Document downloaded from:

http://hdl.handle.net/10251/82004

This paper must be cited as:

Alarte-Aleixandre, J.; Insa Cabrera, D.; Silva, J.; Tamarit Muñoz, S. (2016). Site-Level Web Template Extraction Based on DOM Analysis. Lecture Notes in Computer Science. 9609:36-49. doi:10.1007/978-3-319-41579-6.

The final publication is available at

https://link.springer.com/chapter/10.1007/978-3-319-41579-6_4

Copyright Springer Verlag (Germany)

Additional Information

The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-41579-6_4

Site-Level Web Template Extraction based on DOM Analysis*

Julián Alarte[1], David Insa[1], Josep Silva[1], and Salvador Tamarit[2]

[1] Universitat Politècnica de València
Camino de Vera s/n, E-46022 Valencia, Spain.
{jalarte,dinsa,jsilva}@dsic.upv.es
[2] IMDEA Software, Universidad Politécnica de Madrid
Campus Montegancedo UPM, E-28223 Pozuelo de Alarcón, Madrid
stamarit@software.imdea.org

Abstract. One of the main development resources for website engineers are Web templates. Templates allow them to increase productivity by plugin content into already formatted and prepared pagelets. For the final user templates are also useful, because they provide uniformity and a common look and feel for all webpages. However, from the point of view of crawlers and indexers, templates are an important problem, because templates usually contain irrelevant information such as advertisements, menus, and banners. Processing and storing this information leads to a waste of resources (storage space, bandwidth, etc.). It has been measured that templates represent between 40% and 50% of data on the Web. Therefore, identifying templates is essential for indexing tasks. In this work we propose a novel method for automatic web template extraction that is based on similarity analysis between the DOM trees of a collection of webpages that are detected using an hyperlink analysis. Our implementation and experiments demonstrate the usefulness of the technique.

Keywords: Information Retrieval, Content Extraction, Template Extraction

1 Introduction

A web template (in the following just template) is a prepared HTML page where formatting is already implemented and visual components are ready to insert content into them. Templates are an essential component of nowadays websites, and they are important for web developers, users, and also for indexers and crawlers:

* This work has been partially supported by the EU (FEDER) and the Spanish *Ministerio de Economía y Competitividad (Secretaría de Estado de Investigación, Desarrollo e Innovación)* under grant TIN2013-44742-C4-1-R and by the *Generalitat Valenciana* under grant PROMETEOII/2015/013. David Insa was partially supported by the Spanish Ministerio de Eduación under FPU grant AP2010-4415.

- Web developers use templates as a basis for composing new webpages that share a common look and feel. This also allows them to automate many tasks thanks to the reuse of components. In fact, many websites are maintained automatically by code generators that generate webpages using templates.
- Users can benefit from intuitive and uniform designs with a common vocabulary of colored and formatted visual elements.
- Crawlers and indexers usually judge the relevance of a webpage according to the frequency and distribution of terms and hyperlinks. Since templates contain a considerable number of common terms and hyperlinks that are replicated in a large number of webpages, relevance may turn out to be inaccurate, leading to incorrect results (see, e.g., [3, 21, 19]). Moreover, in general, templates do not contain relevant content, they usually contain one or more pagelets [7, 3] (i.e., self-contained logical regions with a well defined topic or functionality) where the main content must be inserted. Therefore, detecting templates helps indexers to identify the main content of the webpage. Gibson et al. [10] determined that templates represent between 40% and 50% of data on the Web and that around 30% of the visible terms and hyperlinks appear in templates. This justifies the importance of template removal [21, 19] for web mining and search.

Our approach to template extraction is based on the DOM [8] structures that represent webpages. Roughly, given a webpage in a website, (1) we first identify a set of webpages that are likely to share a template with it, and then, (2) we analyze these webpages to identify the part of their DOM trees that is common with the original webpage. (3) This slice of the DOM tree is returned as the template.

Some of the ideas in this paper were previously discussed in their earlier version in [1, 2]. Herein we further develop them, we put them all together, we add new technical results and algorithms, and finally, we describe our implementation of the whole system.

The rest of the paper has been structured as follows: In Section 2 we discuss the state of the art and show some problems of current techniques that can be solved with our approach. In Section 3, we present our technique with examples and explain the algorithms used. In Section 4 we give some details about the implementation and show the results obtained from a collection of benchmarks. Finally, Section 5 concludes.

2 Related Work

Template detection and extraction are hot topics due to their direct application to web mining, searching, indexing, and web development. For this reason, there are many approaches that try to face this problem. Some of them have been presented in the CleanEval competition [4], which periodically proposes a collection of examples to be analyzed with a gold standard. The examples proposed are especially thought for boilerplate removal and content extraction.

Content Extraction is a discipline very close to template extraction. Content extraction tries to isolate the pagelet with the main content of the webpage. It is an instance of a more general discipline called *Block Detection* that tries to isolate every pagelet in a webpage. There are many works in these fields (see, e.g., [11, 20, 6, 12]), and all of them are directly related to template extraction.

In the area of template extraction, there are three main different ways to solve the problem, namely, (i) using the textual information of the webpage (i.e., the HTML code), (ii) using the rendered image of the webpage in the browser, and (iii) using the DOM tree of the webpage.

The first approach is based on the idea that the main content of the webpage has more density of text with less labels. For instance, the main content can be identified selecting the largest contiguous text area with the least amount of HTML tags [9]. This has been measured directly on the HTML code by counting the number of characters inside text and the number of labels. This measure produces a ratio called CETR [20] used to discriminate the main content. Other approaches exploit densitometric features based on the observation that some specific terms are more common in templates [16, 14]. The distribution of the code between the lines of a webpage is not necessarily the one expected by the user. The format of the HTML code can be completely unbalanced (i.e., without tabulations, spaces or even carriage returns), specially when it is generated by a non-human directed system. As a common example, the reader can see the source code of the main Google's webpage. At the time of writing these lines, all the code of the webpage is distributed in only a few lines without any legible structure. In this kind of webpages CETR is useless.

The second approach assumes that the main content of a webpage is often located in the central part and (at least partially) visible without scrolling [5]. This approach has been less studied because rendering webpages for classification is a computational expensive operation [15].

The third approach is where our technique falls. While some works try to identify pagelets analyzing the DOM tree with heuristics [3], others try to find common subtrees in the DOM trees of a collection of webpages in the website [21, 19]. Our technique is similar to these last two works.

Even though [21] uses a method for template extraction, its main goal is to remove redundant parts of a website. For this, they use the Site Style Tree (SST), a data structure that is constructed by analyzing a set of DOM trees and recording every node found, so that repeated nodes are identified by using counters in the SST nodes. Hence, an SST summarizes a set of DOM trees. After the SST is built, they have information about the repetition of nodes. The most repeated nodes are more likely to belong to a noisy part that is removed from the webpages.

In [19], the approach is based on discovering optimal mappings between DOM trees. This mapping relates nodes that are considered redundant. Their technique uses the RTDM-TD algorithm to compute a special kind of mapping called *restricted top-down mapping* [18]. Their objective, as ours, is template extraction, but there are two important differences. First, we compute another kind of map-

ping to identify redundant nodes. Our mapping is more restrictive because it forces all nodes that form pairs in the mapping to be equal. Second, in order to select the webpages of the website that should be mapped to identify the template, they pick random webpages until a threshold is reached. In their experiments, they approximated this threshold as a few dozens of webpages. In our technique, we do not select the webpages randomly, we use a new method to identify the webpages linked by the main menu of the website. We only need to explore a few webpages to identify the webpages that implement the template. Moreover, contrarily to us, they assume that all webpages in the website share the same template, and this is a strong limitation for many websites.

3 Template extraction

Our technique inputs a webpage (called key page) and it outputs its template. To infer the template, we identify what concrete other webpages in the same website should be analyzed. Our approach introduces three new ideas to solve the following three problems:

1. Minimize the number of webpages to be analyzed from the (usually huge) universe of directly or indirectly linked webpages. For this, starting from the key page, we identify a *complete subdigraph* in the website topology.
2. Solve conflicts between those webpages that implement different templates. For this, we establish a *voting system* between the webpages.
3. Extract the template by comparing the set of webpages analyzed. For this, we calculate a new mapping called *equal top-down mapping* (ETDM) between the DOM tree of the key page and the DOM trees of the webpages in the complete subdigraph.

The three processes are explained in the following sections.

3.1 Finding webpage candidates to extract the template

The first phase of our technique identifies a set of webpages that share their template with the key page. This phase was proposed and described in [1] as an independent process that can be used by any template extraction technique. In fact, this phase is orthogonal to the other phases that extract the template. Roughly, we detect the template's menu and analyze the hyperlinks of the menu to identify a set of mutually linked webpages. One of the main functions of a template is in aiding navigation, thus almost all templates provide a large number of hyperlinks, shared by all webpages implementing the template. Locating the menu allows us to identify in the topology of the website the main webpages of each category or section. These webpages very likely share the same template.

Given a website topology, a *complete subdigraph* (CS) represents a collection of webpages that are pairwise mutually linked. A n-complete subdigraph (n-CS) is formed by n nodes. Our interest in complete subdigraphs comes from the observation that the webpages linked by the items in a menu usually form a CS.

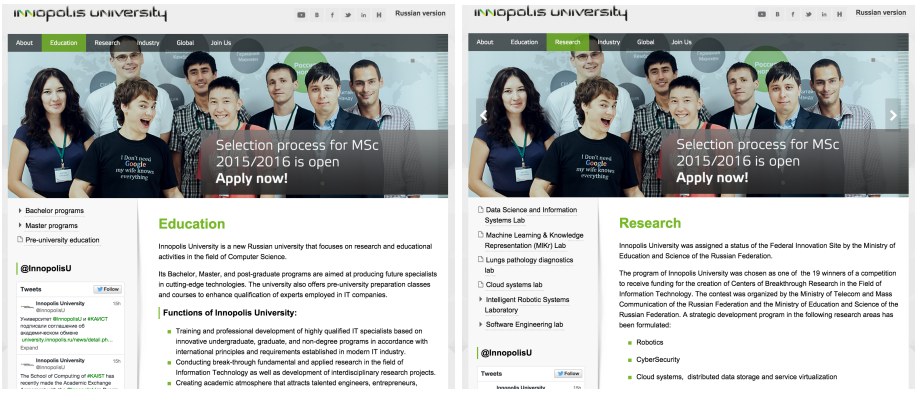

Fig. 1. Webpages of Innopolis University sharing a template

This is a new way of identifying the webpages that contain the menu. At the same time, these webpages are the roots of the sections linked by the menu, and thus they very likely share a common template.

Example 1. In Figure 1, we see two webpages of Innopolis University that share the same template. The left webpage is reached from the menu option "Education". The right webpage is reached from the menu option "Research". In both pages the main content is at the bottom right. They both share the same header, menu, and general structure, and they form a 2-CS. Similarly, the 6 webpages linked by the menu at the top form a 6-CS, and they all implement the whole template. Our technique uses these webpages as candidates.

This simple idea is so powerful that it significantly increases the quality of the webpage candidates (main webpages of a category normally maximize the amount of template implemented), and at the same time it increases performance: contrarily to other approaches, we only need to investigate a reduced set of webpages linked by the key page, because they will for sure contain a CS that represents the menu. Contrarily to our approach, with independence of the approach followed to compare the candidates, the most extended way of selecting them is manually. For instance, the *ContentExtractor* algorithm and its improved version, the *FastContentExtractor* algorithm [17], take as input a set of webpages that are given by the programmer. The same happens in the methodology proposed in [13].

Other approaches select the candidates randomly. For instance, in [21], SSTs are built from a collection of webpages. They do not have a methodology to select the webpages, and they do not propose a number of webpages needed. In their experiments, they randomly sample 500 webpages, and the time taken to build a SST is always below 20 seconds. Similarly, in [19], in order to select the webpages of the website that should be mapped to identify the template, they pick random webpages until a threshold is reached. In their experiments, they approximated this threshold as a few dozen of webpages. They need 25 webpages

to reach a 0.95 F1 measure using a collection of product description webpages that share the same template. Therefore, contrarily to us, they assume that all webpages in the website share the same template, and this is too restrictive for many websites.

By analyzing the hyperlinks (in the following, links) in the key page, it is possible to select those links that most likely produce a CS. This is essential to avoid analyzing all links and thus significantly increasing the performance. Our strategy to identify the links that should be analyzed is based on the structure of the website, and the structure of the website can be inferred from the own links. In particular, by analyzing the links in the key page, we can establish an order of relevance (i.e., an order that states what links should be analyzed first). For this, we use the hyperlink distance and the DOM distance:

Hyperlink distance. It represents the distance in the file system between the directories pointed by two links. This can be observed in Figure 2 (left), which represents a tree of directories that contain webpages. There, we can see the distance of the webpage in the gray directory to the rest of webpages. Note that the hyperlink distance can be negative and it is asymmetric. This can be also observed in Figure 2 (right) where hyperlink distance is represented with *hDistance*.

DOM distance. It is just the standard tree nodes distance in the DOM tree between two link nodes. Hence, two hyperlink nodes have zero DOM distance if and only if they are exactly the same node. Contrarily, two different hyperlink nodes (even if they have the same URL, and thus the same hyperlink distance) necessarily have a positive DOM distance. An example of DOM distance can be observed in Figure 2 (right) where DOM distance is represented with *dDistance*.

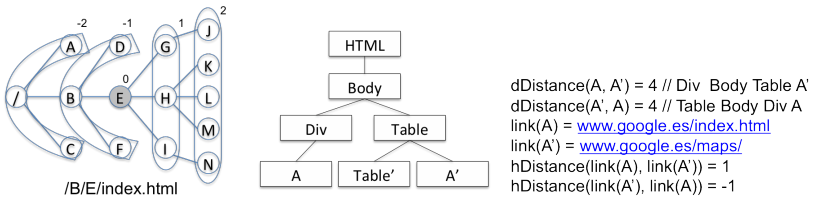

Fig. 2. Hyperlink distance (left). A DOM tree T (center) with its information (right).

There exists a clear relation between hyperlink distance and the probability of the linked webpages to share the template. Another observation is that we want the candidates that share the template to be as different as possible to ensure representativity of the website (e.g., avoiding to select all webpages about the same sport in a sports website). Therefore, the process of obtaining webpages that share the same template tries to identify webpages with an hyperlink

distance as close to zero as possible, but at the same time maximizing the DOM distance (to ensure that the webpages are as different as possible), and giving priority to the hyperlink distance.

Concretely, to compute the n-CS, we sort the links of the key page, and iteratively explore them until they form a n-CS. The order of the links is created using both the hyperlink distance and the DOM distance. The order is the following: First, those links with zero hyperlink distance, then, those links that are closer to the key page with a positive distance, and finally those links that are closer to the key page with a negative distance. In the three cases, if a draw occurs, then, the draw is broken using the DOM distance: those links that are farer to the already selected links are collected. A formalization together with the algorithms used to compute a n-CS can be found in [1].

3.2 Solving conflicts between webpages with different templates

One problem that we detected in previous techniques is the general assumption that the website has a unique template. Contrarily, a single webpage can implement various templates, or even subsets of different templates. This is illustrated in the following example.

Example 2. In Figure 3 we see the key page and two webpages used to extract its template. The two webpages implement a different template, and they are disjoint except for the root node. If we assume that all webpages implement the same template, then, the template extracted would be only the root node (it is the only one shared by the three webpages). Contrarily, it is possible that the key page implements a part of the template of one webpage, and a part of the other webpage, being the template the gray nodes. Thus, even if the webpage candidates are disjoint, they can contribute to the template.

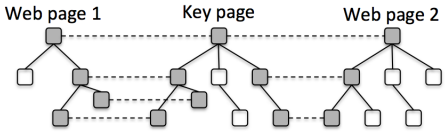

Fig. 3. Template extracted from webpages with different template

Example 2 shows that not all webpages must share a node to consider this node as template. But, how many of them are necessary? The answer is: it depends on the size of the CS. We experimented with a benchmark suite and measured the recall and precision obtained with all combinations of CS size and number of votes needed. The results are summarized in the first two columns of Table 2. For instance, with a CS of size 6, 3 votes are enough to get the best F1. As a result, our algorithm implements a voting system to extract the template

from a set of candidates, and it uses a parameter that represents the number of votes needed for a node to be considered template. This algorithm is presented in the next section in such a way that it is parametric so that it can be used for any size of the CS and for any number of votes needed.

3.3 Template extraction from a complete subdigraph

In the following, given a DOM tree $T = (N, E)$, $parent(n)$ represents node $n' \in N$ such that $(n', n) \in E$. Similarly, $subtree(n)$ denotes the subtree of T whose root is $n \in N$.

In order to identify the part of the DOM tree that is common in a set of webpages, our technique uses an algorithm that is based on the notion of mapping. A mapping establishes a correspondence between the nodes of two trees. In order to identify templates, we define a very specific kind of mapping that we call *equal top-down mapping* (ETDM) (see Figure 4).

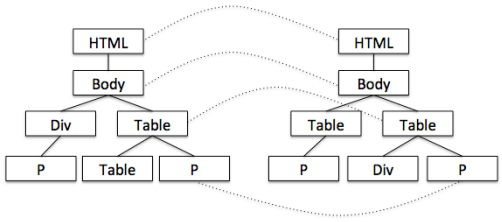

Fig. 4. Equal top-down mapping between DOM trees

Definition 1 (equal top-down mapping). *A* mapping *from a tree $T = (N, E)$ to a tree $T' = (N', E')$ is any set M of pairs of nodes $(n, n') \in M$, $n \in N, n' \in N'$ such that, for any two pairs (n_1, n'_1) and (n_2, n'_2) in M, $n_1 = n_2$ iff $n'_1 = n'_2$. Given an equality relation $\triangleq$ between tree nodes, a mapping M between two trees T and T' is said to be* equal top-down *if and only if*

- *equal: for every pair $(n, n') \in M$, $n \triangleq n'$.*
- *top-down: for every pair $(n, n') \in M$, with $n \neq root(T)$ and $n' \neq root(T')$, there is also a pair $(parent(n), parent(n')) \in M$.*

Note that this definition is parametric with respect to the equality relation $\triangleq$. We could simply use the standard equality ($=$), but we left this relation open, to be general enough as to cover any possible implementation. In particular, other techniques consider that two nodes n_1 and n_2 are equal if they have the same label. However, in our implementation we use a notion of node equality much more complex that compares two nodes considering their HTML *id*, CSS classes, their number of children, their relative position in the DOM tree, and their HTML attributes. We refer the interested reader to our open and free

implementation (`http://www.dsic.upv.es/~jsilva/retrieval/templates/`) where relation $\triangleq$ is specified.

This definition of mapping allows us to be more restrictive than other mappings such as, e.g., the *restricted top-down mapping* (RTDM) introduced in [18]. While RTDM permits the mapping of different nodes (e.g., a node labelled with *table* with a node labelled with *div*), ETDM can force all pairwise mapped nodes to have the same label. Figure 4 shows an example of an ETDM using: $n \triangleq n'$ if and only if n and n' have the same label.

After we have found the webpage candidates (the CS), we identify an ETDM between the key page and a set of webpages in the CS. For this, initially, the template is considered to be empty. Then, we iteratively compute an ETDM between the template and v webpages in the set, being v the number of votes needed for a node to be considered template. The result is a template with all those nodes of the key page appearing in at least v other webpages of the CS. This process is formalized in Algorithm 1, which uses function *ETDM* to compute the biggest ETDM between a set of trees. Algorithm 1 uses a loop (`foreach` $(\{p_1 \dots p_v\}$ `in` $P)$) that iterates over all possible partitions of P formed with v pages (because v votes are needed). Then, an ETDM is computed between these webpages and the key page. Observe that function *ETDM* is recursive. It traverses the trees top-down collecting all those nodes that are equal modulo $\triangleq$. Note that function *ETDM* assumes that, given two webpages $p_1 = (N_1, E_1), p_2 = (N_2, E_2)$, only one node $n_1 \in N_1$ satisfies $n_1 \triangleq n_2$ for a given $n_2 \in N_2$. Of course, this strictly depends on the definition of $\triangleq$. In the case that $\exists\ n_1, n_1' \in N_1,\ n_2 \in N_2\ .\ n_1 \triangleq n_2\ \wedge\ n_1' \triangleq n_2$, then, the algorithm should be augmented with a mechanism to select only one node (either n_1 or n_1').

Algorithm 1 Extract a template from a set of webpages

Input: A key page $p_k = (N, E)$, a set P of n webpages, and the number of votes v needed for a node to be considered template.
Output: A template for p_k with respect to P and v.

begin
  $template = (N_t, E_t) = (\emptyset, \emptyset)$;
  foreach $(\{p_1 \dots p_v\}$ **in** $P)$
    if $root(p_k) \triangleq root(p_1) \triangleq \dots \triangleq root(p_v)$
      $(N', E') = ETDM(p_k, p_1, \dots, p_v)$;
      $(N_t, E_t) = (N_t \cup N', E_t \cup E')$;
      $template = (N_t, E_t)$;
  return $template$;
end

function $ETDM$(tree $T_0 = (N_0, E_0)$, tree $T_1 = (N_1, E_1)$, ..., tree $T_v = (N_v, E_v)$)
  $r_0 = root(T_0)$; $r_1 = root(T_1)$; ...; $r_v = root(T_v)$;
  nodes $= \{r_0\}$;
  edges $= \emptyset$;
  foreach $n_0 \in N_0$, ..., $n_v \in N_v\ .\ n_0 \triangleq \dots \triangleq n_v, (r_0, n_0) \in E_0$, ..., $(r_v, n_v) \in E_v$
    $(nodes\_st, edges\_st) = ETDM(subtree(n_0), \dots, subtree(n_v))$;
    $nodes = nodes \cup nodes\_st$;
    $edges = edges \cup edges\_st \cup \{(r_0, n_0)\}$;
  return $(nodes, edges)$;
end function

As in Definition 1, we left the algorithm parametric with respect to equality relation $\triangleq$. This is done on purpose, because this relation is the only parameter that is subjective and thus, it is a good design decision to leave it open. For instance, a researcher can decide that two DOM nodes are equal if they have the same label and attributes. Another researcher can relax this restriction ignoring some attributes (i.e, the template can be the same, even if there are differences in colors, sizes, or even positions of elements. It usually depends on the particular use of the extracted template). Additionally, this design decision gives us control over the recall and precision of the technique. Because the more restrictive $\triangleq$ is, the more precision (and less recall).

4 Implementation

The technique presented in this paper, including all the algorithms, has been implemented as a Firefox's extension accepted by Mozilla as an official add-on (`https://addons.mozilla.org/en-US/firefox/addon/template-extractor/`). In this tool, the user can browse on the Internet as usual. Then, when he/she wants to extract the template of a webpage, he/she only needs to press the "Extract Template" button and the tool automatically (internally) loads the appropriate linked webpages to form a CS, analyzes them, and extracts the template. The template is then displayed in the browser as any other webpage.

4.1 Empirical evaluation

Initially, we wanted to use a public standard collection of benchmarks to evaluate our tool, but we are not aware of any public dataset for template extraction. In particular, the standard CleanEval suite [4] contains a gold standard prepared for content extraction (each part of the webpages is labelled as *main-content* or *non-content*), but it is not prepared for template extraction. We tried to use the same benchmark set as the authors of other template extraction papers. However, due to privacy restrictions, copyright, or unavailability[3] of the benchmarks we could not use a previous dataset. It is surprising, and quite disappointing, to see how few systems are open-source, or even otherwise (freely) available. In many papers, it is stated that a prototype was developed but we were not able to find the tool. In some cases, a system might be mentioned to be open source but you need to contact the authors to get it. This is the cause why we are reinventing the wheel, implementing similar systems once and again. Moreover, not providing the dataset makes impossible to validate or replicate experiments. For this reason, we made our system, open-source and publicly available, so that other researchers can reuse it or join efforts to further developing it. And we decided to create a new suite of benchmarks that is also publicly accessible, both the dataset and the gold standard. This is one of the main contributions of

[3] Some authors answered that their benchmarks were not stored for future use, or that they did not save the gold standard.

our work. Any interested researcher can freely access and download our dataset from: `http://www.dsic.upv.es/~jsilva/retrieval/teco/`.

The dataset is composed of a collection of web domains with different layouts and page structures. This allows us to study the performance of the techniques in different contexts (e.g., company websites, news articles, forums, etc.). To measure our technique, we randomly selected an evaluation subset. Table 1 summarizes the results of the performed experiments. First column contains the URLs of the evaluated website domains. For each benchmark, column `DOM` shows the number of nodes in the key page's DOM tree; column `Template` shows the number of nodes in the gold standard template; column `Retrieved` shows the number of nodes that were identified by the tool as the template; column `Recall` shows the number of correctly retrieved nodes divided by the number of nodes in the gold standard; column `Precision` shows the number of correctly retrieved nodes divided by the number of retrieved nodes; column `F1` shows the F1 metric that is computed as $(2 * P * R)/(P + R)$ being P the precision and R the recall; finally, column `Time` shows the total milliseconds used to obtain the template.

Experiments reveal an average precision of more than 96%, and an average recall of more than 95% which, from the best of our knowledge, produce the highest F1 in the state of the art. To produce this result, we have performed more than half a million experiments to tune our definition of $\triangleq$ combining different DOM parameters such as label, class, id, children, position, etc. See `http://www.dsic.upv.es/~jsilva/retrieval/templates/` for details.

In the experiments, we also evaluated empirically what is the ideal size of the CS computed. Results are shown in Table 2. This table summarizes many experiments. Each row is the average of repeating all the experiments in Table 1 with a different value for n in the n-CS searched by the algorithm and for a different value for all $v < n$. All possible combinations were evaluated. Column `Size` represents the size of the CS that the algorithm tried to find in the websites. And column `Votes` represents the best v value obtained for each CS size. In the case that there did not exist a CS of the searched size, then the algorithm used the biggest CS with a size under the specified size. Column `Loads` represents the average number of webpages loaded to extract the template.

We determined that a subdigraph of size 3 is the best option because it keeps almost the best F1 value, while being very efficient (only 10 webpages must be loaded to extract the template). Therefore, the results shown in Table 1 have been computed with a 3-CS.

5 Conclusions

This work presents a new technique for template extraction. It uses a hyperlink analysis to identify the menu of a given webpage. With this menu, the technique collects a set of webpages that form a CS and, thus, they probably share the same template. The DOM structures of these webpages are then compared with a new mapping called ETDM to identify the blocks that are common to some of them. The exact number has been approximated empirically. Our best values

Benchmark	DOM	Template	Retrieved	Recall	Precision	F1	Time
water.org	948	711	668	93,95 %	100 %	96,88 %	5661
www.jdi.org.za	626	305	305	100 %	100 %	100 %	2928
stackoverflow.com	6450	447	461	100 %	96,96 %	98,46 %	18348
www.eclipse.org	256	156	160	97,44 %	95 %	96,20 %	3382
www.history.com	1246	669	593	88,19 %	99,49 %	93,50 %	16946
www.landcoalition.org	1247	393	433	98,47 %	89,38 %	93,70 %	4901
es.fifa.com	1324	276	239	84,78 %	97,91 %	90,87 %	8171
cordis.europa.eu	959	335	327	97,01 %	99,39 %	98,19 %	5115
clotheshor.se	459	231	225	97,40 %	100 %	98,68 %	2176
www.emmaclothes.com	1080	374	368	98,40 %	100 %	99,19 %	8641
www.cleanclothes.org	1335	266	288	100 %	92,36 %	96,03 %	7725
www.mediamarkt.es	805	337	329	97,63 %	100 %	98,80 %	5903
www.ikea.com	1545	407	565	99,75 %	71,86 %	83,54 %	7326
www.swimmingpool.com	607	499	349	69,94 %	100 %	82,31 %	2514
www.skipallars.cat	1466	842	828	98,34 %	100 %	99,16 %	10042
www.tennis.com	1300	463	419	90,50 %	100 %	95,01 %	7312
www.tennischannel.com	661	303	236	77,89 %	100 %	87,57 %	3520
www.turfparadise.com	1057	726	818	99,72 %	88,51 %	93,78 %	6756
riotimesonline.com	2063	879	861	97,96 %	100 %	98,97 %	50528
www.beaches.com	1928	1306	1172	89,74 %	100 %	94,59 %	11201
users.dsic.upv.es/~jsilva	197	163	163	100 %	100 %	100 %	7419
users.dsic.upv.es/~dinsa	241	74	88	100 %	84,09 %	91,36 %	1457
www.engadget.com	1818	768	767	99,09 %	99,22 %	99,15 %	19116
www.bbc.co.uk/news	2991	364	355	97,53 %	100 %	98,75 %	13806
www.vidaextra.com	2331	1137	992	87,25 %	100 %	93,19 %	17787
www.ox.ac.uk/staff	948	525	533	99,43 %	97,94 %	98,68 %	59599
clinicaltrials.gov	543	389	394	97,17 %	95,94 %	96,55 %	4746
en.citizendium.org	1083	414	447	100 %	92,62 %	96,17 %	13414
www.filmaffinity.com	1333	351	355	100 %	98,87 %	99,43 %	5279
edition.cnn.com	3934	192	180	93,75 %	100 %	96,77 %	31076
www.lashorasperdidas.com	1822	553	536	96,93 %	100 %	98,44 %	19379
labakeryshop.com	1368	218	193	80,73 %	91,19 %	85,64 %	7893
www.felicity.co.uk	300	232	232	100 %	100 %	100 %	2217
www.thelawyer.com	3349	1293	1443	93,81 %	84,06 %	88,67 %	19998
www.us-nails.com	250	184	215	100 %	85,58 %	92,23 %	3386
www.informatik.uni-trier.de	3085	64	63	98,44 %	100 %	99,21 %	10174
www.wayfair.co.uk	1950	702	700	99,29 %	99,57 %	99,43 %	30990
catalog.atsfurniture.com	340	301	304	100 %	99,01 %	99,50 %	2862
www.glassesusa.com	1952	1708	1656	96,96 %	100 %	98,45 %	19462
www.mysmokingshop.co.uk	575	407	428	100 %	95,09 %	97,49 %	89887
Average	1444,3	499,1	492,2	95,44 %	96,35 %	95,61 %	14226,08

Table 1. Results of the experimental evaluation

Size	Votes	Recall	Precision	F1	Loads
1	1	88,56 %	94,89 %	88,69 %	2
2	1	96,34 %	90,32 %	91,93 %	5,6
3	2	95,44 %	96,35 %	95,61 %	10,13
4	3	94,61 %	96,88 %	95,27 %	16,52
5	3	94,69 %	96,96 %	95,40 %	18,68
6	3	95,21 %	96,82 %	95,69 %	23,68
7	3	95,46 %	96,31 %	95,57 %	30
8	4	95,14 %	96,57 %	95,54 %	32,08

Table 2. Determining the ideal size of the complete subdigraph

considering both F1 and performance are a size of the CS of 3, and 2 votes needed to be considered template. To the best of our knowledge, the idea of using the

menus to locate the template is new, and it allows us to quickly find a set of webpages from which we can extract the template. This is especially interesting for performance, because loading webpages to be analyzed is expensive, and this part of the process is minimized in our technique. As an average, our technique only loads 10 pages to extract the template (a mean of less than 15 seconds for the overall template extraction process).

References

1. Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. Automatic detection of webpages that share the same web template. In Maurice H. ter Beek and António Ravara, editors, *Proceedings of the 10th International Workshop on Automated Specification and Verification of Web Systems (WWV 14)*, volume 163 of *Electronic Proceedings in Theoretical Computer Science*, pages 2–15. Open Publishing Association, July 2014.
2. Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. Web template extraction based on hyperlink analysis. In Santiago Escobar, editor, *Proceedings of the XIV Jornadas sobre Programación y Lenguajes (PROLE 15)*, volume 173 of *Electronic Proceedings in Theoretical Computer Science*, pages 16–26. Open Publishing Association, September 2015.
3. Ziv Bar-Yossef and Sridhar Rajagopalan. Template Detection via data mining and its applications. In *Proceedings of the 11th International Conference on World Wide Web (WWW'02)*, pages 580–591, New York, NY, USA, 2002. ACM.
4. Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. Cleaneval: a competition for cleaning web pages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC'08)*, pages 638–643. European Language Resources Association, may 2008.
5. Radek Burget and Ivana Rudolfova. Web page element classification based on visual features. In *Proceedings of the 1st Asian Conference on Intelligent Information and Database Systems (ACIIDS'09)*, pages 67–72, Washington, DC, USA, 2009. IEEE Computer Society.
6. Eduardo Cardoso, Iam Jabour, Eduardo Laber, Rogério Rodrigues, and Pedro Cardoso. An efficient language-independent method to extract content from news webpages. In *Proceedings of the 11th ACM symposium on Document Engineering (DocEng'11)*, pages 121–128, New York, NY, USA, 2011. ACM.
7. Soumen Chakrabarti. Integrating the Document Object Model with hyperlinks for enhanced topic distillation and information extraction. In *Proceedings of the 10th International Conference on World Wide Web (WWW'01)*, pages 211–220, New York, NY, USA, 2001. ACM.
8. W3C Consortium. Document Object Model (DOM). Available at URL: `http://www.w3.org/DOM/`, 1997.
9. Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. Introducing and evaluating ukWaC, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4)*, pages 47–54, 2008.
10. David Gibson, Kunal Punera, and Andrew Tomkins. The volume and evolution of web page templates. In Allan Ellis and Tatsuya Hagino, editors, *Proceedings of the 14th International Conference on World Wide Web (WWW'05)*, pages 830–839. ACM, may 2005.

11. Thomas Gottron. Content code blurring: A new approach to Content Extraction. In A. Min Tjoa and Roland R. Wagner, editors, *Proceedings of the 19th International Workshop on Database and Expert Systems Applications (DEXA'08)*, pages 29–33. IEEE Computer Society, sep 2008.
12. David Insa, Josep Silva, and Salvador Tamarit. Using the words/leafs ratio in the DOM tree for Content Extraction. *The Journal of Logic and Algebraic Programming*, 82(8):311–325, 2013.
13. Vidya Kadam and Prakash R. Devale. A methodology for template extraction from heterogeneous web pages. *Indian Journal of Computer Science and Engineering (IJCSE)*, 3(3), jun-jul 2012.
14. Christian Kohlschütter. A densitometric analysis of web template content. In Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl, editors, *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*, pages 1165–1166. ACM, apr 2009.
15. Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text teatures. In Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu, editors, *Proceedings of the 3rd International Conference on Web Search and Web Data Mining (WSDM'10)*, pages 441–450. ACM, feb 2010.
16. Christian Kohlschütter and Wolfgang Nejdl. A densitometric approach to web page segmentation. In James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury, editors, *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*, pages 1173–1182. ACM, oct 2008.
17. Dat Quoc Nguyen, Dai Quoc Nguyen, Son Bao Pham, and The Duy Bui. A fast template-based approach to automatically identify primary text content of a web page. In *Proceedings of the 2009 International Conference on Knowledge and Systems Engineering*, KSE 2009, pages 232–236. IEEE Computer Society, 2009.
18. Davi de Castro Reis, Paulo Braz Golgher, Altigran Soares Silva, and Alberto Henrique Frade Laender. Automatic web news extraction using tree edit distance. In *Proceedings of the 13th International Conference on World Wide Web (WWW'04)*, pages 502–511, New York, NY, USA, 2004. ACM.
19. Karane Vieira, Altigran S. da Silva, Nick Pinto, Edleno S. de Moura, João M. B. Cavalcanti, and Juliana Freire. A fast and robust method for web page template detection and removal. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM'06)*, pages 258–267, New York, NY, USA, 2006. ACM.
20. Tim Weninger, William Henry Hsu, and Jiawei Han. CETR: Content Extraction via tag ratios. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*, pages 971–980. ACM, apr 2010.
21. Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD'03)*, pages 296–305, New York, NY, USA, 2003. ACM.