ViPER: Augmenting Automatic Information Extraction with Visual Perceptions

Kai Simon Georg Lausen

Institut für Informatik, Universität Freiburg
Georges-Köhler-Allee, Gebäude 51
79110 Freiburg i.Br., Germany

{ksimon,lausen}@informatik.uni-freiburg.de

ABSTRACT

In this paper we address the problem of unsupervised Web data extraction. We show that unsupervised Web data extraction becomes feasible when supposing pages that are made up of repetitive patterns, as it is the case, e.g., for search engine result pages. Hereby the extraction rules are generated automatically without any training or human interaction, by means of operating on the DOM tree respectively the flat tag token sequence of a single page.

Our contribution to automatic data extraction through this paper is twofold. First, we identify and rank potential repetitive patterns with respect to the user's *visual* perception of the Web page, well aware that location and size of matching elements within a Web page constitute important criteria for defining relevance. Second, matching subsequences of the pattern with the highest weightiness are aligned with *global* multiple sequence alignment techniques. Experimental results show that our system is able to achieve high accuracy in distilling and aligning regularly structured objects inside complex Web pages.

Categories and Subject Descriptors

H.3.m [**Information Storage and Retrieval**]: Miscellaneous—*Data Extraction, Wrapper Generation, Web*

General Terms

Algorithms, Experimentation, Performance

Keywords

Data extraction, data record alignment, visual features

1. INTRODUCTION

Rife information content available on the World Wide Web is published through representation-oriented semi-structured HTML pages. According to the process of their generation they could be divided into *static* and *dynamic* pages. Static pages, bearing stable or rarely changing content, can easily be crawled and indexed by common search engines. Dynamic pages, on the other hand, consist of rapidly changing content, e.g., news sites or pages changing their content based on user requests. Typically these sites are filled with information from back-end databases and generated by predefined templates or server-sided scripts. Although they have a unique URL address, crawling and indexing becomes otiose by virtue of their volatile nature. Hence there arises the need for new information services that can help users locate information in the *Hidden Web*. Possible solutions might be high dimensional meta-search engines with hundreds of thousands of subordinate information resources or Web agents scanning the Web for hidden information. Hereby both approaches have to be equipped with mechanisms which distill the relevant information and subsequently align the extracted data in order to provide more suitable data representation.

Under the aspect of extracting information from thousands of information resources manual or semi-automatic approaches become infeasible, therefore this paper presents a new fully-automatic information extraction tool named **ViPER** (**Vi**sual **P**erception-based **E**xtraction of **R**ecords). The principle assumption made is that a Web page contains at least two multiple spatially consecutive *data records* building a *data region* which exhibits some kind of structural and visible similarity. ViPER is then able to extract and discriminate the relevance of different repetitive information contents with respect to the user's *visual* perception of the Web page. Having identified the most relevant data region the tool aligns these records utilizing a fast and robust *multiple sequence alignment* (MSA) technique. In our opinion *global alignment* techniques, which have not yet been applied, bear most flexible opportunities in this context. To show the efficiency and accuracy of the extraction and alignment process we used an available third-party test bed with manually extracted and labeled data. Additionally we compared ViPER with existing state-of-the-art wrapping tools resulting in encouraging results.

2. RELATED WORK

Wrapper tools for extracting information from HTML pages started attracting major research interest during the midnineties. One significant characteristic is their degree of au-

CIKM'05, October 31–November 5, 2005, Bremen, Germany.

tomation, reaching from specially designed standalone wrapper programming languages, for manual wrapper generation, over machine learning, and interactive approaches with more or less human interaction to fully-automatic wrapper tools. We refer the interested reader to [12] for a brief survey of different wrapping techniques. Whereas the majority of the approaches rely on user interaction to generate extraction rules, more recently, interest in automatically generate wrappers without human involvement has grown substantially. Most relevant to our approach are IEPAD[4], DeLa[16], MDR[13] and recently ViNTs[14], and DEPTA[19] which fall into the category of fully-automatic wrapper tools.

IEPAD generates extraction rules utilizing a decoded binary string of the HTML tag sequence and tries to find maximal repeated patterns using a *PAT tree*, which is rather similar to a suffix tree. The extracted pattern becomes generalized using multiple string alignment techniques. Finally the user has to choose one of these generalized patterns as an extraction rule. The approach is fairly simple and fast but tests in [13] have shown that it provides poor results when the data records have a complex, nested structure. In our system we also search for maximal repeats but instead of trying to find data records with this technique we use it in the context of data record alignment.

DeLa tries to overcome the problem of nested-structured data records recognition by multi-level *continuous repeat* (c-repeat) detection using suffix trees. The drawback of this approach is that repetitive structures are not that obvious contained in a data record because of optional tag elements. Finally heuristics are needed to prune the number of different patterns discovered. Inspired by this work we integrated a *tandem repeat* (c-repeats) identification instance into our approximate pattern search to cope with additional repetitive tag elements which often discards possible matches.

ViNTs [14] fully-automatically generates *SRR* (*search result record*) extraction rules using visual context features and tag structure information. Therefore it first utilizes the visual content without considering the tag structure to identify content regularities denoted as *content lines* and then combines them with the HTML tag structure regularities to generate wrappers. To weight the relevance of different extraction rules visual and non-visual features have been used. ViNTs builds a wrapper for a search engine using several result pages and one no-result page. The resulting wrapper is represented by a regular expression of alternative horizontal separators tags, i.e., <HR> or

, which segment descendants into SRRs. Due to the fact that the ViNTs system only supplies horizontal separators, which is sufficient when considering document result pages, it could not separate horizontally arranged data records which will require vertical separators. Additionally at least four data records have to be present in a Web page for wrapper building and in case the data records are distributed over multiple sections only the major section is reported.

MDR [13] identifies data regions by searching for multiple generalize-nodes using edit-distance similarity where generalize-nodes are a fix combination of multiple child nodes and their corresponding subtrees. MDR does not identify the most relevant data records but rather reports each repetitive sub-region contained in a Web page. Recently the authors proposed an improvement of their system named DEPTA [19] (MDR-2) operating on a tag tree built according to visual rendering information. Additionally they mention that gap information is incorporated to eliminate false node combinations but nothing is said about the realization. Finally they proposed an approach for data record alignment by progressively growing a seed tag tree. The alignment is *partial* because only these nodes of a data record become aligned whose position for inserting into the seed tree can be uniquely determined. They tested their MDR system on a collection of handpicked sample pages with near perfect results (precision 100% and recall 99.8%) compared to (56%,39%) OMINI [2] and (67%,39%) IEPAD. Test result reported in [14] indicated that the performance on third-party test beds yield to worse results. In [19] they showed the improvements of their new system DEPTA and reported encouraging results for the alignment process.

Our objective in this paper is to enhance the extraction technique realized in the MDR by identifying tandem repeats and visual context information for record segmentation which has not yet been proposed. Additionally we only report the relevant data records according to the visual perception of a Web page in contrast to MDR and DEPTA. We do not extract data records according to separator tags like ViNTs does, rather we consider the tag structure that data records consist of. Thus we could also manage Web pages containing at least two consecutive data records, extract similar data records distributed over multiple sections, and extract horizontally arranged data records. The visual relevance weighting mechanism is similar to ViNTs but we computed it in a different way because our data records may range over multiple sections. Our data record alignment method is totally different to the method proposed for DEPTA because of global matching mechanisms and incorporated text content information.

3. DATA EXTRACTION

In this section we will describe in detail our contributions made to improve the extraction power. First, we carry out a pre-processing step to enhance the recognition robustness. Next, we adapt the edit-distance metric to cope with typical HTML structures. Based on the observation in [13] that a data record is either formed by a single coherent subtree (*subtree pattern*) or ranges over *multiple* adjacent sibling nodes (*forest tree pattern*) the pattern search is restricted to child subtrees belonging to the same parent node. This search technique has already successfully been implemented in the MDR system. We improve this technique by *variable* rather than *fix* node combinations, and finally incorporate *visual* information to separate and weight the identified data regions.

3.1 Definition

Semi-structured data can be described as data which is neither raw nor very strictly typed as in traditional database systems. In case of HTML, predefined markup tags could be used to control the appearance of untyped text. Therefore we could formalize HTML documents as a class of *labeled unordered trees*. A labeled unordered tree is a directed acyclic graph $T = (\mathcal{V}, \mathcal{E}, r, \delta)$ where $\mathcal{V}$ denotes a set of nodes with a distinguished node r called the *root*, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ a set of edges between two different nodes and a label function $\delta : \mathcal{V} \times \mathcal{L}$ where $\mathcal{L}$ is a string. If $(u, v) \in \mathcal{E}$ we call u *parent node* of v, and v *child node* of u. Each node $v \in \mathcal{V} \setminus \{r\}$ has exactly one parent node in the graph. If node u is an inner node then $\mathcal{C}_u = \{v \,|(u, v) \in \mathcal{E}\}$ denotes the set of child nodes

belonging to u. For a node v, we define T_v as the *subtree* of T rooted at v, where the subgraph of T_v is induced by the set of all descendants.

3.2 Pre-Processing

To enhance the pattern extraction accuracy in HTML documents pre-processing has to be performed. For instance, numerous Web pages are not even well-formed with respect to their HTML syntax. We used the Open Source Mozilla Browser and its underlying rendering engine Gecko to convert HTML documents into valid XHTML. The reason is that most HTML authors verify their pages with standard browsers before publishing their sites. We controlled Gecko over the XPCOM interface via the Java Framework JREX (http://jrex.mozdev.org/) which enables us to use the correction power of Gecko within Java. Furthermore, we have the ability to access the parsed document tree T^* with additional rendering information whereas every tag element is augmented with bounding box information by the upper-left corner's (x,y) pixel coordinates along with width and height. Furthermore, we compute the matching bounding box of text elements. To analyze T^* we work on an abstract representation where each HTML tag is restricted to its tag name ignoring attributes. Moreover, trimmed text between two tag elements is represented by a new abstract element denoted as <*TEXT*> element tag. Additionally, <TABLE> related tags with colspan or rowspan attributes are regulated. Due to the fact that matching parts of a query string are often highlighted in search results, and the corresponding style tags are often spread very inhomogeneously over a document, we additionally ignore HTML style tags. These transformations are done by the function $\sigma : T^* \rightarrow T$. Finally the pre-processed document is represented by its *restricted tag tree* T and the *plain tag sequence structure* S of T where each element in the tree representation has a link to the corresponding element in the sequence representation and vice versa.

3.3 Pattern Search Strategy

One common technique to measure the similarity between two plain sequences S_i, S_j with length n, m, respectively, is the *edit-distance* [9], which computes the minimal cost to transform one sequence into the other, utilizing uniform cost operations *insert*, *delete* and *rename*. Using Dynamic Programming techniques we can compute an $n \times m$ edit-distance matrix D in $\mathcal{O}(nm)$ time. A typical characteristic of data records is that single data record instances vary in optional or repetitive subparts. For instance, optional or multiple authors in the description of a book data record. An obvious disadvantage of the edit-distance computation is that repetitive and optional subparts inside the sequences S_i, S_j could increase the edit cost, therefore causing possible matches to be discarded. Optional subparts are usually handled with an approximate similarity threshold value θ by defining two sequences similar if their accumulated edit-distance is less or equal to a threshold value $\widetilde{D}^*(S_i, S_j) = \widetilde{D}_{n,m} \leq \theta$. This approach has already successfully been implemented in the MDR system. To overcome the problem of accumulated edit costs according to repetitive subparts it is suggested that two sequences are similar if they have at least a similarity of 60%. By virtue of the low similarity threshold it is very likely that sequences match each other although they belong to different contexts. Moreover, many situations could be generated where actually matching sequences are discarded because of repetitive subparts. For instance, when comparing the following data records S_i = <P><B>Title</B><A>Author1</A><A>Author2</A> <A>Author3</A></P> and S_j = <P><B>Title</B> <A>Author1</A></P> the repetitive authors make the sequences be considered as dissimilar despite of the low similarity threshold. Therefore we identify so called *tandem repeats* present in both sequences and allow zero cost for delete and insert operations inside additional repetitive instances.

3.4 Primitive Tandem Repeats

A *tandem repeat* contained in a sequence S is a subpart of the form α^k with $k \geq 2$ where α is a non-empty sequence of elements. For instance, the character sequences S_i=ABCBCBCDD and S_j=ABCDDDD contain the following tandem repeats $\mathcal{P}_i = \{\alpha_{i,1}^3 = \text{BC}, \alpha_{i,2}^2 = \text{CB}, \alpha_{i,3}^2 = \text{D}\}$ and $\mathcal{P}_j = \{\alpha_{j,1}^4 = \text{D}, \alpha_{j,2}^2 = \text{DD}\}$ and have the following repeats in common $\mathcal{P} = \mathcal{P}_i \cap \mathcal{P}_j = \{\alpha_1^{2,4} = \text{D}\}$. Consequently tandem repeats build an array of consecutive repeats. If we additionally claim that the repeats have to be *primitive* then α may not contain shorter repeats. As a result $\alpha_{j,2}^2$ in $\mathcal{P}_j$ becomes rejected. In the context of HTML tags we only consider primitive tandem repeats with $|\alpha| \geq 3$. For the running example we disregard this condition owing to space constraints. We implemented the algorithm described in [10] based on suffix trees to identify all z primitive tandem repeats in a sequence of length n in $\mathcal{O}(n+z)$ time before computing the edit-distance. Next we mark each extra repetitive instance with different marker elements. If, e.g., $\alpha_1^{k_i,k_j} \in \mathcal{P}$ and S_i contains fewer consecutive repeats of α_1 than S_j, then each element in the $m = k_j - k_i$ extra repeats of α_1 in S_j becomes marked. Additionally we mark the last element in the array of the consecutive repeated elements of α_1 contained in S_i with the same marker. Moreover, for each primitive tandem repeat $\alpha \in \mathcal{P}_l \setminus \mathcal{P}$ with $l \in \{i, j\}$, if α at most occurs once in the other sequence the elements become marked in the same way. According to these marked tag elements the recursive computation of a single matrix entry of D is adapted as follows:

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

for $1 \leq k \leq n$, $1 \leq l \leq m$ with

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

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

Consequently $\text{f}(a_k, b_l) = 0$ if the set of markers assigned to a_k and b_l have at least one marker in common. This assures that additional repetitive subparts contained in both sequences do not unduly increase the edit-distance. The cost function $c(a_k, b_l)$ equals zero if both elements belong to the same tag type which we denote as *shallow equal*.

marker						x	x	x	x		o
		S_j	A	B	C	B	C	B	C	D	D
	S_i	0	1	2	3	4	5	6	7	8	9
	A	1	**0**	1	2						
	B	2	1	**0**	1	2					
x	C	3	2	1	**0**	**0**	**0**	**0**	**0**	1	
	D	4		2	1	1	1	1	1	**0**	1
	D	5			2	2	2	2	2	1	**0**
o	D	6								2	**0**
o	D	7									**0**

The table shows the resulting edit-distance matrix of the two sequences and the different markers assigned to additional repetitive elements. With respect to the modified edit-distance computation the two sequences can be mapped with zero edit cost $D^*(S_i, S_j) = 0$ despite of optional repetitive sub-elements. For instance, the first mismatch in the diagonal occurs at position $(4, 4)$ in the matrix. Due to the fact that element C in sequence S_i shares the same marker as the second B in sequence S_j zero edit cost are mapped to matrix entry (3,4). To speed up the computation of the matrix $D_{n,m}$ we eliminate all path hypotheses exceeding the threshold value $\theta = 0.25 \cdot \max(|S_i|, |S_j|)$. In most cases this simple strategy reduces the computation workload to almost linear time and only best matching elements are compared with each other.

3.5 Identifying Data Regions and Records

According to the observations made in [13] we have to find similar subtrees T_{v_i} of child nodes $v_i \in \mathcal{C}_u$ of all inner nodes $u \in T$. This can be done by computing the pairwise similarity of each subtree T_{v_i}, T_{v_j}, respectively, their corresponding sequences S_{v_i}, S_{v_j} with $1 \leq i < j \leq |\mathcal{C}_u|$. Despite of comparing fix pairs of subtree combinations we generalize the search process to variable subtree combinations. Therefore single data records inside a data region could consists of variable numbers of subtrees.

When we compute the pairwise similarity between all subtree sequences we obtain an upper triangular subtree *similarity matrix* M_u^{s} for each inner node u. To simplify the pattern discovery we do not store the edit-distance values inside the matrix. Instead, a cell entry $M_u^{\mathrm{s}}(i, j)$ becomes 1 if the edit-distance between two sequences S_{v_i}, S_{v_j} satisfies the condition $D^*(S_{v_i}, S_{v_j}) \leq \theta_{v_{i,j}}$ with $\theta_{v_{i,j}} := 0.25 \cdot \max\left(|S_{v_i}|, |S_{v_j}|\right)$. Next we try to identify sets of adjacent sibling nodes having the highest matching frequency. The following example illustrates the problem:

Let u be an inner node in T and $v_1, \ldots, v_n \in \mathcal{C}_u$ its n children. The corresponding subtree similarity matrix M_u^{s} is given as follows:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	.
v_1	–	0	0	0	0	0	0	0	0	0	0	.
v_2		–	0	0	**1**	0	**1**	**1**	0	0	**1**	.
v_3			–	0	0	**1**	0	0	**1**	0	0	.
v_4				–	0	0	**1**	0	0	**1**	0	.
v_5					–	0	0	**1**	0	0	**1**	.
v_6						–	0	0	**1**	0	0	.
v_7							–	0	0	**1**	0	.
v_8								–	0	**1**	0	.
.									–	0	0	.

Interpreting the matrix entries we notice that the first node v_1 has no approximative match with its sibling nodes. Scanning the matrix from left to right and top-down, the first matching sequences are found at position $M_u^{\mathrm{s}}(2, 5) = 1$. Focusing on the diagonal entries starting at this position we recognize that several matches follow after node v_5. The fact that several consecutive sibling nodes are similar to each other indicates a potential repetitive matching region. Hereby we could weight the region by counting all consecutive matches found inside the diagonal, starting at the corresponding position. In our case the matching diagonal starts at position $M_u^{\mathrm{s}}(2, 5)$ and ends at position $M_u^{\mathrm{s}}(7, 10)$. Thus, the *diagonal weight* $w(\mathrm{diag}_{(2,5)}^{\mathrm{matches}}) = 6$. Next we have to split the data region into blocks forming the potential single data records. To this end, we scan the matrix horizontally starting at position $M_u^{\mathrm{s}}(2, 5)$ for nearest matches within the range defined by $w(\mathrm{diag}_{(2,5)}^{\mathrm{matches}}) = 6$. In our example the nearest match in row 2 occurs at position $M_u^{\mathrm{s}}(2, 7)$ with $w(\mathrm{diag}_{(2,7)}^{\mathrm{matches}}) = 1$, $M_u^{\mathrm{s}}(2, 8)$ with $w(\mathrm{diag}_{(2,8)}^{\mathrm{matches}}) = 4$ and the last match within the range is $M_u^{\mathrm{s}}(2, 11)$ with $w(\mathrm{diag}_{(2,11)}^{\mathrm{matches}}) = 1$. By comparing the diagonal weight starting at these positions we are able to measure the splitting possibility for the current data region, which is the likeliest for position $M_u^{\mathrm{s}}(2, 8)$ in our running example. If no match exists inside the range then the data record block divides the data region evenly.

Applying these steps iteratively to every pair of matching sequences we are able to identify potential data regions and their corresponding likeliest data block segmentations. In our example we obtain one main data region which is built by the nodes $v_2, .., v_{10}$ where every third node matches each other. Consequently we get a forest tree pattern where a single data record inside the data region consists of three nodes and their corresponding subtrees. Finally a data region $\mathcal{R}_k$ is represented by the data record, denoted as pattern S_{p_k}, with the minimal pairwise distance to each remaining data.

Knowing that the matrix is symmetric we only have to compute values for the upper triangular entries. To further reduce the computation of the matrix we consider an edit-distance dependent, transitive similarity relation. Hereby, the matrix is computed line-by-line, where each match with edit-distance $D(S_{v_i}, S_{v_j}) \leq 1$ is stored in an array A_i corresponding to the row i. After completing the computation of a row we set each pair $(l, k) \in 2^{A_i}$, which is part of the power set, to one, $M_u^{\mathrm{s}}(l, k) = 1$. Additionally, if $\big||S_{v_i}| - |S_{v_j}|\big| > \theta_{v_{i,j}}$ and in the case the sequences share no tandem repeats, we do not compute the edit-distance and set $M_u^{\mathrm{s}}(i, j) = 0$. Therefore we have to compute n matrix entries in the best case (subtree pattern) and $\frac{n^2}{2}$ matrix entries in the worst case (no transitive similarity).

3.6 Visual Data Segmentation

To enhance the data record separation process we incorporate visual cues which bears advantages for both vertical and horizontal segmentation of data region. When a data region $\mathcal{R}_k$ has been identified we additionally analyze the corresponding image representation defined by $\mathcal{R}_k$. Thus the bounding boxes of <TEXT> elements contained in $\mathcal{R}_k$ are used, to compute the vertical and horizontal projection profiles. This is realized by summing the width and respectively the height of all boxes with respect to their x/y-coordinates. Figure 1 shows the x/y-projection profiles of a data region. To demonstrate the correlation between text information as normally viewed on the Web site and bounding box information we represented the data region in both views. Additionally the projection profiles, exclusively computed according to the bounding box information, surround the data region in the figure. With respect to these pro-

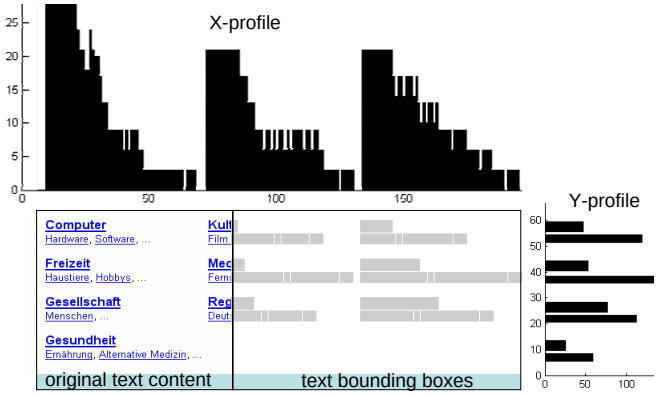

Figure 1: Visual data record segmentation utilizing x/y-profile information of text element bounding boxes.

files we analyze the probability of a potential segmentation according to characteristic valleys between peaks. Valleys between peaks corresponds to blank between text lines and the distance between two significant valleys corresponds to a potential separation of the data region into smaller data records. We establish a relationship between the valleys found in the profiles and the corresponding tag elements by a containment check. For this purpose each child element $v_i \in \mathcal{C}_u$, respectively the corresponding bounding box, in the data region $\mathcal{R}_k$ which is totally or partially contained in an interval, described by a valley, becomes a potential separation tag. Next we test the splitting probability with respect to the computed similarity matrix for these tag elements as described above.

3.7 Visual Data Region Weighting

After the pattern extraction process took place we compute the relevance of each pattern $S_{p_1}, S_{p_2}, ...S_{p_k}$ respectively their *result sets* $\mathcal{R}^{l_1}_{p_1}, \mathcal{R}^{l_2}_{p_2}, \ldots \mathcal{R}^{l_k}_{p_k}$ which contain all matching sub-sequences $\mathcal{R}^{l_i}_{p_i} := \{S_{p_i}, S_{p_i,1}, \ldots, S_{p_i,l_i}\}$ of a pattern S_{p_i} with $1 \leq i \leq k$ contained in the complete Web page. To accomplish this task several heuristics could be used for measuring the individual weight of a region. In case of dynamically generated HTML sites, we might reward data regions which partially contain the requested keywords according to some given user request. Moreover, it is possible to compute the textual coverage of the data region with respect to the pattern size. This heuristic often fails if the target pattern primarily consists of images and links or if a data region inside the menu bar has a higher textual coverage. Inspired by the work of Deng [3] who described a vision-based page segmentation, we introduce a ranking technique based on visual location. Hereby a significant feature for a potential target pattern is determined by its visual location inside the page. Generally, Web pages are divided into different information regions, so-called *slots*, filled with navigation bars, banners, adds and the proper query result. In most cases these slots reside in appropriate locations. Actually it can be observed that the slot filled with the target information has often a *centered* location and covers a large part of the total page. Consequently a heuristic which measures the coverage and the deviation of the result set $\mathcal{R}^{l_i}_{p_i}$ from the page center is used to weight pattern S_{p_i}. Given the bounding box information we ascertain the rectangle $\zeta_{\mathcal{R}^{l_i}_{p_i}}$ spanned by the result set $\mathcal{R}^{l_i}_{p_i}$ computing the extremum coordinates over all single rectangles represented by each match and define its total area $A^{\text{tot}}_{\mathcal{R}^{l_i}_{p_i}} := \sum_{p \in \mathcal{R}^{l_i}_{p_i}} A_p$ by the sum of each single rectangle area. Whereas the area of a matching instance $S_{p_{i,j}} \in \mathcal{R}^{l_i}_{p_i}$ is defined as the area of the rectangle spanned by the parent node, in case $S_{p_{i,j}}$ is a subtree pattern, or by the sum of the area of all sibling nodes if it is a forest tree pattern. Computing the total area as sum of each matching rectangle area, we could handle the case having a small matching instance in the upper part and a second instance at the bottom of the page which results in a rectangle spanning over the complete page but effectively occupying only a small area. Finally the weight $\text{w}_{S_{p_i}}$ of the pattern S_{p_i} is computed as follows:

$$\text{w}_{S_{p_i}} = \frac{1}{1 + \log(1 + \text{d}(\mathbf{c}_{\text{page}}, \mathbf{c}_{\mathcal{R}_{p_i}}))} A^{\text{tot}}_{\mathcal{R}^{l_i}_{p_i}},$$

where $\mathbf{c}_{\text{page}}$ denotes the page center coordinate, $\mathbf{c}_{\mathcal{R}_{p_i}}$ defines the center of the result set rectangle $\zeta_{\mathcal{R}^{l_i}_{p_i}}$, and $\text{d}(\mathbf{u}, \mathbf{v})$ is the Euclidian distance.

4. DATA ALIGNMENT

Having extracted the most relevant pattern and its corresponding approximate matches we try to align the multiple data records next. Hereby corresponding information elements (data items) should be arranged in adequate columns belonging to the same attribute, making it easy to label and store the data records in a database, export them as XML or synchronize them with data records extracted from other Web pages. Especially in case of *schema matching*, i.e., discovering semantically attributes in different schemas, which is fundamental for enabling query mediation and data exchange across multiple information sources, accurate data alignment plays an important role.

Typical properties of similar extracted data records are optional or multi-valued attributes or even several attributes encoded inside one single text element. These attributes often lead to non-trivial alignment scenarios. To motivate the complexity of multiple data record alignment we depicted a small example scenario in Figure 2a). Here three similar data records and their appropriate abstract tag sequences are represented. Please note that <TEXT> elements, which abstract from the original content information contained in the Web site, have been abbreviated by the letter T to enable a compact representation. Due to the fact that each sequence has optional sub-sequences, the alignment process becomes already difficult for such a small example. Before we start to explain our alignment technique we give a short introduction into existing multiple sequence alignment techniques and motivate our approach.

In the domain of bioinformatics *multiple (protein) sequence (genome) alignment* (MSA) is a fundamental task and also one among the most studied and difficult problems in computational biology. Although the notion of multiple alignment could be easily extended from two sequences to many sequences, the *score* or *quality* of a multiple alignment cannot be simply generalized. One very intuitive candidate is the well known *sum-of-pairs* (SP) score function which is given by the sum of the induced pairwise alignment scores of each pair in the alignment. An *optimal* solution of n se-

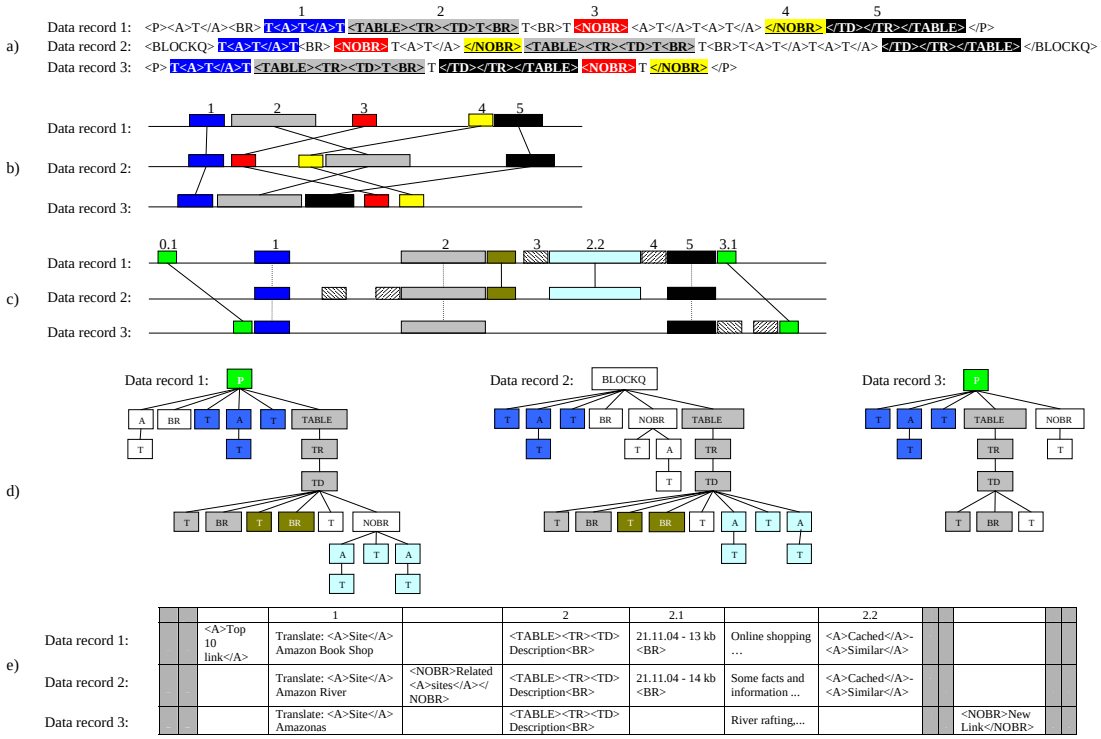

Figure 2: The data record alignment process is divided into different stages.

quences, each of length k, can be computed in $\Theta(k^n)$ time by dynamic programming [15]. However, such an approach is not practical for more than a few sequences. Moreover, the optimal SP alignment problem has been proven to be NP-complete [1]. It should be mentioned that in the domain of data record alignment the quantifier optimal is not always definite, making it difficult to express the quality of the alignment by an objective score function. To decrease the computational costs, a large body of research exists for the design of efficient heuristics with sub-optimal solutions. For instance, in [8] an alignment heuristic with guaranteed error bounds, less than twice the score of the optimal SP problem, and polynomial worst-case time have been proposed based on pairwise sequence alignment utilizing the edit-distance computation. The key idea in this approach is based on a *center-star tree* alignment algorithm which first tries to find a sequence (center sequence) minimizing the overall edit cost to each remaining sequence. According to this center sequence each remaining sequence is iteratively aligned to construct the final multiple alignment. Such an edit-distance based approach has already been ported into the information extraction domain, e.g., in [4, 5]. In [4] Chang et al. proposed IEPAD which generalizes the discovered exact repeated patterns to allow approximate matching by adopting the center-star multiple string alignment technique. Furthermore, in [5] Chang et al. proposed a semi-supervised information extraction tool called OLERA where the user could interactively generate extraction rules according to a set of training pages. OLERA is able to automatically discover other records similar to the enclosed examples and present the data in form of a spreadsheet by aligning the data records with the center-star alignment technique. Hereby, a string comparator function is incorporated into the alignment process to improve the alignment result. We implemented the second alignment proposal and identified several drawbacks of the center-star technique in the context of data record alignment. First, if the center sequence does not contain all optional sub-sequences, which is pretty often the case for complex data records, the multiple sequence alignment is fairly poor for these missing sub-sequences. Additionally, the alignment process lacks any global control instance to resolve alignment mismatches due to the fact that unaligned sequences are not considered when aligning new sequences to the growing multiple center sequence. Furthermore, the performance of the alignment process becomes very slow when the number of data records increases. And finally, tree structure and HTML structure information, which provides in our opinion valuable clues, should not be ignored during the alignment process.

Therefore we utilized a new method inspired by the algorithms in [6, 11, 7] based on global sequence alignment using *general suffix trees* as an alternative to edit-distance algorithms. The main idea is to find global matches which reduce the multiple alignment problem in a Divide and Conquer fashion. Thus, we are able to dramatically speed up the alignment process and extend it by a global control instance. Next we briefly describe the main stages.

4.1 Global Data Record Alignment

First we try to find *maximal unique matches* (*MUMs*) contained in **all** data records. Whereas maximal denotes that we cannot simultaneously extend the given match to the left or to the right in every sequence and unique means that the matches occur only once in each of the n sequences. For instance, in Figure 2a) five such like MUMs are contained in the sequences, respectively, MUM_1=T<A>T</A>T,

MUM_2=<TABLE><TR><TD>T
, MUM_3=<NOBR>, MUM_4=</NOBR>, and MUM_5=</TD></TR></TABLE>. To visualize the MUM arrangement we can represent the MUMs by intervals corresponding to the starting position and number of elements on horizontal lines. Figure 2b) shows the resulting 3-*level MUM diagram* with the corresponding *MUM chains*. The objective is to select a *MUM-sequence* of non-overlapping MUMs that has maximal weight, where the weight of a MUM-sequence is defined by the sum of the weights of its members, respectively, the number of elements they consist of. Solving this optimization problem could be done with algorithms from computational geometry in $\mathcal{O}(k^2)$ time, for details see [7], where k denotes the number of MUMs identified. When we align the data records according to the MUM-sequence of length l with optimal weight, the problem decomposes into $l+1$ smaller unaligned sub-regions. Figure 2c) shows the result of the computed optimal non-overlapping MUMs-sequence $\text{MUM}_{\text{seq}}^{\text{opt}} = [\text{MUM}_1, \text{MUM}_2, \text{MUM}_5]$. In the next stage we iteratively align each unaligned sub-region between global matches separately. To reflect the characteristics of optional and repetitive elements we attenuate our definition by searching for *MUMs*$^-$ appearing in **most of the sequences** and *maximal multiple exact matches (multiMEMs)* which may appear multiple times in a sequence. In Figure 2c) sub-region 0 contains one $\text{MUM}_{0.1}^-$=<P> which occurs in the first and the third sequence. The matching closing tags are located in sub-region 3 labeled 3.1. Moreover, sub-region 2 contains $\text{MUM}_{2.1}^-$=T
 and $\text{MUM}_{2.2}^-$=<A>T</A>T<A>T</A>, which do not intersect and therefore build an optimal non-overlapping MUMs-sequence. The algorithm to find all multiMEMs, respectively, MUMs takes $\mathcal{O}(zn+k)$ time utilizing a general suffix tree, where z is the length of the concatenated n sequences and k is the total number of multiMEMs and MUMs. We refer the interested reader to [9] for more details. Finally, remaining gaps are closed by mapping unaligned sub-sequences to the largest sub-sequence inside a sub-region with respect to their tree structure using a standard tree mapping function. Figure 2d) visualizes the tree structure of the data records. And Figure 2e) shows the result of the global alignment with the text content. In the end we assign to each group of aligned text, link and image elements a separate column. This course of action guarantees that we have a global control instance over the alignment process and we are also able to reduce the computational cost significantly.

4.2 Aligning Text Content

So far, we defined two elements shallow equal if they belong to the same tag type. Due to the fact that the content of <TEXT> elements establish a good indication for global correspondences we modify the comparison function of <TEXT> elements when we build the general suffix tree to find regularities contained in all sequences. Therefore we define two abstract <TEXT> elements A and B *content similar* if $f_{\text{sim}}(\sigma^{-1}(A), \sigma^{-1}(B)) > \theta_{\text{string}}$, i.e., if they are similar with respect to their original trimmed text content given a string comparator function f_{sim} and a threshold value θ_{string}. For instance when a new suffix $S_i = \alpha$<TEXT>β have to be inserted into the general suffix tree we have to compute the content similarity to each <TEXT> element leaving a node v. In case no content similarity is given a new edge has to be created, otherwise the suffix <TEXT>β is assigned to the edge having the highest content similarity value. Therefore each edge leaving a node v and starting with a <TEXT> element stands for an equivalence class of a text content where the representative is the first <TEXT> content which generated a new edge. This special treatment of abstract <TEXT> elements is necessary because during the alignment process we need to guarantee that only similar text content becomes aligned in the global matching stages. Best alignment results have been achieved with the Jaro-Winkler [17] string metric and $\theta_{\text{string}} = 0.7$.

5. EXPERIMENTS

In this section we compare our wrapper prototype system with the two most relevant existing fully-automatic state-of-the-art information extraction systems, ViNTs and MDR. We do not compare our tool with DEPTA because at the time of this writing neither the test bed, experiments were performed on, nor the system was available. Instead we took the accessible datasets 2 and 3 generated within the ViNTs prototype for testing and comparing. Furthermore, we conducted experiments on the manually labeled *Testbed for Information Extraction from Deep Web TBDW* [18] Ver. 1.02 available at (http://daisen.cc.kyushu-u.ac.jp/TBDW/) to additionally measure the performance of the alignment process. The performance was measured with the standard metrics recall $= \frac{E_{\text{correct}}}{N_{\text{total}}}$ and precision $= \frac{E_{\text{correct}}}{E_{\text{total}}}$. Where N_{total} defines the number of data records contained in all dynamic pages, E_{correct} is the total number of correctly extracted data records and E_{total} denotes the total number of data records extracted by the wrapper. Due to the fact that the number of data records represented inside different Web pages varies from just a few up to hundreds of records, the final performance metrics are basically dominated by pages which consists of many data records. Therefore only the first 25 data records are used.

5.1 Experimental Results

To test the performance of our system ViPER, we used dataset 2 from the ViNTs Web page featuring sample pages from 100 different search engines with 10 (5 test/5 training) result pages and one additional no-result page per engine. Due to the fact that our wrapper only works on a single Web page we randomly took 1 result page per search engine resulting in 100 pages and operated on these pages independently. As we can see from Table 1, the performance of our system on dataset 2 tends to result in high quality precision and recall values. We use the abbreviation *SRRs search result records* as introduced in [14], which is equivalent to our definition of data records to conform with the different notions. The ViNTs results, reported in [14], were obtained during building the wrapper on 5 sample pages per search engine using one no-result page and finally extracting SRRs according to these 5 pages. Despite of the different extracting approaches, DOM tree-based vs. identifying visual separators, both techniques yield high quality results. The reason is that data records in document result pages could be easily separated by horizontal separators and the relevant data records often cover a large slice of the result page.

Next, we compare our system according to dataset 3 from the ViNTs system with the results published in [14] obtained by using MDR and ViNTs. The authors of [14] took this dataset to compare their ViNTs system with the available

	Datasets used within the ViNTs system					Testbed for IE	
	DATASET 2 [A]		DATASET 3 [B]			TBDW Ver. 1.02 [C]	
#search engines	100		42			51	
#pages per search engine	5	1	1			1	
asterisk marks results reported in [14]	ViNTs*	ViPER	ViNTs*	MDR*	ViPER	ViNTs	ViPER
#SRRs	6905	1390	795			693	
#Extracted SRRs	6872	1419	795	479	790	661	686
#Correct SRRs	6740	1378	785	420	779	618	676
Recall	97.6%	99.1%	98.7%	52.8%	98.0%	89.2%	97.6%
Precision	98.1%	97.1%	98.7%	87.7%	98.6%	93.5%	98.5%

Table 1: Extraction performance achieved on 3 different third-party test beds.

MDR tool, calibrating the similarity threshold value at 60%. While MDR reports all identified data regions, only the major region is considered if it is contained inside the reported data regions. In the middle of Table 1 the reported extraction results of the two different extraction tools ViNTs and MDR are provided, along with the results of our system ViPER. First of all it could be recognized that the performance of ViNTs and ViPER is considerably better than that of MDR on this dataset. This indicates that despite of the same underlying technique, used for MDR, the improvements made by incorporating tandem repeats, visual segmentation, and visual relevance selection the final result of the extraction could be enhanced enormously. Comparing the performance of ViNTs with ViPER's, ViNTs performs slightly better than our system. One reason is that in [14] only the number of SRRs contained in a consecutive data region are count as correct result. If, e.g., the relevant data region is split into 2 subparts, by an advertisement for instance, ViNTs only reports and counts results from the larger part as correct. This is for example the case for the result page lycos.html. Here our system additionally reported the 4 data records which we had to count as false records. Finally, we performed extraction tests on data set TBDW Ver. 1.02 which have not been used in [14]. Here our system performed better than ViNTs, due to the fact that some data records have a more complex structure. Additionally, we tested the alignment quality with respect to the manually labeled field information. Where label information is provided in the data set for the first data record of each Web page. The number of data items contained in the set of labeled data records add up to 367, resulting in 4,846 data items overall. With respect to the 676 correctly extracted data records by ViPER only 11 data items could not be aligned correctly. Hereby the execution time for the alignment was always less than 150msec on a P4 2.4GHz with 768MB RAM for every page.

6. CONCLUSION

In this paper, we presented a fully-automatic information extraction tool called ViPER. The tool is able to extract and separate data exhibiting recurring structures out of a single Web page with high accuracy by identifying tandem repeats and using visual context information. Additionally, we proposed a new fast and robust data record alignment technique based on global matching information and text content information. Tests performed on several third-party data sets finally underpin the high accuracy of the our system. In the future we plan to release a plugin of ViPER for the Firefox Web browser with additional interactive functionalities to extract non-repetitive data.

7. REFERENCES

[1] P. Bonizzoni and G. D. Vedova. The complexity of multiple sequence alignment with SP-score that is a metric. *TCS*, 259(1-2):63–79, 2001.

[2] D. Buttler, L. Liu, and C. Pu. A Fully Automated Object Extraction System for the World Wide Web. In *ICDCS'01*, page 361. IEEE Computer Society, 2001.

[3] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Extracting content structure for web pages based on visual representation. In *APWEB'03*, pages 406–417, 2003.

[4] C.-H. Chang, C.-N. Hsu, and S.-C. Lui. Automatic information extraction from semi-structured web pages by pattern discovery. *SCI expanded*, 35(1):129–147, 2003. Special Issue on Web Retrieval and Mining.

[5] 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.

[6] A. L. Delcher, S. Kasif, R. D. Fleischmann, J. Peterson, O. White, and S. L. Salzberg. Alignment of whole genomes. *NAR*, 27(11):2369–2376, 1999.

[7] J. Deogun, J. Yang, and F. Ma. EMAGEN: An efficient approach to multiple whole genome alignment. In Y.-P. P. Chen, editor, *APBC'04*, volume 29 of *CRPIT*, pages 113–122, Dunedin, New Zealand, 2004. ACS.

[8] D. Gusfield. Efficient methods for multiple sequence alignments with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55:141–154, 1993.

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

[10] D. Gusfield and J. Stoye. Linear-Time algorithms for finding and representing all tandem repeats in a string. *JCSS*, 69:525–546, 2004.

[11] M. Höhl, S. Kurtz, and E. Ohlebusch. Efficient multiple genome alignment. *Bioinformatics*, 18(Suppl. 1):S312–S320, 2002.

[12] A. H. F. Laender, B. Ribeiro-Neto, A. S. D. Silva, and J. S. Teixeira. A brief survey of web data extraction tools. *ACM SIGMOD Record*, 31(2):84–93, 2002.

[13] B. Liu, R. Grossman, and Y. Zhai. Mining data records in web pages. In *SIGKDD'03*, 2003.

[14] W. Meng, V. Raghavan, and C. Yu. Fully automatic wrapper generation for search engines. In *WWW14*, May 2005.

[15] D. Sankoff. Minimal mutation trees of sequences. *SIAM*, 28:35–42, 1975.

[16] J. Wang and F. H. Lochovsky. Data extraction and label assignment for web databases. In *WWW12*, pages 187–196. ACM Press, 2003.

[17] W. E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 1999.

[18] Y. Yamada, N. Craswell, T. Nakatoh, and S. Hirokawa. Testbed for information extraction from deep web. In *WWW Alt. '04*, pages 346–347, New York, NY, USA, 2004. ACM Press.

[19] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *WWW14*, May 2005.