

## A Tool for Computing the Visual Similarity of Web Pages

María Alpuente and Daniel Romero  
 DSIC-ELP, Universidad Politécnica de Valencia  
 Camino de Vera s/n, 46022 Valencia, Spain  
 {alpuente,dromero}@dsic.upv.es

**Abstract**—Recently, we proposed a functional technique for identifying similar Web pages that is based on measuring tree similarity. The key idea behind the method is to transform each Web page into a compressed, normalized tree that effectively represents its visual structure. In this work, we develop an optimization of this technique that is based on memoization and that achieves significant improvements in efficiency in both time and space. This work also presents a tool that implements the proposed technique as well as two case studies for two real scenarios. Experiments on real documents show that the optimized algorithm performs significantly better than the original technique and demonstrate the practicality of our approach.

**Keywords**—Web page comparison; visual similarity; tree edit distance; Web document clustering.

### I. INTRODUCTION

The idea of information visualization is to gain insights from great amounts of abstract data. Comparing document sets that are found by searching the World Wide Web is a new challenge for visualization-related techniques. Typical scenarios include automatically identifying documents that are *visually* similar to ones previously viewed or fixed by the user. Also, considering visual similarities can sometimes improve Web page clustering quality [1].

Apart from sheer volume, one particular vexing problem of information retrieval (IR) and related techniques such as filtering or categorization is that they mainly deal with text. However, the possibility to recognize heterogeneous depictions of pieces of information that have similar semantics is a common situation in the Web that should be taken into account. Contemporary research has begun to supplement basic IR approaches with techniques that collect indicators of “information value” and are able to assess semantic similarities [2]. Unfortunately, one factor that is hardly considered by state-of-the-art Web handling techniques is whether two different pieces of code can express the same visual sensation. Actually, when we look at a Web page, we are not aware of the underlying HTML code, and we are only able to distinguish the visual structure given by the groupings, columns, rows, and data (see Figure 1). This has led us to define the “visual structure of a Web

page” as the apparent structure of the Web page that is perceived by a human independently of the source code that produces it. The ability to assess visual similarities in an accurate, automated, and scalable way can be a key determinant of the effectiveness of information handling and decision support software that deals with the WWW. For example, in [3], several methods are proposed to determine the structure of a given Web page. A ranked list of possible structures for the page is produced, which can be used for several important applications. For instance, it can be used to define suitable wrappers<sup>1</sup> for Web-based information integration systems. As reported in [3], by adding some visual requirements regarding to the Web page that we are interested in exploring, the fitness of the proposed top-ranked structures is meaningfully improved with respect to an earlier ranking that lacks this extra information. As a second example, in the scenario of Web security, the visual similarity of Web pages can also be applied to phishing<sup>2</sup> detection [4], [5], [6].

In [7], a novel application of the techniques for recognizing the visual information within Web pages is presented. There, Web pages are considered to comprehend not only the information but also social aspects where there is a hidden interaction among users, developers, and Web site owners. The paper then establishes that the visual perception is usually independent from the user’s ability (e.g., it is expected that a sales page can be recognized by an Arab-speaking user, a Chinese-speaking user, or an English-speaking user). Thus, everybody should be able to use the functionalities of the Web page. Moreover, the visual information that can be extracted from the pages that users visit could be productively used to determine the social groups of the users themselves. In this manner, this information could be used to improve the accuracy of search engines.

The above-mentioned applications have motivated us to investigate the problem of Web page visual similarity. To the best of our knowledge, the recognition of the visual structural information is an area that is still unexplored by the scientific community, and only few works have addressed

<sup>1</sup>A wrapper is a special-purpose program that extracts information from Web pages written in a specific format.

<sup>2</sup>Phishing is the criminally fraudulent process of attempting to acquire sensitive information such as usernames, passwords, and credit card details by masquerading as a trustworthy entity in an electronic communication.

The image shows a screenshot of a web directory page from the University of Zaragoza. The page is titled 'Directorio' and contains a table of staff members. Annotations with arrows point to various parts of the page: 'columns' points to the vertical layout of the table; 'rows' points to the horizontal layout of the table; 'data' points to the individual entries in the table; and 'grouping' points to the red dashed boxes that categorize the data into columns like 'Profesores', 'Administración', 'Becarios y Colaboradores', 'Consejería', and 'Técnicos'.

Profesores		Administración		Becarios y Colaboradores		Consejería		Técnicos	
Nombre	E-mail	Nombre	E-mail	Nombre	E-mail	Nombre	E-mail	Nombre	E-mail
Fco. José	ffcoj@dsic.upv.es	Abad Cerda	abad@dsic.upv.es	1038				73574	
Silvia Mara	silmar@dsic.upv.es	Abraham Gonzales	abraham@dsic.upv.es	3085				83510	
Jorge Luis	jagorlu@dsic.upv.es	Agüero Medina	aguero@dsic.upv.es	2128				83528	
Vicente	vicente@dsic.upv.es	Albaso Gonzalez	albaso@dsic.upv.es	1105				83531	
Seabitz	seabitz@dsic.upv.es	Alarcón Jiménez	alarcon@dsic.upv.es	2102				73537	
Mauricio Fernando	mferna@dsic.upv.es	Alba Castro	alba@dsic.upv.es	2101				83529	
Juan Miguel	jamiguel@dsic.upv.es	Alberola Oltra	alberola@dsic.upv.es	2105				83527	
Manuela	manuela@dsic.upv.es	Albert Abad	albert@dsic.upv.es	2213				83511	
Josep	josemar@dsic.upv.es	Alemany Mollá	alemany@dsic.upv.es	F4D9				28592	
Antonio	antoni@dsic.upv.es	Alfani Navarro	alfani@dsic.upv.es	ADM1				73503	
Shella	shella@dsic.upv.es	Allaga Varea	allaga@dsic.upv.es	1106				73533	
Pedro	pedro@dsic.upv.es	Alonso Jordá	alonso@dsic.upv.es	2094				83507	
José Miguel	josemiguel@dsic.upv.es	Alonso Abalos	alonso@dsic.upv.es	1D92				73562	
Maria	mpuente@dsic.upv.es	Alpuente Frasedo	alpuente@dsic.upv.es	2D40/2D40				79726/79354	
Alvaro	alvaro@dsic.upv.es	Alvarez Rodríguez	alvarez@dsic.upv.es	3007				83501	
Antonio Félix	antonio@dsic.upv.es	Alvarez Rodríguez	alvarez@dsic.upv.es	C2-1				73554	
Fernando	fernando@dsic.upv.es	Alvarez Bermejo	alvarez@dsic.upv.es	1D93				83518	
Jesus	jesus@dsic.upv.es	Andrés Ferrer	andres@dsic.upv.es	1101				83533	
Nathalie	nathalie@dsic.upv.es	Aguino Salvioni	aguino@dsic.upv.es	1104				83534	
Ana Isabel	ana@dsic.upv.es	Araci Payá	araci@dsic.upv.es	-					

Figure 1. Example of the human perception of the visual structure of a Web page

this aspect. A description of the few existing techniques and a comparison with our approach can be found in [8].

In [8], we developed a preliminary technique for Web page comparison that takes into account their visual structure. The key idea behind the method is to transform each Web page into a compressed, *normalized* tree that effectively represents its visual structure. The transformation is based on a classification of the set of html-tags that is guided by the visual effect of each tag in the entire structure of the page. Then, a metric to compute the distance between two Web pages that is based on the classical *tree edit distance* algorithm [9] was formalized. This algorithm defines the dissimilarity of two trees by determining the minimal cost of the edit operations (node insertion, node deletion, and label change) that are needed to transform one tree into the other. A preliminary implementation that is written in Maude [10] was also presented in [8]. Unfortunately, the vast (fixpoint) computation that is involved in the Web comparison algorithm of [8] leads to unsatisfactory performance, so the prototypical tool can only efficiently process very small examples.

In order to reduce the time and space complexity of the tree edit distance algorithm, different optimizations have been developed. These are based on computing only a subset of a large dynamic programming table (referred to as *table of subproblems* [11], [12]), by using memoization. In this paper, we present a powerful optimization of our previous Web comparison algorithm [8] that is based on similar memoization techniques and allows us to compare documents that are extracted from real document collections and databases. We have also reimplemented the former Web comparison tool in Maude in order to improve both the functionality and the performance of the previous version. Finally, we report two experiments in realistic scenarios. The first case study considers the Web pages of the workshops WWV'07 and WWV'08, which have a similar visual

structure even though their respective HTML encodings are plainly different. The tool computes a similarity measure of 92% between the two Web sites in a very short time. The second case study is a Web document clustering. The aim was to identify a set of Web page templates within a large collection of Web pages. The study identified 11 possible representatives in a collection of 470 Web pages.

The paper is organized as follows. In Section II, we introduce some preliminary notions, including Web page descriptions. In Section III, we briefly recall the original technique for the Web page comparison given in [8]. Section IV discusses popular optimizations of the tree edit distance algorithm based on memoization. Section V describes the main features of the tool. In Section VI, we present an experimental evaluation of our tool in two real scenarios. Finally, Section VII presents our conclusions.

## II. PRELIMINARIES

In this section, we briefly recall the essential notions and terminology used in this paper. We refer to a finite set of symbols as *alphabet*.  $\Sigma$  denotes a set of *function symbols* (also called *operators*), or *signature*. We consider varyadic signatures as in [13] (i.e., signatures in which symbols do not have a fixed arity).

Terms are viewed as labelled trees in the usual way.  $\tau(\Sigma)$  denote the *term algebra* built on  $\Sigma$ . Positions are represented by sequences of natural numbers denoting an access path in a term. Syntactic equality between objects is represented by  $\equiv$ . We also need  $\max(x, y)$  and  $\lceil n \rceil$ ,  $\max(x, y) = x$  if  $(x \geq y)$ , else  $y$ ; and  $\lceil n \rceil$  is the least integer number greater than or equal to  $n$ .

In this work, a *Web page* is either an XML [14] or an XHTML [15] document, which we assume to be well-formed<sup>3</sup>. We only consider the tags that can occur inside the tag `body`, since these are the tags that influence the visual structure of a web page. Let us consider two alphabets  $T$  and  $Tag$ . We denote the set  $T^*$  by *Text*. An object  $t \in Tag$  is called *tag element*, while an element  $w \in Text$  is called *text element*. Since Web pages are provided with a tree-like structure, they can be straightforwardly translated into ordinary terms of the term algebra  $\tau(Text \cup Tag)$ . Note that XML/XHTML tag attributes can be considered as common tagged elements and can thus be translated in the same way.

## III. A VISUAL TECHNIQUE FOR WEB PAGE COMPARISON

Let us briefly recall the top-down technique for visual comparison of Web pages given in [8]. First, a transformation of Web pages is introduced. It translates each Web page into a tree that effectively represents its visual structure. Then, by means of a compression process, we obtain a visual representative for the computed tree. Finally, a measure of

<sup>3</sup>Since there are plenty of programs and online services that can be used to validate XHTML/XML syntax and perform links checking (e.g. [16], [17]).

Visual (abstract) tag $\alpha(f)$	HTML tag $f$
<i>grp</i>	table, ul, html, body, tbody, div, p
<i>row</i>	tr, li, h1, h2, hr
<i>col</i>	td
<i>text</i>	otherwise

Table I  
CLASSIFICATION OF HTML TAGS

similarity between two Web pages that is based on the classical *tree edit distance* algorithm is given.

### A. Translation

Our translation technique is based on a classification of the set of html-tags, which is guided by the visual effect of each tag on the whole structure of the page, that allows us to infer the visual structure of the page from the HTML tags in it.

Let  $\Sigma_V = \{grp, col, row, text\}$  be a signature of abstract (visual) HTML tags, where *grp*, *col*, *row*, and *text* can be seen as the “abstraction” of a number of different concrete HTML tags according to the abstraction classification function  $\alpha$  given in Table I. It is straightforward to translate the Web pages into ordinary terms of the term algebra  $\tau(\Sigma_V)^4$ . This translation brings to light the repetitive structures that occur on the page and gets rid of those chains of tags that do not influence the visual aspect of the page.

### B. Canonical representation

In order to avoid losing information after compressing a Web page, we need to record the approximate number of nodes before applying the compression. Let us introduce the notion of marked term  $[\mathbb{N}]t$  given in [8], which allows us to record the number  $\mathbb{N}$  of “similar” subterms  $t$  that appear in a given term. We denote the corresponding term algebra by  $\tau([\mathbb{N}]\Sigma_V)$ . For instance, consider the term in Figure 2(a). The corresponding marked term is shown in Figure 2(b). The subterm “[2]row([1]text)” represents that the term *row*([1]text) appears twice. Note that this representation is not commutative, thus (in Figure 2(a)) the first subterm *row*(text) cannot be packed together with the last two. When no confusion can arise, we simply write  $[1]grp([2]row([1]text)) = grp([2]row(text))$ .

<sup>4</sup>Obviously, the list of tags is limited and is only taken as a “proof of concept” that could eventually be enlarged with new ones.

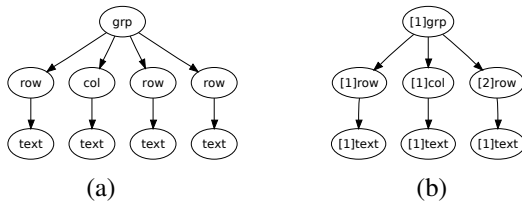


Figure 2. Example of marked algebra

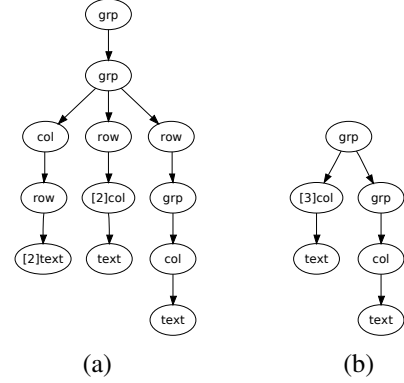


Figure 3. Irreducible term

We say that two marked terms are comparable if the corresponding (unmarked) terms are comparable. In symbols

$$[n_1]f(t_1, \dots, t_n) \equiv_{\Sigma_V} [n_2]g(v_1, \dots, v_n) \\ \text{iff } \alpha(f) \equiv_{\Sigma_V} \alpha(g) \text{ and } t_i \equiv_{\Sigma_V} v_i, 1 \leq i \leq n$$

Note that the equivalence between function symbols  $f$  and  $g$  is determined by whether or not they can be abstracted by the same visual tag (see Section III-A).

### C. Web page compression

In order to reduce the complexity of tree comparison, two compression techniques that dramatically reduce the size of the terms were proposed in [8]. Horizontal compression (*hrz*) packs together those subterms that represent repetitive structures. Vertical compression (*vert*) shrinks those chains of tags that do not visually influence the result.

The auxiliary function  $join(t_1, \dots, t_n)$  sum the number of roots and approximates the number of leaves. For example, for terms  $t_1 = [3]grp([4]row[8]col,)$  and  $t_2 = [5]grp([2]row, [6]row)$ , we have  $join(t_1, t_2) = [8]grp([3]row, [7]col)$ . A detailed description of the function  $join$  can be founded in Section 4.2 in [8].

**Definition 3.1: (horizontal compression)** Let  $t = f(t_1, \dots, t_n) \in \tau([\mathbb{N}]\Sigma_V)$  be a term (Web page). The horizontal compression function  $hrz :: \tau([\mathbb{N}]\Sigma_V) \rightarrow \tau([\mathbb{N}]\Sigma_V)$  on marked terms is defined by:

$$hrz(t) = \begin{cases} t & \text{if } n = 0 & (1) \\ hrz(f(t_1, \dots, t_{i-1}, s, t_{j+1}, \dots, t_n)) & (2) \\ \quad \text{where } s = join(t_i, \dots, t_j) \\ \text{if } ((1 \leq i \leq j \leq n) \text{ and} \\ \quad (t_i \equiv_{\Sigma_V} t_{i+1} \dots t_{j-1} \equiv_{\Sigma_V} t_j)) \\ f(hrz(t_1), \dots, hrz(t_n)) & \text{otherwise} & (3) \end{cases}$$

Roughly speaking, Definition 3.1 states that all the arguments that are equal w.r.t.  $\equiv_{\Sigma_V}$  and occur at the same level  $i$  are packed together. Then, compression recursively proceeds to level  $(i + 1)$ .

XML/XHTML is a markup language for documents containing semi-structured information. In XML/XHTML, all elements must be properly nested within each other as follows  $\langle b \rangle \langle i \rangle$ This text is bold and italic $\langle /i \rangle \langle /b \rangle$ . Considering the translation given in Section III-A, this structured information favours the formation of chains of tags that do not influence the overall structure of the page. In the following, we describe how to shrink the chains of tags while preserving the overall structure.

First of all, let us establish the following: A term  $t$  obey the *safe vertical condition* if (i) it is not marked; (ii) it has only one child; (iii) it preserves the structure of the page ( $grp$  has a higher status than other elements); (iv) it preserves the information in the page ( $text$  is not compressed).

Now, we are ready to formalize the vertical compression transformation. Roughly speaking, this is done by moving the inner more influential nodes as high as possible in the tree, while preserving the outer structure of the term whenever possible.

**Definition 3.2: (vertical compression)** Let  $t = [r]f(t_1, \dots, t_n) \in \tau([\mathbb{N}]\Sigma_V)$  be a term that satisfies the condition for safe vertical compression. The vertical compression function  $vrt :: \tau([\mathbb{N}]\Sigma_V) \rightarrow \tau([\mathbb{N}]\Sigma_V)$  is defined by:

$$vrt(t) = \begin{cases} t & \text{if } n = 0 & (1) \\ vrt(shr(t)) & \text{if } t \text{ obeys the safe vertical condition} & (2) \\ [r]f(vrt(t_1), \dots, vrt(t_n)) & \text{otherwise} & (3) \end{cases}$$

where the *shrinking* function  $shr :: \tau([\mathbb{N}]\Sigma_V) \rightarrow \tau([\mathbb{N}]\Sigma_V)$  is defined by:

$$shr([r]f([m]g(t_1, \dots, t_n))) = \begin{cases} [r]f(t_1, \dots, t_n) & \text{if } m = 1 \wedge g \neq grp & (1) \\ [m]g(t_1, \dots, t_n) & \text{otherwise} & (2) \end{cases}$$

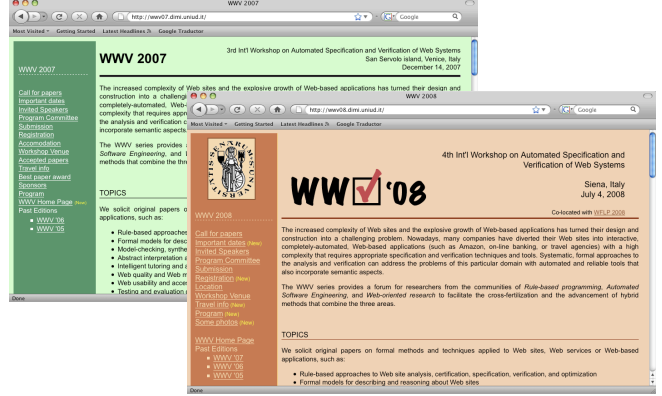
Then, the overall compression of a term is given by the following operator.

**Definition 3.3: (global compression)** Let  $f \in \tau([\mathbb{N}]\Sigma_V)$  be a Web page. The *compression* function is defined by:

$$\begin{aligned} compress &:: \tau([\mathbb{N}]\Sigma_V) \rightarrow \tau([\mathbb{N}]\Sigma_V) \\ compress(f) &= hrz(vrt(f)) \end{aligned}$$

In other words, we first remove the tags that belong to a chain of tags that does not influence the aspect of the resulting page and then we join the subterms. Given a term  $t$ , if the operator in Definition 3.3 is applied repeatedly we obtain a compressed *irreducible term*  $t_{zip}$ . This is because both the vertical and horizontal transformations are confluent and terminating.

**Example 3.4:** Consider the term in Figure 3(a). By applying repeatedly the operator in Definition 3.3, we get the *irreducible term* in Figure 3(b).



(a) (b)

Figure 4. Web pages of workshops WWV'07 and WWV'08

### D. A similarity measure between terms

In our framework, the problem of comparing Web pages essentially boils down to comparing trees. In the literature, the most widely used approach for comparing trees consists in computing the “edit distance” between the two rooted ordered trees, i.e., the minimum cost sequence of edit operations (node insertion, node deletion and label change) that transforms one tree into another [11].

The tree edit distance algorithm was introduced by Tai in the late 1970’s [18] as a generalization of the well-known string edit distance algorithm [19]. This algorithm states that if  $F$  and  $G$  are two rooted trees with a left-to-right order among siblings where each vertex is assigned a label from an alphabet  $\Sigma$ , the edit distance between  $F$  and  $G$  is the minimum cost of transforming  $F$  into  $G$ . This is done by a sequence of elementary operations consisting of deleting and relabeling existing nodes, as well as inserting new nodes (allowing at most one operation to be performed on each node at each step).

The *tree edit distance* algorithm assumes that there exists a *cost function* defined on each *edit operation*. In our technique, we state that the cost of an *edit operation* between two marked terms is given by the largest number of repetitions within the terms. The *edit distance* function between two trees (Web pages)  $t_1$  and  $t_2$  is denoted as  $\delta(t_1, t_2)$ .

**Definition 3.5: (Web pages similarity)** Let  $t, s \in \tau([\mathbb{N}]\Sigma_V)$  be two Web pages. Let  $t_{zip}$  and  $s_{zip}$  be two irreducible visual representatives of  $t$  and  $s$ , respectively. The comparison between  $t$  and  $s$  is defined by:

$$\begin{aligned} cmp &:: \tau([\mathbb{N}]\Sigma_V) \times \tau([\mathbb{N}]\Sigma_V) \rightarrow [0..1] \\ cmp(t, s) &= 1 - \frac{\delta(t_{zip}, s_{zip})}{|t_{zip}| + |s_{zip}|} \end{aligned}$$

According to Definition 3.5, the similarity of two pages is quantitatively expressed as a real number between 0 and 1.

#### IV. OPTIMIZATION OF THE COMPARISON TECHNIQUE BY USING MEMOIZATION

Memoization is a popular optimization technique that can be seen as a cross between simply performing a calculation and looking up the result in a table.

In the worst case, the (time and space) complexity of the recursive tree edit distance algorithm is  $O(n^2m^2)$ , with  $n$  and  $m$  being the size of the compared forests. A memoizing-based optimization of the tree edit distance algorithm is achievable by using a global  $n * m$  table  $\Delta$  of subproblems with values  $\Delta_{vw} = \delta(F_v, G_w)$ , for  $v \in F$  and  $w \in G$ . At each recursive invocation, the optimized algorithm looks up and updates this table  $\Delta$ , with the table populated transparently on an *as-needed* basis, so that the program does not waste any time performing calculations ahead of time to construct the table. By this optimization, the space complexity is reduced to  $O(nm)$  (see [11] and [12] for an in depth discussion on time and space of the tree edit distance algorithm; particularly [11] provides an account of different optimizations using memoization).

The table  $\Delta$  of subproblems has been implemented in our tool by means of the function  $\text{Map}\{x,y\}$  of Maude [10].  $\text{Map}$  represents a function  $f$  on a finite set  $A = \{a_i\}_{i=1}^n$  as the set of pairs  $\{(a_1, f(a_1)), \dots, (a_n, f(a_n))\}$  that represent the graph of the function  $f$ ; each pair  $(a_i, f(a_i))$  is called an *entry*. Each entry in the table  $\Delta$  is given by a key which corresponds to the position of each of the subtrees calculated, together with the result of running the tree edit distance algorithm on it. More formally, let  $p_v$  and  $p_w$  be the positions of  $v$  in  $F$  and  $w$  in  $G$  respectively. Then,  $\Delta_{vw}$  is defined as:  $\text{Map}\{p_v \bullet p_w, \delta(F_v, G_w)\}$ , where  $\bullet$  stands for a tuple constructor.

By memoizing calls in our recursive edit distance algorithm for marked terms, we achieve a huge improvement in both time and space, which we discuss in the Section VI.

#### V. IMPLEMENTATION

The optimized comparison technique presented above has been implemented in Maude [10], with a new interface developed in Java. Memoization is essential to this implementation in order to prevent duplicate calculation. The experimental system is publicly available at <http://www.dsic.upv.es/~dromero/cmp.html>. The Maude programming language, which implements the semantic as well as the logical framework of rewriting logic, provides a formal analysis infrastructure (such as state-space breadth-first search) with competitive performance.

The main features of our tool are:

- The implementation consists of approximately 660 lines of source code written in Maude. This includes the processing modules for trees and lists taken from [10].
- The online parser for semistructured expressions (i.e., XML/XHTML documents) is written in Java. A Java

#### Web Pages Comparison

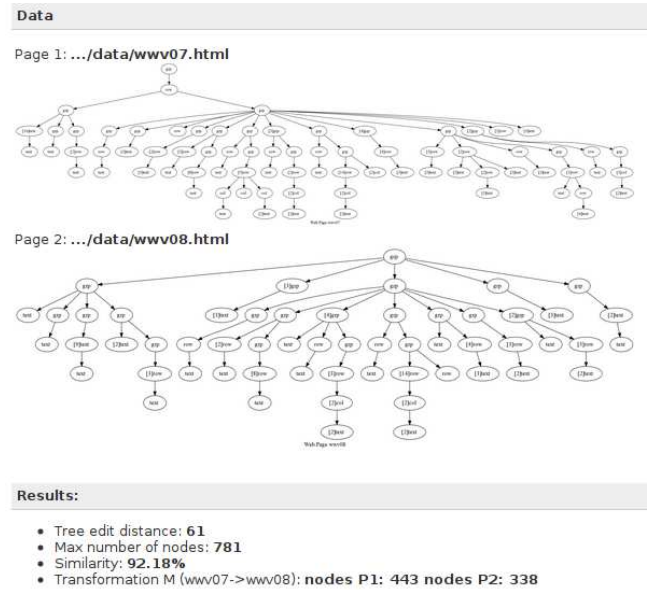


Figure 5. Output screen of the tool

class provides a single access point that hides the technical detail to every possible user.

- The tool includes a *config file* to configure the initial settings for some parameters, e.g., the location of the Web pages and the output folder.
- This version includes the memoization-based optimization, which makes the tool more efficient in time and space than the previous version.

Figure 5 shows the Web page output by the tool when fed with the input pages shown in Figure 4. Two parts are presented, *Data* and *Results*. In *Data*, two trees show the structure (irreducible term) of each page entry. In *Results*, the results of the comparison are shown.

#### VI. EXPERIMENTAL EVALUATION

We have checked that the developed system works very satisfactorily on a number of experiments, including all the examples in this paper.

In order to test the performance of our technique for visual comparison of Web pages, in this section we report on some of the experiments that we conducted with our tool. The results shown below were obtained on a personal computer equipped with 1 GB of RAM memory, 120 GB hard disk and an Intel Core 2 Duo at 2.2 GHz running Mac OS X.

##### A. Case study

Let us begin with the comparison of the Web sites of the workshops WWV'07 (<http://www07.dimi.uniud.it/>) and WWV'08 (<http://www08.dimi.uniud.it/>), whose homepages are illustrated in Figure 4(a) and Figure 4(b),

respectively. These two pages have a similar appearance but the HTML code is quite different. For instance, the Webpage of Figure 4(a) is defined using tables, whereas Figure 4(b) uses divisions (tag `<div>`).

Let  $t, s \in \tau(\mathbb{N} \Sigma_{\mathbb{V}})$  be two terms such that  $t$  corresponds to the *WWW'07* Web page and  $s$  corresponds to the *WWW'08* Web page. By using the Web page comparison measure (given in Definition 3.5), we obtain

$$\begin{aligned} |t_{zip}| = 443 \text{ and } |s_{zip}| = 338 & \quad \delta(t_{zip}, s_{zip}) = 61 \\ cmp(t, s) = 0.9218 & \sim \end{aligned}$$

We say that the similarity between  $t$  and  $s$  is 92%. The execution time took  $13251ms$ , and 21695 calculations of subproblems (entries in table  $\Delta$ ) were made. This is, more than 1600 values  $\Delta_{v,w}$  per second, which witnesses the practicability of our approach. It is important to note that this example does not run with the former implementation of our tool.

### B. Web document clustering

Our second case study is a *Web document clustering* experiment. Web document clustering is a specialized classification technique that categorizes into clusters a collection of Web documents according to a measure of similarity. The aim of the experiment was to identify a set of Web page templates that could be considered as the representatives of the different clusters of *visually similar documents* within a large collection of Web pages. Such a tool could be quite helpful in a scenario where a rough draft for a Web page exists and we are interested in individualizing a possible (normalized) template for constructing such a Web page.

Our experiment was conducted as follows. First, we (pairwise) compared a sufficiently representative set of Web pages (randomly selected) from the collection given within the *CMU World Wide Knowledge Base* (Web→KB project [20]). In our experiment, we handled a set of 470 Web pages. From the computed similarity matrix, the values in the table were used as data points for the *complete-link hierarchical clustering* algorithm and represented as a dendrogram<sup>5</sup> (see Figure 6). For executing the clustering algorithms, we used the data mining library *RuG-L04* [21]. This way, we could identify and visually represent 11 clusters (each of groups is labeled with the number of group in Figure 6).

The next step was to identify the web pages (templates) that could be considered as the visual representatives of each of the clusters. First of all, we constructed a suitable bi-dimensional data representation by using the *classical multidimensional scaling* algorithm (MDS)<sup>6</sup>. Then, within each cluster, we run the *minimum distance* algorithm, which

<sup>5</sup>A dendrogram is a tree diagram used to illustrate the arrangement of the clusters produced by a clustering algorithm.

<sup>6</sup>Multidimensional Scaling refers to a family of models where the structure in a set of data is represented graphically by the relationships between a set of points in a space.

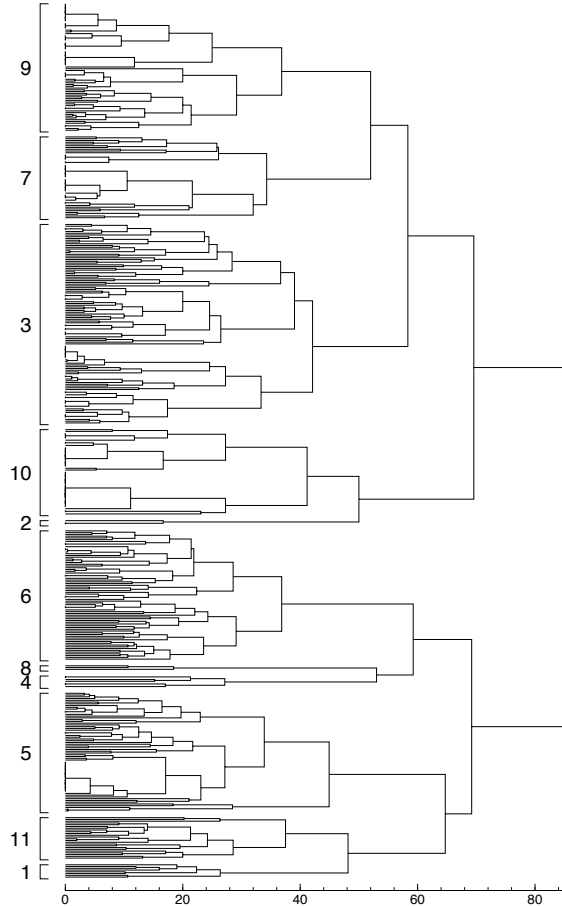


Figure 6. Dendrogram

computes the element with a minimum distance with respect to the rest of elements in a cluster. This way, we identified the web page templates that represents each of the clusters.

The table in Figure 7 summarizes the results of our Web document clustering experiment. Each row in the table corresponds to a group, and lists the group name, the number of elements in the group, the web page template that represents the group, and the average similarity percentage among the elements of this group.

The details of the experiments and some diagrams not included in this paper are available at <http://www.dsic.upv.es/~dromero/cmp.html>.

## VII. CONCLUSION

Web page comparison is currently an open problem whose importance extends from search engines to Web categorization. We define the “visual structure” of a Web page as the structure perceived by a human, and then develop a translation of Web pages to a canonical representative that effectively represents the visual structure of the page. A functional method for computing the similarity between

Groups	#	Rep.	%
Group 1	8	id94	82.76
Group 2	3	id379	92.59
Group 3	114	id330	84.97
Group 4	7	id31	82.99
Group 5	68	id191	86.22
Group 6	74	id499	84.87
Group 7	47	id371	87.64
Group 8	3	id217	89.73
Group 9	73	id296	89.01
Group 10	49	id363	87.06
Group 11	24	id299	83.31

Figure 7. Summary of Web pages templates

two Web pages is also proposed which is based on a generalization of the *tree edit distance* algorithm.

We have reimplemented in Maude our technique for recognizing and comparing the visual structural information of Web pages by using memoization. Our experiments demonstrate considerable improvements in both time and space over the former version of the tool, which makes the technique applicable in real scenarios.

As future work, we plan to extend our analysis by also considering stylesheets and the visual effect of including images on the web pages. This extension can be easily achieved by dividing a Web page into a collection of HTML code blocks that are determined by the CSS attributes of the Web page and the dimensions of the images. Then the tree structure of the Web page can be easily assembled by considering the coordinates given by the property “position” of its elements. This simple extension allows us to use our visual comparison methodology for collections of Web pages that include stylesheets.

#### ACKNOWLEDGMENT

Thanks to Jose Hernandez–Orallo for many helpful discussions on the data mining applications of this work.

#### REFERENCES

- [1] P. Lakkaraju, S. Gauch, and M. Speretta, “Document similarity based on concept tree distance,” in *Proc. of the 19th ACM Conf. on Hypertext and hypermedia*. New York, NY, USA: ACM, 2008, pp. 127–132.
- [2] A. Paepcke, H. Garcia-Molina, G. Rodriguez-Mula, and J. Cho, “Beyond document similarity: understanding value-based search and browsing technologies,” *SIGMOD Rec.*, vol. 29, no. 1, pp. 80–92, 2000.
- [3] W. W. Cohen, “Recognizing structure in Web pages using similarity queries,” in *Proc. of the 16th Nat. Conf. on Artificial Intelligence and the 11th Innovative App. of Artificial Intelligence*, Menlo Park, CA, USA, 1999, pp. 59–66.
- [4] A. Y. Fu, L. Wenyin, and X. Deng, “Detecting phishing web pages with visual similarity assessment based on earth mover’s distance (emd),” *IEEE Trans. Dependable Secur. Comput.*, vol. 3, no. 4, pp. 301–311, 2006.
- [5] E. Medvet, E. Kirda, and C. Kruegel, “Visual-similarity-based phishing detection,” in *SecureComm ’08: Proc. of the 4th Intl. Conference on Security and Privacy in Communication Networks*. New York, NY, USA: ACM, 2008, pp. 1–6.
- [6] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, and X. Deng, “Detection of phishing webpages based on visual similarity,” in *Proc. of the 14th Intl. Conference on World Wide Web (WWW 2005), Chiba, Japan*. ACM, 2005, pp. 1060–1061.
- [7] M. Kudelka, V. Snasel, Z. Horak, and A. Abraham, “Social aspects of web page contents,” in *Proc. of the 2009 Intl. Conf. on Computational Aspects of Social Networks*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 80–87.
- [8] M. Alpuente and D. Romero, “A Visual Technique for Web Pages Comparison,” in *Proc of 4th Int’l Workshop on Automated Specification and Verification of Web Systems (WWW’08), Siena, Italy*. ENTCS, 2008, pp. 239–253.
- [9] K. Zhang and D. Shasha, “Simple fast algorithms for the editing distance between trees and related problems,” *SIAM J. Comput.*, vol. 18, no. 6, pp. 1245–1262, 1989.
- [10] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott, *All About Maude: A High-Performance Logical Framework*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2007, vol. 4350.
- [11] P. Bille, “A survey on tree edit distance and related problems,” *Theor. Comput. Sci.*, vol. 337, no. 1-3, pp. 217–239, 2005.
- [12] E. D. Demaine, S. Mozes, B. Rossman, and O. Weimann, “An optimal decomposition algorithm for tree edit distance,” in *Proc of the 34th Intl. Colloquium on Automata, Languages and Programming (ICALP’07)*, 2007, pp. 146–157.
- [13] N. Dershowitz and D. Plaisted., “Rewriting,” *Handbook of Automated Reasoning*, vol. 1, pp. 535–610, 2001.
- [14] W. Wide Web Consortium (W3C), “Extensible Markup Language (XML) 1.0, second edition,” 1999. [Online]. Available: <http://www.w3.org>
- [15] —, “Extensible HyperText Markup Language (XHTML),” 2000. [Online]. Available: <http://www.w3.org>
- [16] A. I. Solutions, “CSE HTML validator,” 2008. [Online]. Available: <http://www.htmlvalidator.com/>
- [17] W. Wide Web Consortium (W3C), “Markup Validation Service,” 2005. [Online]. Available: <http://validator.w3.org/>
- [18] K. Tai, “The tree-to-tree correction problem,” *J. ACM*, vol. 26, no. 3, pp. 422–433, 1979.
- [19] R. A. Wagner and M. J. Fischer, “The string-to-string correction problem,” *J. ACM*, vol. 21, no. 1, pp. 168–173, 1974.
- [20] CMU World Wide Knowledge Base (Web→KB) project. [Online]. Available: <http://www.cs.cmu.edu/WebKB/>
- [21] P. Kleiweg, “Rug-I04. software for dialectometrics and cartography.” [Online]. Available: <http://www.let.rug.nl/kleiweg/indexs.html>