F(Vi)DE: A Fusion Approach for Deep Web Data Extraction

Saranya V

Assistant Professor
Department of Computer Science and Engineering
Sri Vidya College of Engineering and Technology, Virudhunagar, Tamilnadu, India

***Abstract* -** Web mining is the application of data mining techniques to discover patterns from the Web. Many web applications such as information retrieval, information extraction and automatic page adaptation can benefit from web structure mining. This paper presents an automatic top-down, tag-tree independent approach to detect web content structure. Comparing to other existing techniques, our approach is independent to underlying documentation representation such as HTML and works well even when the HTML structure is far different from layout structure. Web2DB is a web data extraction service. It takes unstructured data from web html pages and converting it into structured records. They analyze several types of visual features which exist in all response pages, including position features, layout features, appearance features and content features. Based on these visual features, in this paper, we propose a system that can extract informative or useful content from Web pages across different sites. We implemented F(Vi)DE which can performs the extraction using only the visual information of the response pages when they are rendered on web browsers. Our approach is independent to underlying documentation representation such as HTML.

Keywords: **Data mining, web mining, regrouping algorithm, page segmentation, web structure mining.**

I INTRODUCTION

The extraction of lists from the Web is useful in a variety of Web mining tasks, such as annotating relationships on the Web [20], discovering parallel hyperlinks, enhancing named entity recognition, disambiguation, and reconciliation. The many potential applications have also attracted large companies, such as Google, which has made publicly available the service Google Sets to generate lists from a small number of examples by using the Web as a big pool of data.

Finding information [5] about people in the World Wide Web is one of the most common activities of Internet users. Person names, however, are highly ambiguous. In most cases, the results for a person name search are a mix of pages about different people sharing the same name. The user is then forced either to add terms to the query (probably losing recall and focusing on one single aspect of the person), or to browse every document [1] in order to filter the information about the person he/she is actually looking for. In an ideal system the user would simply type a person name, and receive search results clustered according to the different people sharing that name. One particular case of this people document association task is referred to as personal name resolution. The task is as follows: given a set of documents all of which refer to a particular person name but not necessarily a single individual (usually called referent), identify which documents are associated with each referent by that name. Different methods have been used to represent documents that mention a candidate, including snippets, text around the person name, entire documents, extracted phrases, etc

Data records [9] are usually displayed visually neatly on Web browsers to ease the consumption of users. When the user submits a query in their search interface, a page is created dynamically which has a list of mobiles that matches the query. This dynamically created page is an example of deep [20] web page. Each mobile detail is displayed in the form of structured data records; each data record contains data items like price, discount, features, color, etc. Data records are structured not only for the ease of humans but also for many applications like deep web crawling were data items need to be extracted from the deep web page. Recently the deep web [20] crawling has gained a lot of attention and many methods have already been proposed for data record extraction from deep web pages. But these proposed methods are structure-based; either based on analyzing

HTML codes or the tag types of the web pages. the inherent limitation of They are dependent on the programming language of the web page. Most of these methods are meant for HTML. Even if we assume that only HTML is used to write all the web pages, the previously proposed methods are not fully efficient and fool proof. The evolution of HTML is non-stop and hence the addition of any new tag will require amendment in the previous works in order to adapt to the new version. An example of such extraction is shown in Figure 1.

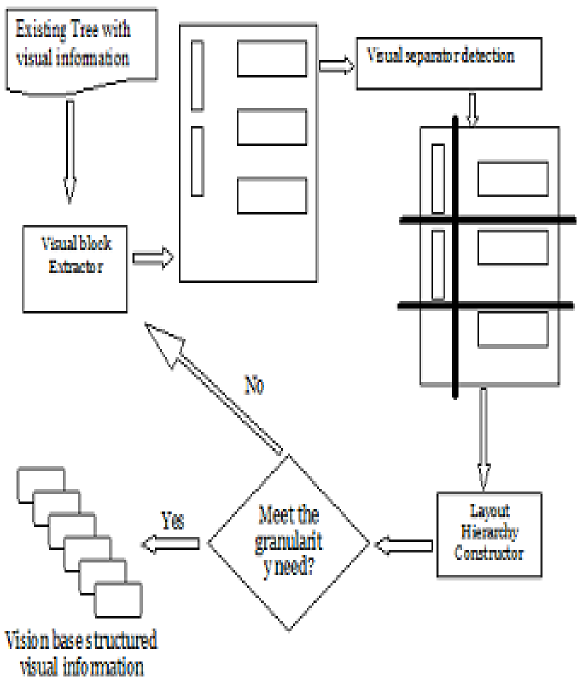

Existing Tree with visual information
Visual block Extractor
Visual separator detection
Layout Hierarchy Constructor
Meet the granularity need?
No
Yes
Vision base structured visual information

Figure 1. Architecture of Visual Extraction.

A good aspect of extracting information from web pages [5] is that even if the page is badly formatted and contains sloppy code, it almost always has some sort of underlying structure. The HTML code itself promotes it with the use of <table> or <div>-tags. The rows and columns might be packed with non-consistent information and aesthetic layout tags, but they're at least confined in a table. This is at least some comforting fact when tackling this problem.

II RELATED WORKS

A Robust Approach of Automatic Web Data Record [9] Extraction [Yongquan DONG, Qingzhong LI]. Structured data objects [2] are a very important type of information [5] on the Web. Such data objects are often records from underlying databases and displayed in Web pages with some fixed templates. We also call them data records. Mining data records [17] in Web pages is useful because they typically present their host pages' essential information, such as lists of products and services.

Extracting these structured data objects enables one to integrate data/information [5] from multiple Web pages to provide value-added services, e.g., comparative shopping, meta-querying and search. Due to the huge amount of web pages, it is unreasonable to extract data records by hand or in semi-automatic ways. The project proposes a robust automatic web data record extraction. Firstly, it utilizes the visual [4] information to identify data region containing data records. Secondly, it uses clustering algorithm to divide the similar nodes into one group. Thirdly, it filters the noisy nodes. Finally, it uses some heuristic rules to generate data records.

Deep Web Content Mining [Shohreh Ajoudanian, and Mohammad Davarpanah Jazi]. To extract information from deep web [20] that is a large collection [8] of dynamic query databases, we need a system that can extract automatically. For this purpose we use web content mining techniques [17] that uses XML version of HTML query interfaces. Web content mining is a form of text mining and can take advantage of the semi-structured [6] nature of web page text. Query interfaces share similar or common query patterns. For instance, a frequently used pattern is a text followed by a selection list with numeric values.

Automatic Retargeting of Web Page Content [Ranjitha Kumar, Juho Kim, Stanford University HCI Group]. Automatically adapting content to different design layouts poses two significant technical challenges.

First, given a pair of web pages, both must be segmented into their constituent perceptual blocks [3]. Second, a mapping between the blocks of the two pages must be computed.

Current template-based systems that support automatic layout changes require pages to be composed of labelled content blocks [3]. Different template layouts are simply different arrangements of the same content blocks, making retargeting straightforward.

III METHODOLOGY

A. Wrapper and Extraction

Wrapper [22] in data mining is a program that extracts content of a particular information [5] source and translates it into a relational form which is shown in figure 2. Many web pages present structured data - telephone directories, product catalogs, etc. formatted for human browsing using HTML language. Structured data are typically descriptions of objects [2] retrieved from underlying databases and displayed in Web pages following some fixed templates. Software systems using such resources must translate HTML content into a relational form. Wrappers are commonly used as such translators.

Formally, a wrapper [15] is a function from a page to the set of tuple it contains. There are two main approaches to wrapper generation: wrapper induction and automated data extraction. [15] [22].

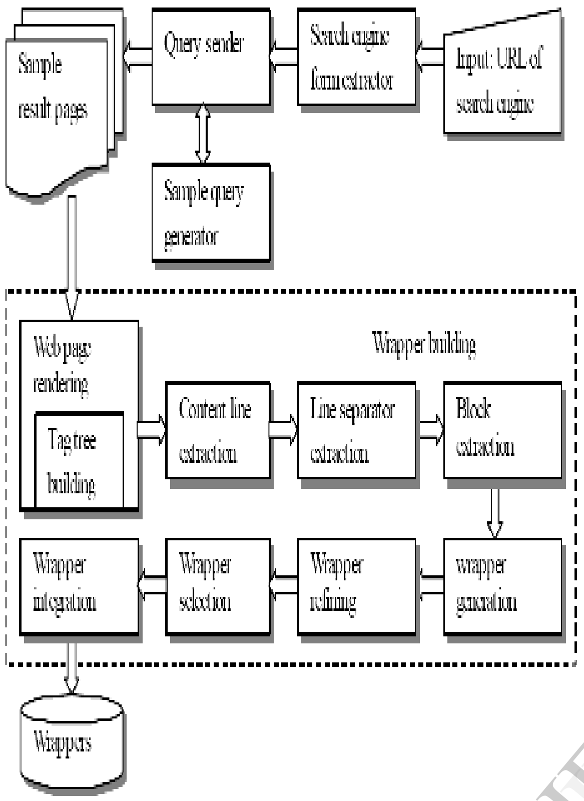


Figure 2. Architecture of Wrapping

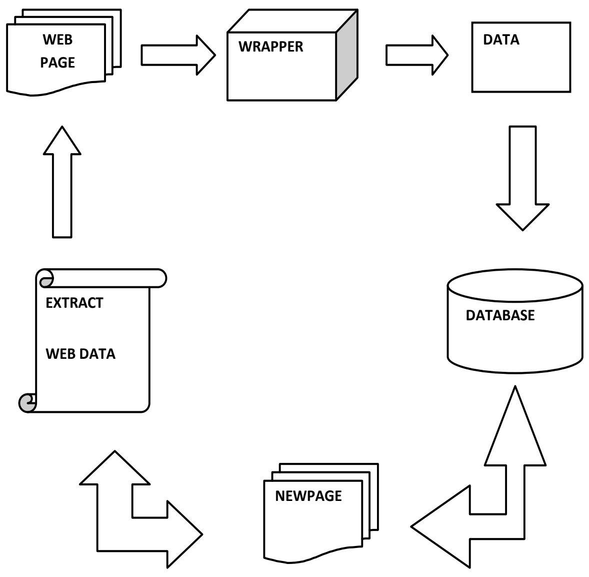


Figure 3. Architecture of Visual Data Extraction.

IV. FUSION METHOD

Detecting the schema of a web site has been a key step for value-added services like comparative shopping and information integration systems [21]. Our proposed work is aimed to filter schema of website & extract data.

A. Hybrid List Extractor

```
Input: Web site S, level of similarity α, max DOM-nodes β
Output: set of lists L
RenderedBoxTree T(S);
Queue Q;
Q.add(T.getRootBox());
while !Q.isEmpty() do
Box b = Q.top();
list candidates = b.getChildren();
list aligned = getVisAligned(candidates);
Q.addAll(getNotAligned (candidates));
Q.addAll(getStructNotAligned (candidates,α, β));
aligned = getStructAligned (aligned, α, β);
L.add(aligned);
return L;
```

B. Block Regrouping Algorithm

Input: $C_1, C_2, ., C_m$; a group of clusters generated by blocks [3] clustering from a given sample deep web [20] page P.

Output: $G_1, G_2, ..., G_m$; each of them corresponds to a data record on P

```
Begin
//Step 1: sort the blocks in $C_i$ according to their positions in the page (from top to bottom and then from left to right)
 1 for each cluster $C_i$ do
 2 for any two blocks $b_{i,j}$ and $b_{i,k}$ in $C_i$ //$1 \le j < k \le |C_i|$
 3 if $b_{i,j}$ and $b_{i,k}$ are in different lines on P,and $b_{i,k}$ is above $b_{i,j}$
 4 $b_{i,j} \leftarrow > b_{i,k}$;      //exchange their orders in $C_i$;
 5 else if $b_{i,j}$ and $b_{i,k}$ are in the same line on P, and
```

```
$b_{i,k}$ is infront of $b_{i,j}$
6 $b_{i,j}$ <→$b_{i,k}$;
7  end until no exchange occurs;
8    from  the  minimum-bounding  rectangle
$Rec_i$ for $C_i$;
//Step 2: initialize n groups, and n is the number
of data records on P
  9   $C_{max}=\{C_i| |C_i|=max\{|C_1|,|C_2|,\ldots,|C_m|\}\}$;  //
  $n=|C_{max}|$
 10   for each block $b_{max,j}$ in $C_{max}$
 11          Initialize group $G_i$;
 12          Put $b_{max,I}$ into $G_i$;
 //Step 3: put the blocks into the right groups, and
 each group corresponds to a data record
 13 For each cluster $C_i$
 14  if $Rec_i$ overlaps with $Rec_{max}$ on P
 15  if $Rec_i$ is ahead of (behind) $Rec_{max}$
 16  for each blocks $b_{i,j}$ in $C_i$
 17 find the nearest block $b_{max,k}$ in $C_{max}$ that is behind
      (ahead of) $b_{i,j}$ on the web page;  place $b_{i,j}$ into
                          group $G_i$;
         End
```

C. Noisy Chunk Removal

A page from a news site on the Web might contain an article about an international political event and "noise information" like a diet advertisement, a legal notices section, and a navigation bar.

A search engine attempting to index the full content of the page might choose keywords based upon the noise information [5] instead of from text related to the primary topic of the page – the political event.

To crawl the web [20], a search engine service may use a list of root web pages to identify all web pages that are accessible through those root web pages.

The keywords of any particular web page can be identified using various well-known information retrieval techniques, such as identifying the words of a headline, the words supplied in the metadata of the web page, the words that are highlighted, and so on. As a result, a search engine service may select keywords based on noise information, rather than the primary topic of the web page.

A chunk of a web page represents an area of the web page that appears to relate to a similar topic. The importance system provides the characteristics or features of chunk to an importance function that generates an indication of the importance of that chunk to its web page. This paper asks users to provide an indication of the importance of blocks [3] of web pages in a collection of web pages which is shown in fig(5).

D. Similarity between Visual Blocks

Our approach decides if two visual blocks have visual, width or content similarity. Two visual blocks [20], A and B, are visually similar if all of the visual properties of both blocks [3] are the same. Let PA = fPa1; Pa2; :::; Pang be a set of visual properties of A, and PB = fPb1; Pb2; :::; Pbng be a set of visual properties of B. The visual similarity between A and B, Sim(A;B), is defined as follows:

$$Sim(A,B) = \begin{cases} 1 \text{ if Pai } = \text{ Pbi}; \ i = 1;\ 2; \because : ;\ n; \\ 0\ Otherwise \end{cases} \quad (1)$$

For example, as shown in Figure 1, the visual blocks [3] that contain the product name in each record on the query result page have the same visual proprieties and are, therefore, visually similar.

E. Measuring Block Content Similarity

Our approach uses a similarity measure to determine if two container blocks [3] have similar block [3] contents. Assume that block A contains a set of child blocks fa1; a2; :::; amg and block B contains a set of child blocks fb1; b2; :::; bng. It is reasonable to expect that a container block may contain more than one child block with the same visual [4] properties. For example, a data record on a car sales web site may contain an individual child block for each feature of the car, such as the engine size, number of doors and fuel type. These child blocks [3] could share the same visual properties. Accordingly, our approach uses a multiset representation for each container block. This generalization of the notion of a set, in which members may appear more than once, allows our approach to represent the child blocks of a container block.

Our approach uses a one-pass algorithm to cluster each of the child blocks contained in a set into a strict partition based on their visual similarity.

The function Sim, defined in Definition 1, is used for this purpose. Two child blocks [3], a and a0, are clustered together if Sim (a; a0) = 1 by equation(1). So a set of child blocks, A, is clustered into a strict partition Ac = fA1;A2; :::;Amg.

$$A_c = \{\{a_1,a_2\},\{ a_3\}, \{a_4\}\} \quad (2)$$

Select one child block from each cluster in Ac as its representative, so we have a set of representative child blocks, Ax, for Ac : by equation (2) & (3).

$$A_x = \{\{x_1,x_2,\ldots\ldots x_n\}\} \quad (3)$$

Given two sets of visual blocks, A and B, we use our similarity measure to find the block content similarity between A and B, defined as follows:

$$SimBlockContent = \frac{|A \cap B|}{|A \cup B|} \tag{4}$$

F. Web Page Layout

When an HTML document [1] is rendered in a Web browser, the CSS2 visual formatting model [8] represents the elements of the document by rectangular boxes that are laid out one after the other or nested inside each other which is shown in fig(4). By associating the document with a coordinate system whose origin is at the top-left corner, the spatial position of each text/image element on the Web page is fully determined by both the coordinates (*x*, *y*) of the top-left corner of its corresponding box, and the box's height and width. The spatial positions of all text/image elements in a Web page define the *Web page layout*. Each Web page layout has a tree structure, called *rendered box tree*, which reflects the hierarchical organization of HTML tags in the Web page.

More precisely, let *H* be the set of occurrences of HTML tags in a Web page *p*, *B* the set of the rendered boxes in *p*, and *map* : $H \rightarrow B$ a bijective function which associates each $h \in H$ to a box $b \in B$. The markup text in *p* is expressed by a rooted ordered tree, where the root is the unique node which denotes the whole document [1], the other internal nodes are labeled by tags, and the leaves are labeled by the contents of the document or the attributes of the tags. The bijective map defines an isomorphic tree structure on *B*, so that each box $b \in B$ can be associated with a parent box $u \in B$ and a set $CB = \{b1, b2, \cdot\cdot\cdot , bn\}$ of child boxes. Henceforth, a box *b* will be formally defined as a 4-tuple _*n*, *u*,*CB*, *P*_, where $n = map{-}1(b)$ and $P = (x, y)$ is the spatial position of *b*, while a rendered box tree *T* will be formally defined as a directed acyclic graph $T = \{B, V, r\}$, where $V \subset B \times B$ is the set of directed edges between the *boxes*, and $r \in B$ is the root box representing the whole page. An example of a rendered box tree is shown in Figure 2. The leaves of a rendered box tree are the non-breakable boxes that do not include other boxes, and they represent the minimum units of the Web page. Two properties can be reported for the rendered box trees.

Property 1. If box *a* is contained in box *b*, then *b* is an ancestor of *a* in the rendered box tree.

Property 2. If *a* and *b* are not related under property 1, then they do not overlap visually on the page.

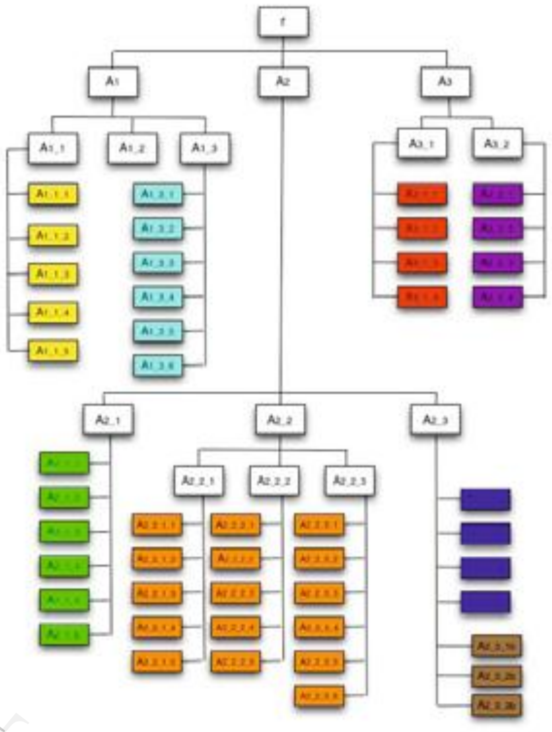


Figure 4. Web Page Layout (Blocks)

G. Chunk Segmentation

We see a chunk within a web page that signifies the principal topic of that page may aid a search engine choose which words are the most important ones on the page when it aims to associate the page with keywords that someone might search with to find that page.

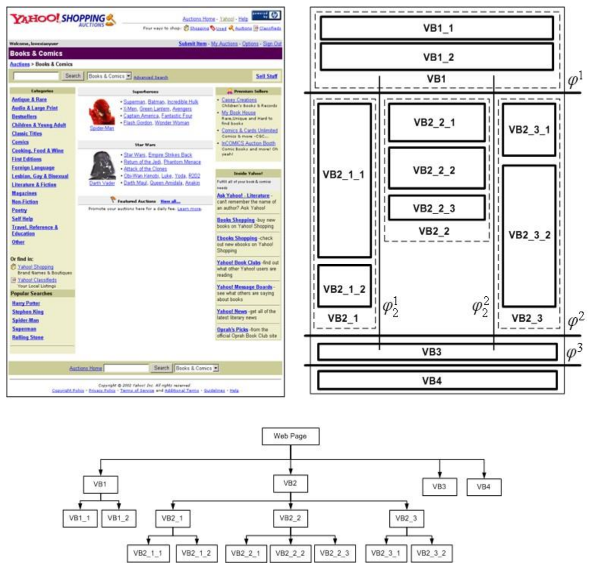


Figure 5. Web Page Chunk Segmentation

Determining the most important chunk on a page could influence the weight and importance of links from different chunks on a page, so that a link from the most important chunk on a page has more value than a link from the least important chunk. Removing these noises will help in improving the mining of web [17]. To assign importance to a region in a web page ( ), we first need to segment a web page into a set of chunks. Hence, to clean a web page, a pre-processing step called Chunk Splitting Operation (fig.5) is performed [10].

H. The VIPS Algorithm

In the VIPS algorithm, the vision-based content structure of a page is deduced by combining the DOM structure and the visual cues. The segmentation process is illustrated in Fig. 2. First, DOM structure and visual information [5], such as position, background color, font size, font weight, etc., are obtained from a web browser. Then, from the root node, the visual block extraction process is started to extract visual blocks of the current level from the DOM tree based on visual cues. Every DOM node is checked to judge whether it forms a single block or not. If not, its children will be processed in the same way.

When all blocks of the current level are extracted, they are put into a pool. *Visual separators* among these blocks are identified and the weight of a separator is set based on properties of its neighboring blocks. After constructing the layout hierarchy of the current level, each newly produced visual blocks is checked to see whether or not it meets the granularity requirement. If no, this block will be further partitioned. After all blocks are processed, the final vision-based content structure for the web page is outputted. Below we introduce the visual block extraction, separator detection and content structure construction phases respectively.

V EXPERIMENTAL SETUP

We present the experimental evaluation of data extraction phenomenon over the web sites [17]. We have addressed the concrete areas of page segmentation, chunk segmentation problems, Block similarity.

These areas tend to have significant interest among the users and it was shown that our methodologies are outperforming, predicting [1] and presenting best results to the user‘s point of view.

Also our method solutions are evaluated by human and system judgement called Kappa Score [20] which is efficient in providing the correct score toward the relevancy of the web pages.

A. Evaluation Metrics

Precision: The fraction of the predicted *web page* information needs that were indeed rated satisfactory by the user .And can also be defined as the fraction of the retrieved pages which is relevant. Precision at K for a given query is the mean fraction of relevant pages ranked in the top K results; the higher the precision, the better the performance is. The score is Zero if none of the top K results contain a relevant page. The problem of this metric is, the position of relevant pages within the top *K* is irrelevant, while it measures overall use potential satisfaction with the top *K* results.

Recall*:* This is used to separate high-quality content from the rest of the contents and evaluates the quality of the overall pages. If more blocks are retrieved, recall increases while precision usually decreases. Then, a proper evaluation has to produce precision/recall values at given points in the rank. This provides an incremental view of the retrieval performance measures. The received page is analyzed from the top pages and the precision-recall values are computed when we find each relevant page

.

F measure: The weighted harmonic mean of precision and recall, the traditional F-measure or balanced F-score is:

$$F\ Measure = \frac{2.Precision\ *Recall}{(Precision\ +Recall\ )} \tag{5}$$

This is also known as the *F*1 measure, because recall and precision are evenly weighted using equation (5).

Two other commonly used F measures are the F2 measure, which weights recall twice as much as precision, and the F0.5 measure, which weights precision twice as much as recall. Fβ "measures the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision". It is based on van Rijsbergen's effectiveness measure E=1− (1/ (α/P+ (1−α)/R)). Their relationship is Fβ = 1 − E where α = 1 / (β2 + 1).

The experimental results of the proposed method for Fusion approach for vision based deep web data [20] extraction for data extraction from the web are presented in this section. The proposed approach has been implemented in java (jdk 1.7) and the experimentation is performed on a 3.0 GHz Pentium PC machine with 2 GB main memory. For experimentation, we have taken many deep web pages which contained all the noises such as Navigation bars, Panels, Ads and Frames, Page Headers and Footers, Copyright and Privacy Notices, Advertisements and Other Uninteresting Data. These pages are then applied to the proposed method for removing the different noises. The removal of noise blocks and extracting of useful content chunks are explained in this sub-section.

VI. EMPIRICAL RESULTS

The sample of results obtained by the proposed approach is given in this sub-section. A sample of deep webpage [20] considered for experimentation is shown in Fig. 6.

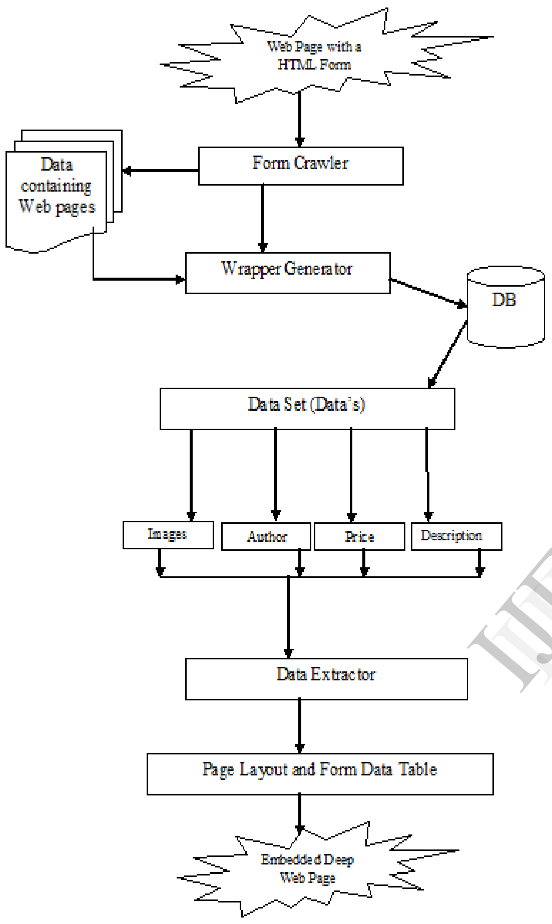


Figure 6. Deep Web Page Extraction

This Section shows the results of satisfaction prediction level by our proposed methods. From the collected web pages [17] snapshots, 70% of data is considered as a training set and the rest for testing.

A. Performance analysis of phase 1 of our technique

The performance analyses of the Vision based data extractions are presented in this section. Table 1 shows the experimental results on above said methods. Totally, F(Vi)DE performs significantly better than Vision based data extraction. *Precision:* The precision values of three methods are plotted as a graph shown in Fig 5, in which our proposed method F(Vi)DE performs better precision value. *Recall:* The recall values obtained for three different methods are plotted in the figure 6, in which our F(Vi)DE performs better recall value.

Table 1 Precision, Recall and F measure values for Web Data Extraction methods

Methods	**Precision**	**Recall**	**F Measure**
ViDE	0.9812	0.9123	0.945496
Parse Tree Generation	0.9910	0.9389	0.964247
VIPS	0.9509	0.9049	0.92733
Visual Architecture based Extraction	0.9054	0.8367	0.869695
Page Level Data Extraction	0.9823	0.9138	0.946813
F(Vi)DE	0.9965	0.9289	0.961513

Interestingly our proposed method called F(Vi)DE produces higher precision of 0.9965 than other web data extraction methods which is shown in the fig(7).

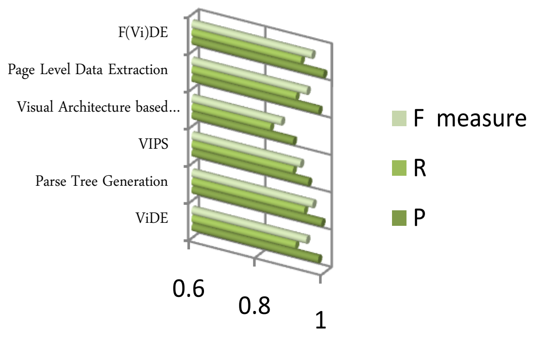


Figure 7. User Satisfaction Prediction

VII CONCLUSION

In this paper, we have implemented a new approach called vision-based deep [20] web data extraction for web data extraction. The web page information is classified into various chunks. From which, surplus noise and duplicate chunks are removed using three parameters, such as hyperlink percentage, noise score and cosine similarity. To identify the relevant chunk, three parameters such as Title word Relevancy, Keyword frequency-based chunk selection, Position features are used and then, a set of keywords are extracted from those main chunks. Finally, the extracted keywords are subjected to web data extraction using Fusion based deep data [20] extraction. Our experimental results showed that the proposed F(Vi)DE method can achieve stable and good results of about 99.65% and 92.89%. Finally we observe that, performance of vision based methods are very similar to each other which is shown in Table (1).

VIII FUTURE WORK

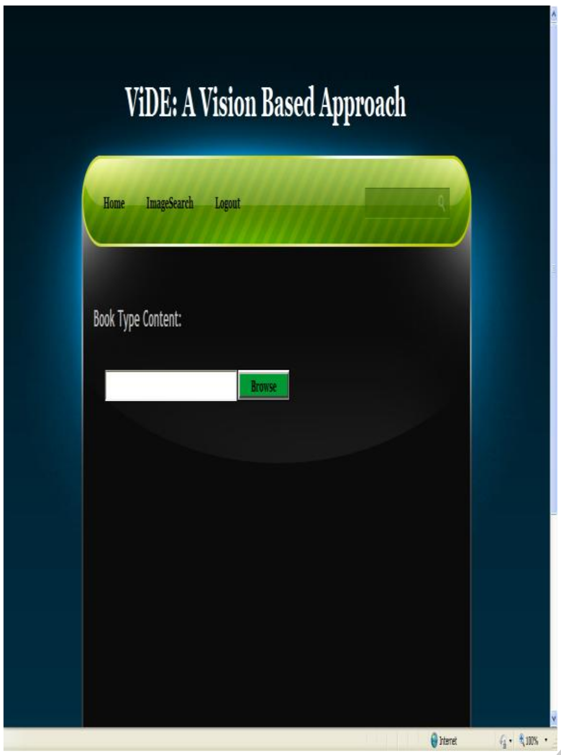
There are still some remaining issues and we plan to address them in future. F(Vi)DE can only process deep web pages containing one data region, while there is a significant number of multi data-region deep web pages. We intend to propose vision based approach to tackle HTML dependent and its performance. The efficiency of F(Vi)DE can be improved. In the current F(Vi)DE, the visual information of web pages is obtained by calling the programming AIPs of IE, which is time consuming process.

REFERENCES

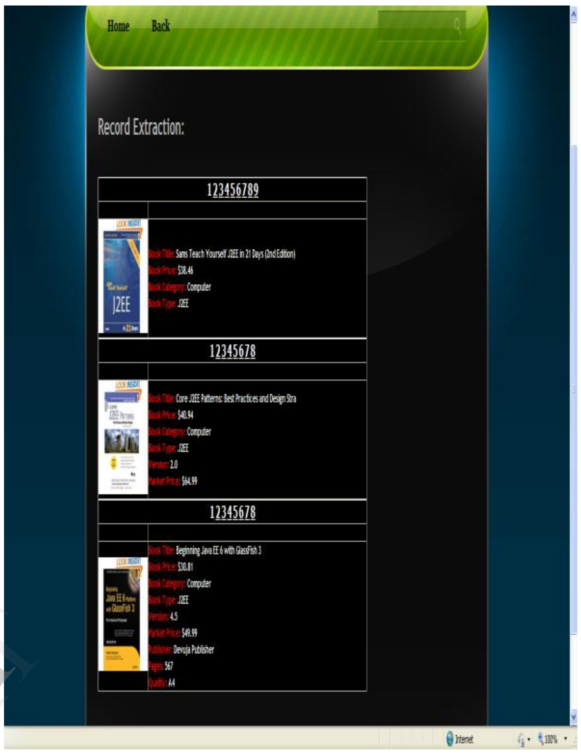
[1] G.O. Arocena and A.O. Mendelzon, "WebOQL: Restructuring Documents, Databases, and Webs," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 24-33, 1998.
[2] D. Buttler, L. Liu, and C. Pu, "A Fully Automated Object Extraction System for the World Wide Web," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS),* pp. 361-370, 2001.
[3] D. Cai, X. He, J.-R. Wen, and W.-Y. Ma, "Block-Level Link Analysis," *Proc. SIGIR,* pp. 440-447, 2004.
[4] D. Cai, S. Yu, J. Wen, and W. Ma, "Extracting Content Structure for Web Pages Based on Visual Representation," *Proc. Asia Pacific Web Conf. (APWeb),* pp. 406-417, 2003.
[5] C.-H. Chang, M. Kayed, M.R. Girgis, and K.F. Shaalan, "A Survey of Web Information Extraction Systems," *IEEE Trans. Knowledge and Data Eng.,* vol. 18, no. 10, pp. 1411-1428, Oct. 2006.
[6] C.-H. Chang, C.-N. Hsu, and S.-C. Lui, "Automatic Information Extraction from Semi-Structured Web Pages by Pattern Discovery,"*Decision Support Systems,* vol. 35, no. 1, pp. 129-147, 2003.
[7] V. Crescenzi and G. Mecca, "Grammars Have Exceptions," *Information Systems,* vol. 23, no. 8, pp. 539-565, 1998.
[8] V. Crescenzi, G. Mecca, and P. Merialdo, "RoadRunner: Towards Automatic Data Extraction from Large Web Sites," *Proc. Int'l Conf. Very Large Data Bases (VLDB),* pp. 109-118, 2001.
[9] D.W. Embley, Y.S. Jiang, and Y.-K. Ng, "Record-Boundary Discovery in Web Documents," *Proc. ACM SIGMOD,* pp. 467-478, 1999.
[10] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krpl, and B. Pollak, "Towards Domain Independent Information Extraction from Web Tables," *Proc. Int'l World Wide Web Conf. (WWW),* pp. 71-80, 2007.
[11] J. Hammer, J. McHugh, and H. Garcia-Molina, "Semistructured Data: The TSIMMIS Experience," *Proc. East-European Workshop Advances in Databases and Information Systems (ADBIS),* pp. 1-8, 1997.
[12] C.-N. Hsu and M.-T. Dung, "Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web," *Information Systems,* vol. 23, no. 8, pp. 521-538, 1998.
[13] http://daisen.cc.kyushu-u.ac.jpTBDW/, 2009.
[14] http://www.w3.org/html/wghtml5/, 2009.
[15] N. Kushmerick, "Wrapper Induction: Efficiency and Expressiveness," *Artificial Intelligence,* vol. 118, nos. 1/2, pp. 15-68, 2000.
[16] A. Laender, B. Ribeiro-Neto, A. da Silva, and J. Teixeira, "A Brief Survey of Web Data Extraction Tools," *SIGMOD Record,* vol. 31, no. 2, pp. 84-93, 2002.
[17] B. Liu, R.L. Grossman, and Y. Zhai, "Mining Data Records in Web Pages," *Proc. Int'l Conf. Knowledge Discovery and Data Mining (KDD),*pp. 601-606, 2003.
[18] W. Liu, X. Meng, and W. Meng, "Vision-Based Web Data Records Extraction," *Proc. Int'l Workshop Web and Databases (WebDB '06),* pp. 20-25, June 2006.
[19] L. Liu, C. Pu, and W. Han, "XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 611-621, 2000.
[20] Y. Lu, H. He, H. Zhao, W. Meng, and C.T. Yu, "Annotating Structured Data of the Deep Web," *Proc. Int'l Conf. Data Eng. (ICDE),* pp. 376-385, 2007.
[21] J. Madhavan, S.R. Jeffery, S. Cohen, X.L. Dong, D. Ko, C. Yu, and A. Halevy, "Web-Scale Data Integration: You Can Only Afford to Pay As You Go," *Proc. Conf. Innovative Data Systems Research (CIDR),* pp. 342-350, 2007.
[22] I. Muslea, S. Minton, and C.A. Knoblock, "Hierarchical Wrapper Induction [19] for Semi-Structured Information Sources," *Autonomous Agents and Multi-Agent Systems,* vol. 4, nos. 1/2, pp. 93-114, 2001.

EVALUATION RESULTS

Phase 1: Pass the Query from the user.



Phase 2: Record Extraction



Record Extraction:

123456789

Book Title: Sams Teach Yourself J2EE in 21 Days (2nd Edition)
Book Price: $38.46
Book Category: Computer
Book Type: J2EE

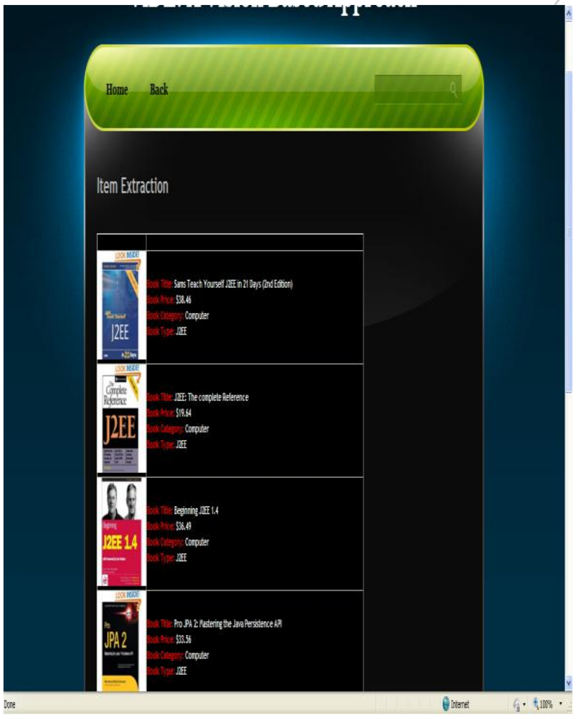
12345678

Book Title: Core J2EE Patterns: Best Practices and Design Stra
Book Price: $40.94
Book Category: Computer
Book Type: J2EE
Version: 2.0
Market Price: $64.99

12345678

Book Title: Beginning Java EE 6 with GlassFish 3
Book Price: $30.81
Book Category: Computer
Book Type: J2EE
Version: 4.5
Market Price: $49.99
Publisher: Devuja Publisher
Pages: 567
Quality: A4

Phase 3: Item Extraction



Item Extraction

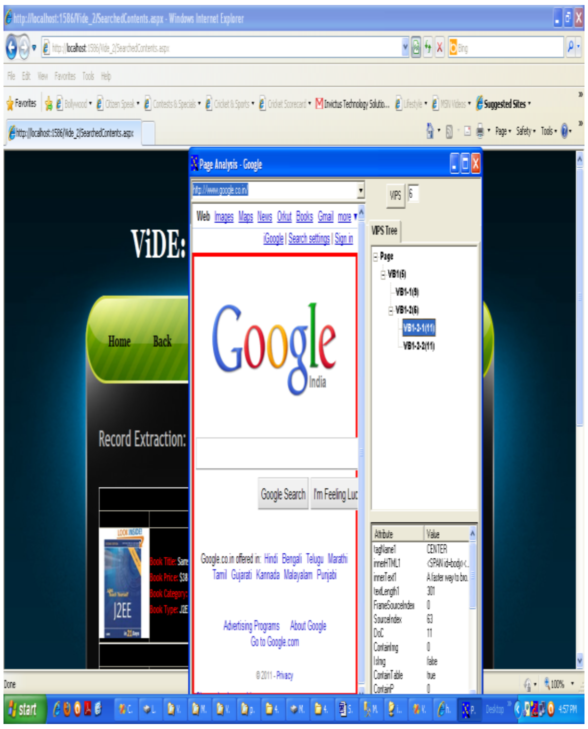
Book Title: Sams Teach Yourself J2EE in 21 Days (2nd Edition)
Book Price: $38.46
Book Category: Computer
Book Type: J2EE

Book Title: J2EE: The complete Reference
Book Price: $19.64
Book Category: Computer
Book Type: J2EE

Book Title: Beginning J2EE 1.4
Book Price: $36.49
Book Category: Computer
Book Type: J2EE

Book Title: Pro JPA 2: Mastering the Java Persistence API
Book Price: $33.56
Book Category: Computer
Book Type: J2EE

Phase 4: Web Page Segmentation



Attribute	Value
tagName1	CENTER
innerHTML1	<SPAN id=body><...
innerText1	A faster way to bro...
textLength1	301
FrameSourceIndex	0
SourceIndex	63
DoC	11
ContainImg	0
IsImg	false
ContainTable	true
ContainP	0