Learning from Semantic Flora & Fauna

Gunnar AAstrand Grimnes, Pete Edwards and Alun Preece
Computing Science Dept.
King's College
University of Aberdeen
AB24 3UE Scotland
{ggrimnes,pedwards,apreece}@csd.abdn.ac.uk

Abstract

We argue that for a personal agent working within a Semantic Web framework, machine learning is essential. We identify two topologies in the Semantic Web, and refer to these as: (1) semantic forests (disjoint trees) and (2) true semantic webs (complex interconnected graphs). An example of (1) is Citeseer BibTeX mapped to RDF; an example of (2) is FOAF, an RDF representation of people and their relationships.

In this paper we explore a number of techniques (naïve Bayes, K-NN, ILP, clustering) for learning knowledge that is neither explicitly stated nor deducable from such data. The learned knowledge itself consists of first-class Semantic Web statements, maximizing its usefulness and re-usability. We also discuss the need for preprocessing and fault tolerance when learning from real distributed Semantic Web data.

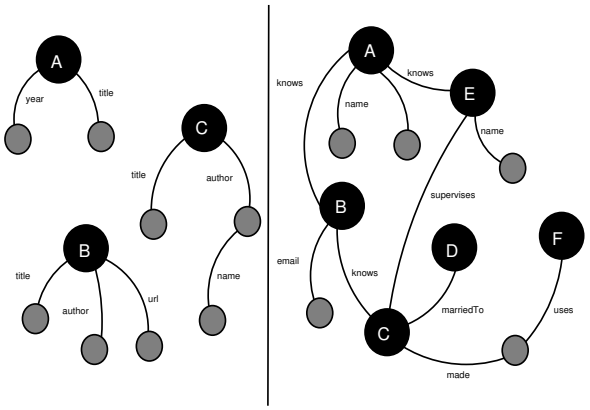

Figure 1: Schematic Illustration of Semantic Forests and Semantic Webs.

Introduction

When the vision of the Semantic Web as set in out (Berners-Lee, Hendler, & Lassila 2001) becomes reality, and everyone has personal agents roaming the semantic cyberspace, one might imagine that machine learning will no longer be required, as by manipulation of logical statements about semantic resources and their descriptions anything can be inferred and understood. However, we believe that no matter how wide-spread and extensive the Semantic Web becomes, every little fact is still unlikely to be explicitly stated, and in addition agents will need to know personalised facts, that although not generally true may be true in a certain context for a certain user. We believe that acquisition of user models from Semantic Web data will be of tremendous importance for personal agents working within a Semantic Web framework.

For our discussion of learning from the Semantic Web, we assume it is using a representation based on *(subject, object, predicate)* triples, for instance RDF (Lassila & Swick 1999). We identify two types of Semantic Web data:

- "Semantic forests" - these consist of many small, disconnected, shallow resource trees. The structure of such a forest is isomorphic to that of an XML document. Real world examples of such semantic forests include metadata using Dublin Core[1], or the use of RDF Site Summary (RSS)[2]. See Figure 1(left).
- "Semantic webs" - consist of a large graph of resources linking to each other, with no clear distinction where one resource description ends and another begins. The only significant real-world example we are aware of to date is Friend of a Friend (FOAF)[3]. See Figure 1(right).

Currently semantic forest data is pre-dominant. The number of true semantic webs should increase as the Semantic Web develops, however we do not expect semantic forests to disappear altogether. These different types of data present us with a challenge: How should Semantic Web learners deal with them? How can the different structures be exploited to improve learner performance? How should the learning outcome be represented to be as useful as possible in both the original context (where it was learned) as well as being portable to other scenarios?

As an example of an application where learning from Semantic Web data would be profitable, consider GraniteNights (Grimnes *et al.* 2003): a multi-agent application

[1] http://dublincore.org
[2] http://web.resource.org/rss/1.0/
[3] http://www.foaf-project.org

which allows a user to schedule an evening out in Aberdeen (aka the "Granite City"). Intended users of GraniteNights are visitors to Aberdeen or anyone else requiring assistance in identifying suitable dining and entertainment venues, as part of an evening schedule. GraniteNights uses Semantic Web technology extensively: RDF is used for all agent communication, for user profiles, for information about public houses, restaurants and cinemas. In addition an RDF based query language (QbEx[4]) is used. To enable personalisation in applications like GraniteNights we must explore how RDF based profiles can be acquired, how they can be stored in RDF, and how learning from RDF can be used to generate them.

In this paper we describe a series of experiments which aim to learn user models; first we describe our experiments with learning from semantic forests, using Naïve Bayes (Mitchell 1997), K Nearest Neighbour (Cover & Hart 1967) and Progol (Muggleton 1995), then we describe our search for true *semantic web* data, the problems we encountered when we found it, and how clustering and ILP were used to learn from it. Finally we discuss related work in this area and present a discussion of what we have achieved to date and our future plans.

Semantic Forests

In our initial experiments we compared learning from semantic forests with learning from plain-text (Edwards, Grimnes, & Preece 2002). We explored the application of two machine learning algorithms commonly used for WWW personalisation, as well as one more knowledge intensive algorithm. As data from the Semantic Web has well-defined structure and semantics, and does not contain as much redundant or irrelevant information as plain text, our hypothesis was that learning from the Semantic Web should outperform traditional learning from today's World Wide Web for both performance and accuracy. For this investigation we used two datasets where instances were available in both plain-text and a format which used semantic mark-up.

The first dataset was derived from a Web site used to showcase Semantic Web technology, called ITTalks[5]. The site provided information about Computing Science seminars at different universities in the US. The dataset contained 64 instances, each of which was manually classified into classes "interesting" and "not interesting" by three users, to create three different learning problems. ITTalks used DAML+OIL (Hendler & McGuinness 2000) for semantic markup.

The second dataset was generated from the NEC ResearchIndex (Citeseer)[6], a digital library of research papers within Computing Science. A total of 4220 papers were extracted from the Citeseer *Computer Science Directory*[7], classified into 17 different subject areas of computer science, e.g. machine learning, human computer interaction, agents, etc. Each instance consisted of the full text of the paper and the BibTeX converted to a very simple RDF/XML representation.

[4]http://www.csd.abdn.ac.uk/research/AgentCities/QueryByExample

[5]http://www.ittalks.org, now defunct.

[6]http://citeseer.nj.nec.com/cs

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

Knowledge Sparse Learning Methods

The first two algorithms used were *knowledge sparse learning* algorithms, i.e. probabilistic or statistical machine learning algorithms which consider each feature of an instance as a separate attribute, and where the learning outcome can be used to classify further instances, but does not offer any insight into how the classification is made. We selected Naïve Bayes and K Nearest Neighbour, both of which use a similar vector based representations for the instances.

We used standard pre-processing methods for the plain-text data: removal of HTML tags, stop-words, short words, numbers and non-letter characters; a binary vector was then created based on the presence or absence of the 1500 words with highest TF/IDF weights (Salton & Buckley 1988) over the corpus.

For the marked up data experiments were performed with two different representations: one treating the RDF/XML or DAML+OIL as plain-text, but keeping the XML tags; and one attempting to preserve the structure of the mark-up in the instance representation. For this second approach each attribute of the vector corresponded to one XML tag, and the value of the attribute became the set of words appearing within that tag, pre-processed as described above. An example of this representation is shown in Figure 2.

Results We do not present the full results of these experiments here, just a selection of observations; refer to (Edwards, Grimnes, & Preece 2002) for the details. For both datasets and both algorithms the performance of the text and RDF/XML as text representations were not significantly different, and the second RDF/XML representation was consistently worse. However, it should be noted that although learning from RDF did not yield better accuracy than learning from text, it did constitute a significant gain in efficiency. For the Citeseer dataset a typical instance consists of a 6000 word paper, or a fragment of RDF with about 5-15 triples. This result is very important in a web context where scalability is an important requirement. These results were not what we had expected and failed to confirm our hypothesis; we identified several reasons why this might be the case:

1. Learning from the Semantic Web should really be done using a machine learning algorithm that can learn symbolic representations, as any potential gain comes from the added structure and semantics which are discarded when using algorithms like K Nearest Neighbour or Naïve Bayes.
2. Our data was flawed, as the datasets did not have any ontological support providing us with background knowledge (and indeed, our methodology did not make use of any such knowledge). In addition our data was very shallow, i.e. there was a superficial level of detail for each instance. The ITTalks dataset was very small by empirical machine learning standards, and the Citeseer dataset was very artificial, being converted from BibTeX, and not real RDF.

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*} ...

Figure 2: Second RDF Instance Representation.

Learning Symbolic Knowledge

Having concluded that a symbolic learning approach was necessary, some further experiments were conducted using the Inductive Logic Programming (ILP) system Progol (Muggleton 1995). Progol takes Prolog predicates as input and learns a model represented as Prolog rules.

For our Progol experiments we used only the Citeseer dataset, in order to focus on one type of markup – RDF. We ran 17 separate experiments with the Citeseer data, exploring the creation of a binary classifier for each class. As the RDF generated by converting BibTeX is very simple, we initially used a very naïve mapping of RDF to Prolog: each RDF triple became a Prolog predicate *triple/3* and the actual RDF properties became the first argument to this triple; see Figure 3 for an example. Note that as only one namespace was used within this dataset that was discarded.

Perhaps as expected, this approach did not give very good results as the search space for the Progol algorithm 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.

In our next approach we changed from using a single predicate *triple/3*, to making the RDF properties first class binary Prolog predicates, taking the resource and object as arguments. We also added some pre-processing stages to

triple(url, zucker92performance,
'citeseer.nj.nec.com/zucker92performance.html').
triple(booktitle, zucker92performance,
'Proceedings of the 19th International Symposium on Computer Architecture').
triple(type, zucker92performance, *'#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').

Figure 3: RDF to Prolog Mapping – Initial Approach.

normalise author names and extract keywords from titles and abstracts. An example of the revised representation is shown in Figure 4. Note how this representation is still somewhat naive, as there is no support for namespaces, no use of background ontologies and no concept of resources having RDF types. However, as the Citeseer data had none of these features these representational aspects were not required. We discuss these problems in more details in a later section.

url(zucker92performance,
'citeseer.nj.nec.com/zucker92performance.html').
booktitleword(zucker92performance, *'computer'*).
booktitleword(zucker92performance, *'architecture'*).
type(zucker92performance, *'#inproceedings'*).
address(zucker92performance, *'Gold Coast, Australia'*).
titleword(zucker92performance, *'performance'*).
titleword(zucker92performance, *'study'*).
year(zucker92performance, *'1992'*).
author(zucker92performance, *'R. Zucker'*).
author(zucker92performance, *'J. Baer'*).
...

Figure 4: RDF to Prolog Mapping – Second Approach.

Results For the majority of classes Progol failed to find rules which resulted in compression of the data. However, despite these negative results, a number of rules were produced that were very interesting; some of these are presented in Figure 5. Most of the rules are straightforward and encapsulate obvious facts about the class they describe. Rule 2 – "*is an agents paper if the title mentions 'bdi'*" is interesting, because while an active researcher in the agents field would find this obvious (*bdi* being an abbreviation for *beliefs, desires and intentions*), it might be unknown to people outside this field. Rule 2 is thus a piece of general knowledge, which is not only usable in trying to classify new research papers from Citeseer, but could also potentially be re-used outside this experiment. (Edwards, Grimnes, & Preece 2002) provides more discussion of these results.

Agents:
1: **inClass**(A) :- **titleword**(A,agent), **titleword**(A,mobile).
2: **inClass**(A) :- **titleword**(A,bdi).

Artificial Intelligence:
3: **inClass**(A) :- **journal**(A,*'SIAM Journal on Control and Optimization'*).

Databases:
4: **inClass**(A) :- **titleword**(A,warehousing).
5: **inClass**(A) :- **titleword**(A,transactions).

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

Theory:
8: **inClass**(*A*) :- **volume**(*A,18*).

Figure 5: Progol Results – Sample Rules.

Semantic Webs

The main conclusion to be drawn from our initial experiments was that our disappointing results were at least partly due to the poor quality of our data. We therefore identified a set of desirable properties for a Semantic Web dataset upon which to conduct empirical machine learning experiments:

i The data should be from distributed sources. Data from only a single source would not reflect the true nature of the Semantic Web and would be highly artificial.

ii The data should be heterogeneous. This follows from (i). Any methodology that does not take into account the complexities of dealing with the diverse nature of Web data will not scale to realistic problems.

iii The data should have ontological support providing background knowledge. Ontologies are required to allow inference over relations such as subject hierarchies or *related to* links.

iv The data should describe semantic webs, not semantic forests. Although this was not the main flaw of the data in our initial experiments (and we do feel that there is scope for learning from semantic forests), we felt that learning from semantic webs would make for a more interesting problem.

v The data should ideally reflect some real-world problem where machine-learning would be appropriate, for example a user modelling or personalisation task.

As a part of the GraniteNights (Grimnes *et al.* 2003) project mentioned earlier, a small amount of RDF encoded information about public houses, beer types, restaurants and cinemas available in Aberdeen was created manually. However, it was clear that the amount of effort required to generate a sizable dataset was substantial, and would still not satisfy our criteria. While the GraniteNights system is a publically available service[8], and we had hoped that over time logging user interactions could add up to a sizable dataset, there has to date been insufficient traffic.

Acquiring a dataset with the properties outlined above was not easy, and instead we had to rely on the Semantic Web community developing a killer application that would generate sufficient data.

Friend of a Friend

The Friend of a Friend (FOAF)[9] project aims to create a Web of machine-readable home-pages describing people, the links between individuals and the things they create and do. The FOAF ontology is described using the Ontology Web Language (OWL) (McGuinness & van Harmelen 2003). To join the FOAF world all one has to do is generate a FOAF profile describing oneself and publish it on the web. The profile must adhere to the ontology and could either be generated by hand, or more often, by copy, paste and edit of other people's FOAF, or by semi-automated tools such as FOAF-a-matic[10]. Part of an example profile is shown in Figure 6. This example illustrates several important things about FOAF:

- The *foaf:knows* property points to other people known by this person, creating a networked community.
- People in the FOAF world don't need URIs, they are identified through their *foaf:mbox* (or *foaf:mbox_sha1sum*, i.e. the email address). In the FOAF ontology these are identified as *owl:inverseFunctionalProperty*, meaning they uniquely identify a person.
- *foaf:knows* properties do not take the value of the URI of other people's FOAF, instead they point to an anonymous RDF node of type *foaf:Person*, which contains the *foaf:mbox* of the other person. Whether two anonymous nodes represent the same person can then be decided based on the *foaf:mbox* values; merging these nodes is known as "smushing"[12].
- *foaf:mbox_sha1sum* is used to disguise the email-address for privacy reasons. The use of a checksum rather than just omitting the value allows people to confirm that the address they've got actually belongs to a person.
- Other FOAF files are linked through *rdfs:seeAlso*, allowing Semantic Web bots to crawl through FOAF space.
- The *wot:assurance* property at the bottom of the file points to a signature of this file, signed with the person's PGP key, providing a secure way to know who made these statements. This provides a basis for the trust layer of the Semantic Web architecture.

Topology The FOAF project began around 1999, but only gained significant momentum in the past two years, due to the increased awareness of the Semantic Web and the existence of FOAF visualisation tools such as Foafnaut[13]. The

[8] http://www.csd.abdn.ac.uk/research/AgentCities/GNInfo/

[9] http://www.foaf-project.org/

[10] http://www.ldodds.com/foaf/foaf-a-matic.html

[12] http://rdfweb.org/topic/Smushing

[13] http://jibbering.com/foaf/foafnaut.svg

```
<foaf:Person>

 <foaf:mbox rdf:resource="mailto:ggrimnes@csd.abdn.ac.uk" />
 <foaf:name>Gunnar AAstrand Grimnes</foaf:name>
 <foaf:homepage rdf:resource="&csd;/~ggrimnes" />
 <foaf:workplaceHomepage rdf:resource="&csd;"/>
 <foaf:projectHomepage rdf:resource="&csd;/research/agentcities"/>
 <foaf:groupHomepage rdf:resource="&csd;/research/agentsgroup" />
 <foaf:phone rdf:resource="tel:+441224272835" />

 <foaf:depiction rdf:resource="&csd;/~ggrimnes/gfx/me.jpg" />

 <foaf:interest rdf:resource="http://www.w3.org/2001/sw/" />
 <foaf:interest rdf:resource="http://www.agentcities.net" />

 <foaf:made rdf:resource="&csd;/research/AgentCities/GraniteNights" />

 <contact:nearestAirport>
  <airport:Airport rdf:about='http://www.daml.org/cgi-bin/airport?ABZ' />
 </contact:nearestAirport>

 <foaf:knows><foaf:Person>
     <foaf:mbox rdf:resource="mailto:maym@foobar.lu" />
     <rdfs:seeAlso rdf:resource="http://martinmay.net/foaf.rdf"/>
  </foaf:Person></foaf:knows>
 <foaf:knows><foaf:Person>
     <foaf:mbox rdf:resource="mailto:apreece@csd.abdn.ac.uk" />
  </foaf:Person></foaf:knows>
 <foaf:knows><foaf:Person>
     <foaf:mbox rdf:resource="mailto:pedwards@csd.abdn.ac.uk" />
  </foaf:Person></foaf:knows>
 <foaf:knows>
  <foaf:Person foaf:name="Sonja A Schramm">
     <foaf:mbox_sha1sum>
      83276f91273f2900cf0b6657b3708b736276ef81
     </foaf:mbox_sha1sum></foaf:Person>
 </foaf:knows>

 <rdfs:seeAlso rdf:resource="&csd;/~ggrimnes/codepict.rdf" />
 <rdfs:seeAlso rdf:resource="&csd;/research/agentsgroup/foaf.rdf" />

</foaf:Person>

<rdf:Description rdf:about="">
 <wot:assurance rdf:resource="foaf.rdf.asc" />
</rdf:Description>
```

Figure 6: Parts of Example FOAF File.[11]

FOAF co-depiction search facility[14] allows pictures depicting multiple people to be identified, effectively proving that their *foaf:knows* relationship. The co-depiction search allows one to visually document the link from oneself to famous people, for example Bill Clinton or Frank Sinatra, increasing the fun-factor and "instant-gratification" of creating a FOAF profile. For our experiments we used a FOAF crawl from September 2003[15], which contains 9097 nodes of type

[14]http://swordfish.rdfweb.org/discovery/2001/08/codepict/

[15]http://jibbering.com/foaf/dumps/

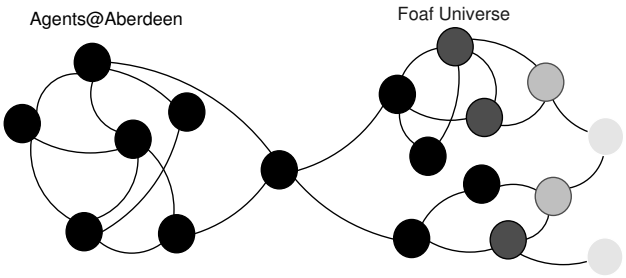

Figure 7: FOAF Group with Narrow Connection to FOAF World.

person. When smushed, this is equivalent to 8908 people, of which 1980 people know at least one other person, i.e. they are not leaf-nodes of the FOAF knows-graph. The data consists of 147527 triples, using 201 different namespaces, and 1066 distinct properties (compared to 49 in the FOAF ontology). Many of these properties are not widely used, only 116 are used more than 100 times.

Within the FOAF graph one can typically identify groups of people that are very "close" in real-life. For example, the people within a single research group. In such a group there are many interconnecting *foaf:knows* links, and the level of detail about each person is similar since their profiles are often generated by copy and pasting one person's FOAF, or because they are all generated by the same person or from a database. A group often has only a very narrow connection to the rest of the FOAF graph, or in certain cases no connection at all. In the Aberdeen FOAF graph most people know each other, but as shown in Figure 7, the only link to the rest of the FOAF network is through a single person node.

Problems While the heterogeneity and distributed nature of FOAF is clearly a good thing and makes a dataset based on FOAF very realistic, it does introduce a number of problems when one attempts to reason with or learn from the data. These are summarised below:

- Human errors. The majority of FOAF content is manually generated using a text editor, causing several types of human error:
  - Simple typing mistakes, i.e. *foaf:knosw*.
  - Using properties with the wrong namespace, e.g. *rdf:seeAlso* vs. *rdfs:seeAlso*.
  - Misunderstanding or misinterpretation of the FOAF ontology, e.g. using *foaf:mbox* with the email address as a literal string as opposed to an *rdf:resource* with a *mailto:* link.
- Weaknesses and/or inconsistencies of the FOAF ontology:
  - *foaf:mbox* vs *foaf:mbox_sha1sum*. Both properties are declared as *owl:inverseFunctionalProperty* as detailed above. However, nowhere is it formally declared that *foaf:mbox_sha1sum* is the Secure Hash Algorithm[16]

[16]http://www.w3.org/PICS/DSig/SHA1_1_0.html

checksum of the *foaf:mbox* property. The intention is of course that a node with a *foaf:mbox_sha1sum* matching the checksum of another's *foaf:mbox* should be smushed together. At the moment this must be hard-coded in an application specific manner.

– No standard way of expressing interest. *foaf:interest* has range *foaf:Document*, and most commonly points to the URL of a page about the concept. Again, the use of literals vs. *rdf:resource* is inconsistent, but the main problem is that people use different URLs for the same concept. For example: http://www.w3.org/RDF/, http://rdfweb.org, http://rdfweb.org/, http://www.rdfweb.org/.

- Level of detail varies greatly. Our initial experiments with learning from FOAF returned several rules based on the presence of an attribute, such as *foaf:groupHomepage*, as opposed to the value of the attribute.

Enriching FOAF The AKT project[17] aims to tackle a number of challenges of knowledge management, and as a show-case has created an ontology for representing academic researchers and their organisations. An RDF dataset conforming to this ontology has been created by "screen-scraping" the Web pages of UK based research institutions. The lack of detail in the FOAF data could be addressed by enriching it using the information available in the AKT RDF repository. However, this would involve further complicating the learning task by including yet another ontology. We therefore decided to map the instances from the AKT ontology to FOAF, as the ontologies have very similar domains; the majority of the mappings were straightforward, such as :

[*akt:has-email-address* ⇒ *foaf:mbox*]

Others were more complicated, for instance:

[*rdf:type akt:Professor-In-Academia* ⇒
(*rdf:type foaf:Person* & *foaf:title 'Professor'*)].

For the sake of re-usability these mappings were represented in OWL using *owl:equivalentProperty* for the trivial mappings and our own RDF mapping of RuleML[18] for the more sophisticated rules. The RDF representation of the two rules given above is shown in Figure 8.

RDF & ILP

Encouraged by our experience with learning from semantic forests using Progol, the next step was to explore the application of ILP to FOAF data. However, simply mapping RDF to Prolog (as in our first experiments) would no longer be sufficient given the complexity of FOAF, so the following improvements were made:

- Namespace handling. Namespaces of properties were converted to prefixes in Prolog, with *namespace* predicates giving the mapping from prefixes to actual namespaces. For example *foaf:mbox* becomes:

  foaf__mbox(A, *'mailto:ggrimnescsd.abdn.ac.uk'*).
  namespace(*'foaf'*, *'http://xmlns.com/foaf/0.1/'*).

[17]http://www.aktors.org

[18]http://www.csd.abdn.ac.uk/
~qhuo/program/generaltool_sources/ruleml.rdfs

```
<rdf:Description rdf:about="&aktonto;has-email-address">
 <owl:equivalentProperty rdf:resource="&foaf;mbox"/>
</rdf:Description>

<rule:Imp>
 <rule:body>
  <rule:And>
   <rule:arg>
    <rule:Atom>
     <rule:atomObjectArg>
       Professor
     </rule:atomObjectArg>
     <rule:rel rdf:resource="&foaf;title"/>
     <rule:atomSubjectArg>
    <rule:Var rdf:about="#p">
     <rule:varName>P</rule:varName>
    </rule:Var>
      </rule:atomSubjectArg>
    </rule:Atom>
   </rule:arg>
   <rule:arg>
    <rule:Atom>
     <rule:atomObjectArg rdf:resource="&foaf;Person"/>
     <rule:rel rdf:resource="&rdf;type"/>
     <rule:atomSubjectArg rdf:nodeID="#p"/>
    </rule:Atom>
   </rule:arg>
  </rule:And>
 </rule:body>
 <rule:head>
  <rule:Atom>
   <rule:atomObjectArg
    rdf:resource="&aktonto;Professor-In-Academia"/>
   <rule:atomSubjectArg rdf:nodeID="#p"/>
   <rule:rel rdf:resource="&rdf;type"/>
  </rule:Atom>
 </rule:head>
</rule:Imp>
```

Figure 8: Ontology Mapping in RDF Excerpts.

- RDF types. For each class in the ontology Prolog rules are created to determine if a resource is a member of the specific class, or any sub-class thereof. This allows RDF types to be mapped to ILP internal types, used for limiting which predicates may be applied to a given resource, reducing the search-space dramatically. Figure 9 contains an example.

- Normalisation and inference over interests. In our initial experiments with the FOAF data we attempted to fix the inconsistent *foaf:interest* problem by "smushing" nodes that represented the same concept, for example *http://rdfweb.org/foaf/* and *http://www.foaf-project.org/*. In addition super/sub-concept links were created between concepts such as *http://www.debian.org/* and *http://www.linux.org/*, and some general nodes that did not appear in the original data, e.g. *#ProgrammingLanguages* were added. Our preliminary experiments demonstrated that the ILP learner did not use these generali-

sations, probably due to the low number of *foaf:interest* links actually appearing in the data. As a result, these extra rules were not included in our experiments.

```
foaf_Person(A):-
  instanceOf(A,'http://xmlns.com/foaf/0.1/Person').
foaf_Document(A):-
  instanceOf(A,'http://xmlns.com/foaf/0.1/Document').

instanceOf(A,B):-rdf_type(A,B).
instanceOf(A,B):-rdf_subClassOf(B,C),instanceOf(A,C).
instanceOf(A,unknown):-nonvar(A).

castAsfoaf_Person(A,A):-foaf_Person(A).

:-modeb(*,foaf_interest(+foaf_Person, -foaf_Document)).
:-modeb(1,castAsfoaf_Person(+resource,-foaf_Person)).
:-modeb(1,castAsResource(+foaf_Person,-resource)).
```

Figure 9: RDF Type Inference in Prolog.

An additional advantage of using ILP with RDF is that converting the learned results back into RDF is trivial, given some way of representing Horn clause rules in RDF, e.g. the Semantic Web Rule Language (SWRL) (Horrocks *et al.* 2003).

Learning from FOAF

For our experiments with FOAF data we moved from Progol to using Aleph (Srinivasan 2001), which has much broader scope for customisation. Before attempting to learn from the FOAF data it was pre-processed by first smushing it, and then removing any duplicate properties resulting from this merger. Aleph was initially configured to use any of the predicates appearing in the input data when constructing a hypothesis, only restricted by the RDF typing as detailed above. However, as there were 1066 predicates in the full dataset, this was too much for Aleph to deal with and we moved to only using a subset based on the most frequently occurring predicates. The 15 most frequent predicates are shown in Figure 10, and the preliminary experiments were done with these, excluding *rdf:type* as it is applied to every person node, and *jibbering:isKnownBy* which is the generated inverse of *foaf:knows*. However, this did not give very good results and we moved to using the 100 most frequent predicates; for space reasons the list is not re-produced here.

Initial exploratory experiments with Aleph highlighted problems with the scale of the FOAF dataset. Even with a very small subset of the full data (less than 10% of the people in the full crawl), the search-space was still far too large, and Aleph was unable to make any generalisations over the data. To reduce the size of the search space the problem was broken into sub-problems by first applying a clustering algorithm and then feeding each cluster to Aleph separately. Such an operation does make sense in the context of FOAF, as there are often clusters of people, reflecting real-life groups, e.g. research groupings, where the group membership may not be explicitly stated. To perform the

Frequency	Property
1244	http://www.w3.org/1999/02/22-rdf-syntax-ns#type
1120	http://jibbering.com/foaf/jim.rdf#isKnownBy
1119	http://xmlns.com/foaf/0.1/knows
908	http://xmlns.com/foaf/0.1/mbox_sha1sum
906	http://xmlns.com/foaf/0.1/name
846	http://www.w3.org/2000/01/rdf-schema#seeAlso
419	http://xmlns.com/foaf/0.1/depiction
392	http://xmlns.com/foaf/0.1/surname
344	http://xmlns.com/foaf/0.1/firstName
327	http://purl.org/dc/elements/1.1/title
273	http://xmlns.com/foaf/0.1/codepiction
266	http://xmlns.com/foaf/0.1/mbox
246	http://xmlns.com/foaf/0.1/nick
236	http://xmlns.com/foaf/0.1/homepage
230	http://purl.org/dc/elements/1.1/description

Figure 10: 15 Most Frequent Predicates.

clustering step a hierarchical agglomerative clustering algorithm (HAC) (Vorhees 1986) has been employed. HAC is a greedy bottom-up clusterer which works by initially creating one cluster for each individual, then repeatedly merging the two closest clusters until there is only one left or some threshold for similarity is reached. Our version of HAC computes the distance between two clusters as the average distance between each of the individuals in the two clusters.

A modified version of Hamming distance (Hamming 1950) is used as similarity metric; modifications were as follows: each RDF property appears as an attribute of the instance, as does each (property, value) pair. All properties and values are treated as nominal, including anonymous nodes. Although there could have been some scope for treating datatyped properties as ordinal, few of the properties used in FOAF are typed, so dealing with extra complexity was unlikely to pay off. The intention behind this similarity metric is that two people that both have a certain property, say *foaf:interest*, are more similar than two people who do not share any attributes, but less similar than two people who have the same value for *foaf:interest*. Initial clustering experiments with this similarity metric were not very successful, manual inspection of the clusters showed that they did not reflect the real-life groupings were well. It was clear that considering only direct attributes of the Person node was flawed. A similarity metric is needed that can take into account the FOAF graph immediately around a person, and her position in the bigger graph, not just the immediate attributes. We are not aware of any work to date on the subject of similarity metrics for RDF data. However, in (Montes-y-Gómez, Gelbukh, & López-López 2000) a similarity metric for conceptual graphs is presented. Conceptual graphs are a data-structure commonly used for natural language processing. They consist of a network of concept nodes, representing entities, attributes, or events and relation nodes between them. A simple conceptual graph representing *John loves Mary* is as follows:

$$[\text{John}] \Leftarrow (\text{subj}) \Leftarrow [\text{love}] \Rightarrow (\text{obj}) \Rightarrow [\text{Mary}]$$

The similarity metric developed is based on the idea of the Dice coefficient (Rasmussen 1992), but incorporating

a combination of two complementary sources of similarity: the *conceptual similarity* and the *relational similarity*, i.e. the overlap of nodes and the overlap of edges within the two graphs. Full details of the similarity metric can be found in (Montes-y-Gómez, Gelbukh, & López-López 2000).

Conceptual graphs and RDF instances are sufficiently similar that the same similarity metric should be appropriate in both cases. The similarity metric is designed to work on separate graphs, and in order to apply it to RDF we had to modify the algorithm to extract a sub-graph around each person. The RDF graph is traversed in either direction from the person node, i.e. triples were considered where the node in question is either the subject or the object. To limit the size of the sub-graph the number of triples traversed is limited. Trial and error showed that the optimal sub-graphs for clustering were obtained if traversal was allowed of two triples forwards and one backwards. Note that these traversals may **not** be in any order, and backward-traversals are only permitted from the initial node. Consider for instance Figure 11; the subgraph for the person "Gunnar Grimnes" would include one backwards traversal, i.e. the "Comp. Sci. Dept" node, but not the "University of Aberdeen" node. It would also include two forward traversals, to both the anonymous node and the literal title node.

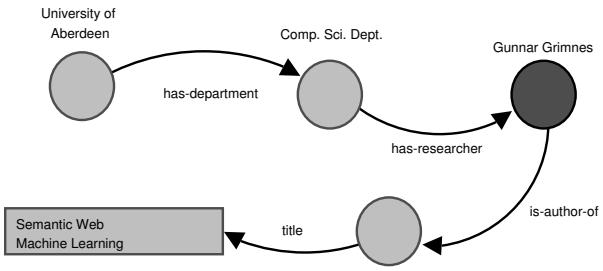

Figure 11: Example of Extracting Person Subgraph.

Clustering with this similarity metric gave acceptable results. For example the algorithm was able to discover clusters of people from different research-groups at Aberdeen. Aleph was applied to the generated clusters (as detailed above), in an attempt to learn a concise description of each cluster.

Preliminary Results

At the time of writing empirical investigation into use of these methods on the FOAF data is incomplete. Some preliminary results follow which illustrate interesting properties of the investigation to date:

- Two research groups from the Computing Science Department were clustered separately: the Agents@Aberdeen and Advanced Knowledge Technologies group. The description of the AKT group at Aberdeen was learned by Aleph as:

  member(A) :- **foaf__groupHomepage**(A, *'http://www.csd.abdn.ac.uk/research/akt/'*).

  However, the description learned for the Agents@Aberdeen group was not as simple:

  member(A) :-
  foaf__knows(B,A), **foaf__firstname**(B,C).
  member(A) :-
  foaf__knows(B,A), **foaf__projectHomepage**(B, *'http://www.csd.abdn.ac.uk/research/conoise'*).

  The Conoise[19] project appearing in the rule above is one active project of the Agents@Aberdeen group. The reason behind the complex nature of this rule appears to be one person who is a member of both the AKT and Agents@Aberdeen research groups. The clustering algorithm used does not allow an instance to be a member of more than one cluster, and this person was therefore clustered into the AKT group, but still had the property *foaf:groupHomePage = 'http://www.csd.abdn.ac.uk/research/agentsgroup'*. A rule defining the Agents group involving this attribute was therefore not a viable option for Aleph, which generated two alternative rules to cover all the instances.

- People were often grouped together based on work they have done, either projects or publications. This is consistent with our design goals for the similarity metric algorithm and subgraph extraction. For example:

  member(A) :-
  dc__creator(B,A), **dc__title**(B,
  'The Pipeline of Enrichment: Supporting Link Creation for Continuous Media').
  Note: dc is the Dublin Core schema.

  This rule is interesting, because the actual publication (Variable B) is sometimes given a URI, and sometimes just referred to as an anonymous node, Aleph must therefore use another clause to identify it. This is analogous to the FOAF use of *foaf:mbox* to identify people. However, in the FOAF case we can pre-process the data and smush the nodes because *foaf:mbox* is declared to be inverse functional in the ontology; this illustrates how background knowledge in an ontology can facilitate learning.

Related work

Improving learning performance by taking advantage of the structure that is inherit in data that is marked up using XML is discussed in (Zaki & Aggarwal 2003). They represent XML documents as ordered and labeled trees and present an algorithm called XMiner to extract the most frequent subtrees for a given class; these are then converted into rules for classifying new instances. The authors demonstrate that their classifier out-performs information retrieval or association rule classifiers when learning from XML data.

Exploiting structure and semantics for learning is also discussed in (Berendt, Hotho, & Stumme 2002), where ontologies are used to enrich plain text and do feature selection and aggregation. The aim being to improve clustering results. The authors also use semantic meta-data about Web pages to perform web-mining; a user's navigational path through a site becomes a path through semantic concepts, which might be more comprehensible than the raw access-log. The paper

[19] http://www.conoise.org/

also includes a brief discussion of applying ILP to Semantic Web data, highlighting the challenge of solving the scalability problems of ILP to make it usable on the Semantic Web.

(Alani *et al.* 2003) uses ontologies to detect Communities of Practice (CoP) that are only implicitly expressed. For instance, two people might not have a direct relation, but they might have written a paper together. The detection is based on analysis of the graph of people and properties and allows weights to be attached to possible relations. For example, it is more significant that two people have written a paper together, than the fact that they subscribe to the same journal. Experiments are performed using the same AKT ontology used for the work described in this paper. In (Middleton *et al.* 2002), the CoP detection mechanism is combined with ontologies to generate an initial user-profile for a hybrid-recommender system called Quickstep. Analysis of a user's publications taken from her homepage are used to determine interest weights for concepts in the ontology, the user is then matched with similar users in the CoP and their combined profiles are in turn matched with the concept weights of research papers to recommend papers of interest, even to new users of the system. Middleton et al. also present experiments comparing ontology supported recommendations to those made without ontological inference. This work is continued further in (Middleton, Shadbolt, & Roure 2004), where a new system called Foxtrot which includes profile visualisation, email notification and user feedback.

The European Elena project[20] aims to demonstrate the feasibility of smart spaces for education, and is using Semantic Web technology to achieve this. In (Dolog *et al.* 2003) RDF meta-data for educational resources is combined with an RDF profile to provide a personalised and adaptive view of a hypermedia learning-space.

Discussion

In this paper we have identified two types of Semantic Web structure – semantic forests and semantic webs, and have explored the application of learning to each of these. We argue that machine-learning on Semantic Web data will be essential because it is unlikely that all details will be explicitly stated. Additionally, the learning-outcome may only be useful in a personalised or context dependent manner and it would not make sense for it to be explicitly stated. In our discussion of learning from the Semantic Web we had emphasised the importance of using a learning algorithm capable of taking symbolic data as input, and also learning a symbolic model. This is essential in order to make the most of the symbolic nature of Semantic Web data, and to facilitate expressing the learned model in a Semantic Web form so that it may be re-used and thus applicable outside the context where it was learned. We believe that re-use is essential in the Semantic Web context. The added value of using RDF in a single disconnected application does not outweigh the added complexity of RDF, and a simple relational database would probably have been simpler to implement, and also more efficient. The true value of RDF only becomes apparent in a distributed context, where the semantic data is shared and the data originator often cannot control (or even imagine) the possible uses her data might have.

We have demonstrated that data pre-processing is critical on the Semantic Web, because of the distributed and heterogeneous nature of the data. We believe inference has an important role on the Semantic Web and learning should not be seen as replacing inference, but rather as building upon it. Many rules and relationships will always be well known and will either already have been encoded in ontologies or can be easily coded by hand; applying learning to raw data to re-discover these is counter-productive. Semantic Web Machine Learning needs to build on top of inference and be designed to take advantage of any background information available.

As part of our exploration we have used different datasets and different algorithms, and have had interesting and promising results to date using the Inductive Logic Programming tool Aleph to learn descriptions of automatically generated clusters from FOAF data. This work continues.

[20] http://www.elena-project.org/

References

Alani, H.; Dasmahapatra, S.; O'Hara, K.; and Shadbolt, N. 2003. Identifying communities of practice through ontology network analysis. In *IEEE IS*. IEEE. 18–25.

Berendt, B.; Hotho, A.; and Stumme, G. 2002. Towards semantic web mining. In *International Semantic Web Conference*, 264 – 278.

Berners-Lee, T.; Hendler, J.; and Lassila, O. 2001. The semantic web. *Scientific American* 28–37.

Cover, T., and Hart, P. 1967. Nearest neighbour pattern classification. *IEEE Transactions on Information Theory* 13:21–27.

Dolog, P.; Henze, N.; Nejdl, W.; and Sintek, M. 2003. Towards the adaptive semantic web. In *1st Workshop on Principles and Practice of Semantic Web Reasoning*, 51–68.

Edwards, P.; Grimnes, G. A.; and Preece, A. 2002. An empirical investigation of learning from the semantic web. In *ECML/PKDD, Semantic Web Mining Workshop*, 71–89.

Grimnes, G. A.; Chalmers, S.; Edwards, P.; and Preece, A. 2003. Granitenights - a multi-agent visit scheduler utilising semantic web technology. In *Seventh International Workshop on Cooperative Information Agents*, 137–151.

Hamming, R. 1950. Error detecting and error correcting codes. *Bell System Techincal Journal* 29:147–160.

Hendler, J., and McGuinness, D. L. 2000. The darpa agent markup language. *IEEE Internet Computing* 67–73.

Horrocks, I.; Patel-Scheider, P.; Boley, H.; Tabet, S.; Groshof, B.; and Dean, M. 2003. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. DARPA DAML Program.

Lassila, O., and Swick, R. R. 1999. Resource description framework (rdf) model and syntax specification. W3c recommendation, World Wide Web Consortium.

McGuinness, D. L., and van Harmelen, F. 2003. Web ontology language (owl): Overview. W3c recommendation, World Wide Web Consortium.

Middleton, S.; Alani, H.; Shadbolt, N.; and De Roure, D. 2002. Exploiting synergy between ontologies and recommender systems. In *11th International WWW Conference, Semantic Web Workshop*, 41–50.

Middleton, S.; Shadbolt, N.; and Roure, D. D. 2004. Ontological user profiling in recommender systems. In *ACM Transactions on Information Systems*, volume 22(1), 54–88.

Mitchell, T. 1997. *Machine Learning*. McGraw-Hill. 154–200.

Montes-y-Gómez, M.; Gelbukh, A.; and López-López, A. 2000. Comparison of conceptual graphs. In *Lecture Notes in Artificial Intelligence*, volume 1793. Springer Verlag. 548–556.

Muggleton, S. 1995. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming* 13(3-4):245–286.

Rasmussen, E. 1992. Clustering algorithms. In Frakes, W., and Baeza-Yates, R., eds., *Information Retrieval: Data structures & Algorithms*. Prentice Hall.

Salton, G., and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management* 24:513–523.

Srinivasan, A. 2001. *The Aleph Manual*. http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/.

Vorhees, E. 1986. Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. In *Information Processing & Management*, volume 22, 465–476.

Zaki, M. J., and Aggarwal, C. C. 2003. Xrules: An effective structural classifier for xml data. In *9th International Conference on Knowledge Discovery and Datamining*, 316–325.