What Web Template Extractor Should I Use? A Benchmarking and Comparison for Five Template Extractors

JULIÁN ALARTE and JOSEP SILVA, Universitat Politècnica de València, Spain
SALVADOR TAMARIT, Universitat Politècnica de Madrid, Spain

A Web template is a resource that implements the structure and format of a website, making it ready for plugging content into already formatted and prepared pages. For this reason, templates are one of the main development resources for website engineers, because they increase productivity. Templates are also useful for the final user, 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. There exist many techniques and tools for template extraction, but, unfortunately, it is not clear at all which template extractor should a user/system use, because they have never been compared, and because they present different (complementary) features such as precision, recall, and efficiency. In this work, we compare the most advanced template extractors. We implemented and evaluated five of the most advanced template extractors in the literature. To compare all of them, we implemented a workbench, where they have been integrated and evaluated. Thanks to this workbench, we can provide a fair empirical comparison of all methods using the same benchmarks, technology, implementation language, and evaluation criteria.

CCS Concepts: • **Information systems → Information extraction**; **Document filtering**; **Presentation of retrieval results**;

Additional Key Words and Phrases: Information retrieval, template extraction, content detection, web mining, block detection

ACM Reference format:
Julián Alarte, Josep Silva, and Salvador Tamarit. 2019. What Web Template Extractor Should I Use? A Benchmarking and Comparison for Five Template Extractors. *ACM Trans. Web* 13, 2, Article 9 (March 2019), 19 pages.
https://doi.org/10.1145/3316810

This work has been partially supported by the EU (FEDER) and the *Spanish Ministerio de Ciencia, Innovación y Universidades*/AEI under grant TIN2016-76843-C4-1-R and by the *Generalitat Valenciana* under grants PROMETEO-II/2015/013 (SmartLogic) and Prometeo/2019/098 (DeepTrust).
Authors' addresses: J. Alarte and J. Silva, Universitat Politècnica de València, Departament de Sistemes Informàtics i Computació, Camí de Vera s/n, València, E-46022, Spain; emails: jualal@doctor.upv.es, jsilva@dsic.upv.es; S. Tamarit, Universitat Politècnica de Madrid, Departamento de Computación, Campus de Montegancedo s/n, Madrid, 28660, Spain; email: salvador.tamarit@upm.es.

1559-1131/2019/03-ART9 $15.00
https://doi.org/10.1145/3316810

1 INTRODUCTION

Web template extraction. A web template (in the following just "template") is a predesigned resource that implements the structure for the comprehensive layout and display features of a website. It is often implemented with HTML and CSS and gives designers ways to plug content (e.g., text and images) into prepared containers, providing a basis for composing new webpages that share a common look and feel [4, 15] (see Figure 1).

Templates are good for web development, because many tasks can be automated thanks to the reuse of components [14, 15]. In fact, many websites are maintained automatically by code generators that generate webpages using templates. Templates are also good for users, who can benefit from intuitive and uniform designs with a common vocabulary of coloured and formatted visual elements [4, 53].

Besides, templates are also important for crawlers and indexers, because they usually judge the relevance of a webpage according to the frequency and distribution of terms and hyperlinks (in the following just "links") [7, 54]. Since templates contain a considerable number of common terms and links that are replicated in a large number of webpages, relevance may turn out to be inaccurate, leading to incorrect results (see, e.g., References [4, 54, 57]). Template extraction helps indexers to isolate the main content. This allows for enhancing indexers by assigning higher weights to the really relevant terms. Once templates have been extracted, they are processed for indexing—templates can be analyzed only once for all webpages that share the same template. Moreover, links in templates allow indexers to discover the topology of a website (e.g., through navigational content such as menus), thus identifying the main webpages. Gibson et al. [15] determined that templates represent between 40% and 50% of data on the Web and that around 30% of the visible terms and links appear in templates. This justifies the importance of template removal [25, 49, 54, 57] in web mining and search.

Other important fields where template extraction is particularly useful are boilerplate removal [12, 16, 46], wrapper generation [32, 43, 60], wrapper induction [40, 59], wrapper maintenance [29, 39], and automated data extraction (see, e.g., References [16, 27, 31]).

This article fills a gap in the area of web template extraction: It surveys for the first time, and from an empirical point of view, the main web template extractors. In particular, it uses concrete measures and focussed experiments to compare the template extractors. The main objective of this study is to help developers and researchers to choose one template extractor according to their needs.

Motivation. The requirements of a template extractor depend on the specific use and system where the template extractor will be inserted. Some systems require high precision, others require high recall, and for others efficiency is crucial. An analysis of the current literature about template extraction reveals that there are several different approaches, and each technique presents its own precision, recall, and runtime. Unfortunately, there does not exist a publicly available comparison of template extractors. One could make an exercise of literature review and produce a table that collects the precision, recall, efficiency, time trendlines, asymptotic costs, and so on, reported by the authors of each technique; but, unfortunately, this information is mostly useless, because this comparison would be unfair, biased, imprecise, and inaccurate. The reason is that each technique has been implemented with a different language and with different components that affect the efficiency. But also, they have been evaluated with different evaluation criteria (e.g., counting retrieved words [53, 54] vs. characters [25] vs. DOM nodes [2] vs. text blocks [47, 55]) and with a different collection of benchmarks. Using different benchmarks to compare template extractors is unacceptable, because some techniques used artificial webpages [6] (automatically generated webpages sharing exactly the same template), while others used real webpages implemented by

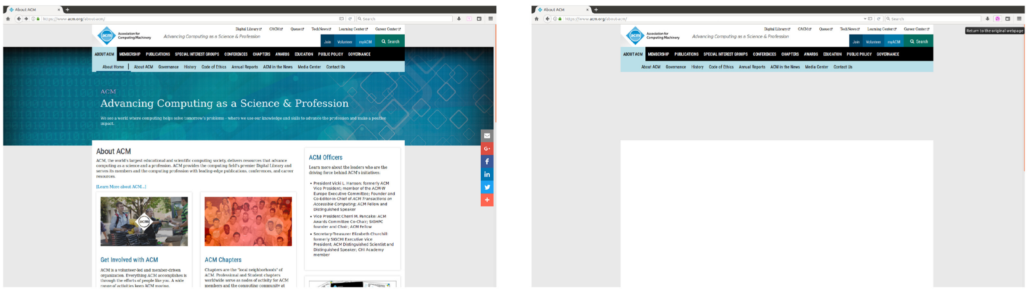

Fig. 1. Template (right) of ACM's "*about*" webpage (left) extracted with the template extractor TemEx.

heterogenous designers [2, 54]. Similarly, some used random webpages [26, 56, 57], possibly using different templates, while others manually provided webpages that implement exactly the same template [53, 54]. Other authors used the CleanEval [5] benchmark suite [46, 55].

Currently, there is no objective evidence, empirical data, or widely accepted (subjective) consensus about the current state of template extraction tools (regarding precision, recall, performance, scalability, and accepted technologies; i.e., HTML, Javascript, CSS, etc.). Moreover, the results reported in the bibliography are incomparable.

We wanted to compare all techniques with the same benchmarks and with the same accuracy measures, but we could not access the implementation of any tool even when they were reported as free. We asked the authors of the most known and cited template extractors to provide us their tools with very few positive answers. Unfortunately, very 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 access the tool.

To solve this, we reimplemented from scratch the main template extraction systems in the literature [2, 53, 54, 56, 57]. All of them are now available as open-source at http://www.dsic.upv.es/~jsilva/retrieval/Web-TemEx/extractors.html. Actually, our implementation is not a template extractor but a workbench for template extraction that includes several extraction facilities, including a set of template extractors. It is able to work offline (with a repository of websites) and online (it is integrated into the Firefox browser as a toolbox that allows for automatically extracting the template of a given website). One interesting property is that the loading of the webpages, the transformation from HTML to DOM trees, the renderization, and so on, is orthogonal to the analysis of the webpages. Therefore, these features can be used by different template extraction algorithms. This is an important contribution, because other researchers can now evaluate and compare their work using a common evaluation criterion but also a common workbench so that technology or external factors such as loading time do not influence the comparison.

Our experiments reveal the algorithm that produces the best recall, F1, and accuracy.[1] Moreover, our workbench solves one important problem of previous approaches: while previous algorithms select a set of webpages from a given website randomly, the new workbench implements a novel method to detect those webpages of a website that share the same template, thus increasing precision.

In this article, we describe the process used to select, implement, and compare the template extraction techniques, and the obtained results. In this process, we developed several tools that contribute to the advance of the web template extraction research area.

[1]The F1 metric is computed as $(2 * P * R)/(P + R)$, P being the precision and R the recall. The accuracy metric is computed as the number of correct classifications (e.g., DOM nodes classified either as template or not-template) divided by the total number of classifications.

The main contributions of our work are the following:

(1) A comparison and empirical evaluation of the main web template extraction tools.
(2) An open-source and publicly available implementation of five web template extraction tools.
(3) A workbench for web template extraction that includes:
   - All necessary infrastructure to load webpages, analyze and transform DOM trees, and visualize webpages.
   - Interfaces to add new template extraction algorithms to the workbench.
   - All necessary infrastructure to evaluate, test, and compare template extraction algorithms.
(4) A repository of benchmarks specific for template and content extraction.

Structure of the article. In Section 2, we describe the tools that participate in our study. In Section 3, we describe our workbench and give details about its implementation and about how it is being distributed. In Section 4, we compare all the tools from different perspectives, including recall, precision, computation time, scalability, and asymptotic costs. The comparison of template extractors is done with a benchmark suite that we specifically produced for this purpose. This suite is also described in Section 4. Finally, Section 5 concludes.

2 SELECTION AND DESCRIPTION OF WEB TEMPLATE EXTRACTORS

2.1 Methodology for the Selection of Template Extractors

In this section, we describe the process followed to identify and select the template extractors that participate in the study. We first formulate research questions and define inclusion and exclusion criteria. Then, we describe the processes of search and screening of primary studies.

(1) *Research questions.* Two research questions were formulated to identify the current state of the art in template detection and boilerplate removal methods:
   - RQ1: What methods for template detection and boilerplate removal have been developed? This research question intends to provide an overall perspective of the existing template detection and boilerplate removal methods, with special emphasis on those that have been developed over the last 15 years.
   - RQ2: What are the main characteristics of each template detection and boilerplate removal method? This question enhances the previous one, giving a deeper understanding of the template detection and boilerplate removal methods.
(2) *Search process.* We conducted a search process to assess the body of knowledge related to template extraction and to systematically deal with the research questions. This search process was strict and unbiased, and it involved the most relevant databases in the computer science area: Science Direct, Springer Computer Science, Web of Science, SCOPUS, IEEE Explore, Citeseer X, ACM Digital Library, Arxiv, and Google Scholar.
   For the search, we created the following search string:
   (template OR boilerplate OR noise)
   AND (detection OR extraction OR removal OR cleaning)
   AND ("web page" OR webpage)
   The search string was based on the analysis of several keywords from relevant literature, which was found by searching relevant articles and by analyzing their related bibliography.

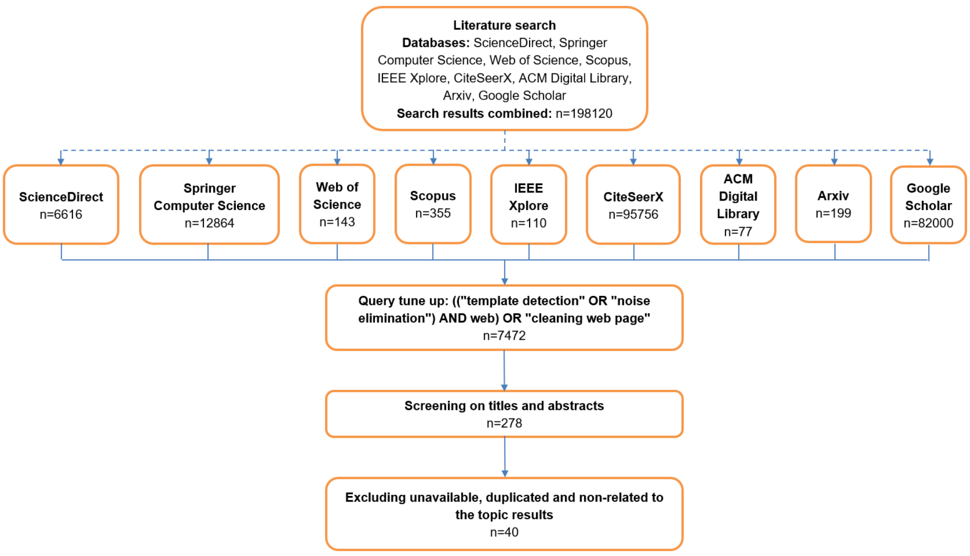
Literature search
Databases: ScienceDirect, Springer Computer Science, Web of Science, Scopus, IEEE Xplore, CiteSeerX, ACM Digital Library, Arxiv, Google Scholar
Search results combined: n=198120
ScienceDirect n=6616
Springer Computer Science n=12864
Web of Science n=143
Scopus n=355
IEEE Xplore n=110
CiteSeerX n=95756
ACM Digital Library n=77
Arxiv n=199
Google Scholar n=82000
Query tune up: (("template detection" OR "noise elimination") AND web) OR "cleaning web page" n=7472
Screening on titles and abstracts n=278
Excluding unavailable, duplicated and non-related to the topic results n=40

Fig. 2. QUORUM flow chart.

Table 1. Assessment Criteria

AC1	Is the paper based on research?
AC2	Is there a clear statement of the aims of the research?
AC3	Was the research design appropriate to address the aims of the research?
AC4	Was the data analysis sufficiently rigorous?
AC5	Is there a clear statement of findings?
AC6	Is the study of value for research or practice?

This query turned out to be extremely wide. For instance, ScienceDirect returned more than 6,000 documents, so we filtered the results by refining the search string:

((“template detection” OR “noise elimination”) AND web) OR “cleaning web page” [Title/abstract/keywords]

As a result of the search process, 278 studies were identified. Excluding unavailable, non-related to the topic, and duplicated results, we obtained 40 papers.

(3) *Inclusion and exclusion criteria.* To address the research questions, the following inclusion and exclusion criteria were defined:
- IC1: Those papers that use the DOM tree to represent the webpages.
- IC2: Those papers published in a conference rated with A in the *GGS Conference Rating*.[2]
- IC3: Those papers that describe a site-level technique.
- EC1: Those papers that have less than 15 cites.[3]

(4) *Quality assessment.* Each paper was evaluated using our own adaptation of a quality checklist that can be used across multiple study types proposed in Reference [22] (see Table 1).

[2]http://gii-grin-scie-rating.scie.es/.
[3]The number of cites was extracted from the corresponding editorial where the paper was published (e.g., ACM). If the number of cites was not available, then it was extracted from Google Scholar.

We used 6 quality assessment questions based on the original 11 questions proposed by the authors:

2.2 Search Results

After the primary screening process, we selected a total of 40 papers, and we excluded 238 from the nine sources. The QUORUM flow chart of the reviewing process is depicted in Figure 2. Table 2 shows the list of selected papers with the metadata used in the selection process. We read in depth the full text of each selected paper to decide whether to include this method in our study (and, thus, implement it). The papers that met the inclusion criteria and did not meet the exclusion criteria were selected. Finally, four papers were selected and, additionally, we included our template extraction method, called TemEx [1]. The final selection are the rows with grey background in Table 2. All of them are described below:

SST (2003) [57]: This algorithm introduces a data structure called *Site Style Tree* (SST) to represent a collection of webpages. The SST can be seen as the union of all DOM trees it represents. Each DOM node present in one webpage is represented in the SST in the same position, and moreover, brotherhood relations are kept, so that groups of sibling nodes are explicitly represented in the SST. Those groups of nodes repeated in more than one webpage are represented in the SST with a counter. Hence, the SST provides information about the repetition of groups of nodes. The most repeated groups of nodes are more likely to belong to the template (a threshold is used to select them). Evaluation experiments were carried out with five commercial websites producing an F1 of 75.1%. No information is provided about the measure unit used.

- *Main goal*: Removing noisy blocks from webpages. They call "noisy blocks" those that are not main content (navigation pans, advertisements, copyright and privacy notices, etc.).
- *Technologies used*: The authors do not provide any information about the programming language used, nor the different technologies or layouts accepted by the tool (the use of iframes, flash, javascript, etc.).
- *Benchmarks used in their evaluation*: Real webpages from five commercial websites: Amazon, CNet, PCMag, J&R, and ZDnet.
- *Limitations/Problems*: The main limitation of this technique is that it needs a large amount of webpages (authors used 500) to build the SST of each website. Moreover, the evaluation was done with homogeneous websites.

RTDM-TD (2006) [54]: This algorithm compares a set of webpages by identifying what parts of their DOM trees are exactly equal in all webpages. Those DOM nodes repeated in all webpages are the template. To compare a set of DOM trees, it uses a top-down variant of the tree edit distance (TED) algorithm. Roughly, they randomly pick two webpages and compute their TED. All mapped nodes represent the current template. Then, they iteratively compute the TED between the current template and another random webpage until a predefined number of webpages have been processed. Evaluation experiments were carried out with "a few dozen" of manually selected webpages. The F1 reported with 10 websites is "higher than 95%" in 9 out of 10 websites, and "above 85%" in the other one. It was calculated by counting the number of correctly retrieved words of the template.

- *Main goal*: Removing templates found in collections of webpages to enhance web IR and web mining methods. First, the algorithm detects the template of a small

Table 2. Selected Papers That Describe Web Template Extraction Tools

Article	Year	Venue	Core	LS	MA	JCR	Cites	DOM	P-L/S-L
[4]	2002	WWW	A++	A++	A++	-	387	No	S-L
[57]	2003	SIGKDD	A++	A++	A++	-	507	Yes	S-L
[54]	2006	CIKM	A	A++	A+	-	95	Yes	S-L
[33]	2006	CSCWD	B	B	C	-	4	No	S-L
[30]	2006	DEXA	B	-	B	-	25	Yes	S-L
[20]	2006	IKE	C	-	-	-	10	No	S-L
[7]	2006	SAC	B	-	-	-	71	Yes	S-L
[6]	2007	WWW	A++	A++	A++	-	120	Yes	P-L
[23]	2007	PKDD	A	A	A+	-	10	No*	S-L
[38]	2007	WAC5	-	-	-	-	37	No	P-L
[12]	2008	LREC	C	A	A	-	54	No	P-L
[5]	2008	LREC	C	A	A	-	111	No	N/A
[56]	2008	WWW	A++	A++	A++	-	23	Yes	S-L
[24]	2009	WWW	A++	A++	A++	-	34	No	P-L
[41]	2009	KSE	B	-	-	-	11	No	S-L
[53]	2009	WWW	A++	A++	A++	-	21	Yes	S-L
[25]	2010	WSDM	A++	A+	A+	-	484	No	P-L
[14]	2011	IC3K	C	C	-	-	1	No	S-L
[44]	2011	ICCSIT	-	-	C	-	1	Yes	P-L
[21]	2011	IEEE TKDE	-	-	-	Q1	69	No	S-L
[19]	2012	IJCSE	-	-	-	-	8	Yes	S-L
[35]	2012	IJACR	-	-	-	-	1	Yes	S-L
[3]	2013	Inf. sci.	-	-	-	Q1	15	Yes	S-L
[52]	2013	Inf. proc. & man.	-	-	-	Q1	41	No	P-L
[45]	2013	IJCA	-	-	-	-	4	No	P-L
[42]	2013	IJRTE	-	-	-	-	11	Yes	P-L
[36]	2013	ICGCE	-	-	-	-	2	Yes	S-L
[28]	2014	App. mech. & mat.	-	-	-	Q4	0	Yes	S-L
[48]	2014	Wir. per. comm.	-	-	-	Q3	6	No	P-L
[13]	2014	ICDM	A++	A++	A	-	2	Yes	S-L
[9]	2014	ICACNI	-	-	-	-	1	Yes	S-L
[10]	2014	ICACCI	-	-	-	-	9	No	S-L
[18]	2014	IJCA	-	-	-	-	3	Yes	S-L
[50]	2015	IJCA	-	-	-	-	3	Yes	S-L
[11]	2015	IJCA	-	-	-		6	No	P-L
[8]	2015	IJETCR	-	-	-	-	1	Yes	S-L
[1]	2015	PSI	B	C	C	-	3	Yes	S-L
[17]	2017	GJPAM	-	-	-	Q4	1	Yes	S-L
[51]	2018	Clus. comp.	-	-	-	Q2	0	No	P-L
[58]	2018	WISE	A	B	B	-	0	Yes	P-L

set of sample webpages. Then, it removes the derived template from the remaining pages in the collection.

- *Technologies used*: Not reported.
- *Benchmarks used in their evaluation*: Real webpages from 10 websites: the 5 websites used in Reference [57] (Amazon, CNet, PCMag, J&R, and ZDnet), and other 5 well-known websites (CNN, E-Jazz, Encyclopedia Mythica, UBL, and Wikipedia).
- *Limitations/Problems*: To achieve an F1 value near 95% they use 25 webpages. Computing a top-down mapping between the DOM trees of 25 webpages can take a long time depending on the amount of DOM nodes of the webpages. Furthermore, the evaluation was done with only 10 websites.

IWPTD (2008) [56]: This algorithm divides the DOM trees of the webpages into several subtrees whose roots are the DOM nodes associated with concrete HTML tags (i.e., DIV, TABLE, UL, etc.). Then, it compares the text segments (DOM nodes of type `#text`) inside the subtrees of all the webpages. If a text segment appears in five or more webpages, then it is considered as a template segment. Finally, a subtree is considered a template subtree if the ratio between the length of all its template segments and all its text segments is higher than 0.7. Evaluation experiments were carried out with five websites and 400 random webpages for each website, producing a precision of 98% and a recall of 80%. The article does not describe how these measures are computed.

- *Main goal*: Crawling. Implementation of an incremental framework to detect templates in which a page is processed as soon as it has been crawled.
- *Technologies used*: Not reported.
- *Benchmarks used in their evaluation*: Authors selected five webpages from the dataset used in Reference [34]. They do not provide any information about which webpages were selected.
- *Limitations/Problems*: The evaluation was done with only five websites.

RBM-TD (2009) [53]: This approach is similar to RTDM-TD, but they use a bottom-up variant of the TED algorithm (instead of the top-down variant). During the comparison of DOM trees, it introduces a restriction to classify a common subtree as template: those subtrees repeated in all webpages must be exactly in the same position (i.e., the path from the root to them must be the same in all webpages). Evaluation experiments were carried out with 24 manually selected webpages and ensuring that all of them implement the same template. The F1 reported with 10 websites is close to 90%. It was calculated by counting the number of correctly retrieved words of the template.

- *Main goal*: Template removal to improve webpage indexing and processing.
- *Technologies used*: Not reported.
- *Benchmarks used in their evaluation*: The same 10 real websites used in [54]: Amazon, CNet, PCMag, J&R, ZDnet, CNN, E-Jazz, Encyclopedia Mythica, UBL, and Wikipedia.
- *Limitations/Problems*: The main limitation is that the evaluation was done with only 10 websites, and also the 24 webpages used were not randomly selected, but all of them implementing the template (this scenario is easier for a template extractor).

TemEx (2015) [2]: As in RTDM-TD and RBM-TD, this algorithm also uses a mapping between the DOM trees to determine what nodes belong to all webpages, but it does not force a node to belong to all webpages, only to a subset, as in SST. In this respect, this algorithm is more democratic, because it uses a number of votes to determine that a node has been

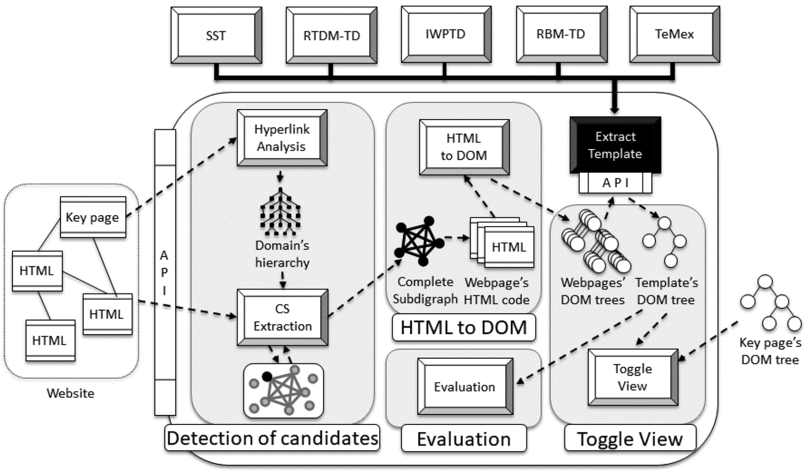

Fig. 3. Workbench architecture.

repeated in enough webpages to be considered as part of the template. This provides a degree of tolerance, because we can classify a set of DOM nodes as part of the template even if there exist webpages that do not contains these nodes. This can in some cases reduce precision, but it often raises recall (which is the main problem of all techniques). The F1 reported with 75 websites is 86.94%. It was calculated by counting the number of correctly retrieved nodes of the template.

- *Main goal*: Reusing templates by website developers.
- *Technologies used*: Implementation done with Javascript and distributed as a Firefox add-on.
- *Benchmarks used in their evaluation*: 100 real heterogeneous websites.
- *Limitations/Problems*: The execution time of the mapping is the bottleneck.

It is worth noting that the F1 values reported by each tool are incomparable, because they were evaluated with different evaluation sets. In Section 4, we provide a fair comparison of these techniques.

3 A WORKBENCH FOR TEMPLATE EXTRACTION

Our implementation is really a workbench with several features for template extraction. It is available at http://www.dsic.upv.es/~jsilva/retrieval/Web-TemEx/.

What is interesting for template extraction developers is that the workbench is parametric with respect to the algorithm used for template extraction, and it provides an API so that a template extraction algorithm can access all the workbench resources. We describe our workbench by using the architecture shown in Figure 3.

In this figure, each module is represented with a 3D white box, and the input and output of the module is connected to it with dashed arrows. For instance, the `Hyperlink Analysis` module inputs the URL of a website, and it outputs a data structure representing the domain's hierarchy.

All modules have been organized in four grey areas that implement a particular functionality. The areas are: `Detection of candidates`, `HTML to DOM`, `Toggle View`, and `Evaluation`. There is one module coloured in black that implements the template extraction functionality. This black module should be replaced with one concrete algorithm such as one of the five already implemented and drawn at the top of the figure. Each grey area is described separately:

Detection of candidates: In the general case, a template extractor inputs a webpage and it outputs its template by comparing the webpage with other webpages that implement the same template. Therefore, the template extraction process should be divided into two different phases: (1) Exploring the website to identify webpage candidates that probably implement the template, and (2) comparing the webpage with the candidates.

Unfortunately, none of the described algorithms implement the first phase. In their experiments, candidates were taken randomly from the website (IWPTD [56]), randomly from a predefined set of webpages, all of them implementing the same template (SST [57]), or manually; again, all of them implementing the same template (RBM-TD [54], RTDM-TD [53]).

Contrarily, our workbench uses a hyperlink analysis to automatically identify webpages that very likely implement the web template. Compared to the random selection, this analysis highly improves the recall and precision of the second phase (for all algorithms), and compared to the manual selection, it allows the system to work online (it just needs a URL to automatically explore the website following links). Moreover, this phase significantly improves performance by reducing the number of webpage candidates needed to find the template. According to our experiments (see Section 4), collecting the webpage candidates takes less than 100ms in 90.67% of benchmarks. What is especially interesting is that it is orthogonal to the second phase, and thus it can be used by all algorithms, thus the workbench performs this phase for all template extraction algorithms.

HTML to DOM: One important problem to compare the results reported from the different tools was that they were evaluated with different criteria, or the evaluation criterion was not reported [56, 57]. Traditionally, the papers in the web template extraction literature measure how good the template extractor is with two measures: text or DOM nodes retrieved. The precision can be very different if we measure the number of extracted template characters, the number of words, the number of HTML tags, and so on. Therefore, the evaluation criterion must be fixed to compare the tools.

Those papers that use text in their measures (precision, recall ...), can be further classified: those that measure paragraphs, those that measure words, and those that measure characters. In principle, one could think that words are better than paragraphs and characters are better than words, because, e.g., by measuring words, one can detect that a word that should have been retrieved is missing in a paragraph, but this would not be possible by measuring paragraphs (the whole paragraph is marked as retrieved or not). The same happens with words and characters.

However, in general, it does not make sense to only extract a subset of the characters in a word. Moreover, measuring retrieved characters is strongly dependent on how many characters have a word. That is, if a template word "computer" is not retrieved, this would produce a penalty higher than if a word "hello" is not retrieved. Simply because it has more characters. This may distort the measures.

All these problems related to text are often solved by using DOM nodes instead, because each individual block of text appearing in the HTML source code is represented with a DOM node of type `#text`. Therefore, this measure is specially interesting for template extraction, because templates reuse HTML labels and their blocks of text. It is extremely rare to find a template that only reuses a portion of a text in a DOM node. For these reasons, DOM nodes are more appropriate to evaluate template extractors. Therefore, the workbench includes a module to automatically transform every webpage to its associated DOM tree.

Toggle View: Once the template has been extracted, the workbench allows for graphically showing it directly on the browser and toggling the view by swapping from the original webpage to the extracted template and vice versa (in Figure 1, the "Toggle View" button is the pink one located in the top right toolbox. Clicking this button makes the browser to swap between the webpages on the left and right).

Evaluation: The workbench includes a module to evaluate the produced template (precision, recall, F1, and extraction time). The algorithms only have to generate a template, and the workbench compares this template with the gold standard and generates a report with the results. For instance, the confidence intervals in Tables 4 and 5 (they will be explained in Section 4.2) were generated from the information obtained by the workbench.

4 EMPIRICAL EVALUATION

The empirical evaluation of template extractors can be done with our workbench, because it has been extended with evaluation features (e.g., it can monitor the time or memory used). Therefore, it allows for comparing all algorithms: implemented in the same language, using the same webpage candidates and the same number of them,[4] and with the same evaluation criterion. Of course, we should also use the same benchmarks. Nevertheless, there does not exist a benchmark suite for template extraction.

Each described system uses a different benchmark suite. Unfortunately, these benchmarks are not usable for comparison. The reason is that their webpages are in some cases artificial (generated ad hoc to use the same template); but even if they are real websites, they are not heterogenous (e.g., in Reference [54] all benchmarks were product description webpages) or outdated (i.e., using old versions of HTML, frames, etc.). We consider that our comparison must be done with real online current webpages that use heterogeneous technologies (`<table>`-based designs, `<div>`-based designs, CSS, HTML 5, Javascript, etc.). Therefore, we produced a new benchmark suite for template extraction. This benchmark suite was constructed before we implemented and compared the template extractors.

4.1 The Benchmark Suite

We produced a dataset of 100 real and heterogeneous webpages (ikea.com, bbc.com, etc.) with different layouts and page structures, including different languages (English, German, French, and Spanish) to allow for the testing of language-independent features. Table 3 shows the benchmarks classified into five categories.

The suite was constructed following this procedure:

(1) We selected from the CleanEval suite those webpages that are still maintained, thus, obsolete benchmarks were discarded. 20 benchmarks come from CleanEval.
(2) We asked several researchers to select real webpages for us (hence, the engineers that extracted the template did not select the benchmarks). Around 150 webpages were selected.
(3) From that set of webpages, we randomly selected a subset of 80 benchmarks.
(4) For each webpage (considered as the *key page*),
    (a) we followed its links and collected all webpages in a maximum distance of three clicks, thus forming a net where the center is the key page. This is useful for both page-level

[4]Phase 1 in our algorithm (i.e., detecting the webpage candidates to extract the template from them) has been integrated into the workbench. Therefore, even though the other techniques select the candidates randomly (thus, producing worse results on average), we made the comparison of all the techniques by sharing for all algorithms exactly the same candidates produced by the workbench. Hence, the comparison is completely fair, because they all use the same webpages to compute the template.

Table 3. Benchmark Classification

Category	Number of pages
Companies/Shops	29
Forums/Social	16
Personal websites/Blogs	17
Media/Communication	17
Institutions/Associations	21

and site-level techniques: page-level techniques only use the key page, and site-level techniques use the key page and a set of webpages accessible from it. Collecting webpages not accessible from the key page or further than three clicks would be useless for template extraction.

(b) Every single HTML tag of the webpages was manually classified as *mainContent*, *template*, *notContent*, and/or *notTemplate*. Therefore, the suite is useful for both content and template extraction. This labelling was implemented with HTML classes so that it can be used by any technique (not only DOM-based techniques). For instance, given a main content node, it was labelled by including it in an HTML class called *mainContent*, so the node and all its descendants are marked as main content nodes. Moreover, this labelling does not influence the original aspect or structure of the webpages, and it is automatically processable. Finally, the webpages and all their resources (images, media, CSS, Javascript, etc.) were localized so that all links reference their local copy to ensure independency of the benchmark with respect to the evolution of the online websites.

(c) This process was done by three engineers, that later put their results in common to avoid errors and produce the final labelling.

With this suite, researchers can evaluate or compare their technique very easily thanks to the labelling and because the suite also includes scripts to automate the analysis of webpages. It is open-source, free, and publicly available at http://www.dsic.upv.es/~jsilva/retrieval/teco/.

4.2 Evaluation of Tools

Our evaluation dataset is composed of 75 benchmarks selected randomly from the benchmark suite. We extracted the template from the evaluation dataset using all algorithms, and we compared their efficacy and performance. To evaluate the efficacy of the methods, we use the recall, the precision, the F1 measure, and the accuracy. The use of these measures is explained and justified in Section 8.3 of Reference [37].

Recall, precision, F1, and accuracy. For each algorithm, we computed the 0.99 confidence interval across the arithmetic mean of the recall, precision, F1, and accuracy from the different 75 benchmarks. In all the experiments, the evaluation unit was the DOM node. Therefore, `Recall` shows the number of correctly retrieved nodes divided by the number of nodes in the gold standard; `Precision` shows the number of correctly retrieved nodes divided by the number of retrieved nodes; `F1` shows the F1 metric; and `accuracy` shows the accuracy metric. The results obtained are summarized in Table 4. In the table, we use the notation $[_a\, b\, _c]$ that represents a symmetric 0.99 confidence interval between a and c with centre in b.

Several conclusions can be extracted from Table 4. First, we see that SST is strictly worst than IWPTD, and IWPTD is, in turn, strictly worst than TemEx. Similarly, RTDM-TD is strictly worst

Table 4. Comparison of Template Extraction Tools

Benchmark	Recall		Precision	
	$\overline{X}, 1-\alpha = 99\%$	σ	$\overline{X}, 1-\alpha = 99\%$	σ
SST	$[_{40.69}\ 44.80\ _{48.92}]$	13.83	$[_{46.71}\ 54, 57\ _{62.43}]$	26.42
RTDM-TD	$[_{5.96}\ 15.67\ _{25.38}]$	32.65	$[_{95.43}\ 97.86\ _{100.00}]$	8.17
IWPTD	$[_{67.16}\ 73.31\ _{79.47}]$	20.70	$[_{60.20}\ 68.18\ _{76.17}]$	26.84
RBM-TD	$[_{29.13}\ 41.12\ _{53.11}]$	40.31	$\mathbf{[_{100.00}\ 100.00\ _{100.00}]}$	**0.00**
TemEx	$\mathbf{[_{86.78}\ 91.46\ _{96.15}]}$	**15.76**	$[_{84.82}\ 89.47\ _{94.12}]$	15.63

Benchmark	F1		Accuracy	
	$\overline{X}, 1-\alpha = 99\%$	σ	$\overline{X}, 1-\alpha = 99\%$	σ
SST	$[_{40.12}\ 45.21\ _{50.30}]$	17.10	$[_{41.81}\ 45.58\ _{49.36}]$	12.69
RTDM-TD	$[_{7.37}\ 16.89\ _{26.42}]$	32.02	$[_{51.59}\ 55.78\ _{59.96}]$	14.07
IWPTD	$[_{58.97}\ 66.03\ _{73.08}]$	23.72	$[_{60.95}\ 65.08\ _{69.22}]$	13.89
RBM-TD	$[_{33.32}\ 46.18\ _{59.04}]$	43.24	$[_{62.04}\ 67.46\ _{72.89}]$	18.25
TemEx	$\mathbf{[_{84.38}\ 88.46\ _{92.54}]}$	**13.71**	$\mathbf{[_{82.10}\ 85.87\ _{89.63}]}$	**12.67**

than RBM-TD. TemEx produced the best recall, accuracy, and F1; while RBM-TD produced the best precision. In fact, the precision of RBM-TD is 100%. Thus, this algorithm should be selected for those applications where precision must be maximized (e.g., in template removal one can use RBM-TD with a high confidence that no content will be removed). Nevertheless, RBM-TD is not the best option when an application needs to maximize the amount of template that should be detected. In this case, TemEx should be selected, because it showed the best recall. TemEx also produced the best F1, thus it has the best balance between precision and recall.

The range of the confidence intervals reveals that all methods can be strongly influenced by the concrete analyzed website. This is quantified by the standard deviation computed for each algorithm, which is between 14% and 43%. Websites that implement several templates or websites that do not implement any template usually obtain the lowest metric values.

We observed that some of the template extractors are sensible to the type of webpages from which they extract the template. This is logical, because webpages of a particular type (e.g., forums) often share some specific features. For instance, "forum" webpages usually contain a lot of text; in "shops" there usually exist many images; in "blogs" many webpages have been generated with a content management system (CMS) and, thus, the template is often rigid; and so on. To study this phenomenon, we analyzed the performance of the algorithms with different types of webpages following the classification in Table 3. This is shown in Table 5, which presents the mean F1 of the different benchmark categories for each algorithm. On the one hand, RTDM-TD and RBM-TD vary significantly depending on the evaluated benchmark category. On the other hand, IWPTD and SST are fairly constant and their F1 does not seem to depend on the benchmark category. The highest value in the table corresponds to TemEx in the "Institutions/Associations" category. In this particular category, TemEx is about 91% F1. Contrarily, in category "Media/Communication" the F1 is about 78%. The corresponding results for accuracy are shown in Table 6.

Computation time. Regarding the computation time, the webpage loading, hyperlink analysis, CS extraction, and HTML to DOM times are the same in all tools, because they use the same workbench. The algorithm computation time is completely different in all tools. While IWPTD and TemEx are the quickest algorithms, the others are significantly slower. Table 7 shows the mean of the execution time of each algorithm for the 75 evaluation benchmarks. IWPTD is extremely quick, it only takes an average of about 2s per benchmark. TemEx is not as quick as IWPTD, it

Table 5. Benchmark Groups F1 Comparison

Category	RTDM-TD		RBM-TD		IWPTD	
	$\overline{X}, 1-\alpha = 99\%$	σ	$\overline{X}, 1-\alpha = 99\%$	σ	$\overline{X}, 1-\alpha = 99\%$	σ
Companies/Shops	[$_{0.00}$ 16.95 $_{33.96}$]	32.36	[$_{28.17}$ 51.50 $_{74.82}$]	44.36	[$_{\mathbf{65.17}}$ **73.95** $_{\mathbf{82.73}}$]	**16.70**
Forums/Social	[$_{\mathbf{5.08}}$ **41.72** $_{\mathbf{78.36}}$]	**47.18**	[$_{9.88}$ 43.91 $_{77.93}$]	43.81	[$_{18.39}$ 44.66 $_{70.94}$]	33.84
Personal websites/Blogs	[$_{0.00}$ 20.57 $_{48.11}$]	37.04	[$_{\mathbf{27.25}}$ **60.57** $_{\mathbf{93.90}}$]	**44.81**	[$_{46.75}$ 63.50 $_{80.24}$]	22.52
Media/Communication	[$_{0.00}$ 5.70 $_{18.22}$]	16.84	[$_{0.00}$ 28.71 $_{60.24}$]	42.41	[$_{60.96}$ 71.02 $_{81.08}$]	13.53
Institutions/Associations	[$_{0.81}$ 5.38 $_{9.94}$]	7.09	[$_{15.85}$ 42.08 $_{68.31}$]	40.73	[$_{51.00}$ 67.00 $_{83.00}$]	24.85

Category	TemEx		SST	
	$\overline{X}, 1-\alpha = 99\%$	σ	$\overline{X}, 1-\alpha = 99\%$	σ
Companies/Shops	[$_{81.28}$ 88.52 $_{95.77}$]	13.78	[$_{\mathbf{41.22}}$ **48.98** $_{\mathbf{56.74}}$]	**14.76**
Forums/Social	[$_{83.14}$ 92.82 $_{100.00}$]	12.47	[$_{21.62}$ 40.15 $_{58.67}$]	23.85
Personal websites/Blogs	[$_{75.34}$ 87.09 $_{98.84}$]	15.80	[$_{33.19}$ 42.10 $_{51.01}$]	11.99
Media/Communication	[$_{67.32}$ 79.53 $_{91.74}$]	16.42	[$_{32.83}$ 43.96 $_{55.08}$]	14.96
Institutions/Associations	[$_{\mathbf{88.58}}$ **93.09** $_{\mathbf{97.59}}$]	**6.99**	[$_{33.30}$ 46.31 $_{59.32}$]	20.21

Table 6. Benchmark Groups Accuracy Comparison

Category	RTDM-TD		RBM-TD		IWPTD	
	$\overline{X}, 1-\alpha = 99\%$	σ	$\overline{X}, 1-\alpha = 99\%$	σ	$\overline{X}, 1-\alpha = 99\%$	σ
Companies/Shops	[$_{48.14}$ 55.22 $_{62.31}$]	13.48	[$_{60.13}$ 69.86 $_{79.59}$]	18.50	[$_{\mathbf{60.80}}$ **68.04** $_{\mathbf{75.29}}$]	**13.78**
Forums/Social	[$_{\mathbf{50.75}}$ **67.31** $_{\mathbf{83.87}}$]	**21.33**	[$_{51.83}$ 66.04 $_{80.26}$]	18.30	[$_{44.52}$ 57.42 $_{70.32}$]	16.61
Personal websites/Blogs	[$_{44.99}$ 58.43 $_{71.87}$]	18.08	[$_{\mathbf{59.75}}$ **75.22** $_{\mathbf{90.69}}$]	**20.81**	[$_{55.61}$ 63.88 $_{72.14}$]	11.11
Media/Communication	[$_{49.69}$ 50.37 $_{51.04}$]	0.91	[$_{47.98}$ 60.76 $_{73.54}$]	17.19	[$_{56.24}$ 64.94 $_{73.65}$]	11.71
Institutions/Associations	[$_{50.07}$ 50.76 $_{51.45}$]	1.07	[$_{53.79}$ 64.06 $_{74.34}$]	15.96	[$_{57.30}$ 66.93 $_{76.56}$]	14.95

Category	TemEx		SST	
	$\overline{X}, 1-\alpha = 99\%$	σ	$\overline{X}, 1-\alpha = 99\%$	σ
Companies/Shops	[$_{78.57}$ 85.34 $_{92.11}$]	12.88	[$_{40.93}$ 47.63 $_{54.33}$]	12.74
Forums/Social	[$_{\mathbf{82.03}}$ **91.60** $_{\mathbf{100.00}}$]	**12.32**	[$_{32.13}$ 45.79 $_{59.44}$]	17.58
Personal websites/Blogs	[$_{75.47}$ 85.17 $_{94.87}$]	13.04	[$_{32.56}$ 40.84 $_{49.12}$]	11.14
Media/Communication	[$_{62.28}$ 78.18 $_{88.08}$]	13.31	[$_{34.27}$ 40.60 $_{46.93}$]	8.51
Institutions/Associations	[$_{82.60}$ 89.00 $_{95.40}$]	9.94	[$_{\mathbf{42.26}}$ **49.68** $_{\mathbf{57.09}}$]	**11.51**

takes an average of about 42s for each benchmark. However, 73% of the benchmarks take less than 10s. SST, RBM-TD and RTDM-TD are relatively slow algorithms, their computation time is significantly higher, and above 1min.

Scalability. Scalability has been empirically evaluated by measuring the evolution of the computation time with regards to the growth of the DOM trees (i.e., their number of nodes). Here again, IWPTD and TemEx are significantly better than SST, RBM-TD, and RTDM-TD. This is shown in Figure 4, which draws the computation time trendline of each algorithm with respect to the number of nodes of the key page. While the trendlines of RTDM-TD, RBM-TD, and especially SST are quadratic, the trendlines of IWPTD and TemEx are linear with a gentle incline. Therefore, in terms of scalability, IWPTD and TemEx are better than the other algorithms.

Table 7. Benchmark Classification

Phase	$\overline{X}, 1-\alpha = 99\%$	σ
Webpage Loading	$[_{0.00}\ 1309.21\ _{4000.40}]$ms	9048.11ms
Hyperlink Analysis	$[_{0.00}\ 91.69\ _{203.68}]$ms	376.50ms
CS Extraction	$[_{0.00}\ 617.55\ _{2164.86}]$ms	5202.26ms
HTML to DOM	$[_{0.00}\ 1403.52\ _{4084.52}]$ms	9013.86ms

Template extractor	$\overline{X}, 1-\alpha = 99\%$	σ
SST	$[_{63.58}\ 144.25\ _{224.93}]$s	271.24s
RTDM-TD	$[_{39.37}\ 127.34\ _{215.31}]$s	295.77s
IWPTD	$[_{\mathbf{1.04}}\ \mathbf{2.22}\ _{\mathbf{3.40}}]$s	**3.98s**
RBM-TD	$[_{17.31}\ 116.70\ _{216.09}]$s	334.15s
TemEx	$[_{0.00}\ 42.83\ _{89.38}]$s	156.50s

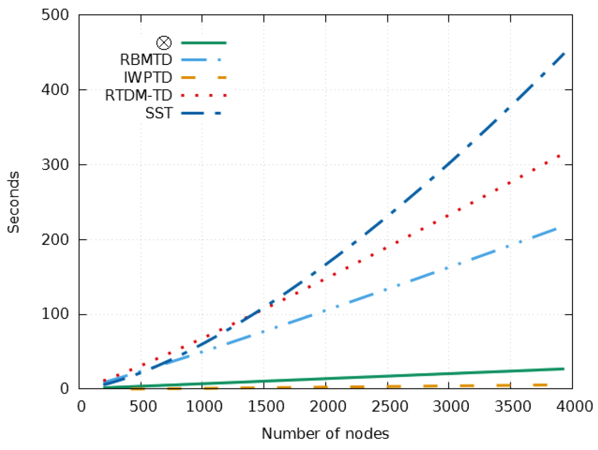

Fig. 4. Time trendlines associated with DOM sizes.

Asymptotic costs. Thanks to the fact that we reimplemented the algorithms, we can also study their scalability from a theoretical perspective. Concretely, we analyzed their asymptotic costs based on their source codes. The following measures were obtained:

- SST $\in O(W * (n^2 + T))$, n being the number of nodes in the key page, W the maximum width of the Site Style Tree, and T the size of the Site Style Tree.
- RTDM-TD $\in O(n^2)$, n being the number of nodes in the key page.
- IWPTD $\in O(n + T \log T)$, n being the number of nodes in the key page and T the number of text segments obtained.
- RBM-TD $\in O(n^2)$, n being the number of nodes in the key page.
- TemEx $\in O(n * W)$, n being the number of nodes in the key page and W the maximum width of the DOM tree.

The asymptotic cost analysis confirms the empirical results. IWPTD and TemEx have linear growth while RTDM-TD, RBM-TD, and SST have quadratic growth. The cost of IWPTD is the best. It is practically $O(n)$, because the number of text segments is always significantly lower than the number of nodes. TemEx also has a linear growth. The other three algorithms (RTDM-TD, RBM-TD, and SST) have a quadratic cost, RTDM-TD and RBM-TD being asymptotic cost $O(n^2)$, which is better than the cost of SST.

5 CONCLUSIONS

This work presents a comparison of template extractors in terms of precision, recall, F1, accuracy, efficiency, and scalability. The motivation of this work is that current template extractors were incomparable from an empirical point of view due to the different benchmarks and measures used in the articles/webpages/reports where they were described. To compare them, we reimplemented from scratch the five most important tools in the bibliography. For the comparison, we implemented a workbench with all the necessary infrastructure to load, analyze, and process online webpages, so that all algorithms can work now online and with modern webpages.

The workbench includes a new analysis able to automatically select the webpage candidates to be processed by the template extractors. This is especially useful, because it allows the process to be completely automatic (the workbench itself navigates the website to find the candidates). Moreover, because it is independent of the selected algorithm, all present and future template extractors can use this phase. In addition, we produced a collection of benchmarks to perform the comparison of all tools with the same suite (our study revealed that no current benchmark suite was adequate, modern, and fair). After the comparison, we have an empirical evidence about the precision and recall of all tools, and also about their runtimes and scalability. TemEx produced the best average recall, accuracy, and F1. Meanwhile, the RBM-TD algorithm produced the best precision. The scalabilities of the tools are very different. IWPTD is the most scalable algorithm with a linear growth. TemEx also showed a linear growth, while the other algorithms showed to be quadratic.

In consequence, we now have enough information to decide which template extractor to use according to our specific needs: If time is crucial, e.g., because we want no analyze thousands or millions of webpages, then IWPTD is the best candidate due to its linear growth. However, this algorithm has a F1 of 66%. If time is not crucial, then TemEx should be selected, because its scalability is similar but it has a F1 of more than 88%. If time is not crucial and we are interested in retrieving only template elements (this is useful in filtering and boilerplate removal techniques), then RBM-TD should be selected, because it has the best precision (100% with all the benchmarks of our suite). In any other case, i.e., if recall, accuracy, or F1 should be maximized, then TemEx is the best option (91.46% recall and 88.46% F1).

The comparison of the tools has been also performed considering different types of websites (forums, companies, etc.), so that one can decide which template extractor to use depending on the type of website considered. This has also been done in the design of TemEx, so that the internal ponderations used by TemEx can be also parameterized for a specific kind of website. The design and evaluation of TemEx has produced several side results such as an empirical evaluation of how different are the templates of two webpages when they are pointed or not from the main menu, when they have a specific distance in the directory tree of the website, and when they are pointed from links located at a specific distance in the key page's DOM tree.

REFERENCES

[1] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. 2015. TeMex: The web template extractor. In *Proceedings of the 24th International Conference on World Wide Web (WWW'15 Companion)*. ACM, New York, NY, 155–158. DOI: https://doi.org/10.1145/2740908.2742835

[2] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. 2016. *Site-Level Web Template Extraction Based on DOM Analysis*. Springer International Publishing, Cham, 36–49. DOI: https://doi.org/10.1007/978-3-319-41579-6_4

[3] Derar Alassi and Reda Alhajj. 2013. Effectiveness of template detection on noise reduction and websites summarization. *Info. Sci.* 219 (2013), 41–72. DOI: https://doi.org/10.1016/j.ins.2012.07.022

[4] Ziv Bar-Yossef and Sridhar Rajagopalan. 2002. Template detection via data mining and its applications. In *Proceedings of the 11th International Conference on World Wide Web (WWW'02)*. ACM, New York, NY, 580–591. DOI: https://doi.org/10.1145/511446.511522

[5] Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. 2008. Cleaneval: A competition for cleaning web pages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC'08)*.

[6] Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. 2007. Page-level template detection via isotonic smoothing. In *Proceedings of the 16th International Conference on World Wide Web (WWW'07)*. ACM, New York, NY, 61–70. DOI: https://doi.org/10.1145/1242572.1242582

[7] Liang Chen, Shaozhi Ye, and Xing Li. 2006. Template detection for large scale search engines. In *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC'06)*. ACM, New York, NY, 1094–1098. DOI: https://doi.org/10.1145/1141277.1141534

[8] Nirmala Devi et al. 2015. Noisy elimination for web mining based on style tree approach. *Int. J. Engineer. Technol. Comput. Res.* 3, 2 (2015).

[9] Amit Dutta, Sudipta Paria, Tanmoy Golui, and Dipak Kumar Kole. 2014. Noise elimination from web page based on regular expressions for web content mining. In *Advanced Computing, Networking and Informatics—Volume 1*, Malay Kumar Kundu, Durga Prasad Mohapatra, Amit Konar, and Aruna Chakraborty (Eds.). Springer International Publishing, Cham, 545–554.

[10] A. Dutta, S. Paria, T. Golui, and D. K. Kole. 2014. Structural analysis and regular expressions based noise elimination from web pages for web content mining. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI'14)*. 1445–1451. DOI: https://doi.org/10.1109/ICACCI.2014.6968377

[11] Hassan F. Eldirdiery and A. H. Ahmed. 2015. Detecting and removing noisy data on web document using text density approach. *Int. J. Comput. Appl.* 112, 5 (2015).

[12] Stefan Evert. 2008. A lightweight and efficient tool for cleaning web pages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association. Retrieved from http://www.lrec-conf.org/proceedings/lrec2008/summaries/885.html.

[13] Bo Gao and Qifeng Fan. 2014. Multiple template detection based on segments. In *Advances in Data Mining. Applications and Theoretical Aspects*, Petra Perner (Ed.). Springer International Publishing, Cham, 24–38.

[14] Filippo Geraci and Marco Maggini. 2011. A multi-sequence alignment algorithm for web template detection. In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval (KDIR'11)*. 121–128.

[15] David Gibson, Kunal Punera, and Andrew Tomkins. 2005. The volume and evolution of web page templates. In *Proceedings of the 14th International Conference on World Wide Web (WWW'05)*, Allan Ellis and Tatsuya Hagino (Eds.). ACM, 830–839. DOI: https://doi.org/10.1145/1062745.1062763

[16] Christian Girardi. 2007. Htmcleaner: Extracting the relevant text from the web pages. In *Proceedings of the 3rd Web as Corpus Workshop, Incorporating Cleaneval: Building and Exploring Web Corpora (WAC'07)*, Vol. 4. 141–143.

[17] Gaurav Gupta and Indu Chhabra. 2017. Optimized template detection and extraction algorithm for web scraping of dynamic web pages. *Global J. Pure Appl. Math.* 13, 2 (2017), 719–732.

[18] Kulkarni A. H. and Patil B. M. 2014. Article: Template extraction from heterogeneous web pages with cosine similarity. *Int. J. Comput. Appl.* 87, 3 (Feb. 2014), 4–8.

[19] Vidya Kadam and Prakash R. Devale. 2012. A methodology for template extraction from heterogeneous web pages. *Indian J. Comput. Sci. Engineer.* 3, 3 (2012).

[20] Byeong Ho Kang and Yang Sok Kim. 2006. Noise elimination from the web documents by using URL paths and information redundancy. In *Proceedings of the International Conference on Information and Knowledge Engineering (IKE'06)*.

[21] C. Kim and K. Shim. 2011. TEXT: Automatic template extraction from heterogeneous web pages. *IEEE Trans. Knowl. Data Engineer.* 23, 4 (Apr. 2011), 612–626. DOI: https://doi.org/10.1109/TKDE.2010.140

[22] Barbara Ann Kitchenham, David Budgen, and Pearl Brereton. 2015. *Evidence-Based Software Engineering and Systematic Reviews*. Chapman & Hall/CRC.

[23] Aleksander Kocz and Wen-tau Yih. 2007. Site-independent template-block detection. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 152–163.

[24] Christian Kohlschütter. 2009. A densitometric analysis of web template content. In *Proceedings of the 18th International Conference on World Wide Web*. ACM, 1165–1166.

[25] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the 3rd International Conference on Web Search and Web Data Mining (WSDM'10)*, Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu (Eds.). ACM, 441–450. DOI: https://doi.org/10.1145/1718487.1718542

[26] A. H. Kulkarni and B. M. Patil. 2014. Template extraction from heterogeneous web pages with cosine similarity. *Int. J. Comput. Appl.* 87, 3 (2014).

[27] Nicholas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. 1997. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, Martha E. Pollack (Ed.). Morgan Kaufmann, 729–737.

[28] Xiao Yan Le. 2014. A web text de-noising algorithm based on machine learning. In *Applied Mechanics and Materials*, Vol. 536. Trans Tech Publications, 516–519.

[29] Kristina Lerman, Steven N. Minton, and Craig A. Knoblock. 2003. Wrapper maintenance: A machine learning approach. *J. Artific. Intell. Res.* 18 (2003), 149–181.

[30] Jing Li and C. I. Ezeife. 2006. Cleaning web pages for effective web content mining. In *Database and Expert Systems Applications*, Stéphane Bressan, Josef Küng, and Roland Wagner (Eds.). Springer, Berlin, 560–571.

[31] Bing Liu. 2006. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications).* Springer-Verlag New York, Inc., Secaucus, NJ.

[32] Ling Liu, Wei Han, David Buttler, Calton Pu, and Wei Tang. 1999. An XJML-based wrapper generator for web information extraction. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'99).* ACM, New York, NY, 540–543. DOI: https://doi.org/10.1145/304182.304570

[33] L. Lo, V. T. Ng, P. Ng, and S. C. Chan. 2006. Automatic template detection for structured web pages. In *Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design.* 1–6. DOI: https://doi.org/10.1109/CSCWD.2006.253257

[34] Ling Ma, Nazli Goharian, Abdur Chowdhury, and Misun Chung. 2003. Extracting unstructured data from template generated web documents. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM'03).* ACM, New York, NY, 512–515. DOI: https://doi.org/10.1145/956863.956961

[35] Trupti B. Mane and Girish P. Potdar. 2012. Template extraction from heterogeneous web pages. *Int. J. Adv. Comput. Res.* 2, 4 (2012), 197.

[36] R Manjula and A Chilambuchelvan. 2013. Extracting templates from web pages. In *Proceedings of the International Conference on Green Computing, Communication and Conservation of Energy (ICGCE'13).* IEEE, 788–791.

[37] Christopher D. Manning, Prabhakar Raghavan, and Hinrich SchÃijtze. 2008. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY.

[38] Michal Marek, Pavel Pecina, and Miroslav Spousta. 2007. Web page cleaning with conditional random fields. In *Proceedings of the 5h Web as Corpus Workshop, Incorporationg CleanEval: Building and Exploring Web Corpora (WAC'07).* 155–162.

[39] Xiaofeng Meng, Dongdong Hu, and Chen Li. 2003. Schema-guided wrapper maintenance for web-data extraction. In *Proceedings of the 5th ACM International Workshop on Web Information and Data Management (WIDM'03).* ACM, New York, NY, 1–8. DOI: https://doi.org/10.1145/956699.956701

[40] Ion Muslea, Steven Minton, and Craig A. Knoblock. 2003. Wrapper Induction by Hierarchical Data Analysis. U.S. Patent 6,606,625.

[41] Dat Quoc Nguyen, Dai Quoc Nguyen, Son Bao Pham, and The Duy Bui. 2009. A fast template-based approach to automatically identify primary text content of a web page. In *Proceedings of the International Conference on Knowledge and Systems Engineering (KSE'09).* IEEE, 232–236.

[42] Alpa K. Oza and Shailendra Mishra. 2013. Elimination of noisy information from web pages. *Int. J. Recent Technol. Engineer.* 2, 1 (2013), 115–117.

[43] A. Pouramini and S. Nasiri. 2015. Web content extraction using contextual rules. In *Proceedings of the 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI'15).* 1014–1018. DOI: https://doi.org/10.1109/KBEI.2015.7436183

[44] Xin Qi and JianPeng Sun. 2011. Eliminating noisy information in webpage through heuristic rules. In *Proceedings of the International Conference on Computer Science and Information Technology.*

[45] Neeraj Raheja and V. K. Katiyar. 2013. A noise reduction approach based on NX 1 table and XSL display method for efficient web data extraction. *Int. J. Comput. Appl.* 64, 11 (2013).

[46] Pan Ei San. 2014. Boilerplate removal and content extraction from dynamic web pages. *Int. J. Comput. Sci. Engineer. Appl.* 4, 6 (2014), 27.

[47] Roland Schäfer. 2017. Accurate and efficient general-purpose boilerplate detection for crawled web corpora. *Lang. Resources Eval.* 51, 3 (Sep. 2017), 873–889. DOI: https://doi.org/10.1007/s10579-016-9359-2

[48] P. Sivakumar. 2015. Effectual web content mining using noise removal from web pages. *Wireless Personal Commun.* 84, 1 (Sep. 2015), 99–121. DOI: https://doi.org/10.1007/s11277-015-2596-7

[49] Dandan Song, Fei Sun, and Lejian Liao. 2015. A hybrid approach for content extraction with text density and visual importance of DOM nodes. *Knowl. Info. Syst.* 42, 1 (Jan. 2015), 75–96. DOI: https://doi.org/10.1007/s10115-013-0687-x

[50] Rashmi D. Thakare and Manisha R. Patil. 2015. Extraction of template using clustering from heterogeneous web documents. *Int. J. Comput. Appl.* 119, 11 (2015).

[51] R. Uma and B. Latha. 2018. Noise elimination from web pages for efficacious information retrieval. *Cluster Comput.* (Mar. 2018). https://link.springer.com/article/10.1007/s10586-018-2366-x#citeas.

[52] Erdinç Uzun, Hayri Volkan Agun, and Tarik Yerlikaya. 2013. A hybrid approach for extracting informative content from web pages. *Info. Process. Manage.* 49, 4 (2013), 928–944. DOI: https://doi.org/10.1016/j.ipm.2013.02.005

[53] Karane Vieira, André Luiz da Costa Carvalho, Klessius Berlt, Edleno S. de Moura, Altigran S. da Silva, and Juliana Freire. 2009. On finding templates on web collections. *World Wide Web* 12, 2 (2009), 171–211. DOI: https://doi.org/10.1007/s11280-009-0059-3

[54] Karane Vieira, Altigran S. da Silva, Nick Pinto, Edleno S. de Moura, João M. B. Cavalcanti, and Juliana Freire. 2006. 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).* ACM, New York, NY, 258–267. DOI: https://doi.org/10.1145/1183614.1183654

[55] Thijs Vogels, Octavian-Eugen Ganea, and Carsten Eickhoff. 2018. Web2Text: Deep structured boilerplate removal. *CoRR* abs/1801.02607 (2018). Retrieved from http://arxiv.org/abs/1801.02607.

[56] Yu Wang, Bingxing Fang, Xueqi Cheng, Li Guo, and Hongbo Xu. 2008. Incremental web page template detection. In *Proceedings of the 17th International Conference on World Wide Web (WWW'08).* ACM, New York, NY, 1247–1248. DOI: https://doi.org/10.1145/1367497.1367749

[57] Lan Yi, Bing Liu, and Xiaoli Li. 2003. 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).* ACM, New York, NY, 296–305. DOI: https://doi.org/10.1145/956750.956785

[58] Chenxu Zhao, Rui Zhang, and Jianzhong Qi. 2018. Web page template and data separation for better maintainability. In *Web Information Systems Engineering (WISE'18)*, Hakim Hacid, Wojciech Cellary, Hua Wang, Hye-Young Paik, and Rui Zhou (Eds.). Springer International Publishing, Cham, 439–449.

[59] Shuyi Zheng, Ruihua Song, Ji-Rong Wen, and C. Lee Giles. 2009. Efficient record-level wrapper induction. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM'09).* ACM, New York, NY, 47–56. DOI: https://doi.org/10.1145/1645953.1645962

[60] Shuyi Zheng, Ruihua Song, Ji-Rong Wen, and Di Wu. 2007. Joint optimization of wrapper generation and template detection. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07).* ACM, New York, NY, 894–902. DOI: https://doi.org/10.1145/1281192.1281287

Received February 2018; revised January 2019; accepted February 2019