

UNDERGRADUATE REPORT

Information Extraction Tool

by Alex Lo

Advisor: S.K. Gupta, Edward Yi-tzer Lin

UG 2001-1



ISR develops, applies and teaches advanced methodologies of design and analysis to solve complex, hierarchical, heterogeneous and dynamic problems of engineering technology and systems for industry and government.

ISR is a permanent institute of the University of Maryland, within the Glenn L. Martin Institute of Technology/A. James Clark School of Engineering. It is a National Science Foundation Engineering Research Center.

Web site <http://www.isr.umd.edu>

Information Extraction Tool Final Report

Undergraduate Research Report
Research Experience for Undergraduates – Summer 2001
Sponsored by the Institute for Systems Research at the University of
Maryland and the National Science Foundation

8/3/01

Alex Lo
loaj@rose-hulman.edu

Advisors:
Dr. Lin
Dr. Gupta

Table of Contents

ABSTRACT	3
INTRODUCTION.....	3
RELATED WORK	4
APPROACH	5
1. DATA MODEL DEVELOPMENT.....	5
2. EXTRACTION MODEL DEVELOPMENT.....	9
1. <i>A Priori Knowledge</i>	9
2. <i>Extraction Process</i>	9
SUMMARY.....	10
FUTURE WORK	10
ACKNOWLEDGEMENTS.....	10
REFERENCES.....	11
APPENDIX A: A BRIEF INTRODUCTION TO HTML AND XML.....	12
APPENDIX B: DESIRED INPUT/OUTPUT EXAMPLE	13
APPENDIX C: IDENTIFICATION OF INFORMATION METHODS	15
APPENDIX D: OTHER DOCUMENTS.....	16

Abstract

In the “Internet age”, where a wealth of information is available, it becomes important to get the information desired. This becomes difficult when the information on the web is not uniformly formatted. While technologies such as XML promise to bring more organization to the Internet, it is not commonly used. Many projects have been based around “dumb” extraction – simply taking information for a specified place and storing it to a specified location. A more intelligent method can be used when dealing with information that is semi-structured and needs to be cataloged. The method developed here incorporates information division and recognition to identify and catalog information on the Internet.

Introduction

“How can you keep product design updated with the frequent changes of component specification and price?” is the question that motivates a project called the “Information Alert System”. Right now, developers must keep track of the markets of any products they are interested in “by hand” – looking through the Internet, catalogs, and other vendor information. The Information Alert System is to offer an automated retrieval of information pertinent for a developer.

One of the large problems with such a system is acquiring the information about vendors and products. The system is currently set up to accept information in XML format¹; however vendors do not often provide information in XML format on the Internet, more common is the use of HTML² pages to display and format the information. This is the motivation for the Information Extraction Tool project, the problem:

For the Information Alert System which requires information in a specified format where the data includes metadata (XML), we must take the information source (vendor HTML pages on the Internet) that is unlabeled and unstructured, and convert it into very specific XML format. The problem is to convert from not only unstructured to structured information but also from unlabeled to labeled information. Not only do we have to restructure and locate the information required, but identify the data and sort it accordingly.

On a practical level, given a XML DTD file that specifies the format and the information desired and a URL, how do we, given an unstructured and unlabeled HTML file, create a XML file that contains the information that is structured, labeled, and formatted according to the XML DTD file.

Appendix B contains an example of the HTML (before processing) and XML (after) that we are attempting to create.

¹ See Appendix A for a brief introduction to HTML and XML.

² See Appendix A for a brief introduction to HTML and XML.

Related Work


Most related work in this area that I could find has been in information retrieval. Many programs simply develop ways to describe the points in an HTML document through the HTML DOM structure – for example: traverse page x, go to the 3rd paragraph of the body and retrieve the first word in bold – a “dumb” agent. Several models/implementations have been developed to do this, most notably the World Wide Web Wrapper Factory [1] and ANDES [2]. I could not find a program or model that would perform any identification of the information that I felt would be necessary to label the information.³

³ I could not find such a model or program in academia to the best of my research. I am certain there is no such commercial product at this time.

Approach

First, let us examine some examples of the web pages that we will be working with and try to create some sort of model for the data.

1. Data Model Development



Module Type	Module Description	Part No.	Price	Order
PC66	64MB 168PIN UB PC66 SDRAM DIMM 4K 10NS	M8X64PC66	\$29.00	ORDER NOW!
PC66	128MB 168PIN UB PC66 SDRAM DIMM 4K 10NS	M16X64PC66	\$52.00	ORDER NOW!
PC100	128MB 168PIN UB PC100 SDRAM DIMM CL2/3	M16X64PC100	\$38.00	ORDER NOW!
PC100	256MB 168PIN UB PC100 SDRAM DIMM CL2/3	M32X64PC100	\$65.00	ORDER NOW!

Figure 1: Example Web Page 1 – Tabular Information
(Taken from <http://www.minnesotamemory.com/standard.asp> with permission from author)


MemoryX





Open M-F 9am-7pm PST
Walk in purchases welcome
Free Apple memory installation
no sales tax outside California

Secure online ordering powered by **YAHOO!** 1-866-MemoryX

128mb SDRAM Dimm

 128mb M-Tec PC133 168-pin SDRAM DIMM
\$19.00

 128mb PC133 168-pin SDRAM Dimm
\$23.00

 Intel Approved 128mb PC100-222 (CL2) 168-pin SDRAM Dimm
\$23.00

 128mb PC100-322 (CL3) Registered ECC 168-pin SDRAM Dimm
\$50.00


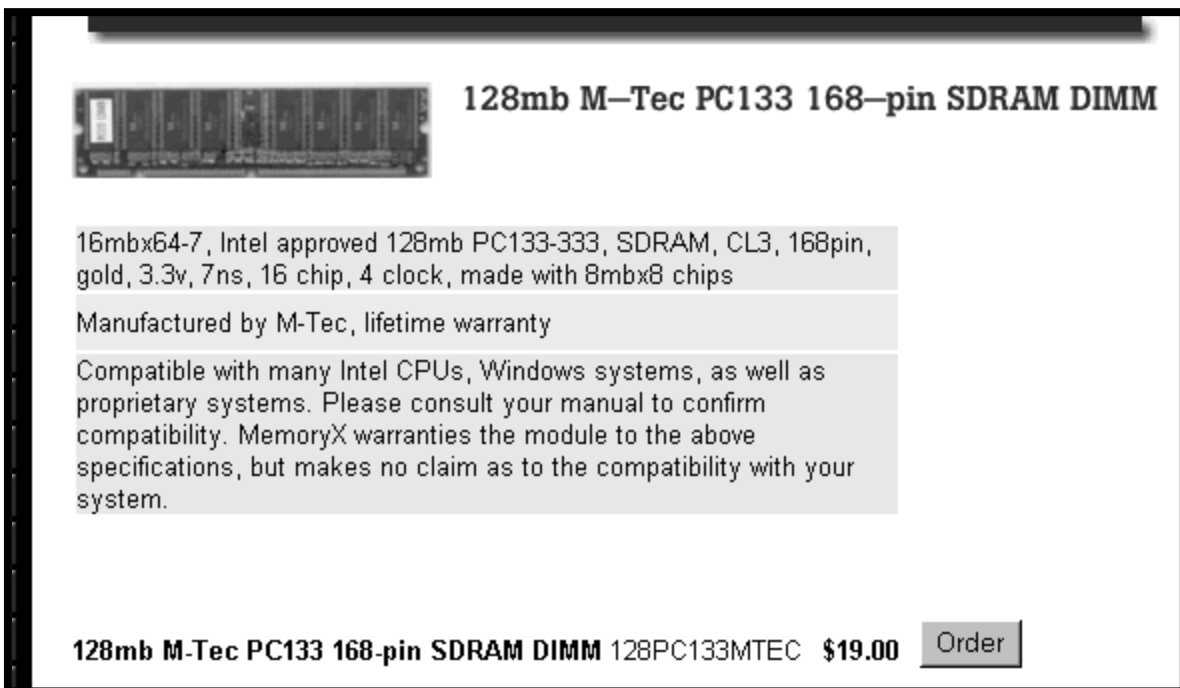
 128mb PC100-322 (CL3) ECC 168-pin SDRAM Dimm
\$43.00

Figure 2: Example Web Page 2 – Parent to Child Information Organization (Parent)
(Taken from <http://www.memoryx.net/128mbsdramdimm1.html> with permission from author)

When the first link is clicked, it brings up another page:



128mb M-Tec PC133 168-pin SDRAM DIMM

16mbx64-7, Intel approved 128mb PC133-333, SDRAM, CL3, 168pin, gold, 3.3v, 7ns, 16 chip, 4 clock, made with 8mbx8 chips

Manufactured by M-Tec, lifetime warranty

Compatible with many Intel CPUs, Windows systems, as well as proprietary systems. Please consult your manual to confirm compatibility. MemoryX warrants the module to the above specifications, but makes no claim as to the compatibility with your system.

128mb M-Tec PC133 168-pin SDRAM DIMM 128PC133MTEC \$19.00 [Order](#)

Figure 3: Example Web Page 2 – Parent to Child Information Organization (Child)

(Taken from <http://www.memoryx.net/128pc133mtek.html> with permission from author)

These two examples cover the majority of vendor web pages surveyed.

In the first example, each row of a table corresponds to one product. Similar web might use columns to correspond to one product. Some web pages fit two products per row. This format is popular due to its simplicity, often seen on smaller vendor's sites.

In the second example, the parent page had a list of links to separate child pages that contain information pertinent to each product. This is a popular method because queries on products can return a list of child pages and not have to build an entire table where information may not match, mostly implemented on larger sites that do not specialize in one product.

In both examples it is easy to see that for each product, there is a "space" where all the pertinent information about a product is organized. In the first example, each "product-space" was a row in the table. In the second example, each "product-space" was the entire child page. You can also note that within these "product-spaces" are delimiters, in the first example the rows are delimited by the columns, in the second example the page itself is delimited by paragraph tabs or even within a table. So we can say that for most information that we will be dealing with, the information about a product will somehow be distinct and occupy its own "space", furthermore, within that space may be further divided.

Here are some graphical representations of the abstract data model:

Attribute 1	Divider	Attributes	Divider	More Attributes
Attribute 1	Divider	Attributes	Divider	More Attributes
Attribute 1	Divider	Attributes	Divider	More Attributes
Attribute 1	Divider	Attributes	Divider	More Attributes
...

Figure 4: A table with products listed one in every row. A different shade is used to represent a different “product-space”; the dividers are the column lines and are exhibited here (“Divider”) to illustrate their role in dividing information making it clearer.

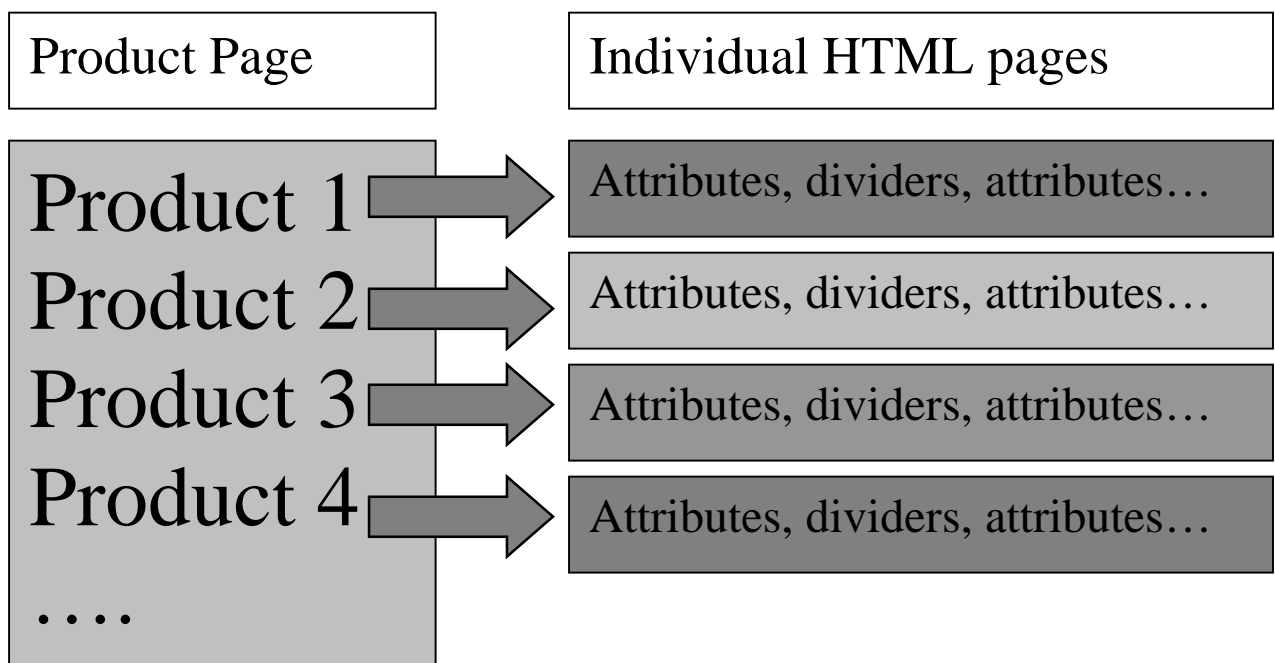


Figure 5: A Parent-Child illustration. A different shade is used to represent a different “product-space” (each individual HTML page); the dividers could be various HTML-marks and are exhibited here (“Divider”) to illustrate their role in dividing information making it clearer.

The data model is very robust and covers other examples of web pages. It gives us a nice beginning structure to create our extraction model.

2. Extraction Model Development

1. A Priori Knowledge

Before we begin processing web pages, we need to determine exactly what information will be available / necessary to run the process of extracting information.

It became clear to me that information regarding the product would be necessary, which we will call “product information”. As well, for the level of precision that we wanted (100%), some information about the way the data was structured on the page would be required, we will call this “web site information”.

In the abstract sense, product information would include how to format the output (to fit the DTD file), what information about the product is pertinent (what are it's attributes), how to identify information as being a certain attribute.

The information about the website would be where the information is located in the web page, how to recognize a “product space” and how it is divided, if at all, within the “product space”.

How this information is obtained is irrelevant for now. User configuration would be the most accurate, but intelligent agents could be used to determine this information with high degrees of accuracy requiring less human interaction.

2. Extraction Process

So, by defining the information that we expect we need, the process becomes easy to define:

- Find the area of interest.
- Gather all information in one “product space”.
- Perform whatever operations necessary to figure out how the information is useful in each entry, including looking at delimiting factors within the information. Identification of information is used within this step.⁴
- Repeat the process until the entire area of interest has been cataloged.

This process is a simple but robust solution for extracting information when the data model holds true. For more details on the implementation of this system, refer to the technical documentation (see “Appendix D: Other Documents”).

⁴ See Appendix C for information on current identification techniques.

Summary

There are several other methods and programs that acquire information from web pages, most all of them rely on using some sort of HTML-DOM grammar to find the information, and then storing it where it is specified. They are mostly unsuitable for this task because they do not incorporate word recognition that is a key feature to the model we have created.⁵

The data model developed for the Information Extraction Tool is an accurate and robust description of the majority of cataloged product information on the Internet surveyed. The extraction process provides for intelligent identification of information that is necessary to get good accuracy and high percentage of information retrieval.

Future Work

With the model we have developed, it seems that the amount of information that is needed to do accurate extraction has been defined. Right now, all of that information is acquired by user input. The largest area of improvement will be to create an agent that can discern, with high accuracy, the information needed to “run” the model engine. Locating the area of interest and determining the semi-structure of the product “space” are the two tasks that an agent could probably be made to do to some high degree of accuracy. If this were done, then only information about the product would have to be configured by hand.

Future implementations could certainly improve the user interface for collecting data. Improvements for the way to recognize information as attributes are needed. Implementations of looking at other page organizations are also needed.

Acknowledgements

I would like to thank:

- ❖ Dr. Lin for being my advisor.
- ❖ ChangXin Xu for creating the need for my research.
- ❖ Ms. Ihasz, the ISR department at the University of Maryland and The National Science Foundation for providing this excellent program.
- ❖ My friends Chris Shultz and Brian Kiefer for their Java advice.
- ❖ My parents for their support.

⁵ That I could find in my research.

References

1. Arnaud Sahuguet, Fabien Azavant. "Building light-weight wrappers for legacy web datasources using **w4f**". In *International Conference on Very Large Databases (VLDB)* (1999), pages 738--741, 1999.
Associated URLs:
<http://citeseer.nj.nec.com/265449.html>
<http://www.tropea-inc.com/>
2. Myllymaki, Jussi. "Effective Web Data Extraction with Standard XML Technologies". In *WWW10*, May 2-5, 2001, Hong Kong. ACM 1-58113-348-0/01/0005.
Associated URLs:
<http://www10.org/cdrom/papers/102/>
<http://www-106.ibm.com/developerworks/web/library/wa-wbdm/>

Appendix A: A Brief Introduction to HTML and XML

HTML is a language that has more to do with how information should look, displaying the information nicely. HTML does not have meta-data (“above” information – information about the information), which is not a concern when used to make web pages viewed by people; it is the standard way to format pages on the Internet. An example of HTML code can be found in Appendix B.

XML is more concerned about information exchange and information understanding, to this end it has meta-data about all of its information – it however is not commonly used right now. To understand what is ment by “meta-data”, see figure A.1.

Figure A.1: Sample XML


```
<catalog>
  <person>
    <name>Alex Lo</name>
    <birthday>2/20/81</birthday>
    <occupation>student</occupation>
  </person>
  ...
</catalog>
```

XML deals with catalogs of information. In this case there is a catalog of people, one of whom has an attribute “name” with the value “Alex Lo” and other attributes. Another example of XML code can be found in Appendix B.

Appendix B: Desired Input/Output Example

This is what the aim of the implementation was: to convert HTML to XML. Here is the HTML code before conversion:

Figure B.1: Row as seen in browser

Type	Module Description	Part No.	Price	Order
PC66	64MB 168PIN UB PC66 SDRAM DIMM 4K 10NS	M8X64PC66	\$29.00	

HTML Code: Row From Table

```
<table border="0" WIDTH="100%" cellpadding="1" cellspacing="1" bgcolor="#FFFFFF">
...
<tr bgcolor="#FFFFFF">
<td><font SIZE="2" color="#000000" face="Verdana, Arial"><b>PC66</b></font> </td>
<td><font SIZE="2" color="#000000" face="Verdana, Arial">64MB 168PIN UB PC66 SDRAM
DIMM 4K 10NS </font></td>
<td><font SIZE="2" color="#000000" face="Verdana, Arial">M8X64PC66 </font></td>
<td align=right><font SIZE="2" color="#000000" face="Verdana,
Arial">$29.00</font></td>
<td align=center><font SIZE="2" color="#0000FF" face="Verdana, Arial">
<a href="/cart/addcart.asp?itemid=1"></a>
</font></td>
</tr>
...
</table>
```

(Source and figure taken from
<http://www.minnesotamemory.com/standard.asp> with permission from
author)

(Continued next page)

XML Code: The Desired Output

```
<catalog>
  <memory_Chip>
    <capacity>64MB</capacity>
    <pin pinNumber="168"/>
    <buffer bufferType="Unbuffered"/>
    <speed>PC66</speed>
    <chip memoryType="SDRAM"/>
    <packaging packagingType="DIMMs"/>
    <OEMpartNumber>M8X64PC66</OEMpartNumber>
    <price>$29.00</price>
  </memory_Chip>
  ...
</catalog>
```

As you can see, the conversion loses the HTML formatting information, but gains structure and meta-data. (Actual output from implemented program)

Appendix C: Identification of Information Methods

The current implementation of the Information Extraction Tool has very simple information recognition capabilities. The exact process depends on the classification of the search associated with the attribute in the product information. There are two processes:

1. “Must be possible value”: the information about the attribute has provided a list of values, if the information matches one of these values then it is a match. It also provides a list of “aliases” that a value might go by. For example: the attribute “packaging type” for the product memory has the possible values “DIMMs” and “SIMMs”, however, some vendor web pages might display only “DIMM” which would be converted to “DIMMs” and identified as being the information for “packaging type”.
2. “Sub-string search”: the information about the attribute has provided a list of possible sub-strings the information might contain that would indicate that it is the attribute. For example, memory’s “capacity” attribute has several possible values, to search all of them would be inefficient and may not be flexible enough to catch new values, so instead we search for the sub-strings “MB” and “GB” and similar names, this allows us to catch “32MB” as easily as “1024MB” which might be written as “1GB”.

This is an area that needs more research and development. Future improvements might include doing a regular expression search that might be useful for identifying serial numbers. It may be useful to train some agent to do better identification through examples and training, but that might be overkill.

Appendix D: Other Documents

Other documents related to this project:

- Technical Documentation
- Source Code
- Users Documentation
- Examples (HTML file and configuration files)
- Presentation

See the Readme.txt file for information on where to find these documents.