Automating the Extraction of Domain Specific Information from the Web—a Case Study for the Genealogical Domain

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

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

Troy Walker
January 2004

This research is supported by the National Science Foundation under grant #IIS-0083127.

1 Introduction

The past decade has seen an explosion in the use of the Web to share information. An increase in the use of the Web parallels an increase in data available, especially in areas where collaboration is helpful, such as genealogy. Thousands of amateur genealogists have posted the results of their research online. Organizations including governments, companies and churches make their databases available for querying. All together, there are probably over a million pages containing genealogical data on the Web in addition to countless pages generated from database queries.

It is clear that when looking for specific data on the Web the challenge is not scarcity; it is locating the data we want. Search engines help with this problem [ACG+01]. A search engine takes an information retrieval approach—it presents each Web page as a result depending on its relevance to a query. Since search engines do retrieval on a whole-document basis, the user must manually load each page returned and read it for the precise data wanted, or find that the document is irrelevant. For example, a search on Google for "Walker genealogy" returns over 200,000 results. Although a few of the pages are advertisements for genealogical services and are therefore irrelevant to the query, most of these results are relevant and represent a wealth of information. Unfortunately, the amount of time necessary to visit all these sites is discouraging: Assuming a human could read each of these pages in one minute; it would take over four months to read all this information.

Rather than requiring a human to sift through this mountain of data, it would be helpful to have a software agent that could extract the desired information automatically. A software agent would be able to process many pages each second, reducing the search process to less than one-hundredth of the time it takes now. There are, however, many obstacles to this approach. To begin with, a software agent must be able to deal with the format of each page and translate it to its own representation.

The Web presents information in a variety of formats. Even within the standard HTML format, information is presented in many ways. HTML documents can be classified by the way they present data.

- Single-Record Document: contains information on one object.
- Multiple-Record Document: contains many objects laid out as a list or chart.
- Table Document: has information laid out in rows and columns where each row (column) is an object and columns (rows) group attributes.
- Form: has fields that must be populated to retrieve pages containing information possibly in one of the formats above.

As humans view these pages, they effortlessly adapt to these and other presentation formats. Computers need the ability to detect the format and the capability to extract the information from each of these formats.

Web sites have different conceptual models of what genealogical information is. As humans, we adapt to these varying schemas and translate them to our own. One site may list birth date, birth state, and birth city, while another lists birth with a date and location. Any system to gather information from different sources much deal with this schema-mapping problem.

Currently, the most prevalent approach to extraction of data from the Web is using page-specific wrappers. Since these wrappers extract data from specific Web pages, they are sensitive to changes in the formatting. These wrappers are tedious to write and must be written for each new page as well as every time a page changes significantly. Because of this, researchers have recently focused on the semi-automatic generation of wrappers [LRN+02, KMA+98, RS97, SA99]. There are at least 39 commercial and non-commercial wrapper generators in existence [KT02]. Even with semi-automatically generated wrappers, the use of wrappers requires a substantial amount of work to produce and maintain.

The Data Extraction Group (DEG) at Brigham Young University has developed techniques to handle the problems of varying formats and schema without hand crafting wrappers [ECJ+98, EJN99, EJX01, ETL02]. To date, however, no system integrates these techniques. The DEG bases its work on ontologies, or conceptual models, to define what data to extract. Other researchers make use of ontologies as well [dRC+98, SMN01], but they have not developed their ideas as far as the DEG. In the DEG approach, we specify the data to extract by data frames that include regular expressions and possible keywords [Emb80]. Ontologies also express the relationship between the pieces of data extracted, using participation constraints to specify the minimum, average, and maximum number of relationships in which an object can participate. The ontology for a domain of interest consists of a primary object set, other object sets (possibly indirectly) related to it, and the relationship sets that meaningfully connect the object sets. In the genealogical domain, for instance, we are interested in extracting person objects. Each person has exactly one name and gender and a part in many events such as birth, marriage, and death. These events have attributes such as date and location. We extract data that matches the regular expressions found in the data frames and that conforms to the participation constraints. This approach has the advantage of being resilient to changes in page format and is capable of extracting from new pages with no human intervention. This makes it perfect for the genealogy application domain where pages come from a variety of authors, pages may be constantly updated, and new pages are continually brought online.

2 Thesis Statement

We propose a system to answer genealogical queries called GeneTIQS (Genealogical Target-based Integration Query System). GeneTIQS will automatically extract data from the Web and will be scalable and resilient to changes in the underlying documents. We will build GeneTIQS by integrating and expanding techniques already developed by students in the Data Extraction Group. Because GeneTIQS will be modular and based on an ontological definition of genealogical data, it will be easily adaptable for use in other domains.

3 Methods

GeneTIQS's operations will be straightforward. Figure 1 shows an overview of the system. The user will present GeneTIQS with a *User Query* through a query interface, which is built automatically given a domain *Ontology*. Each subsequent module will have access to this *User Query* and the underlying *Ontology*. The *URL Selector* will choose from a preexisting *URL List* those that are likely to contain the requested information. The *Document Retriever and Structure Recognizer* will retrieve the documents from the Web and examine each to determine which data extraction engine should process the document. One data extraction engine will process the document and extract the raw data in the form of attribute/value pairs. These engines will also have a mechanism for adding URLs to the *Document Retriever and Structure Recognizer's* queue in order to handle additional pages encountered while processing. The *Data Constrainer* will select from these pairs and assemble complete objects that conform to the participation constraints in the *Ontology*. The *Result Filter* will select those objects that match the query and, finally, the *Result Presenter* will gather these objects and present them to the user.

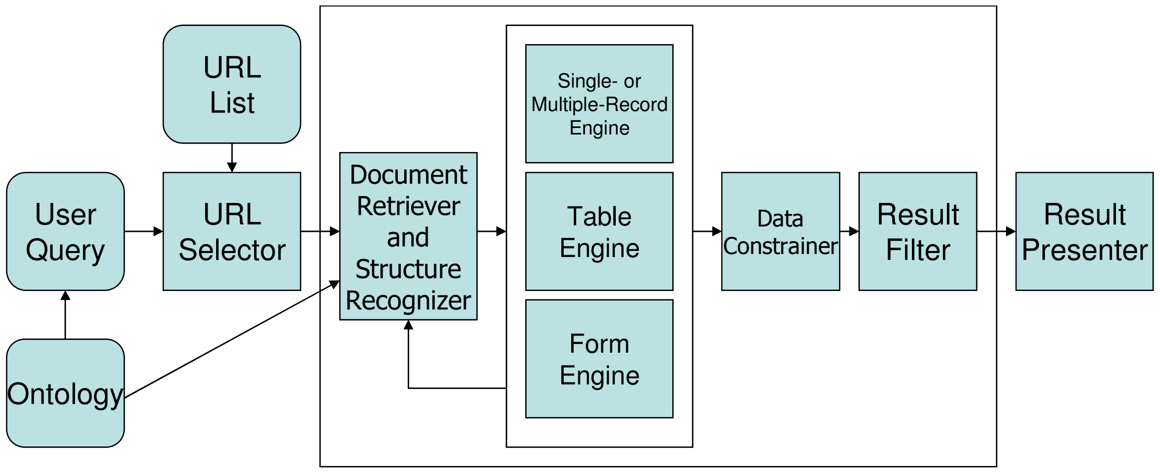

Figure 1: System overview.

3.1 User Query

The user will specify a query through an HTML form. We will generate the form automatically to allow entry of search terms for all the lexical object sets in the ontology. Starting with the relationship sets branching from the root object set, the form generator will create fields based on the participation constraints. For attributes that occur at most one time (such as a person's gender), the form will contain a single-entry field. For attributes that may occur multiple times (such as events), the form will contain multiple-entry fields. For each attribute encountered that also has attributes, the form will contain fields for the attributes related to it nested underneath. When the user completes and submits a form, the browser will submit the query for processing.

3.2 URL List and URL Selector

In order to find information pertaining to a user query, GeneTIQS will use a list of URLs applicable to the genealogy domain and a method for selecting the documents at the URLs likely related to the current query. We will use a list rather than a search engine to avoid the problem of determining the relevancy of documents to the application domain. To reduce the time spent retrieving documents, the list will contain a summary of the content of the page at each URL that the URL selector will use to choose documents likely to be relevant to the query. We will obtain this content summary automatically by extracting the data found at that site and creating summary information containing either ranges or specific values for some of the lexical object sets of the ontology.

When presented with a query, the URL selector will use this information to determine which documents to process. This is a necessary step in keeping the system scalable. With a large URL list, a significant amount of time will be spent retrieving documents from the Web. Determining which documents cannot contain relevant information will dramatically decrease query time.

Once the URL selector has created a list of documents relevant to a query, GeneTIQS will contact each document's server to download the content[1] and extract the data from it. Since these steps are independent for each document, this process is highly parallelizable. We may make use of DOGMA [JCS98] to allow potentially hundreds of PCs to help the main server process the documents. The variables inherent in distributed computation present opportunities for experimentation.

---

[1] We will use proper robot etiquette when crawling web sites.

3.3 Document Retriever and Structure Recognizer

At this point, the document retriever will download the document from the host site and pass it on for processing. As mentioned in the introduction, there are many ways to format information in a Web page. The document structure recognizer will determine which data extraction engine to use to process each document by requesting that each data extraction engine calculate its confidence that it should process a document. The document structure recognizer will select the engine with the highest confidence.

3.4 Data Extraction Engines

A data extraction engine is a processor that extracts attribute/value pairs from a document using knowledge inferred from the format of the document. We will encapsulate each engine as an engine with a standard API, each having the user query and the ontology to assist in extraction. These engines will extract data from single-record and multiple-record documents, tables and forms. The set of engines may also be expanded to extract data in other formats.

3.4.1 Single-Record and Multiple-Record Document Engine

A record is a block of text containing information on an object of interest as declared in the ontology (e.g. a person). Each record may contain simple formatting or links to more data on other pages. Pages that contain one such record are single-record documents while pages containing many records are multiple-record documents.

The DEG has already implemented an engine for extracting information from multiple-record documents [ECJ+99, EJN99]. Although the approach works well for obituaries, car ads, and many other domains, it does not work well with pages depicting family trees that are prevalent in the genealogical domain. Another disadvantage is that it does not handle single-record documents well. This is because it only divides records within one section of the document and it finds this section by looking for the section with the highest branching factor. We will implement an improved engine using this approach and we will incorporate vector space model (VSM) techniques to locate and refine the records [EX97]. Rather than relying on maximum fan-out to locate the subtree containing the objects of interest as in [EJN99], we will start at the root of the tree and work toward the leaves, exploring branches that have high VSM scores and pruning those with low scores or folding them into other subtrees. Since this strategy does not rely on finding the subtree with the maximum fan-out, it will handle single-record documents as well as find data located in multiple subtrees.

Since the majority of genealogy sites created by individuals fit in this category, this component must work well. The variety of locations and formats currently make these sites the most difficult to search. This system must be able to search and reliably extract from a large percentage of these sites in order to make a substantial contribution. This capability will rely in large part on the enhancement of this component. We expect good performance with genealogy to carry over into simpler domains.

3.4.2 Table Engine

Tables in web document present enough challenges and opportunities to the technique above to justify separate handling. Web designers often present data in tables containing one record per row with attribute descriptions in a header or first row. Attributes are separate from values, which sometimes makes it difficult for the single- or multiple-record Engine to reconcile the data with the ontology.

The DEG has made progress in extracting information from tables [ETL02]. The challenge of matching columns to attributes in the ontology is akin to the schema-matching problem. The current implementation uses regular expression to match data in each column with expected values for ontology object sets. By using both attribute matches and data matches, the table engine can resolve the schema-matching problem with high accuracy [ETL02].

This assumes, however, that the table has rows of data that each contains data for the object of interest in the ontology. The implemented engine thus detects valid domain-applicable tables with a variety of heuristics including table size and ratio of recognized values to recognized keywords. Since web authors often use tables today as a tool in Web page layout, the engine must (and can) differentiate between tables containing data and these 'false' tables used for layout [Tao03].

For this thesis, we will encapsulate the functionality of this implemented table engine in our larger system.

3.4.3 Form Engine

The form engine will attempt to map the ontology scheme to the scheme of a form [LES+02, Chen02]. The engine can then populate the form with values obtained from the query and create a URL that simulates the user submission of that form. This engine might require multiple submissions of the form to retrieve all data specified in the query. Once the form engine has generated the URL(s), it will submit them back to the document retriever. Depending on the format of the documents retrieved, one of the engines above can then process the results. For this thesis, we will encapsulate the capability of this already implemented system in our form engine. We will detect valid forms based on the success of attempting to match each form with the ontology.

3.4.4 Other Engines

This system will be easily expandable with engine to integrate data from other sources. XML is one possibility, although genealogies in XML format are not yet common on the Web. Other possible sources are relational databases. In both of these cases, integration is achievable with schema matching [XE03]. GeneTIQS will allow easy addition of these and other engine as they are developed.

3.5 Result Constrainer

Once the data extraction engine has gathered attribute/value pairs from a Web page, we must map this data to the ontology schema while preserving constraints. The DEG has already implemented an engine, Ontos, employing a number of heuristics to do so [ECJ+99]. We will integrate this engine into the system and make improvements to its heuristics.

The current heuristics fail when dealing with some conflicts when two data frames match the same piece of data. When the document does not contain the keywords or keywords are just not given, Ontos selects the matches according to the order of object sets within the ontology. We will alter the current heuristics to ensure that execution order does not play a part in match selection. Instead, we will select matches that minimize the distance between related objects in the document.

To accomplish this, we plan to assign each potential match a cost based on the distance from the match to the nearest keyword. If there is no keyword match in the record, a high value (the length of the document) will be used. Once we assign these values, the system can evaluate the effect of accepting different combinations of matches by scoring each possible combination. GeneTIQS will accept the combination with the lowest cost. The system reduces the search space by always accepting the match closest to the keyword when no conflict exists.

3.6 Result Filter

Once we have results that conform to the constraints of the ontology, the result filter will select those that conform to the user's query and send them to the main server for presentation.

3.7 Result Presenter

The result presenter will collect the results from all of the distributed machines and present the results to the user as a Web page. It will format the output as a forest of trees with the primary object (in this case, person) as the root of each tree. Each tree will contain the data found in one record on the Web (the attributes of one person). This data will be output as an XML file with schema created from the ontology [EM01, EM03] along with an XML style sheet to guide the browser in formatting the output. Results will be presented as soon as they are found. In addition, control will be given to the user to stop processing of a query at any time.

3.8 Evaluation of Results

We desire that this system will answer user queries quickly and accurately. During development, we will use a small list of URLs for tuning. Once GeneTIQS is complete, we will test its scalability, precision and recall, and adaptability with a large URL list.

One of the most important goals of GeneTIQS is the speed of a search. We will measure the speed of our system by giving it queries on a large URL list. Our goal is to minimize the time required to do a query.

Answering queries quickly does not matter if the answers are incorrect, so we will analyze the precision and recall of our system. Since recall is difficult to determine on a large data set, we will select a small list of URLs (distinct from the tuning list) on which to evaluate the precision and recall.

We will also test the adaptability of GeneTIQS by doing an application domain different from genealogy. We can, for example, use the DEG's cars ontology and test its precision and recall as outlined above.

4 Contribution to Computer Science

This thesis will provide a working system on which to query the Web for genealogical data. This system will be easy to alter for other domains. The completed system will be an example of the practical use of extraction techniques developed by the DEG. Future researchers can use this system to develop and test their contributions.

5 Delimitations of the Thesis

This thesis will not attempt to:

- Provide a theoretically complete query interface capable of accepting arbitrarily complex queries.
- Automatically fill the URL list (this includes crawling for relevant pages as well as creation of sophisticated content filters).
- Monitor pages in the URL list to adjust filters based on changes in page content.
- Extract data in formats other than semi-structured HTML documents, tables, or forms.
- Attempt to solve the data integration problem. (There will be no attempt to identify and merge semantically identical records.)

6 Thesis Outline

1. Introduction and Related Work (7-8 pages)
2. Methodology (15-20 pages)
   - 2.1. User Query
   - 2.2. URL List and URL Selector
   - 2.3. Document Retriever and Structure Recognizer
   - 2.4. Data Extraction Engine
     - 2.4.1. Single-Record and Multiple-Record Document Engine
     - 2.4.2. Table Engine
     - 2.4.3. Form Engine
   - 2.5. Data Constrainer
   - 2.6. Result Filter
   - 2.7. Result Presenter
3. Results and Analysis (8-10 pages)
4. Conclusion and Future Work (4 pages)

7 Thesis Schedule

Literature Search and Reading	December 2002 – September 2003
Chapter 2	November 2003 – January 2003
2.1, 2.3, 2.4.1, 2.4.2	November 2003
2.4.3	December 2003
2.2, 2.5-2.7	January 2003
Chapter 3	February 2003
Chapters 1 and 4	March 2003

8 Bibliography

[ACG+01] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. Searching the Web. *ACM Transactions on Internet Technology,* 1(1):2-43, August 2002.

This paper describes basic search engine technology. It explains the use of search engines and the how the components overcome the challenge of searching the Web.

[Che02] X. Chen. *Query Rewriting for Extracting Data Behind HTML Form,* Thesis Proposal, Brigham Young University, Provo, Utah, August 2002.

The paper proposes a technique for extracting data behind forms by rewriting a user query. This work is to be incorporated in this project.

[DEG] BYU Data Extraction Research Group Home Page: http://www.deg.byu.edu

This Web page hosts the DEG's papers and Web demos. When the proposed system is completed, we will place a demo of its feature son this site.

[DMOZ] Open Directory RDF Dump: http://rdf.dmoz.org

This page links to XML documents containing the dmoz Open Directory. This data will be used to build a list of genealogy pages.

[dRC+98] M. De Rosa, T. Catarci, L. Iocchi, D. Nardi, G. Santucci. Materializing the Web. In *Proceedings of the Third IFCIS International Conference on Cooperative Information Systems (CoopIS '98)*, pages 24-31, New York, August 1998.

This paper proposes a system to extract information from the Web as defined by ontologies modeled as extended ER diagrams. They materialize all data locally and run queries on local data.

[ECJ+98] D.W. Embley, D.M. Campbell, Y.S. Jiang, Y.-K. Ng, R.D. Smith, S.W. Liddle, and D.W. Quass. A Conceptual-Modeling Approach to Extracting Data from the Web. In *Proceedings of the 17th International Conference on Conceptual Modeling (ER'98)*, pages 78-91, Singapore, November 1998.

This paper introduces the ontological approach used by the DEG for data extraction. This approach is unique in its resiliency to changes in page format.

[ECJ+99] D.W. Embley, E.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-Model-Based Data Extraction from Multiple-Record Web Documents. *Data and Knowledge Engineering,* 31(3): 227-251, November 1999.

This paper describes the system based on ontologies for extracting data from multiple-record Web documents. This system takes one page in HTML format, infers record separations and employs various heuristics to match extracted data with attributes specified in the ontology.

[EJN99] D.W. Embley, Y.S. Jiang, and W.-K. Ng. Record-Boundary Discovery in Web Documents. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 467-478, Philadelphia, Pennsylvania, June 1999.

This paper introduces the approach used in extracting data from multiple-record HTML documents. More specifically, it evaluates different methods of detecting record separating tags and their use in combination.

[EJX01] D. W. Embley, D. Jackman, and L. 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 suggests the use of a combination of metadata clues to make attribute-matching decisions.

[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 data frames concept, which the DEG extraction engine uses to identify values for attributes.

[EM01] D. Embley and W. Mok. Developing XML with Guaranteed 'Good' Properties. In *Proceedings of the 20th International Conference on Conceptual Modeling (ER2001),* pages 426-441, Yokohama, Japan, November 2001.

This paper gives the algorithm we will use for creating the XML schema from the ontology model.

[EM03] D. Embley and W. Mok. On Guaranteeing 'Good' Properties for XML. In *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics* (SCI 2003), pages 195-199, Orlando, Florida, July 2003.

This paper refines the algorithm above to create schema using the nesting properties of XML.

[ETL02] D. W. Embley, C. Tao, and S. W. Liddle. Automatically Extracting Ontologically Specified Data from HTML Tables with Unknown Structure, In *Proceedings of the 21st International Conference on Conceptual Modeling (ER2002)*, pages 322-337, Tampere, Finland, October 2002.

This paper presents the work that we will integrate for extracting data from Web pages with tables. The approach is to infer a column's meaning from the data in that column and pairing that meaning with the data as the records are output row by row. This work will be incorporated into this project.

[EX00] D. W. Embley, and Li Xu. Locating and Reconfiguring Records in Unstructured Multiple-Record Web Documents, In *Proceedings of the Fifth International Workshop on the Web and Databases(WebDB 2000),* pages 123-128, Dallas, Texas, May 2000.

This paper describes a technique using VSM for separating records and adjusting the results to get the maximum information which we will use in this project to improve record separation.

[JCS98] G. Judd, M. Clement, and Q. Snell. DOGMA: Distributed Object Group Management Architecture. *Concurrency: Practice and Experience*, 10(11- 13), September 1998.

This paper outlines DOGMA, the system we propose to use to manage distributed document retrieval and extraction.

[KMA+98] C. Knoblock, S. Minton, J. Ambite, P. Ashish, I. Muslea, A. Philpot, and S. Tejada. Modeling Web Sources for Information Integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI),* pages 211-218, Madison, Wisconsin, July 1998.

This paper discusses Ariadne, a system for extracting and integrating data from Web pages. It allows the user to specify a query answered from a set of Web pages using wrappers to extract the content.

[KT02] S. Kuhlins, and R. Tredwell. Toolkits for Generating Wrappers—A Survey of Software Toolkits for Automated Data Extraction from Websites. In Mehmet Aksit, Mira Mezini, Rainer Unland (Eds.): *Objects, Components, Architectures, Services, and Applications for a Networked World*, International Conference NetObjectDays (NODe 2002), pages 184-198, Erfurt, Germany, October 2002.

This paper gives an overview 16 non-commercial wrapper generators and 23 commercial ones. It goes into depth describing the authors' LAPIS system.

[LES+02] S. Liddle, D. Embley, D. Scott, and S. Yao. Extracting Data Behind Web Forms. *Proceedings of the Workshop on Conceptual Modeling Approaches for e-Business*, pages 38-49, Tampere, Finland, October, 2002.

This paper outlines an approach that attempts to retrieve all data behind a Web form. It uses statistical approaches to sample the input space in an attempt to speed the search. This approach does not handle text input fields.

[LRN+02] A. Laender, B. Ribeiro-Neto, A. Silva, and J. Teixeira. A Brief Survey of Web Data Extraction Tools. *SIGMOD Record*, 31(2):84-93, June 2002.

This paper gives an overview of various wrapper generation approaches including the BYU ontological approach. It groups these approaches based on similar methods with Ontos in a class by itself because of its ontological foundations.

[RS97] M. Roth, and P. Schwarz. Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources. In *Proceedings of International Conference on Very Large Data Bases (VLDB '97)*, pages 266-275, Athens, Greece, August 1997.

This paper introduces Garlic, a wrapper system for reusing information stored in legacy sources. They apply their technique to draw information from ten sources including databases and searchable Web sites.

[SA99] A. Sahuguet, and F. Azavant. Web Ecology: Recycling HTML Pages as XML Documents Using W4F. In *Proceedings of the ACM International Workshop on the Web and Databases (WebDB'99)*, pages 31-36, Philadelphia, Pennsylvania, June 1999.

This paper discusses the W4F wrapper generator as a tool for converting HTML into XML data. It states that over 80% of Web documents are database generated.

[SMN01] H. Snoussi, L. Magnin, J.Y. Nie. Heterogeneous Web Data Extraction Using Ontology. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 99-110, Montreal, Canada, May 2001.

This paper describes a prototype of a system to create wrappers that map HTML data to an ontology. The generation of mappings is completely manual but is assisted by the GUI the researchers developed. They argue that even though these wrappers are sensitive to any format change of the page, this is a rare occurrence so the system is still useful despite this sensitivity.

[Tao03] C. Tao. *Schema Matching and Data Extraction over HTML Tables*, Masters Thesis, Brigham Young University, Provo, Utah, September 2003.

This thesis outlines the system for extracting data from tables we will encapsulate in our system.

[XE03] L. Xu and D. Embley. Using Domain Ontologies to Discover Direct and Indirect Matches for Schema. *Proceedings of Semantic Integration Workshop,* Sanibel Island, Florida, October 2003.

This paper discusses schema mapping and its role in data integration. It shows how an extension to relational algebra allows queries to be performed on heterogeneous sources in a way that is scalable and maintainable.

9 Artifacts

In addition to the written thesis, we will produce a working implementation of this system. It will take a user query and return the results found by extracting data from Web sites selected from its database. We will place this system on the DEG Web site [DEG] as a demonstration of our techniques. We will develop in Java using PHP and JavaScript for the Web interface.

10 Signatures

This thesis proposal by Troy Walker 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 Masters of Science.

Date	David W. Embley, Committee Chair
	Dan R. Olsen, Committee Member
	Robert P. Burton, Committee Member
	David W. Embley, Graduate Coordinator

Appendix A: Criteria for Evaluation of Component Completion (in order of priority)

Component	**Criteria Description**
Single-/Multiple-Record Engine	Accurately determines records for single- and multiple- record documents.
Ontology	Includes basic objects and relationships pertaining to genealogical data and data frames as required for extraction.
User Query	Correctly generates form allowing submission of queries formed by the conjunction of arbitrary fields.
Result Presenter	Shows results as soon as they are found. Allows user to stop processing at any time.
Result Filter	Returns correct results according to query. Must not discard any applicable data or include irrelevant data.
Data Constrainer	Returns correct results based on ontology as well as previous implementation.
URL List/Selector	Has large and small lists needed for evaluation. Selects URLs based on simple filters.
Table Engine	Includes functionality of existing engine.
Form Engine	Includes functionality of existing engine. Results processed by other engines as needed.