Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 2 (1): 10-22, 2008
ISSN 1819-4311

A Bespoked Secure Framework for an Ontology-Based Data-Extraction System

J. Indumathi and Dr. G.V. Uma
Department of Computer Science and Engineering, College of Engineering,
Anna University, Chennai 600 025, Tamilnadu, India

Abstract: In this Bespoked Secure Framework for an Ontology-Based Data-Extraction System study, we report on the implementation of existing generalized framework with alternate technology. Implementation is done using Natural language processing instead of heuristic based method. Heuristic methods are based on assumptions. The assumptions are just unspecified and as a consequence not understood. If for a given secure data extraction limitation problem, the realization of model-based solutions appears to be too complicated or too pricey to carry out. Heuristic approaches need to be incorporated with a meticulous analysis designed at checking the extent to which the approach formalizes rational agency preference structures and/or data user behaviors. Our Secure Data Extraction system will allow new algorithms and ideas to be incorporated into a Data extraction system. Extraction of information from semi-structured or unstructured documents, such as web pages, is a useful yet complex task. Ontologies can achieve a high degree of accuracy and Privacy in Data extraction system while maintaining resiliency in the face of document changes. Ontologies do not, however, diminish the complexity of a Data-extraction system. As research in the field progress, the need for a modular Data-extraction system that decouples the associated processes continues to grow.

Key words: Ontology, data extraction, natural language processing, semi-structured or unstructured documents, web pages

INTRODUCTION

Making sense of vast amount of information available on the World Wide Web has become an increasingly important and lucrative endeavour. The success of web search companies such as Google demonstrates the vitality of this industry. Yet the web search has much still to deliver. While traditional search engines can locate and retrieve documents of interest, they lack the capacity to make sense of the information those documents contain (Alan Wessman *et al*., 2005).

Data extraction addresses many of the problems associated with typical web searches based on standard information retrieval techniques. Data extraction is the activity of locating values of interest within electronic textual document and mapping those values to a target conceptual schema (Laender *et al*., 2002). The conceptual schema may be as simple slots in a template (a wrapper) used to locate relevant Data within a web page, or it may be as complex as a large domain ontology that defines hierarchies of concepts and intricate relationships between those concepts. The conceptual schema is usually linked to a storage structure such as an XML file or a physical Database model to permit users to query the extracted data. In this way the meaning of a document is detected, captured and made available to user queries or independent software programs.

Much of the research in data extraction has aimed at developing more accurate wrappers while requiring less human intervention in the process (Hammer *et al*., 1997; Kushmerick *et al*., 1997;

Corresponding Author: J. Indumathi, Department of Computer Science and Engineering, College of Engineering, Anna University, Chennai-600 025, Tamilnadu, India

Ashish and Knoblock, 1997; Crescenzi *et al*., 2001; Laender *et al*., 2002). The primary drawback of wrappers, whether they are generated manually or semi automatically, is that they depend on the particular syntax of the mark up to detect boundaries between relevant and irrelevant data. The main implication is that when a site's mark up changes, the corresponding wrappers often break.

The generalized framework (Alan Wessman *et al*., 2005) has focused on the use of richer and more formal conceptual schemas (ontologies) to improve accuracy in Data extraction. This system modifies the generalized framework with use of NLP in order to improve the performance. Because ontology describes a subject domain rather than a document, ontology-based data-extraction systems are resilient to changes in how source documents are formatted and they can handle documents from various sources without impairing the accuracy of the data extraction.

This system will accepts multi-record HTML documents, determines record boundaries within those documents and extracts the data from each record. It generates SQL DDL statements for the model structure and stores the extracted information as DML statements. This facilitates the querying of the results but also removes certain meta data attached to the data during the extraction process.

OSMX ONTOLOGY CONSTRUCTIONS

For ontology construction our system relies on OSMX ontology editor (Liddle *et al*., 2000). Ontology editor is a predominantly WYSIWYG tool. OSMX is the ontology language which is derived from the existing OSML (Object oriented System Model Language) (Liddle *et al*., 2000) plus XML schema.

We use the Java Architecture for XML binding (JAXB) to generate java classes and interfaces that represent OSMX constructs from the OSMX specification. Modifying the OSMX definition is generally a straightforward process: we adjust the definition in the XML schema document and then execute a JAXB program that rebuilds the classes and interfaces automatically.

OSMX's ORM (Object relationship Model) with its data frame can serve as an Ontology language. In ORM concepts are represented by object sets, which group values (objects) that have similar characteristics and connections between concepts are expressed via relationship sets, which group object tuples (relationships) that share common structure. Generalization-specialization is a special type of relation that expresses is-a relationships between object sets.

Data frame describe the characteristics of objects, similar to an abstract data type (Embley, 1980). Data frames are attached to object sets and provide a means to recognize lexical values that correspond to objects in the ontology (Fig. 1). The following diagram explains the structure of data frames.

ObjectSet
|
*--DataFrame (0 or 1)
|--defaultCanonicalizationMethod (attribute)
|--InternalRepresentation <TypeSpecification> (exactly 1)
|--ValuePhrase (0 or more)
| |--hint (attribute)
| |--confidenceTag (attribute)
| |--caseSensitive (boolean attribute)
| |--canonicalizationMethod (attribute)
| |--ValueExpression <DataFrameExpression> (0 or 1)
| |--ExceptionExpression <DataFrameExpression> (0 or 1)
| |--LeftContextExpression <DataFrameExpression> (0 or 1)
| |--RightContextExpression <DataFrameExpression> (0 or 1)

```
| |--RequiredContextExpression <DataFrameExpression> (0 or 1)
| |--SubFromExpression <DataFrameExpression> (0 or 1)
| |--SubToExpression <DataFrameExpression> (0 or 1)
| *--<Keyword Phrase> (0 or more)
|
|--<KeywordPhrase> (0 or more)
*--Method (zero or more)
|--hint (attribute)
|--name (attribute; required)
|--language (attribute; default is 'java')
|--ReturnType <TypeSpecification> (0 or 1)
|--Parameter (0 or more, ordered)
| |--name (attribute; required)
| |--decorator (attribute) <--- e.g., 'OUT', 'BYVAL', 'const', etc.
| |--Type <TypeSpecification> (exactly 1)
| *--<KeywordPhrase> (0 or more)
*--<KeywordPhrase> (0 or more)
TypeSpecification (NOTE: children are mutually exclusive)
|--ObjectSetReference
| |
| *--name (attribute; required) <--- ref to an object set
*--DataType
|
|--typeName (attribute; required)
*--unitOfMeasure (attribute)
KeywordPhrase
|--hint (attribute)
|--confidenceTag (attribute)
|--caseSensitive (attribute)
*--KeywordExpression <DataFrameExpression> (exactly 1)
DataFrameExpression
|--color (attribute) <--- only for graphical tool use
|--ExpressionText (exactly 1)
| |
| *--<element content> <--- contains the extended regular expression
*--MatchedText (0 or more)
|--document (attribute)
|--location (attribute)
|--startPos (attribute)
|--endPos (attribute)
|--status (attribute)
*--<element content> <--- contains a matched text string
```

The most basic element of an extraction rule for a data frame is a matching expression. This an augmented forms of a Perl-5 compatible regular expression. The level of regular expression support is defined by the java regular expression package java.util.regex. We augment regular expression by allowing the rule designer to embed macro and lexicon references within the expression itself.

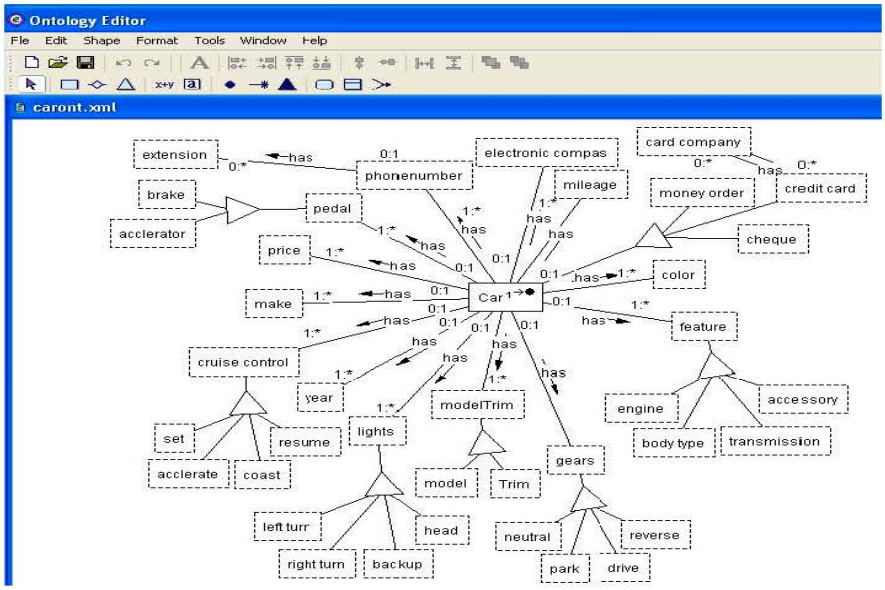

Fig. 1: Car-advertisements application ontology

SYSTEM ARCHITECTURE OF THE DATA EXTRACTION SYSTEM

This part explains the overall architecture and functionality of the system. A graphical overview of the system is described as in Fig. 2 and 5. The user is expected to initiate the system by giving the input files names.

The first four stages, document retriever, document recognizer, document parser and content filter together called as document pre-processing.

Document Retriever

The document retriever is responsible for supplying input document to the system. The current system will retrieve the document as shown in Fig. 3 from the local file system. This accepts the URI as the input and produces a set of documents.

Document Recognizer

The document recognizer analyzes the input document to determine which available document parser is best suited to decompose the document. This module is needed when we giving the different types of file as inputs. The current system accepts the HTML and plaintext sources as shown in Fig. 4.

Document Parser

The document parser breaks up the documents into sub-documents in order to make extraction easier. It divides the multi-record document into sub-documents, each constituting an individual record, allows us to process one record at a time without having to worry about missing a record boundary and extracting values from an adjacent record as shown in Fig. 4.

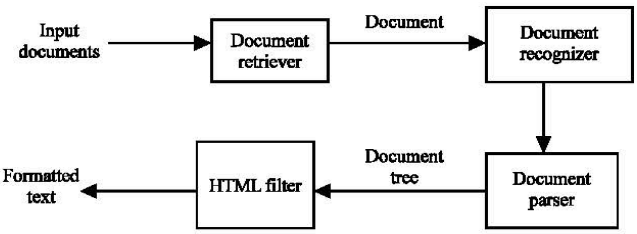

Fig. 2: System architecture of the document preprocessing system

Fig. 3: Document retriever output

Fig. 4: HTML document parser

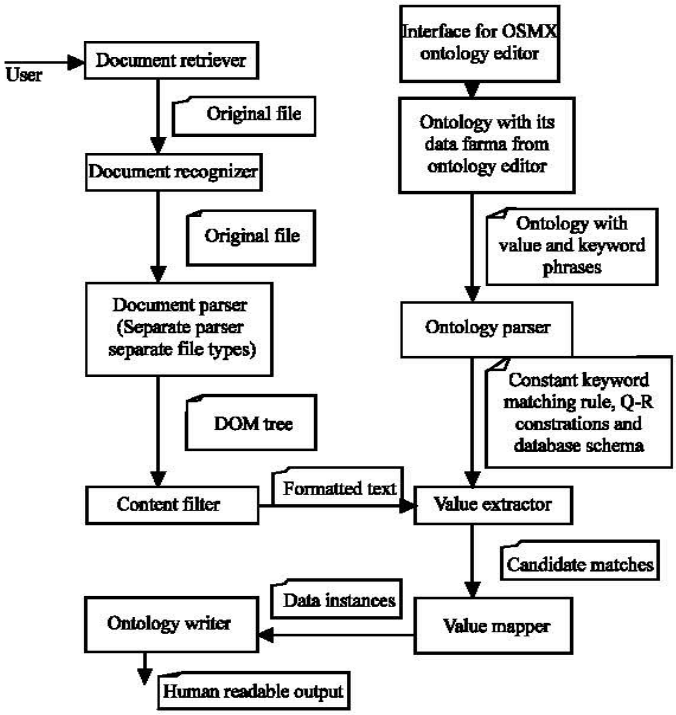

Fig. 5: System architecture of the data extraction system (part 1)

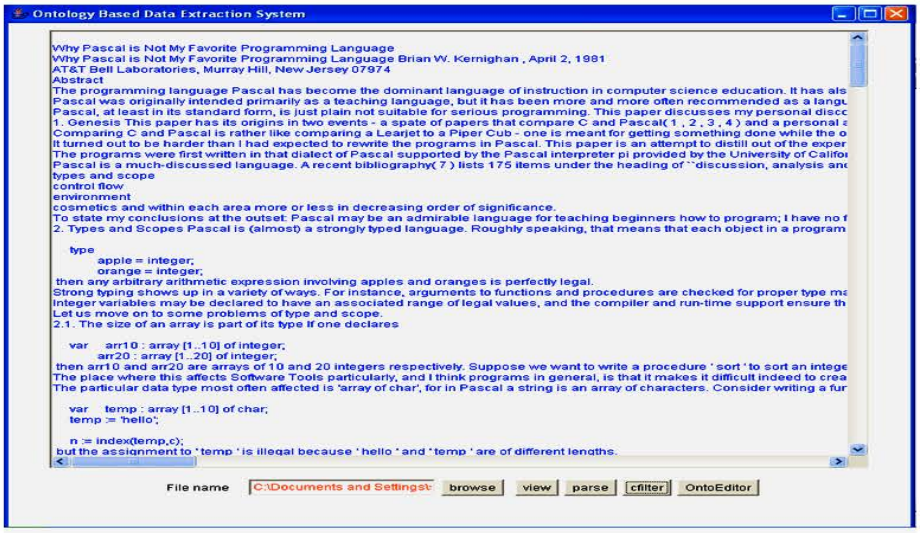

Fig. 6: Content filter output

Content Filter

Content filter eliminates all tag information from the HTML documents since these tags do lend additional meanings. It is convenient for extraction the elimination of tags as shown in Fig. 6.

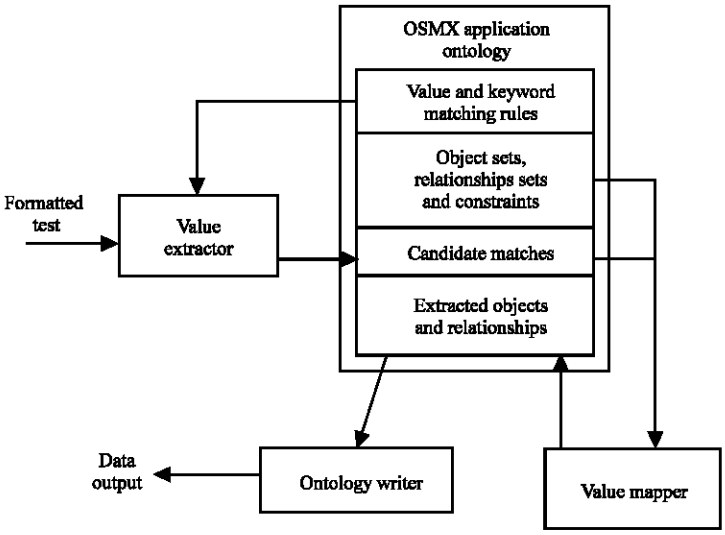

Fig. 7: System architecture of the data extraction system (part 2)

Ontology Parser

Ontology parser produces the set of constant-keyword matching rules, object relationship constraints and a set of database schema by parsing the input ontology. The above mentioned processes are only the preliminary process for extraction. The following steps initiate and extract the information from input documents.

Value Extractor

The responsibility of value extractor as shown in Fig. 7 is to apply the value recognition rules produced by the ontology parser to the input document and producing the set of candidate matches. The conflicts among the candidate matches do not resolved in this stage. The one more functionality of value extractor is to maintain the location information for each candidate values. This provides a traceable back to the document content and also can supply the useful data for algorithms that resolve match conflicts and create mappings from candidate values to elements to the ontology.

Value Mapper

The most important and difficult part of the extraction system is the process that takes the candidate matches and uses them to build a data instance consistent with the constraints specified by the ontology. The value mapper module does this job. The generalized framework implements the value mapper by using the heuristics approach. This system implements the value module by using the Natural Language Processing. The important tasks of value mapper module to transform the candidate matches into data instances are (1) resolve conflicting claims that different elements of the ontology make upon the same matched value, (2) transform lexical values into objects (instances of concepts defined in the ontology), (3) infer the existence of objects that have no direct lexical representation in the text and (4) infer relationships between objects. Finally a collection of objects and relationships between those objects are obtained.

Ontology Writer (Object Relationship Writer)

The ontology writer will converts the data instance into suitable for storage. Also it displays the result as a human readable HTML format.

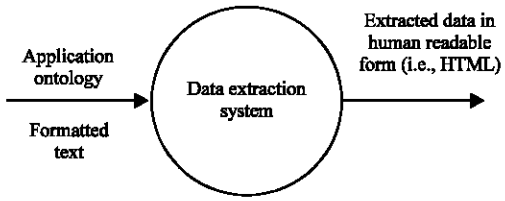

Fig. 8: Context diagram

Information Flow

The flow of information as shown in Fig. 8 between the components of the system is given in the following subsections.

IMPLEMENTATION

The labeling technique described above has been implemented and experiments have been conducted with web-based document articles selected from several domains. All documents used in these experiments have HTML-based file format. Samples consisting of 20 documents from four domains (car-ads, genealogy, computer job ads. obituary) downloaded from various online news papers. The current framework has been implemented by using java 1.5. To convert ontology as a java class file the system used JAXB (Java Architecture for XML Binding). The accuracy of result is same as that of generalized framework.

The Data frames for the application ontology are defined by using OSMX ontology editor. The ontology parser module is implemented by the use of JAXB (Java Architecture for XML binding) facility available with java. It parses the input ontology and extracts the constraints defined by the ontology. The Value extractor implements the function which produces the set of candidate matches by applying the value recognition rules associated with the extraction ontology to the input document. It also implements the function which maintains the location information for each candidate values. The value mapper module finalizes the extraction process. The ontology writer function implements the function which converts the extraction result into a human readable format. This project is implemented and tested for the following application ontology: real estate, restaurant, cell phone plan, digital camera, camp ground and person.

Pertinent Processes

Data Entities

The data entities in the flow diagram are Input file (formatted text),application ontology with its data frame, human readable output.

Data Frame Definition

The data frame defines the extraction rules through its value phrase and keyword phrase constructs. The data frame's regular expression is compatible with Perl-5 regular expression.

Ontology Parser

This process combines application ontology and its data frames and produce constant keyword matches rule, objects, relationships and constraints and database schema.

Value Extractor

This process uses OSMX with its data frames as the ontology language. OSMX will be formed by OSML with XML specification. Locates the recognition rules specified by each

data frame and applies them to the input text. The extractor identifies all substrings in the text that match the recognition rules and for each substring construct a matched text object.

Value Mapper

This process is most important and difficult part of the extraction system and it takes candidate value matches and uses them to build a data instance consistent with the constraints specified by the ontology.

Ontology Writer

This process provides a human readable hierarchical list of objects and relationships in each data instance stored with the input ontology. The output format is HTML, which suits our present purpose.

Process Description

Process 1: Data Frame Definition

Input data entities

Application ontology.

Algorithm for process.

I. Start

II. For each object set of ontology define value phrase and keyword phrase (value phrase consists the following: required context, left context, right context, exception, substitute from/to)

III. Repeat step-II until all the object sets have data frames

IV. Stop

Affected entities

Input application ontology

Process 2: Ontology Parser

Input data entities

Application ontology with its data frames.

Algorithm for process.

I. Start

II. Identify constant keyword matching rules

III. Identify the objects, relationships and constraints

IV. Identify the database schema for storage

V. Stop.

Affected entities

Input application with its data frame

Process 3: Value Extractor

Input data entities

Formatted text and application ontology with its data frames.

Algorithm for process

I. Start

II. Apply the value recognition rules associates with the extraction ontology to the input document, producing a set of candidate matches.

III. Write the routine to maintain location information for each candidate matches (this provide a traceable path back to the document content and also supply the useful data for the algorithms that resolve match conflicts and create mappings from candidate values to elements of the ontology).

IV. Stop

Affected entities

Process 4: Value Mapper

Input data entities

Set of candidate matches

Algorithm for process

- Generate an object for the object set (but do not add it to the data instance yet).
- If object set is lexical:
  - Find the best value phrase match for this object set within the current matching context.
  - If a value phrase is found in the matching context:
    - Store information about the match (such as the matched text and location) with the object.
    - Add the object to the data instance and return it.
  - Otherwise, discard the generated object and return null.
- Determine which object sets to process next.
  - Find all relationship set connections for this object set, except those already visited for this object (prevents infinite cycle).
  - Determine an order for processing the connections.
- Process the child objects sets.
  - Invoke the appropriate heuristic (singleton, functional group, nested group, nonfunctional), based on the nature of the relationship set and the participation constraint on the connection, to obtain one or more relationships.
  - If relationships were generated, bind the object to each relationship.
- If object is non-lexical:
  - If not bound to any child relationships, discard it.
  - Otherwise add it to the data instance.
- Return the generated object if it was not discarded.

Affected entities

Candidate matches.

Process 5: Ontology Writer

Input data entities

Objects and relationships returned by the value mapper.

Algorithm for process

I. Start

II. Convert the objects and relationships into corresponding database schema.

III. Store the database schema in database

IV. Display the output as HTML file.

V. End

Affected entities

Data instances

The following Fig. 9 shows the screen shot of the extraction framework for the user interface.

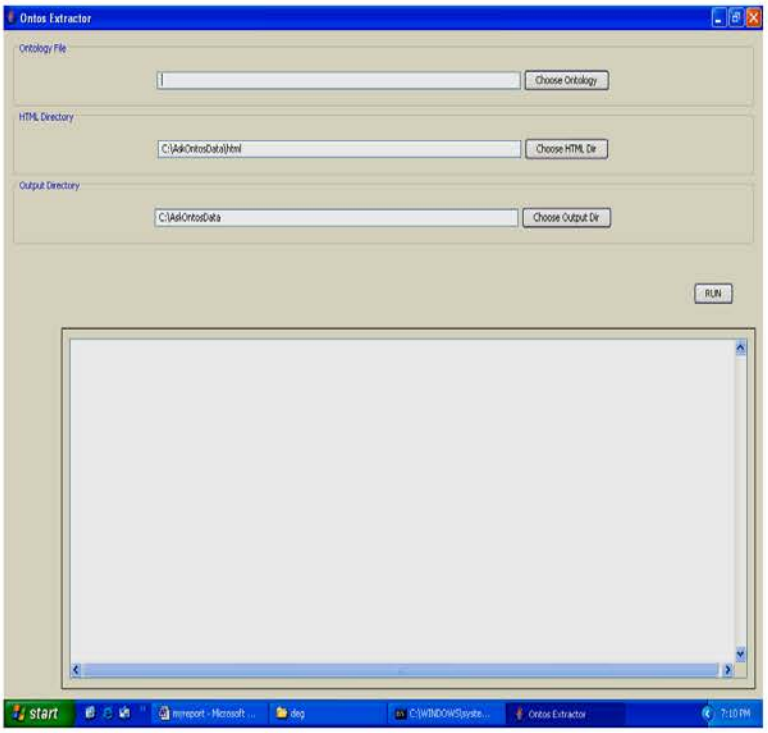

Fig. 9: User interface

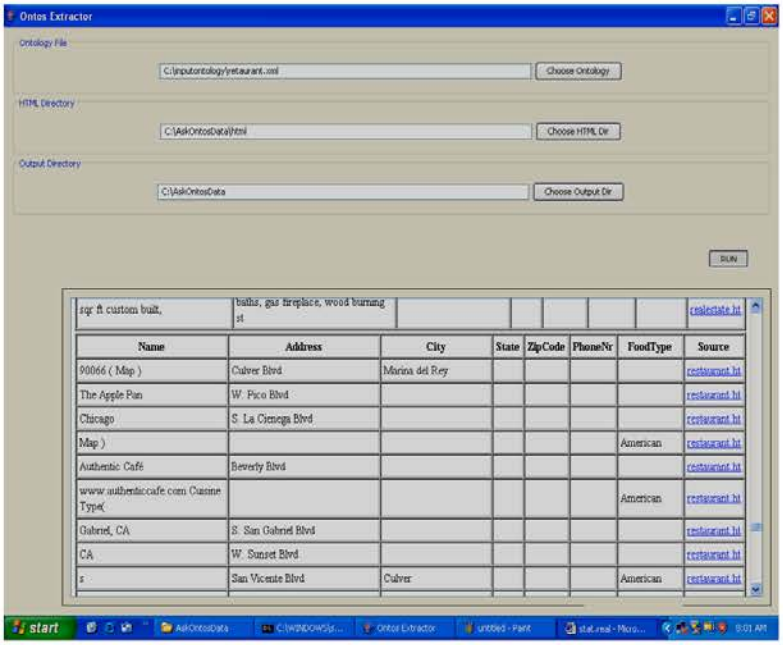

Fig. 10: Output-restaurant domain

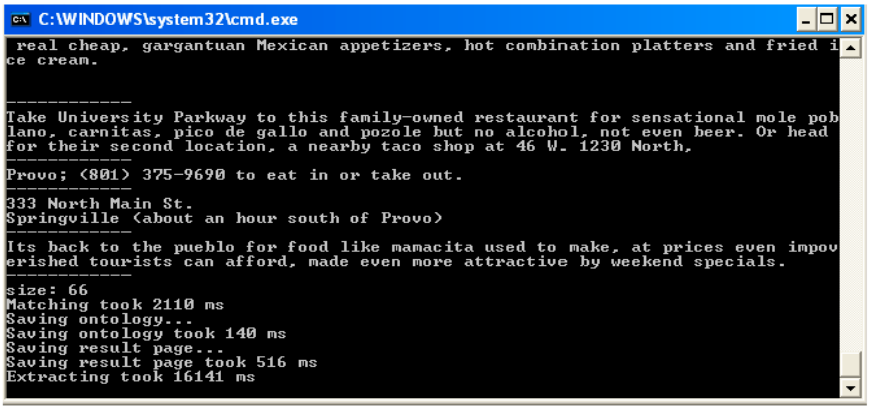

Fig. 11: Result statistics of restaurant domain

The developed system was tested with the following domains: Real Estate, Digital Camera, Restaurant, Cell Phone Plan and genealogy. At the end of the execution the system will produce a resultant OSMX document which can be used for reasoning purpose in future.

Figure 10 shows the output screen shots of the Restaurant domain whereas Fig. 11 shows the resulting statistics of Restaurant domain.

CONCLUSION

We have designed and implemented a modified framework for an ontology-based Data-extraction. The replacement of existing technique by our technique has toiled our efforts. The generalized framework is less flexible for module replacement. The modified framework result is also same as that of generalized framework. This work provides a solid basis for continued research on ontology-based Data-extraction. In this project, the solution was determined for extraction of text information from HTML document and plaintext document. The input documents are retrieved from the local system. The proposed solution will not consider the image, audio and video files. For small scale input the developed system is more efficient than the search engine. For large scale input the accuracy will be high but it will take more time than search engine.

FUTURE ENHANCEMENTS

In future, one might write a Document Retriever that invokes the Google APIs to take advantage of the search engine's existing capabilities for retrieval from the Internet and may develop the solution for retrieving the information from acrobat reader and other kinds of documents by using this ontology-based framework. The current system will process only the text information and accepts the HTML/plaintext document as its input. In future the current system may be extended to process the image, audio and video information. The current system may be extended to accepts the other types of documents, such as PDF as its input. In our further research we plan to extend our works and then implement them.

ACKNOWLEDGMENTS

I would like to acknowledge the overwhelming involvement and incessant hours invested in me by my guide, advisor, Prof. G.V. Uma. Without her, I would not still be close to being

done. For my parents, no words can suffice. I would also like to be grateful to Mr. P. Jeyakumar who saved me asphyxiation due to lack of social life.

REFERENCES

Alan Wessman, S.W. Liddle and D.W. Embley, 2005. A generalized framework for an ontology-based data-extraction. In: The Proceedings of the International Conference On Information Systems Technology and its Application, 63: 239-253.

Ashish, N. and C. Knoblock, 1997. Wrapper generation for semi-structured Internet sources. SIGMOD Rec., 26 (4): 8-15.

Crescenzi, V., G. Mecca and P. Merialdo, 2001. RoadRunner: Towards automatic data extraction from large Web sites. In: Proceedings of the 27th International Conference on Very Large Data Bases, Rome, Italy, pp: 109-118.

Embley, D.W., 1980. Programming with data frames for everyday data items. AFIPS '80 Proceedings, Anaheim, California, pp: 301-305.

Hammer, J., H. Garcia-Molina, S. Nestorov, R. Yerneni, M. Breunig and V. Vassalos, 1997. Template-based wrappers in the Tsimmis system. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, pp: 532-535.

Kushmerick, N., D. Weld and R. Doorenbos, 1997. Wrapper induction for information extraction. In: Proceedings of the International Joint Conference on Artificial Intelligence, Nagoya, Japan, pp: 729-737.

Laender, A.H.F., B.A. Ribeiro-Neto, A.S. da Silva and J.S. Teixeira, 2002. A brief survey of Web data extraction tools. SIGMOD Rec., 31 (2): 84-93.

Liddle, S.W., D.W. Embley and S.N. Woodfield, 2000. An Active, Object-Oriented, Model-Equivalent Programming Language. Advances in Object-Oriented Data Modeling, MIT Press, pp: 333-361.