4.1. Introduction

The booming of e-commerce in recent years began with the introduction of Automatic Teller Machines (ATM) and Electronic Data Interchange (EDI) [Kalakota and Whinston, 1997]. E-commerce, a new and smart way of doing business, is a massively crowded online market, which has partially or fully affected everyone’s life. Online shopping has revolutionized the shopping behaviour of consumers. In India Amazon, SnapDeal and Flipkart are now treated as new showrooms. These websites advertise and sell different kinds of products and reach to a large number of users despite of time and distance limitations. With the growing competition in the e-commerce market, Customer’s information is important to merchant. So, need is to understand the customer’s need in this era. The extraction of product information from multiple e-commerce sites is one of the practical applications in the Web mining realm. As we know there are billions of the websites on the Web, which differ in their underlying structure, and visual style, but the beauty of the e-commerce websites is that, upon a webpage request the information about the products is embedded into the webpage. The information about the products is actually stored in the underlying database and is automatically embedded into the web pages using scripts. In e-commerce we mine online shopping stores as much attention is being paid to price comparison, customer attraction, customer retention, etc.
Extracting such information from the websites can help business organizations in answering questions like identifying profitable customers and identifying those web sites from where the customers buy frequently.

The full description of a particular product in a webpage is known as data record. These data records are important for web shoppers (users purchasing online). But, due to a large number of online shopping sites, users are compelled and forced to spend their time and effort to find the products of their choice. To extract product information from e-commerce websites, one of the main approaches is to write a wrapper for a particular website, but manually writing wrappers is difficult and even impossible at a large scale. [Ansari, 1997] proposed an integrated architecture for an e-commerce system with Data Mining. Their system can dramatically reduce the pre-processing, cleaning and data understanding, in knowledge discovery projects. [Petra Perner et al., 2003] proposed an architecture according to users needs and preferences by extending an e-shop into an intelligent e-marketing and selling platform. Their method uses two types of Data Mining techniques, namely classification and clustering. In e-commerce sites, to predict the users behaviour, a new approach was proposed by [Vallamkondu and Gruenwald, 2003]. In order to predict the purchase and traversal behaviour of future users, the proposed approach involves extracting of information from integrated data of past users. [Lee et al., 2004] propose a model for e-commerce known as integrated path traversal patterns (IPA) and association rules for web usage mining. This IPA model takes into consideration both the traversing and purchasing behaviour of the customers at the same time. This model not only take the traversal forward information of the user, but also takes into account the users backward traversal information, which makes the model accurate and correct for capturing users purchasing and traversal behaviours.
[Satokar and Gawali, 2010] present a personalization system, which depends on features extracted from hyperlinks, for web search. Their personalization system which uses a weighted URL rank algorithm can help users not only to get relevant webpages, but also domains the user is interested in it. [Kiruthika et al., 2011] discusses the use of association rules in discovering patterns in web usage mining. They propose a system in which they preprocess the web server logs, cluster the closely related transactions, and discover the association rules. Their proposed system can help the website designers to improve their website design. [Todi et al., 2012] developed an application which extracts information from e-commerce websites for classification of data in order to benefit both customers as well as companies. They use Naïve Bayes and Decision Trees, the two most popular supervised algorithms for classification and compared them. The results show that Decision Trees perform better than Naïve Bayes. Using such kind of applications, customers can understand the qualities of the available products and a competitor can understand how their competitors are priced.

In the field of Web mining, the extraction of information from Web has become very vital in order to survive in this competitive world. Every business organization need to perform timely analysis of market. The main aim of this chapter is to extract and integrate the data records from multiple e-commerce websites so that web shoppers will be convinced and benefited.

As we know web pages are written in Hyper-Text Markup Language but HTML can only display the information to users and do not tell them what it means, which may hinder the data extraction process. But fortunately e-commerce websites usually obey some sort of regularities as follows:
I). E-commerce websites are automatically generated from an underlying database.

II). Data Records in such sites are displayed in some repetitive fashion.

III). A list of data records constitutes a data region. Near a data region is paging information which can be used to identify the data region.

IV). A data record hold information about a particular product such as product image, product name and product price and are known as data items of the product as can be seen from figure 4.1. These data items in various data records of a particular data region share similar properties such as font colour, font style etc.

4.2. The Proposed Architecture

We propose an automatic method for extraction of structured web data records from multiple e-commerce websites. Our proposed method introduces a new extraction technique based on the similarity between the DOM tree tag node patterns. The extracted data records can be used further for mining purposes and other manipulations. It consists of five phases viz. (1) Crawling (2) Pre-Processing (3) Structural Similarity Calculation (4) Clustering and (5) Wrapper generation. These five phases are called sequentially by the main method. We will explain below each module individually. The overall algorithm of our proposed method and its high level architecture are shown in figures 4.1 and 4.2 respectively.
Algorithm: Extraction of structured data from multiple websites

Input: Set of HTML files (WebPageFile) of web documents.
Output: data records

1. Begin
2. for each WebPageFile
3. Crawl all webpages from WWW and store them into local directory.
4. Cleanup HTML code and generate DOM trees of webpages.
5. Generate Column Similarity Matrix.
6. Cluster the structurally similar webpages.
7. Extract and store information.
8. endfor
9. End

Figure 4.1: Algorithm for Extraction of Structured Data from Multiple Websites

Figure 4.2: Block Diagram of Proposed Method
4.2.1. Web Crawling
Web Crawlers are the programs or automated scripts, which browse the WWW in a methodical and automated manner. Also known as ants, spiders or bots. Crawlers are the programs that resolve the problem of IR (Information Retrieval) more effectively and efficiently from the WWW. [Chakrabarti, 1999] was the first to introduce web Crawling and the best web crawler with a best fit strategy was proposed by [Choj et al., 1998]. Based on the type of application web crawlers can be classified into two main types:

I). General Crawler
II). Focused Crawler

I). General Crawler: In this type of crawling, crawling starts with a given seed of URL’s. The URL’s are fetched and stored in a data structure known as URL queue. Various threads are executed concurrently, where each thread receives a URL from the queue and fetches the required web page from the server. Lastly the web page is then parsed in order to extract hyperlinks. These hyperlinks are also added to a URL queue for further processing. The basic structure of a crawler is shown in figure 4.3:

Figure 4.3: Structure of General Crawler
II). Focused/Topical Crawler: In this type of crawler, the crawler downloads only those web pages that are relevant to a pre-defined query/topic, while traversing the Web. If given a seed of URL’s, it recursively tries to find similar pages on the Web in a better manner. To develop a focused crawler we have three approaches, namely Classification based, Semantic based and Link analysis based on which Crawling policy depends. Figure 4.4 shows a typical Focused crawler.

![Figure 4.4: Structure of Focused Crawler](image)

According to our need we create a crawler in JavaScript containing two modules only, finder and downloader. The finder receives the URL’s of the web pages and gives them to the Downloader module which in turn downloads the web pages and stores them. The crawler maintains two
queues, visited queue (VQ), which stores the visited URL’s and unvisited queue, (UQ) which stores the unvisited URL’s. Initially the crawler is given a seed URL where from the processing starts. The finder module extracts all the links from the current URL, pushes the current URL to the VQ, passes it to the downloader and also pushes all other links to the UQ. The finder repeats the process recursively and the downloader, downloads the web pages and stores them in local machine as shown in figure 4.5.

![Block Diagram of Crawler Module](image)

**Figure 4.5:** Block Diagram of Crawler Module

### 4.2.2. Preprocessing of web pages

To clean the web pages, we use HTMLCleaner-2.2 software. It is an open source HTML parser written in Java. If the webpage is not clean that means if it contains bad and ill formatted tags, Document Object Model (DOM) tree cannot be built properly. The cleansed web page is given to dom4jmodule which then constructs the DOM tree for that webpage as shown in figure 4.6. The DOM is a very flexible and very powerful way of working with XML data through a flexibility point of view than any other methods. The DOM was adopted by the W3C as a standard in 1998. It provides a standardized way of discovering, manipulating and changing the content of a document using programmatic techniques such as
JavaScript, C or C++. The DOM is platform-independent, browser independent and language neutral. It represents a document as a tree structure and allows access to the objects in the tree using a set of programmatic functions and properties. All XML content such as elements, comments, processing instructions, data sections etc. are treated as objects in this model. These objects are known as “Nodes” and these nodes have parent-child relationship in the document tree.

![Diagram of Webpage Pre-processing](image)

**Figure 4.6:** Webpage Pre-processing

### 4.2.3. Calculating Structural Similarity of web pages:

In order to calculate structural similarity we use the Levenshtein edit-distance algorithm to compute similarity between two web pages \((P_i)\) and \((P_j)\). We do not take into consideration the text of the web pages while calculating similarity as we are only interested in their structural similarity. Initially we convert each DOM tree of a web page into a string \(S\). Accordingly, we represent
two web pages as $S_i$ and $S_j$ respectively. While transforming a DOM tree of a web page into a string, DOM tree is traversed in depth first order and each text node is replaced by a text tag. Consider two web pages ($P_i$) and ($P_j$) as shown in figure 4.7.

Figure 4.7: DOM Trees of Two Webpages

These pages are represented by two strings as shown below:

$S_i: HTML \ HEAD \ TITLE \ TEXT0 \ BODY \ TEXT1$
$S_j: HTML \ HEAD \ TITLE \ TEXT0 \ BODY \ DIV \ TR \ TD \ TEXT1$

Edit-distance between two web pages ($S_i$) and ($S_j$) is a function that calculates the minimum number of operations required to transform one string to another. These operations are of three kinds viz. insert a character, delete a character and replace a character. The equation (4.1) is used to
calculate the edit-distance between \(S_i\) and \(S_j\). We normalize the equation (4.1) by dividing it by \(\text{Max} (|S_i|, |S_j|)\) and subtracting the result from 1 as shown below in equation 4.2.

\[
\text{Ed} (S_i, S_j) = \min \{\text{Ed} (S_i, S_{j-1}) + 1, \text{Ed} (S_{i-1}, S_j) + 1, \text{Ed} (S_{i-1}, S_{j-1}) + S\}
\]

where \(S = 1\) if \(S_i \neq S_j\) and \(S = 0\) if \(S_i = S_j\)  

(4.1)

\[
\text{Ned} = 1 - \frac{\text{Ed}(S_i, S_j)}{\text{Max} (|S_i|, |S_j|)}
\]

(4.2)

In the structural similarity phase the set of web pages \(\{P_1, P_2, ..., P_n\}\) can be represented as a string \(\{S_1, S_2, ..., S_n\}\). We then compute the similarity square matrix \(M_{n \times n}\) where \(n\) is the number of web pages and \(M_{ij}\) represents the Normalized edit distance similarity (Ned) between \(P_i\) and \(P_j\). For calculating the similarity between two pages we would use column similarity which seems more robust. We can calculate the column similarity by using the following equation 4.3. Finally a column similarity matrix is generated where each cell denoted by \(\text{pos}_{ij}\) is obtained as \(\text{Colsim} (P_i, P_j)\).

\[
\text{Colsim}(P_i, P_j) = 1 - \frac{\sum_{k=1}^{n} |\text{pos}_{ik}| - |\text{pos}_{jk}|}{n}
\]

(4.3)

4.2.4. Clustering: Clustering is the process of grouping a set of objects into classes of similar objects. An ideal clustering algorithm is that which clusters, web pages in such a way that documents within a cluster should be similar and document’s from different clusters are dissimilar. We would use Hierarchical Agglomerative clustering to group the web pages of the same type. There are two forms of Hierarchical clustering viz. Divisive and Agglomerative. In divisive clustering all data items are put in a single cluster. The cluster is divided into sub-clusters based on the similarity or distance. This process is repeated iteratively till a desired clustering
structure is obtained. On the other hand agglomerative clustering is its reverse process where each data item represents a single cluster. The clusters are merged based on the similarity or distance. This process is repeated until a desired clustering structure is obtained [Jain et al., 1999]. Hierarchical Agglomerative Clustering (HAC) was first proposed by S C Johnson in 1967 [Johnson, 1967]. It uses a bottom-up approach to create clusters in contrast to Hierarchical divisive clustering, which uses a top-down approach. We would use HAC algorithm to group the web pages. In this method we assign each element to its own cluster, then by computing the similarity between each of the clusters, we join the most similar clusters resulting into as many clusters as preferred. That means we cluster those groups where inter-element similarity is high. This inter-element of a set Ø, is measured by the auto - similarity formula. Two thresholds namely auto-wise similarity threshold denoted by t1 and pairwise similarity threshold t2 are measured before forming a new group. The algorithm starts with a collection P of n singleton clusters where each cluster represents a single web page. We repeat the algorithm till only one cluster is left. Initially we find a pair of clusters (cᵢ, cⱼ) within which inter-element similarity is high. The inter-element similarity of a set, S(∅) is calculated by following equation 4.4. Then merge the clusters (cᵢ, cⱼ) into one new cluster (cᵢ + j) and remove (cᵢ, cⱼ) from collection P and add cluster (cᵢ + j) to it. Repeat the process till a Hierarchical tree known as Dendrogram is formed. The pseudo code for bottom-up clustering is given in figure 4.8.

\[ S(∅) = \frac{2}{|∅||∅|-1} \sum_{Pᵢ, Pⱼ \in ∅} \text{Colsim} (Pᵢ, Pⱼ) \]  

(4.4)
Algorithm: Clustering

Input: pageset, \( t_1, t_2 \)

1. Begin
2. Suppose \( D_{ij} \) be the distance between \( P_i \) and \( P_j \) in the pageset
3. Let \( C \) be set of groups
4. Initialize each page to a group
5. do while \( (C \geq 1) \)
6. Choose two webpages \( w_1, w_2 \in C \) having high auto-similarity
7. Compute \( S(w_1 \cup w_2) \)
8. if \( (S(w_1 \cup w_2) > t_1 \&\& \text{Colsim}(P_i, P_j) > t_2) \) then
9. Remove \( w_1 \) and \( w_2 \) from \( C \)
10. Let \( \emptyset = w_1 \cup w_2 \)
11. Insert \( \emptyset \) into \( C \)
12. else
13. Break
14. endif
15. endwhile
16. return \( C \)
17. End

Figure 4.8: Clustering Algorithm

4.2.5. Wrapper Generation: According to [Laender et al., 2002], data can be extracted from web sources by writing specialized programs known as Wrappers. Approaches [Atzeni and Mecca, 1997; Sahuguet and Azavant, 1999], were based on manually writing wrappers. But writing and maintaining such wrappers are difficult to maintain and are labour-intensive. That is why generating wrappers automatically were presented to address the wrapper maintenance problem. For automatic wrapper generation and data extraction there are two types of approaches viz. grammar induction approach and website structure based approach. After clustering of web pages of the same type, we use a structure based
approach for automatic generation of wrappers. The Algorithm for Wrapper Generation is given in figure 4.9.

**Algorithm:** Wrapper Generation

\[ \text{Input: pageset, } t3 \]

1. \textbf{Begin}
2. \hspace{1em} \textbf{Set} \text{temp} \leftarrow \text{Page having high auto-similarity and maximum potential nodes}
3. \hspace{1em} \textbf{Remove} the selected template from the pageset
4. \hspace{1em} \textbf{Apply} sorting technique on pageset in descending order
5. \hspace{1em} \text{Setcount}=1
6. \hspace{1em} \textbf{do while(each page in pageset)}
7. \hspace{2em} \text{Ned(template, page)}
8. \hspace{2em} \text{S = getMatchNodes(template, page)}
9. \hspace{2em} \textbf{for} each pair \((n1, n2)\) in \text{S}
10. \hspace{3em} \text{n1.Setcount} = n2. \text{Setcount}+1
11. \hspace{3em} \text{Add nodes to template when not mapped (n1,n2)}
12. \hspace{2em} \textbf{endfor}
13. \hspace{1em} \textbf{endwhile}
14. \hspace{1em} C = \text{ceil} \((\text{pageset.count}+1)^* t3)\)
15. \hspace{1em} \textbf{Ignore} the nodes where \text{Setcount}<C
16. \hspace{1em} \textbf{Return} \text{temp}
17. \hspace{1em} \textbf{End}

**Figure 4.9:** Wrapper Generation Algorithm

The algorithm takes as input a set of web pages and compares the web pages to find the similarities and differences between them and generating a wrapper (union free regular expression) in this process. Initially from a cluster a webpage is chosen as a wrapper say \(W\), which satisfies the equation 4.5.

\[
\max_{i=0}^{n} \frac{\sum_{j=0}^{n} \left(1 - \text{Ned} (P_i, P_j)\right)}{n} \quad (4.5)
\]

Then the rest of the web pages of the cluster are matched against \(W\), to refine the wrapper. By solving mismatches between \(W\) and a page, it
Chapter 4: Automatic Extraction of Data from Multiple E-commerce Websites

generalizes wrapper. When some string in a web page doesn’t match with the grammar, mismatch occurs. The two types of mismatches are viz. string mismatch (they indicate data items or fields) and tag mismatch (they indicate optional items and iterators) i.e. given a webpage we can find its underlying relational model. The algorithm can be best explained with the help of an example. As shown in figure 4.11, n1 is the initial wrapper and n2 is the page to be matched. Lines 1-3 of both n1 and n2 are same and thus match. Line 4 of both web pages mismatch as they are text strings. They actually are data items to be extracted. Line 6 of n1 matches with the n2’s 7th line. Thus Line 6 of n1 is treated as optional line. Line 9 of n1 and line 10 of n2 again mismatches as they are text strings and are the data items to be extracted. Another mismatch occurs at line 13 of n1 and line 14 of n2. Lines 16-19 of n2 are same as lines 11-14 so square matching is done to extract the data items. The final wrapper (regular expression) of n1 and n2 is given below in figure 4.10.

<HTML><B>Products</B><U> STRING DATA</U>
( <a href=......> </a> ) ?
<UL>
( <LI><I>Title</I> STRING DATA</LI> )+
</UL></HTML>

Figure 4.10: Wrapper after Solving Mismatches
Figure 4.11: A Typical Example of Wrapper Generation
4.3. Experimental Setup and Evaluation

In this section, we experimentally evaluate our approach for extracting structured data from web pages of the same template. The approach has been implemented in Java Programming language and the experiments were conducted on a laptop with 64 bit Windows 7 operating system, i5 CPU @ 3.1 GHZ, 8GB RAM and 7200 RPM hard drive. We use precision, recall and F-score standard measures to evaluate our proposed approach. These evaluation metrics are most popular measures for evaluating Information Retrieval systems. Precision is computed as the number of web pages correctly extracted from the number of web pages extracted, recall is computed as the number of web pages extracted from the number of web pages accessed and both precision and recall determine the F-score value. The equations for calculating precision, recall and F-score is given by the following equations:

\[
\text{Precision} = \frac{\text{number of correctly extracted}}{\text{number of extracted}} \quad (4.6)
\]

\[
\text{Recall} = \frac{\text{number of extracted}}{\text{number of accessed}} \quad (4.7)
\]

\[
F - \text{score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (4.8)
\]

Table 4.1 shows the experimental results of our proposed method. For each website, we start from the base URL and we set our system to 150 as a target web pages to extract. As can be seen from the table below, our proposed method achieves satisfactory precision and recall on web data under experiment.
Table 4.1: Experimental Results of Our Proposed Approach

<table>
<thead>
<tr>
<th>Website</th>
<th>Web pages accessed</th>
<th>Web pages extracted</th>
<th>Web pages correctly extracted</th>
<th>Recall (%)</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snapdeal.com</td>
<td>150</td>
<td>139</td>
<td>136</td>
<td>92.60</td>
<td>97.84</td>
</tr>
<tr>
<td>Flipkart.com</td>
<td>150</td>
<td>110</td>
<td>104</td>
<td>73.33</td>
<td>94.54</td>
</tr>
<tr>
<td>Homeshop18.com</td>
<td>150</td>
<td>135</td>
<td>129</td>
<td>90.00</td>
<td>95.55</td>
</tr>
<tr>
<td>Myntra.com</td>
<td>150</td>
<td>124</td>
<td>106</td>
<td>82.66</td>
<td>85.48</td>
</tr>
<tr>
<td>Bigbuy.com</td>
<td>150</td>
<td>139</td>
<td>127</td>
<td>92.66</td>
<td>91.36</td>
</tr>
<tr>
<td>eBay.com</td>
<td>150</td>
<td>132</td>
<td>125</td>
<td>88.00</td>
<td>94.69</td>
</tr>
<tr>
<td>Currys.co.uk</td>
<td>150</td>
<td>127</td>
<td>123</td>
<td>84.66</td>
<td>96.85</td>
</tr>
<tr>
<td>Argos.co.uk</td>
<td>150</td>
<td>130</td>
<td>115</td>
<td>86.66</td>
<td>88.46</td>
</tr>
<tr>
<td>Craftsvilla.com</td>
<td>150</td>
<td>130</td>
<td>126</td>
<td>86.66</td>
<td>96.92</td>
</tr>
<tr>
<td>Kashmirbox.com</td>
<td>150</td>
<td>125</td>
<td>109</td>
<td>83.33</td>
<td>87.20</td>
</tr>
</tbody>
</table>

4.3.1. The Data Set

This section describes the experimental evaluation of our proposed method with three benchmark datasets, ViNTs [Zhao et al., 2005], Alvarez [Alvarez et al., 2008] and FiVaTech [Kayed and Chang, 2007]. We use these datasets to evaluate structured web data extraction systems. These datasets contain template generated web pages (web pages filled with data from underlying databases). Figure 4.12 shows the precision and recall of these datasets and clearly reveals that our approach achieves perfect precision and nearly perfect recall. Also extraction results of ViNTs-2 and Alvarez Datasets are given in figures 4.13 and 4.14 respectively.
Chapter 4: Automatic Extraction of Data from Multiple E-commerce Websites

**Figure 4.12:** Precision and Recall Results of Our Approach with other Datasets.

**Figure 4.13:** Data records reported with ViNTs-2 and Alvarez Datasets
Chapter 4: Automatic Extraction of Data from Multiple E-commerce Websites

Figure 4.14: Precision, Recall and F-score with ViNTs-2 and Alvarez Datasets

4.4. Conclusion

We have programmed our method in Java language. Our method can be very useful and effective to our online shopkeepers. As the market of e-commerce is changing very fast. So, the shopkeepers are paying more attention on price analysis. In this chapter, we introduce a new method for extracting data records from structured e-commerce web pages. Our method works in five phases: DOM tree creation, similarity measure, clustering, wrapper generation and extraction of data. Initially the web pages are downloaded using a web crawler and stored in a local machine. Then the HTML web pages are converted to DOM trees. By analysing the DOM trees, the web pages of the same type are clustered into groups by using structural similarity of web pages and clustering of the web pages is done using hierarchical agglomerative clustering. We then analyse the
web pages from each cluster to generate the extraction rules and finally the data is extracted and integrated. The proposed method makes full use of the structural similarity of web pages. Experimental results show that our proposed method effectively extracts data records from famous e-commerce websites.