ISSN:0975-9646

Page-Level Data Extraction Approach for Web Pages Using Data Mining Techniques

K. Syed Kousar Niasi[1], Dr. E. Kannan[2], M. Mohamed Suhail[3]

[1]*Assistant Professor, Dept of Computer Science, Jamal Mohamed College, India.*
[2]*Dean. Dr. RR & SR University, VelTech, Avadi, India*
[3]*Lecturer, Dept of Computer Science, Jamal Mohamed College, Trichy, India.*

Abstract: Web data extraction has been an important part for many Web data analysis applications. In this paper, we formulate the data extraction problem as the decoding process of page generation based on structured data and tree templates[1]. We propose a unsupervised, page-level data extraction approach to deduce the schema and templates for each individual Deep Website, contains either singleton or multiple data records in one Webpage. FiVaTech applies tree matching, tree alignment, and mining techniques to achieve the challenging task. In experiments, FiVaTech has much higher precision than EXALG and is comparable with other record-level extraction systems like ViPER and MSE. The experiments show an encouraging result for the test pages used in many state-of-the-art Web data extraction works [3].

Keywords: Web data extraction, Multiple trees merging, wrapper induction.

1. INTRODUCTION

DEEP Web, as is known to everyone, contains magnitudes more and valuable information than the surface Web. However, making use of such consolidated information requires substantial efforts since the pages are generated for visualization not for data exchange. Thus, extracting information from Web pages for searchable Websites has been a key step for Web information integration. Generating an extraction program for a given search form is equivalent to wrapping a data source such that all extractor or wrapper programs return data of the same format for information integration[5].

An important characteristic of pages belonging to the same Website is that such pages share the same template since they are encoded in a consistent manner across all the pages. In other words, these pages are generated with a predefined template by plugging data values. In practice, template pages can also occur in surface Web (with static hyperlinks)[2]. For example, commercial Websites often have a template for displaying company logos, browsing menus, and copyright announcements, such that all pages of the same Website look consistent and designed. In addition, templates can also be used to render a list of records to show objects of the same kind. Thus, information extraction from template pages can be applied in many situations.

Finding such a common template requires multiple pages or a single page containing multiple records as input. When multiple pages are given, the extraction target aims at page-wide information When single pages are given, the extraction target is usually constrained to record wide information which involves the addition issue of record-boundary detection[4].

Page-level extraction tasks, although do not involve the addition problem of boundary detection, are much more complicated than record-level extraction tasks since more data are concerned. A common technique that is used to find template is alignment: either string alignment or tree alignment [3].

In this paper, we focus on page-level extraction tasks and propose a new approach, called FiVaTech, to automatically detect the schema of a Website. The proposed technique presents a new structure, called fixed/variant pattern tree, a tree that carries all of the required information needed to identify the template and detect the data schema.

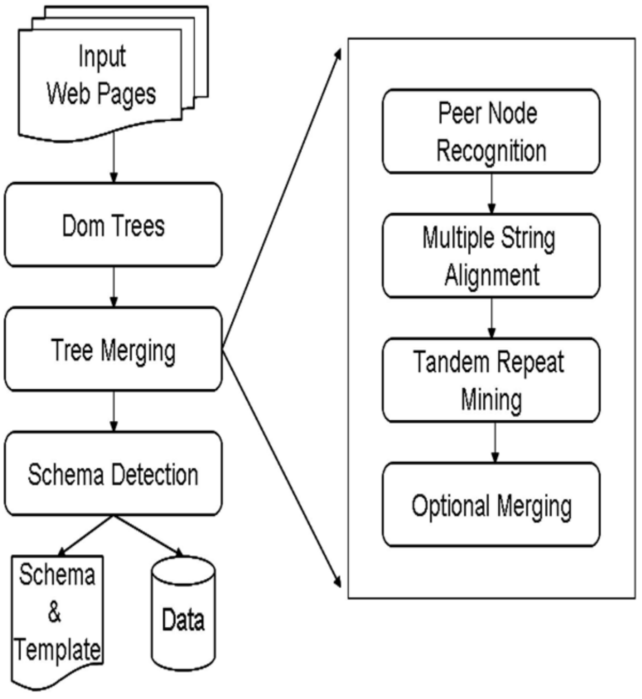


Figure 1.The FiVaTech approach for wrapper induction.

2. PROBLEM FORMULATION

The development of FivaTech all pages, occur quite fixed as opposed to data values which vary across pages. Finding such a common template requires multiple pages or a single page containing multiple records as input. When multiple pages are given, the extraction target aims at page - wide information.

When single pages are given, the extract ion target is usually constrained to record wide information, which involves the addition issue of record - boundary detection. Page - level extraction tasks, although do not involve the addition problem of boundary detection, are much more complicated than record – level extraction tasks since more data are concerned. As for the problem of distinguishing template and data, most approaches assume that HTML tags are part of the template.

All data instances of a web site shall conform to a common schema which can be defined as follows.

Definition 1: (Structured data) A data schema can be of the following types:

• A basic type β represents a string of tokens which are basic units of text.

• If τ1 ,τ2 ,. . . ,τk are types, then their order list ¡τ1 ,τ2 ,. . . ,τk ¿ also forms a type τ . We say the type τ is constructed from the types τ1 ,τ2 ,. . . ,τk using a type constructor of order k. An instance of the k-order τ is of the form ¡x1 , x2 , ..., xk ¿ where x1 , x2 , ..., xk are in- stances of types τ1 ,τ2 ,. . . ,τk respectively. The type τ is called

1. a tuple, denoted by ¡k¿τ , if the cardinality (the number of instances) is 1 for every instantiation.
2. an option, denoted by (k)?τ , if the cardinality is either 0 or 1 for every instantiation.
3. a set, denoted by {k}τ , if the cardinality is greater than 1 for some instantiation.
4. a disjunction, denoted by(τ1 |τ2 |...|τk)τ , if all τi (i = 1, ..., k) are options and the cardinality sum of the k options: τ1 to τk equal to 1 for every instantiation of τ .

Definition 2: (Wrapper Induction) Let λ(TΩ , D) de- notes the generation model of some Web pages at time t, where TΩ denotes the templates for schema Ω and D de- notes its extracted data. The problem of page-level wrapper verification is to decide whether a new Web page P at time t' has been changed from its generation model λ(TΩ , D). We call this problem a record-level wrapper verification if Ω is simply a set of k-tuples.

In this paper, we assume that all of the peer nodes must be in the same DOM tree level which is not true for all Web sites. We adopt FivaTech, a page-level, unsupervised wrapper induction approach which merges the input DOM trees into a pattern tree. A pattern tree removes duplicate occurring patterns of set types and contains one representation for each data types (tuple, option, disjunction, etc). As an example, Figure 2 shows a pattern tree which contains the merged DOM tree and detected schema, where we have a set, two tuples, two options and five basics.

Definition 3: Given a set of n DOM trees, DOM I = λ(T, xi) (1≤i≤n), created from some unknown template T and values { x 1 ,. . .,x n }, deduce the template and values, from the set of DOM trees alone. We call this problem a page-level information extraction. If one single page (n=1) which contains tuple constructors is given as input, the problem is to deduce the template for the schema inside the tuple constructors. We call this problem a record-level information extraction task.

3. System Design

3.1 Multi-tier data center Architecture

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase.

Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified. Design will explain software components in detail.

This will help the implementation of the system. Moreover, this will guide the further changes in the system to satisfy the future requirements.

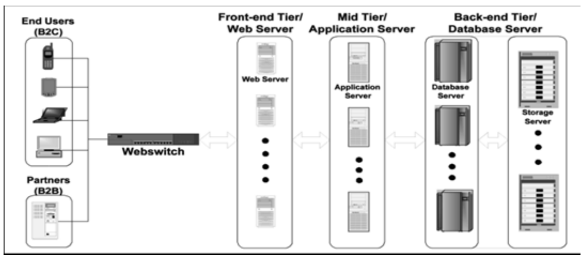


Figure 2. Multi-tier data center Architecture

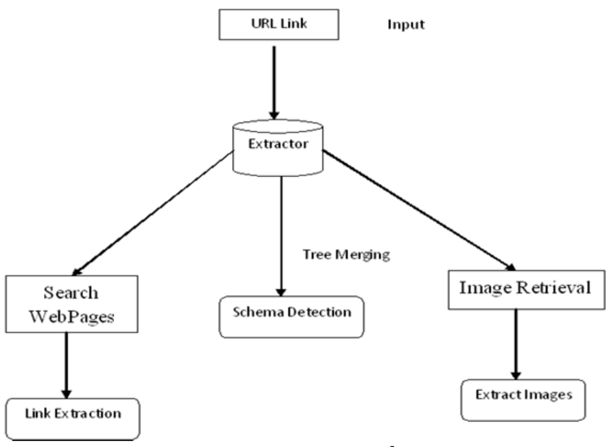


Figure 3. System Architecture

4. FiVaTech Approach

4.1 FiVaTech Tree Merging

There are two main phases in FiVaTech. The first phase is merging input DOM trees to construct the fixed/variant tree, and the second phase is detecting the schema and the template of a Web site based on the constructed pattern tree.

1. In the peer node recognition step, a modified tree edit distance is designed to calculate a matching score for two nodes with the same tag 1 name such that set data with various occurrences still have high similarity.
2. In the second step, the child nodes will fill up a matrix such that all peer child nodes (similar sub trees measured in the previous step) will be denoted with the same symbol. The matrix alignment algorithm then traverses the matrix to obtain an aligned peer matrix.

3. In pattern mining step, the algorithm discovers repetitive patterns and merge them to deduce the aligned list.
4. Finally, in the last step, optional node merging, the algorithm detects optional nodes based on the occurrence vectors. If a set of adjacent nodes have the same occurrence vectors, they will be grouped as optional. If a set of adjacent optional nodes have complement occurrence vectors, they will be grouped as disjunction.

4.2 Schema Detection

In this phase, FiVaTech detects the structure of the web site which includes identifying the schema and defining the template for each type constructor of this schema. By the end of the previous phase, basic type, set type and optional type are already identified. The remaining task for schema detection in this phase is to recognize tuple type as well as the order of the set type and the optional data.

FiVaTech traverses the fixed-variant pattern tree from the root downward and marks nodes as k-order (if the node is already marked as some data type) or k-tuple. For nodes with only one child and not marked as set or optional types, there is no need to mark it as 1-tuple (otherwise, there will be too many 1-tuples in the schema), thus the system simply traverses down the path to discover other type nodes. For nodes with more than one branch (child), the system will mark them as k-order if k children contain a data type.

Finally, the schema tree S can then be obtained by excluding all of the tag nodes that have no types.

Note that FiVaTech uses tree edit distance to measure the similarity among nodes with the same tag in the same level of the inputted Web pages, which exploits only structural information to measure the similarity. Since visual information is recommended and important for similarity measure, we propose a classifier-based approach for peer node recognition.

4.3 Peer Matrix Alignment

After peer node recognition, all peer sub trees will be given the same symbol. For leaf nodes, two text nodes take the same symbol when they have the same text values, and two <img> tag nodes take the same symbol when they have the same SRC attribute values. To convert M into an aligned peer matrix, we work row by row such that each row has (except for empty columns) either the same symbol for every column or is a text (<img>) node of variant text (SRC attribute, respectively) values. In the latter case, it will be marked as basic-typed for variant texts. From the aligned matrix M, we get a list of nodes, where each node corresponds to a row in the aligned matrix.

4.4 Pattern Mining

This pattern step is designed to handle set-typed data, where multiple values occur; thus, a naive approach is to discover repetitive patterns in the input. However, there can be many repetitive patterns discovered and a pattern can be embedded in another pattern, which makes the deduction of the template difficult. The good news is that we can neglect the effect of missing attributes (optional data) since they are handled in the previous step. Thus, we should focus on how repetitive patterns are merged to deduce the data structure. In this section, we detect every consecutive repetitive pattern (tandem repeat) and merge them (by deleting all occurrences except for the first one) from small length to large length. This is because the structured data defined here are nested and if we neglect the effect of optional, instances of a set-type data should occur consecutively according to the problem definition.

4.5 Filtering Out Template Blocks in the Inputted DOM Trees

The Deep Web usually contains two types of blocks in the generated Web pages: template data blocks and data rich blocks. Template data blocks are the frames/sections of the Web pages that contain template data such as advertisements, navigational panels and so on. Data rich blocks are the frames/sections of the Web pages that contain relevant data of interest to the user. Although template blocks can be detected by FiVaTech through recursive comparison of peer nodes from root, the process could be quite time consuming. To improve the efficiency, we propose an image- based step to filter out template blocks before applying the peer nodes recognition step for tree merging.

Our algorithm filtering Template Blocks has two main assumptions. First, template blocks for various pages of a Web site are displayed with the same content. Not only does the rendered image look the same, the tags that correspond to such template blocks also co-located in the same path of the DOM trees. Second, the area for a data rich block often occupies the biggest area in the whole Web page. Based on the first assumption, we can remove template sub- trees in the preprocessing step.

However, subtrees with the same images in a data rich block are usually not template. As shown in Figure 5, the algorithm recursively traverses one of the inputted DOM trees from the root downward and checks for the existence of some child node c with percentage Area(c) > 40% where percentageArea(c) is defined as the percentage of the image area corresponds to node c to the whole area of the displayed page):

$$\text{percentageArea}(c) = \frac{\text{nodeArea}(c)}{\text{nodeArea}(<\text{Body}>)}\ \%$$

If all child nodes have percentage Area less than 40%, the algorithm stops or the algorithm identifies the child node with the biggest percentage Area value. The algorithm then keeps the biggest node and all remaining children nodes that have no identical subtrees in the other DOM trees.

Our experiments show that these values are useful and give good results.

- Parent: the parent node in the DOM tree.
- TextContent: the text contents within the subtree.
- NoChildren: number of child nodes in the subtree.

• ChildHeight: The depth of the subtree of the current node
• LeafNode: whether the node has children or not (Boolean feature).
• ClassAttr: The class name of the HTML tag.
• Path: The numbered path of the DOM tree node.

The partial path includes the tag name and its parent tag name. Each tag name is followed by a number representing location in the parent node (from left to right starting at 1).

• Token: the number of tokens in leaf text content.
• Digit: the number of digits in leaf text content.
• Letter: the number of letters in leaf text content.
• UpperCase: the number of capitalized letter in leaf text content.
• LowerCases: the number of lower case.

The appropriate values of the attributes are based on our empirical results. For example, the value "Both" of the at- tribute Node1IsLeaf means, that both node1 and node2 are leaves. As another example, the value "Sim" of the at- tribute Same Children means that the percentage of the size difference of the two nodes is less than 10%. Also, the value "Empty" means that the contents of the two nodes are empty. Our experiments show that these values are useful and give good results.

5. Related Works:

Given a set of training pages from a Web site, we use DOM trees of the Web pages as input to detect the schema of this site we try to merge all DOM trees at the same time into a single tree called a fixed/variant pattern tree. From this pattern tree, we can recognize variant leaf nodes for basic - typed data and mine repetitive nodes for set - typed data. The resulting pattern tree is then used to detect the template and the schema of the Web site. The key challenge here is how to merge multiple trees at the same time. Our solution is to break down the multiple trees merging problem from a tree level to a string level and design a new algorithm for multiple string alignment that considers both missing data and multiple - value data.

5.1 Approaches using DOM Tree

Information many approaches for Web data extraction consider and operate on DOM tree structure. MDR [18] analyzes the child nodes under each parent node and finds generalized nodes and data regions by enumerating possible combinations of child nodes. In MDR string edit distance is used to compute the similarity between tag sequences of two generalized nodes. However, the goal of MDR is to identify data records. MDR does not align the data items in each data record. Meanwhile, due to missing and noisy information, it may find wrong combination of sub trees, DEPTA [8] uses visual gaps between data records to find out data records t uses partial tree alignment technique to align data fields of data records. NET [10] extends DEPTA by supporting extraction of nested records. ViPER [4] uses primitive tandem repeats and visual context information for record segmentation and enhances the concept of generalized nodes. This provides a better subtree comparing method than MDR which allows consecutive data records with various lengths. In DeLa[9] sufix trees are built to detect C-Repeated patterns in webpage string and its algorithm can extract the nested objects. FiVaTech [1] uses tree matching score for subtree comparison, however, the bigger goal is to find the schema and template for the whole page.

5.2 Approaches using Visual Information

Some approaches improve the task of web data extraction by using the visual information. ViNTs [2] and MSE [3] use visual content features on a browser to identify candidate content line. ViPER [4] uses visual information for global multiple sequence alignment. Although visual information is used in these approaches, for similarity calculation they still use HTML tag structure as primary information. ViDE[5] constructs a visual block tree. Its main visual features are position features, layout features, appearance features, and content features and they can be obtained from web page layout (location, size, and font).

5.3 Page - level Extraction Systems

EXLAG [6] and Road Runner [7] are unsupervised systems for page level web data extraction. Road Runner extracts data by comparing a pair of web pages to get the template. *It works in three steps: A (Align), CM (Collapse under Mismatch), and E (Extract).* It supports the backtracking mechanism if optional or iterated tags are found. EXLAG extracts data by forming and analyzing equivalence classes. In EXLAG d Tokens (Differentiating Tokens) are aggregated in equivalence classes if they have same occurrence frequency in all input web pages. For template generation large and frequent equivalence classes (LFEQs) are extracted EXALG and Road Runner operates on HTML tags, while FiVaTech manipulate DOM trees in order to find out peer nodes (i.e. nodes with the same tag names but different Roles)

6. Experiments

6.1 Performance Metrics

We conducted two experiments to evaluate the schema resulted by our system and compare FiVaTech with other recent approaches. The first experiment is conducted to evaluate the schema resulted by our system, and at the same time, to compare FiVaTech with EXALG [1]; the page- level data extraction approach that also detects the schema of a Website. The second experiment is conducted to evaluate the extraction of data records or interchangeably search result records (SRRs), and compare FiVaTech with the three state-of-the-art approaches: DEPTA, ViPER, and MSE.

To conduct the second experiment, FiVaTech has an extra task of recognizing data sections in a Website. A data section is the area in the Webpage that includes multiple instances of a data record (SRRs). FiVaTech recognizes the set of nodes n_{SRRs} in the schema tree that corresponds to different data sections by identifying the outermost set type nodes, i.e., the path from the node n_{SRR} to the root of the schema tree has no other nodes of set type. A special case is when the identified node n_{SRR} in the schema tree has only one child node of another set type, this means that data records of this section are presented in more than one column of a Webpage, while FiVaTech still catches the data.

Given a set of Web pages of a Website as input, FiVaTech outputs three types of files for the Website. The first type (a text file) presents the schema (data values) of the Website in an XML-like structure. We use these XML files in the first experiment to compare FiVaTech with EXALG. The second type of file (an html file) presents the extracted SRRs (of each dynamic section) of the test and the training Web pages of the Website. A simple extractor program that uses both the identified n_{SRR} nodes in the schema tree and the templates associated with these nodes is implemented to output these HTML files. We use these files in the second experiment to evaluate FiVaTech as an SRRs extractor and compare the system with the three record-level approaches DEPTA, ViPER, and MSE. Finally, the third type of file (an Excel file) contains the data items of the set of all attributes of a basic type; every column in the file has the set of all instances of a basic type that are collected from the test and the training Web pages. We use these Excel files in the second experiment to compare the alignment results of FiVaTech with the alignment results of DEPTA.

6.1 FiVaTech as Schema Extractor

Given the detected schema S_e of a Website and the manually constructed schema S_m for this site, EXALG evaluates the resulted schema S_e by comparing data extracted by leaf attributes A_e of this schema from collections of Web pages of this site. However, this is not enough for two reasons. First, many Web applications (e.g., information integration systems) need such schemas as input, so it is very important to evaluate the whole schema S_e. Second, for Web data extraction, the values of an attribute may be extracted correctly (partially correct as defined by EXALG [1]) but its schema is incorrect, and vice versa. For example, the first instance of a repetitive data record is often excluded from the set but is recognized as a tuple. Thus, all instances of the data record are extracted although the schema is wrong (the first instance is identified as of a tuple type while the remaining are instances of a set type). Meanwhile, many disjunctive types and empty types (corresponding to no data in the schema S_m) are extracted by EXALG but are considered correct because they did not extract wrong results.

6.2 FiVaTech as a SRRs Extractor

The popular approaches that extract SRRs from one or more data sections of a Webpage, the main problem is to detect record boundaries. The minor problem is to align data inside these data records. However, most approaches concern with the main problem except for DEPTA, which applies partial tree alignment for the second problem. Therefore, we compare FiVaTech with DEPTA in both steps and focus on the first step when comparing with ViPER and MSE.

In the second experiment (a comparison with DEPTA), we configure FiVaTech to detect the schema from a single Webpage, although this will give an incorrect schema outside the span of sections of multiple data records (n_{SRRs}), but we are only concerning with data sections and the SRRs inside each section. We got the system demo from the author and ran DEPTA on the manually labeled Testbed for Information Extraction from Deep Web TBDW [12] Version 1.02 available at http://daisen.cc.kyushu- u.ac.jp/TBDW/. Unfortunately, DEPTA gave a result only for 11 Websites and could not produce any output for the remaining 40 sites. So, we conducted the following experiment for these 11 Websites. For SRRs extraction, we just used the Web pages that have multiple data records. DEPTA gave a good result for six Websites and extracted incorrect SRRs for four Websites. For the last Website (the site numbered 13 in Test bed), DEPTA merged every two correct data records and extracted them as a single data record. We considered half of the data records are not extracted for this last site.

The last experiment compares FiVaTech with the two visual-based data extraction systems, ViPER and MSE. The first one (ViPER) is concerning with extracting SRRs from a single (major) data section, while the second one is a multiple section extraction system. We use the 51 Websites of the Testbed referred above to compare FiVaTech with ViPER, and the 38 multiple sections Websites used in MSE to compare our system with MSE. Actually, extracting of SRRs from Web pages that have one or more data sections is a similar task. The results in Table 3 show that all of the current data extraction systems perform well in detecting data record boundaries inside one or more data sections of a Webpage. The closeness of the results between FiVaTech and the two visual-based Web data extraction systems ViPER and MSE gives an indication that until this moment visual information do not provide the required improvement that researchers expect. This also appeared in the experimental results of ViNTs [15]; the visual-based Web data extraction with and without utilizing visual features. FiVaTech fails to extract SRRs when the peer node recognition algorithm incorrectly measures the similarities among SRRs due to the very different structure among them. Practically, this occurred very infrequently in the entire test page (e.g., site numbered 27 in the Testbed). Therefore, now, we can claim that SRRs extraction is not a key challenge for the problem of Web data extraction.

On a Core 2 Duo (2.00 GHz) laptop, the response time is about 5-50 seconds, where the majority of time is consumed at the peer node recognition step. Therefore, the running time of FiVaTech has a wide range (5-50 seconds) and leaves room for improvement.

6.3 Performance Evaluation

TABLE1 Performance on 11 Websites from Testbed Version 1.02

	SRRs Extraction		Alignment	
	#Actual SRRs: 419		#Actual attributes: 92	
	DEPTA	FiVaTech	DEPTA	FiVaTech
#Extracted	248	409	93	91
#Correct	226	401	45	82
Recall	53.9%	95.7%	48.9%	89.1%
Precision	91.1%	98.0%	48.4%	90.1%

Table 2 Performance Comparison between ViPER and MSE

Dataset	TBDW [12]		MSE [16]	
#Actual SRRs:	693		1242	
System	ViPER	FiVaTech	MSE	FiVaTech
#Extracted	686	690	1281	1260
#Correct	676	672	1193	1186
Recall	97.6%	97.0%	96.1%	95.5%
Precision	98.5%	97.4%	93.1%	94.1%

Chart 1.Performance on 11 Websites from Testbed Version 1.02

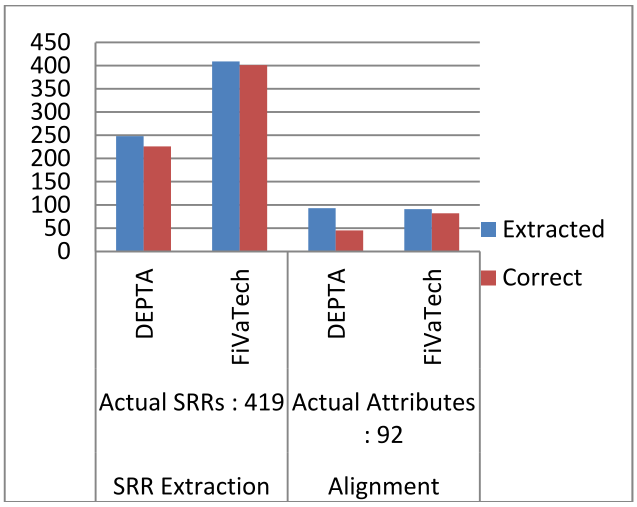
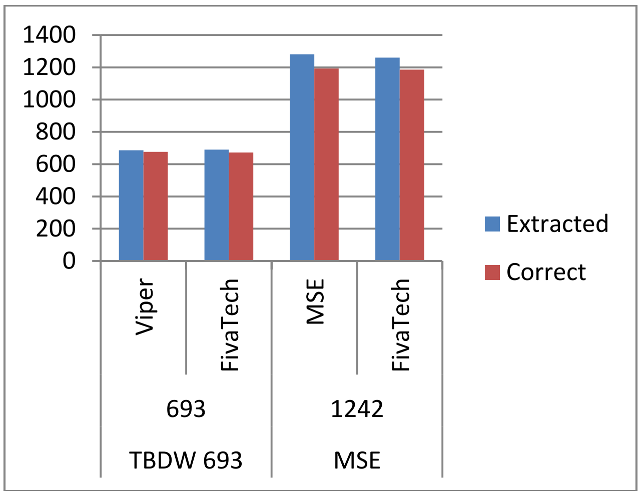


Chart 2.Performance Comparison between ViPER and MSE



7. Conclusion

We can highlight three conclusion, Grid applications have an increasing need of database systems. Combining Grid and database technologies is an essential approach to meet the requirements of large-scale Gird applications. Almost every application running on Grid has many requirements for access to structured data, Grid as a platform with their resources can provide many benefits for such king of database system.

7.1 Future Enhancement

Future improvements in project management may be made through better tools and practices. In future the develop extraction template pages using on tree matching algorithm and also used DOM tree matching algorithm in well successful new project to the corresponding developer.

Reference

[1] A. Arasu and H. Garcia-Molina, “Extracting Structured Data from Web Pages,” Proc. ACM SIGMOD, pp. 337-348, 2003.
[2] C.-H. Chang and S.-C. Lui, “IEPAD: Information Extraction Based on Pattern Discovery,” Proc. Int’l Conf. World Wide Web (WWW-10), pp. 223-231, 2001.
[3] C.-H. Chang, M. Kayed, M.R. Girgis, and K.A. Shaalan, “Survey of Web Information Extraction Systems,” IEEE Trans. Knowledge and Data Eng., vol. 18, no. 10, pp. 1411-1428, Oct. 2006.
[4] V.Crescenzi, G. Mecca, and P. Merialdo, “Knowledge and Data Engineerings,” Proc. Int’l Conf. VeryLarge Databases (VLDB), pp. 109-118, 2001.
[5] C.-N. Hsu and M. Dung, “Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web,” J. Information Systems, vol. 23, no. 8, pp. 521-538, 1998.
[6] N.Kushmerick, D. Weld, and R. Doorenbos, “Wrapper Induction for Information Extraction,” Proc. 15th Int’l Joint Conf. Artificial Intelligence (IJCAI), pp. 729-735, 1997.
[7] A .H .F. Laender, B.A. Ribeiro-Neto, A.S. Silva, and J.S. Teixeira, “A Brief Survey of Web Data Extraction Tools,” SIGMOD Record, vol. 31, no. 2, pp. 84-93, 2002.
[8] B. Lib, R. Grossman, and Y. Zhai, “Mining Data Records in Web pages,” Proc. Int’l Conf. Knowledge Discovery and Data Mining (KDD), pp. 601-606, 2003.
[9] I. Muslea, S. Minton, and C. Knoblock, “A Hierarchical Approach to Wrapper Induction,” Proc. Third Int’l Conf. Autonomous Agents (AA ’99), 1999.
[10] K. Simon and G. Lausen, “ViPER: Augmenting Automatic Information Extraction with Visual Perceptions,” Proc. Int’l Conf. Information and Knowledge Management (CIKM), 2005.
[11] J. Wang and F.H. Lochovsky, “Data Extraction and Label Assignment for Web Databases,” Proc. Int’l Conf. World Wide Web (WWW-12), pp. 187-196, 2003.
[12] Y. Yamada, N. Craswell, T. Nakatoh, and S. Hirokawa, “Testbed for Information Extraction from Deep Web,” Proc. Int’l Conf. World Wide Web (WWW-13), pp. 346-347, 2004.
[13] W. Yang, “Identifying Syntactic Differences between Two Programs,” Software—Practice and Experience, vol. 21, no. 7, pp. 739- 755, 1991.
[14] Y. Zhai and B. Liu, “Web Data Extraction Based on Partial Tree Alignment,” Proc. Int’l Conf. World Wide Web (WWW-14), pp. 76-85, 2005.
[15] H.Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu, “Fully Automatic Wrapper Generation for Search Engines,” Proc. Int’l Conf. World Wide Web (WWW), 2005.
[16] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu, “Automatic Extraction of Dynamic Record Sections from Search Engine Result Pages,” Proc. Int’l Conf. Very Large Databases (VLDB), pp. 989-1000, 2006.
[17] M. Kayed, C.-H. Chang. “FiVaTech: Page-Level Web Data Extraction from Template Pages”, IEEE TKDE, vol. 22, no. 2, pp. 249-263, Feb. 2010.
[18] B. Liu, R. Grossman. Y. Zhai. “Mining data records from Web pages.” KDD-03, 2003.