleword(A,
learning).
inClass(A) :- titleword(A,based), titleword(A,case).

Programming:
inClass(A) :- pages(A,225), booktitleword(A,conference).


Security:
inClass(A) :- booktitleword(A,privacy).
inClass(A) :- titleword(A,watermarking).
inClass(A) :- titleword(A,encryption).

Theory:
inClass(A) :- volume(A,18).
```

**Fig. 17.** Progol Results - Sample Rules

| | ML | AI | IR | HCI | DB | Agents |
|---|---|---|---|---|---|---|
| **Recall:** | 62.50% | 58.93% | 26.92% | 45.31% | 37.50% | 58.93% |

**Fig. 18.** Recall for Pruned Progol Rules - ResearchIndex Dataset.

For example, Syskill & Webert [17] uses machine learning to acquire a model able to predict which links on a Web page a user will find useful. It does this by analysing a set of Web pages manually rated by a user, which are then processed using structural IR techniques. Syskill & Webert uses a Naïve Bayes classifier, but the authors also report investigations using nearest neighbour algorithms, ID3, perceptrons and multi-layer neural networks. Webwatcher [1, 12], like Syskill & Webert, is a browsing aid which attempts to annotate a Web page with information on what links a user might find useful. The authors explored a variety of learning algorithms, such as Winnow [9] and Wordstat. Underlying the system was an instance representation which did attempt to exploit more of the structure of the HTML documents, as link text, headers, etc. were treated differently. Letizia [8] is a browser helper, displayed in a separate window next to the user's browser. It pre-fetches all outgoing links from the current page and will do a breadth first search to advise the user on which links to visit next. Letizia uses TF/IDF to extract content from pages, and uses the weighted terms to identify documents matching the user's interest.

## 6 Conclusions

Our results have demonstrated that today's available Semantic Web markup cannot be expected to outperform conventional machine learning applied to plain text, with regards to accuracy of the learned model. However, it must be noted that applying the same algorithm to the full text of an article of 6000 words, and to 10 lines of RDF code, while still getting equally good predictive accuracy does constitute an increase in performance and scalability. In the Web context this is especially important as algorithms will be expected to scale to millions of pages. Nevertheless, we remain somewhat unsatisfied with our current results for a number of reasons. Although we attempted to find real Semantic meta-data, we admit that we are not completely happy with our datasets. The ITTalks dataset is too small to be able to draw any firm conclusion from any results, and the ResearchIndex dataset has generated meta-data from a source which was never meant to provide real *meaning*. Also, when the Semantic Web becomes reality, many supporting technologies should be available, most significantly ontological support and the availability of inference engines, which should allow for easy generalisations of the kind: *A*, Simulated Annealing and Depth-first are all types of Search algorithms* which could be used to facilitate classification tasks such as the ones we have attempted in this paper, but which are nearly impossible to discover without any background information.

### 6.1 Future Work

We plan to continue exploring issues concerned with Progol and Knowledge Intensive Learning on the Semantic Web, primarily attempting to utilise background information to help Progol generalise better over classes. We plan to explore generation of such background information from ontologies referenced in the meta-data, as well as through the use of general background information such as as synonyms or word similarity.[9].

[9] This data could for example be taken from WordNet (http://www.cogsci.princeton.edu/∼wn/)

We are very interested in trying to apply the resulting Prolog clauses outside the original experiment. As the results are first order logic it should be possible to map them back to a representation in RDF or a similar logic based format, thus exporting the *model* that was learned.

Due to the current shortcomings of Semantic Web data we plan to do further experiments with our current datasets. Primarily we intend to re-classify the ResearchIndex papers based on personal interest, thus moving further towards the personalisation and learning user models scenarios used as motivation for this work. We would also like to run Progol with the full ResearchIndex dataset, not just a small subset of the instances, as well as attempting to find a good way of mapping DAML+OIL to Prolog, so that we may run Progol experiments with the ITTalks dataset.

## References


1. Robert Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. Webwatcher: A learning apprentice for the world wide web. In *AAAI Spring Symposium on Information Gathering*, pages 6–12, 1995.
2. Tim Berners-Lee. What the semantic web isn't but can represent. http://www.w3.org/DesignIssues/RDFnot.html, 1998.
3. Scott Cost and Steven Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.
4. S. Dudani. The distance-weighted k-nearest-neighbour rule. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-6(4):325–327, 1975.
5. Paul W. Finn, Stephen Muggleton, David Page, and Ashwin Srinivasan. Pharmacophore discovery using the inductive logic programming system PROGOL. *Machine Learning*, 30(2-3):241–270, 1998.
6. Salton G and Buckley C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.
7. Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of International Conference on Machine Learning*, pages 331–339, 1995.
8. Henry Lieberman. Letizia: An agent that assists web browsing. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 924–929. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 20– 25 1995.
9. N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.
10. D. Michie, D. Spiegelhalter, and C. Taylor. *Neural and Statistical Classification*, chapter 70, pages 1297–1300. Ellis Horwood, 1994.
11. T. Mitchell. *Bayesian Learning*, chapter 6, Machine Learning, pages 154–200. McGraw-Hill, 1997.
12. D. Mladenic. Using text learning to help web browsing. In *Proceedings of the 9th International Conference on Human-Computer Interaction*. HCI International 2001, New Orleans, LA, USA, 2001.
13. S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.
14. S. Muggleton, R. King, and M. Sternberg. Protein secondary structure prediction using logic-based machine learning. *Protein Engineering*, 5(7), 1992.
15. Stephen Muggleton and John Firth. Cprogol4.4: A tutorial introduction.
16. Terry Payne. *Dimensionality Reduction And Representation For Nearest Neighbour Learning*. PhD thesis, University of Aberdeen, 1999.

17. Michael J. Pazzani, Jack Muramatsu, and Daniel Billsus. Syskill webert: Identifying interesting web sites. In *Proceedings of the American National Conference on Artificial Intelligence, Vol. 1*, pages 54–61, 1996.
18. M Porter. An algorithm for suffix stripping. Technical Report 14, Program, 1980.
19. A. Sloman and B. Logan. Architectures and tools for human-like agents, 1998.
20. A. Srinivasan, R. D. King, S. Muggleton, and M. J. E. Sternberg. Carcinogenesis predictions using ILP. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297, pages 273–287. Springer-Verlag, 1997.
21. M. Stone. Asymptotics for and against cross-validation. *Biometrika*, 64:29–35, 1977.