Ontology Based Extraction of RDF Data from the World Wide Web[1]

A Thesis Proposal
Presented to the
Department of Computer Science
Brigham Young University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Tim Chartrand
August 2002

[1]This research is supported by the National Science Foundation grant #IIS-0083127.

1 Introduction

The advent of the World Wide Web (WWW) has taken the availability of information to an unprecedented level. The simplicity of the Web has been a major factor in its proliferation [KM02]. Anyone can easily publish a document about anything or link to anyone's site. The document need not be structured according to any particular format or even contain correct information, and the link need not be valid [BLHL01]. Placing such restrictions on the Web would have increased the level of expertise necessary to create Web content and decreased the amount of useful information available.

The Web's simplicity, however, does have a downside; although the unstructured and unregulated nature of the Web makes publishing information easier, it also makes finding and using information much more difficult. The typical way to find information on the Web today is to do a keyword search, where a search engine tries to guess the meaning of the combination of keywords in the query and the meaning of each page it has indexed to find pages relevant to the query. Even when we can find the desired information, it is usually very difficult for a machine to interpret, so the user generally has to manually review and use the information.

Research is underway in universities and companies around the world to develop the next generation of the Web, called the Semantic Web. The Semantic Web will add meaning, or semantics, to Web content in order to make it easier to find and use for both humans and machines. Adding formal semantics to the Web will aid in everything from resource discovery to the automation of all sorts of tasks [BLHL01][LS99].

The basic data model used to build the Semantic Web is called the Resource Description Framework (RDF). RDF is a domain-independent model for describing resources, where a resource is anything that can be represented by a URI, including Web pages, parts of Web pages, or even physical objects. RDF describes resources by making statements about resources in the form <subject><predicate><object>. The subject is the URI of some resource, the resource being described. The predicate, also represented by a URI, expresses some relationship between the subject and the object. The object is either a literal or another resource represented by a URI.

For example, if we want to say Tim Chartrand is 24 years old, we would write

the statement: <mailto:tim@cs.byu.edu><genealogy#age>"24"[1]. Or to say Tim Chartrand is the father of Tyler Chartrand: <mailto:tim@cs.byu.edu><genealogy#fatherOf> <mailto:tyler@thechartrands.com>. Thus, RDF allows us to remove ambiguity; there may be more than one person with the name Tim Chartrand, but there is exactly one person represented by the URI <mailto:tim@cs.byu.edu>. So some agent, either a human or a program, searching for information about Tim Chartrand would look for statements with <mailto:tim@cs.byu.edu> as the subject or object [LS99][MM02].

RDF helps give structure to Web content, but what kind of resource is `mailto:tim@cs.byu.edu` and how is the `genealogy#fatherOf` predicate related to it? Ontologies (i.e. vocabularies, schemas) that define classes of objects and their properties answer these questions. RDF Schema (RDFS) is a simple ontology language written in RDF that allows the creation of vocabularies with classes, properties, and subclass/superclass hierarchies [BG02]. DAML+OIL is an extension of RDFS that allows finer-grained control of classes and properties with features such as cardinality constraints and inverses of properties [CvHH+01].

As an example of how to use an ontology, suppose we have an ontology called *genealogy* that defines a class *Person* with a property *fatherOf* whose value is a *Person*. We would add to the RDF from the above example a specification that `mailto:tim@cs.byu.edu` and `mailto:tyler@thechartrands.com` both represent objects of type *Person* (i.e. they belong to class *Person*). This tells us that everything we know about a *Person* directly applies to the given resources; if every *Person* has a *Name*, then the object identified by `mailto:tim@cs.byu.edu` must also have one. Further, suppose that our ontology defines a property called *parentOf* as a generalization of *fatherOf* and a property *childOf* as the inverse of *parentOf*. Without any extra work on our part, a fairly general Semantic Web agent would be able to infer that Tyler Chartrand is a child of Tim Chartrand, even though we only stated that Tim Chartrand is the father of Tyler Chartrand.

With all the advantages of the Semantic Web, what keeps it from reaching a critical mass where it will gain widespread acceptance and use? One reason is the newness of the area and the related tools to help people publish and use Semantic Web

[1]The property denoted genealogy#age is shorthand for the URI http://www.thechartrands.com/genealogy#age. The full URI is omitted for readability. The same applies for the genealogy#fatherOf property.

content. Another reason, equally important, and the one that this research will address, is the lack of useful content. For years people have been publishing Web documents on every topic imaginable and building systems to continually generate new content, so there is a vast amount of human readable information on the Web. It is hard to imagine rewriting the current Web content to be accessible to Semantic Web agents. As long as newspapers, for example, continue to generate simple HTML documents for their daily car advertisements or obituaries, Semantic Web agents will have a hard time performing useful tasks with that information.

Given the vast amount of information contained in HTML pages on the Web, a possible method of building the Semantic Web is to automatically structure the existing information semantically to make it accessible to Semantic Web agents. There is a whole field of research called Information Extraction that tries to extract unstructured or semistructured Web content so it can be stored and queried more efficiently. This extraction approach could be extended to extract information as RDF.

The most common approach to extracting information from the Web is to create, either by hand, semiautomatically, or automatically, a wrapper for a Web site that is essentially a grammar for its pages. This wrapper uses extraction rules to describe where each data field is in the page so that a program can extract data from any number of pages as long as they follow the format prescribed by the wrapper [Eik99][LRNdST02].

The BYU Data Extraction Group has taken a different approach to Information Extraction. Instead of describing the page of interest with a wrapper, they describe the application domain of interest with an ontology. A data-extraction ontology is a conceptual model written in an extension of the OSM modeling language [Emb98]. It defines the structure of the desired data by specifying *object sets*, similar to classes, and *relationship sets* among them. It also specifies detailed extraction rules called data frames [Emb80] for each piece of information to be extracted. Given an ontology for a particular domain, the BYU Ontos system can extract from Web pages containing data in that domain even if the pages are from different Web sites [ECJ+99].

Since there are so many human readable Web documents available and so much work has been done to extract information from them, the next step is to leverage that work to automate the creation of the Semantic Web by extracting Semantic Web content

from the World Wide Web. This could be done using existing techniques to extract the data and structure it as RDF with respect to an ontology. Given the prevalence of the use of ontologies in the Semantic Web, the BYU ontology-based approach seems to be a better choice than wrapper-based extraction techniques. Since we are extracting Semantic Web data from documents, our approach has an added benefit; we can keep track of the location where each piece of data is extracted in the original document. Thus, the extracted data can be superimposed over [MD99][BDM02], or be an index into, the original data.

2 Thesis Statement

This research will help bridge the gap between the WWW and the Semantic Web. It will do this by semiautomatically converting RDFS and DAML+OIL ontologies into OSM data-extraction ontologies that will be used to extract relevant data from semi-structured Web pages. This data will then be converted to RDF, making it accessible to Semantic Web agents, including an agent to browse the RDF as an index into the original documents.

3 Methods

We can separate the methods to be used in accomplishing the research for this thesis into four steps. First, we convert a DAML+OIL or RDFS ontology, to which we henceforth refer as a DAML ontology, to a data-extraction ontology. We do this by converting the structure and constraints of a DAML ontology to an OSM model instance, matching object sets to data frames, and then allowing a user to make any necessary additions or corrections. Second, we use the generated ontology to extract data from HTML documents. Third, we restructure the data as RDF conforming to the original ontology. Fourth, we create a simple agent to allow a user to browse the RDF as an index superimposed over the original document.

Before proceeding, we make two important assumptions. First, we assume that the user is interested in RDF data with respect to a particular DAML ontology, and thus the user is responsible for finding or creating an ontology that models the desired data.

Also, the ontology should model *only* the desired data; the extraction engine works best with an ontology that models relatively few concepts [ECJ+99]. Second, we assume that the user is interested in data that appears in multiple-record Web documents and that each record fits well with the provided ontology. If these assumptions do not hold, the extraction process is unlikely to produce useful results.

3.1 Converting DAML Ontologies to Extraction Ontologies

To use a DAML ontology for data extraction, we first convert the ontology to an OSM model instance. Then we extend the model instance to include data frames. Finally, we allow the user to edit data frames and constraints.

3.1.1 Automatic Translation of Ontology Structure and Constraints

This section gives an introduction to the DAML and OSM modeling languages and then presents an algorithm for converting DAML to OSM. As previously mentioned, DAML is a language for creating ontologies, where an ontology is a set of concepts and the relations that exist among them. Like many conceptual modeling languages, DAML allows for the specification of classes of objects (i.e. entity types or object sets) and properties (i.e. associations or relationship sets).

DAML is a property-centric rather than class-centric language; whereas many languages define a class in terms of its properties, DAML defines a property in terms of the classes to which it applies. DAML defines a property in terms of its domain and range, where the domain is the set of classes whose instances can *have* a value for this property, and the range is the set of classes whose instances can *be* the value for this property. A DAML property can also have cardinality restrictions or additional type restrictions.

For example, we can define a *takesCourse* property with a domain of class *Student*, and for the subclass *FullTimeStudent* of *Student*, we can restrict the *takesCourse* property with a minimum cardinality constraint of four, meaning that a *FullTimeStudent* must take at least four courses.

OSM is another conceptual modeling language. The main construct of the language is the object set, which can be lexical (i.e. literal, have values) or non-lexical (i.e. abstract, objects represented by object identifiers). Object sets are connected by relationship sets, where a relationship set has connections to one or more object sets. Cardinality is specified by a participation constraint on the connection of a relationship set to an object set. For example, to say a person has at least one name, a modeler would put a minimum participation constraint of one on the *Person* connection of the *Person has Name* relationship set.

An overview of the algorithm for translating DAML to OSM follows:

1. For each DAML class create a *non-lexical* OSM object set with the same name.
   Example: *Person class* ⇒ *Person non-lexical object set*

2. For each DAML property whose range is a literal, create a *lexical* object set with the same name as the property. Then create a binary relationship set with connections to the object set created for the property's domain and the newly created lexical object set. Name the relationship set *[domain-object-set name] has [range-object-set name].*
   Example: *Name property (with domain of Person)* ⇒ *Name lexical object set and Person has Name relationship set*

3. For each DAML property whose range is another class, create a relationship set with connections to the object sets created for the domain and range of the property. Name the relationship set *[domain-object-set name] [property name] [range-object-set name].*
   Example: *fatherOf property (with domain and range of Person)* ⇒ *Person fatherOf Person relationship set*

4. For each subclass/superclass relationship in the DAML ontology, create a corresponding generalization/specialization in the OSM model instance, preserving union, intersection, and mutual exclusion constraints.
   Example: *Person class defined as disjointUnionOf Man, Woman, and Child classes* ⇒ *generalization/specialization with the Person object set as the generalization and*

the Man, Woman, and Child object sets as specializations, with a partition constraint.

5. For each DAML restriction or cardinality constraint, add the corresponding OSM construct. For some restrictions, we will need to add an OSM general constraint using a predicate calculus statement.

3.1.2 Automatic Addition of Data Frames

Once the structure has been translated, the next step is to turn the OSM model instance into an extraction ontology by adding a data frame for each lexical object set. A data frame contains a recognizer with a detailed description of how a piece of information generally appears in an HTML document. It uses regular expressions to specify how the data itself appears as well as regular expressions for the immediate context of the data. Data frames also include keywords that should appear in the document close to the data to be extracted. For example, a *BirthDate* data frame would have a regular expression to recognize a date and the keywords *birth* and *born* [Emb80].

To associate a data frame with each lexical object set, we draw from a library of common prebuilt data frames. Mapping object sets to data frames becomes essentially a schema matching problem. Researchers have used several different techniques for schema matching, including instance- or schema-based, element- or structure-level, and constraint- or linguistic-based matching[RB01]. Because one of our schemas, the data frame library, has neither instances nor structure, we are limited to element-level linguistic-based matching.

To generate a mapping from object sets to data frames, we compare each lexical object-set name with each data-frame name and generate a similarity measure from zero to one. Then for each object set, we simply choose the data frame with the highest similarity as long as the similarity is above a given threshold. Note that a data frame may be the best match for more than one object set; for example, the *Date* data frame would match both *BirthDate* and *DeathDate*.

To calculate a similarity measure between an object set and a data frame, we use a combination of string matching techniques on their respective names. First, we

apply some standard IR style stemming to get a root for each name. Next, we combine Levenshtein edit distance [Lev65], soundex [HD80], and longest common subsequence to generate a similarity value. To take advantage of the semantics of names as well as their string values in calculating similarity, we introduce an alias field in the data frame which will list possible alternate names. An object-set name can then be compared with each alias of the data frame as well as its name. Some name matching techniques have also used synonyms from a thesaurus. This method, however, introduces the possibility of false positive matches between words with multiple senses [EJX01]. For example, *address* might match with *speech* even though the *address* referred to is a *mailing address*. Therefore, we use only manually specified aliases rather than use synonyms from a general thesaurus.

The output of this process is a suggested mapping from lexical object sets to data frames. This mapping may not be complete; if no data frame has a similarity value above the threshold, we assume that no data frame exists for the object set, and a user must create one or provide a mapping missed by the matching process.

3.1.3 User Ontology Editing

DAML ontologies may not contain cardinality constraints, and the data frame matching process may produce only a partial mapping. Therefore, the generated ontology may not yet be suitable for data extraction without some user intervention.

To allow the user to edit constraints and data frames, we provide an extended version of the Ontology Editor [Hew00]. When a user opens a DAML ontology, the Ontology Editor translates the ontology, matches object sets to data frames, and shows the generated extraction ontology graphically. Since a DAML ontology has no display information for its classes and the Ontology Editor displays the ontology graphically, we use a modified version of the AGLO graph layout algorithm [Col93] to find a suitable screen layout.

After the name matching process completes, the system presents to the user with a list of suggested mappings from lexical object sets to data frames. In the simplest case, where the system finds a match for each object set and the match is correct, the

user can simply accept the suggested matches. If the system does not find a data frame or finds an incorrect data frame for an object set, the user can choose from a list of existing data frames or launch the data-frame editor to create one. In many cases, a generic data frame is suitable for an object set once keywords are added. For example, we can specialize a *Date* data frame to *BirthDate* by adding the keywords *birth* and *born*. Therefore, we provide a convenient way for the user to specialize a data frame by simply entering a list of keywords.

After the translation process, the user can edit the ontology like any other ontology but may not change the structure. The user can modify participation constraints directly and further edit and test data frames using the provided data-frame editor.

The Ontology Editor uses an XML format for its ontology serialization. The current version of the Data Extraction Group's extraction engine, however, uses an older text-based format. Therefore, when the ontology-creation process completes, we will give the user an option to save the ontology in the old format.

3.2 Extracting Data from HTML Pages

With a suitable ontology and a set of HTML pages available, the extraction process proceeds mostly unchanged from the way it originally worked [ECJ+99]. First, the record separator attempts to automatically find an HTML tag that separates the records such that there is one data record between each occurrence of the tag. Then it separates the page into records based on that tag and strips out all markup in the record, leaving only plain text [EJN99].

Next, the extraction engine parses the ontology to generate a relational database schema from the ontology structure and generate matching rules from the data frames. Then it evaluates the matching rules against each of the records and generates SQL insert statements for the data it finds. The end result is a relational database filled with data extracted from the HTML page.

3.3 Converting Extracted Data to RDF

Once the system finishes extracting the data, converting the data to RDF is straightforward. The database will have a main table, where each row represents an object of interest. Therefore, each row becomes an instance of the DAML class corresponding to the object set of the object of interest, and the value for each field in the record becomes the value for the corresponding DAML property.

Since the table structure of the database depends on the participation constraints in the extraction ontology, the extraction engine may split a record over multiple tables. Therefore, we need to refer back to the constraints in the extraction ontology to collect all the data for a record. For example, a *Person* may have an unspecified number of *GivenName*s, so the *GivenName*s would be stored in a separate table.

One complication is the problem of finding URIs for the generated objects. In general, an object does not have an associated URI in the page from which it is extracted, so we need to generate one. The extraction engine helps with this problem; it generates a unique id for each record extracted. We can therefore create a URI with the following form: *[URL]#[ClassName][RecordId]*, where the URL is the address of the original document and the ClassName is the name of the class to which the object belongs. This solution is similar to a proposed approach for publishing relational databases in RDF [BL98]. Observe, however, that although this method of producing URIs from arbitrary data (or any other known method), uniquely identifies an object, the object may have more than one URI. For example, the same object might be described on different Web sites and therefore be given different URIs. This object identity problem will not be addressed in this thesis.

3.4 Viewing the RDF with the Original Document

Once we have the extracted data formatted as RDF, it will be accessible to various kinds of Semantic Web agents. However, a user may want to simply browse the data. To verify the extraction or see the context of the extracted data, a user may also want to refer to the original document while browsing the data. Thus we can think of the structured data as superimposed over, or as an index into, the original document [MD99][BDM02].

We provide an application, a simple agent, that allows the user to browse the RDF data by class. Expanding a class in the browse tree shows its subclasses and instances, and expanding an instance shows its property values. When the user selects a property value for an instance (i.e. an extracted data item), the application highlights the data item in the original document.

3.5 Evaluating the results

Because this work builds directly on the BYU Ontos data-extraction engine, evaluating the results will be not as straightforward as for similar projects. Standard precision and recall measures over the whole process would tend to evaluate the extraction process, which is not in dispute. Therefore, the RDF browser agent will serve another purpose; it will help verify the success of the project. If a user can use the system to process a DAML ontology and a related HTML page, which must both satisfy the stated assumptions, and then browse the extracted RDF with respect to the ontology, the project will be considered successful. For testing and evaluation, we will experiment with several DAML ontologies; we will create about five, and we will use about five existing ontologies from the DAML repository [dam00]. Further, we will produce a Web demo, which will allow users to apply the system to their own ontologies and Web pages.

One part of the project, however, uses heuristics, and we can therefore evaluate it numerically. For the automatic matching of object sets with data frames, we will use standard recall and precision measures [BYRN99] to report the results.

4 Contribution to Computer Science

The main contribution this thesis will make is to the advancement of the Semantic Web. It will apply existing ideas from the fields of information extraction and superimposed information to the fairly new problem of building the Semantic Web from existing data. It will also produce an algorithm for translating DAML ontologies into OSM-based data-extraction ontologies.

5 Delimitations of the Thesis

This thesis will not attempt to do the following:

1. Improve on the data-extraction process.
2. Fully automate the conversion from DAML ontologies to data extraction ontologies. (The user will have to be involved in many cases, especially in the creation of data frames.)
3. Provide an editor for creating DAML ontologies. (These must either preexist or a user must build these by hand or with some other tool.)
4. Convert edited data-extraction ontologies back into DAML ontologies; the conversion is one-way. (We will, however, convert the *extracted data* to conform to the original DAML schema.)

6 Thesis Outline

1. Introduction and Related Work (7-10 pages)
2. Converting DAML Ontologies to Extraction Ontologies (20-25 pages)
   (a) Automatic Translation of Structure and Constraints
   (b) Automatic Addition of Data Frames
   (c) User Ontology Editing
3. Extracting RDF Data (3 pages)
   (a) Extracting Relational Data
   (b) Converting Relational Data to RDF
4. Viewing the RDF (3-4 pages)
5. Analysis and Results (8-10 pages)
6. Conclusions and Future Work (3 pages)

7 Thesis Schedule

Literature Search and Reading	January - October 2002
Chapter 2	November 2002
Chapters 3 & 4	December 2002
Chapters 1, 5, & 6	January 2003
Thesis Revision and Defense	February 2003

8 Bibliography

References

[BB01] A. Barstow and D. Beckett. RDF test cases. W3C working draft, World Wide Web Consortium, Cambridge, MA, USA, September 2001. http://www.w3.org/TR/2001/WD-rdf-testcases-20010912/.

This W3C working draft defines N-Triples, a simplified syntax for RDF. We use it for displaying RDF in this document.

[BDM02] S. Bowers, L. Delcambre, and D. Maier. Superimposed schematics: Introducing E-R structure for in-situ information selections. In *21st International Conference on Conceptual Modeling (ER2002)*, Tampere, Finland, October 2002. Accepted for Publication.

This paper describes a system for introducing a new structured layer of information over unstructured documents using an E-R model. The new superimposed layer contains marks or pointers to the data in the original document so that users can browse the information in either context. This is similar to what we are trying to do with our extracted RDF data.

[BG02] D. Brickley and R. Guha. RDF vocabulary description language 1.0: RDF Schema. W3C working draft, World Wide Web Consortium, Cambridge, MA, USA, April 2002. http://www.w3.org/TR/rdf-schema/.

This W3C working draft defines RDF Schema (RDFS), a simple vocabulary language for RDF. The draft presents a core set of classes such as *Class* and *Resource* and properties such as *domain* and *subClassOf* that define domain specific vocabularies or schemas. RDFS is an extension of (and written in) RDF. We will use RDFS (and DAML+OIL) ontologies as target schemas for extracted RDF Data.

[BL98] T. Berners-Lee. Relational databases on the Semantic Web. Semantic Web Design Issues, World Wide Web Consortium, Cambridge, MA, USA, 1998. http://www.w3.org/DesignIssues/RDB-RDF.html.

This design issue suggests a way to publish relational database content on the Semantic Web. We use a similar system for producing URIs.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, pages 28–31, May 2001. http://www.sciam.com/2001/0501issue/0501berners-lee.html.

This is a Scientific American feature article meant as a high-level introduction to the Semantic Web. It gives examples to illustrate the motivation for the Semantic Web and therefore the motivation for automating the transformation from the WWW.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*, chapter 3: Retrieval Performance Evaluation. Addison Wesley, Menlo Park, California, 1999.

This chapter defines several retrieval evaluation measures, including precision and recall, which we will use in this thesis.

[Col93] M. Coleman. Aesthetics-based graph layout for human consumption. Master's thesis, University of California, Los Angeles, Los Angeles, California, USA, March 1993.

This thesis describes a graph layout algorithm called AGLO. Since we want to edit DAML ontologies in the Ontology Editor, we need to assign screen locations to the object sets to make the ontology human readable. We will use an extended version of the Java implementation of AGLO.

[CvHH+01] D. Connolly, F. van Harmelen, I. Horrocks, P. Patel-Schneider D. L. McGuinness, and L. A. Stein. DAML+OIL (March 2001) reference description. Technical report, DARPA Agent Markup Language Program, December 2001. http://www.daml.org/2001/03/reference.

Darpa Agent Markup Language (DAML) is a Semantic Web modeling language that builds on RDF and RDFS. It provides a richer set of modeling primitives than is found in RDFS, adding finer grained specification of properties and classes. We will use DAML (and RDFS) ontologies as target schemas for extracted RDF Data.

[dam00] The DARPA Agent Markup Language Homepage, 2000. http://www.daml.org.

This is the central resource for information about DAML, including specifications, news, sample ontologies, etc. Aside from the language specification, we will use sample ontologies from the repository for testing.

[ECJ+99] D. Embley, D. Campbell, Y. Jiang, Y. Ng S. Liddle, D. Quass, and R. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data Knowledge Engineering*, 31(3):227–251, 1999.

This paper describes the BYU Ontos approach to ontology-based data extraction from multiple-record Web documents. It explains ontologies and data frames as well as the methods used for extraction.

[Eik99] Line Eikvil. Information extraction from world wide web - a survey. Technical Report 945, Norweigan Computing Center, Oslo, Norway, 1999.

This paper gives an introduction to the field of Information Extraction (IE), including its history and various approaches. It gives a detailed explanation of several systems for wrapper generation, including WEIN, SoftMealy, STALKER, RAPIER, SRV, and WHISK. Finally, it proposes a classification system for the features of different IE systems.

[EJN99] D. Embley, S. Jiang, and Y. Ng. Record-boundary discovery in Web documents. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *Proc. 1999 ACM SIGMOD International Conference on Management of Data*, pages 467–478, Philadelphia, Pennsylvania, USA, June 1999. ACM Press.

This paper describes a method for automatically separating multiple-record Web documents into individual records based on a separator tag. We need to extend this approach so that a document location is saved for each record.

[EJX01] David W. Embley, David Jackman, and Li Xu. Multifaceted exploitation of metadata for attribute match discovery in information integration. In *Proceedings of the International Workshop on Information Integration on the Web (WIIW'01)*, pages 110–117, Rio de Janeiro, Brazil, April 2001.

This paper describes attribute matching using synonyms and hypernyms. We will use a variation of the synonyms approach.

[Emb80] D.W. Embley. Programming with data frames for everyday data items. In *Proceedings of the 1980 National Computer Conference*, pages 301–305, Anaheim, California, May 1980.

This paper introduces the concept of data frames. The BYU data extraction engine uses part of a data frame for a data recognizer.

[Emb98] D. Embley. *Object Database Development: Concepts and Principles*. Addison-Wesley, Reading, Massachusetts, 1998.

This book defines the OSM modeling language, which is the basis for the BYU Ontos method of data extraction.

[HD80] P. Hall and G. Dowling. Approximate string matching. *ACM Computing Surveys*, 12(4):381–402, 1980.

This paper introduces the soundex algorithm for approximate string matching. We use soundex to match object sets to data frames.

[Hew00] K. Hewett. An integrated ontology development environment for data extraction. Master's thesis, Brigham Young University, Provo, Utah, USA, April 2000.

This thesis describes a Java-based graphical ontology editor for OSM data extraction ontologies. The implementation of the proposed research will build directly onto this code so it can use the graphical data-frame editor.

[Hom02] Home Page and Extraction Demo for BYU Data Extraction Group, 2002. http://www.deg.byu.edu.

This Web site contains the BYU DEG data-extraction demo as well as various papers and proposals on the subject. We will put the demo we develop on this site.

[KKEP01] J. Kahan, M. Koivunen, and R. Swick E. Prud'Hommeaux. Annotea: An open RDF infrastructure for shared Web annotations. In E. Hyvonen, editor, *The Tenth International World Wide Web Conference. Conference Proceedings*, Hong Kong, China, May 2001. Published on line: http://www10.org/cdrom/papers/488/index.html.

This paper describes a system for annotating Web pages and parts of Web pages. The system stores Annotations separately

from the documents with pointers from the annotation to the document instead of the other way around. This may be a good way to store the extracted RDF data. Also, this system uses XPointers to specify the object of the annotation; that may be a good way to link extracted data to the place from which it was extracted.

[KM02] M. Koivunen and E. Miller. W3C Semantic Web activity. In E. Hyvonen, editor, *Semantic Web Kick-Off in Finland*, pages 27–44, Helsinki, Finland, May 2002. Helsinki Institute for Information Technology, HIIT Publications.

This paper gives an overview of and motivation for the Semantic Web. It lists six main principles to follow in designing the Semantic Web and explains the Semantic Web layers.

[Lev65] V. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, 1:8–17, 1965.

This paper introduces the well-known Levenshtein edit distance, which we will use to match object sets to data frames.

[LRNdST02] A. Laender, B. Ribeiro-Neto, A. da Silva, and J. Teixeira. A brief survey of Web data extraction tools. *ACM Sigmod Record*, 31(2):84–93, June 2002.

This paper gives an overview of approaches and tools to data extraction from the Web. It also proposes a taxonomy for grouping those approaches. It includes in its survey the BYU-ontology approach, which we use in our approach.

[LS99] O. Lassila and R. Swick. Resource description framework (RDF) model and syntax specification. W3C Recommendation REC-rdf-syntax-19990222, World Wide Web Consortium, Cambridge, MA, USA, February 1999. http://www.w3.org/TR/REC-rdf-syntax/.

This W3C Recommendation is the normative definition of the Resource Description Framework, the base language of the Semantic Web. It describes the data model, including resources, properties, and statements. It also defines the basic and abbreviated syntaxes for XML serialization.

[MD99] D. Maier and L. Delcambre. Superimposed information for the Internet. In *SIGMOD Workshop on the Web and Databases (WebDB) (Informal Proceedings)*, pages 1–9, Philadelphia, Pennsylvania, USA, June 1999.

This paper represents early work in exploring the types and uses of superimposed information. The paper proposes a bi-level browser where the user can browse the superimposed information and the base information together.

[MM02] F. Manola and E. Miller. RDF primer. W3C working draft, World Wide Web Consortium, Cambridge, MA, USA, April 2002. http://www.w3.org/TR/rdf-primer/.

This W3C Working Draft provides an entry point to understanding RDF with its data model and different syntaxes. It is a non-normative companion to the various documents that define RDF and RDFS. This document is a good place to start learning about RDF and the Semantic Web.

[Pal02] S. Palmer. RDF in HTML: Approaches. http://infomesh.net/2002/rdfinhtml/, May 2002.

There is currently no standard way of associating RDF data with an HTML document. In this article, the author discusses the pros and cons of different approaches. We may use one of these approaches (the embedding approach) for associating extracted RDF with the document from which it is extracted.

[RB01] E. Raham and P. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):344–350, 2001.

This survey paper sets forth a taxonomy of schema-matching approaches including schema- vs. instance-based, element- vs. schema-level, and constraint- vs. linguistic-based approaches. Of particular interest is the description of element-level linguistic-based methods, which we use to match object sets to data frames.

9 Artifacts

In addition to the written thesis, we will produce a demo implementation. It will be an extension of the current Data Extraction Web Demo found on the BYU Data Extraction Group's Web site [Hom02]. We will replace the ontology page of the demo with the Ontology Editor, extended with facilities for converting from DAML ontologies and saving them in the format currently recognized by the extraction engine. We will also create an application to allow the user to browse the extracted RDF in one frame while referring to the extracted data from the original document in another. The Programming languages we will use are Java, JavaScript, PHP, and C++.

This thesis proposal by Tim Chartrand is accepted in its present form by the Department of Computer Science of Brigham Young University as satisfying the thesis proposal requirement for the degree of Master of Science.

Date	David W. Embley, Committee Chair
	Stephen W. Liddle, Committee Member
	Kent E. Seamons, Committee Member
	David W. Embley, Graduate Coordinator