DOM STRUCTURE BASED WEB PATTERN MINING

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Naresh Pillarikuppam

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

April 2011

Fargo, North Dakota

North Dakota State University
Graduate School

Title

DOM STRUCTURE BASED

WEB PATTERN MINING

By

NARESH PILLARIKUPPAM

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

ABSTRACT

Pillarikuppam, Naresh, M.S. in Software Engineering, Department of Computer Science, College of Science and Mathematics, North Dakota State University, April 2011. DOM Structure Based Web Pattern Mining. Major Professor: Dr. Jun Kong.

A rapid expansion in the Web has motivated several studies to understand and recognize the implementation structure underlying the interface. Though the presentation of the Web pages looks different, those Web pages may share the same semantic structure to organize information. Those common semantic structures are referred to as Web patterns. There are no strict rules for implementing the HTML structure of the web pages, and the implementation of each web page might not be consistent across the entire website. Also, the HTML implementation of one website varies from other websites. This makes it difficult to recognize the Web patterns that have been used for implementing the websites. In this paper, Document Object Model (called "DOM" hereafter) structure based web pattern mining has been proposed, where the HTML structure and the common patterns are represented in DOM structure format. As an approach for deriving the common web pattern, the implemented patterns observed across different websites are analyzed and summarized manually. Those Web patterns are represented by using the Pattern Structure Definition (PSD) format, which is derived based on the DTD model. Then, an efficient algorithm has been proposed to recognize Web patterns that match with the definition and comply with all the properties defined in the PSD. To recognize the pattern structure, a tool was developed that can take the URL as an input and recognize summarized patterns. The experiment results and evaluation of the tool show the high accuracy of the approach. The implemented approach achieved 91.35% accuracy in finding the navigation pattern structure in the online shopping websites.

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my adviser, Dr. Jun Kong, for providing this research topic. His advice, guidance and support greatly helped in making this work possible. I would like to thank Dr. Kendall Nygard, Dr. Changhui Yan and Dr. Chao You for their valuable time as my committee members.

Special thanks to my friend, Krishna Chaithanya Chinthakayala, for giving a lot of support and encouragement during the project. I would like to express my special thanks to my parents, Chinnabba Reddy Pillarikuppam and Kumari Pillarikuppam, for encouraging and supporting me as I pursued my graduate degree. Finally, I would like to thank my family and friends for their support, without whom it would have been impossible for me to pursue my graduate degree.

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGMENTS iv

LIST OF TABLES viii

LIST OF FIGURES ix

1. INTRODUCTION 1

2. RELATED WORK 5

3. WEB PATTERNS ANALYSIS 11

3.1 Navigation Patterns 11

3.2 Patterns Observed 13

3.2.1 TARGET 14

3.2.2 TOYSRUS 14

3.2.3 BIZRATE 15

3.2.4 CIRCUIT CITY 16

3.2.5 BUY 16

3.2.6 AMAZON 17

3.2.7 OVERSTOCK 18

3.2.8 SMART BARGAINS 18

3.2.9 THE NATURE STORE 19

3.2.10 NEW EGG 20

3.3 Common Patterns Derived 20

3.3.1 List Structure 21

3.3.2 Tabular Structure 22

3.3.3 Expected Websites for Patterns 24

TABLE OF CONTENTS (Continued)

4. DESIGN 25

4.1 Existing Techniques and Limitations 25

4.1.1 DTD Implementation 25

4.1.2 XML Schema 27

4.2 Proposed Solution 30

4.2.1 Pattern Structure Definition (PSD) Syntax 31

4.2.2 Rules Considered 33

4.2.3 Assumptions 33

4.2.4 Pattern Match Finder Tool Design 34

4.2.5 Tool Features 36

4.3 Common Patterns 38

4.3.1 List Structure 38

4.3.2 Tabular Structure 39

5. IMPLEMENTATION AND TOOL VISUALIZATION 40

5.1 Technologies Considered 40

5.1.1 JAVA 40

5.1.2 ECLIPSE 40

5.1.3 HTML Graph 41

5.1.4 Cobra: Java HTML Parser 41

5.2 Pattern Match Finder Tool 41

5.2.1 Input Parse Tree 43

5.2.2 Common Pattern Files 43

5.2.3 Tree Comparison 45

5.2.4 Display Matched Paths 47

TABLE OF CONTENTS (Continued)

6. EVALUATION 48

6.1 Precision and Recall 49

6.2 Evaluation Using Additional Websites 51

6.3 Evaluation Results 53

7. CONCLUSION AND FUTURE WORK 55

REFERENCES 57

LIST OF TABLES

Table Page

1. Expected Websites for Each Pattern 24

2. Attributes Used for Node Property 32

3. List of Websites and Patterns Matched 48

4. Precision and Recall Calculations for Websites Listed 49

5. Additional Websites Used for Evaluating the Tool 51

LIST OF FIGURES

Figure Page

1. Navigation Pattern 12

2. Target Navigation Pattern 14

3. ToysRUs Navigation Pattern 15

4. Bizrate Navigation Pattern 15

5. CircuitCity Navigation Pattern 16

6. Buy.com Navigation Pattern 17

7. Amazon Navigation Pattern 17

8. Overstock Navigation Pattern 18

9. SmartBargains Navigation Pattern 19

10. TheNatureStore Navigation Pattern 19

11. NewEgg Navigation Pattern 20

12. List Structure Pattern 21

13. Sample Screenshot of Amazon [17] Website that Follows List Structure 22

14. Tabular Structure Pattern 23

15. Sample Screenshot of NewEgg [16] Website that Follows Tabular Structure 24

16. DTD Format of Amazon Navigation Pattern 26

17. Syntax of DTD Declaration in XML 27

18. XML Schema Format of Amazon Navigation Pattern 29

19. Syntax of Schema Declaration in XML 30

20. Sample PSD of Navigation Pattern 32

21. Design of Pattern Discovery Architecture 35

LIST OF FIGURES (Continued)

22. List Structure 38

23. Tabular Structure 39

24. Output of HTML Graph Tool 42

25. HTML Pattern Match Finder 43

26. Tree Structure Representation of Pattern 44

27. List of Patterns Matched and Expanded Matched Patterns 47

28. Average Precision and Recall 53

29. Sample Code for Invalid Exception 56

1. INTRODUCTION

A rapid expansion in the Web has motivated several studies to understand and recognize the implementation structure underlying the interface. Websites belonging to the same category contain information that is generated using a common layout [1]. There are no strict rules for implementing the HTML structure of the web pages and the implementation of each web page might not be consistent across the entire website. And also, the HTML implementation of one website varies from other websites. This makes it difficult to recognize the Web patterns that have been used for implementing the websites. By analyzing different websites that show similar structure, the common layout structure that has been shared among the websites can be derived. Zhen Zhang, Bin He, and Kevin Chen-Chuan Chang [2] made a considerable study, where the patterns of web query interfaces and the hidden syntax behind them are analyzed and the common patterns across different websites are observed. This study [2] also proposed pattern specification, which represents the grammar rules of pattern and pattern recognition that used a parser to recognize the pattern. This study also motivated the current work made in this paper, where the current work emphasizes recognizing and extracting the common pattern from the HTML implementation structure instead.

In this paper, a list of popular websites under the online shopping category has been considered for analysis. Though the visual appearance of these websites differs with each other, the document structure (HTML structure implementation) underlying the interface can be the same for similar scenarios. For instance, the implementation in amazon.com and buy.com for displaying a product, shares a similar structure by showing the product image,

followed by product title and price details. Observing these kinds of structures which occur repeatedly can help to recognize the pattern occurrence among different websites.

The primary objective of this paper is to discover the common web patterns that are being implemented in the online shopping websites. The Web pattern referred to in this paper can be defined as "repeated common semantic structures implemented in a website." The common web patterns are determined by discovering the common HTML semantic structure used across different websites. In the implementation perspective, the goal of this paper is to introduce a new algorithm for identifying the type of web patterns implemented in the given website.

Identifying and extracting all the occurrences of a pattern in a huge sized data tree is a major concern in most of the pattern based recognition studies [5]. F.Mandreoli and P.Zezula [5] have done a considerable study in searching for all occurrences of a small query tree in the data trees. Mohammed J. Zaki [3] proposed a tree miner, an algorithm that mines frequent sub trees in a database, which is useful in the fields of web mining and bioinformatics. There is not much contribution made to recognize and extract the pattern occurrence of a HTML structure underlying the web interface, which is represented in JTree format. This paper emphasizes discovering the pattern matched based on JTree structure. The goal of the paper is achieved by comparing the pattern structure and input tree (HTML implementation), where both the pattern structure and the input are represented in the JTree format.

The key contribution of this paper is implementing a tool which recognizes and extracts the pattern structure of the input by comparing it against a list of common patterns. And the other contributions include:

- The pattern structure that specifies the syntax of pattern, called *Pattern Structure Definition,* which is extended from Document Type Definition (DTD) has been proposed.
- The pattern recognition algorithm which compares the input and common pattern that are in JTree format has been proposed. This algorithm recognizes and extracts the pattern structure in the input.

Unlike the traditional parsing mechanisms, the pattern recognition parsers do not enforce the grammar rules and will not reject any input [2]. So, the pattern recognition parser in this paper also recognizes and extracts partial matches along with the full matches. A heuristic rule has been considered while finding the pattern match. The rule that is considered is, the proposed algorithm finds a pattern match only if the node with "+" attribute has been appeared at least five times under its parent node. As long as the web page is rendered correctly, the common pattern match can be recognized efficiently.

Recognizing the pattern can be useful, as the pattern can be inherited while implementing new websites or new web pages in the same website. Also, there is a lack of standards that need to be followed by the online websites and a lack of ways to find out the patterns that have been implemented. This model can be useful if the websites need to be imposed with any standards and to verify or observe, whether the desired patterns are implemented in the websites.

The rest of the paper is organized as follows. Chapter 2 discusses some of the closely related works. Manual summarization of the web patterns across different websites are provided in Chapter 3. The design of the proposed tool and syntax are discussed in Chapter 4. Along with the design, Chapter 4 describes existing techniques and their

limitations. Chapter 5 presents the implementation and tool visualization of the proposed idea. Evaluation of the tool is done in Chapter 6. The conclusion and future work are described in Chapter 7.

2. RELATED WORK

Recognition and extraction of the web pattern from a website is an interesting application, especially with the growth of a number of online websites and with the growth of the number of people using them. But there is limited work related to this field of study that has been contributed. The work in this paper is related to other works based on the following aspects.

- Observing common patterns across multiple websites
- Pattern specification and recognition
- Finding patterns based on tree structure

Firstly, observing the common pattern of the web query forms was studied by Zhen Zhang, Bin He, and Kevin Chen-Chuan Chang [2] which is the motivation for the current work. That study [2] found the common patterns for web query forms which repeated more than once and discussed that, even though each query form is different, they share common patterns. In a similar way, this paper also summarizes the patterns of each website, by evaluating multiple websites and deriving common web patterns that are shared by them. While the study [2] deals with observing the common patterns of web query forms, this paper focus on common patterns of HTML implementation structure behind web interface.

A. Arasu and H. Garcia-Molina [1] studied deriving the common layout from a large set of web pages. While the common layout mentioned in the study [1] is based on structured data behind the web interface, the common pattern defined in this paper is not based on actual data. Their study also presents an algorithm that takes web pages as input, traces the template or layout used and extracts the values encoded in the pages. The algorithm in this paper also takes web pages as input and finds the pattern used in the web

that has maximum repeated expressions in the set. The user can select and save the desired pattern to be extracted from the web page.

D. C. Reis, P. B. Golgher, A. S. Silva and A. F. Laender [26] presented a study regarding Automatic Web News Extraction Using Tree Edit Distance. In the study [26], a new approach to extract the data (news) from the websites based on the data matching and extraction has been proposed. The first step in the algorithm proposed in the study [26] is page clustering. To derive the clustering of pages, the training set of pages is given as input and the clustering of pages is generated based on the common formatting or layout features. Each cluster is generalized into a node extraction pattern, simply *ne-pattern.* The *ne-pattern* defined in the study [26] is used as a common pattern for recognizing the pattern occurrences in the web page.

J. Y. Hsu and W.-T. Yih [15] proposed template-based mining from HTML documents. In the study [15], the concept of *document templates* is introduced. Each template specifies the structural components of a class of documents to be captured. These *document templates* are used as common patterns for the information extraction. The study [15] defines the *document template* as a logical structure that has been extracted from a collection of similar websites. This paper also uses a similar idea to derive the common patterns by analyzing a set of online shopping websites.

Second, Zhen Zhang, Bin He, Kevin Chen-Chuan Chang [2] proposed the pattern specification and pattern recognition. In the study [2], 2P Grammar which encodes the patterns and their precedence for web query forms is proposed where as in this paper, the Pattern Structure Definition for representing the HTML structure pattern is proposed. For the pattern recognition, both the study [2] and this paper use an algorithm, which uses the

pattern specification as input. However, the pattern recognition in the study [2] is completely different to the one in this paper. The work [2] is closer to the current paper because both deal with observing and recognizing the common patterns across multiple websites.

In the study presented by M. Cosulschi, A. Giurca, B. Udrescu, N. Constantinescu, and M. Gabroveanu [30], proposed a pattern extraction process. The structure similarities are computed using *tree edit distance* between the trees. The extraction process in the study [30] focused on grouping the web pages into clusters and generating the extraction pattern for each cluster. The extraction pattern proposed contains labels and special elements called wild cards. These wild cards must be a leaf in the tree. Four types of wild cards are defined – single (.), plus (+), option (?) and kleene (*). In this paper also, the properties of the nodes in the pattern tree are defined in the same way as that of the study [30]. But the difference is nodes of the pattern tree in this paper need not to be a leaf node.

The pattern extraction has been proposed in the IEPAD system presented by C.-H. Chang, C.-N. Hsu, and S.-C. Lui [29]. The extraction of the pattern matching records is achieved through a standard pattern matching algorithm called Boyer-Moore's algorithm. The major difference between the study [29] and this paper is the input patterns in the study [29] are defined as a subset of HTML tags where in this paper they are defined as nodes. The result of the extraction in [29] gives a matrix that contains information slots, whereas the extracted result of this paper gives the set of matched patterns in a tree structure. Partial pattern matches are not achieved by the study [29] where as this paper provides the partial matches also.

In the study [26] presented by D. C. Reis, P. B. Golgher, A. S. Silva, and A. F. Laender, the data matching approach has been proposed to find an appropriate *ne-pattern* from the set of HTML pages. The *ne-pattern* is defined as a rooted tree that contains sibling of sub-trees as child trees. And the trees contain special vertices called wildcards. Similar to the study [30], each wildcard can be one of the following types – Single (.), Plus (+), Option (?), and Kleene (*). This paper also defines similar structure for the derived common pattern. But the difference is the wild card in the study [26] must be a leaf node where as the node of the pattern defined in this paper can appear at any point of the tree. For the pattern recognition, the study [26] has used the *RTDM* (Restricted Top-Down Mapping) algorithm. Once the mapping between the ne-pattern and HTML Page is found, the matched trees are extracted. The algorithm used for pattern recognition in the study [26] did not address the partial pattern matches where the proposed algorithm in this paper provides the partial matches also.

An Information Extraction Algorithm has been proposed by J. Y. Hsu and W.-T. Yih [15]. The algorithm in the study [15] accepts an electronic document, a collection of document templates and a set of extraction targets. In return, the algorithm identifies the best matched template for the document and extracts the matched information in the document. This paper uses the algorithm which has some similarities to that of the algorithm defined in the study [15]. But the algorithm proposed in this paper is based on the comparison of tree structures and extracting the information based on the tree comparison. The similarities between the study [15] and this paper is both papers propose an information extraction in the given document based on template/common pattern structure.

Third, F.Mandreoll and P.Zezula contributed studies that have been made on tree pattern matching [5] to find the occurrences of a twig pattern in a tree-structured document, where twig pattern is a query represented in tree and tree structured document is a data tree. The current paper also emphasizes tree pattern matching, but the tree in this paper represents the HTML structure instead of data. And the pattern matching is not matching the data; it is recognizing and extracting the HTML structure of a website.

In the study presented by M. Cosulschi, A. Giurca, B. Udrescu, N. Constantinescu and M. Gabroveanu [30], an algorithm for the pattern matching process has been proposed. The algorithm accepts the pattern tree input and data tree input which are in XHTML format. The algorithm proposed in the current paper also accepts the tree structures as inputs. But the procedure in the study [30] accepts list of nodes every time whereas the algorithm in this paper accepts only the current node but not the list.

3. WEB PATTERNS ANALYSIS

A pattern can be defined as a sequence of events that can appear a repeated number of times [4]. By analyzing the set of patterns, the behavior of the program can be mined [4]. To analyze the pattern structures across different websites for a particular category, online shopping has been chosen in this paper. The reason behind selecting the online shopping category in this paper is due to a drastic increase in the online shopping websites and their purposes. A number of organizations that are using the web for marketing, promoting, and transacting products and services with consumers are increasing [6]. A study reported that about 40% of participants indicated shopping as a primary use of the web [6]. This increases the need for study of online shopping websites and to understand the design of the websites. This motivated us to consider the online shopping category for the analysis in this paper. For implementing online shopping websites, one of the important factors to be considered is Navigation [7]. For any website to be constructive and efficient, a good navigation always plays an important role. Especially, users of online shopping websites look for pathways to navigate through the site. By considering navigation analysis for this paper, an important factor of the website has been analyzed.

3.1 Navigation Patterns

The navigation of an online shopping website is related to the user interface of an online store [7]. Navigation patterns can be determined based on the layout that the online shopping website has used.

The navigation pattern represents a set of navigation links, which are of type hyperlink. The links can be organized in a hierarchical structure, and the second level is indented. Each link directs the user to the corresponding page. The grouping can be done

by representing the hyperlinks either in a list or in a tabular format. The following Figure 1 is an example of the navigation pattern format which has a Department Name and each department can have multiple Branch Names. Each branch can have multiple category names.

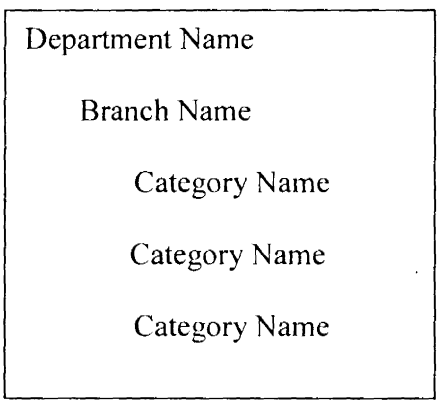

Figure 1. Navigation Pattern

For analyzing the behavior of a program, it is not sufficient by considering few scenarios. To retrieve the correctness of the pattern recurrence and the pattern structure of multiple systems, a set of websites needs to be analyzed. Ten different popular online shopping websites are chosen for the analysis. The list of the chosen websites is shown below:

- Target.com
- ToysRUs.com
- Bizrate.com
- CircuitCity.com
- Buy.com
- Amazon.com
- Overstock.com

- SmartBargains.com
- ThenatureStore.com
- NewEgg.com

The websites in the above list are selected as they are the most popular among the online shopping category. The Target and Amazon companies are listed among the Top 100 companies of the Fortune 500 [31]. And also the other websites considered here cover diverse sources of online shopping. For example, ThenatureStore does not belong to regular online shopping, but it provides products belonging to nature-related education. Similarly ToysRUs which is in the list provides specific products dedicated to kids toys. This company was ranked 189th in Fortune 500 [31]. The websites such as CircuitCity and NewEgg are meant for selling electronic products only. The other websites are listed as top online shopping websites when searched in web search engines (eg., google.com). By considering these websites for the analysis, the proposed solution in this paper can achieve a good coverage in the online shopping category.

For each website listed above, the navigation implementation has been studied. Each website shows its own way of representing the pattern for the Navigation. But most of the websites showed some similar structure for the navigation pattern. The similar structures that are observed can be grouped together to make a common pattern.

3.2 Patterns Observed

The pattern in each website listed above is studied individually to find the similarities and differences between them.

3.2.1 TARGET

The pattern that has been observed in the Target [20] website contains multiple DIV tags appearing more than once under the BODY tag. And also under the UL tag, the LI tag has appeared more than one time. When the observed pattern is converted to the DOM structure format, it looks as shown in Figure 2.

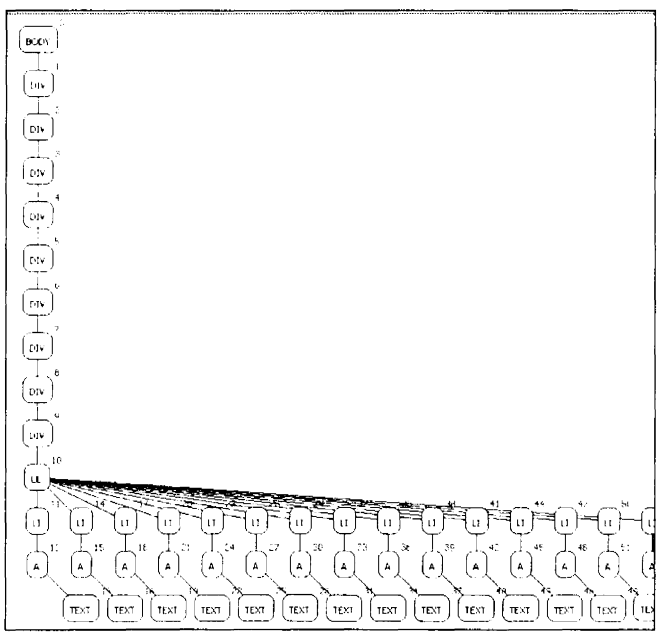

Figure 2. Target Navigation Pattern

3.2.2 TOYSRUS

The navigation pattern observed in the TOYSRUS [25] website has a similar structure as that found in the TARGET website. The major difference is that the DIV tag under the BODY tag does not appear more than one time in this pattern. And also the SPAN tag does not appear after the A tag. But the A tag appears more than one time which does not in the TARGET pattern. The navigation pattern found in the TOYSRUS website contains more than one LI tag under UL tag. The following Figure 3 shows the DOM structure format of the TOYSRUS website.

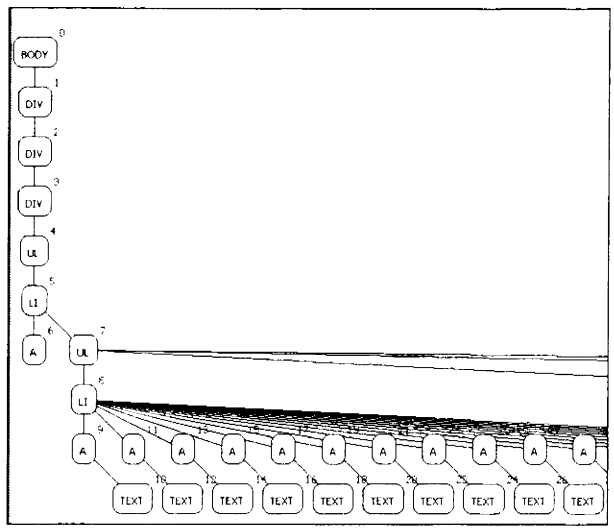

Figure 3. ToysRUs Navigation Pattern

3.2.3 BIZRATE

The navigation pattern observed in the BIZRATE [23] website has a similar structure as that found in the TOYSRUS website. The navigation pattern found in the BIZRATE website contains more than one LI tag under the UL tag. The following Figure 4 shows the DOM structure format of the BIZRATE website.

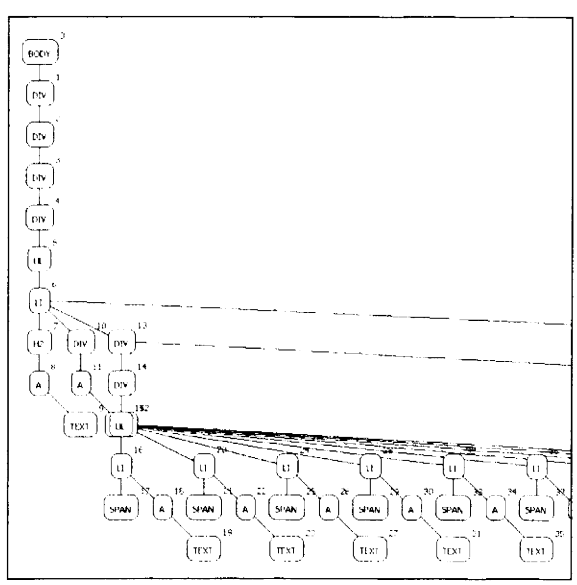

Figure 4. Bizrate Navigation Pattern

3.2.4 CIRCUIT CITY

The navigation pattern observed in the CIRCUIT CITY [24] website has a similar structure as that found in the TOYSRUS website. The navigation pattern found in the CIRCUIT CITY website contains more than one LI tag under the UL tag. The structure of the CIRCUIT CITY website in the DOM structure format looks as shown below in the Figure 5.

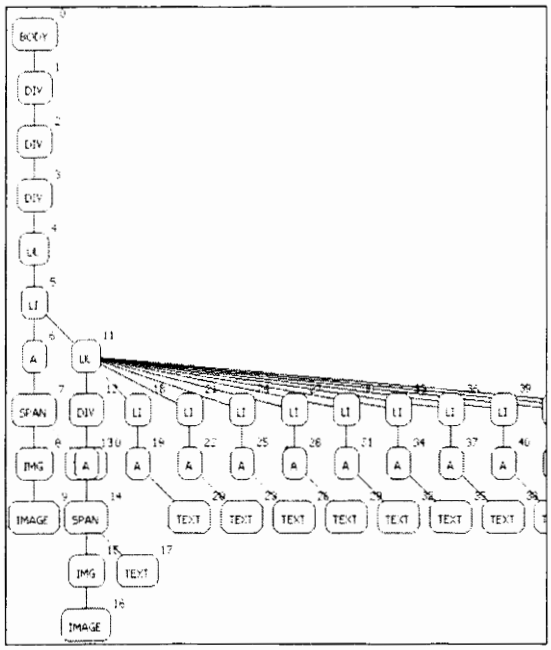

Figure 5. CircuitCity Navigation Pattern

3.2.5 BUY

The navigation pattern observed in the BUY.com [18] website has a similar structure as that found in the TOYSRUS website. The navigation pattern found in the BUY.com website contains more than one LI tag under the UL tag. The structure of the BUY.com website in the DOM structure format looks as shown below in Figure 6.

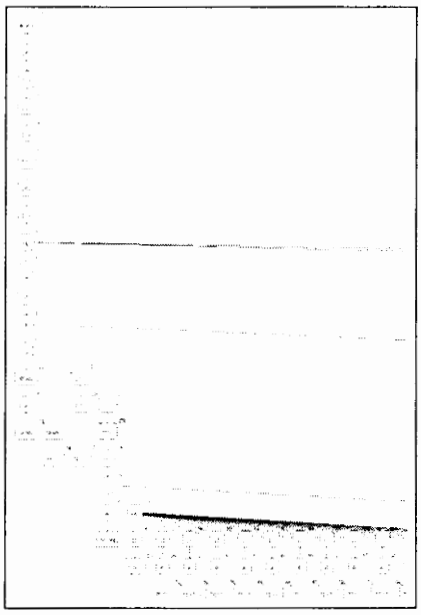

Figure 6. Buy.com Navigation Pattern

3.2.6 AMAZON

The navigation pattern observed in the AMAZON [17] website has a similar structure as that found in the TOYSRUS website. The navigation pattern found in the AMAZON website contains more than one LI tag under the UL tag. The structure of the AMAZON website in the DOM structure format looks as shown below in Figure 7.

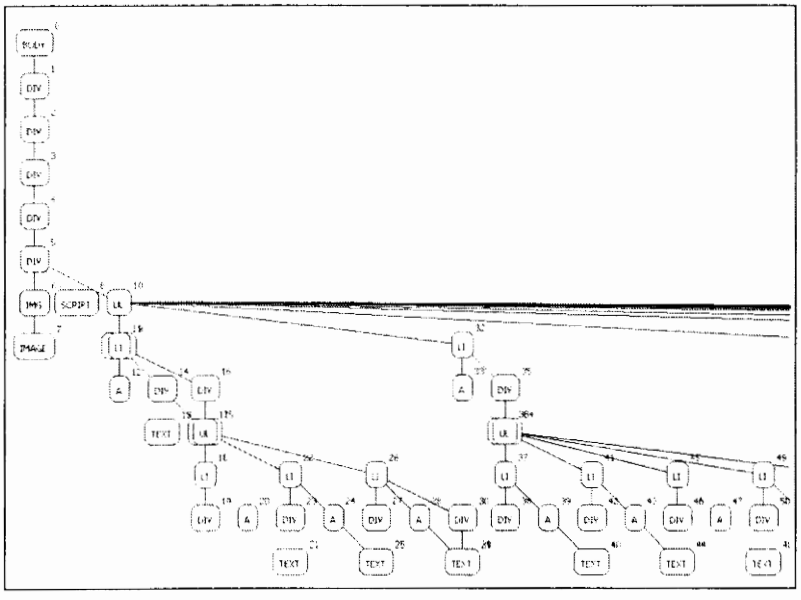

Figure 7. Amazon Navigation Pattern

3.2.7 *OVERSTOCK*

The navigation pattern observed in the OVERSTOCK [19] website has a similar structure as that found in the TOYSRUS website. The navigation pattern found in the OVERSTOCK websites contains more than one LI tag under the UL tag. The structure of the OVERSTOCK website in the DOM structure format looks as shown in Figure 8 below.

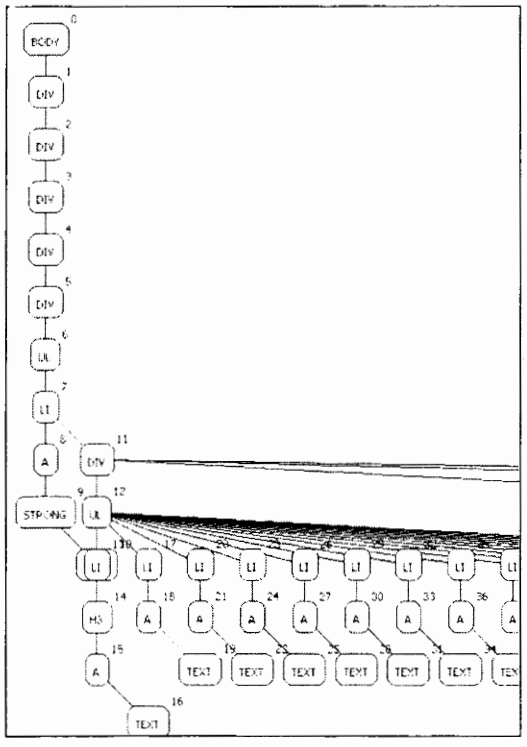

Figure 8. Overstock Navigation Pattern

3.2.8 *SMART BARGAINS*

The implementation of the Navigation pattern for the smart bargains [21] website contains a table structure under the BODY tag. The pattern structure contains an implementation of Table where each column of each row has a UL tag defined with multiple LI tags. Figure 9 below represents the DOM structure format of the observed pattern structure.

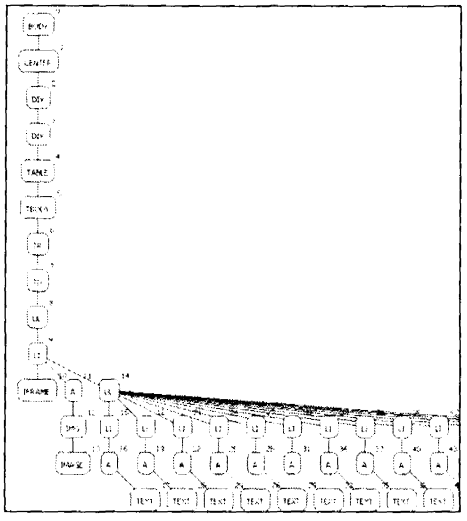

Figure 9. SmartBargains Navigation Pattern

3.2.9 THE NATURE STORE

The navigation pattern observed in the NATURE STORE [22] website has a similar structure that has been observed in the SMART BARGAINS website. The major difference observed in this pattern from the SMART BARGAINS pattern is, the DIV tag has been implemented after TD tag. And also the A tag does not contain any IMG tags as child elements. The DOM structure representation of NATURE STORE navigation pattern is shown below in Figure 10.

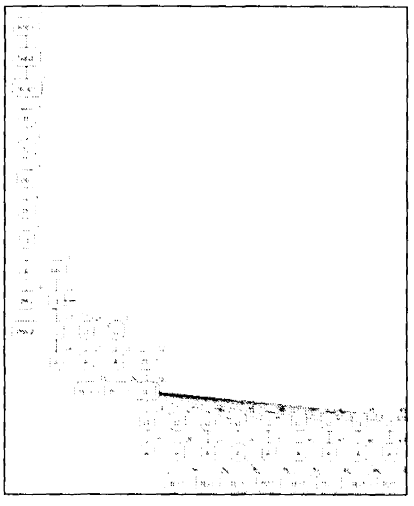

Figure 10. TheNatureStore Navigation Pattern

3.2.10 NEW EGG

The navigation pattern observed in the NEWEGG [16] website has a similar structure that has been observed in the SMART BARGAINS website. The major difference observed in this pattern from the SMART BARGAINS pattern is the DIV tag appeared after the BODY tag where as there is no DIV tag in the SMARTBARGAINS pattern. The IMG tag does not appear as a child for the A tag. The tree structure representation of the NEWEGG navigation pattern is shown below in Figure 11.

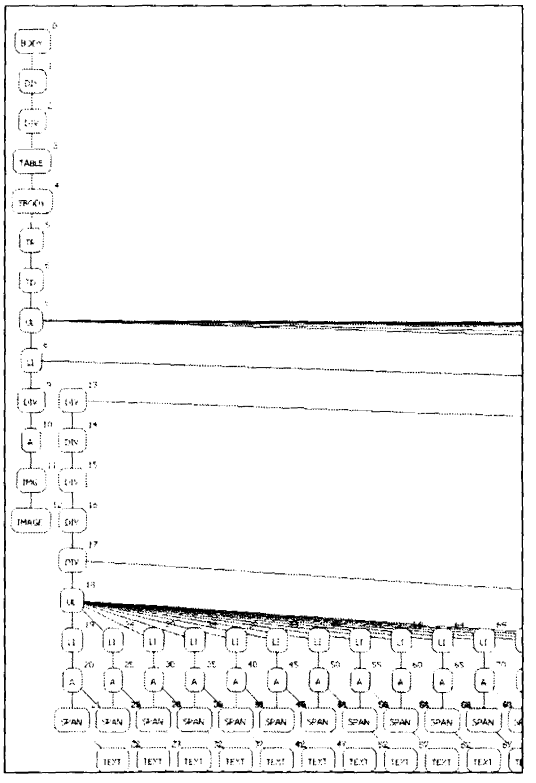

Figure 11. NewEgg Navigation Pattern

3.3 Common Patterns Derived

Based on the analysis made manually in the previous section and by summarizing the similarities and differences between each pattern, the following two common patterns have been derived.

- List Structure
- Tabular Structure

To obtain the common patterns, some of the nodes are made optional and some of them are made to appear more than one time.

3.3.1 List Structure

This kind of pattern is named as List Structure because the pattern structure contains multiple lists of hyperlinks for the navigation of the website. Figure 12 below shows the list structure of the pattern. It has been observed that the LI node appeared multiple times under the UL node. And also some websites have STRONG has child node and some have SPAN as child node for the A node and most of them do not have either. So both STRONG and SPAN can be considered optional nodes. Figure 13 is the screenshot of the AMAZON.com website that shows the list structure for navigation implementation.

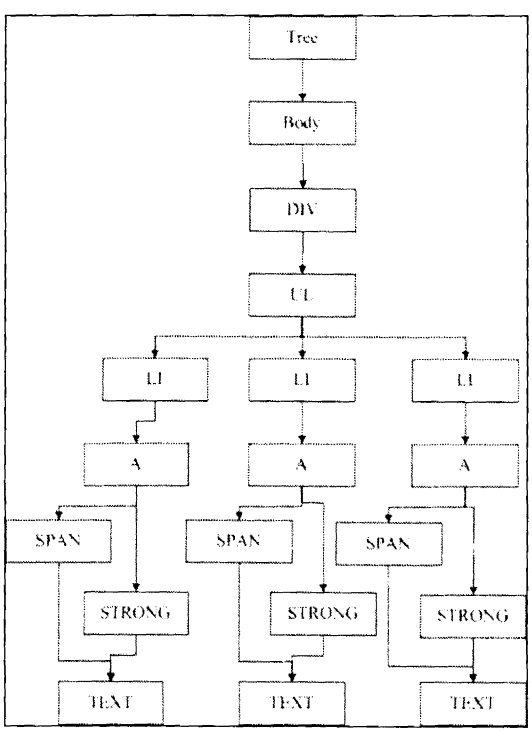

Figure 12. List Structure Pattern

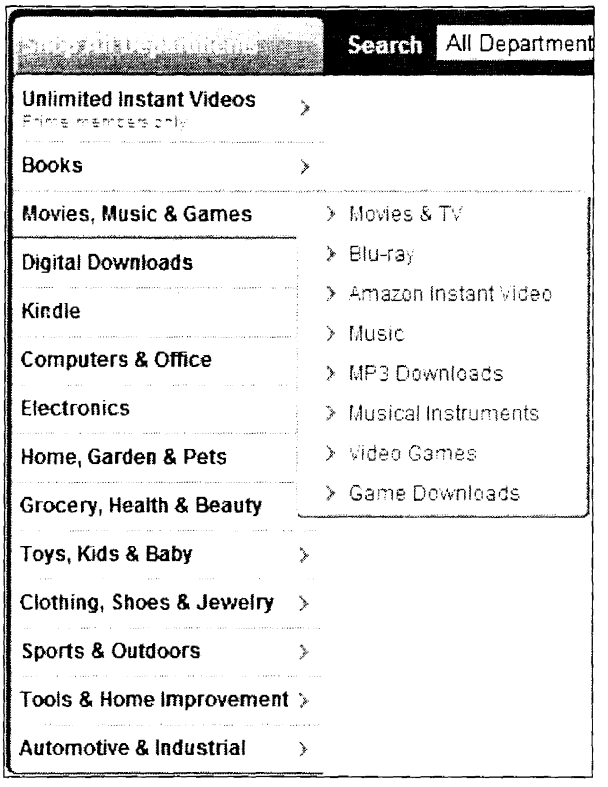

Figure 13. Sample Screenshot of Amazon [17] Website that Follows List Structure

3.3.2 Tabular Structure

This kind of pattern is named Tabular Structure because the pattern structure is implemented using Table tags. Figure 14 below shows the tabular structure of the pattern. A list of navigation hyperlinks are implemented in the table. Some websites have the DIV or CENTER nodes after BODY followed by the <TABLE> node. So these can be considered as optional nodes. The TABLE node shall have TR followed by TD. The TD node shall have the navigation hyperlinks implemented as a UL tag. Figure 15 is the screenshot of the NEWEGG website that shows the tabular structure.

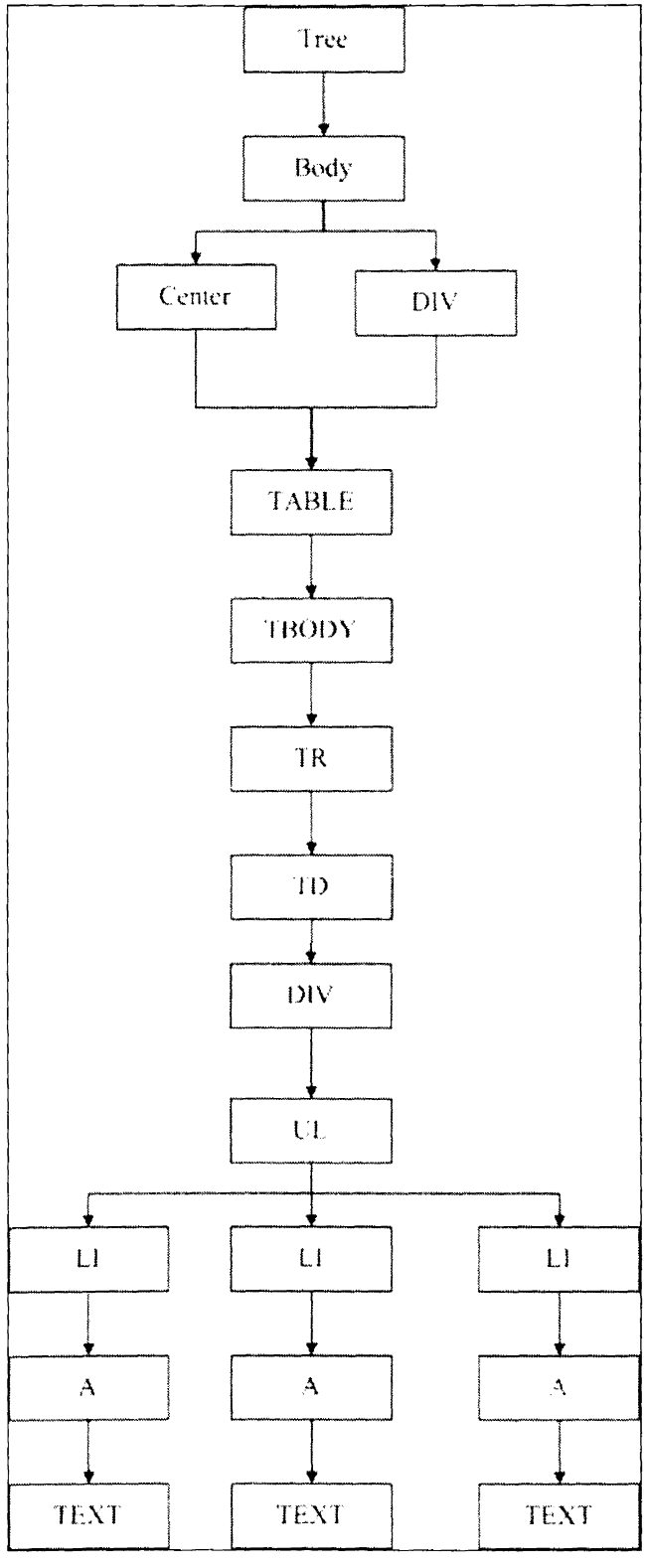

Figure 14. Tabular Structure Pattern

Figure 15. Sample Screenshot of NewEgg [16] Website that Follows Tabular Structure

3.3.3 Expected Websites for Patterns

For the websites chosen for analysis, this section discusses to which of the common patterns the websites belong to. The following Table 1 shows the pattern name and the websites that has the corresponding pattern structure.

Table 1. Expected Websites for Each Pattern

Pattern Name	Websites
List Structure	BUY, NEWEGG, BIZRATE, TARGET, SMARTBARGAINS, AMAZON, TOYSRUS, CIRCUITCITY, OVERSTOCK.
Tabular Structure	NEWEGG, SMARTBARGAINS, THENATURESTORE, TOYSRUS

4. DESIGN

This chapter explains the existing techniques and their limitations and the proposal of a new solution for finding the matched patterns. The main idea behind the implementation of the existing techniques like DTD and XML Schema is to convert the parse tree generated from the HTML Graph to XML. And the output XML from this conversion can be used for the validation. If the XML is validated without any errors, then the XML is considered to have pattern matches. The idea to convert parse tree to XML is considered because the parse tree is represented with nodes containing parent, children and sibling nature. This kind of structure can be represented by XML in a standard way.

4.1 Existing Techniques and Limitations

4.1.1 DTD Implementation

The idea behind this implementation was to convert parse tree which is output of HTML Graph tool to XML and use the standard XML validation through DTD. The reason for considering the DTD approach is the navigation pattern can be represented as DTD and this pattern can be used for the validation of the input HTML source, which in turn validates the pattern structure of the given website. If the validation fails then it means the structure is not being followed which implies the pattern is not observed.

Definition:

Document Type Definition (DTD) is a set of markup declarations that define a document type for SGML-family markup languages (SGML, XML, and HTML) [11]. DTD is used to declare the elements and their attributes that may appear in the document [11]. This way, with a DTD, the format of XML can be described and the elements in the

XML can be verified [12]. Since the elements described in the DTD format is a standard approach, this can be used in all of the applications to verify the data is valid [12].

Implementation:

In order to implement the DTD approach for the pattern matching in the input website, the following steps need to be considered.

- The parse tree generated by the HTML Graph is converted to XML.
- A DTD is created manually based on the defined pattern.
- The DTD developed is used for validating the XML generated by the tool by adding the DTD in the XML file.

Figure 16 shows the DTD representation of the Navigation Pattern derived for the Amazon website. Assume that this file is named as navigationpattern.dtd:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Tree (BODY)>
<!ELEMENT BODY (DIV)>
<!ELEMENT DIV (UL*)>
<!ELEMENT UL (LI+)>
<!ELEMENT LI (A)>
<!ELEMENT A (TEXT)>
<!ELEMENT TEXT (#PCDATA)>
```

Figure 16. DTD Format of Amazon Navigation Pattern

The XML generated from the parse tree of the HTML Graph is validated using the DTD prepared, by adding the following tag in the XML. As the DTD is being declared

inside the XML file, it must be enclosed with DOCTYPE definition and the syntax looks as in Figure 17 [12]. Since the DTD file has been declared in a separate file, the filename must be specified as shown in Figure 17 [12].

```
<!DOCTYPE Tree SYSTEM "navigationpattern.dtd">
```

Figure 17. Syntax of DTD Declaration in XML

Limitations:

- All the elements in the DTD need to be defined prior to the validation of the XML.
- Order of the child elements is one of the constraints to be considered. Validation fails if the order is incorrect. The order of the input XML cannot be predicted.
- It is difficult to decide and include all the elements of the XML, because each website has its own elements defined. The structure of the pattern may be similar in multiple websites, but the way of implementation does not need to be the same.

The above limitations make the DTD approach not suitable for the pattern match recognition.

4.1.2 XML Schema

The limitations discussed in DTD can be addressed by replacing DTD with XML Schema for defining the common pattern. The XML Schema approach is considered because all elements need not to be known while writing the Schema. These can be suppressed by using the attribute "XSD: ANY" in the XML schema. The <any> element enables the program to extend the XML document with elements not specified by the schema.

Definition:

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntactical constraints imposed by XML itself [13].

This approach is similar to that of the DTD approach in which the common pattern can be represented as XML Schema. This can be used for the validation of the XML, which in turn validates the input HTML source, thereby validating the pattern structure of the given website. If the validation fails, then it means the structure is not being followed which implies the pattern is not observed.

Implementation:

The following steps are to be considered for the implementation.

- The parse tree generated by the HTML Graph is converted to XML.
- An XML Schema is created manually based on the defined pattern.
- And the XML Schema is used for validating the XML generated by the tool.
- A Java class is written for validating the xml based on the schema.

Figure 18 shows the XML Schema representation of the Navigation Pattern derived for Amazon website.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:html="http://www.w3.org/1999/xhtml">
<xsd:element name="Tree"/>
<xsd:element name="DIV">
<xsd:complexType>
<xsd:choice minOccurs="1">
<xsd:sequence>
<xsd:element name="UL" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="LI" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:choice minOccurs="1">
<xsd:element name="A">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="TEXT"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Figure 18. XML Schema Format of Amazon Navigation Pattern

The XML generated from the parse tree of the HTML Graph is validated using the XML Schema prepared. by adding the following tag in the XML. The XML Schema Location can be specified by using the xsi:schemaLocation attribute [14]. The schema attribute has two values, the first value is used for namespace and the second is used for declaring the location of the XML Schema [14]. The declaration of XML Schema inside the XML file is as shown in Figure 19.

```
<note xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com navigation.xsd">
```

Figure 19. Syntax of Schema Declaration in XML

Limitations:

- It is hard to define attributes that describe other than the mandatory elements can also be allowed to the element. The input XML of the parsed input source might have an element other than the specified attribute for a particular element. This makes XML Schema a not good technique for the pattern recognition.

  Eg: As per the pattern, DIV must be followed by an UL tag. The HTML Structure could have additional elements along with the UL, either before or after. These additional elements cannot be defined while writing the schema.

4.2 Proposed Solution

The limitations of using DTD and XML Schema for pattern recognition are addressed by proposing a new solution in this paper. A new type of structure recognition has been proposed to find the pattern structure in the given input. The syntax of the

proposed pattern extends from DTD. For recognizing the desired pattern structure, an algorithm that uses the pattern file to find the match in the input file has been implemented.

Implementation:

The following steps are to be considered for the implementation.

- The parse tree generated by the HTML Graph is given as input to the tool.
- The common patterns are derived and defined in the proposed format.
- Based on the common patterns, the program finds the matched patterns in the parse tree.

4.2.1 Pattern Structure Definition (PSD) Syntax

Since the implementation does not use any existing techniques for the proposed solution, to represent the common patterns a new syntax has been defined. The pattern can be converted into this syntax format and can be saved as text (.txt) file. Since the new format proposed extends from DTD, and this is being used for recognizing the pattern structure, this can be called as PATTERN STRUCTURE DEFINTION (PSD). The syntax of the PSD for the navigation pattern looks like as shown in Figure 20. For the declaration of the pattern the following rules must be followed:

- Each Node must have a declaration that starts with the word "ELEMENT".
- The child nodes for a node must be embedded within parentheses – "( )".
- Two or more child nodes can be declared separated by a comma.
- A comment can be added by starting the line with "#" sign.

```
#Navigation Pattern
ELEMENT Tree (BODY)
ELEMENT BODY (DIV)
ELEMENT DIV (UL)
ELEMENT UL (LI+)
ELEMENT LI (A?)
ELEMENT A (TEXT)
```

Figure 20. Sample PSD of Navigation Pattern

The following Table 2 shows the type of attributes that can be used in the declaration of the node for setting the property of the node.

Table 2. Attributes Used for Node Property

Attribute	**Description**
#	Used for adding comments to the PSD.
+	An element ending with '+' must appear at least 5 times.
?	An element ending with '?' is considered as optional, can appear 0 or 1 time.
	An element without any ending character is considered as mandatory element which means it must exist.

If the line in the PSD file is added with '#' character at the start of the line, then it is treated as comment and the line is ignored for the implementation.

4.2.2 Rules Considered

An element defined with '+' property sign must appear at least five times to become a matching pattern. This rule is considered in order to avoid tree structure matches that do not actually belong to the pattern. By adding this rule to the algorithm, unnecessary pattern matches were avoided. But it does have a disadvantage, where some of the actual patterns that are defined with less than five occurrences are not considered as pattern matches. The minimum number of occurrences for the node with '+' is considered as FIVE because during the analysis of the websites, it has been observed that all the websites have at least five links provided for the navigation. But there are some exceptional cases in some websites where the occurrences of the links are less than FIVE. By considering FIVE as a threshold value, the pattern discovery can be achieved efficiently.

For example, consider a node LI defined with "+" attribute a pattern match is found if the LI node appears five times or more. If the actual pattern has only four occurrences of the LI node, then it will be ignored.

4.2.3 Assumptions

For the implementation of the proposed solution, the following assumptions are taken into consideration:

- An optional node must have only one child
    - For eg., ELEMENT A (STRONG?)

      ELEMENT STRONG (TEXT)

  Here the assumption is STRONG must have only one child for it.
- The child node of the Optional Node must not be an Optional.

4.2.4 Pattern Match Finder Tool Design

This section explains the design of the pattern match finder tool for the proposed solution. The design of the tool starts with gathering the requirements that are needed for the development of the tool. The following are the requirements that the tool must meet.

- The purpose of the tool is to find the matched pattern structures in the given input HTML file.
- The tool must be able to accept the parse tree structure which acts as input.
- The tool shall go to the location of the common pattern files that are derived manually and must be able to read the files.
- From the list of common pattern files from the provided location, the tool must be able to find a match in the input file provided.
- The tool must list all the matched patterns from the common patterns derived.
- The tool shall display the patterns matched for the corresponding pattern.

A tool has been designed which uses the algorithm that compares the desired pattern structure and provided input file. The architecture of the Parse Tree Comparison approach is as shown in the Figure 21 below.

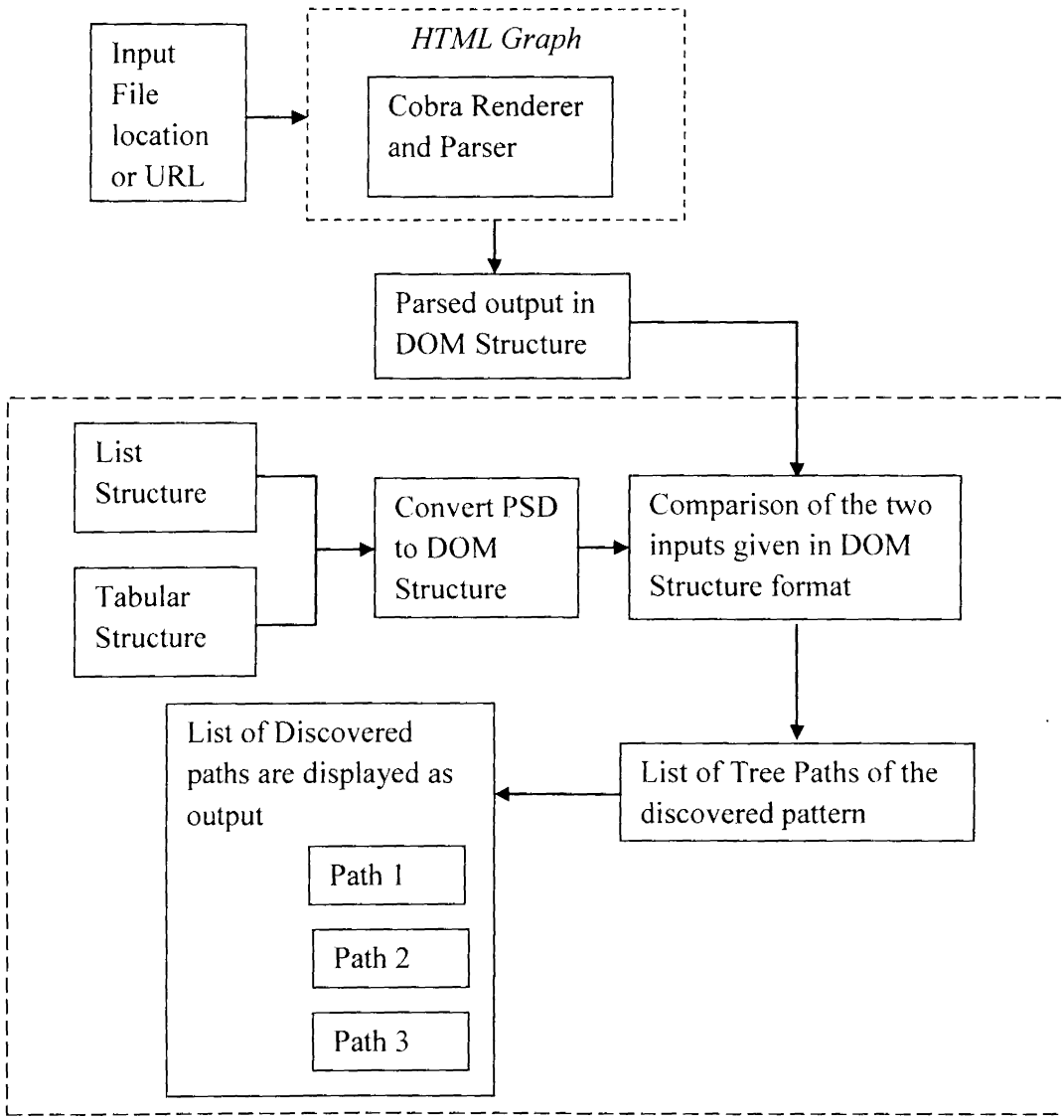

Figure 21. Design of Pattern Discovery Architecture

The parse tree output generated by the HTML Graph tool is used for the comparison to find the desired pattern structure. For the proposed Pattern Match Finder tool, the following are provided as input:

- The Parse Tree output from the HTML Graph tool in DOM structure format.
- Location of the common patterns analyzed based on the manual summarization.

The program implementation looks for each of the PSD files in the common patterns located in the location provided. The program fetches the PSD file of List and Table structure and converts them into DOM Structure format. This is given as input for traversing through the input parse tree generated by the HTML Graph. Once the pattern match is found, then the system gets the path of the tree from root node to leaf. This process repeats till the end of the parse tree. Once all the files are processed, then the list of matched PSD files is noted.

4.2.5 Tool Features

The algorithm proposed and the tool implemented can be used for different websites belonging to the online shopping category. The tool looks for the common pattern matches, and if there is a match then the tool extracts the matched patterns. Along with the pattern match technique, the proposed algorithm has the following additional features:

- The algorithm accepts the common patterns in PSD structure format. As PSD supports the optional nodes, the websites that share similar structure can be represented using optional ("?") property. This makes the tool more generic for different websites that have similar semantic structure implementation.

  Example: Consider the following sample structure:

  Website-A: [Tree, BODY, CENTER, DIV, UL, LI, A, TEXT] and

  Website-B: [Tree, BODY, DIV, UL, LI, A, TEXT].

  By making the CENTER node optional in the PSD, the pattern match can be found in both the websites. The sample PSD looks as below.

  Sample PSD: [Tree, BODY, CENTER?, DIV, UL, LI, A, TEXT]

- The proposed algorithm supports the partial matches, where the tool finds the pattern match if the structure matches till the parent node of the node that has "+" attribute. This makes the tool efficient in matching the partial matches.

  Example: Consider the following sample structure:

  Website-A: [Tree, BODY, CENTER, DIV, UL, LI, A, IMG] and

  Sample PSD: [Tree, BODY, DIV, UL, LI+, A, TEXT].

  The algorithm finds the match till the path reached UL node and returns the path as a match. By this the partial matches can be achieved.

- The proposed algorithm is flexible to find the child node match of a parent node, even though the child node is not an immediate child.

  Example: Consider the following sample structure:

  Website-A: [Tree, BODY, DIV, DIV, CENTER, B, UL, LI, A, TEXT] and

  Sample PSD: [Tree, BODY, DIV, UL, LI+, A, TEXT].

  The algorithm finds the match by ignoring the nodes between DIV and UL. This helps the algorithm to find a pattern match for different websites that have a similar semantic structure behind the navigation pattern.

- As long as the order of the nodes in the given PSD is matched with the extracted path from the input, the tool finds the match even though there are additional nodes in the input that are not declared in the PSD.

  Example: Consider the following sample structure:

  Website-A: [Tree, BODY, DIV, [UL, IMG], LI, A, IMG] and

  Sample PSD: [Tree, BODY, DIV, UL, LI+, A, TEXT].

In this the DIV node has more than one child - UL and IMG. The algorithm finds the match by considering only the path through UL node for the pattern match. The additional nodes along with the matched nodes are ignored.

4.3 Common Patterns

For the implementation of the proposed solution, the common patterns derived in the Section 3.3 must to be converted to the syntax format described in the above section (4.2.1). The defined patterns are stored in a location and the location is provided for the program to fetch the files.

4.3.1 List Structure

The similarities in the patterns observed in websites like, TOYSRUS, Target, Buy, Bizrate, Amazon, Circuit City and Overstock are summarized. By adding the properties to few nodes, the common pattern that can be used to find the pattern structure among these websites has been derived.

Since the LI node must appear more than one time, "+" attribute is added to the node. And since the SPAN and STRONG nodes are optional nodes, they are followed with "?" attribute. Figure 22 below shows the syntax representation of the list structure.

```
ELEMENT Tree (BODY)
ELEMENT BODY (DIV)
ELEMENT DIV (UL)
ELEMENT UL (LI+)
ELEMENT LI (A)
ELEMENT A (SPAN?,STRONG?)
ELEMENT SPAN (TEXT)
ELEMENT STRONG (TEXT)
```

Figure 22. List Structure

4.3.2 Tabular Structure

The similarities in the patterns observed in websites like NewEgg, Smart Bargains, Thenaturestore and JCPenny are summarized. By adding the properties to few nodes, the common pattern that can be used to find the pattern structure among these websites has been derived.

The DIV, CENTER, and TBODY nodes are treated as optional and hence they are followed with "?" attribute. Figure 23 below shows the tabular structure pattern in the syntax format.

```
ELEMENT Tree (BODY)
ELEMENT BODY (DIV?,CENTER?)
ELEMENT DIV (TABLE)
ELEMENT CENTER (TABLE)
ELEMENT TABLE (TBODY?)
ELEMENT TBODY (TR)
ELEMENT TR (TD)
ELEMENT TD (DIV?)
ELEMENT DIV (UL)
ELEMENT UL (LI+)
ELEMENT LI (A)
ELEMENT A (TEXT)
```

Figure 23. Tabular Structure

5. IMPLEMENTATION AND TOOL VISUALIZATION

This chapter explains the implementation and the visualization of the pattern match finder developed to recognize the pattern structure in the input HTML file. This chapter gives a detailed description of various steps that are involved in the development of the pattern match finder and also some additional steps that include embedding the pattern match finder with the HTML Graph tool.

5.1 Technologies Considered

The following technologies are used for the implementation of the pattern match finder tool. The tool has been implemented using JAVA and Eclipse IDE.

5.1.1 JAVA

JAVA is popularly known object-oriented programming language. JAVA is the most chosen programming language by the software developers because of its versatility, efficiency and portability [9]. Swing, an API provided in JAVA for developing the Graphical User Interface components. JAVA has been used as a core developing language in this paper because the application which is used for developing Cobra HTML Renderer is fully implemented in JAVA. For the GUI Development of the application, Swing API implementation has been used.

5.1.2 ECLIPSE

Eclipse is an open source integrated development environment used for developing software applications. Eclipse is written mostly in JAVA and can be used as IDE for developing JAVA as well as other language by installing corresponding plug-ins [10]. Eclipse is released under the terms of the Eclipse Public License [10]. For this paper, Eclipse is chosen as IDE for the application development.

5.1.3 HTML Graph

HTML Graph is a JAVA based application which converts HTML to DOM Structure. HTML Graph uses the Cobra HTML Renderer and Parser from the Lobo Project for parsing the input HTML file. HTML Input can be provided in two ways:

- By loading the HTML file which is located in the local disk of the system. Once the file is loaded, it is automatically converted and saved as *"file_name.graph."*
- The input can also be provided by entering a URL. The URL is loaded, converted, and saved as *"parsed_URL_name_microsec.graph."*

5.1.4 Cobra: Java HTML Parser

Cobra is a pure Java HTML renderer and DOM parser. Cobra is developed in order to support HTML 4, Javascript and CSS 2 technologies [8]. The Cobra can be downloaded from Source Forge [8]. Cobra is an open source project and is available free of cost. The source code of Cobra is released under the LGPL license [8]. Cobra does not support browser functionalities such as navigation, cookies, HTTP Authentication and so on [8].

The input HTML is processed by Cobra which parses and renders the HTML into DOM structure format. The output DOM structure of the given HTML from Cobra is used as one of the inputs for the Pattern Match Finder tool.

5.2 Pattern Match Finder Tool

For the implementation of the tool, the JAVA Swing API has been used for the GUI development. The HTML Graph tool accepts input file either by providing the absolute path of the HTML file or by providing the URL of the desired website. The output of the HTML Graph tool is a Frame that contains the graph notation of the input file in the left

panel and the browser that navigates the given input on the right panel. Figure 24 below shows the output of HTML Graph Tool.

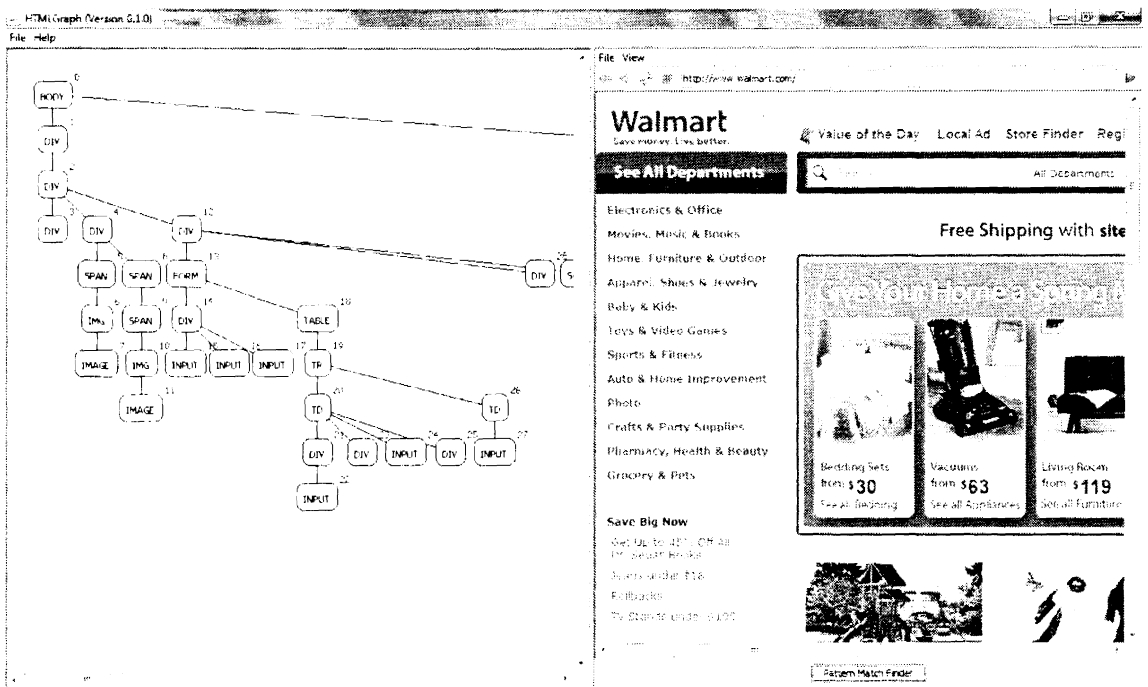

Figure 24. Output of HTML Graph Tool

The pattern match finder is implemented and embedded with the HTML Graph tool. The pattern match finder is initiated when the user clicks on the "Pattern Match Finder" button located at the bottom center of the right panel, which is under the browser panel of the HTML Graph output. On clicking the button, a new Frame is opened that contains parse tree of the HTML file on the left panel and the right panel contains a button "Find Pattern." When the "Find Pattern" button is clicked, the dropdown box is populated with a list of patterns. When the user selects a desired pattern from the list, the pane in the right panel shows the corresponding pattern. Figure 25 below shows the frame of the Pattern Match Finder.

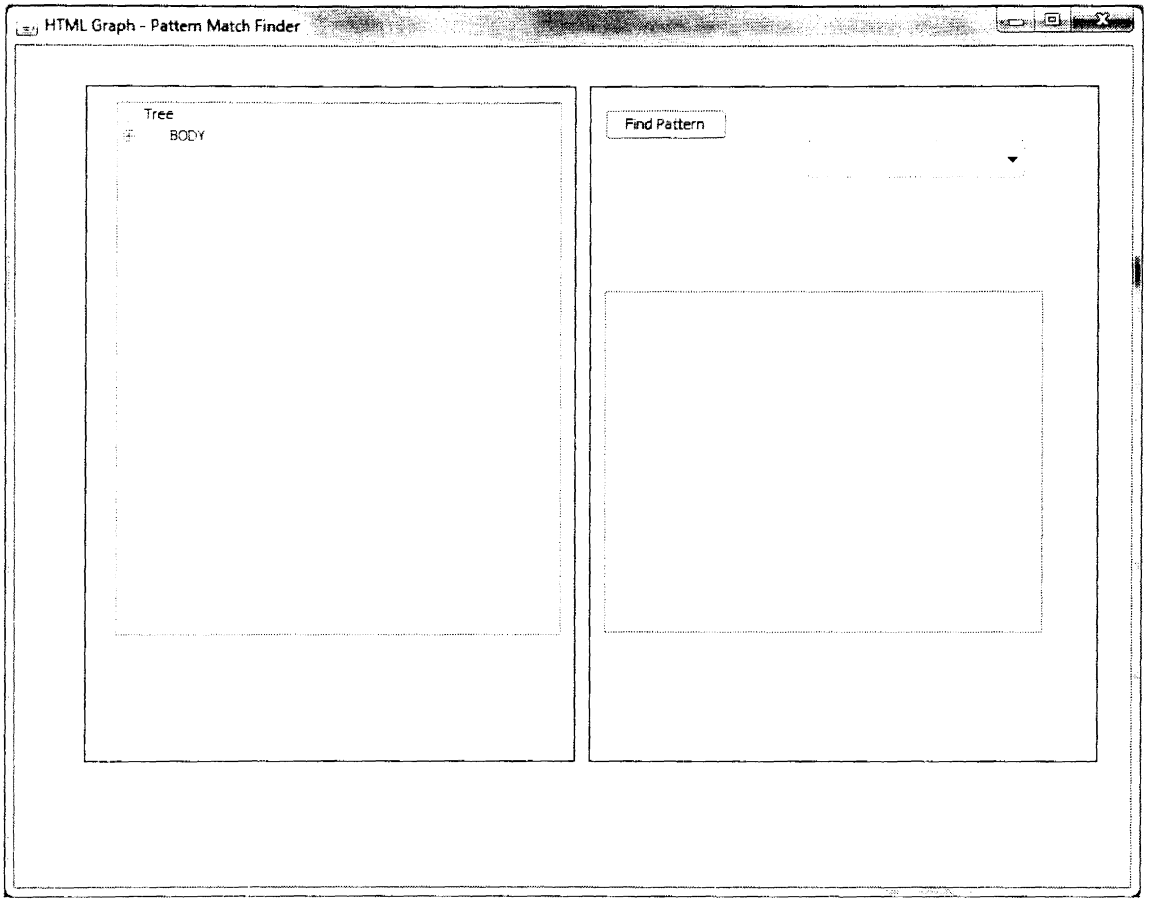

Figure 25. HTML Pattern Match Finder

Once the user clicks on the "Find Pattern" button in the right panel, the following steps are invoked.

5.2.1 Input Parse Tree

The output parse tree of the HTML Graph tool is treated as input for the pattern match finder tool. Hence the parse tree generated by the HTML Graph tool is displayed in the left panel of the Pattern Match Finder Tool. And the matches that are discovered in the parse tree will be displayed in an expanded format.

5.2.2 Common Pattern Files

This step of the tool fetches the list of common pattern files from the location specified. For each file in the list, the program reads the pattern file contents. For each

element specified in the input file, the program creates a corresponding node. The program removes the attribute that is added as a property and returns the name of the node. The program returns a JTree object from the pattern file which is Tree Structure representation of the pattern file as shown in the Figure 26 below.

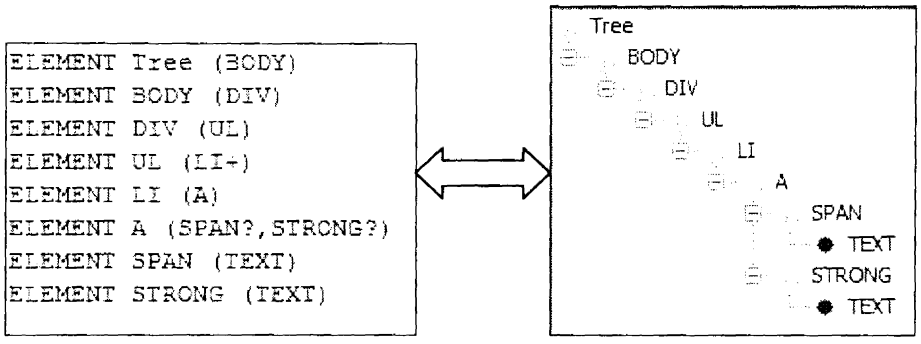

Figure 26. Tree Structure Representation of Pattern

All the node properties that are observed while reading the pattern file are maintained separately for the corresponding nodes. By doing this, the tool can find whether the node is an optional, multiple or mandatory node.

- For each file in the location specified, the tool reads the contents of the file.
- If the line starts with '#' character, then the line is assumed as comment and the line is ignored.
- If the line starts with "ELEMENT" keyword, then the tool reads the parent and child elements by splitting the line.
- For each parent element, the list of child elements that are within the parentheses and separated by commas (",") is fetched.
- The above steps are repeated till all the lines in the file are read.
- For each parent element read from the file, a tree node is created, and the list of the child elements for the parent node is fetched.

- For each child element.
    - Checks if any property has been set, if so, then the tool adds the node to the list of optional nodes or multiple nodes depending on the property.
    - A child node is created by removing the property attribute and adding it to the parent node.

5.2.3 Tree Comparison

This is the crucial step which compares the Input Parse Tree with the JTree generated from Pattern File. The step deals with traversing the Pattern Tree generated from the Pattern File. Based on the nodes found in the pattern tree, the Input Parse Tree is traversed for the comparison.

5.2.3.1 Traverse Pattern Tree

In this step of the program the tool gets the parent-child combination of the Pattern Tree. The flow starts by executing the following steps:

- The root node of the Pattern Tree is passed for the traversing method.
- The child count of the node (initially root node) is fetched.
- For each child node of the node (parent node)
    - Check whether the child node is optional. If optional, then child node of the optional is retrieved as sub-child node
- Once the parent-child combination is obtained, then Traverse Input Parse Tree will be invoked.
- If more than one child exists for a node, then the above steps are repeated for each child node.

- The above steps are repeated until all the nodes of the pattern tree are traversed.

5.2.3.2 Traverse Input Parse Tree

By the time this step gets initiated it is assumed that a parent-child combination of the Pattern Tree has been obtained. The flow starts by executing the following steps:

- The root node of the Input Parse Tree is passed for the traversing method.
- The child count of the node (initially root node) is fetched.
- The program checks whether the current node match with the parent node of the Pattern tree.
- For each child node of the node (parent node)
    - If parent match is found, then:
        - The program checks if the current child node is multi node (with "+" property)
            - If above is true and the child node appears more than or equal to five then, number of pattern matches found is increased by one.
        - The program checks if the current child node matches with the child node of the Pattern Tree.
            - If above is true and if the current child node is the leaf node, then path from root to current node is captured.
- Repeat from step 1 by passing the child node as current node.
- The above steps are repeated until all the nodes of the input parse tree are traversed.

This step of the program gives the path of the matched pattern (from root to leaf) and number of times each node has appeared in the input parse tree.

5.2.4 Display Matched Paths

After executing the Tree Comparison step of the program, the list of pattern files that has the matched combinations in the input parse tree are obtained. The dropdown box on the right panel is populated with the pattern files that have a match (see Figure 26). The user can select the desired pattern to see the pattern structure as well as the matches in the input parse tree.

A program that deals with the presentation of the matched patterns of the input tree has been implemented. In order to provide the user with the matched patterns in the input parse tree, the matched paths are displayed by expanding the path from the root. The number of matches found in the input parse tree is also displayed on the right panel below the drop down. Figure 27 shows the output that is displayed with the matched patterns.

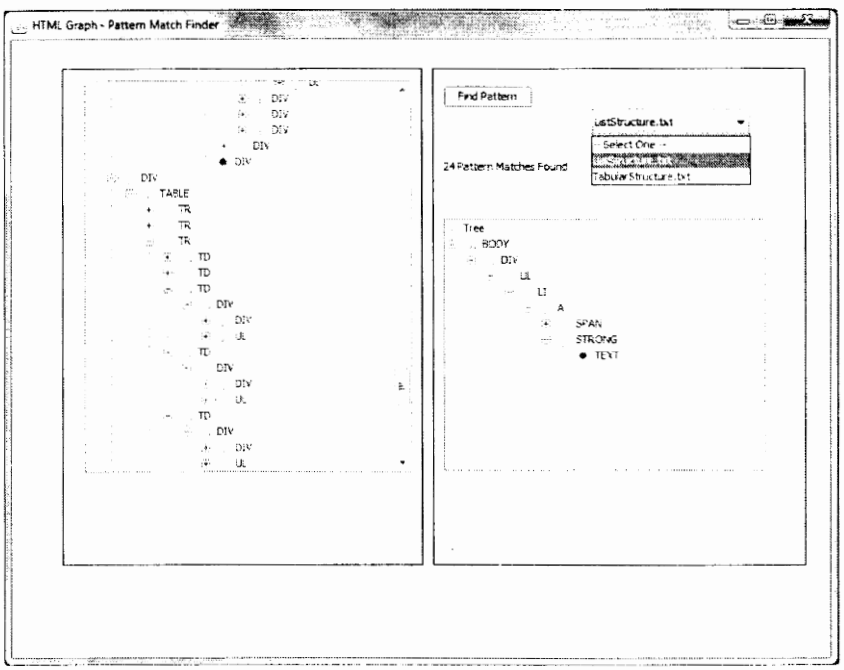

Figure 27. List of Patterns Matched and Expanded Matched Patterns

6. EVALUATION

This chapter explains how the proposed idea implementation is evaluated. To evaluate the Pattern Match finder tool, all the websites that are considered for manual analysis are considered here. For testing the websites, either the URL of the website or static HTML page can be given as input. And the list of common patterns is also provided as input. The following Table 3 shows the list of websites under the Website Name column and the pattern structure found for the corresponding website has been marked.

Table 3. List of Websites and Patterns Matched

S. No	Website Name	List Structure	Tabular Structure
1	Buy	X	
2	NewEgg	X	X
3	Bizrate	X	
4	ThenatureStore		X
5	Target	X	
6	SmartBargains	X	X
7	Amazon	X	
8	ToysRUs	X	X
9	CircuitCity	X	
10	Overstock	X	

To test the correctness of the implemented *Pattern Match Finder* tool, the results are compared with the expected results, derived in section *3.3.4*. The results matched with the expected results and the tool has met the requirements.

6.1 Precision and Recall

For evaluating the performance of the proposed algorithm and the implemented tool, the Precision and Recall values are calculated. The Precision and Recall are defined for this paper based on the evaluation made by T. Hassan [27].

$$\text{Precision} = \frac{\text{Number of correctly retrieved pattern matches}}{\text{Total number of retrieved pattern matches}}$$

$$\text{Recall} = \frac{\text{Number of correctly retrieved pattern matches}}{\text{Total number of actual matches}}$$

"Number of correctly retrieved pattern matches" are the pattern matches that are retrieved by the pattern match tool. "Total number of actual matches" are actual pattern matches that are present in the input which are found manually. "Total number of retrieved pattern matches" are the total pattern matches that are found by tool.

To evaluate the precision and recall, the Home page of the website is derived and given as static HTML input to the tool. All the websites chosen for manual summarization (*Section 3*) are considered here to measure the precision and recall, the following Table 4 summarizes the results.

Table 4. Precision and Recall Calculations for Websites Listed

S. No	Website Name	Calculations	List Structure	Tabular Structure
1	Buy.com	Rcorrect	31	
		Atotal	33	
		Rtotal	36	
		Recall	0.939393939	
		Precision	0.861111111	
2	NewEgg.com	Rcorrect	25	25
		Atotal	27	27
		Rtotal	25	25
		Recall	0.925925926	0.925925926
		Precision	1	1

Table 4 (Continued)

3	Bizrate.com	Rcorrect	16	
		Atotal	20	
		Rtotal	22	
		Recall	0.8	
		Precision	0.727272727	
4	ThenatureStore.com	Rcorrect		16
		Atotal		24
		Rtotal		16
		Recall		0.666666667
		Precision		1
5	Target.com	Rcorrect	28	
		Atotal	38	
		Rtotal	31	
		Recall	0.736842105	
		Precision	0.903225806	
6	SmartBargains.com	Rcorrect	9	9
		Atotal	12	12
		Rtotal	9	9
		Recall	0.75	0.75
		Precision	1	1
7	Amazon.com	Rcorrect	12	
		Atotal	19	
		Rtotal	12	
		Recall	0.631578947	
		Precision	1	
8	ToysRUs.com	Rcorrect	18	8
		Atotal	25	8
		Rtotal	19	8
		Recall	0.72	1
		Precision	0.947368421	1
9	CircuitCity.com	Rcorrect	12	
		Atotal	16	
		Rtotal	12	
		Recall	0.75	
		Precision	1	

Table 4 (Continued)

10	Overstock.com	Rcorrect	14	
		Atotal	15	
		Rtotal	14	
		Recall	0.933333333	
		Precision	1	

(where *Rcorrect* is the number of correctly retrieved patterns; *Rtotal* is the total number of retrieved pattern matches; *Atotal* is the total number of actual matches)

6.2 Evaluation Using Additional Websites

Along with the websites considered above, the tool has been tested by providing ten additional different websites belonging to the online shopping category. For all the websites chosen, the tool was able to find a pattern match from the two derived patterns. The following Table 5 shows the list of additional websites used for evaluating the tool.

Table 5. Additional Websites Used for Evaluating the Tool

S. No	Website Name	Calculations	List Structure	Tabular Structure
1	BestBuy	Rcorrect	24	
		Atotal	34	
		Rtotal	28	
		Recall	0.705882353	
		Precision	0.857142857	
2	Lowes	Rcorrect	11	
		Atotal	11	
		Rtotal	12	
		Recall	1	
		Precision	0.916666667	
3	AmericanEagle	Rcorrect	15	
		Atotal	28	
		Rtotal	16	
		Recall	0.535714286	
		Precision	0.9375	

Table 5 (Continued)

4	Meritline	Rcorrect	6	6
		Atotal	6	6
		Rtotal	6	6
		Recall	1	1
		Precision	1	1
5	Quill	Rcorrect	33	
		Atotal	45	
		Rtotal	33	
		Recall	0.733333333	
		Precision	1	
6	Kohls	Rcorrect	2	
		Atotal	2	
		Rtotal	3	
		Recall	1	
		Precision	0.666666667	
7	Ebay	Rcorrect	10	
		Atotal	15	
		Rtotal	11	
		Recall	0.666666667	
		Precision	0.909090909	
8	Walgreens	Rcorrect	15	6
		Atotal	16	6
		Rtotal	15	6
		Recall	0.9375	1
		Precision	1	1
9	SunglassHut	Rcorrect	10	
		Atotal	11	
		Rtotal	12	
		Recall	0.909090909	
		Precision	0.833333333	
10	BarnesAndNoble	Rcorrect	50	
		Atotal	53	
		Rtotal	50	
		Recall	0.943396226	
		Precision	1	

(where Rcorrect is the number of correctly retrieved patterns; Rtotal is the total number of retrieved pattern matches; Atotal is the total number of actual matches)

6.3 Evaluation Results

The following Figure 28 shows the Average Precision and Recall and Standard Deviation of the two common patterns for twenty websites chosen for the evaluation.

From the Figure 28, it can be observed that the Average Precision for List structure is 0.925 approximately and the Average Recall for List structure is 0.8209. The Average Precision for Tabular structure is 1 and the Average Recall for Tabular structure is 0.918 approximately. For the observed Precision and Recall of the List and Tabular Structure patterns, the Standard Deviation has been calculated and is shown in Figure 28 below.

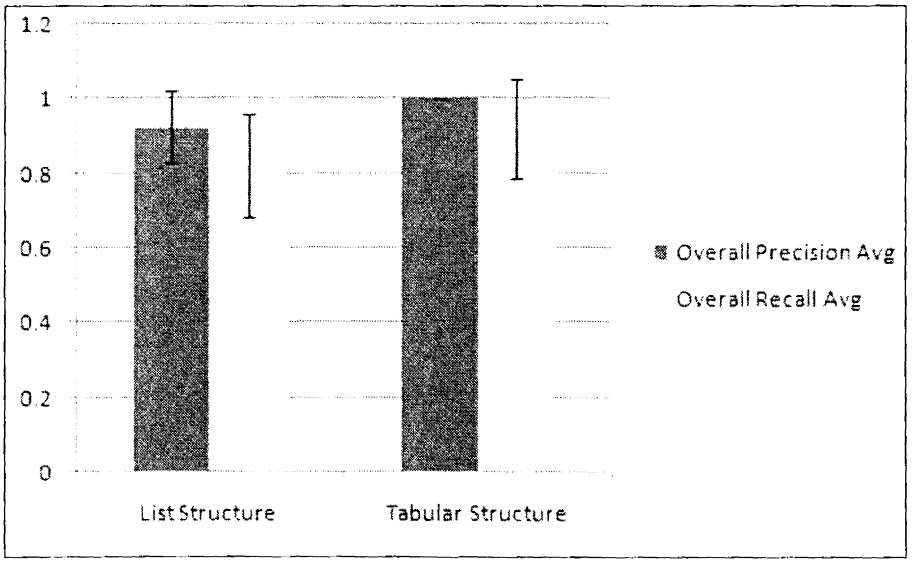

Figure 28. Average Precision and Recall

For measuring the accuracy, F-1 score has been calculated by using the harmonic mean of precision and recall [28].

$$\text{Accuracy} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

By using the above formula, the Accuracy of the implemented tool is 0.9135. Thus, it is found that pattern match tool achieves about 91.35% accuracy for extracting the patterns across multiple web pages under the online shopping category. The performance

of the pattern match tool can be further evaluated to test the pattern structures of more web pages under different domains of the websites.

7. CONCLUSION AND FUTURE WORK

The main objective of this paper is to summarize the web patterns manually and derive the common patterns. Also one of the goals of the paper is to design and develop a tool which can be used to find the pattern matches in the parser tree based on the common patterns derived. The work in this paper has clearly met the intended objectives of this paper. The paper has presented a new structure called Pattern Structure Definition (PSD) which has been extended from DTD for representing the common pattern structure. Tree comparison approach has been presented to discover the pattern matches in the parse tree based on the list of common patterns summarized manually.

The evaluation and experiment results indicate that the *Pattern Match Finder* tool finds the pattern matches for navigation in the online shopping websites. The tool has not only been tested for the list of websites chosen for analysis but also tested on additional websites belonging to the online shopping category. The tool was able to find a pattern match from the common patterns derived. The precision, recall and accuracy calculations show the effectiveness of the tool. Results show that an accuracy of 91.35 % has been achieved by the tool.

The pattern match discovery approach discussed in this paper, however, still have some limitations. First, this approach cannot discover the pattern match if the number of items in the list is less than five. To consider a pattern as a matched pattern, the number of occurrences of the node with '+' attribute must be at least five or more. If a true pattern which is supposed to be a pattern match has only four or less occurrences, then it will not be discovered as a pattern match. Second, the pattern discovery is mainly performed on the output generated by the Cobra Renderer and Parser tool. There are some web pages where

the Cobra tool cannot recognize the source code. In such cases, the implemented tool throws exception and the entire process is terminated. For example, for the website staples.com, a DOM Exception is thrown with the message *"INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified."* The reason behind the exception is due to the source code of the web page containing a metadata defined with characters that cannot be processed by the Cobra. In this example, the invalid code that caused the error is as shown in Figure 29 below.

```
<meta http-equiv="Pics-Label" content="(pics-1.1
"http://www.icra.org/ratingsv02.html" comment "ICRAonline EN v2.0" l gen true for
"http://staples.com" r (nz 1 vz 1 lz 1 oz 1 cb 1) "http://www.rsac.org/ratingsv01.html" l
gen true for "http://staples.com" r (n 0 s 0 v 0 l 0))" />
```

Figure 29. Sample Code for Invalid Exception

The cause for the exception could be any invalid code. The Cobra cannot recognize such type of code. Hence, this is considered as one of the limitations for the work done in this paper since the output of the Cobra tool is accepted as one of the inputs for the pattern discovery.

This work can be extended to other patterns like product results patterns that occur in the online shopping category, as well as to other domains like online cars searches and online travel reservations.

REFERENCES

[1] A. Arasu and H. Garcia-Molina, *"Extracting structured data from web pages"*, Proceedings of the 2003 ACM SIGMOD International Conference on Management Of Data, pp: 337-348.

[2] Z. Zhang, B. He, K. Chen-Chuan Chang, *"Understanding Web query interfaces: best-effort parsing with hidden syntax"*, Proceedings of the 2004 ACM SIGMOD International Conference on Management Of Data, June 13-18, 2004, Paris, France.

[3] Mohammed J. Zaki, *"Efficiently mining frequent trees in a forest"*, Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Datamining, 2002.

[4] David Lo, Siau-Cheng Khoo, *"Mining Patterns and Rules for Software Specification Discovery"*, Proceedings of the VLDB Endowment, Vol 1, No. 2, Aug 2008.

[5] F. Mandreoli, R. Martoglia, P. Zezula, *"Principles of Holism for sequential twig pattern matching."* The VLDB Journal – The International Journal on Very Large Data Bases, Vol 18, No. 6, Dec 2009, pp: 1369-1392.

[6] C Ranganathan, S.Ganapathy, *"Key dimensions of business-to-consumer web sites"*, Information and Management, Vol 39, Issue 6, May 2002.

[7] C. Park, Young-Gul Kim, *"Indentifying Key Factors affecting consumer purchase behavior in an online shopping context"*, International Journal of Retail & Distribution Management, Vol 31, No. 1, 2003, pp: 16-29.

[8] Cobra HTML Renderer and Parser, http://lobobrowser.org/cobra.jsp, Date Visited: 03/01/2011.

[9] Learn about Java Technology, http://www.java.com/en/about/ Date Visited: 03/01/2011.

[10] Eclipse IDE, http://en.wikipedia.org/wiki/Eclipse_(software) Date Visited: 03/01/2011.

[11] DTD, http://en.wikipedia.org/wiki/Document_Type_Definition Date Visited: 03/05/2011.

[12] DTD, http://www.w3schools.com/dtd/dtd_intro.asp Date Visited: 03/05/2011.

[13] XML Schema, http://en.wikipedia.org/wiki/XML_schema Date Visited: 03/05/2011.

[14] XML Schema, http://www.w3schools.com/schema/schema_schema.asp Date Visited: 03/05/2011.

[15] J. Y. Hsu and W.-T. Yi. *"Template-based information mining from html documents"*. In Proceedings of AAAI-97, July 1997, pp: 256–262

[16] http://www.newegg.com, Date Visited: 03/30/2011

[17] http://www.amazon.com, Date Visited: 03/30/2011

[18] http://www.buy.com, Date Visited: 03/30/2011.

[19] http://www.overstock.com, Date Visited: 03/30/2011

[20] http://www.target.com, Date Visited: 03/30/2011

[21] http://www.smartbargains.com, Date Visited: 03/30/2011

[22] http://thenaturestore.com, Date Visited: 03/30/2011

[23] http://www.bizrate.com, Date Visited: 03/30/2011

[24] http://www.circuitcity.com, Date Visited: 03/30/2011

[25] http://www.toysrus.com, Date Visited: 03/30/2011

[26] D. C. Reis, P. B. Golgher, A. S. Silva and A. F. Laender, *"Automatic web news extraction using tree edit distance"*, In Proceedings of 13th International Conference on World Wide Web, May 17-20, 2004, New York, NY, USA, pp:502-511

[27] T. Hassan, *"Towards a common evaluation strategy for table structure recognition algorithms"*, In Proceedings of the 10th ACM symposium on Document Engineering 2010.

[28] Chen, J. and Xiao, K. *"Perception-oriented online news extraction"*, In Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries, June 16 – 20, ACM, New York, NY, 2008, pp: 363-366.

[29] C.-H. Chang, C.-N. Hsu, S.-C. Lui, *"Automatic information extraction from semi-structured Web pages by pattern discovery"*, Decision Support Systems - Web retrieval and mining, Vol 35, Issue 1, April 2003, pp: 129-147

[30] M. Cosulschi, A. Giurca, B. Udrescu, N. Constantinescu and M. Gabroveanu, *"HTML Pattern Generator – Automatic Data Extraction from Web Pages"*, SYNASC 2006, In Proceedings of the 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp:75-78.

[31] http://money.cnn.com/magazines/fortune/fortune500/2011/full_list/, Date Visited: 06/12/2011.