An Overview of Web Data Extraction Techniques

Devika K[1], Subu Surendran[2]

Department of Computer Science and Engineering, SCT College of Engineering, Trivandrum, Kerala.
k_devu@yahoo.co.in, subusurendran@gmail.com

Abstract – **Web pages are usually generated for visualization not for data exchange. Each page may contain several groups of structured data. Web pages are generated by plugging data values to predefined templates. Manual data extraction from semi supervised web pages is a difficult task. This paper focuses on study of various automatic web data extraction techniques. There are mainly two types of techniques one is based on wrapper induction another is automatic extraction. In wrapper induction set of extraction rules are used, which are learnt from multiple pages containing similar data records.**

Keywords -- **data extraction, wrapper induction, DOM tree, web crawler, Data alignment, pattern mining**

I. INTRODUCTION

Internet is a powerful source of information. Most business applications depend on web to collect information that is crucial for decision making process. By analyzing web we can identify market trends, price details, product specification etc. Manual data extraction is time consuming and error prone. In this context automatic web data extraction plays an important role. Example of web data extraction are i) Extract competitor's price list from web page regularly to stay ahead of competition, ii) Extract data from a web page and transfer it to another application iii) Extract people's data from web page and put it in a database.

Websites are usually designed for visualization not for data exchange. All pages of same website will be well designed. They may follow same template. Templates can be used to display objects of same type. Web page construction is the process of combining data to templates. Web data extraction is the reverse process of page generation. If multiple pages are given as input the extraction target will be the page wide information. If one page is given as input extraction target will be record level information.

Automatic extraction is also plays an important role in processing results from search engines. Wrapper is an automated tool that extract search result records(SRRs) from HTML pages returned by search engines. Automated extraction is easier with Google and Amazon because they have web service interfaces. But search engines that support business to customer applications does not have web service interfaces. Search engine result contains query independent (static contents), query dependent (dynamic) contents, contents affected by many queries but independent of content of specific query(semi-dynamic). Typically dynamic sites are filled with data from back-end database and generated by predefined templates. Extracting such data enables one to collect data from multiple sites and provide services like comparative shopping, meta-querying.

The purpose of this paper is overview of various information extraction techniques like FivaTech[1], EXALG[4], RoadRunner[7], ViPER[8], DeLa[2], DEPTA[3], NET[6], IEPAD[5]. Section II considers more details about above techniques, section III provides a comparison and section IV concludes the paper.

II. WEB DATA EXTRACTION TOOLS

A. DeLa (Data Extraction and Label Assignment for Web Databases)

DeLa automatically extract data from web site and assigns meaningful labels to data. This technique concentrates on pages that querying back end database using complex search forms other than using keywords.

DeLa system consists of four components: a form crawler, wrapper generator, data aligner, label assigner. Fig. 1 shows the architecture of DeLa.

Form crawler: It collect labels of the website form elements. Hidden web crawler HiWe[10] is used for this purpose in DeLa. Wrapper generator automatically generates regular expression wrappers from data contained in pages. Most form elements contain text that helps users to understand the characteristics and semantics of the element. So form elements are labeled by the descriptive text. These labels are used further to compare with attributes of data extracted from query-result page.

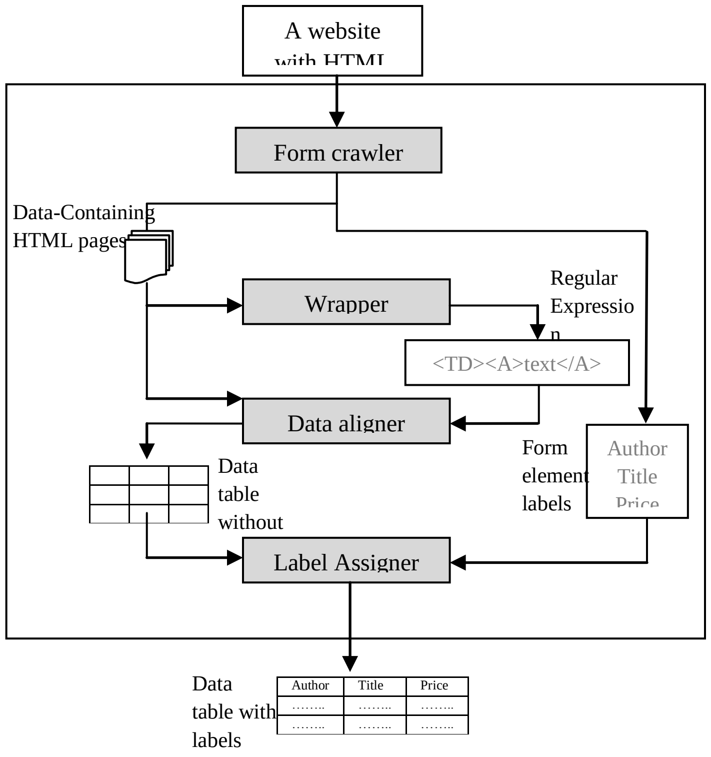

Figure 1 : DeLa architecture

Wrapper Generation: Pages gathered by the form crawler are given as input of the wrapper generator. Wrapper generator produce regular expression wrapper based on HTML tag structures of the page. If a page contains more than one instance of data objects then tags enclosing data objects may appear repeatedly. Wrapper generator considers each page as a sequence of tokens composed of HTML tags. Special token "text" is used to represent text string enclosed with in HTML tag pairs. Wrapper generator then extracts repeated HTML tag substring and introduces a regular expression wrapper according to some hierarchical relationship between them. Techniques used in wrapper generator are

i) Data-rich section extraction

Advertisement, navigational panel are considered as noisy data. These noisy data make data extraction complicated. Noisy data may be wrongly matched with results in inefficient or incorrect wrappers. So it is necessary to identify parts of the page that contain data objects of user interest i.e, data-rich section. Data-rich Section Extraction (DSE) algorithm [11] is used to identify data-rich section. It is performed by comparing two pages of same site. For this traverse DOM trees of the two pages in depth-first order. Each node will be compared and those nodes with identical subtrees at the same depth are discarded.

ii) C-repeated pattern

Structure of data objects appear repeatedly if one page contains more than one data object. These continuous repeated (C-repeated) patterns are discovered as wrapper candidates from token sequences. If a page contains only one data object the data-rich section can be identified by combining multiple pages into single token sequence that will contain multiple data objects.

Definition:- Given an input string S, a C-repeated substring (pattern) of S is a repeated substring of S having at least one pair of its occurrences that are adjacent[2].

Internal structure of a string is exposed by data structure called token suffix-tee [12]. Leaf of suffix tree represented by square with a number. The number indicates the starting position of suffix. Solid circle represents internal node with a number which indicates the token position where its child node differ. Sibling nodes with same parent are arranged in alphabetical order. Label associated with edge between two internal nodes is the sibling between two token positions of the two nodes. Label associated with edge connecting internal node and leaf node is the token at the position of the internal node in the suffix starting from leaf node[2]. Token suffix tree is special suffix tree which can be constructed in O(n) time.

In order to discover C-repeated patterns path-labels of all internal nodes and their prefixes in the token suffix-tree are considered as candidates. A candidate repeated pattern is a C-repeated pattern if any two of its occurrences are adjacent. Repeated patterns are adjacent if the distance between two starting positions is equal to the pattern length. Algorithm that discovers all C-repeated patterns from a suffix tree in O(n log n) time is presented in [13].

For discovering nested structures a hierarchical pattern tree is used. A pattern tree can be used to represent dependence and independence between discovered C-patterns.

iii) Optional attributes and disjunction

Optional attributes appears once or zero times in a page. Wrapper generator will find out repeated patterns. Among repeated patterns it will select highest nested-level as wrapper candidates. There may be multiple patterns with highest nested-level for each page. So number of wrapper candidates may be greater than number of pages in the website. Wrapper candidates may be with some optional missing attributes or some attributes with disjunction values. So there arises a need to construct generalized wrapper from multiple discovered patterns. This can be performed by string alignment. String alignment is performed in O(mn) where n and m are size of two strings S1 and S2. For example consider the pattern P_1= "ABCDXF" and P_2 = "BCEXF" alignment will be performed as follows.

P_1:	A	B	C	D	X	F
P_2:	-	B	C	E	X	F

Generalized wrapper for P_1 and P_2 will be "A?BC(D/E)XF".

Data Alignment : Data aligner has two phases. They are Data extraction and attribute separation.

i) Data exaction

This phase extracts data from web pages according to the wrapper produced by wrapper generator. Then it will load extracted data into a table. In data extraction phase we have regular expression pattern and token sequence that representing web page. A nondeterministic finite automation is constructed to match the occurrence of token sequences representing web pages. A data-tree will be constructed for each regular expression. Data tree is defined recursively as follows[2]

- If the regular expression is atomic, then the data-tree is a single node and occurrence of the expression is the node label.

- If regular expression is $E_1E_2E_3……E_n$ then data-tree is a node with n children and the i^{th} (1< i < n) child is a data-tree that records the occurrence of $E_{i.}$

- If the regular expression is (E_1/E_2) , then the data-tree is a node with one child that records the occurrence of either E_1 or E_2.

- If the regular expression is (E)* and there are m occurrences of E, then the data-tree is a node with m children and the i^{th} (1 < i < m) child is a data-tree that records the m^{th} occurrence of E.

Internal nodes of data-tree can be classified in to two – star type and cat-type. The star type represents (E)* like sub expressions and its children record multiple values of same attributes of data objects. A table is constructed from data-tree by traversing the tree in depth first order and filling table for one node by adding the tables of its child nodes (star-type internal nodes) or multiplying the table of its child nodes (cat-type internal nodes)[2].

ii) Attribute separation

Before attribute separation it is needed to remove all HTML tags. If several attributes are encoded in to one text string then they should be separated by special symbol(s) as separator. For instances "@", "$", "." are not valid separator. When several separators are found to be valid for one column, the attributes strings of this column are separated from beginning to end in the order of occurrence portion of each separator.

Label Assignment: To assign labels to the columns of the table containing extracted data four heuristics are employed [2]:

Heuristic 1: Match from element labels to data attributes.

Heuristic 2: Search for voluntary labels in table header

Heuristic 3: Search for voluntary labels encoded together with data attributes.

Heuristic 4: Label data attributes in conventional formats.

B. DEPTA (Data Extraction based on Partial Tree Alignment).

DEPTA is a two step approach. First step is to identify data record. Second step extracts data items using partial tree alignment method.

i) Data record extraction

The purpose of this step is to segment the page to identify data records. This step is an enhancement of the MDR technique [14].

MDR algorithm based on two observations about data records in a page and an edit distance string matching algorithm [2]. The two observations are

a). A *data record* region describes set of similar objects that typically appear in contiguous region of a page and are formatted using same sequence of HTML tags. By considering HTML tags as string we can use string matching to find similar tags which enclose similar data records. Problem with this observation is a set of data record can start from any tag and end at any tag. A set of data records typically does not have the same length in terms of its tag strings because it may not contain exactly same pieces of information [3]. This problem can be handled in next observation.

b). This observation is based on the tag tree which is formed by HTML tags in the page. Similar data records should be child sub-trees of same parent node. A web page may contain many data region and different data region have different data records.

Steps in data record extraction

a. Building HTML tag tree

Tag tree is constructed as follows.

- Find four boundaries of rectangle of each HTML tag by calling embedded parsing and rendering engine of browser [3].
- Detect containment relationship between rectangles. Construct tag tree based on containment relationship.

b. Mining data region

This step mine the data region by comparing tag strings of individual nodes including descendants and combination of multiple adjacent nodes. Similar nodes are labeled as data region. Generalized node is introduced to denote each similar individual node and node combination. Adjacent generalized nodes form a data region. Gaps between data records are used to eliminate false node combinations. Visual observations about data records states that gap between the data records in a data region should be no smaller than any gap with in a data record [3].

c. Identifying data records

Data records are identified from generalized nodes. Two cases in which data records not contained in contiguous segment are explained below :

Case1 : Data region contains two generalized nodes each of which contains two tag nodes which indicate that they are no similar to each other. Each node has same number of children which are similar to each other.

Case 2: Two or more regions form multiple data records .

ii) Data extraction

It is performed based on partial tree alignment technique to match corresponding data item or fields from all data records. Two sub-steps are

- Production of one rooted tag tree for each data record. Subtrees of all data record are arranged into a single tree.

- Partial tree alignment: tag trees of data records in each data region are aligned using partial alignment. This is based on tree matching. No data item are involved in matching process. Only tag nodes are used for matching.

Tree edit distance [16, 17] between two trees is cost associated with minimum set of operations needed to transform A in to B. In tree edit distance, mapping can cross levels as well as there is also replacement. So restricted matching algorithm called simple tree matching[18] is used which will produce maximum matching between two trees with complexity $O[n_1n2]$ n_1, n_2 are sizes of two trees A and B respectively.

Multiple tag trees of multiple data records are needed to align in order to produce a data base table. In this data table each row represents a data record and column represents data field. This can be performed using multiple alignment method. Partial tree alignment is used in DEPTA. This approach aligns multiple tag trees by progressively growing a seed tag tree. Seed tree is the tree with minimum number of data fields that is picked initially. The selection of seed tree should be in such a way that it should have a good alignment with data fields in other data records. For each tree $T_i[i \neq s]$ the algorithm tries to find a matching node in T_s. When a match is found for node n_i, a link is created from n_i to n_s to indicate its match in the seed tree. If no match found then algorithm attempts to expand the seed tree by inserting n_i in to T_s. The expanded seed tree is used for subsequent match.

C. ViPER (Visual perception based Extraction of Records)

It is a fully automated information extraction tool. This technique is based on assumption that the web page contains at least two consecutive data records which exhibits some kind of structural and visible similarity. ViPER is able to extract relevant data with respect to user’s visual perception of the web page. Multiple Sequence Alignment (MSA) technique is used to align these relevant data regions.

i) Data Extraction

HTML document can be viewed as labeled unordered trees. A labeled unordered tree is a directed acyclic graph T = (V, E, r, η) where V is set of vertices, E is set of edges, R is root and η is the label function η: V x L where L is a string.

Preprocessing is performed to improve pattern extraction accuracy. Preprocessing provides the ability to access parsed document tree T* with additional rendering information. Every tag element is augmented with bounding box information by the upper left corner's (x,y) pixel coordinates along with width and height. For analysis abstract representation of T* is created in which each HTML tag is restricted to tag name ignoring attributes. Text between two tags represented by a new element denoted as <TEXT> element tag. Preprocessed document is called restricted tag tree T and plain tag sequence structure S of T where each element in the tree has a link to the corresponding element in the sequence representation and vice versa [8].

Pattern search : Similarity between two plain sequences S_i, S_j with length m, n respectively is measured using technique edit distance. One disadvantage with edit distance is that repetitive and optional subparts inside the sequence S_i, S_j should increase edit cost, so possible matches may be discarded. These optional subparts are handled by similarity threshold value θ. Two sequences will be similar if their accumulated edit distance is less or equal to threshold value.

Primitive tandem repeats : Tandem repeat contained in a sequence is a subpart of S. Tandem repeat construct an array of consecutive repeats. A repeat is primitive if it does not contain shorter repeats. Each extra repetitive instance will be marked with different marker elements. According to these marked tag elements the recursive computation of a single matrix entry of D is adapted as follows[8].

$$D_{k,l} = \min \begin{cases} D_{k-1,l} + f(a_k, b_l) \\ D_{k,l-1} + f(a_k, b_l) \\ D_{k-1,l-1} + c(a_k, b_l) \end{cases}$$

For $1 \le k \le n$, $1 \le l \le m$

$$f(a_k, b_l) = \begin{cases} 0 & \text{if marker}(a_k) \cap \text{marker}(b_l) \neq \emptyset \\ 1 & \text{else} \end{cases}$$

and

$$c(a_k, b_l) = \begin{cases} 1 & \text{if } a_k \neq b_l \\ 0 & \text{else} \end{cases}$$

Identifying data regions and record: Single data records with in a data region may consist of variable number of subtrees. When computing pair wise similarity between all subtree sequences produce an upper triangular subtree similarity matrix M_u for each inner node u. To simplify pattern discovery edit-distance values are not stored inside the matrix. Cell entry $M_u(i, j)$ becomes 1 if the edit distance between two sequences S_{vi}, S_{vj} satisfies specific conditions. Next to identify sets of adjacent sibling nodes having highest matching frequency.

Visual Data Segmentation: After identifying data region R_k, the corresponding image representation defined by R_k is analyzed. Bounding boxes of <TEXT> elements contained in R_k are used to compute vertical and horizontal projection profiles. This is realized by summing the width and respective height of all boxes with respect to their x/y coordinates[8].

In x-y profile information valleys between peaks corresponds to blank between text lines and distance between two significant valleys corresponds to potential separation of data region into smaller data records. Relationship between valleys and tag elements are established by containment check.

Visual data region weighting: After pattern extraction, the next step is measure the relevance of each pattern. Several heuristics are used to measure individual weight of a region. One heuristics is to compute the textual coverage of data region using pattern size. But this fails if pattern contain images, links or the region that has highest textual coverage. Vision-based page segmentation can be performed by ranking technique, where features of a pattern are determined by visual location on the page. Web pages are divided into different information regions called slots. Slot filled with target information has often a centered location and covers large part of the page.

ii). Data Alignment

sub-optimal solution can be obtained using center-star tree alignment algorithm which finds a sequence (center sequence) minimizing overall edit cost to each remaining sequences. OLERA[19] is a semi supervised information extraction tool where user can generate extraction rules according to training pages. There are several drawbacks in center star technique[8]. This can be overcome by using global sequence alignment which uses general suffix tree.

Global data record alignment try to find maximal matches (MUMs) contained in all records. Maximal means for every sequence we cannot extend a match to left or right and unique means match occurs only once in each n sequences. MUM sequences of non overlapping MUMs with maximal weight are selected. Weight of MUM sequence is the sum of weights of its constituting elements. If we are trying to align data records using MUM sequence of length l, the problem decomposes into l+1 smaller unaligned subregions. Each subregion areteratively aligned separately using global matching[20].

Next is text alignment. This contains building general suffix tree and checking regularities contained in all sequences. Two abstract <TEXT> elements A and B are content similar if $f_{sum}(\sigma^{-1}(A), \sigma^{-1}(B)) > \theta_{string}$. That is if they are similar w.r.t their original trimmed text content. f_{sum} is string comparator function and θ_{string} is threshold value. Best alignment results can be achieved with $\theta_{string} = 0.7$.

D. ROAD RUNNER

Road runner defines data extraction problem as "given a set of sample HTML pages belonging to the same class, find the nested type of the source data set and extract the source data set from which the pages have been generated"[7].

The site generation process is encoding of database content into strings of HTML tags. So data extraction is a process of decoding. Lattice-theoretic approach can be used for data extraction. This is based on correspondence between nested types and union-free regular expressions. Given a special symbol #PCDATA and an alphabet of symbols $\sum$ not containing #PCDATA, a union free regular expression (UFRE) over $\sum$ is a string over alphabet $\sum \cup \{$ #PCDATA, ., +, ?, (,) $\}$. Empty string $\in$ and all other elements of $\sum \cup \{$ #PCDATA $\}$ are UFRE[7].

If $s_1, s_2, \ldots, s_k$ are html tags and $i_1, i_2, \ldots, i_k$ instances of data objects of nested type τ, we can discover type τ by inferring minimal union-free regular expression σ whose language $L(\sigma)$ contains $s_1, s_2, \ldots, s_k$ and reconstruct original data set $i_1, i_2, \ldots, i_k$. Data extraction needs to find minimal UFRE. If we consider UFRE whose language contains input HTML strings such that $\sigma_1 \leq \sigma_2$ iff $L(\sigma_1) \subseteq L(\sigma_2)$. The minimal UFRE is least upper bound of input strings. i.e, $\sigma = LUB(s_1, s_2, \ldots, s_k)$. By iteratively computing LUB on regular expression lattice we can identify a common wrapper for input pages.

Matching Technique : Matching technique is based on ACME (Align, collapse under mismatch and Extract) technique. There are two assumptions i) tags are properly closed ii) source pages are transformed into tokens by a lexical analyzer. Matching algorithm works on list of tokens and wrappers. Wrapper for each page will be created initially. Then they are progressively refined to find common regular expression among them. This can be performed by solving mismatches between wrappers and tokens. Mismatch occurs when token does not comply with grammar specified by the wrapper at the time of parsing.

Usually there are two types of mismatches. String mismatches and tag mismatches. String mismatches happen when different strings occur at same position of wrapper. This is solved by generalizing wrapper by replacing newly discovered fields by same symbol. String mismatches are used to discovering fields.

Tag mismatches leads to discovery of iterative or optional patterns. In the case of optional patterns, we should able to resume the parsing by skipping the piece of code. It is performed in two steps. Optional pattern location by cross-search and wrapper generation. Tag mismatches due to iteration can be solved by identifying repeated patterns called squares. Wrapper is generalized accordingly then parsing can be resumed. This is performed in three steps.

i) Square location by Terminal-Tag search

One hint about the square is that both wrapper and sample contain at least one occurrence of square. If O_w and O_s are no of occurrences of square in wrapper and token respectively, then we can assume that $\min(O_w, O_s)$ occurrences have been matched. Last token of square i.e, the token immediately before mismatch, is called terminal tag. There are two possibilities. Candidate square may be in wrapper or in sample. This can be checked by searching both sample and wrapper.

ii) Square Matching

This verifies whether candidate tokens really form a square. This is done by searching the candidate square backwards. If search succeeds then we can conclude it is a square.

iii) Wrapper generation

If S is an optional tag then it can be represented as (S)? . If S is a square then it can be represented as (S)+.

E. EXALG

EXALG performs template extraction in two stages. First stage is Equivalence Class Generation Stage (ECGM) and second is analysis stage as shown in Fig. 2. ECGM stage computes equivalence classes. i.e, set of tokens having same frequency of occurrence in every page. This is performed by FindEquiv sub module. There may be many equivalence classes. But EXALG only considers equivalence classes that are large and contain tokens which occur in large number of pages. Such equivalence classes are called Large and Frequently Occurring Equivalence Classes(LFEQs). It is very unlikely for LFEQs to be formed by chance. LFEQs are formed by tokens associated with the same type constructor in the template[4].

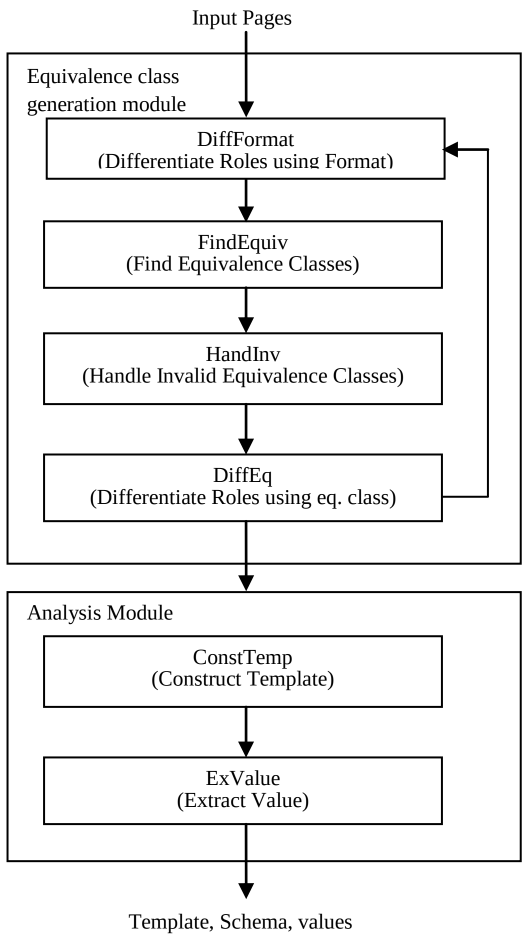
Input Pages
Equivalence class generation module
DiffFormat
(Differentiate Roles using Format)
FindEquiv
(Find Equivalence Classes)
HandInv
(Handle Invalid Equivalence Classes)
DiffEq
(Differentiate Roles using eq. class)
Analysis Module
ConstTemp
(Construct Template)
ExValue
(Extract Value)
Template, Schema, values

Figure 2 : Modules of EXALG

Sub module HandInv detects and removes invalid LFEQs. Occurrence vector of a token is a vector <f_1, f_2, …., f_n> where f_i is the number of occurrences of f in pages p_1, p_2, …., p_n. Equivalence class is a maximal set of tokens having the same occurrence vector[4]. Invalid equivalence classes are handled using violation of ordered and nesting properties.

DiffFormat sub module differentiates roles of non tag tokens using context in which they occur. Occurrence path of a page-token is path from root to page-token in the parse tree representing the page. Page-tokens with different occurrence path have different roles. Occurrences of the token in different contexts have different roles.

LFEQ can be expanded by adding differentiated tokens or dtokens which are frequently occur in almost all pages and not already added to LFEQs. DiffEq sub module differentiates role of tag tokens. FindEquiv, HandInv, DiffEq sub modules will iterate in a loop.

Analysis stage constructs a template using LFEQs. For this EXALG first considers root LFEQs whose tokens occur exactly once in every input page. Position between two consecutive tokens is empty if they occur contiguously otherwise it is non empty. EXALG will determine tokens of LFEQs which are non empty. It constructs output template T' by generating a mapping from each type constructor in S' to ordered set of strings.

F. NET (Nested data extraction using Tree matching and Visual cues)

NET extract data items from data records even it handles nested data records also. There are two main steps.

Building tag tree is difficult because page may contain erroneous and unbalanced tags. This is performed based on nested rectangles. For this four boundaries of each HTML element are determined by calling embedded parsing and rendering engine of a browser. A tree is constructed based on containment check, whether one rectangle contained inside another.

NET traverses the tag tree in post order (bottom up) in order to find nested data records which are found at lower level. These tasks are performed using two algorithms traverse and output[6]. Traverse() find nested records by recursive call if the depth of sub-tree from the current node greater than or equal to three. Simple tree matching algorithm is used here. Data items are aligned after tree matching. All aligned data items are then linked in such a way that a data item will point to its next matching data item.

Output() algorithm will put linked data into a relational table. Data structure used is linked list of one dimensional array which represents columns. This data structure makes it easier to insert columns in appropriate location for optional data items. All linked data items are put in same column. A new row is started if an item is being pointed to by another item in an earlier column[6].

G. FivaTech

FivaTech is a page-level web data extraction technique. Data extraction is performed in two modules. First module takes DOM trees of web pages as input and merges all DOM trees into a structure called fixed/variant pattern tree. In the second module template and schema are detected from fixed/variant pattern tree. Fig. 3 shows FivaTech approach.

First module arranges all nodes of input DOM trees into a matrix form. This module can be divided into four sub modules. They are Peer node recognition, Multiple string alignment, Pattern mining, Optional node merging. Nodes which have same tag name but different functions are called peer nodes. Peer nodes are denoted using same symbol in order to facilitate string alignment. Pattern mining on aligned string will remove extra occurrences of discovered pattern.

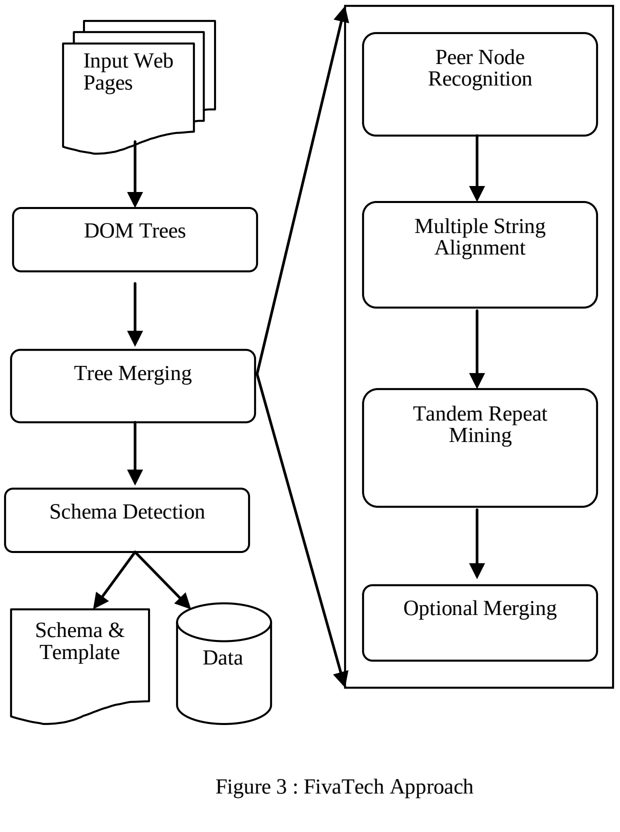

Figure 3 : FivaTech Approach

Peer node recognition: Peer nodes are identified and they are assigned same symbol. Simple Tree Matching [STM] algorithm together with score normalization [1] is used for identifying peer nodes.

Matrix alignment: This step aligns peer matrix to produce a list of aligned nodes. Matrix alignment recognizes leaf nodes which represent data item.

Optional node merging: This step recognizes optional nodes, the nodes which are which disappears in some column of the matrix. This step groups nodes according to their occurrence vector.

Schema detection module detects structure of the website i.e, identifying the schema and defining the template. The items contained in a page can be divided into basic type, set type, optional type and tuple type[1]. This step recognizes tuple type as well as order of set type and optional data which are already identified by previous module.

H. IEPAD

IEPAD is an information extraction system which applying pattern discovery techniques. It has three components, an extraction rule generator, pattern viewer and an extractor module. Extraction rule generator accepts input web page and generate extraction rules. Extraction rule generator includes a token translator, PAT tree constructor, pattern discoverer, a pattern validator and an extraction rule composer as shown in Fig. 4. Pattern viewer is a graphical user interface which shows the repetitive pattern discovered. Extractor module extracts desired information from pages. Extraction rules generated by extraction rule generator can be used by the extractor module to extract information from other pages which are following similar structure.

Translator generates tokens from input webpage. Each token is represented by a binary code of fixed length l. PAT tree constructor receives the binary file to construct a PAT tree. PAT tree is a PATRICIA tree (Practical Algorithm to Retrieve Information Coded in Alphanumeric [21]). PAT tree is used for pattern discovery. Discoverer uses PAT tree to discover repetitive patterns called maximal repeats. Validator filters out undesired patterns from maximal repeats and generates candidate patterns. Rule composer revises each candidate pattern and to form an extraction rule in regular expression [5].

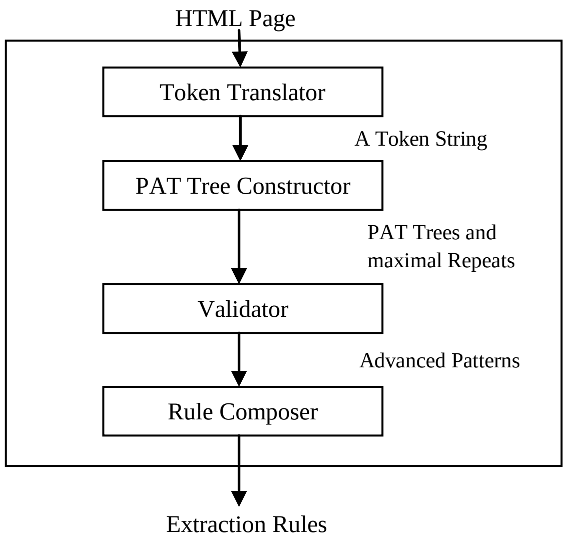

Figure 4 : Extraction Rule Generator

III. COMPARISON

Among the webpage extraction techniques discussed above, some techniques reveals flat records and some other techniques are trying to extracts nested records also. DEPTA and NET will find out nested records in addition to flat records. All other techniques produce only flat records. DeLa, RoadRunner and IEPAD extracts records using wrapper induction method, others are based on operations on tree structure of the page such as tree alignment, tree merging and tree matching. In DEPTA extraction is performed mainly by partial tree alignment. FivaTech uses tree merging technique whereas NET using tree matching. Other extraction methods are based on visual perception, equivalence class generation which is used by ViPER and EXALG respectively. RoadRunner, EXALG and FivaTech considers multiple pages of website and other techniques considers only single page. Summary of comparison is shown in Table I.

IV. CONCLUSION

This paper studied various approaches to extract structured data from web pages. Some of these techniques are either inaccurate or make many strong assumptions. These techniques reconstructs hidden back-end database. Some techniques use regular expression wrappers to extract data objects.

Techniques	Type of record considers	Extraction method	Single page/ Multiple pages
DeLa	Flat	Wrapper induction	Single
DEPTA	Flat	Partial tree alignment	Single
ViPER	Flat	Visual Perception based	Single
RoadRunner	Flat and nested	Wrapper induction	Multiple
EXALG	Flat	Equivalence class generation	Multiple
FivaTech	Flat	Tree merging and schema detection	multiple
NET	Flat and nested	Tree matching	single
IEPAD	Flat	Wrapper induction	single

Table I : Comparison of web data extraction techniques

V. REFERENCES

[1] Mohammed Kayed and Chia-Hui Chang. “FiVaTech: Page-Level Web Data Extraction from Template Pages,” IEEE Transactions on Knowledge and Data Engineering, VOL. 22, NO. 2, 2010.

[2] J. Wang and F.H. Lochovsky, “Data Extraction and Label Assignment for Web Databases,” Proc. international conference on World Wide Web (WWW-12), pp. 187-196, 2003.

[3] Y. Zhai and B. Liu, “Web Data Extraction Based on Partial Tree Alignment,” Proc. international conference on World Wide Web (WWW-14), pp. 76-85, 2005.

[4] A. Arasu and H. Garcia-Molina, “Extracting Structured Data from Web Pages,” Proc. ACM SIGMOD, pp. 337-348, 2003.

[5] C. H. Chang and S. C. Lui, “IEPAD: Information Extraction Based on Pattern Discovery,” Proc. International Conference on World Wide Web (WWW-10), pp. 223-231, 2001.

[6] Bing Liu and Yanhong Zhai, “NET - A System for Extracting Web Data from Flat and Nested Data Records,” Proc. WISE'05 Proceedings of the 6th international conference on Web Information Systems Engineering, pp. 487-495, 2005.

[7] V. Crescenzi, G. Mecca, and P. Merialdo. ROADRUNNER: Towards automatic data extraction from large web sites. In Proc. of the 2001 international conference on Very Large Data Bases, pp 109–118, 2001.

[8] K. Simon and G. Lausen, "ViPER: Augmenting Automatic Information Extraction with Visual Perceptions," Proc. International Conference on Information and Knowledge Management (CIKM), 2005.

[9] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu, "Automatic Extraction of Dynamic Record Sections from Search Engine Result Pages," Proc. international conference on Very Large Databases (VLDB), pp. 989-1000, 2006.

[10] S. Raghavan and H. Garcia-Molina. "Crawling the hidden web," Proc. 27th VLDB Conf., 2001, 129-138.

[11] J. Wang and F. Lochovsky. "Data-rich section extraction from HTML pages," Proc. 3rd Conf. on Web Information Systems Engineering, 2002, 313-322.

[12] D. Gusfield. Algorithms on Strings, Trees, and Sequences. Cambridge, 1997.

[13] J. Wang and F. Lochovsky. *"Wrapper Induction based on nested pattern discovery."* , Technical Report HKUSTCS-27-02, Dept. of Computer Science, Hong Kong U. of Science and Technology, 2002

[14] Liu, B., Grossman, R. and Zhai, Y. "Mining data records from Web pages." *KDD-03*, 2003.

[15] Baeza-Yates, R. Algorithms for string matching: A survey. *ACM SIGIR Forum*, 23(3-4):34-58, 1989.

[16] Tai, K. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, 1979

[17] Valiente, G. Tree edit distance and common subtrees. Research Report LSI-02-20-R, Universitat Politecnica de Catalunya, Barcelona, Spain, 2002.

[18] Yang, W. Identifying syntactic differences between two programs. *Softw. Pract. Exper.*, 21(7):739–755, 1991.

[19] C.-H. Chang and S.-C. Kuo. OLERA: A semi-supervised approach for web data extraction with visual support. IEEE Intelligent Systems (SCI, EI), 19(6):56–64, 2004.

[20] D. Gusfield. Algorithms on Strings, Trees, and Sequences. CUP, Cambridge, UK, 1997.

[21] Gonnet, G.H.; Baeza-yates, R.A.; and Snider, T. 1992. New Indices for Text: Pat trees and Pat Arrays. Information Retrieval: Data Structures and Algorithms, Prentice Hall.