Departamento de Sistemas Informáticos y Computación

Valencian Research Institute for Artificial Intelligence

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Information Retrieval Based on DOM Trees

Julián Alarte Aleixandre

Supervised by:

Josep Silva Galiana

A Thesis presented for the degree of
Doctor of Computer Science at the Technical University of Valencia

July 2023

Information Retrieval Based on DOM Trees

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Julián Alarte Aleixandre

Supervisor

Josep Silva Galiana	Universitat Politècnica de València

Reviewers

Jesús Almendros Jiménez	Universidad de Almería
Pascual Julián Iranzo	Universidad de Castilla-La Mancha
Ginés Damián Moreno Valverde	Universidad de Castilla-La Mancha

Examiners

Francisco Javier Oliver Villarroya	Universitat Politècnica de València
Pascual Julián Iranzo	Universidad de Castilla-La Mancha
José Ángel Olivas Varela	Universidad de Castilla-La Mancha

A mis hijos, Julián y Gonzalo.

Por vuestras innumerables y maravillosas
visitas al estudio mientras trabajaba en la tesis.
Momentos de pérdida de concentración tan inocentes y
entrañables que siempre recordaré con una gran sonrisa.

Ojalá esta tesis os inspire para cumplir vuestros sueños.

"You never fail until you stop trying."

Albert Einstein

Agradecimientos

En un momento de mi vida decidí dejar un trabajo como desarrollador en una pequeña empresa para realizar el máster de ingeniería del software en la UPV. Terminé las asignaturas del máster, pero al empezar a trabajar en el colegio, no llegué a realizar el TFM en ese momento. Posteriormente, al cabo de unos años, me puse en contacto con mi actual director de tesis Josep Silva para realizar dicho TFM, y él inmediatamente me introdujo en el área de Web Mining. De esa manera llegué al apasionante mundo de la investigación, el cual me fascinó desde el primer momento. Empecé a colaborar con doctorandos del grupo MiST de la UPV, los cuales me acogieron extraordinariamente bien desde el primer minuto.

Al terminar el TFM no tuve ninguna duda en continuar. Tenía clarísimo que quería seguir en el mundo de la investigación y convertirme en doctor. Sin embargo, debido a mi trabajo fuera de la universidad, sabía que no iba a ser un doctorando "típico". Por ejemplo, no iba a poder trabajar en un laboratorio junto a mis compañeros doctorandos, ni iba a poder realizar estancias en el extranjero, etc. Sabía que iba a dedicar una gran parte de mi tiempo libre a investigar, pero no me importaba, al contrario, era una idea que me fascinaba y que hoy en día puedo decir que fue un gran acierto.

Durante estos años he trabajado con varias personas. Todas ellas me han ayudado y apoyado incondicionalmente durante todo el doctorado, pero no tengo palabras para expresar mi agradecimiento a la labor que mi director de tesis, Josep Silva, ha realizado conmigo. Josep ha sido y siempre será mi profesor, mi guía, mi mentor, mi amigo. Él me abrió las puertas y me ha enseñado todo lo que sé sobre el mundo de la investigación, y pese a no estar en un laboratorio ni contratado me ha hecho sentir parte del grupo. Josep, gracias por enseñarme este mundo tan apasionante y por confiar en mí para poder llevar a cabo esta investigación. Gracias por tu tiempo y tu paciencia para explicarme cualquier duda o concepto que no sabía o no entendía. Gracias por guiarme y siempre tener un consejo o una

idea que darme en los momentos en los que estaba atascado. Gracias por valorar mi trabajo; y aunque algo necesitase muchas correcciones, llevarlas a cabo siempre destacando el trabajo de una manera constructiva. Gracias por ver más allá y, ante cualquier contratiempo o adversidad, quedarte siempre con el lado positivo. En definitiva, gracias por ser, además de mi profesor, mi amigo y un gran referente. Da pena que acabe esta etapa en la que me has hecho sentir no solo como un doctorando, sino también como un colaborador y amigo. Investigar ya forma parte de mi vida y espero poder seguir haciéndolo contigo por mucho tiempo.

Durante estos años he intentado siempre aprender algo de las personas de las que me rodeaba. Pero debo destacar la gran labor realizada por dos personas con las que he colaborado. Ellos me acogieron como uno más desde el primer minuto, tuvieron paciencia ayudándome a descubrir un mundo totalmente desconocido para mí hasta ese momento. Recuerdo las reuniones interminables en los seminarios del departamento, los emails enviados y respondidos a cualquier hora, y mis visitas al laboratorio para que me explicaseis conceptos que no entendía. Gracias David y Tama por vuestra ayuda y por todo el tiempo que me habéis dedicado.

En el departamento he conocido también a varias personas que, aunque ellos no lo sepan, me han ayudado a mantener la motivación en los momentos en que lo necesitaba. Todos ellos magníficas personas que no dudan en echarte una mano cuando los necesitas. Algunos de ellos han colaborado con nosotros en las sesiones de lluvia de ideas que realizábamos en los seminarios; a otros los he conocido en las famosas reuniones que seguimos realizando, también llamadas “Scaquing Games”, que son de desconexión, aunque para mí siempre suponen una inyección de motivación. Gracias Sergio, Adrián, Damián, Carlos y Javier, que no se pierdan nunca esos momentos.

También quiero agradecer su labor y su colaboración a Luis, el cual en el marco de su trabajo de fin de grado nos ayudó en la conversión a WebExtension de las herramientas. Además, quiero destacar y agradecer el trabajo de Joan, el cual ha diseñado la portada de la tesis. Los dos bocetos que realizó son tan buenos que he estado bastante tiempo dudando hasta decidirme por la portada definitiva.

Durante todos estos años no he podido evitar contarles en innumerables ocasiones a mis amigos aspectos relacionados con la tesis: que si estábamos preparando un artículo, que si habíamos publicado en tal revista, etc. Agradezco su interés y su paciencia. Además, echaré de menos la mítica frase ¿pero cuándo la terminas?.

Quiero agradecer también su colaboración a mis compañeros del colegio Hermanos Maristas de Valencia. Durante la realización de la tesis, varias veces he necesitado su ayuda. Ellos, hermanos, directivos o profesores, no han dudado ni un segundo en ayudarme incondicionalmente. Saben lo importante que es para mí esta investigación y yo valoro de corazón toda la ayuda que me han prestado.

Es imposible agradecer su inmensa colaboración a Ana, mi mujer, y Julián y Gonzalo, mis hijos. Ellos no lo saben, pero me han dado, entre otras cosas, lo que más me hacía falta para realizar la tesis, el tiempo. Debo pedirles disculpas por todo el tiempo que les he robado, tiempo que podría haber pasado con ellos y lo he dedicado a investigar. He intentado compaginar la familia, el trabajo y la tesis de la mejor manera posible, pero sé que a ellos les he robado muchísimo tiempo y les prometo que haré un esfuerzo para compensarles. Gracias por todo, sin vosotros no lo habría podido conseguir.

Agradezco enormemente a mis padres Leonor y Julián, y mi hermana Nora que siempre me hayan apoyado en todas las decisiones que he tomado. De hecho, nunca tendré suficientes palabras para agradecer a mis padres que confiaran en mí y me apoyaran en un momento en el que les necesitaba enormemente. Necesitaba que me dieran una oportunidad y así lo hicieron. Gracias a su cariño, sus ánimos, y su apoyo pasé de tenerlo todo perdido a volver a ser yo. Si no me hubiesen ayudado a superar esa etapa de mi vida, estoy seguro de que nunca sería la persona que soy ahora y tampoco hubiese conseguido ser doctor. Muchas gracias por todo.

Abstract

For several years, the amount of information available on the Web has been growing exponentially. Every day, a huge amount of data is generated and it is made immediately available on the Web. Indexers and crawlers browse the Web daily to find the new information that has been added, and they make it available to answer the users' search queries. However, the amount of information is so huge that it must be preprocessed. Given that users are only interested in the relevant information, it is not necessary for indexers and crawlers to process other boilerplate, redundant or useless elements of the web pages. Processing such irrelevant elements lead to an unnecessary waste of resources, such as storage space, runtime, bandwidth, etc. Different studies have shown that between 40% and 50% of the data on the Web are noisy elements. For this reason, several techniques focused on the detection of both, relevant and irrelevant data, have been developed over the last 20 years. The problems of identifying the relevant content of a web page, its template, its menu, etc. can be faced in various ways, and for this reason, there exist completely different techniques to address those problems. This thesis is focused on the development of information retrieval techniques based on DOM trees. Its goal is to detect different parts of a web page, such as the main content, the template, and the main menu. Most of the existing techniques are focused on the detection of text inside the main content of the web pages, mainly by removing the template of the web page or by inferring the main content. The techniques proposed in this thesis do not only extract text by eliminating the template or inferring the main content, but also extract any other relevant information from web pages such as images, animations, videos, etc. Our techniques are not only useful for indexers and crawlers but also for the user browsing the Web. For instance, in the case of users with functional diversity problems (such as blindness), removing noisy elements can facilitate them to read (or listen to) the web pages. To make the techniques broadly accessible to everybody, we

have implemented them as browser extensions, which are compatible with Mozilla-based and Chromium-based browsers. In addition, these tools are publicly available, so any interested person can access them and continue with the research if they wish to do so.

Resumen

Desde hace varios años, la cantidad de información disponible en la web crece de manera exponencial. Cada día se genera una gran cantidad de información que prácticamente de inmediato está disponible en la web. Los buscadores e indexadores recorren diariamente la web para encontrar toda esa información que se ha ido añadiendo y así, ponerla a disposición del usuario devolviéndola en los resultados de las búsquedas. Sin embargo, la cantidad de información es tan grande que debe ser preprocesada con anterioridad. Dado que el usuario que realiza una búsqueda de información solamente está interesado en la información relevante, no tiene sentido que los buscadores e indexadores procesen el resto de elementos de las páginas web. El procesado de elementos irrelevantes de páginas web supone un gasto de recursos innecesario, como por ejemplo espacio de almacenamiento, tiempo de procesamiento, uso de ancho de banda, etc. Se estima que entre el 40% y el 50% del contenido de las páginas web son elementos irrelevantes. Por eso, en los últimos 20 años se han desarrollado técnicas para la detección de elementos tanto relevantes como irrelevantes de páginas web. Este objetivo se puede abordar de diversas maneras, por lo que existen técnicas diametralmente distintas para afrontar el problema. Esta tesis se centra en el desarrollo de técnicas basadas en árboles DOM para la detección de diversas partes de las páginas web, como son el contenido principal, la plantilla, y el menú. La mayoría de técnicas existentes se centran en la detección de texto dentro del contenido principal de las páginas web, ya sea eliminando la plantilla de dichas páginas o detectando directamente el contenido principal. Las técnicas que proponemos no sólo son capaces de realizar la extracción de texto, sino que, bien por eliminación de plantilla o bien por detección del contenido principal, son capaces de aislar cualquier elemento relevante de las páginas web, como por ejemplo imágenes, animaciones, videos, etc. Dichas técnicas no sólo son útiles para buscadores y rastreadores, sino que también pueden ser útiles directamente

para el usuario que navega por la web. Por ejemplo, en el caso de usuarios con diversidad funcional (como sería una ceguera) puede ser interesante la eliminación de elementos irrelevantes para facilitar la lectura (o escucha) de las páginas web. Para hacer las técnicas accesibles a todo el mundo, las hemos implementado como extensiones del navegador, y son compatibles con navegadores basados en Mozilla o en Chromium. Además, estas herramientas están públicamente disponibles para que cualquier persona interesada pueda acceder a ellas y continuar con la investigación si así lo deseara.

Resum

Des de fa diversos anys, la quantitat d'informació disponible en la web creix de manera exponencial. Cada dia es genera una gran quantitat d'informació que immediatament es posa disponible en la web. Els cercadors i indexadors recorren diàriament la web per a trobar tota aqueixa informació que s'ha anat afegint i així, posar-la a la disposició de l'usuari retornant-la en els resultats de les cerques. No obstant això, la quantitat d'informació és tan gran que aquesta ha de ser preprocessada. Atés que l'usuari que realitza una cerca d'informació solament es troba interessat en la informació rellevant, no té sentit que els cercadors i indexadors processen la resta d'elements de les pàgines web. El processament d'elements irrellevants de pàgines web suposa una despesa de recursos innecessària, com per exemple espai d'emmagatzematge, temps de processament, ús d'amplada de banda, etc. S'estima que entre el 40% i el 50% del contingut de les pàgines web són elements irrellevants. Precisament per això, en els últims 20 anys s'han desenvolupat tècniques per a la detecció d'elements tant rellevants com irrellevants de pàgines web. Aquest objectiu es pot afrontar de diverses maneres, per la qual cosa existeixen tècniques diametralment diferents per a afrontar el problema. Aquesta tesi se centra en el desenvolupament de tècniques basades en arbres DOM per a la detecció de diverses parts de les pàgines web, com són el contingut principal, la plantilla, i el menú. La majoria de tècniques existents se centren en la detecció de text dins del contingut principal de les pàgines web, ja siga eliminant la plantilla d'aquestes pàgines o detectant directament el contingut principal. Les tècniques que hi proposem no sols són capaces de realitzar l'extracció de text, sinó que, bé per eliminació de plantilla o bé per detecció del contingut principal, són capaços d'aïllar qualsevol element rellevant de les pàgines web, com per exemple imatges, animacions, vídeos, etc. Aquestes tècniques no sols són útils per a cercadors i rastrejadors, sinó que també poden ser útils directament per a l'usuari que navega per la web. Per exemple, en el cas

d'usuaris amb diversitat funcional (com ara una ceguera) pot ser interessant l'eliminació d'elements irrellevants per a facilitar-ne la lectura (o l'escolta) de les pàgines web. Per a fer les tècniques accessibles a tothom, les hem implementades com a extensions del navegador, i són compatibles amb navegadors basats en Mozilla i en Chromium. A més, aquestes eines estan públicament disponibles perquè qualsevol persona interessada puga accedir a elles i continuar amb la investigació si així ho desitjara.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

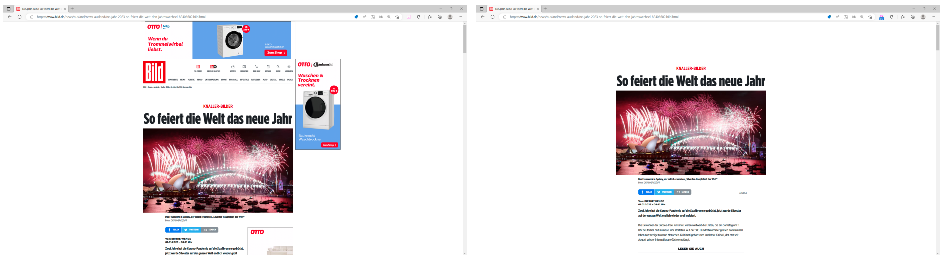
Preamble

1.1 Motivation

A web page contains information that can be classified as relevant or irrelevant content according to the user's needs. For that reason, the extraction of information from web pages is a productive task for both, humans and computer systems. Usually, authors refer to the relevant content in a web page as *main content* (see, e.g., [10, 17, 22, 120, 70]). It can consist of text, images, and any other multimedia; and it is generally surrounded by or even mixed with irrelevant (noisy) information such as menus, advertisements, headers, footers, banners, etc. Removing this irrelevant and noisy information from a web page is key to extracting the relevant data for the user or computer system. The task of extracting the main content from a web page consists in isolating the useful information by removing the elements that do not contain useful knowledge for the user (see, e.g., in Figure 1.1 right, where the main content of a news german article (including images) has been extracted from the Bild newspaper web page (left).

Since the web pages on the World Wide Web are extremely heterogeneous, even in web pages using the same content management system (CMS), the task of extracting information blocks (main content, menu, template, etc.) is a challenging task.

The importance of web templates lies in the fact that they are used as a basis for building new web pages that share a common look and feel. This is a key element in web development because the reuse of components allows for the automation of tasks. It should be highlighted that most websites are maintained automatically by code generators that produce web pages based on templates. The use of web templates is also essential for users who browse the web because they can benefit from intuitive and uniform web designs which share a common vocabulary of coloured and formatted visual elements.



KNALLER-BILDER
So feiert die Welt das neue Jahr
KNALLER-BILDER
So feiert die Welt das neue Jahr

Figure 1.1: News article from Bild newspaper (left) and its main content (right)

The detection of the main content is also a key element for indexers and crawlers:

- The isolation of the main content helps indexers and crawlers to focus on the most relevant information. The main purpose of indexers and crawlers is to provide users with only relevant information. Therefore, extracting the main content is an essential task in order to preprocess that information. Gibson et al. [44] measured that template elements represent between 40% and 50% of all the data on the Web. This justifies the development and application of techniques such as template extraction [120, 114], or main content detection [117, 15] as a preprocessing method.

- Processing the whole web pages (including noisy elements) can lead to a waste of resources such as bandwidth, storage, or time. Hence, indexers and crawlers perform preprocessing tasks on the web pages in order to isolate the main content from other noisy information. As a consequence, due to its importance, the main content is indexed and stored in another way.

Modern crawlers and indexers do not treat all terms in a web page in the same way. Web pages are preprocessed to identify the template because template extraction allows them to identify those pagelets that only contain noisy information such as advertisements and banners. This content should not be indexed in the same way as the relevant content. Indexing the non-content part of templates not only affects accuracy, but it also affects performance and can lead to a waste of storage space, bandwidth, and time. Template extraction helps indexers to isolate the main content. This allows us to enhance indexers by assigning higher weights to the really relevant terms. Once templates have been extracted, they are processed for indexing (they can be analyzed only once for all web pages using 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 web pages. They are also essential to compute pageranks. Modern websites are usually responsive and thus, they are automatically adapted to different screen sizes. However, there is also a large amount of websites not adaptable for different device sizes. Isolating the main content and thus, removing the templates, is also useful to adapt the web pages to small devices [28], focusing only on the relevant content of the web page.

This thesis focuses on HTML-structured web pages. Thus, web pages that are built using alternate technologies such as Flash[1] are ignored. From an engineering standpoint, a web page is represented as a set of Document Object Model (DOM) nodes [29]. As a consequence, the main content of a web page is represented with a subset of those nodes, and it contains the relevant information of that web page. Similarly, the main menu of a web page is represented with a subset of nodes that include all the links present in the menu. Finally, the web template is represented with the whole DOM tree excluding those nodes that are not repeated in other web pages of the same website.

As shown by the following chapters, the representation of a web page as a DOM tree provides many benefits to block detection techniques. For instance, it allows content extraction techniques to extract not only text content but also other kinds of content such as images, videos, animations, etc. Due to its tree structure, it also allows us to easily perform DOM node operations such as traverse, insert, delete, search, etc. Moreover, representing web pages as a DOM tree also facilitates the implementation of block detection techniques as browser extensions.

1.2 Contributions of the thesis

This thesis contributes to the area of web page block detection with the development of new techniques. On the practical side, an important result of the thesis is the development of a workbench for template detection and content extraction. This workbench can be used to plug in different implementations and share common resources, thus the comparison of such implementations can be done in a fair manner. The third contribution of

[1]Multimedia software platform also known as Adobe Flash or Macromedia Flash. It has been discontinued since 2020, but it was mainly used for the production of animations, web applications, mobile games, embedded web browser video players, etc.

this thesis is a large suite of benchmarks that can be used in several block detection techniques. The techniques developed in this thesis, as well as the other contributions, are described below:

Page-level Menu Detection (Chapter 5) receives a web page as input and provides its main menu as output. The technique creates a set of DOM nodes that contain the main menu with high probability. Then, these DOM nodes are analyzed to select the node that actually contains the main menu.

Page-level Content Extraction (Chapter 6) isolates the main content of a web page by just analyzing the information found on that web page. The algorithm also builds a set of nodes that correspond to the main content with high probability. Finally, it selects one or several nodes from that set to infer the main content.

Candidates Selection (Chapter 7) detects a set of web pages from the same website that share the same template. This is important in site-level block detection techniques because they need to load and analyze several web pages from the same website.

Equal Top-Down Mapping (Chapter 8) compares the DOM trees of several web pages to identify the template they share. It is usually combined with candidates selection algorithms.

Site-level Template Detection (Chapter 9) identifies the template of a web page by analyzing several web pages from its website. For this purpose, it uses the *Candidates Selection* and the *Equal Top-Down Mapping* algorithms.

Site-level Content Extraction (Chapter 10) isolates the main content of a web page by analyzing several web pages from its website. As the *Site-level Template Detection* technique, it is also based on the *Candidates Selection* and the *Equal Top-Down Mapping* algorithms.

Hybrid Template Detection (Chapter 11) combines the *Page-level Content Extraction* algorithm with the *Site-level Template Detection* algorithm in order to infer the template of a web page.

Workbench for block detection (Chapter 12) provides a common platform that allows any block detection algorithm to access several resources useful for block detection, such as *HTML to DOM* conversion, *Hyperlink Analysis*, *Complete Subdigraph* computation, etc.

TeCo Benchmark Suite (Chapter 13) is a suite of benchmarks that includes 150 real heterogeneous websites. All the benchmarks are prepared for menu detection, content extraction, and template detection techniques.

All the proposed techniques have been implemented as WebExtensions, which are compatible with a wide variety of web browsers (i.e., Mozilla-based and all the Chromium-based browsers). All of them are open source and publicly available. Moreover, they are officially published by Mozilla in the Firefox browser add-ons website.

As with all WebExtensions, our implementations have been written using standard Web technologies (JavaScript, HTML, and CSS) plus some dedicated JavaScript APIs. However, the algorithms presented in this thesis are language—and paradigm—independent.

1.3 Structure of the thesis

The thesis is divided into seven main parts: Introduction, Foundations, Page-level Block Detection Algorithms, Site-level Block Detection Algorithms, Comparison with the State of the Art, Implementations, and Conclusions and future work.

1. The Introduction justifies the motivations of this thesis in Section 1.1, its contributions in Section 1.2, and the structure of the thesis in Section 1.3. Chapter 2 provides an introduction to the block detection techniques used in this thesis.

2. In the Foundations, we first explain in Chapter 3 the DOM tree, which is the data structure used as the basis of all techniques proposed in this thesis. Then, in Chapter 4, we provide a common theoretical setting with all the definitions and notation that are later used and shared in all sections of the thesis. This allows us to use a common vocabulary throughout the whole thesis. However, the organization of the thesis has been thought to allow the reading of any section without following a concrete order because each section is self-contained. For this purpose, the exposition of a technique included in any section could reference some previously explained techniques or algorithms, but they are shortly described at that point if needed. This allows easy access to any particular technique because, if the reader is only interested in one technique, she is not required to read and understand others.

3. The Page-level Block Detection Algorithms part is divided into two chapters: page-level menu detection, and page-level main content extraction.

    - The page-level menu detection chapter (Chapter 5) describes a page-level technique to extract the main menu from a website.
    - The page-level main content extraction chapter (Chapter 6) explains another page-level technique which isolates the main content of a web page.

4. The Site-level Block Detection Algorithms part is divided into five chapters: candidates selection algorithm, equal top-down mapping, site-level template detection, site-level main content extraction, and hybrid template detection.

    - The candidates selection chapter (Chapter 7) introduces a technique to select a set of web pages from the same website that can be used as input in site-level template detection techniques.
    - The equal top-down mapping chapter (Chapter 8) describes a mapping between several DOM trees that allows us to check whether two nodes from different web pages are equal.
    - The site-level template detection chapter (Chapter 9) presents a technique that combines the candidates selection technique in Chapter 7 and the equal top-down mapping in Chapter 8 to infer the template of a web page.
    - The site-level content extraction chapter (Chapter 10) describes a technique that also combines the candidates selection technique in Chapter 7 and the equal top-down mapping in Chapter 8 to isolate the main content of a web page.
    - The hybrid template detection chapter (Chapter 11) introduces a hybrid technique that combines two block detection algorithms. First, the page-level main content extraction technique identifies the main content of a web page. Then, the site-level template detection technique uses the information previously obtained to infer the template.

5. With respect to the Comparison with the State of the Art part, Chapter 12 explains the process we followed to select several techniques from the state of the art and the results we obtained by comparing them.

6. The Implementations block is divided into two chapters: TeCo benchmarks suite and implementation.

    - The benchmark suite (TeCo) used for training and evaluating the techniques is detailed in Chapter 13.
    - The implementation process of all the techniques is described in Chapter 14. In addition, this chapter shows a usage scenario of the implemented techniques.

7. Regarding the Conclusions and future work part, Chapter 15 exposes the conclusions of the thesis, and Chapter 16 enumerates the open lines of work that can be further explored.

Finally, Appendix A provides a glossary of acronyms that can help the reader to easily and quickly query and understand a concept when they need to.

Chapter 2

Block Detection Techniques

This chapter provides a general overview of block detection techniques. The chapter starts describing the most general discipline, which corresponds to *Data Mining*. Then, it outlines a particular case of *Data Mining*, concretely *Web Mining*. It describes some *Web Mining* related concepts, such as *Wrappers*, *Supervised Learning*, and *Unsupervised Learning*. Finally, the chapter establishes a context for the block detection techniques and algorithms described in this thesis, such as *Template Detection*, *Main Content Extraction*, and *Menu Detection*.

2.1 Data Mining

Data Mining, also known as knowledge discovery in databases (KDD), is the process of searching information from large amounts of possible uncertain data through algorithms [87]. Concretely, [70] defined *Data Mining* as the process of discovering useful patterns or knowledge from data sources, e.g., databases, texts, images, the Web, etc. The patterns must be valid, potentially useful, and understandable. *Data Mining* is a multi-disciplinary field involving machine learning, statistics, databases, artificial intelligence, information retrieval, and visualization.

Data Mining tasks are classified into two categories:

- Supervised learning (predictive): those algorithms that map an input to an output based on training data. Some common supervised learning tasks are classification, regression, time series analysis, and prediction.

- Unsupervised learning (descriptive): algorithms that learn patterns from untagged data. Some common unsupervised learning tasks are clustering, summarization, sequence discovery, and association rules.

Despite some authors define more tasks, the data mining process is often organized [70] in the following processes:

- Pre-processing: This process adapts the raw data to the data mining algorithm. Usually, the raw data is not appropriate for mining due to several reasons, and it has to be cleaned to remove noises or abnormalities. In addition, sometimes, the data is too large or involves irrelevant attributes, so it has to be reduced.

- Data mining: The pre-processed data is the input of a data mining algorithm which will discover patterns or knowledge.

- Post-processing: This task examines the discovered patterns in order to identify which ones are useful for applications. There are several evaluation and visualization techniques to make this decision.

Data Mining and *Information Retrieval* are different but related disciplines. An information retrieval system was defined by [97] as designed to make a given stored collection of information items available to a user population, which is similar to the definition in [71], that states that *Information Retrieval* covers algorithms dealing with retrieval subsets from the large collections based on users' needs.

The data mining process traditionally uses structured data stored in relational databases, spreadsheets, and flat files. In the last 20 years, due to the expansion of the Web, *Web Mining* has become highly important and popular.

2.2 Web Mining

The main objective of *Web Mining* is to discover useful information or knowledge from the Web. *Web Mining* uses many *Data Mining* techniques, however, it can not be considered purely an application of traditional data mining techniques because of the heterogeneity of the Web data. *Web Mining* tasks are classified by [70] according to the primary types of data used in the mining process, in:

- Web structure mining, which aims to discover useful knowledge from hyperlinks, since they represent the structure of the Web. For instance, the analysis of the navigational information contained by the hyperlinks of a website may contribute to the identification of its topology. This is essential for indexers and crawlers. It should be noted that traditional *Data Mining* does not cover this kind of task because it is unlikely to find link structures in the data collections it uses.

- Web content mining, which analyzes the Web to extract or infer useful information, patterns, or knowledge from web pages. For instance, we can extract the main content or the main menu of a web page. Moreover, web content mining techniques can also automatically cluster or classify a web page according to its topics. This kind of task is similar to the traditional *Data Mining*.

  It should be noted that Web content mining is different from Web scraping. Authors of [107] define Web scraping as *the process of retrieving necessary information from a website and converting it into a structured form for future analysis*. However, contrarily to Web content mining, this process does not involve any data processing or analysis. For instance, an example of Web scraping is the work of Murali [82], which uses a web scrapper to extract online price information through automated browsing using the structure of the DOM tree.

- Web usage mining, which refers to discovering user access patterns. For that purpose, algorithms analyze Web usage logs, which register every action made by users. It should be highlighted that web usage mining uses many traditional data mining algorithms.

The *Data Mining* process and the *Web Mining* process are very similar, since as stated before the difference is usually in the data collection. Traditional data mining uses data already collected and stored in a data warehouse, while Web mining algorithms have to crawl web pages, sometimes a large amount. The Web mining process is divided into the same tasks as the traditional data mining process: pre-processing, Web data mining, and post-processing.

2.3 Web Content Classification

The need for web page classification for web information extraction and organization has increased considerably due to the exponential growth of the amount of information on the Internet. Because of the high complexity and diversity of web pages, web classification is a complex task which has a high computational cost. Authors of [96] define a typical classification problem as the process of mapping input variables into discrete categories. Similarly, web page classification can be defined as mapping a web page into one or several categories. The following machine learning algorithms, described in [96], are some of the most used algorithms in web classification:

- K-Nearest Neighbors: KNN algorithm classifies a sample based on its k neighbors. Therefore, samples with similar input values are labeled using the same target label. The algorithm classifies a new sample using similarity measures. If multiple neighbors are considered, the algorithm uses a voting mechanism.

- Support Vector Machine: The goal of the SVM algorithm is to compute the best decision boundary (called hyperplane) that separates a n-dimensional space into classes. This allows in the future to easily classify a new data sample in the accurate category.

- Naïve Bayes: This classification model is based in a probabilistic approach. The label of the sample is determined by the class with the highest probability. The Naïve Bayes classifier assumes that the input features are statistically independent each other. Consequently, for a given class, the value of one feature is not affected by the value of any other feature.

- Artificial Neural Network: It is a computational or mathematical model based on biological neural networks. It is formed by an interrelated group of artificial neurons which model the neurons in a biological brain. The neurons receive inputs and produce outputs based on their predefined activation functions. Most ANNs are adaptive systems which change their structure based on external or internal information that pass through the network in the learning phase.

- Decision Tree: It is a supervised learning method used for data classification. Its main goal is to split the data into regions with samples from only one class. This is not possible with real data, therefore their goal is to create subsets as pure as possible (subsets with many samples as possible from a single class). Regions on a decision tree are separated by the decision boundaries, which are the basis for the classification decisions of the decision tree model.

2.4 Wrappers and Unsupervised Learning

A *wrapper* is a method that extracts content from a particular data source with an unstructured format or converts the data into a structured format [102]. Therefore they are mostly used to convert HTML content into a structured format. Based on the technique used to build the wrapper, [70] enumerates three main approaches:

1. Manual approach: One or several human programmers observe the web page and its source code. They find some patterns, and then they write a program that performs the extraction of the target data. The main problem with this approach is that is not scalable to a large number of sites.

2. Wrapper induction: This is a semi-automatic supervised learning solution. In this approach, a set of web pages or data records are labelled and then, a set of extraction rules is learned from them. Finally, the rules are used to extract the targeted data from similar formatted pages.

3. Automatic extraction: This is an unsupervised approach where given a web page or a website, it automatically finds patterns or grammars from them for data extraction. As it does not require manual labelling, it is easily scalable to a large number of websites and web pages.

Supervised learning (predictive) discovers patterns in the training data that associate data attributes to a class attribute. Then, those patterns are used to predict the values of the class attribute of other data instances. On the other hand, we can find unsupervised learning (descriptive) where data have no class attributes, so those data need to be explored to find some intrinsic structures in them.

A typical example of unsupervised learning is clustering. It is the process that organizes data instances into groups. The instances in a group are similar in some way. Therefore, a cluster is a set of data instances which are "similar" to each other (and "dissimilar" to data instances in other clusters).

The techniques proposed in this thesis belong to the third approach since they are unsupervised learning techniques.

2.5 Block Detection

Block Detection is a *Web Mining* discipline that tries to isolate different functional blocks from a web page. Therefore, this discipline includes techniques such as *Content Extraction*, *Template Detection*, *Menu Detection*, etc. Block detection techniques can belong to any of the approaches defined in Section 2.4, but in the case of the techniques presented in this thesis, they all belong to the automatic extraction (unsupervised learning) approach. However, it is possible to find in the literature a wide variety of predictive and descriptive block detection approaches (see, e.g., [46, 117, 24, 51, 118, 115, 66, 123]).

Bar-Youssef et al. [19] defined a pagelet as *a self-contained logical region within a page that has a well-defined topic or functionality.* While *Content Extraction* tries to detect and isolate the main content pagelets of a web page, *Template Detection* tries to isolate the template. Therefore, both techniques are closely related because they are almost complementary: detecting and removing the template of the web page leaves the main content, or the main content plus maybe another functional block such as comments, sub-menus, etc. In addition, there exist many other block detection techniques, such as menu detection, which tries to isolate the main menu of a web page, comments detection which tries to isolate the users' comments, etc.

Besides the classification based on the technique used to build the wrapper (described in Section 2.4), block detection techniques can be further classified depending on the way in which they internally represent the web pages: (i) web pages are treated as HTML code, (ii) web pages are treated as a rendered image, and (iii) web pages are treated as a DOM tree:

i. HTML-based approaches are mainly based on densitometry methods ([79]) that use the textual information of the web page. Many of them assume that the main content on a web page contains a high text density and a low tag density. For instance, Ferraresi et al. [38] analyze the HTML code and define the main content as the largest continuous text area with fewer amount of HTML tags. Kohlschütter et al. [60] examine a small set of shallow text features to classify the text elements of a web page. Weninger et al. [117] defined the CETR method (Content Extraction via Tag Ratios), which analyzes the HTML code and computes the CETR ratio by counting the number of characters and tags inside each tag. The distribution of the code between the lines of a web page 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), especially when it is generated by a non-human-directed system. Li et al. [68] proposed an algorithm called NBCE that initially transforms the HTML source code into the form of the tree structure (different from the DOM tree). It extracts triples from the HTML, which are used to construct a graph based on neo4j[1] database. Finally, the main content of the given web page can be extracted by deciding whether a node is the main content node or not. As a common example, the reader can see the source code of the main Google web page. At the time of writing these lines, all the code of the web page is distributed in only a few lines without any legible structure.

ii. Some techniques, called vision-based methods, assume that the main content of a web page is frequently located, or at least partially located, in the central part of the web page (e.g., Burget et al. [22]). They suppose that the main content (or at least part of it) is visible without scrolling. Authors from [53] propose a main content extraction algorithm that uses some visual features, such as the elements' positions, size, and distance from three centers. Those centers are computed from the browser window, the first browsing area, and the web document. Berg [21] proposed a hybrid technique that combines a densitometry method with vision-based features. The algorithm adds heuristics based on the appearance of the elements to heuristics obtained from the HTML structure. Vision-based techniques are not so widespread as others because rendering web pages for classification involves expensive computational operations [61].

iii. Currently, the most extended approach is to use the representation of a web page as a DOM tree (see Figure 2.1). In 2002, Bar-Yossef et al. [19] proposed a method that infers information from the web page's DOM tree and computes the frequent pagelet sets. Yi et al. [120], Vieira et al. [114] and Alarte et al. [10] also proposed template detection techniques that use the DOM tree representation of the web page. Roughly, these techniques identify the template by finding common DOM subtrees in different web pages of a website.

In particular, Yi et al. [120] proposed a new data structure called Site Style Tree (SST) that summarizes information from various DOM trees. The technique examines several web pages of the website and

[1]https://neo4j.com/

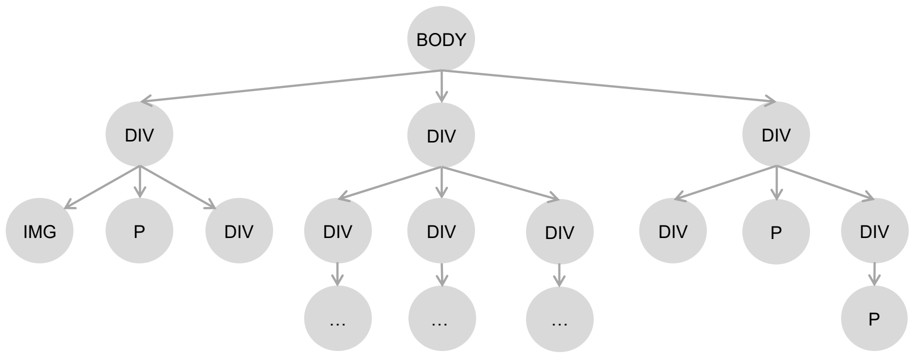


Figure 2.1: Part of a web page represented as a DOM tree

adds the repeated nodes to the SST structure. It is based on the assumption that the most repeated nodes in the SST are template nodes. Sun et al. [106] proposed a general method for extracting content from diverse web pages. It introduces two concepts to measure the importance of nodes: Text Density and Composite Text Density. Insa et al. [51] used a similar notion of density that, for each DOM node, computes the relation between the number of words and leaves in its subtree. Then, among the nodes with a higher density of text, they identify the most relevant node (the main content). In the same line, in [106], the approach is based on computing the ratio between the number of chars and tags in the subtree of a DOM node. For instance, the technique presented in Chapter 6 uses the same idea. It computes a ratio based on the number of text words contained in the descendants of a DOM node. This ratio is later combined with other metrics that also account for non-textual content. Yu and Jin [122] proposed a content extraction algorithm that is based on the DOM tree of the web page and can manage the big heterogeneity and variability of web pages. The algorithm divides the DOM tree of the web page into several blocks, and then it performs the extraction of content blocks based on statistical information. Authors from [121] presented a method that establishes a small neural network which takes multiple features of DOM nodes as input, and predicts whether the nodes contain relevant text information.

It should be highlighted that some techniques combine the rendered image of the web page with the information provided by its DOM tree. Authors of [23] use the DOM tree to extract the information blocks from a web page. Then, they put all the blocks into a pool and an algorithm performs a visual separator detection based on the sizes, positions and separations of the blocks. The input of [64] consists in a screenshot of the web page, a list of bounding boxes, and neighbourhood information for each element obtained from the DOM tree.

All the techniques presented in this thesis are based on the third approach, that is, they all use the representation of a web page as a DOM tree. Therefore, they can take advantage of the properties of the DOM trees.

Block detection techniques can also be classified depending on the number of web pages they can access:

- *Page-level techniques.* They only use the information contained on the target web page.

- *Site-level techniques.* They use the information contained on several web pages (often from the same website).

On the one hand, the main advantage of page-level techniques compared to site-level techniques is that they are faster because they only need to load and analyze one single web page, whereas site-level techniques need to load and analyze a set of web pages. On the other hand, site-level techniques are usually more accurate because they obtain more information since they load and analyze different web pages of the website.

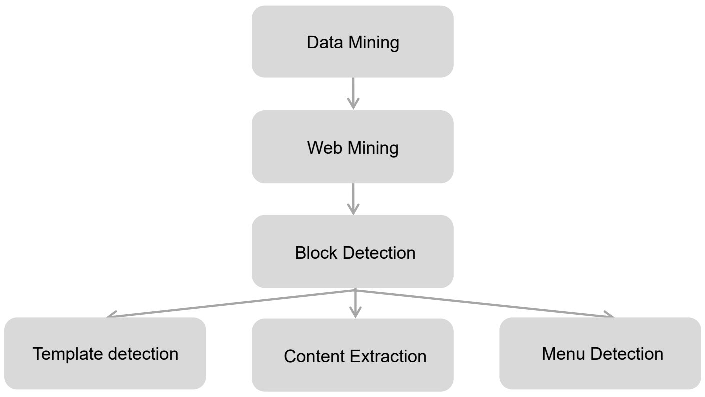


Figure 2.2: Relationship between disciplines

2.6 Conclusions

This thesis presents several block detection techniques, concretely, a site-level template detection technique, another page-level menu detection technique, and a page-level content extraction technique. In addition, it also presents a site-level content extraction technique, and another site-level hybrid template detection technique.

The aim of this chapter is to place those techniques in their corresponding knowledge disciplines. As mentioned above, there are many types of block detection techniques depending on the data they intend to extract. Figure 2.2 helps to place the presented techniques in their corresponding disciplines, as it shows, block detection techniques belong to a more general discipline called *Web Mining*. As mentioned in Section 2.2, *Web Mining* tries to discover useful information from the Web, therefore, it depends on a more general discipline called data mining (see Section 2.1), which is the process of searching information from a large amount of data through algorithms.

Part II

Foundations

Chapter 3

The DOM tree

The Document Object Model (DOM) [29] is basically a way of conceptualizing the contents of a document. Concretely, it is a multi-platform and language-independent interface that represents an HTML or XML document as a hierarchical tree structure, where each node corresponds to a part of the document. Therefore, the DOM produces the representation of an HTML or XML document as a logical tree.

Each node of the tree, except for the root node, is connected by an edge to its parent node, and it is also connected by edges to its child nodes (if any). The different branches end in a node, and each node contains objects. There also exist methods to access the tree or change elements such as the tree structure, the style or the content of a node, as well as event handlers attached to the nodes. There is a direct relationship between an HTML or XML code and its associated DOM tree.

```
<html>
  <head>
    <title>Sample web page</title>
  </head>
  <body>
    <h1>Title of the web page</h1>
    <div>
      <p>Content of the web page</p>
    </div>
    <p>Footer of the web page</p>
  </body>
</html>
```

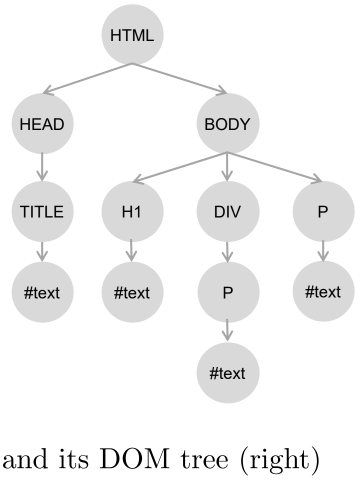


Figure 3.1: HTML document (left) and its DOM tree (right)

Example 3.0.1 *Consider the HTML example code and its associated DOM tree in Figure 3.1. Each HTML tag has a corresponding DOM node in the*

DOM tree. Moreover, each piece of text has its corresponding "#text" DOM node. We can observe that each DOM node has only one unique parent, but a DOM node can have several (or none) children.

3.1 Brief history of DOM

The history of the Document Object Model starts with the *"first browser war"* (1995) between Netscape Navigator and Microsoft Internet Explorer [56]. In December 1995, Netscape Communications released Netscape Navigator 2, which included JavaScript. Later, in august 1996, Microsft released Internet Explorer 3.0, which included a reimplementation of JavaScript called JScript. Both, JavaScript and JScript resulted in a significant improvement because they allowed web developers to create web pages with client-side interactivity. This first approach was partly defined in the specifications of HTML 4, and is known as "DOM level 0" or "Legacy DOM". At that time, the most common usage of the "DOM level 0" allowed us to do basic things such as the creation of image rollovers, or client-side form validation.

Later, in 1997, Netscape Communications and Microsoft released version 4.0 of their browsers (Netscape Navigator and Microsoft Internet Explorer respectively). Those browsers included support for Dynamic HTML (DHTML), which is the first integration between JavaScript, HTML, and CSS. DHTML was the combination of those three techniques and allowed enabling changes to a loaded HTML document. Unfortunately, the development of the DHTML DOM extensions by each browser was in parallel and proved to be incompatible. These versions of the DOM are known as "Intermediate DOM". In conclusion, DHTML tried to offer lots of possibilities, but unfortunately, it was very hard to use due to the incompatibilities between both browsers. Figure 3.2 shows an example of the difference between both versions of DOM while trying to find out the left position of the element "elem" and assign it to the variable "pos". The first line corresponds to a code written for Netscape DOM, while the second line corresponds to a code written for Microsoft DOM.

```
var pos = document.layers['elem'].left;

var pos = document.all['elem'].leftpos;
```

Figure 3.2: Example of the differences between both versions of Intermediate DOM

Finally, in 1998, the World Wide Web Consortium (W3C) standardized the DOM (known as "DOM level 1"). Microsoft released Internet Explorer 5 with full support for W3C's standardized DOM, while Netscape launched Netscape Navigator 6 which also supported the standardized DOM. It should be highlighted that both browsers were still backwards compatible. "DOM level 1" defines a set of objects and interfaces to access and change the objects of the document, which is represented as a hierarchical tree structure. The last version defined by W3C was "DOM level 4", in 2015. Between versions 1 and 4, W3C defined "DOM level 2" in 2000, and "DOM level 3" in 2004.

3.2 Main characteristics of DOM

The Document Object Model (DOM) is an API for well-formed XML and valid HTML documents. It provides a definition for the logical structure of documents, as well as the way a document is accessed and manipulated.

Through the tools provided by the Document Object Model, programmers can create documents, navigate their structure, and also add, change, or remove elements and content. Any element found in an XML or HTML document can be accessed, modified, deleted, or inserted using the DOM.

Since "DOM level 1", as different versions are specified by W3C, the key objective for the DOM is to provide a standard API that can be used in a wide range of environments and applications. In addition, it should be noted that the DOM is designed to be used with any programming language.

Documents in the DOM have a logical structure which is similar to a tree; concretely, it is like a "forest" or "grove", since they can contain more than one tree. Each document includes up to one doctype nodes, exactly one root element node, and zero or more comments or processing instructions. For the document, the root of the element tree corresponds to the root element. Nevertheless, the DOM does not state that documents must be implemented as a grove or a tree, nor does it define how the relationships among objects should be implemented. Therefore, the DOM is a logical model that may be implemented conveniently. It is important to highlight the *structural isomorphism* property of the DOM, which states that if any two Document Object Model implementations are used to create a representation of the same document, they will create the same structure model [29].

As it is an object model, the DOM identifies:

- The objects and interfaces used to manipulate and represent a document.
- The semantics of those objects and interfaces, including both, attributes and behaviour.
- The relationships and collaborations among these objects and interfaces.

3.3 From the document to the browser's screen

A Web browser is a piece of software that transforms an HTML file received from the server into a rendered image that is displayed to the user and allows for user interaction. However, one of the modules of the browser is a software component that establishes what to display to the user based on the files it receives. This core software component is called the *browser engine* and is present in every browser. For instance, the browser engine of Firefox is called Gecko, Safari's is called WebKit, Chrome's is called Blink, etc.

The process followed by a browser to display a web page [47, 48] is known as *critical rendering path* (CRP), and it can be described as follows:

i. The browser reads the raw bytes of data from the server, corresponding to HTML, CSS, and JS files. Those bytes of data are raw data, so they have to be converted to a form that the browser understands:

   - From HTML to DOM (see Figure 3.3): First, based on the character encoding of the HTML file, the raw bytes of data are converted into characters. Then, those characters are parsed into tokens, which are specified by the W3C HTML5 standard[1]. The parser understands each string between angle brackets (e.g., <body>, <div>, etc.) and also understands the set of rules that has to apply to each string. For instance, a token that represents a table has different properties than a token that represents an image. A token is a sort of data structure that includes information about a certain HTML tag. Then, after the tokenization process, those tokens are converted into objects (nodes), which

[1]http://www.w3.org/TR/html5/

define their properties and rules. Finally, the nodes are linked in a tree data structure called DOM tree, which includes all the relationships (parent-child, sibling, etc.) between nodes.

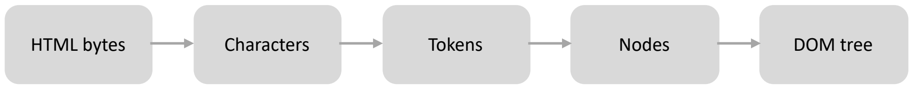


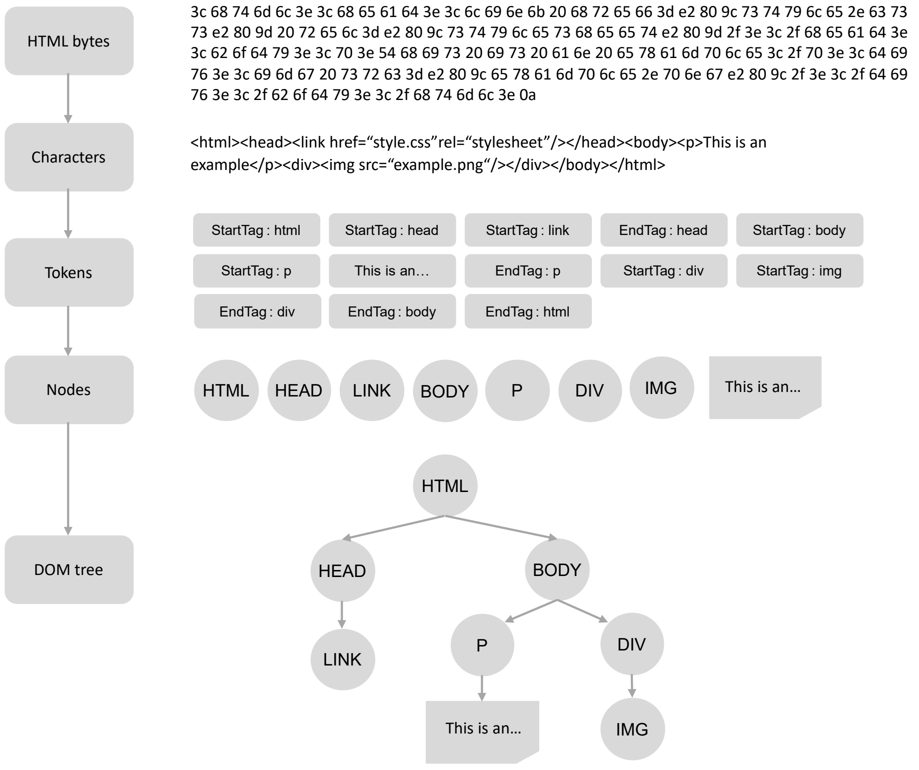
Figure 3.3: HTML to DOM phases

Example 3.3.1 *Consider the HTML example code below. Figure 3.4 details the construction of the DOM tree that corresponds to such HTML code. First, the browser obtains the raw bytes of the HTML file from the network or a local drive. Then, they are converted into characters, which correspond to the HTML source file. The next phase converts strings of characters into distinct tokens. For instance, we can observe that <`html`> is converted into `StartTag:html`, <`head`> is converted into `StartTag:head`, etc. The following phase, called lexing phase, converts the tokens into their corresponding nodes. Finally, the nodes are linked representing all the relationships, which produce the DOM tree that can be observed at the bottom of the figure.*

```
<html>
  <head>
    <link href="style.css" rel="stylesheet"/>
  </head>
  <body>
    <p>This is an example</p>
    <div>
      <img src="example.png"/>
    </div>
  </body>
</html>
```

- From CSS to CSSOM (see Figure 3.5): The raw bytes of CSS follow a process similar to the raw bytes of HTML. To summarize, the raw bytes of CSS data are converted to characters. Then, they are tokenized and converted to nodes. Finally, a tree structure called CSS Object Model (CSSOM) is created. One of the particular features of the CSS is called the *cascade*. The cascade is how the browser infers which styles are applied to an element. When the final set of styles is computed for any object of the web page, the browser starts computing the most general rule applicable to that node (i.e., if it is a "DIV" node which is a child of the "BODY" element, all the styles of the "BODY" element are applied). Then, the browser recursively refines the computed styles with more specific rules. Therefore, the rules "cascade down".

Example 3.3.2 *Consider the CSS example code below. It corresponds to the "style.css" file linked in the HTML in Example 3.3.1. As in Example 3.3.1, first CSS bytes are converted into characters, then characters are converted into tokens, then to-*



```
3c 68 74 6d 6c 3e 3c 68 65 61 64 3e 3c 6c 69 6e 6b 20 68 72 65 66 3d e2 80 9c 73 74 79 6c 65 2e 63 73
73 e2 80 9d 20 72 65 6c 3d e2 80 9c 73 74 79 6c 65 73 68 65 65 74 e2 80 9d 2f 3e 3c 2f 68 65 61 64 3e
3c 62 6f 64 79 3e 3c 70 3e 54 68 69 73 20 69 73 20 61 6e 20 65 78 61 6d 70 6c 65 3c 2f 70 3e 3c 64 69
76 3e 3c 69 6d 67 20 73 72 63 3d e2 80 9c 65 78 61 6d 70 6c 65 2e 70 6e 67 e2 80 9c 2f 3e 3c 2f 64 69
76 3e 3c 2f 62 6f 64 79 3e 3c 2f 68 74 6d 6c 3e 0a
```

```
<html><head><link href="style.css"rel="stylesheet"/></head><body><p>This is an
example</p><div><img src="example.png"/></div></body></html>
```

Figure 3.4: HTML to DOM phases example

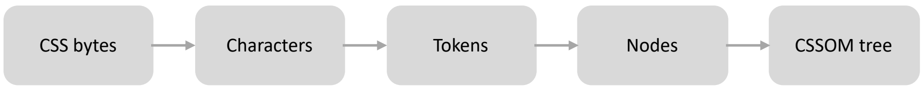


Figure 3.5: CSS to CSSOM phases

kens into nodes, and finally, the tree structure is built. In this case, that structure corresponds to the CSS Object Model (CSSOM). Figure 3.6 shows the CSSOM tree that corresponds to the CSS example code below. We can observe that the CSS rules in red correspond to the rules previously defined for ancestor nodes, while the black CSS rules correspond to the specific rules defined for the node.

```
body { font-color: gray }
p { font-size: 1.5em; font-style: italic }
div { text-align left }
img { float: right }
```

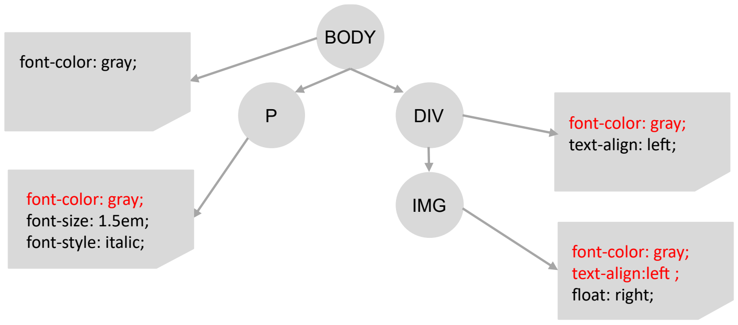


Figure 3.6: Generated CSSOM tree

- In this phase, the browser has to consider the JavaScript code because it can modify the styling and content of the web page. For instance, JavaScript code can modify the CSSOM properties of the elements from the DOM tree, or even it can add and remove elements from the DOM tree.

  When the browser engine finds JavaScript code, it stops the DOM construction process until the script finishes its execution. This is because, as pointed out above, the execution of a script can alter the DOM structure. It should be highlighted that adding the *async* property to the $< script >$ tag overrides this. On the other hand, when the browser engine finds JavaScript code and the CSSOM is not yet ready, it halts the execution of the script until the CSSOM is ready. This guarantees that the script does its job of changing the styling of the web page (if it has to). Therefore, JavaScript code execution is "parser blocking", that is, when the browser encounters JavaScript code, it pauses DOM and CSSOM construction and executes the code before proceeding with it.

ii. Then, the browser combines two independent structures (DOM and CSSOM) into a new structure called *the render tree.* This new structure contains the information of all visible DOM content on the web page, and the CSSOM information required by the different DOM nodes. The render tree is used by the browser to compute the layout of the visible elements, and it is the input of the painting process that performs the rendering of the pixels to the screen.

   The render tree is constructed as follows:

   - The process starts at the root of the DOM tree and traverses each visible node. Not visible nodes (i.e., meta tags, script tags, nodes hidden via CSS, etc.) are omitted since they must not be part of the rendered output.
   - For each visible node, the process finds its corresponding CSSOM rules and applies them.
   - Finally, the process adds the content to the visible nodes.

   It should be noted that the browser will not render any processed content until the CSSOM is built, as CSS is a render-blocking resource. The output of the process is a render that includes both, the style and the information of all the content that should be displayed on the screen.

   Figure 3.7 shows the render tree built from the DOM and CSSOM trees. We can observe that the non-visible nodes ("HTML", "HEAD", and "LINK") do not appear in the render tree. In addition, the "BODY" node appears with a dashed line because it is not a visible node, concretely its descendants are visible nodes.

iii. Once the render tree has been built, the browser performs the layout. That is, it renders the information stored in the render tree to the screen. For that, it computes the exact size and position of each object on the web page.

iv. Finally, considering the information about the sizes and positions of each DOM node, the browser "paints" those nodes to the screen. The result of this phase is the browser screen with the different elements rendered to the screen.

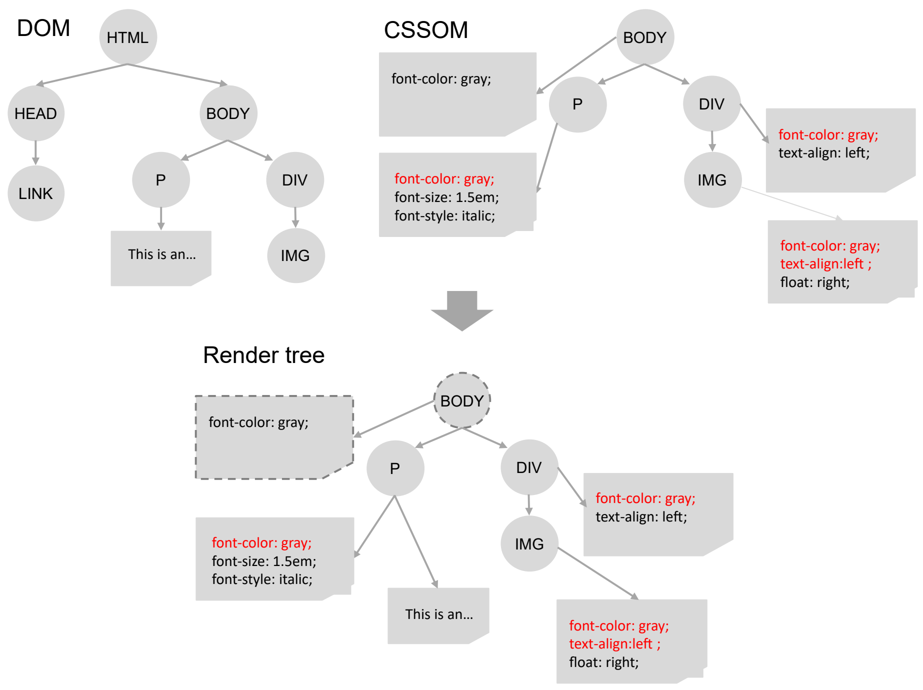


Figure 3.7: Render tree from DOM and CSSOM

To summarize, Figure 3.8 shows the critical rendering path process. The input of the process is the HTML, CSS and JavaScript data. The HTML is converted to DOM and the CSS is converted to CSSOM, but in both cases, the browser engine considers the JavaScript code if it has not been explicitly declared as asynchronous. Once the DOM and CSSOM are built, their combination produces the render tree. Then, the layout is performed. That is, the browser engine computes the exact size and position of each object on the web page. Finally, the nodes are "painted" to the screen producing the representation of the web page on the users' browser.

3.4 Conclusions

This chapter introduces the concept of DOM tree, describing its main characteristics and the process performed by the browser to convert an HTML file into a DOM tree. The DOM tree contains all the relevant data from the HTML web page, except for the styling which is contained by the CSSOM tree.

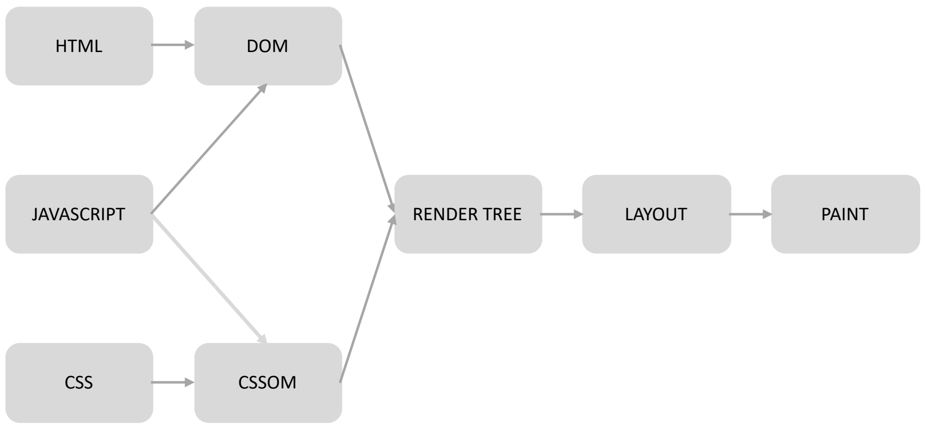


Figure 3.8: Critical Rendering Path scheme

As described in Section 2.5, all the techniques presented in this thesis use the representation of a web page as a DOM tree. Using DOM trees provides many benefits to block detection techniques. For instance, regarding content extraction techniques, it allows us not only to extract text content (unlike other techniques such as [110, 121]), but also other kinds of content such as videos, animations, images, etc. In addition, representing web pages as a DOM tree allows us to easily perform operations such as traverse, insert, delete, search, etc. DOM nodes. Moreover, it should be noted that representing web pages as a DOM tree facilitates the implementation of block detection techniques as browser extensions, and consequently, to show the extracted block on the users' browser maintaining its styling.

Chapter 4

Preliminary Definitions and Notation

Despite block detection techniques can use different forms of representing the *web pages* (see e.g. Section 2.5), all the techniques described in this thesis have a common denominator: they are based on DOM trees corresponding to *web pages*. All of them convert the HTML files into their corresponding DOM trees and perform the required operations in order to extract the desired block. For this reason, it is essential to provide a definition of some basic terms such as web page, website, hyperlink, etc. that are used throughout the whole thesis.

In addition, this chapter also provides some definitions and notations related to block detection that are used in several chapters. Especially, we provide general and complete definitions for the web page's template, main content, and equal top-down mapping, among others.

4.1 Basic definitions

In this section, we provide formal definitions for some key terms that appear in almost all chapters since they are used by all techniques.

As this thesis uses the DOM tree representation of the web pages to identify some of their information blocks, such as the template, the main content, and the main menu, first we provide a definition for a web page based on its representation as a DOM tree.

Definition 4.1.1 (Web page) *A web page P is a tree (N, A) formed from a finite set of nodes N. Every non-leaf node $n \in N$ contains an HTML tag (including its attributes). Leaf nodes can be text nodes, CDATA section nodes, comment nodes, or notation nodes*[1]*. The root node corresponds to*

[1]Processing instruction nodes are also leaf nodes, but they are only supported in XML documents.

the `BODY` *HTML tag.* A *is a finite set of arcs such that* $(n \rightarrow n') \in A$*, with* $n, n' \in N$*, if and only if the tag or text associated with* n' *is inside the tag associated with* n*, and there does not exist an unclosed tag between them.*

Once we have a formal definition for a web page, we provide a set of functions used by our techniques which are very useful to manipulate web pages. Given a node n in a web page $P = (N, A)$, we define:

- $page(n)$ is P.
- $parent(n)$ represents a node $n' \in N$ such that $(n', n) \in A$.
- $ancestors(n)$ is a set of nodes that contains all the nodes that are in the path from the root to n.
- $descendants(n)$ are those nodes that belong to the subtree of n, i.e., those nodes for which n is an ancestor.
- $subtree(n)$ is a set that contains n and all the descendants of n. That is $descendants(n) \cup \{n\}$.
- $leaves(n)$ is the number of leaves in $descendants(n)$.
- $childNodes(n)$ is the number of children of n.
- $depth(n)$ is the length of the path from the root of the DOM tree to node n.
- $maxDepth(P)$ is the maximum depth of the web page P, i.e., it is the maximum value for function $depth(n)$ (for any n).
- $words(n)$ is the total number of words in $descendants(n)$ excluding those that belong to hyperlinks.
- $distance(n_1, n_2)$ is the length (measured with the number of edges) of the path between two nodes n_1 and n_2. The distance from n_1 to n_2 is 0 if and only if $n_1 = n_2$.

Moreover, it is necessary to provide a formal definition for the relation between two web pages. A hyperlink, among other things, represents the link between two web pages.

Before establishing a definition for hyperlink, it is important to outline how it is represented. Thus, we provide first a formal definition of web address.

Definition 4.1.2 (Web address) *A web address is a non-empty sequence of words joined by juxtaposition $w_1w_2w_3...w_n$, where each word finishes with a slash with the exception of the last one. Any web address can be generated by the following grammar:*
$H = dir$
$H = dir/$
$H = dir/H$
where dir represents any sequence of characters (usually a domain or a directory).

Definition 4.1.2 formally describes a web address as a sequence of words separated by a slash. This web address is the one used in hyperlinks to establish a relation from one page to another.

Definition 4.1.3 (Hyperlink Node) *Given a web page $P = (N, A)$, a hyperlink node is a DOM node whose DOM tagName property value is "A".*[2] *Hyperlink nodes include a web address that points to some web page.*

Definition 4.1.3 describes a hyperlink node as a node that points to another web page through a web address. We represent with $\mathit{hyperlinks}(P)$ the set of hyperlink nodes in a web page P. We use $\mathit{link}(n)$ to refer to the link (url) contained in node n (it is assumed that n is a hyperlink node).

Definition 4.1.4 (Target) *Given a web page $P = (N, A)$ and a hyperlink node $n \in N$, $target(n)$ is the web page $P' = (N', A')$ that is pointed from the URL included in n.*

Definition 4.1.4 describes the function $target(n)$, which given a hyperlink node, gives the web page pointed by its URL.

Definition 4.1.5 (Hyperlink) *Given a website S, a web page $P = (N, A) \in S$, and another web page $P' = (N', A')$, a hyperlink is the relation between a pair of web pages (P, P') such that $\exists\ n \in N \mid n \in \mathit{hyperlinks}(P) \land target(n) = P'$.*

Definition 4.1.5 formally describes a hyperlink as the relation between a pair of web pages where one web page of the pair has a link to the other web page. It should be noted that both web pages can be the same, however they are usually different. Given the domain W of web pages, the set of hyperlinks is a relation $R \subseteq W \times W$.

[2]Note that A and "A" are unrelated. While A is a set of arcs, "A" is a kind of tag in the DOM model.

Some of our techniques need to obtain the set of hyperlinks contained in the subtree rooted in a given node. Definition 4.1.6 describes a formal specification of the set and provides a function that returns the set of hyperlinks contained in the descendants of a DOM node.

Definition 4.1.6 (Node's hyperlinks) *Given a web page $P = (N, A)$ and a node $n \in N$, hyperlinks(n) is the set containing all the hyperlink nodes in descendants(n).*

Finally, based on some of the previous definitions, we can formally define a collection of web pages from the same domain called a website. A website is a set of web pages that belong to the same domain, which usually share the same template and main menu. However, it should be noted that most websites implement more than one template, so two web pages that belong to the same website do not necessarily share their template.

Definition 4.1.7 (Website) *A website S is a set of web pages such that*

- $\exists\ U\ :\ \forall\ P \in S$, *$U$ is a non-empty prefix of P's web address.*
- $\exists\ P_{root} \in S\ :\ \forall\ P' \in S, P_{root} \neq P'\ :\ (P_{root}, P') \in hyperlinks(S)^*$, *where X^* is the reflexive and transitive closure of X, and* $hyperlinks(S) = \{(page(n), target(n))\}$ *with* $n \in \cup_{Q \in S}\ hyperlinks(Q)$.

Definition 4.1.7 formally describes a website as a set of web pages that meet two conditions: on the one hand, they share a prefix of their URI (it is usually the domain). On the other hand, all the web pages that belong to the website must be reachable from at least one web page of the website (usually known as `index.html`).

4.2 Site-level techniques

This section provides several key definitions for the site-level block detection techniques developed in this thesis. The template detection techniques described in Chapter 9 and Chapter 11, as well as the content extraction technique described in Chapter 10, share their candidates selection strategy and the mapping they use to check whether two DOM nodes from two different web pages are the same node or not.

4.2.1 Candidates selection

Site-level techniques need to load and analyze several web pages in order to extract the desired block. In order to select those web pages (called *candidates*) we need a candidates selection strategy. The candidates selection strategy of the techniques developed in this thesis is based on the importance of selecting web pages from the same website that share their template. The key idea of this candidates selection strategy is that, if the selected web pages share their template, a fair comparison between them can provide the template or the main content with more accuracy.

The definitions described in this subsection are the base of the candidates selection algorithm explained in Chapter 7, which is used by the template detection techniques in Chapter 9 and Chapter 11, and the content extraction technique in Chapter 10.

The aim of the candidates selection algorithm is to establish an order for the web pages that belong to a website. This order identifies the web pages that should be compared to infer the template of the website. That order is formed as the combination of two orders called *link relevance* and *DOM relevance*. Those orders use functions that provide metrics relative to some web page's elements. Some of those functions are formalized through the following definitions.

Definition 4.2.1 (hyperlink distance) *Given two links h, h', the distance from h to h' is defined as:*

$$hDistance(h,h') = \begin{cases} 0 & \text{if} \quad h = h' \\ +n & \text{if} \quad h' = h/d_1/\dots/d_n \\ -n & \text{if} \quad h = h'/d_1/\dots/d_n \\ -m & \text{if} \quad h = d_1/\dots/d_n/d_{n+1}/\dots/d_{n+m} \text{ and} \\ & \phantom{\text{if}} \quad h' = d_1/\dots/d_n/d'_{n+1}/\dots/d'_{n+o} \text{ and} \\ & \phantom{\text{if}} \quad d_{n+1} \neq d'_{n+1} \\ -n & \text{if} \quad h = d_1/\dots/d_n \text{ and} \\ & \phantom{\text{if}} \quad h' = d'_1/\dots/d'_m \text{ and } d_1 \neq d'_1 \end{cases}$$

where d_i represents any directory used to compose the links.

Definition 4.2.1 computes the distance between two hyperlinks of the same website based on the directories where the web pages pointed by them are located. It should be highlighted that the obtained distance can be positive, negative, or zero.

Because the distance can be positive or negative, then the order of the parameters in $hDistance(h, h')$ is important. Thus, the distance between two hyperlinks is defined from the first hyperlink to the second hyperlink (*hDistance* is not commutative).

Definition 4.2.2 (DOM path) *A DOM path is a non-empty sequence of DOM nodes joined by juxtaposition $n_1 n_2 n_3 ... n_n$, where each node is the parent of the next node.*

Given a DOM node n, the DOM path of n is formed from a sequence of DOM nodes that starts at the root node of the DOM tree and that finishes at n. As pointed out in Definition 4.2.2, each node is the parent of the next node in the sequence.

Definition 4.2.3 (length of a DOM path) *Given a node n in a DOM tree, the length of its path, $path(n)$, is represented with $|path(n)|$, and it denotes the number of nodes in the sequence:*

$$|path(n)| = \begin{cases} 1 & \text{if } path(n) = n_0 \\ 1 + |path(n')| & \text{if } path(n) = n_0 path(n') \\ & \text{where } path(n') \text{ is a DOM path} \end{cases} \tag{4.1}$$

Definition 4.2.3 computes the length of a DOM path as a positive integer by counting the number of DOM nodes in the sequence.

Once the length of a path is known, the distance between two nodes in the DOM tree can be obtained. Note that a path of a node is not forced to start at the root of the tree; but in the definition of distance, both paths start from it.

Definition 4.2.4 (DOM distance) *Given a DOM tree $T = (N, A)$, with $n, n', n_0 \in N$, the* DOM distance *from n to n' is defined as:*

$$dDistance(n, n') = \begin{cases} 0 & \text{if } path(n) = path(n') \\ j + k & \text{if } path(n) = n_0 ... n_i m_1 ... m_j \\ & \text{and } path(n') = n_0 ... n_i m'_1 ... m'_k \\ & \text{and } m_1 \neq m'_1 \end{cases} \tag{4.2}$$

Based on Definition 4.2.3, Definition 4.2.4 computes the distance between two DOM nodes as the length of the shortest path between them in the DOM tree. In this case, the order of the parameters is not relevant. Therefore, the distance from n to n' is equal to the distance from n' to n.

Previous definitions provide a basis for formalizing the two orders that are combined to form the candidates selection order. The following two definitions formalize both orders.

Definition 4.2.5 (link relevance) *Given any set of hyperlink nodes N of a DOM tree and a reference hyperlink h, N is equipped with the preorder $\leq^h_{link}$ called* link relevance *and defined as follows. For any $n_1, n_2 \in N$ we have:*

$$n_1 =^h_{link} n_2 \quad \textit{iff} \quad hd_1 = hd_2$$

$$n_1 <^h_{link} n_2 \quad \textit{iff} \quad \begin{cases} 0 \leq hd_1 < hd_2 \vee \\ hd_2 < hd_1 \leq 0 \vee \\ hd_2 < 0 \leq hd_1 \end{cases}$$

where

$$hd_1 = hDistance(h, link(n_1)) \qquad hd_2 = hDistance(h, link(n_2))$$

Definition 4.2.5 is based on Definition 4.2.1, and establishes an order for the links that have to be explored from the web page whose block has to be extracted (also known as key page). First, this order promotes the links to web pages that are physically located in the same directory as the key page. Then, it explores those links that are physically located in a subdirectory of the directory of the key page. Finally, it considers the links physically located outside the directory of the key page.

Definition 4.2.6 (DOM relevance) *Given any set of hyperlink nodes N of a DOM tree T and a reference set of hyperlink nodes N' in T, N is equipped with the preorder $\leq^{N'}_{DOM}$ called* DOM relevance *and defined as follows. For any $n_1, n_2 \in N$ we have:*

$$n_1 =^{N'}_{DOM} n_2 \quad \textit{iff} \quad \begin{cases} N' = \emptyset \vee \\ dn'_1 = dn'_2 \end{cases}$$

$$n_1 <^{N'}_{DOM} n_2 \quad \textit{iff} \quad dn'_1 > dn'_2$$

where

$$dn'_1 = \min_{n \in N'} dDistance(n, n_1) \qquad dn'_2 = \min_{n \in N'} dDistance(n, n_2)$$

Definition 4.2.6 establishes an order for the hyperlink DOM nodes of a web page based on the definition of the distance between two DOM nodes (see Definition 4.2.4). We can observe that this order promotes the hyperlink DOM nodes that are located further than the rest of the hyperlink DOM nodes in the web page.

4.2.2 Mapping

Once the site-level techniques in this thesis have selected the set of candidate web pages they have to load and analyze, they need to compare the DOM nodes of such web pages using a mapping called *equal top-down mapping* (ETDM). The following definition corresponds to the ETDM, whose algorithm is described in Chapter 8.

Definition 4.2.7 (equal top-down mapping) *A* mapping *from a DOM tree $T = (N, A)$ to another DOM tree $T' = (N', A')$ is an inyective partial relation from N to N'. Given an equality relation $\triangleq$ between tree nodes, a mapping M between two DOM 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 $(parent(n), parent(n')) \in M$.*

This definition is parametric with respect to the equality relation $\triangleq$, since the relation is open to cover any possible implementation.

The *Equal Top-down Mapping* described above establishes two conditions required to consider that two DOM nodes are the same node. First, they have to be equal considering the equality relation $\triangleq$. Then, as it is a top-down algorithm, to map two DOM nodes as equal, their ancestors must also have been mapped as one to one equals.

4.3 Web page blocks

4.3.1 Web page menu

From a web designer point of view, a web page menu is a collection of hyperlinks that point to the same or another web pages. It is often used to establish a hierarchy inside the own website, so that web page menus include essential navigational information.

From the point of view of DOM, as defined in [3], a web page menu is a DOM node whose subtree is the smallest subtree that contains at least two hyperlinks pointing to web pages of the same website; moreover, because a menu provides navigation to the website, the same menu must appear in at least another web page of the website. Formally,

Definition 4.3.1 (Web page menu) *Given a website S, a web page $P = (N, A) \in S$ with a node $n \in N$, and a set H of hyperlink nodes in the subtree rooted at n that point to web pages in S, we say that n is a* web page menu *if and only if:*

- $|H| \geq 2$; *and*
- $\nexists\ m \in N \mid (n, m) \in A^* \ \wedge\ \forall\ h \in H\ .\ h \in descendants(m)$; *and*
- $\exists\ P' = (N', A') \in S \mid n \in N'$.

Definition 4.3.1 establishes the three conditions a DOM node must meet to be considered a web page menu. First, the node at least contains two hyperlinks in its descendants pointing to two web pages in the same website. Second, for each set of hyperlinks that form a menu, its menu node is unique because it cannot contain a child that is a menu node for the same set of hyperlinks. Third, the main menu node must appear at least in another web page of the website.

Based on Definition 4.3.1, DOM-based block detection techniques can infer the main menu of a web page by detecting the DOM node that corresponds to it. Given a web page or a website as input, those techniques compute the main menu and return it as a single DOM node.

4.3.2 Template

The definition of template of a web page cannot be outlined from the point of view of a single web page since it is based on the comparison of that web page with some other web pages from the same website. The part of a web page that is considered a template is shared with other web pages from the same website. It provides a common design and navigation to at least a set of several web pages from the same website. Therefore, a web template can be represented as a set of DOM nodes that are repeated in several web pages from the same website. Formally,

Definition 4.3.2 (Template) *Given a web page $P = (N, A)$, and a website S, $P \in S$, the template is a set of DOM nodes $M \subseteq N$ such that:*

$$\forall m \in M\ .\ \exists P' = (N', A') \in S,\ P' \neq P,\ m \in N'$$

According to Definition 4.3.2, the set of all DOM nodes considered as a template must be present in other web pages from the same template.

4.3.3 Main content

In order to provide a definition of main content which is independent of any method, we follow an engineering perspective based on the structure of the web page. The definition assumes the existence of a labelling *relevant*(n) applied to the leaf nodes. That labelling identifies those leave nodes in the web page that should belong to the main content. Formally,

Definition 4.3.3 (Main content) *The main content of a web page* $P = (N, A)$ *is a set of DOM nodes* $M \subseteq N$ *such that:*

i. All relevant nodes belong to the subtrees of the main content nodes:
$\forall\, n \in N,\ relevant(n)\ .\ n \in subtree(n' \in M).$

ii. All leaf nodes that belong to the subtrees of the main content nodes are relevant:
$\forall\, n\ \in\ leaves(n' \in M)\ .\ relevant(n).$

iii. The set of main content nodes is minimal:
$\nexists\, M' \subset M\ .\ \forall\, n \in N,\ relevant(n),\ n \in subtree(n' \in M').$

Definition 4.3.3 shows that all the DOM nodes labelled as relevant must be descendants from the main content nodes. Moreover, all the descendants of the main content nodes have to be relevant. Finally, the set of main content nodes should be minimal.

4.3.4 Relationship between web page menu and template

The web page's main menu and its template are directly related since the menu is always contained in the template. As stated before, the template provides the web pages in a website with a common design and navigation. Providing navigation involves the inclusion of elements like the main menu, submenus, footers, breadcrumbs, etc.

Web page menus are not considered part of the main content because they do not provide relevant information for the users browsing the website. They provide navigational information which is useful for the users to find relevant information.

4.3.5 Relationship between the template and the main content

The template of a web page and its main content are highly related since they are mutually exclusive. Therefore, the DOM nodes that belong to the template surely do not belong to the main content, and vice versa.

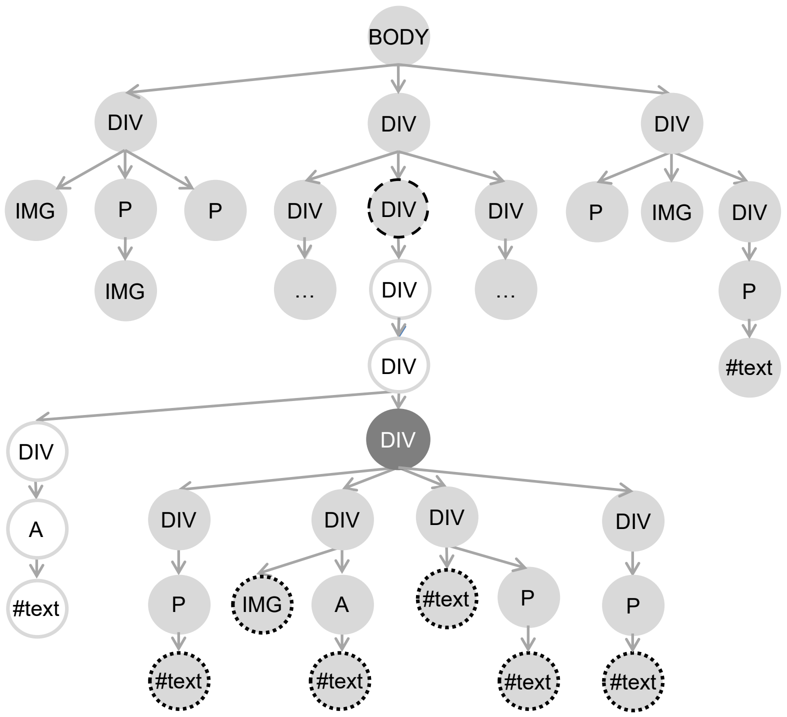


Figure 4.1: Template and main content example

However, despite some papers do consider them as complementary, this fact is not true in all cases. Some authors use boilerplate removal techniques to infer the main content [120, 99, 115], which involves that they consider the template and the main content as complementary. Nevertheless, on many web pages this is not true. For instance, Berg [21] categorizes the content into three groups: main content, optional content, and noisy content (template). The author defines the optional content as content that could be considered part of the main content because it is useful for some users, but it is not the main focus of the web page. An example of optional content could be the comment section of news websites or blogs. It is useful for some users but it is not the most relevant content of the web page.

Moreover, in most web pages there is a small set of DOM nodes that do not belong to the template (they are not repeated in other web pages from the same website), and in addition, they do not belong to the set of main content nodes since it has to contain all the relevant DOM nodes and it also has to be minimal. Figure 4.1 shows an example of this phenomenon. The dashed “DIV” node near the “BODY” node is the borderline node between the template and the rest of the web page. All the descendant DOM nodes of this node do not belong to the template. On the other hand, the darker “DIV” node is the root of the main content nodes. We can observe that there are two “DIV” nodes between them (those “DIV” nodes with white background) that do not belong to the template nor to the main content.

Another example of DOM nodes that belong to none of both, template and main content, are the DOM nodes that are not part of the template and can not be considered relevant. For instance, if we consider a blog with comments enabled for the entries, most users do not consider those comments as relevant since, in most cases, they are not. Figure 4.1 also shows an example of this phenomenon. On the left, we can observe a group of three DOM nodes with white background that are descendants of another node with white background. That group of DOM nodes, as well as their parent, do not belong to the template nor to the main content.

For those reasons, despite sometimes they are complementary, in this thesis we do not consider template and main content nodes as complementary.

4.4 Evaluation metrics

Authors from block detection techniques use a wide variety of criteria to compute the metrics they use to measure and compare their algorithms. Some of the metrics are based on counting retrieved text words [114, 113], retrieved text characters [61], retrieved DOM nodes [10], retrieved text blocks [101, 115], etc. Other authors (e.g., [106, 122]), however, propose more complex metrics, such as computing the common longest subsequence (LCS) between the retrieved text and the gold standard[3].

Regarding this thesis, our metrics are based on the retrieved DOM nodes, the retrieved text words, or both, depending on the technique. These metrics are:

`Recall:` represents (in percentage) the number of DOM nodes (or text words) correctly retrieved divided by the number of DOM nodes (or text words) in the gold standard.

$recall = \frac{number \text{ of correctly retrieved}}{number \text{ in gold standard}}$

[3]The gold standard is the perfect result. In our context, it specifies what parts (measured in DOM nodes, text, etc. depending on the context) form the template, what parts form the main content, or which DOM node represents the main menu.

`Precision:` shows (in percentage) the number of DOM nodes (or text words) that have been retrieved correctly divided by the number of retrieved DOM nodes (or text words).

$precision = \frac{number \text{ of correctly retrieved}}{number \text{ of retrieved}}$

`F1:` is computed using the precision and the recall as:

$F1 = \frac{2*recall*precision}{recall+precision}$

As we can observe in the following chapters, the value of these metrics is expressed in percentage.

Part III

Page-level Block Detection Algorithms

Chapter 5

Page-level Menu Detection

A web page menu or just a menu is an essential block in a website that provides navigation among a subset of web pages from the website. This subset usually includes the most important web pages on the website. As a web page, a menu can be defined as a set of DOM nodes, inasmuch as it is a block from the web page. From a functional point of view, a menu provides valuable information about the website, especially about its structure, since it includes its key sections and implicit information about its sitemap. As a result, web page menus are also significant for crawlers and indexers.

The description of all the links of a website can be useful for search engine optimization (SEO) tasks in order to improve its positioning. Namely, web developers usually use an XML file called “sitemap.xml” that describes the structure of the website and helps to improve its positioning. This file is used by crawlers and indexers to infer the structure of the website. A sitemap containing the main web pages of a website can be built without having to follow all its links by identifying the menu.

Given a web page, the main menu is always found inside its template. Templates typically contain one or several pagelets [26, 19] (i.e., separated logical regions with a well-defined topic or functionality) where the main content is located. Therefore, the process of detecting the menu of a web page could be a helpful tool in the template detection process.

Some aspects taken into account by indexers and crawlers to evaluate the relevance of a web page are the distribution and frequency of hyperlinks and terms. As templates contain a significant number of common hyperlinks and terms appearing on several web pages from the same website, relevance may lead to an inaccurate result (see, e.g., [19, 120, 114]). Consequently, the detection of the template is helpful for indexers in the task of identifying the main content of the web page.

This Chapter proposes a page-level technique (MenEx) that detects and isolates the menu of a web page.

5.1 Related Work

As stated in Chapter 2, *Menu Detection*, *Content Extraction*, and *Template Detection* are directly related disciplines. Since *Template Detection* attempts to isolate the template of the web page, *Content Extraction* attempts to infer its main content, and *Menu Detection* attempts to discover its main menu. These disciplines are an instance of a broader discipline called *Block Detection* that tries to identify and isolate every pagelet in a web page. Most of *Block Detection* papers are related to *Content Extraction* or *Template Detection* (see, e.g., [46, 117, 24, 51, 18, 115, 66, 123]). Moreover, there are some papers in the literature quite related to *Menu Detection*. For instance, in [28], authors proposed a technique that detects the web page structure and adapts it to small screens. The algorithm analyzes the structure of the web page and splits it into small and logically related units that fit into the screen of a mobile device. This process involves the detection of several blocks, including header, footer, sidebars, etc. Therefore, the main menu will be included in one of the blocks detected by the algorithm. Feng et al. [37] combined two machine learning algorithms (Adaboost and SVMs) to partition the text on a web page into information blocks and identify their semantic categories. One of the twelve semantic categories defined by authors is *Menu*, therefore the menu of the web page can be identified by their algorithm.

As well as other block detection techniques, *Menu Detection* techniques could be classified depending on the number of web pages they analyse as page-level and site-level techniques. As described in Chapter 2, the objective is the same, detecting the menus of a given web page; but they use different information. While page-level techniques only use the information contained in the target web page, site-level techniques also use the information contained in other web pages, typically of the same website.

Furthermore, as other *Block Detection* techniques, *Menu Detection* techniques could also be classified depending on the way they solve the problem, concretely, (i) using textual information of the web page (i.e., the HTML code) [38, 62, 68], (ii) using a rendered view of the web page in the browser [22, 61, 53], and (iii) using the DOM tree of the web page [120, 114, 122].

A web page menu has a well-defined functionality not shared with other regions of the web page. Pagelets were defined in [19] as a region of a web page that (1) has a single well-defined topic or functionality, and (2) is not nested within another region that has exactly the same topic or functionality. Despite the fact that in some web pages the menu links also appear in the footer, the main menu is considered a pagelet.

5.2 Menu detection algorithm

This technique, also known as MenEx, is a page-level technique because it only analyzes the information contained in the target web page. The technique receives a web page as input and provides its main menu as output. Since the algorithm uses the DOM tree of the web page, and because of the DOM tree properties, the menu of the web page can be represented using one unique DOM node, that is, the minimum node whose subtree contains the whole menu. First, our algorithm identifies a set of DOM nodes that contain the menu with high probability, called candidate nodes. Then, it explores and analyzes those nodes to select the best candidate as the node that contains the menu.

The technique is divided into three stages:

i. First, the algorithm explores the web page (all its DOM nodes) and, for each DOM node except for the leaves, a weight is computed and assigned. Finally, with those nodes with a higher weight, it builds a set of DOM nodes. It is assumed that the menu of the web page belongs to that set or is an ancestor of one of the nodes in the set.

ii. Second, the algorithm explores each node in the set, it checks its ancestors and evaluates their weight. When a computed value is lower than the weight of an ancestor, the DOM node in the set is replaced with its ancestor.

iii. Finally, the node which represents the menu is detected by comparing the nodes in the set of selected DOM nodes. For each node in the set, an algorithm computes the average weight of its descendants whose weight is over a specified threshold. The node with the best average weight represents the menu node.

5.2.1 Rating DOM nodes

The metric proposed in this section is applied to the DOM nodes of the web page in order to obtain a set of nodes that with high probability represent the main menu or contain a part of it. Therefore, the described algorithm explores the DOM tree of the web page and assigns a weight to each DOM node that meets the following criteria:

i. It is not a leaf of the DOM tree.

ii. It is an element node. Any other types are not considered (e.g., comments, text nodes, etc.).

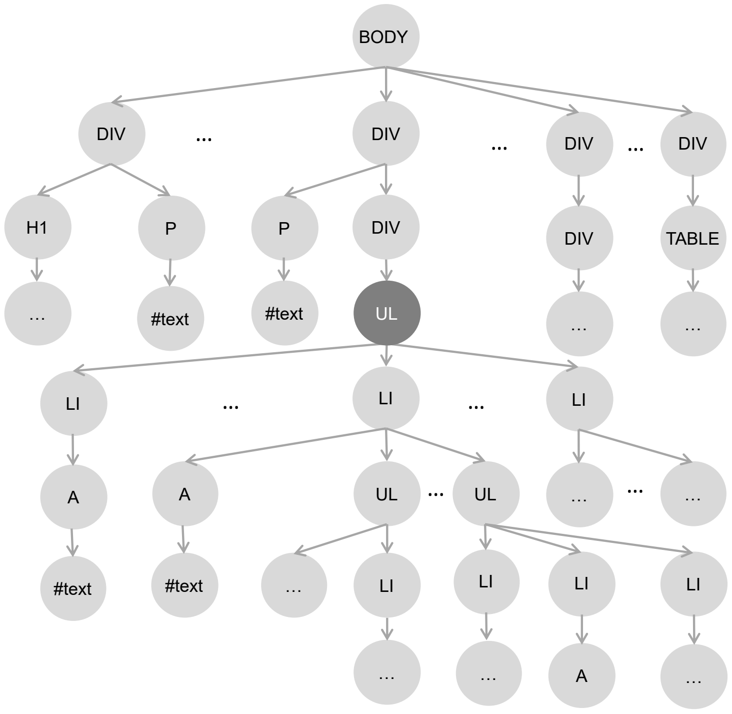


Figure 5.1: Example of a DOM tree where the gray node is the menu

The definition of web page menu (see Definition 4.3.1) uses some of the formal definitions also proposed in Chapter 4: web page (4.1.1), website (4.1.7), hyperlink (4.1.5) and node's hyperlinks (4.1.6).

As stated in Chapter 4, a web page menu can be represented as a DOM node whose subtree is the smallest subtree that contains at least two hyperlinks pointing to web pages from the same website. In addition, the same menu must appear on at least another web page of the website, because the main purpose of a menu is to provide navigation to the website.

On the one hand, an HTML link tag "A" cannot be a leaf node because the element which contains the text of the link has to be a descendant from it. On the other hand, a web page menu contains DOM nodes that are hyperlinks to other web pages. Therefore, the menu of a web page is not a leaf node. Moreover, the only kind of DOM node that is able to contain groups of hyperlinks is the `element` node. Hence, the menu of a web page must be an internal DOM node whose type is `element`.

Among the main features of a web page template, one of the most important is to provide navigation to the web page, therefore most of the hyperlinks of a web page menu are shared by all web pages implementing the same template. Hence, the identification of those DOM nodes which concentrate a high amount of hyperlinks between their descendants is key to identifying the web page menu. Figure 5.1 shows a web page represented with its DOM tree where the menu node is the "UL" node with gray background. Note that the menu is formed by the "UL" node and all its descendants.

The detection of the DOM nodes with a high hyperlink density is essential to identify web page menus, but it is not the only property that helps in this task. In the following we describe several additional properties that must be considered in order to identify the menus accurately [3, 2]. Additionally, we indicate how to label nodes with a weight based on the properties. All the properties are objectively quantifiable and, once they are combined, they provide a weighting that is able to uniquely identify web page menus.

Definition 5.2.1 (Node properties) *In a web page $P = (N, A)$, every node $n \in N \wedge descendants(n) \neq \varnothing$ is rated according to the following properties [3]:*

Node amplitude: *The amplitude of a node n is computed considering its number of children. It is defined as:*

$$\textit{Node amplitude(n)} = 1 - (1/childNodes(n))$$

Link ratio: *The link ratio of $n \in N$ is computed with the following function:*

$$\textit{Link ratio(n)} = \begin{cases} 0 & \text{if } |hyperlinks(n)| < 2 \\ (|hyperlinks(n)| + descendants(n))/2 * descendants(n) & \text{if } |hyperlinks(n)| \geq 2 \end{cases}$$

UL ratio: *It checks whether the HTML tagName of the node is "ul" or not.*

$$\textit{UL ratio(n)} = \begin{cases} 0 & \text{if } n.tagName \neq \text{“}ul\text{”} \\ 1 & \text{if } n.tagName = \text{“}ul\text{”} \end{cases}$$

Text ratio: *It is computed considering the amount of characters of a DOM node and its descendants:*

$$\textit{text ratio(n)} = 1 - (characters(n)/\sqrt{characters(P)})$$

where characters(n) is the amount of text contained in a node and characters(W) is the amount of text of the full web page. Function characters is defined as follows:

$$characters(n) = \begin{cases} chars(n) & \textit{if childNodes}(n) = \emptyset \\ \sum\limits_{n_c \in childNodes(n)} characters(n_c) & \textit{otherwise} \end{cases}$$

where function $chars(n)$ *returns the number of characters in a DOM text node* n*.*

Representative tag: *It evaluates some attributes of the node:*

$$\textit{UL tag(n)} = \begin{cases} 1 & \text{if } n.tagName = \text{“}nav\text{”} \\ 1 & \text{if } n.className = \text{“}menu\text{”} \\ 1 & \text{if } n.className = \text{“}nav\text{”} \\ 1 & \text{if } n.id = \text{“}menu\text{”} \\ 1 & \text{if } n.id = \text{“}nav\text{”} \\ 0 & \text{otherwise} \end{cases}$$

Node position: *The position of* n *in the web page* P *is evaluated using this function:*

$$\textit{Node position(n)} = 1 - (position(n)/|N|)$$

where function $position(n)$ *is the position of node* n *in* P*, if all nodes are sorted with a depth-first traversal.*

The *Node amplitude* property considers the number of children of the DOM nodes. The more children a DOM node has, the higher the probability that the node is part of the menu. Typically, the nodes representing the menu of a web page contain a high number of children that can be either ‘link’ nodes or ‘element’ nodes that contain ‘link’ nodes between their descendants. *Node amplitude* gives importance to the nodes with more children and, consequently, penalizes the nodes with fewer children. Thus, the *Node amplitude* value assigned to a node that contains a significant number of children will be close to 1, while the value assigned to a node without or with few children will be close to 0.

The *Link ratio* property sums the number of hyperlinks contained in the descendants of a DOM node. The more percentage of hyperlinks with respect to the number of descendants, the higher the link ratio. This metric is computed by counting the number of hyperlink nodes between the descendants of a DOM node.

The *Text ratio* property computes the amount of text a DOM node has between its descendants in comparison to the total amount of text of the whole web page. The nodes that represent the menu and their descendants do not use to contain text excluding the text of the hyperlinks. Hence, the text of the hyperlinks is not considered when computing the *text ratio*. Furthermore, as menu nodes and their descendants usually do not contain text, the nodes which contain more text (excluding the hyperlinks) are penalized by the *text ratio* metric.

The *UL ratio* property promotes those element nodes with the HTML tag *UL*[1] because web page menus are usually built using lists of links that commonly are constructed using this HTML tag. It should be noted that in approximately 50% of the websites in the TeCo benchmark suite (see Chapter 13) the *UL* HTML tag is used for the node that contains the menu.

The *Representative tag* property promotes the use of several particular HTML tags or attributes. *UL* HTML tag is commonly used for the node that contains the menu, but we observed some other attributes and HTML tags that are often used in the nodes that represent the menu. Nevertheless, they have not been considered together with the *UL* HTML tag because they are not so common. These HTML attributes are:

- Nav tag: HTML (since version HTML5) describes the *NAV* tag, which is a specific tag that defines a section with a set of navigation links. Despite its use has increased over the last years, most website developers continue using the *UL* tag, or a combination of both. [2]

- Node's id: Some web developers represent the menu using the *nav* or *menu* identifier. For instance, *id="nav"* or *id="menu"*.

[1]HTML Unordered List.

[2]The *nav* tag is the specific tag (and recommendation) in HTML5 for representing menus.

- Node's *className*: The menu is often represented using a node whose *className* contains the *nav* or *menu* classes.

The *Node position* property assumes that menus are commonly positioned at the top or in the top left corner of the web page. Therefore, the DOM node representing the menu should be located in the first nodes of the DOM tree. The *Node position* metric establishes a ponderation that gives more value to the first nodes of the DOM tree (those with a higher probability of being in the visible area) and less value to the last ones (those with a lower probability of being in the visible area).

The combination of these properties provides a weight that can be assigned to each node in the DOM tree and denotes the probability that the node represents the main menu of the web page. Since certainly all the properties do not have the same impact in the computation of the weight, the optimal ponderation to combine all of them must be established using empirical evaluation. Section 5.3.1 discusses in detail the determination process of the best ponderation for the metrics.

5.2.2 Selection of candidates

Once the weight of all the nodes that meet the criteria in 5.2.1 have been computed, the nodes with higher weights are selected. These selected nodes are deemed as 'candidates' because the node representing the main menu is among them. This process of selecting the candidate nodes is trivial: an algorithm visits all the weighted nodes of the DOM tree, checks their weights, and selects those nodes with a weight over a specified threshold. The value of the threshold, which has been calculated based on experimentation, is equal to 0.85 multiplied by the maximum weight of all nodes.

The result of this process is a set of DOM nodes that are candidates to be the menu or part of the menu.

5.2.3 Selection of root nodes

The output of the previous phase is the set of candidate nodes. Those nodes have most probably a high concentration of links with little or no text, but this fact does not guarantee that they form essentially a menu. In fact, they are frequently only part of the menu. For instance, it can be observed in the menu of Figure 5.2 that the DOM node representing the menu option "Staff picks" includes a submenu with a high density of hyperlinks. For this reason, it is identified by our technique as a candidate node.

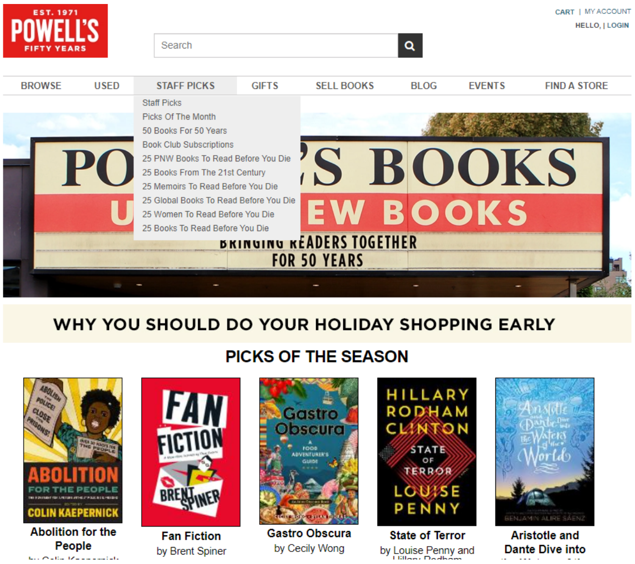


Figure 5.2: Example of menu node with a high density of hyperlinks

In many cases, the DOM node that represents the menu is not selected in the set of candidate nodes. The real menu is usually an ancestor of one of the selected candidates because it is usually a combination of two or more candidates probably with other additional nodes such as images, etc. This phenomenon is more likely to happen in large or complex menus. For instance, in menus that contain several submenus, the selection of candidates often identifies only one of the submenus.

For each candidate node, Algorithm 1 explores its ancestors to search the node that in fact represents the entire menu. Given a candidate node, the algorithm recursively explores its ancestors and checks if over half of their children have a weight higher than the product of the weight of the candidate node by a given threshold t, which is called *root threshold*. When the algorithm finds a node that does not satisfy this criterion, the algorithm stops selecting the last ancestor as the menu node.

Example 5.2.2 *Consider the DOM tree in Figure 5.3, which is a subtree of the DOM tree of a web page. The "UL" node with a dotted border belongs to the candidate nodes set because its weight is 0.87, which is higher than the* root threshold. *Even so, this node does not represent the whole main menu, it only represents a portion of it, so its ancestors have to be analyzed in order to identify the root node of the main menu.*

Algorithm 1 Selection of the root node

Input: A DOM node n and a threshold t.
Output: A DOM node *menuNode* representing a candidate to be the whole menu.

```
begin
  $menuNode = n$;
  $currentNode = n$;
  $baseWeight = n.weight$;
  $found = false$;
  while ($\exists$ $currentNode.parentNode$ $\wedge$ $found == false$)
      $parent = currentNode.parentNode$
      $nodeCount = |\{node \mid node \in parent.children \wedge node.weight > t * baseWeight\}|$;
      if ($2 * nodeCount > |parent.children|$)
          $currentNode = parent$;
          if ($parent.children > 1$)
              $menuNode = parent$;
      else $found = true$
  return $menuNode$;
end
```

Algorithm 1 first visits its parent node, which is the dotted "LI" node, and explores its children to check that more than half of them have a weight *higher than the product of the weight of the candidate node "UL" by the* root threshold*, so the algorithm can continue exploring its ancestors. Subsequently, the "UL" node with a dark background, which is the parent of the "LI" node, is explored. Then, the algorithm checks again that over half of its children have a* weight *higher than the product of the weight of the candidate node "UL" by the* root threshold*, so its parent can be explored. Afterwards, the same process is repeated with the parent of the "UL" node (a "DIV" node with a dashed shape). Again, the algorithm keeps exploring the parent, but as the "DIV" node only has one child, it leaves a pointer to the "UL" node as the menu node. Then, the parent of the "DIV" node with a dashed shape is explored. It is a "DIV" node with two children, one of them is the "DIV" node with a dashed shape and the other is a "P" node. In this case, the* weight *of both nodes is lower than the weight of the candidate node "UL" multiplied by the* root threshold*, so the "DIV" node does not satisfy the criterion. Finally, the algorithm returns the "UL" node with dark background as the root node because it is the last node that satisfies the condition.*

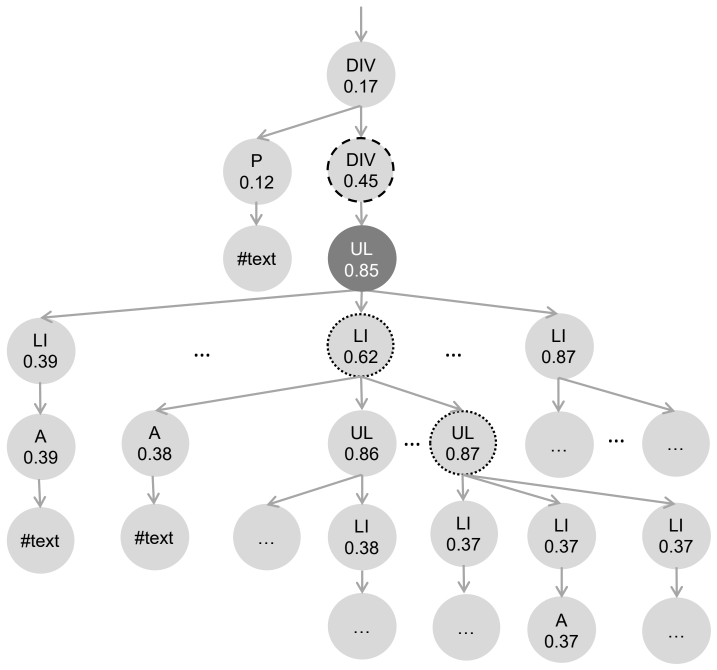


Figure 5.3: Example of the selection of root node candidates

5.2.4 Selection of the menu node

When the number of candidate nodes is greater than one, the application of Algorithm 1 to each node from the set of candidate nodes sometimes produces another set of nodes as output. Due to the fact that there are often several candidates, an algorithm to choose one node from the set of candidate nodes is required. Consequently, this technique needs an algorithm to determine which node from the set of candidate nodes is the real menu of the web page. Algorithm 2 implements this mechanism. This algorithm takes each node in the set and counts the number of descendants whose weight is over a specified threshold, called *menu threshold*. Then, it computes the average weight of those nodes. Finally, the algorithm selects as the menu of the web page the node with the highest average weight. The reason for establishing this criterion is that the menu node is often formed by nodes with a high *weight*.

Algorithm 2 Menu node selection

```
Input: A set of DOM nodes $N$ and a threshold $weight$.
Output: A DOM node $menuNode$ representing the main menu.
begin
  $max = 0$;
  $bestWeight = 0$;
  foreach ($n \in N$)
    $heavyChildren = \{child \mid child \in n.children \ \wedge \ child.weight > weight\}$;
    $nodeCount = |heavyChildren|$;
    $nodeWeight = \sum_{child \in heavyChildren} child.weight$;
    if ($nodeWeight/nodeCount > bestWeight$)
      $menuNode = n$;
      $bestWeight = nodeWeight/nodeCount$;
  return $menuNode$;
end
```

5.3 Implementation

This technique and all its algorithms have been implemented as a WebExtension which is compatible with a wide range of browsers, such as Firefox-based browsers and Chromium-based browsers. When using the tool, users can browse the Internet as usual. Then, if they desire to extract the menu of the loaded web page, they only need to click on the "Extract Menu" button and the add-on automatically does the required actions (DOM nodes rating, DOM nodes analyzing, and menu node selection). Finally, the website gets hidden except for the menu which remains on the browser in its place.

Example 5.3.1 *Figure 5.4 shows a real example of the use of the tool with a web page. The image on the left is the main web page of the* ***www.upv.es*** *website. The image on the right is the output of extracting its menu.*

5.3.1 Empirical evaluation

The theoretical formalization of the technique reveals some parameters of the algorithms that have been left open. This section describes the computation of the value of these parameters based on experimental analysis.

First of all, a weighting must be defined to combine the computed values for the properties proposed in Definition 5.2.1. Then, the nodes with a weight higher than 0.85 are added to a set of candidate nodes. This value

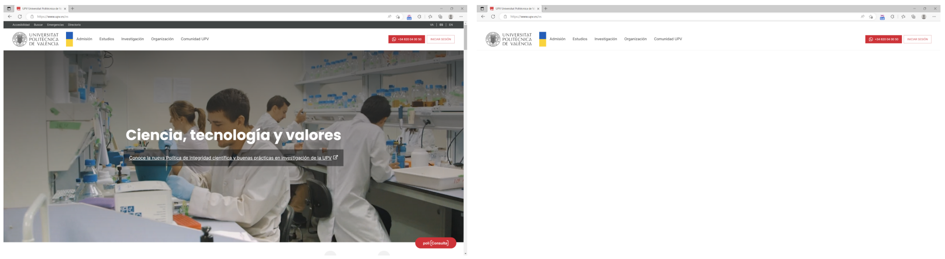


Figure 5.4: Example of the detection of a web page menu

(0.85) has been computed based on experimentation. Our experiments revealed that a value higher than 0.85 produced that, in several benchmarks, the menu node was not added to the set of candidate nodes, while a value lower than 0.85 produced that too many nodes were added to the set, and the process to select the menu node became less accurate.

Subsequently, Algorithm 1 determines which node is a candidate to be the web page menu by exploring the ancestors of a node. The stop condition of this algorithm is a parameter called *root threshold*. Finally, a parameter that selects the menu node among all the potential candidates was established. Algorithm 2 examines the menu nodes in the set of candidate nodes and, for each one, it checks the number of children whose weight is greater than a specified threshold called *menu threshold*.

Determining the root threshold parameter

Algorithm 1 examines the parent nodes of each *candidate node* and selects the node that more likely represents the web page menu.
The algorithm explores recursively the ancestors of a candidate node and, for each one, it checks if more than half of its children have a *node ratio* higher than a specified threshold. The algorithm stops when it finds a node that does not meet this criterion. As previously explained, this process tries to avoid selecting only a submenu or a part of the menu. Therefore, it explores the ancestors of the node to detect if they belong to the menu.

On the one hand, the higher the threshold is, the fewer ancestors are explored. So it will be more difficult to detect nodes from the menu between the ancestors. On the other hand, a low threshold could include the possible nodes of the menu and other nodes that do not really belong to the menu. This means that a high threshold probably increases the recall of the technique, and a low threshold increases the precision.

Determining the menu threshold parameter

Algorithm 2 explores all the nodes in the set of possible menu nodes and determines which node represents the menu of the web page. This algorithm explores all the nodes in the set and, for each one, checks the ratio of all its descendants. The node with more descendants over an established threshold is considered the menu node.

Menu nodes have a big concentration of nodes with a high ratio, so we could think that establishing a high threshold could guarantee the detection of the menu. But, establishing an excessively high threshold could produce the algorithm not selecting any node because none of them has nodes over the threshold. On the other hand, establishing a low threshold could produce the algorithm to select as the menu the node with more descendants, because almost all its descendants would have a ratio over the threshold.

Determining the weights of the features

The first step in the technique is to rate the DOM nodes (5.2.1). A weight is assigned to each node in order to evaluate if it belongs to the menu of the web page. This weight depends on some features of the nodes. The technique establishes that 6 features are needed to compute the weight of a node. Not all these features have the same value, some of them are more relevant than others. Hence, we have to determine the value of these features in the nodes' rating process.

Computing the parameters

The best combination of values for the weighting of node properties and the two thresholds (*root threshold* and *menu threshold*) was computed. The method approximated those values following these steps:

i. First, as TeCo benchmark suite is prepared for menu detection (see Chapter 13), we used the training subset of TeCo (105 web pages) as input and we executed our system for several combinations of parameters.

ii. Next, for each different combination of values for the properties and thresholds, the precision and recall were measured. More than 9 million experiments were performed, with a computing time equivalent to 107 days using an Intel i9 9900k processor.

iii. Finally, the best possible combination of thresholds and properties was selected and evaluated against the evaluation subset of the TeCo benchmark suite.

To measure the technique, we performed several experiments against the training subset of the TeCo benchmark suite. Once the menu was detected, it was manually compared to the real menu to compute the precision, recall and F1 scores of the algorithm. The final weight of each node was computed with the evaluation of different weightings ($NodeWeight = W_1 * Node\ amplitude + W_2 * Link\ ratio + W_3 * Text\ ratio + ...$, where $W_1 + W_2 + W_3 + ... = 1$). All the experiments were repeated using these possible values for the weightings used:

Node amplitude:	[0.00 – 0.35] in steps of 0.05.
Link ratio:	[0.05 – 0.35] in steps of 0.05.
Text ratio:	[0.20 – 0.60] in steps of 0.05.
UL ratio:	[0.00 – 0.35] in steps of 0.05.
Representative tag:	[0.00 – 0.25] in steps of 0.05.
Node position:	[0.00 – 0.25] in steps of 0.05.

In addition, the *Root threshold* and the *Menu threshold* were also evaluated for each possible weighting using the following values:

Menu threshold:	[0.70 – 0.95] in steps of 0.05.
Root threshold:	[0.50 – 0.90] in steps of 0.10.

Table 5.1 presents the best 20 computed combinations after evaluating all possible combinations against all the benchmarks. It summarizes many experiments since each row in the table represents the average of 105 menu detections from 105 different web pages. It should be noted that each row is the result of computing the experiments with one of the possible combinations of the properties and thresholds described in Section 5.2.

As the first row in the table is the fastest combination that produces the best F1 metric, it was selected as the optimum combination of parameters for the technique:

Menu th.	Root th.	Amplit.	Link	Text	UL	Repres.	Position	Recall	Precision	F1	Time (ms.)
0.85	0.90	0.30	0.05	0.10	0.15	0.15	0.25	85.30 %	87.26 %	84.91 %	985 ms.
0.90	0.90	0.30	0.05	0.10	0.15	0.15	0.25	85.30 %	87.26 %	84.91 %	1003 ms.
0.90	0.90	0.35	0.05	0.10	0.15	0.05	0.30	85.30 %	87.14 %	84.85 %	991 ms.
0.90	0.90	0.35	0.05	0.15	0.15	0.00	0.30	85.30 %	87.14 %	84.85 %	1021 ms.
0.80	0.90	0.30	0.05	0.15	0.20	0.05	0.25	85.07 %	87.04 %	84.64 %	1015 ms.
0.80	0.90	0.35	0.00	0.10	0.20	0.05	0.30	84.89 %	87.04 %	84.54 %	981 ms.
0.80	0.90	0.35	0.00	0.15	0.20	0.00	0.30	84.89 %	87.04 %	84.54 %	1019 ms.
0.85	0.90	0.30	0.05	0.15	0.20	0.05	0.25	84.94 %	87.04 %	84.52 %	1050 ms.
0.85	0.90	0.35	0.05	0.10	0.20	0.00	0.30	84.75 %	87.14 %	84.47 %	1024 ms.
0.85	0.90	0.35	0.00	0.10	0.20	0.05	0.30	84.75 %	87.04 %	84.42 %	1007 ms.
0.85	0.90	0.35	0.00	0.15	0.20	0.00	0.30	84.75 %	87.04 %	84.42 %	1030 ms.
0.90	0.90	0.35	0.00	0.10	0.15	0.10	0.30	85.30 %	85.82 %	84.36 %	989 ms.
0.85	0.90	0.35	0.00	0.10	0.15	0.10	0.30	85.30 %	85.82 %	84.36 %	1000 ms.
0.90	0.90	0.40	0.05	0.15	0.15	0.05	0.20	84.59 %	87.14 %	84.35 %	989 ms.
0.90	0.90	0.40	0.00	0.10	0.15	0.00	0.35	85.30 %	85.75 %	84.32 %	995 ms.
0.90	0.90	0.40	0.05	0.10	0.15	0.05	0.25	84.34 %	86.58 %	84.15 %	992 ms.
0.90	0.90	0.40	0.05	0.10	0.15	0.10	0.20	84.23 %	86.58 %	84.08 %	991 ms.
0.85	0.90	0.30	0.00	0.10	0.15	0.20	0.25	84.75 %	86.48 %	84.02 %	1023 ms.
0.85	0.90	0.30	0.00	0.15	0.15	0.05	0.30	84.71 %	86.58 %	84.02 %	1033 ms.
0.90	0.90	0.30	0.00	0.15	0.15	0.05	0.30	84.71 %	86.58 %	84.02 %	1041 ms.

Table 5.1: Determining the best values of the thresholds and node properties

Node amplitude:	0.30.
Link ratio:	0.05.
Text ratio:	0.10.
UL ratio:	0.15.
Representative tag:	0.15.
Node position:	0.25.
Menu threshold (t):	0.85.
Root threshold:	0.90.

Algorithm evaluation

Once the best combination in the training phase was obtained, the technique was evaluated using the evaluation set of TeCo benchmark suite (see Chapter 13). As stated in Section 13.3, the evaluation set is formed by 45 randomly selected benchmarks, concretely 9 benchmarks from each group. Table 5.2 outlines the results obtained for the performed experiments. The first column shows the domains of the evaluation web pages. Column `Nodes` shows the total DOM nodes that form the web page; column `Links` reflects the total number of hyperlinks contained in the menu; column `Detected` specifies the number of hyperlinks detected by the technique that belong to the menu; column `Retrieved` indicates the total number of hyperlinks retrieved by the algorithm; column `Recall` shows the ratio between the number of correctly retrieved hyperlinks and the total number of hyperlinks in the menu; column `Precision` specifies the ratio between the

Domain	Nodes	Links	Detected	Retrieved	Recall	Precision	F1	Runtime
www.jdi.org.za	619	10	10	10	100.00 %	100.00 %	100.00 %	292 ms.
www.premiere-urgence.org	480	30	30	30	100.00 %	100.00 %	100.00 %	52 ms.
www.indiangaming.org	575	7	7	7	100.00 %	100.00 %	100.00 %	83 ms.
hispalinux.es	501	32	6	6	18.75 %	100.00 %	31.58 %	52 ms.
www.gktw.org	767	8	8	8	100.00 %	100.00 %	100.00 %	63 ms.
www.apnic.net	598	59	59	59	100.00 %	100.00 %	100.00 %	67 ms.
www.unicef.org	1037	4	0	26	0.00 %	0.00 %	0.00 %	131 ms.
www.klimabuendnis.org	851	75	75	75	100.00 %	100.00 %	100.00 %	102 ms.
www.isoc-es.org	259	17	17	17	100.00 %	100.00 %	100.00 %	16 ms.
biztechmagazine.com	1892	97	34	34	35.05 %	100.00 %	51.91 %	638 ms.
www.eeo.com.cn	834	11	11	11	100.00 %	100.00 %	100.00 %	101 ms.
www.wishtv.com	2167	77	77	77	100.00 %	100.00 %	100.00 %	956 ms.
news.mit.edu	2117	8	8	8	100.00 %	100.00 %	100.00 %	527 ms.
asia.nikkei.com	869	42	40	40	95.24 %	100.00 %	97.56 %	126 ms.
www.rcnky.com	1738	17	17	17	100.00 %	100.00 %	100.00 %	2214 ms.
news.discovery.com	2826	68	6	6	8.82 %	100.00 %	16.22 %	1837 ms.
www.kathimerini.gr	1825	83	83	83	100.00 %	100.00 %	100.00 %	331 ms.
news.un.org	1726	42	42	42	100.00 %	100.00 %	100.00 %	536 ms.
frances.forosactivos.net	785	9	9	9	100.00 %	100.00 %	100.00 %	157 ms.
www.wysiwygwebbuilder.com	3936	7	7	11	100.00 %	63.64 %	77.78 %	1336 ms.
www.3dprintforums.com	1040	8	8	8	100.00 %	100.00 %	100.00 %	218 ms.
www.strangehorizons.com	631	23	23	23	100.00 %	100.00 %	100.00 %	72 ms.
communities.apple.com	3136	10	10	10	100.00 %	100.00 %	100.00 %	807 ms.
www.sloweurope.com	4193	29	22	22	75.86 %	100.00 %	86.27 %	3581 ms.
community.ricksteves.com	2057	9	9	9	100.00 %	100.00 %	100.00 %	362 ms.
hackercombat.com	1711	6	6	6	100.00 %	100.00 %	100.00 %	465 ms.
www.scbwi.org	876	6	4	4	66.67 %	100.00 %	80.00 %	106 ms.
johngardnerathome.info	395	21	0	1	0.00 %	0.00 %	0.00 %	38 ms.
www.annmalaspina.com	392	8	8	8	100.00 %	100.00 %	100.00 %	43 ms.
foodsense.is	330	5	5	5	100.00 %	100.00 %	100.00 %	17 ms.
sites.google.com	372	32	32	32	100.00 %	100.00 %	100.00 %	119 ms.
whatever.scalzi.com	1648	11	11	11	100.00 %	100.00 %	100.00 %	206 ms.
www.javiercelaya.es	740	12	12	12	100.00 %	100.00 %	100.00 %	254 ms.
diarium.usal.es	604	3	3	3	100.00 %	100.00 %	100.00 %	88 ms.
www.jameslovelock.org	653	20	20	20	100.00 %	100.00 %	100.00 %	92 ms.
www.cipri.info	933	27	27	27	100.00 %	100.00 %	100.00 %	203 ms.
naranjascarcaixent.com	290	6	6	14	100.00 %	42.86 %	60.00 %	34 ms.
www.technicalbookstoreonline.com	2959	12	6	6	50.00 %	100.00 %	66.67 %	1662 ms.
www.floridarealestatecollege.com	1023	29	29	29	100.00 %	100.00 %	100.00 %	936 ms.
www.basf.com	827	12	2	2	16.67 %	100.00 %	28.57 %	95 ms.
www.mcphersonoil.com	831	34	34	34	100.00 %	100.00 %	100.00 %	163 ms.
www.thirteenhou.com	1217	4	4	5	100.00 %	80.00 %	88.89 %	350 ms.
www.embalajesterra.com	2342	106	106	106	100.00 %	100.00 %	100.00 %	845 ms.
www.crypto.ch	338	3	3	3	100.00 %	100.00 %	100.00 %	32 ms.
www.shopbookshop.com	1727	9	9	9	100.00 %	100.00 %	100.00 %	292 ms.
Average	1281.49	25.51	21.00	21.89	85.93 %	93.03 %	86.34 %	460 ms.

Table 5.2: Results of the performed experiments

number of correctly retrieved hyperlinks and the number of retrieved hyperlinks; column `F1` shows the F1 metric (see Section 4.4); finally, column `Runtime` indicates the computing time of the algorithm (in milliseconds) for that benchmark.

In most of the experiments (more than 70%) the menu was precisely detected (F1=100%). Sometimes, the technique retrieves only a part of the whole menu (a subset of the hyperlinks of the menu). Therefore the recall is low due to the fact that only a part of the menu was detected, and the precision is high (in all cases 100%) because the detected subset of hyperlinks belongs to the menu. In addition, we can observe that there are 2 benchmarks with a F1 value of 0%. This result was obtained because the retrieved DOM node did not contain any hyperlink of the main menu. In both cases the technique retrieves one or more hyperlinks, but none of

Benchmark type	Recall	Precision	F1	Time (ms.)
Institutions / Associations	79.86 %	88.89 %	81.29 %	95 ms.
Media / Communication	82.12 %	100.00 %	85.08 %	807 ms.
Forums / Social	93.61 %	95.96 %	93.78 %	789 ms.
Personal websites / Blogs	88.89 %	88.89 %	88.89 %	118 ms.
Companies / Shops	85.19 %	91.43 %	82.68 %	490 ms.

Table 5.3: Results of the performed experiments grouped by benchmark type

them belongs to the main menu. Therefore, in these cases, the precision, the recall, and the F1 are 0%.

The last row of the table shows that the obtained average F1 is higher than 86%, while the average precision is higher than 93%. In addition, it can be observed that the average runtime of the algorithm is 0.46 seconds. This fact is due that the algorithm only needs to load the web page whose menu has to be extracted because it is a page-level technique. Loading and analyzing more web pages would involve a higher runtime.

The obtained results grouped by benchmark type are shown in Table 5.3. For each benchmark type, the table shows the average recall, precision, F1, and runtime. We can observe that the best results are obtained by the benchmarks that belong to the Forums / Social category since they obtain an average recall, precision, and F1 higher than 93%. In addition, the results obtained by the Personal websites / Blogs benchmarks are significantly high, close to 90%. It should be noted that the Media / Communication benchmarks obtain an average precision of 100%. Regarding the runtime, the benchmarks with lower average runtime are those that belong to Institutions / Associations and Personal websites / Blogs, whose average runtime is significantly lower than other categories. By contrast, it can be observed that the benchmarks that belong to the Media / Communication category have a runtime considerably higher than the benchmarks from other categories. For instance, their average runtime is more than 8 times higher than the average runtime of the benchmarks that belong to the Institutions / Associations category.

Runtime analysis

Figure 5.5 presents the relationship between the number of DOM nodes of the key page and the time (in milliseconds) needed to extract the menu for each web page using the evaluation subset of the TeCo benchmark suite (see Chapter 13). It should be noted that about 89% of the benchmarks

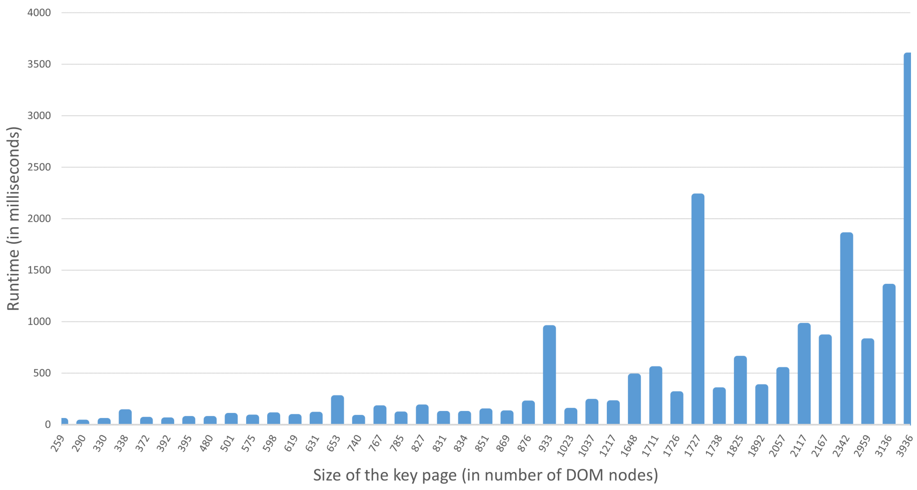


Figure 5.5: Relation between the size of the web page and the runtime

took less than one second, while more than 50% of them took less than 200 milliseconds. The average runtime of the benchmarks was 460 milliseconds.

The figure shows that the highest runtimes correspond to those web pages with the higher number of DOM nodes; however, there are a few benchmarks that do not obtain the expected runtime according to their size. Hence, we decided to explore other variables that, in our opinion, could be related to the runtime of the algorithm.

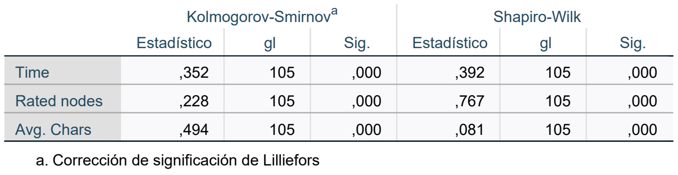
For this purpose, we computed a statistical analysis of the following variables:

- The `number of rated DOM nodes`. The choice of this variable is based on the assumption that the cost of computing the weights assigned to the DOM nodes is significant for the overall runtime. Therefore, the more weighted nodes, the more runtime.

- The `average amount of text per rated DOM node` (measured in chars). This variable was selected because the computation of the `text ratio` is the most complex of the 6 node properties. Therefore, we supposed that computing the `text ratio` of the nodes with more text would affect the runtime significantly.

We used a software called IBM SPSS Statistics[3] to analyze the relationship between these variables. We performed the analysis using the test

[3]https://www.ibm.com/analytics/spss-statistics-software



	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Time	,352	105	,000	,392	105	,000
Rated nodes	,228	105	,000	,767	105	,000
Avg. Chars	,494	105	,000	,081	105	,000

a. Corrección de significación de Lilliefors

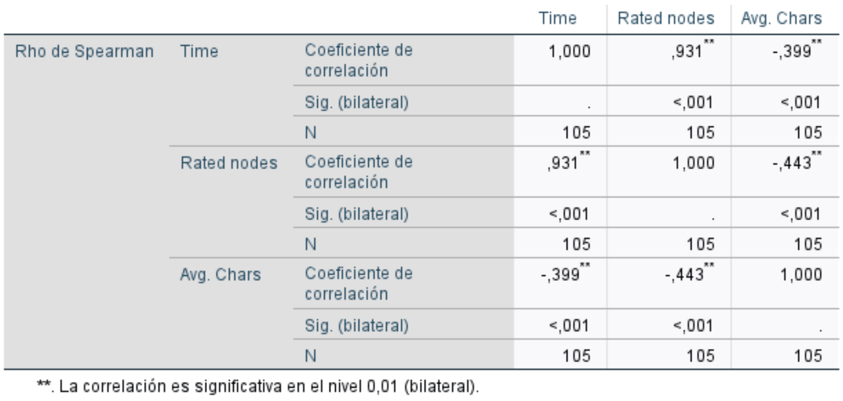
Figure 5.6: Result of the normality test for MenEx

subset of the TeCo benchmark suite, formed by 105 web pages (see Chapter 13). First, we checked if the data were normally distributed. Therefore, we computed Table 5.6. As we can observe in the table, the `sample size` (column gl) is 105, in consequence, the appropriate test is Kolmogorov-Smirnov.

When the `significance` (column Sig.) is less than 0.05, the variable is not distributed normally, hence, the correlation coefficient can be computed through the Spearman test. Figure 5.7 shows the result of the Spearman test for the variables. The correlation coefficient between both variables and the runtime can be observed in the first row of the table. As the correlation coefficient for the `number of rated DOM nodes` variable is close to 1, this variable has a very strong relationship with the `runtime` of the algorithm. In addition, we can observe that the correlation coefficient between the `average amount of text per rated DOM node` variable and the `runtime` is around -0.4, which does not denote a relationship as strong as the `number of rated DOM nodes` variable, but shows a relationship between both variables. The negative sign of the correlation coefficient means that both variables tend to move in opposite directions, while the average text of the DOM nodes increases the runtime decreases. Surprisingly, as argued above, this is the opposite of what we thought when we selected this variable. Our argument to select it was based on that computing the `text ratio` of the nodes with more text would increase the `runtime` significantly.

5.4 Conclusions

Detecting the main menu is a task used by many systems, such as indexers or crawlers. It contributes to the identification of the topology of a website by analyzing the navigational information that is contained in its hyperlinks. It is possible to build an overall sitemap of a website only through the analysis of the navigational information provided by the menu of its web pages. *Menu Detection* is also useful for *Template Detection* since most



			Time	Rated nodes	Avg. Chars
Rho de Spearman	Time	Coeficiente de correlación	1,000	,931**	-,399**
		Sig. (bilateral)	.	<,001	<,001
		N	105	105	105
	Rated nodes	Coeficiente de correlación	,931**	1,000	-,443**
		Sig. (bilateral)	<,001	.	<,001
		N	105	105	105
	Avg. Chars	Coeficiente de correlación	-,399**	-,443**	1,000
		Sig. (bilateral)	<,001	<,001	.
		N	105	105	105

**. La correlación es significativa en el nivel 0,01 (bilateral).

Figure 5.7: Result of the Spearman test for MenEx

Template detection techniques use the web page menu for the detection of the web pages on a website that implement the same template.

This chapter presents a new page-level technique for *Menu Detection*. However, the described algorithms only load and analyze one single web page (the web page whose menu wants to be extracted). This positively affects the performance of the technique because loading several web pages is costly.

A set of features useful in the menu detection process were defined and empirically evaluated in order to define the weighting used. The obtained results reveal that almost three-quarters of the experiments precisely retrieved the menu of the web page. The obtained average precision is 93.03%, and the average F1 metric is 86.34%.

As the present technique is useful to detect the topology of a website, it is able to provide navigational information to other extraction techniques, especially site-level techniques which obtain information by loading several pages from the website. This means that this menu detection technique can be used in combination with other techniques, specially site-level techniques, such as some template detection and content extraction techniques. Table 7.6 in Section 7.3 shows a comparison of several candidate selection methods for our site-level template detection algorithm (TemEx). One of the compared techniques consists in selecting as candidates only hyperlinks from the main menu. First, the technique infers the main menu, and then extracts its hyperlinks.

5.5 Contributions

The menu detection technique described in this chapter provides several contributions that can be used by many systems, such as other block detection techniques, indexers and crawlers, etc.

It establishes a new ponderation system that determines with precision the nodes that more likely belong to the main menu of a web page.

An algorithm that infers the main menu is also provided. The algorithm selects the node that most likely implements the menu based on the ponderation system described before.

Another contribution is a functional implementation of the technique as a WebExtension, which is also officially published by Mozilla in their Firefox browser add-ons website.

Chapter 6
Page-level Content Extraction

Previous chapters provide a classification of the information contained in a web page according to the user's needs. It is classified as relevant or irrelevant content. Therefore, the extraction of information (relevant or irrelevant) from web pages is not only a productive task for computer systems, but also for humans. Chapter 1 refers to the relevant content in a web page as *main content* [10, 17, 22, 70, 120]. The main content is not only formed from text, it is also formed from images, videos, and any other multimedia. It should be highlighted that the main content is usually surrounded by or even mixed with a boilerplate. As defined in previous chapters, a boilerplate is a noisy information such as headers, menus, banners, footers, advertisements, etc. This irrelevant information should be removed in order to infer the relevant data for the user (see, e.g., Figure 6.1).

As all the techniques proposed in this thesis, this algorithm is focused on HTML-structured web pages, therefore, web pages built with alternative technologies are ignored. From an engineering perspective, a web page is formed by a set of Document Object Model (DOM) nodes [29]. As a consequence, the main content can be defined as a subset of those nodes that contain the relevant information of a web page.

Since the web pages on the World Wide Web are extremely heterogeneous, even in web pages using the same content management system (CMS), the task of extracting information blocks (main content, menu, template, etc.) is a challenging task.

As described in Chapter 1, the main content is a key element for indexers and crawlers, and its isolation helps indexers and crawlers to focus on the most relevant information. The main purpose of indexers and crawlers is to provide users with only relevant information. Therefore, extracting the main content is an essential task in order to preprocess that information.

To put the technique in a nutshell: given an arbitrary web page, (1) a set of weights are first assigned to several features of its DOM nodes. This provides a way to (2) represent the DOM nodes of the web page as

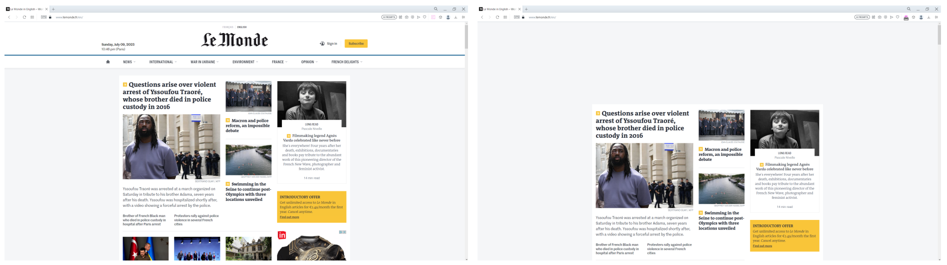

Le Monde
Questions arise over violent arrest of Yssoufou Traoré, whose brother died in police custody in 2016
Macron and police reform, an impossible debate
Swimming in the Seine to continue post-Olympics with three locations unveiled
Filmmaking legend Agnès Varda celebrated like never before
INTRODUCTORY OFFER

Figure 6.1: Web page of www.lemonde.fr's website and its main content (extracted with our web content extraction tool)

points in a four-dimensional Euclidean space $\mathbb{R}^4$, and then (3) the Euclidean distance between the points is computed. Those nodes further away than the median are isolated because they probably contain the main content. Finally, (4) the isolated DOM nodes are analyzed in order to identify the DOM subtree that represents the web page's main content.

6.1 Related Work

This technique takes an arbitrary web page (the key page) as input and outputs a set of DOM nodes that correspond to the main content.

As introduced in previous chapters, *Content Extraction*, *Template Detection*, *Menu Detection*, etc. are interesting *Block Detection* topics due to their relation to web mining, searching, indexing, and web development.

Chapter 2 introduces the concept of pagelet. Bar-Youssef et al. [19] defined a pagelet as *a self-contained logical region within a page that has a well-defined topic or functionality.* While *Content Extraction* tries to detect and isolate the main content pagelets of the web page, *Template Detection* tries to isolate the template. Therefore, as explained in Section 4.3.5, both techniques are closely related because they are almost complementary: detecting and removing the template of the web page leaves the main content, or the main content plus maybe another functional block such as comments, sub-menus, etc.

The last block detection techniques, as those described in this thesis, use a DOM tree for the representation of a web page. In 2002, Bar-Yossef et al. [19] proposed a method that infers information from the web page's DOM tree and computes the frequent pagelet sets. Yi et al. [120], Vieira et al. [114] and Alarte et al. [10] also proposed template detection techniques that use the DOM tree representation of the web page. Roughly, these techniques identify the template by finding common DOM subtrees in different web pages of a website. Moong [80] developed an algorithm that matches DOM trees to classify which nodes are noises and which are contents and, after classification, they are clustered into their group respectively. The algorithm extracts from the web page only the content group. Sun et al. [106] proposed a general method for extracting content from diverse web pages. It introduces two concepts to measure the importance of nodes: Text Density and Composite Text Density. Insa et al. [51] used a similar notion of density that, for each DOM node, computes the relation between the number of words and leaves in its subtree. Then, among the nodes with a higher density of the text, they identify the most relevant node (the main content). Yu and Jin [122] proposed a page-level content extraction algorithm that can adapt to the heterogeneity and variability of web pages, and is based on DOM structure. The algorithm divides a web page into several blocks, and then it performs the extraction of content blocks based on statistical information.

Most recent techniques are based on *Machine Learning*. Vogels et al. [115] describe a sequence labelling technique to classify all text blocks in an HTML page as either main content or boilerplate. Leonhardt et al. [66] propose a neural sequence labelling method for boilerplate removal that is not based on any hand-crafted features and only takes as input the HTML tags and words. Morbieu et al. [81] developed an unsupervised learning method that extracts the textual main content of a web page. The method is divided into three stages: a clustering phase of text blocks, another phase that selects the clusters associated with the main content, and finally a classification phase whose input is the labeled data from the two previous steps. Yu et al. [121] proposed a text extraction algorithm that uses a small neural network which, taking multiple features of DOM nodes as input, predicts whether the nodes contain text information.

6.2 Main content extraction

This content extraction technique, also known as page-level ConEx, rates the DOM nodes of a web page with values for different features in order to isolate the main content of the web page. The input of the technique is a web page, while the output is a set of DOM nodes that represent the main content. It should be noted that the technique is page-level, so it only needs to load and analyze the source web page to isolate its main content. As described in previous chapters, this is especially important because the fact of loading and analyzing only the source web page produces an speed increment and a better use of resources.

The technique is divided into four phases:

i. Some DOM nodes of the web page are selected by the algorithm and, for each one, it computes four weights: position ratio, word ratio, children ratio, and hyperlink ratio.

ii. Then, an algorithm standardizes the value of the assigned weights.

iii. After standardizing the weights, each node is considered a point in $\mathbb{R}^4$. Then, an algorithm explores all these nodes (points in $\mathbb{R}^4$) and computes the centroid. Subsequently, it builds the set of *candidate nodes*, which are those that include the DOM nodes (points) located farther than the centroid.

iv. Then, an algorithm analyzes the nodes in the set of *candidate nodes* in the following way:

   - It removes those nodes which are descendants of other nodes in the set if they share exactly the same text nodes as their ancestors.
   - For each node in the set of *candidate nodes*, it computes the ratio between words and tags. The algorithm selects as the main content the node with a higher ratio together with its siblings that belong to the set of *candidate nodes*.

v. Finally, as a post process, an algorithm analyzes the obtained main content node or nodes and removes groups of links that do not belong to the main content.

The definition of the main content and the four phases are described in the following sections.

6.2.1 The web page's main content

There might be discrepancies between two people in the identification of the main content in a rendered image of a web page. Despite that, it is usually an easy and trivial task for humans. However, in a web page represented with a DOM tree, there are usually several nodes whose subtree contains the nodes that correspond to the main content. In many cases, the hierarchy of the DOM nodes produces that one node and some of its descendants contain exactly the same text, images, etc. (e.g., a “DIV” element with only one child). In this case, a question is raised: which node should be chosen? This question can be answered based on a design policy.

The definitions of web page (4.1.1) and website (4.1.7) provided in Chapter 4 allow us to establish a definition for the main content.

Roughly, the main content can be defined as the information given by a web page except for the web template, metadata, and side information like advertisements or comments. Nevertheless, it should be highlighted that the main content of a web page has a subjective nature. A clear illustration of this subjectivity can be found in the web pages that include news articles with readers’ comments: the comments of the readers can be considered main content by some people, while others can consider them as not belonging to the main content. Hence, establishing a definition of the main content is a controversial issue.

In some approaches (see, e.g., [17, 51]), the main content of a web page is represented with a DOM node whose subtree is the smallest subtree that contains all the relevant nodes of the web page. However, this can be very imprecise because it can happen that the relevant nodes are distributed and mixed with other boilerplate subtrees (for instance, consider three sibling subtrees where two of them contain relevant content and the third is part of the template). Therefore, we consider the main content of a web page (see Definition 4.3.3) as a set of DOM nodes where the union of their subtrees contain all the relevant content, and they do not contain irrelevant content.

That definition is independent of any technique. It assumes the existence of a labelling $relevant(n)$ applied to the DOM tree leaves. The purpose of the labelling is to identify those leave nodes in the web page that should belong to the main content.

When the *relevant* labelling is provided, Definition 4.3.3 is particularly useful. For instance, when benchmarks are evaluated. Alternatively, in the case that the *relevant* labelling is not available, it must be approximated. The rest of the chapter proposes a method that creates an approximation of the *relevant* labelling automatically.

6.2.2 Weighting DOM nodes

This section proposes a metric that identifies the DOM nodes that with high probability are the root nodes of the web page’s main content. All the ideas used to develop the following properties have been empirically checked and validated with the training set of 105 web pages from the TeCo suite of benchmarks (see Chapter 13 for details).

Initially, an algorithm explores the DOM tree of the web page to compute and assign a weight to each DOM node that meets these criteria:

i. The DOM node is not a leaf node of the DOM tree.

ii. The type of the DOM node is an *element*[1] and its *tagName* property is not one of the following: "A", "BODY", "BR", "EM", "H1", "H2", "H3", "H4", "H5", "HEADER", "HR", "IFRAME", "NAV", "SPAN", "SCRIPT", and "UNDEFINED". The algorithm does not consider nodes of a type different from *element* (e.g., *text* nodes, *comment* nodes, *attribute* nodes, etc.).

iii. If the number of children of the "BODY" DOM node (measured as the number of element nodes with their *tagName* not listed in bullet 2) is larger than or equal to the depth of the web page from the "BODY" DOM node (following the same criterion), the main content of the web page is the union of the children of the "BODY" DOM node. It was not possible to validate this idea using the training set of 105 web pages from the TeCo benchmark suite because none of the selected web pages met the criterion. Thus, a set of 65 random web pages from the CleanEval dataset was selected. Eleven of the sixty five web pages met the criterion. Then, the percentage of the text of the whole web page that also belongs to the main content was measured. The average value obtained was 91.07%, that is, practically the main content of that web pages corresponds to the whole web page. It should be noted that this phenomenon was only observed in old web pages (more than 10 years old).

These criteria are based on the fact that, in the DOM model, *text* nodes are always inside *element* nodes, and also they are always leaves. Hence, a text (or image, etc.) can be always accessed by selecting the element node that contains it. In addition, it should be highlighted that there is no visible information (text, images, animations, etc.) contained in element nodes that are leaves. Thus, it is always possible to select the main content with an element node that is not a leaf. For this reason, leaves and nodes that are not elements are discarded. Besides, the *tagNames* listed above (bullet 2) and other tags subsuming them are discarded because they cannot be the main content. Finally, the last criterion is only applied to very wide web pages, namely web pages whose content is distributed between several branches. The determination of this criterion was done empirically, and

[1]http://www.w3.org/TR/domcore/#interface-element

it avoids selecting a branch of the DOM tree with a small concentration of text. It is important to note that the application of this criterion does not mean that the main content node has at most a third of the text of the whole web page, because the algorithm can select the parent node. This just discards the nodes which contain small amounts of text. It has been proved by empirical evaluation that the application of this criterion increases the precision by 1-5% (see Section 6.3.1).

It is a fact (see, e.g., [117]) that there is usually a high density of text and images in the main content of a website; however, in general, this density does not entail by itself the detection of the main content [51]. Definition 6.2.1 introduces several properties that, in combination with the text and image density, must be considered for proper detection of the main content. The combination of all these properties, which are objectively quantified, form a weighting that is able to identify the DOM nodes that form the main content.

Definition 6.2.1 (Node properties) *Given a web page $P = (N, A)$, every node $n \in N$ with $descendants(n) \neq \varnothing$ is rated according to the following properties:*

Word ratio: *It considers the number of words not included in a hyperlink and their depth. The algorithm assigns a higher value to the words nearer the node:*

$$wordRatio(n) = \sum_{k \in leaves(n)} words(k)/distance(k,n)$$
$$\text{where } parent(k).tagName \neq \text{“}A\text{”}$$

Hyperlink ratio: *It is computed considering the number of hyperlinks contained in the descendants of a node n:*

$$hyperlinkRatio(n) = \begin{cases} 1 & \text{if } links = 0 \\ 1/links & \text{if } links > 0 \end{cases}$$
$$\text{where } links = hyperlinks(subtree(n))$$

Children ratio: *It checks whether a node n has more than two children:*

$$childrenRatio(n) = \begin{cases} 0 & \text{if } childNodes(n) \leq 2 \\ 1 & \text{if } childNodes(n) > 2 \end{cases}$$

Position ratio: *It is computed using the following function:*

$$positionRatio(n,P) = \begin{cases} depth(n) & \text{if } depth(n) \leq maxDepth(P)/2 \\ maxDepth(p) \text{ - } depth(n) & \text{if } depth(n) > maxDepth(P)/2 \end{cases}$$

The *Word ratio* property defines a metric that takes account of the number of text words contained by a DOM node and its descendants. The *word ratio* of a node n subsumes the value of all its descendants. It is computed for each descendant as the number of text words it contains divided by the distance from the node to n. Hence, this metric promotes those DOM nodes with a high amount of text in their closer descendants. The closer the text is to a DOM node, the higher its *word ratio* is, and vice versa. Our experiments demonstrate that, on average, main content nodes contain 5.25 times more words than non-main content nodes. On average, main content nodes contain 3.01 words outside hyperlinks, while non-main content nodes contain 0.57 words.

The *Hyperlink ratio* property measures the number of hyperlinks contained in a DOM node and all its descendants. The rationale for this metric is that the main content of a web page usually contains fewer hyperlinks than other blocks in the web page, such as the footer, the main menu, etc. Hence, the *hyperlink ratio* promotes those DOM nodes with few hyperlinks among their descendants. The experiments carried out reveal that, on average, 71.89 % of the hyperlinks of a web page do not belong to the main content.

The *Children ratio* property encourages those nodes that contain more than two children. Typically, the main content in a web page contains several element nodes with text nodes between their descendants. In consequence, those DOM nodes with more than two children are assigned a higher value because they more likely contain text nodes between their descendants. Our experiments show that, as an average, 95.56 % of the main content nodes have more than 2 children.

The *Position ratio* property defines a metric to account for the depth of a DOM node in the DOM tree. The rationale behind this metric is that the main content of a web page is often positioned in the middle of the DOM tree. Thus, this metric stepwise penalizes those DOM nodes located at the top or the bottom of the DOM tree. Our experiments reveal that, in 60% of the web pages, the root node of the main content is located in the second tertile of the depth of the DOM tree.

After these properties are computed, they are assigned to each node in the DOM tree except for the *tagNames* listed in Section 6.2.2 (bullet 2). Higher values for these properties indicate that the DOM node more likely contains the main content of the web page. Since the properties are dissimilar from each other, they are not combined in one single ponderation. However, this chapter illustrates a novel method that compares the rated DOM nodes in order to determine which ones contain the main content of the web page with high probability. The technique is described in the following subsections.

6.2.3 Properties standardization

This page-level content extraction algorithm exposes a novel method to compute the differences between two DOM nodes. The comparison between two DOM nodes is based on the four ratios presented in Definition 6.2.1. These ratios represent a DOM node as a point in $\mathbb{R}^4$ (a four-dimensional Euclidean space). Hence, the comparison between the DOM nodes can be computed through the relative Euclidean distance between the points in $\mathbb{R}^4$ that represent the DOM nodes. The four ratios have been defined to ensure the differentiation of the main content nodes from the non-main content nodes. Therefore, in $\mathbb{R}^4$ the non-main content nodes will be relatively close to each other, while a DOM node containing the main content should be located farther from the other nodes.

This assumption is validated through empirical evaluation (see Section 6.3.1) and is based on the fact that the value of the four properties presented in Definition 6.2.1 is significantly different in main content nodes than in non-main content nodes. Note that a main content node usually contains a considerable amount of text between its descendants, a low number of hyperlinks between its descendants, several children, and it is usually positioned in the first half of the DOM tree. As most of the nodes in the DOM tree are not-main content nodes, most likely they do not meet these properties, thus the centroid of all the nodes of the DOM tree is located with high probability near them in $\mathbb{R}^4$ and, therefore, it is located far from the main content nodes.

In order to ensure that the impact of the four ratios in Section 6.2.2 is the same on the distance measurement, they must be standardized [70]. The standardization process is done by replacing the value of a ratio with the difference between that value and the average of the values taken by it divided by the standard deviation.

Definition 6.2.2 (Node standardization) *The different ratios of a node r are standardized with the following formula:*

$$r_i = (r_i - \overline{r_i})/s_{r_i} \tag{6.1}$$

where $\overline{r_i}$ is the average of the values taken by r_i, and s_{r_i} is the standard deviation.

In the following, DOM nodes are represented as points in $\mathbb{R}^4$.

Definition 6.2.3 (DOM nodes representation in $\mathbb{R}^4$) *A DOM node A is represented in a Euclidean space $\mathbb{R}^4$ with a quadruple (a, b, c, d), where a, b, c, d are the four ratios assigned in Definition 6.2.1. Two nodes $A = (a, b, c, d)$ and $A' = (a', b', c', d')$ are the same node if and only if $a = a' \wedge b = b' \wedge c = c' \wedge d = d'$.*

The implementation of the DOM nodes as points in $\mathbb{R}^4$ based on the ratios affords interesting advantages compared to other perspectives. For instance, the page-level menu detection technique described in Chapter 5 proposes several ratios to identify the DOM node that corresponds to the menu, but the importance of a DOM node is computed with the sum of its ratios. This approach does not allow us to distinguish between two DOM nodes, i.e., $(4, 3, 2, 1) = (1, 2, 3, 4)$ because $4 + 3 + 2 + 1 = 1 + 2 + 3 + 4$. In contrast, the representation of the DOM nodes as points in the Euclidean space actually allows us to distinguish between any combination of ratios. In addition, it defines a distance between the DOM nodes: the Euclidean distance.

Definition 6.2.4 (Euclidean distance) *The Euclidean distance between a node A and a node B for n ratios (in $\mathbb{R}^n$) is computed with the following formula:*

$$eucli\_distance(A, B) = \sqrt{\sum_{i=0}^{n} (A.ratio[i] - B.ratio[i])^2} \tag{6.2}$$

The use of the Euclidean distance also provides a significant advantage: it allows us to determine how far is one point from another although they have the same values for different ratios. This feature significantly reduces the number of experiments needed to train the algorithm. In contrast, the menu detection technique described in Chapter 5 has to limit the number of experiments because it produces a combinatorial explosion, e.g., $0.1 * ratio1 + 0.2 * ratio2 + ...$, with $0.2 * ratio1 + 0.2 * ratio2 + ...$, and so on.

6.2.4 *c–SET* computation

Th *c–SET* process described in this section is similar to a clustering technique. Bing [70] describes a clustering technique as a process that organizes data instances into similarity groups, called clusters such that the data instances in the same cluster are similar to each other. In this case, the algorithm only creates one cluster (*c–SET*), which is formed with the DOM nodes that more likely contain the main content of the web page. Then, the algorithm analyzes the *c–SET* to identify which DOM nodes contain the main content.

In the first step, the centroid of the nodes is computed. For the rated DOM nodes, the centroid corresponds to the arithmetic mean position of all the points in $\mathbb{R}^4$ represented by them.

The computation of the centroid of all the rated nodes is described in Algorithm 3. The centroid is a DOM node surrounded (in $\mathbb{R}^4$) by other nodes that are probably non-main content nodes. Note that, for non-content nodes, the value of their properties (see Definition 6.2.1) is close to zero, so with high probability they must be located near the coordinate axes. In consequence, the DOM nodes in the set of candidate nodes (*c–SET*) are those DOM nodes located farther from the centroid in $\mathbb{R}^4$, because the value of their properties is substantially high. The *c–SET* is built by Algorithm 4 by measuring the distance to the centroid from all the rated DOM nodes in the web page. Then, it selects the c nodes farther from it. The value of c has been determined with an empirical evaluation (see Section 6.3.1).

In subsequent phases, the nodes that form the *c–SET* must be analyzed to select the node that more likely contains the main content.

Algorithm 3 Centroid computation

```
Input: A set of rated DOM nodes $ratedNodes = \{n_1 \ldots n_i\}$
Output: The centroid $c$ of $ratedNodes$

begin
    for $prop = 0$ to 3
        $c.ratio[prop] = (\sum_{node=1}^{i} n_{node}.ratio[prop])/i$;
    return $c$;
end
```

Algorithm 4 *c–SET* computation

```
Input: A set of rated DOM nodes $ratedNodes$, its centroid $cent$, and its size $c$
Output: A set of DOM nodes $c\text{–}SET$

begin
    foreach ($n_1$ in $ratedNodes$)
        $sum = 0$;
        for $i = 0$ to 3
            $sum = sum + (n_1.ratio[i] - cent.ratio[i])^2$;
        $n_1.distance = \sqrt{sum}$;
    $c\text{–}SET = \emptyset$
    for $i = 1$ to $c$
        $node = n \in ratedNodes\ .\ \nexists n' \in ratedNodes, n'.distance > n.distance$;
        $ratedNodes = ratedNodes \backslash \{n\}$;
        $c\text{–}SET = c\text{–}SET \cup \{n\}$;
    return $c\text{–}SET$;
end
```

Algorithm 5 *c–SET* reduction

```
Input: A set of candidate DOM nodes $c\text{–}SET$
Output: A set of candidate DOM nodes $c\text{–}SET$

begin
    foreach ($n_1$ in $c\text{–}SET$)
        foreach ($n_2$ in $c\text{–}SET$)
            if $n_1.innerText == n_2.innerText$ and $n_1 \in ancestors(n_2)$
                $c\text{–}SET = c\text{–}SET \backslash \{n_2\}$;
    return $c\text{–}SET$;
end
```

6.2.5 Selecting the main content nodes

In this section, the nodes of the *c–SET* are analyzed in order to identify the main content nodes.

First of all, a reduction of the number of elements in the *c–SET* is performed. Algorithm 5 explores the *c–SET* and checks whether two or more nodes contain the same text nodes. If this happens, one node is a descendant of the other. Given two nodes in the *c–SET*, one of them is removed if it is a descendant from the other and both contain exactly the same text nodes between their descendants. The aim of selecting the ancestor (and consequently removing the descendant) is to avoid missing non-textual information, such as animations, images, etc. that surround the text nodes.

Algorithm 5 explores the *c–SET* and checks whether there are pairs of nodes (where one is an ancestor from the other) that contain exactly the same text nodes. If this happens, the algorithm removes the descendant node. Then, the remaining DOM nodes in the *c–SET* are explored by Algorithm 6 to select the nodes that correspond to the main content as follows:

- It computes the ratio between text words and tags for all the nodes in the *c–SET*.
- It selects as the main content node the node with the highest ratio if it does not have any siblings that belong to the *c–SET*. If there is a tie between several nodes, all of them are selected as the main content nodes.
- If there are siblings of the node with the highest ratio in the *c–SET*, then the algorithm selects all of them as the main content nodes.

Algorithm 6 Main content selection

```
Input: A set of candidate DOM nodes $c\text{–}SET$
Output: A set of DOM nodes $mainContent$

begin
    $maxRatio = 0$;
    $maxNode = null$;
    foreach ($n$ in $c\text{–}SET$)
        $n.textPond = |n.innerText|/|n.tags|$;
        if $n.textPond > maxRatio$
            $maxRatio = n.textPond$;
            $maxNode = n$;
    $siblings = getSiblings(maxNode, c\text{–}SET)$;
    if $siblings == \emptyset$
        $mainContent = maxNode$;
    else
        $mainContent = \{n\} \cup siblings$;
    return $mainContent$;
end
```

Function $getSiblings(n, set)$ of Algorithm 6 returns the sibling nodes of n that belong to the set of nodes *set*.

6.2.6 Final post-process

We observed in our experiments that, in around 5% of the web pages, the resulting main content includes groups of links that should not have been extracted because they do not belong to the main content of the web page (e.g., links to other sections of the website, breadcrumbs, etc.). With a quite cheap post-process, those groups of links can be removed, resulting in a precision improvement. The main content nodes returned by Algorithm 6 are explored by Algorithm 7, which removes the groups of links that do not belong to the main content, if any. The process is done in the following way:

- For all the hyperlink nodes in the mainContent set, the algorithm checks whether they do not have sibling nodes and whether the number of words of their descendants is lower than a specified threshold called *max words*.
- Then, for the nodes that fulfil those conditions, the algorithm checks if the siblings of their parent nodes share the same structure.
- Finally, the nodes found in the previous step that share the same structure are removed from the main content set.

Algorithm 7 Post-process

```
Input: A set of DOM nodes $mainContent$, and a max words threshold $mw$
Output: A set of DOM nodes $mainContent$ excluding some groups of links

begin
  foreach ($n$ in $mainContent$)
    if ($n.tagName ==$ “$A$”)
      if ($getWords(n) < mw$ and $|n.parentNode.children| == 1$)
        $found = false$;
        $siblings = getSiblings(n.parentNode, mainContent)$;
        foreach ($i$ in $siblings$)
          if ($|i.children| == 1$)
            if ($i.tagName == n.parentNode.tagName$)
              if ($i.children[0].tagName ==$ “$A$”)
                $found = true$;
                $mainContent = mainContent - \{i\}$;
        if ($found == true$)
          $mainContent = mainContent - \{n.parentNode\}$;
  return $mainContent$;
end
function $getWords(n)$
  return $n.textContent.split($/ \s+/$).length$;
end function
```

6.3 Implementation

This technique, as all the techniques presented in this thesis, has been implemented as a WebExtension, which is compatible with Mozilla-based and Chromium-based browsers, among others. The add-on shows a single button in the browser. When that button is pressed, it does the required actions and extracts the main content of the web page loaded by the browser, which is automatically displayed[2] (and it can be saved). If the button is pressed again, the original web page is displayed.

The evaluation of the method was done using the TeCo benchmark suite (see Chapter 13). As it is a suite of real and heterogeneous benchmarks with different layouts and page structures, it is an optimum tool to evaluate the technique. Moreover, it not only consists of textual information but also includes pictures, animations, embedded media, etc. In the experiments, we used 45 web pages (the TeCo subset of evaluation) to evaluate the technique. We also used 15 benchmarks from the TeCo training subset (105 web pages) to compute the optimal size of the $c\text{–}SET$. Another 15 web pages of the TeCo training subset were used to evaluate the post-process phase. Additionally, the remaining 75 web pages from the training subset were used to evaluate the metrics proposed in Section 6.2.2.

Most content extraction techniques in the literature use recall, precision, and F1 metrics of the retrieved words. This is somehow limited because it assumes that the main content of the web page is only text. In our evaluation, we overcome this limitation by measuring the retrieved DOM nodes. Therefore, we perform an evaluation that considers that the main content can include text, video, images, animations, and any other content. Moreover, in order to compare our technique with the related work, we have also evaluated the technique using retrieved words as metric. Besides retrieved words, a wide range of different metrics (see, e.g., [15, 20, 106, 115]) are used for the evaluation and comparison of content extraction algorithms. Therefore, to perform a proper comparison of our technique with other mainstream algorithms, we also adopted and implemented some of these metrics. The obtained results of this comparison are detailed in Chapter 12.

[2]The nodes that do not belong to the main content are properly hidden by changing their *visibility* and *display* attributes to *hidden* or *none*, respectively. Hence, the result is the isolation of the main content, which appears in the same place as on the original web page.

	DOM nodes			Words				
Size	Recall	Precision	F1	Recall	Precision	F1	Avg. nodes	Runtime
1	76.65 %	82.85 %	74.47 %	85.47 %	84.62 %	83.15 %	0.93	333 ms.
2	87.59 %	87.58 %	83.22 %	96.53 %	91.27 %	92.60 %	1.00	327 ms.
3	87.54 %	91.60 %	85.75 %	96.32 %	94.49 %	94.59 %	1.00	330 ms.
4	77.83 %	93.69 %	79.30 %	90.07 %	95.33 %	90.88 %	1.00	326 ms.
5	62.64 %	96.00 %	66.41 %	76.86 %	96.75 %	80.56 %	1.00	338 ms.
6	58.51 %	96.00 %	62.36 %	75.36 %	96.75 %	79.57 %	1.13	339 ms.
7	54.89 %	93.77 %	56.94 %	68.56 %	96.37 %	74.26 %	1.27	340 ms.
8	48.21 %	87.11 %	49.45 %	63.93 %	89.70 %	68.79 %	1.33	339 ms.

Table 6.1: Determining the optimal size of the *c–SET*

6.3.1 Empirical evaluation

First, it should be determined the optimal size of the set of candidate nodes (the *c* value of the *c–SET*. See Section 6.2.4). Several *c–SET* sizes were tested using the training subset of the TeCo benchmark suite.

Table 6.1 shows the obtained results of the experiments with a *c–SET* size from 1 to 8. Each row of the table shows the average `Recall`, `Precision`, and `F1` of executing the algorithm for all the benchmarks in the training subset of the TeCo benchmark suite with a different value for *c* in the *c–SET*. The table also contains the obtained average results for both, retrieved DOM nodes and retrieved text words.

As shown in Table 6.1, best results are produced using an *n–SET* with n=3, as this value obtains the best `F1` value in both, retrieved DOM nodes (85.75 %) and retrieved words (94.59 %). Hence, even though the size of the *n–SET* is configurable, a 3–*SET* has been used by default.

It can be observed that the size of the *c–SET* has a significant impact on recall and precision. On the other hand, it has almost no impact on the performance, since the average runtimes are similar for all the tested *c–SETs* (e.g., the average runtime of the 1–*SET* and the average runtime of the 8–*SET* only differ in 6 milliseconds).

Moreover, it can be observed that bigger *c–SETs* do not necessarily obtain higher F1 values. This occurs because an increment of the size of the c-SET also increases the probability of selecting one or several nodes that are descendants from the main content root node or nodes. Thus, there are more possible nodes that Algorithm 4 can select. In some benchmarks it can select descendants from the root node or nodes, so the recall decreases.

Besides, the impact of the post-process phase was also evaluated using a 3–*SET* for the training subset of the TeCo benchmark suite. The best values for the `max.words` threshold were determined using values from 1 to 8. For each `max. words` value, the best `F1` results were selected for both, DOM nodes and retrieved text words.

The results of the experiments conducted to determine the `max. words` threshold can be observed in Table 6.2. Each row represents the best combination of results for each `max. words` value. It can be observed the minor impact of the post-process phase in both, retrieved DOM nodes and retrieved words. A `max. words` value equal to 1 improves the `F1` value in 0.02 % for retrieved DOM nodes, while a `max. words` value equal to 3 improves the `F1` value in 0.12 % for retrieved words. It is not a substantial gain, but the post-process phase especially improves the precision of the algorithm. It should be highlighted that the runtime of the algorithm is not significantly increased by the post-process phase.

	DOM nodes			Words			
Max. Words	Recall	Precision	F1	Recall	Precision	F1	Runtime
1	87.61 %	91.57 %	85.77 %	96.53 %	94.30 %	94.55 %	340 ms.
2	87.55 %	91.59 %	85.76 %	96.37 %	94.42 %	94.56 %	339 ms.
3	87.54 %	91.60 %	85.75 %	96.32 %	94.49 %	94.59 %	330 ms.
4	87.52 %	91.59 %	85.74 %	96.24 %	94.48 %	94.55 %	338 ms.
5	87.50 %	91.60 %	85.74 %	96.14 %	94.53 %	94.54 %	329 ms.
6	87.48 %	91.59 %	85.73 %	95.82 %	94.48 %	94.37 %	333 ms.
7	87.48 %	91.59 %	85.73 %	95.82 %	94.48 %	94.37 %	326 ms.
8	87.48 %	91.59 %	85.73 %	95.82 %	94.48 %	94.37 %	335 ms.
No post-process	87.61 %	91.54 %	85.75 %	96.53 %	94.17 %	94.47 %	327 ms.

Table 6.2: Determining `max. words` threshold

Algorithm evaluation

As mentioned above, to evaluate the technique, several experiments were performed with the 45 benchmarks of the evaluation subset of the TeCo benchmark suite (see Chapter 13). For each benchmark, the algorithm computed the number of retrieved DOM nodes and the number of DOM nodes correctly classified as the main content. It also computed the `Recall`, `Precision`, and `F1` of the retrieved DOM nodes and the retrieved words. Additionally, the `Runtime` in seconds was also registered.

In Table 6.3, column `Total` shows the total number of DOM nodes of the key page; column `Gold` indicates the number of DOM nodes of the gold standard; column `Retr.` is the number of retrieved DOM nodes; column `Correct` represents the number of retrieved DOM nodes that belong to the gold standard; columns `Rec.`, `Prec.`, and `F1` are the recall, precision, and F1 respectively. In Table 6.4, column `Gold` shows the total number of words on the key page; column `Retr.` is the number of retrieved words; column `Correct` indicates the number of retrieved words that belong to the gold standard; columns `Rec.`, `Prec.`, and `F1` are the recall, precision, and F1 respectively. In both tables, column `Runtime` represents the runtime for that benchmark (in milliseconds).

	Number of nodes				DOM nodes			
Benchmark	**Total**	**Gold**	**Retr.**	**Correct**	**Rec.**	**Prec.**	**F1**	**Runtime**
www.jdi.org.za	619	199	225	134	67.34 %	59.56 %	63.21 %	55 ms.
www.premiere-urgence.org	480	32	31	31	96.88 %	100.00 %	98.42 %	15 ms.
www.indiangaming.org	575	148	147	147	99.32 %	100.00 %	99.66 %	18 ms.
hispalinux.es	501	144	143	143	99.31 %	100.00 %	99.65 %	25 ms.
www.gktw.org	767	130	20	20	15.38 %	100.00 %	26.66 %	47 ms.
www.apnic.net	598	79	75	75	94.94 %	100.00 %	97.40 %	12 ms.
www.unicef.org	1037	381	381	378	99.21 %	99.21 %	99.21 %	54 ms.
www.klimabuendnis.org	851	134	133	133	99.25 %	100.00 %	99.62 %	28 ms.
www.isoc-es.org	259	56	43	43	76.79 %	100.00 %	86.87 %	6 ms.
biztechmagazine.com	1892	454	109	109	24.01 %	100.00 %	38.72 %	119 ms.
www.eeo.com.cn	834	119	247	118	99.16 %	47.77 %	64.48 %	52 ms.
www.wishtv.com	2167	343	345	342	99.71 %	99.13 %	99.42 %	89 ms.
news.mit.edu	2117	128	133	127	99.22 %	95.49 %	97.32 %	212 ms.
asia.nikkei.com	869	116	57	57	49.14 %	100.00 %	65.90 %	69 ms.
www.rcnky.com	1738	112	104	104	92.86 %	100.00 %	96.30 %	208 ms.
news.discovery.com	2826	791	165	165	20.86 %	100.00 %	34.52 %	458 ms.
www.kathimerini.gr	1825	117	113	113	96.58 %	100.00 %	98.26 %	61 ms.
news.un.org	1726	59	58	58	98.31 %	100.00 %	99.15 %	98 ms.
frances.forosactivos.net	785	495	636	494	99.80 %	77.67 %	87.36 %	58 ms.
www.wysiwygwebbuilder.com	3936	3201	3197	3197	99.88 %	100.00 %	99.94 %	745 ms.
www.3dprintforums.com	1040	748	570	570	76.20 %	100.00 %	86.49 %	62 ms.
www.strangehorizons.com	631	403	402	402	99.75 %	100.00 %	99.87 %	48 ms.
communities.apple.com	3136	1306	2643	1305	99.92 %	49.38 %	66.10 %	280 ms.
www.sloweurope.com	4193	2789	2785	2785	99.86 %	100.00 %	99.93 %	1018 ms.
community.ricksteves.com	2057	1177	1176	1176	99.92 %	100.00 %	99.96 %	412 ms.
hackercombat.com	1711	698	913	697	99.86 %	76.34 %	86.53 %	231 ms.
www.scbwi.org	876	506	505	505	99.80 %	100.00 %	99.90 %	44 ms.
johngardnerathome.info	395	188	187	187	99.47 %	100.00 %	99.73 %	13 ms.
www.annmalaspina.com	392	84	13	13	15.48 %	100.00 %	26.81 %	19 ms.
foodsense.is	330	192	229	190	98.96 %	82.97 %	90.26 %	16 ms.
sites.google.com	372	85	87	84	98.82 %	96.55 %	97.67 %	35 ms.
whatever.scalzi.com	1648	243	242	242	99.59 %	100.00 %	99.79 %	221 ms.
www.javiercelaya.es	740	57	49	49	85.96 %	100.00 %	92.45 %	64 ms.
diarium.usal.es	604	524	523	523	99.81 %	100.00 %	99.90 %	29 ms.
www.jameslovelock.org	653	174	185	173	99.43 %	93.51 %	96.38 %	21 ms.
www.cipri.info	933	556	401	401	72.12 %	100.00 %	83.80 %	67 ms.
naranjascarcaixent.com	290	141	147	140	99.29 %	95.24 %	97.22 %	10 ms.
www.technicalbookstoreonline.com	2959	2002	2273	2001	99.95 %	88.03 %	93.61 %	402 ms.
www.floridarealestatecollege.com	1023	65	35	35	53.85 %	100.00 %	70.00 %	180 ms.
www.basf.com	827	62	814	61	98.39 %	7.49 %	13.92 %	22 ms.
www.mcphersonoil.com	831	225	10	10	4.44 %	100.00 %	8.51 %	36 ms.
www.thirteenhou.com	1217	1073	1083	1072	99.91 %	98.98 %	99.44 %	100 ms.
www.embalajesterra.com	2342	470	702	469	99.79 %	66.81 %	80.04 %	114 ms.
www.crypto.ch	338	68	92	65	95.59 %	70.65 %	81.25 %	9 ms.
www.shopbookshop.com	1727	387	1008	382	98.71 %	37.90 %	54.77 %	124 ms.
Average	1281.49	476.91	520.80	433.89	84.95 %	89.84 %	81.70 %	133 ms.

Table 6.3: Evaluation of the precision, recall, F1, and runtime for retrieved DOM nodes

The experiments reveal an average F1 of 81.69% for retrieved DOM nodes, and an average F1 of 92.45% for retrieved words[3].

[3]The average F1 on the bottom of the table represents the average of the values in column F1, and not the F1 computed using the average precision and the average recall.

	Number of words			Words			
Benchmark	**Gold**	**Retr.**	**Correct**	**Rec.**	**Prec.**	**F1**	**Runtime**
www.jdi.org.za	313	301	284	90.73 %	94.35 %	92.50 %	55 ms.
www.premiere-urgence.org	134	134	134	100.00 %	100.00 %	100.00 %	15 ms.
www.indiangaming.org	145	145	145	100.00 %	100.00 %	100.00 %	18 ms.
hispalinux.es	514	514	514	100.00 %	100.00 %	100.00 %	25 ms.
www.gktw.org	478	178	178	37.24 %	100.00 %	54.27 %	47 ms.
www.apnic.net	245	245	245	100.00 %	100.00 %	100.00 %	12 ms.
www.unicef.org	120	118	118	98.33 %	100.00 %	99.16 %	54 ms.
www.klimabuendnis.org	258	258	258	100.00 %	100.00 %	100.00 %	28 ms.
www.isoc-es.org	69	64	64	92.75 %	100.00 %	96.24 %	6 ms.
biztechmagazine.com	815	538	538	66.01 %	100.00 %	79.53 %	119 ms.
www.eeo.com.cn	43	59	43	100.00 %	72.88 %	84.31 %	52 ms.
www.wishtv.com	786	786	786	100.00 %	100.00 %	100.00 %	89 ms.
news.mit.edu	1000	1000	1000	100.00 %	100.00 %	100.00 %	212 ms.
asia.nikkei.com	642	587	587	91.43 %	100.00 %	95.52 %	69 ms.
www.rcnky.com	935	935	935	100.00 %	100.00 %	100.00 %	208 ms.
news.discovery.com	767	622	622	81.10 %	100.00 %	89.56 %	458 ms.
www.kathimerini.gr	737	734	734	99.59 %	100.00 %	99.79 %	61 ms.
news.un.org	303	303	303	100.00 %	100.00 %	100.00 %	98 ms.
frances.forosactivos.net	169	255	169	100.00 %	66.27 %	79.71 %	58 ms.
www.wysiwygwebbuilder.com	2115	2115	2115	100.00 %	100.00 %	100.00 %	745 ms.
www.3dprintforums.com	347	270	270	77.81 %	100.00 %	87.52 %	62 ms.
www.strangehorizons.com	3559	3559	3559	100.00 %	100.00 %	100.00 %	48 ms.
communities.apple.com	608	1216	608	100.00 %	50.00 %	66.67 %	280 ms.
www.sloweurope.com	804	804	804	100.00 %	100.00 %	100.00 %	1018 ms.
community.ricksteves.com	589	589	589	100.00 %	100.00 %	100.00 %	412 ms.
hackercombat.com	383	442	383	100.00 %	86.65 %	92.85 %	231 ms.
www.scbwi.org	247	247	247	100.00 %	100.00 %	100.00 %	44 ms.
johngardnerathome.info	1375	1375	1375	100.00 %	100.00 %	100.00 %	13 ms.
www.annmalaspina.com	114	113	113	99.12 %	100.00 %	99.56 %	19 ms.
foodsense.is	442	493	442	100.00 %	89.66 %	94.55 %	16 ms.
sites.google.com	54	54	54	100.00 %	100.00 %	100.00 %	35 ms.
whatever.scalzi.com	1151	1151	1151	100.00 %	100.00 %	100.00 %	221 ms.
www.javiercelaya.es	359	357	357	99.44 %	100.00 %	99.72 %	64 ms.
diarium.usal.es	869	869	869	100.00 %	100.00 %	100.00 %	29 ms.
www.jameslovelock.org	1308	1310	1308	100.00 %	99.85 %	99.92 %	21 ms.
www.cipri.info	1281	1221	1221	95.32 %	100.00 %	97.60 %	67 ms.
naranjascarcaixent.com	73	73	73	100.00 %	100.00 %	100.00 %	10 ms.
www.technicalbookstoreonline.com	873	942	873	100.00 %	92.68 %	96.20 %	402 ms.
www.floridarealestatecollege.com	310	301	301	97.10 %	100.00 %	98.53 %	180 ms.
www.basf.com	128	237	128	100.00 %	54.01 %	70.14 %	22 ms.
www.mcphersonoil.com	319	96	96	30.09 %	100.00 %	46.26 %	36 ms.
www.thirteenhou.com	1337	1337	1337	100.00 %	100.00 %	100.00 %	100 ms.
www.embalajesterra.com	111	203	111	100.00 %	54.68 %	70.70 %	114 ms.
www.crypto.ch	185	186	185	100.00 %	99.46 %	99.73 %	9 ms.
www.shopbookshop.com	241	385	241	100.00 %	62.60 %	77.00 %	124 ms.
Average	614.56	616.02	588.16	94.58 %	93.85 %	92.61 %	133 ms.

Table 6.4: Evaluation of the precision, recall, F1, and runtime for retrieved words

Other techniques that used heterogeneous websites for evaluation obtained the following F1 results for retrieved words: Insa et al. obtain 74 % [51], Gottron et al. 77 % [46], and Shanchan et al. 82 % [118]. By contrast, other techniques have been evaluated using prepared datasets [85] (MSS),

	DOM nodes			Words			
Benchmark type	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**F1**	**Runtime**
Institutions / Associations	83.50 %	95.42 %	85.81 %	91.01 %	99.37 %	93.57 %	29 ms.
Media /Communication	75.55 %	93.06 %	76.62 %	93.13 %	99.42 %	95.75 %	152 ms.
Forum / Social	97.22 %	89.20 %	91.74 %	97.53 %	87.90 %	91.05 %	322 ms.
Personal websites / Blogs	85.52 %	97.00 %	87.42 %	99.32 %	98.83 %	99.04 %	54 ms.
Companies / Shops	83.32 %	73.90 %	66.53 %	91.91 %	84.83 %	84.28 %	111 ms.

Table 6.5: Results of the performed experiments grouped by benchmark type

RSS feeds, or prepared websites[4]. Unfortunately, a fair comparison of different techniques is not possible if they use different evaluation datasets or if the structure of the website is known before extracting the content. For instance, techniques that evaluated prepared websites usually reported high F1 values (Zhao et al. 88 % [69], Adam et al. obtain 93 % [1], Qureshi et al. 94 % [92], and Pasternack et al. 95 % [85]). However, these F1 results are substantially reduced if heterogeneous websites are used. We performed a comparison of several content extractors in the literature using the same benchmark suite and the same metrics. It is presented in Section 12.4.

Table 6.5 shows the average results grouped by benchmark type. We can observe that the results may differ substantially depending on the type of benchmark. For instance, the average F1 obtained for retrieved DOM nodes by the `Forum / Social` benchmarks is almost 25% higher than the average F1 obtained by the `Companies / Shops` benchmarks. On the other hand, for retrieved words, the average F1 obtained is fairly high for all benchmark types. The lowest value is the 84.28% obtained by the `Companies / Shops` benchmarks. It should be highlighted that the average precision obtained by the `Institutions / Associations`, `Media / Communication`, and `Personal websites / Blogs` benchmarks is close to 100% for both, retrieved DOM nodes and retrieved words. Regarding the runtime, we can observe significant differences between the different benchmark types. While the average runtime of the `Institutions / Associations` is around 30 ms. on average, the runtime of the `Forum / Social` benchmarks is higher than 300 ms.

Runtime analysis

For each web page in the evaluation subset of the TeCo benchmark suite, Figure 6.2 presents the relationship between the number of DOM nodes of the key page and the time (in milliseconds) needed to extract the main

[4]web pages that were generated automatically and share the same template.

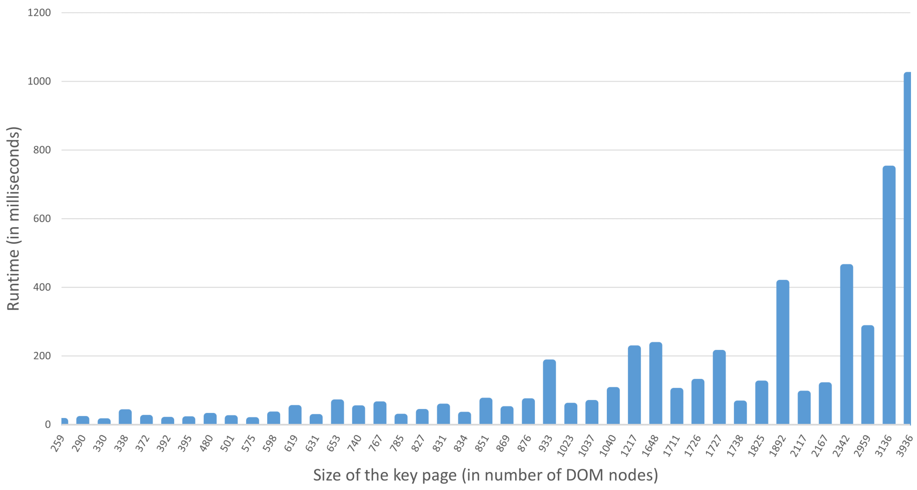


Figure 6.2: Relation between the size of the web page and the runtime

content. More than half of the runtime (approximately 56%) is used by Algorithm 3, whose asymptotic cost is $\mathcal{O}(n^2)$, being n the number of rated DOM nodes. Additionally, more than 27% of the runtime is used by the node properties computation algorithms. Hence, the rest of the algorithms described in this chapter use a small fraction of the runtime (between 3% and 5% each).

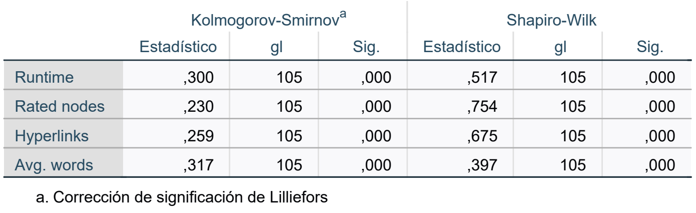
It should be highlighted that more than 95% of the benchmarks took less than half a second, while 66% of them took less than 100 milliseconds. There was only one benchmark whose runtime took more than 1 second.

As can be noted in Figure 6.2, for most web pages, the tool extracts the main content in less than 200 milliseconds.

It can be observed in the figure that usually the highest runtimes correspond to the web pages with the higher number of DOM nodes, however, some benchmarks do not obtain the runtime expected according to their size. For this reason, we explored other variables that could be related to the runtime of the algorithm.

Hence, we computed a statistical analysis of some variables that, in our opinion, could also be related to the runtime. The analyzed variables are:

- The `number of rated DOM nodes` (see Section 6.2.1). The choice of this variable, as in Chapter 5, is based on the assumption that the cost of computing the weights assigned to the DOM nodes is significant for the runtime. Thus, the more weighted nodes, the higher runtime.



	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Runtime	,300	105	,000	,517	105	,000
Rated nodes	,230	105	,000	,754	105	,000
Hyperlinks	,259	105	,000	,675	105	,000
Avg. words	,317	105	,000	,397	105	,000

a. Corrección de significación de Lilliefors

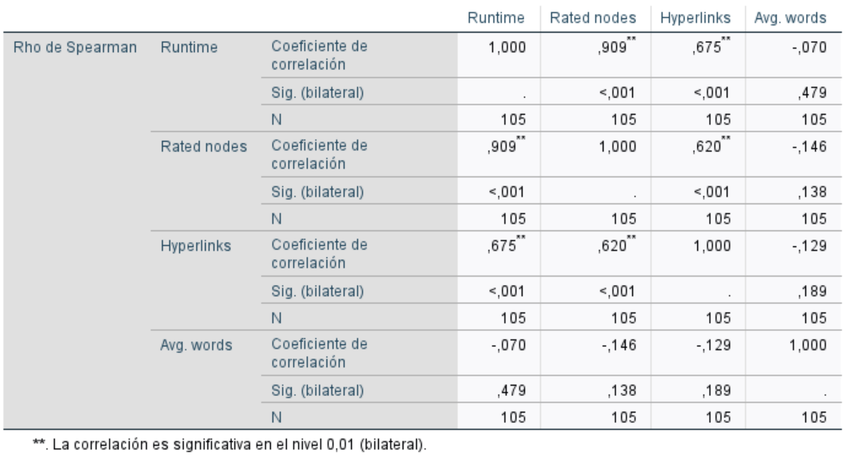
Figure 6.3: Result of the normality test for page-level ConEx

- The `number of hyperlinks in the retrieved main content` before the final post-process phase. This variable is based on the assumption that the final post-process phase is relevant to the total runtime of the technique. As this phase only affects the hyperlink nodes, the number of hyperlinks could increase the runtime.

- The `average amount of text words per weighted DOM node` (measured in words). This variable was selected because the computation of the `word ratio` is the most complex of the 4 node properties. Therefore, we supposed that computing the `word ratio` of the nodes with more text would affect the runtime significantly.

As in Chapter 5, we used IBM SPSS Statistics to analyze the relationship between these variables. We also conducted the analysis using the test subset of the TeCo benchmark suite, formed by 105 web pages (see Chapter 13). First of all, we computed Table 6.3 to check whether the variables were normally distributed. As we can observe in the table, the `sample size` (column gl) is 105, thus, the appropriate test is Kolmogorov-Smirnov.

In this case, the `significance` (column Sig.) of all variables is less than 0.05; therefore, the correlation coefficient can be computed through the Spearman test because the variables are not distributed normally. Figure 6.4 shows the result of the Spearman test for these variables. We can observe the correlation coefficient between all the variables and the runtime in the first row of the table. As in the case of the menu detection algorithm (see Chapter 5), the correlation coefficient of the `number of rated DOM nodes` variable is close to 1. Therefore, this variable has a very strong relationship with the `runtime` of the algorithm. Moreover, we can observe that the correlation coefficient between the `number of hyperlinks in the retrieved main content` variable and the `runtime` is close to 0.7, which denotes also a strong relationship between them. The correlation coefficient value of the third variable, `average amount of text words per`



			Runtime	Rated nodes	Hyperlinks	Avg. words
Rho de Spearman	Runtime	Coeficiente de correlación	1,000	,909**	,675**	-,070
		Sig. (bilateral)	.	<,001	<,001	,479
		N	105	105	105	105
	Rated nodes	Coeficiente de correlación	,909**	1,000	,620**	-,146
		Sig. (bilateral)	<,001	.	<,001	,138
		N	105	105	105	105
	Hyperlinks	Coeficiente de correlación	,675**	,620**	1,000	-,129
		Sig. (bilateral)	<,001	<,001	.	,189
		N	105	105	105	105
	Avg. words	Coeficiente de correlación	-,070	-,146	-,129	1,000
		Sig. (bilateral)	,479	,138	,189	.
		N	105	105	105	105

**. La correlación es significativa en el nivel 0,01 (bilateral).

Figure 6.4: Result of the Spearman test for page-level ConEx

`weighted DOM node` is close to 0, therefore, this variable does not have a significant relationship with the `runtime`.

6.4 Conclusions

This chapter describes a novel page-level content extraction technique from heterogeneous web pages. The technique shows high precision and a low runtime compared to other block detection techniques (see Chapter 12) because it is page-level, so it only needs to load one web page to infer the main content. The main innovation of the technique is the way in which the DOM node features are represented, as points in a 4-dimensional Euclidean space $\mathbb{R}^4$, and how the main content is detected in such a space. The features on which the metrics are based and their ranked values have demonstrated to be accurate in the isolation of the nodes that contain the main content. For that reason, the main content information is contained with high probability in those DOM nodes whose features are significantly different to most of the nodes. Representing the features of the DOM nodes as points in $\mathbb{R}^4$ is useful to quickly and easily identify the candidate nodes by means of the standard Euclidean distance between the points.

The main content of the websites can be formed by several different elements apart from text, such as images, animations, videos, embedded content, etc. Usually, the main content is formed by text combined with other elements. One of the main benefits of this technique is that, as it is based on DOM nodes and analyzes other features apart from text-based attributes, it not only extracts the text as the main content but also other main content elements. Therefore, the algorithm outputs one or several DOM nodes and their subtrees as the main content of the analyzed website.

6.5 Contributions

The page-level content extraction technique presented in this chapter provides several contributions that can be implemented by many systems, especially by other block detection techniques.

It describes a set of features (on which the metrics are based) that have been demonstrated to be accurate to differentiate between main content nodes and the rest of the nodes. Based on this set of features, the technique introduces the representation of a DOM node as a point in a 4-dimensional Euclidean space $\mathbb{R}^4$. This representation has been demonstrated to be useful to easily compare DOM nodes.

An algorithm that infers the main content is also provided. The algorithm selects the node or nodes that most likely include the main content based on the Euclidean distance between a set of DOM nodes represented as points in a 4-dimensional Euclidean space $\mathbb{R}^4$.

The functional implementation of the technique as a WebExtension is also an important contribution. In addition, the WebExtension is also officially published by Mozilla in their Firefox browser add-ons website.

Part IV

Site-level Block Detection Algorithms

Chapter 7

Candidates selection algorithms

As introduced in Chapter 2, site-level block detection techniques, such as content extraction or template detection techniques, need to load and analyze several web pages in order to detect the information blocks. Given a web page w, this chapter and the following propose two steps for these algorithms:[1]

i. Detection of a set S of web pages from the same website which share the same template than w.

ii. Analysis of all web pages in $S \cup \{w\}$ to obtain the required information.

This chapter proposes a new technique for detecting a set of web pages from the same website that share the same template by analyzing the hyperlinks.

The technique analyzes all the links in a given web page and sorts them based on certain criteria defined to promote the web pages that probably share the same template. Once the hyperlinks are sorted, the first ones form a set of web pages if they are pairwise and mutually linked. This set of web pages probably implements the same template. In consequence, one of the main benefits of the technique is that it only needs to analyze a reduced set of web pages. This contrasts with other techniques that analyze large sets of web pages [120, 116, 113].

It is evident that all web pages in a website do not necessarily share the same template. Usually, a website implements several templates, especially large websites. As a result, the algorithm has two seemingly contradictory goals: (i) the selected web pages must be as similar as possible to the target one, therefore it is easy to infer the template, and (ii) the selected web pages must be as different as possible between them so that the set of web pages is as heterogeneous as possible and they implement different templates.

[1]These two steps are not necessarily sequential, indeed, they are often interlaced.

7.1 Related Work

Chapter 2 categorizes block detection techniques into two groups: site-level and page-level. In both cases, their aim is to obtain information from web pages, but using different information. On the one hand, page-level techniques only obtain information from the target web page. On the other hand, site-level techniques also obtain information from other web pages, usually on the same website.

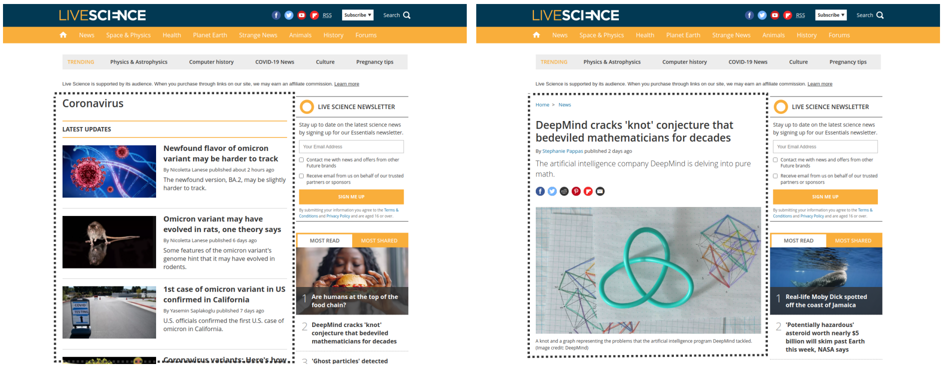
Site-level block detection techniques are often divided into two phases that are not necessarily independent. First, they build a set of web pages from which they (hopefully) extract the necessary information to achieve their objective. Finally, they extract the desired block by comparing the target web page with the web pages in the set. This technique automates the first step. It should be emphasized that very few techniques in the literature describe the web page candidates detection procedure. On many occasions, this process is done manually or authors do not explain how it is done.

In some cases, programmers prepare the input with sets of web pages selected by them. For instance, in the content extraction algorithm presented in [83], and the template extractor proposed in [54].

The template extraction technique described in [120] inputs a set of web pages to build the Site Style Tree structure. They do not use a specific technique to select the web pages, but they prepare the input by randomly sampling 500 web pages.

The template extraction technique in [114] is based on discovering optimal mappings between the DOM trees of different web pages from the same website. The technique picks the web pages randomly until a threshold is reached. The authors describe the threshold as a few dozen of web pages. In fact, their technique needs to load 25 web pages to reach an F1 measure of 0.95. On the other hand, the technique presented in this chapter does not choose the web pages randomly, given that the hyperlinks of the web pages are analyzed in order to select them. It manages to build a set of candidates that implement the template by exploring only a few web pages. In contrast, many techniques such as [114, 113], assume that all web pages on the same website always share the same template, an assumption that many websites do not accomplish.

Other techniques like [95] assume that those web pages located in the same directory share the same template. Their algorithm uses as web page candidates other pages stored in the same directory as the target web page. Part of our algorithm exploits this idea, but it is not restricted to



Live Science is supported by its audience. When you purchase through links on our site, we may earn an affiliate commission. Learn more

Coronavirus

LATEST UPDATES

Newfound flavor of omicron variant may be harder to track
By Nicoletta Lanese published about 2 hours ago
The newfound version, BA.2, may be slightly harder to track.

Omicron variant may have evolved in rats, one theory says
By Nicoletta Lanese published 6 days ago
Some features of the omicron variant's genome hint that it may have evolved in rodents.

1st case of omicron variant in US confirmed in California
By Yasemin Saplakoglu published 7 days ago
U.S. officials confirmed the first U.S. case of omicron in California.

Coronavirus variants: Here's how

LIVE SCIENCE NEWSLETTER
Stay up to date on the latest science news by signing up for our Essentials newsletter.
Contact me with news and offers from other Future brands
Receive email from us on behalf of our trusted partners or sponsors
By submitting your information you agree to the Terms & Conditions and Privacy Policy and are aged 16 or over.

1 Are humans at the top of the food chain?
2 DeepMind cracks 'knot' conjecture that bedeviled mathematicians for decades
3 'Ghost particles' detected

DeepMind cracks 'knot' conjecture that bedeviled mathematicians for decades
By Stephanie Pappas published 2 days ago
The artificial intelligence company DeepMind is delving into pure math.

A knot and a graph representing the problems that the artificial intelligence program DeepMind tackled. (Image credit: DeepMind)

1 Real-life Moby Dick spotted off the coast of Jamaica
2 'Potentially hazardous' asteroid worth nearly $5 billion will skim past Earth this week, NASA says

Figure 7.1: Web pages of LiveScience sharing a template

one directory, because it establishes an order of relevance which uses the tree of directories through a definition of distance between directories.

7.2 Identifying web pages that implement the same template

Templates usually provide enough information to infer the different web-page blocks since almost always they consist of a set of pagelets, such as the menu, advertisements, etc. Obtaining a set of web pages from the same website that share the template is useful to infer both, the template and the main content, by comparing those web pages between them [4]. Figure 7.1 shows two web pages from the LiveScience website. At the top of the web pages, below the advertisement area, we can observe the main menu containing links to all LiveScience principal topics. The left web page belongs to the “Coronavirus” section, while the right web page belongs to a story called “DeepMind cracks ‘knot’ conjecture that bedevilled mathematicians for decades”. Both share the same menu, the bar on the right, and the general structure. In both web pages, the pagelet in the dashed square defines the main content, i.e., the news. Furthermore, both web pages contain a common pagelet which includes links to some relevant topics, another one for subscribing to the newsletter, and another that compiles the most read and the most shared news. Additionally, there is a common footer at the bottom of all web pages.

The technique described in this chapter gets a web page (called *key page* in the following) as input and returns a set of web pages of the same website that implement the template (or part of it). A complete subdigraph is identified in the website topology in order to discover the web pages that implement the template.

7.2.1 Complete subdigraphs

A complete subdigraph (CS) in a website topology is a set of web pages that are pairwise mutually linked [9]. Thus, an n-complete subdigraph (n-CS) is a CS formed by n nodes.

It has been observed that, usually, the web pages linked by the items in a menu form a CS. Therefore it can be used to identify web pages that contain the menu. In addition, we can observe that these web pages are usually the roots of the various sections linked by the menu. The following idea is based on these observations:

Idea 7.2.1 *Those web pages pointed out by the links in the menu of the template probably share the template, because they are the main web pages of each section of the website.*

Idea 7.2.2 *Those web pages pointed out by the links of a menu and that contain the menu form a complete subdigraph in the website topology because they all are mutually linked.*

Example 7.2.3 illustrates why menus provide useful information about the interconnectivity of the different web pages in a given website.

Example 7.2.3 *Consider the LiveScience website. Figure 7.1 shows two of its web pages. All web pages on this website share the same template, and this template contains the main menu that obviously appears on all web pages, an advertisement area, and a "Most Popular" news area. The site map of the LiveScience website can be described with the topology shown in Figure 7.2.*

In Figure 7.2 nodes represent web pages and edges represent links between two web pages. For simplicity, we only draw some of the nodes and some of the edges. Note that solid edges are bidirectional while dotted and dashed edges are directed arcs. Web pages pointed by the main menu are represented by black nodes. In addition, black nodes are web pages pointed by the main menu links. Obviously, the main menu appears on all web pages, so all nodes are connected to all black nodes. Given that the set of black nodes forms a complete graph, therefore there is an edge connecting each pair of black nodes. Grey nodes also form a complete graph because they are web pages pointed by a submenu. On the other hand, white nodes

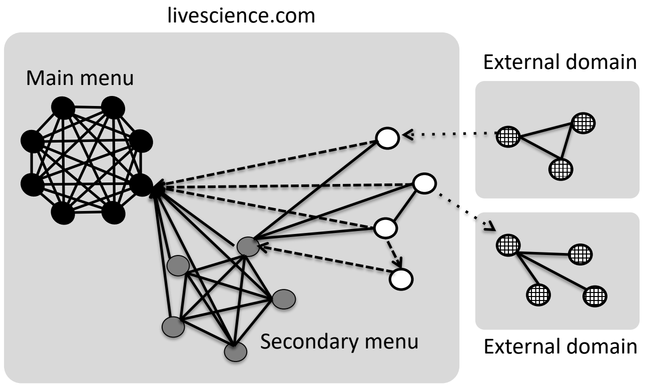


Figure 7.2: LiveScience Website topology

do not form a complete graph because they correspond to web pages inside one section of the submenu.

It is important to emphasize that not all web pages in a website share the same template, while some web pages only implement a subset of a template, others extend the template by adding new pagelets. Therefore, one of the main problems of site-level block detection techniques is deciding which web pages should be analyzed. It is very important to minimize the number of analyzed web pages in order to reduce computation resources. This technique lays out a new idea to identify which web pages must be analyzed: it analyzes and sorts the links in the key page that form a CS. Note that only the web pages pointed by the key page are analyzed. If a web page pointed by the key page does not have any mutual link (pointing to the key page), it is discarded.

It must be highlighted that not all the links that form a CS produce equally good CSs. If we consider the topology shown in Figure 7.2 and we suppose that the key page corresponds to one of the white nodes, then, it is better to form a CS using the grey nodes (the submenu) than with the black nodes (the main menu). Note that the submenu is a common substructure shared by all white and grey nodes. This fact justifies establishing an order to the links of the CS. In the example, the white nodes belong to one of the items in the submenu, so they are probably more related in semantic terms and, with a high probability, they share more syntax components.

7.2.2 Hyperlink analysis

The selection of the links in the key page that most likely produce the best CS can be done via a hyperlink analysis strategy. This leads to increased performance because the strategy avoids analyzing all links on the key page. The links that should be analyzed are identified by examining the structure of the website since the URLs of the links provide valuable information about the structure of the website.

Idea 7.2.4 *In a website, those web pages located in the same folder probably share the same template. The probability of two web pages sharing the same template can be approximated based on the distance between them in the tree of directories.*

Example 7.2.5 *Consider a key page P whose URL is:*
`www.springer.com/gp/computer-science/become-an-author`
Consider that P contains four links pointing to the following URLs:

- *URL 1 =* `www.springernature.com`
- *URL 2 =* `www.springer.com/gp/computer-science/stay-informed`
- *URL 3 =* `www.springer.com`
- *URL 4 =* `www.springer.com/gp/engineering/contact-us`

URL 1 *points to another domain. The pointed web page is located in a different domain. Thus, the template of the pointed web page and the template of the key page is with a very high probability entirely different.*

URL 2 *points to a web page which is located in the same directory as the key page. Consequently, both web pages very likely belong to the same section in the hierarchy of the website, hence they probably share the same template.*

URL 3 *points to the main web page of the website, which is located inside a directory that is two levels above the directory of the key page. Hence, the layout of the key page and the web page pointed out by URL 3 are arguably different, and probably they only share a fraction of their templates.*

URL 4 *points to a web page that is located inside a directory at the same level as the reference directory. With high probability, it points to another section of the website (e.g., to another subject in the main menu called* engineering*).*

There are also other (infinite) options. For instance, consider a web page located in a subdirectory inside the directory of the key page. This web page is most probably semantically related to the key page, and it probably extends its template adding complementary information.

Consequently, the analysis of the links in the key page can lead to an order of relevance (see Algorithm 8). The definition of partial order uses the definition of the length of a DOM path (4.2.3), the definition of hyperlink (4.1.5), and the definition of distance between hyperlinks (4.2.1).

It must be emphasized that the name of the resource pointed out by the URL is ignored by the proposed definition of hyperlink. Thus, it only considers the domains and directories (structure).

It is just enough to include the URLs such as $www.springer.com/$, $gp/computer-science/$ and $gp/computer-science/information-systems-and-applications/$.

In the following, function *head* is used to select the first word (i.e., directory) of a hyperlink:

$$head(directory1/directory2/directory3/) = directory1.$$

Definition 4.2.1 develops the notion of distance between two URLs. Note that the order of the parameters is important because the distance can be positive or negative. Therefore, the distance between two hyperlinks is defined from the first hyperlink to the second one.

Figure 7.3 represents a tree of directories containing web pages from a website. We can observe the distance from all web pages to a web page located in the gray directory.

Example 7.2.6 *Consider the following URLs from the Springer's website:*

(1) `gp/computer-science/`

(2) `gp/computer-science/information-systems-and-applications/`

(3) `gp/`

(4) `gp/engineering/`

(5) `www.springer.com/gp/`

The following URL distances are computed:

$hDistance(1,1) = 0$
$hDistance(1,2) = +1$
$hDistance(1,3) = -1$
$hDistance(1,4) = -1$
$hDistance(1,5) = -2$

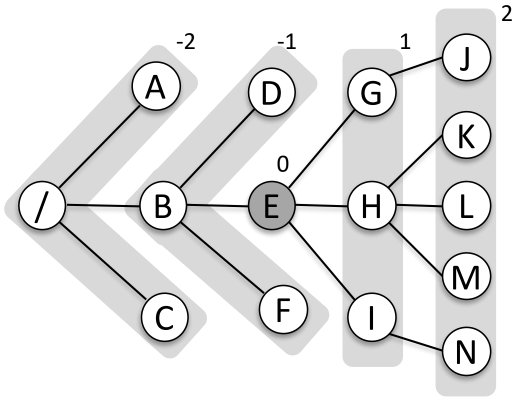


Figure 7.3: Hyperlink distance

Idea 7.2.7 *Those links located closer in the DOM tree are more likely to be semantically related, particularly if they belong to the same block. Thus, the distance between two links in the DOM tree is an indicator of the semantical relation among the linked web pages.*

We can observe that a distance of 0 between two given links, $h1$ and $h2$, means that they point to the same directory of the website. Even so, a positive distance between two links, $h1$ and $h2$, means that $h2$ points to a subdirectory of the directory pointed by $h1$. Likewise, a negative distance between two links, $h1$ and $h2$, means that $h2$ points to a directory located outside of the directory pointed by $h1$. This directory pointed by $h2$ can either be an ancestor of the directory pointed by $h1$ or not.

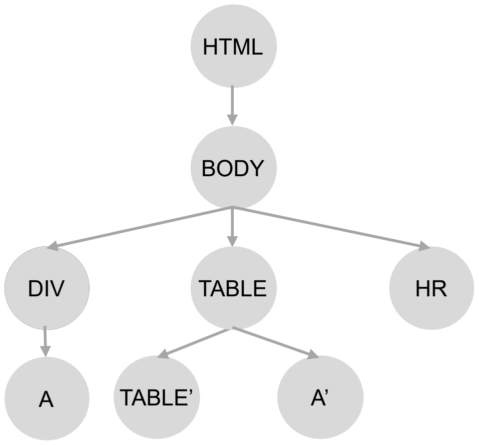
The links in the key page are analyzed and their distances in reference to the directory where the URL of the key page points are computed. The best links are those with a distance of 0. Then, those with a positive distance. Finally, the links with a negative distance. Note that the hyperlink distance defines a partial order (see Definition 4.2.1). Therefore, the case of a draw should be taken into account, since it is very common to find an equal distance from two different links to a third link.

In the case of a draw, based on the idea 7.2.7, another algorithm is used to determine which link is better. That algorithm analyzes the position of the link nodes in the DOM tree. Usually, pagelets agglutinate semantically related information, therefore, the information is distributed in pagelets, so two links belonging to different pagelets should point to web pages whose content differs semantically. This is particularly important given that site-level block detection techniques analyze the differences between the selected web pages to detect the desired information (template, main content, etc.).

Example 7.2.8 *Consider the menu of a University web page. Probably, we can find a set of links pointing to the different Bachelor's degrees provided by the University. Often, those web pages share the same template, which is filled with information about the different Bachelor's degrees. Therefore, it is possible that block detection algorithms confuse some of the information because it can appear repeated on several web pages. A template detection algorithm could identify some of the information (main content) repeated in many web pages as template information. Hence, site-level block detection algorithms should prevent analyzing these web pages because they do not contain enough information to compare them.*

Consequently, in case of a draw, it is preferable to first select the links further away from the already selected links in the DOM tree. Therefore, for the links with the same hyperlink distance, the partial order gives preference to the links located further away from the already selected links, that is, the links with (probably) different semantic information from the ones that are already selected. In conclusion, the order of relevance establishes an order to the links of the key page choosing first the ones that implement the same template (using the hyperlink distance) but being as distinct as possible (based on their position in the DOM tree). As a consequence, in case of a draw, we prefer those links that are as separated as possible from the other already selected links in the DOM tree. In this way, we give preference to links with (probably) different semantic information. In summary, observe that we obtain web pages that share the same template (using the hyperlink distance) but are as different as possible (using their position in the DOM tree of the key page).

Figure 7.2.2 shows part of a web page's DOM tree. The DOM tree contains two link nodes, and we can observe some examples of hyperlink distance and DOM distance.



$link(A)$ = www.upv.es
$link(A')$ = www.upv.es/organizacion/la-institucion/index-en.html
$path(A)$ = HTML BODY DIV A
$path(A')$ = HTML BODY TABLE A'
$dDistance(A, A')$ = 4 // DIV BODY TABLE A'
$dDistance(A', A)$ = 4 // TABLE BODY DIV A
$hDistance(link(A), link(A'))$ = 2
$hDistance(link(A'), link(A))$ = -2

Figure 7.4: Hyperlink and DOM distance examples

The position in the DOM tree of two links is measured by comparing the length of both paths. Note that $path(n)$ represents the DOM path of a node n, namely, the path from the root of the DOM tree to that node.

Definition 4.2.4 determines the distance between two nodes in the DOM tree. It should be noted that the DOM distance of two links (or DOM nodes) is zero if and only if they are exactly the same link (or DOM node). Otherwise, two different links have imperatively a positive DOM distance, even if they have the same URL, and consequently the same hyperlink distance. In this case, the DOM distance is computed as the length of the DOM path from the first node to the root node plus the length of the DOM path from the root node to the second node. Note that the length of a DOM path is computed using Definition 4.2.3.

Once hyperlink distance and DOM distance algorithms have been determined, an order for the links in a web page can be defined. It promotes the links that should be explored by site-level block detection algorithms. This order combines two orders, both proposed in [5]: link relevance and DOM relevance. Link relevance (see Definition 4.2.5) order ($\leq^h_{link}$) uses the link distance algorithm (see Definition 4.2.1), and establishes an order for a set of hyperlinks based on the hyperlink distance definition. On the other hand, DOM relevance (see Definition 4.2.6) order ($\leq^N_{DOM}$) uses the DOM distance algorithm (see Definition 4.2.4), and establishes an order based on the DOM distance between the DOM nodes.

7.2.3 Finding web page candidates in a website

The application of both orders, link relevance and DOM relevance, is used to choose the links that must be explored first in order to find a CS from the web pages pointed by them. This process is done by Algorithm 8.

Algorithm 8 Sort links

```
Input: A set of hyperlink nodes $links$ and a reference hyperlink $h$.
Output: A sorted list of $links$ with respect to the preorders $\leq^h_{link}$ and $\leq^N_{DOM}$.
begin
  $sortedLinks = []$;
  while ($links \neq \emptyset$)
    $links' = \{l \in links \mid \nexists l' \in links \wedge l' <^h_{link} l\}$;
    $links = links \backslash links'$;
    $sortedLinks' = []$;
    while ($links' \neq \emptyset$)
      $link = l \in links' \mid \nexists l' \in links' \wedge l' <^{sortedLinks'}_{DOM} l$;
      $links' = links' \backslash \{link\}$;
      $sortedLinks' = sortedLinks' \mathbin{++} [link]$;
    $sortedLinks = sortedLinks \mathbin{++} sortedLinks'$;
  return $sortedLinks$;
end
```

Algorithm 8 combines link relevance and DOM relevance algorithms to sort the links in the key page. First, it sorts the links in the order obtained from the link relevance algorithm. Then, it uses the DOM relevance to sort each set of links with the same link relevance value. The resulting order is the concatenation of each sorted set.

Table 7.1 contains the links obtained from the news web page[2] from the Caltech University website. Note that their order is random because they have been obtained with the *links* read-only property of de *Document* interface[3]. Table 7.2 shows the result of sorting the same links with Algorithm 8. Column `Link d.` shows the value obtained for hyperlink distance, while Column `DOM d.` refers to the value obtained for DOM distance. Note that they are sorted using hyperlink distance, obtaining 3 sets of links. Then, the links in each of the sets are sorted using the DOM distance. As we can observe in the table, the combination of both sorting algorithms produces the final order of the links. For instance, all the links with a hyperlink distance equal to 0 are then ordered based on their DOM distance, in order from highest to lowest. This is then repeated for the links with positive hyperlink distance, and finally for those with negative hyperlink distance. As we can observe in the table, the first URL of each hyperlink distance group has no value for its DOM distance. This is produced because, as stated before, the DOM distance is computed as the distance from a hyperlink to the other already sorted links of the same hyperlink distance group. When the number of sorted links of each hyperlink distance group is zero, it is not possible to compute the distance between each link and the already sorted links. Therefore, the first link of each hyperlink distance group is selected randomly. Then, when a hyperlink distance group has one or more sorted links, the following links are sorted using the DOM distance.

Algorithm 9 explores the links in the key page following the order obtained by Algorithm 8 to find a CS. Function *loadWebPage*(*link*) is a trivial function that, given the input link, it loads and returns the web page it points to, while *getLinks*(*webpage*) returns a set of non-repeated links[4] in the input web page (without considering self-links). The algorithm iteratively explores the links contained in the set *sortedLinks*, which is the result of sorting the links in the key page using link relevance and DOM relevance. When a n-CS is found the algorithm stops. It is important to remark that only the web pages processed until the n-CS is completed are loaded. The following mathematical expression is the key of the algorithm:

$$CS = \{ls \in \mathcal{P}(processedLinks) \mid link \in ls \land \forall l, l' \in ls \;.\; (l \to l'), (l' \to l) \in connections\}$$

where $\mathcal{P}(X)$ returns all possible non-empty partitions of set X.

[2]https://www.caltech.edu/about/news

[3]https://developer.mozilla.org/en-US/docs/Web/API/Document/links

[4]In our implementation, those links pointing to other domains or subdomains were removed because they have a very low probability of containing the same template.

Hyperlink
https://www.caltech.org/about/visit/plan-your-visit
https://www.caltech.org/quick-links-faculty
https://www.caltech.org/quick-links-students
https://www.caltech.org/quick-links-staff
https://www.caltech.org/quick-links-alumni
https://www.caltech.org/campus-life-events/caltech-today
https://www.caltech.org/
https://www.caltech.org/about
https://www.caltech.org/about/at-a-glance
https://www.caltech.org/about/legacy/history-milestones
https://www.caltech.org/about/legacy/historic-awards-honors
https://www.caltech.org/map/history
https://www.caltech.org/about/visit/directions
https://www.caltech.org/about/visit/campus-maps
https://www.caltech.org/about/visit/tours
https://www.caltech.org/about/offices-departments
https://www.caltech.org/research
https://www.caltech.org/research/jpl
https://www.caltech.org/research/centers-institutes
https://www.caltech.org/research/research-facilities
https://www.caltech.org/research/faculty-listing
https://www.caltech.org/academics
https://www.caltech.org/academics/resources/academic-calendar
https://www.caltech.org/admissions-aid
https://www.caltech.org/campus-life-events
https://www.caltech.org/campus-life-events/master-calendar
https://www.caltech.org/campus-life-events/emergency-information
https://www.caltech.org/media-contacts
https://www.caltech.org/rssfeeds
https://www.caltech.org/about/news/caltech-mourns-passing-manuel-manny-soriaga
https://www.caltech.org/about/news/what-it-be-caltech-seismologist-during-big-quake
https://www.caltech.org/about/news/caltechs-apollo-connection
https://www.caltech.org/about/news/performing-chemistry-floating-droplets
https://www.caltech.org/about/news/bronner-named-director-beckman-institute
https://www.caltech.org/about/news/hima-vatti-tapped-to-lead-caltechs-equity-and-title-ix-office
https://www.caltech.org/about/news/three-caltech-professors-receive-presidential-early-career-awards
https://www.caltech.org/about/news/ztf-spots-asteroid-shortest-year
https://www.caltech.org/about/news/edge-philosophy-and-physics
https://www.caltech.org/about/news/bioengineers-guide-design
https://www.caltech.org/about/news/seeing-farther-and-deeper-interview-katie-bouman
https://www.caltech.org/about/caltech-media
https://www.caltech.org/contact
https://www.caltech.org/claimed-copyright-infringement
https://www.caltech.org/privacy-notice

Table 7.1: Links obtained from a Caltech's web page

Algorithm 9 uses this instruction to compute the set of all CS that can be built using the current *link*. To ensure that we make progress, that is, we do not repeat the same search of the previous iteration, the current link must belong to the CS ($link \in ls$). Intuitively, if we find a n-CS without the current *link*, then it implies that we have already found a n-CS in the previous iteration. Therefore, we can avoid the sets of $\mathcal{P}(processedLinks)$ that do not contain *link* because these sets were discarded in previous iterations. Furthermore, as the CS is built incrementally, the statement

$$\textbf{if } |maximalCS| = n \textbf{ then return } maximalCS$$

ensures that as long as an n-CS is built, it is returned.

Order	Hyperlink	Link d.	DOM d.
1	https://www.caltech.org/about/at-a-glance	0	-
2	https://www.caltech.org/about/caltech-media	0	13
3	https://www.caltech.org/about/offices-departments	0	4
4	https://www.caltech.org/about/news/caltech-mourns-passing-manuel-manny-soriaga	1	-
5	https://www.caltech.org/about/legacy/history-milestones	1	17
6	https://www.caltech.org/about/visit/plan-your-visit	1	10
7	https://www.caltech.org/about/news/what-it-be-caltech-seismologist-during-big-quake	1	6
8	https://www.caltech.org/about/news/caltechs-apollo-connection	1	6
9	https://www.caltech.org/about/news/performing-chemistry-floating-droplets	1	6
10	https://www.caltech.org/about/news/bronner-named-director-beckman-institute	1	6
11	https://www.caltech.org/about/news/hima-vatti-tapped-to-lead-caltechs-equity-and-title-ix-office	1	6
12	https://www.caltech.org/about/news/three-caltech-professors-receive-presidential-early-career-awards	1	6
13	https://www.caltech.org/about/news/ztf-spots-asteroid-shortest-year	1	6
14	https://www.caltech.org/about/news/edge-philosophy-and-physics	1	6
15	https://www.caltech.org/about/news/bioengineers-guide-design	1	6
16	https://www.caltech.org/about/news/seeing-farther-and-deeper-interview-katie-bouman	1	6
17	https://www.caltech.org/about/visit/directions	1	6
18	https://www.caltech.org/about/legacy/historic-awards-honors	1	2
19	https://www.caltech.org/about/visit/campus-maps	1	2
20	https://www.caltech.org/about/visit/tours	1	2
21	https://www.caltech.org/quick-links-faculty	-1	-
22	https://www.caltech.org/media-contacts	-1	12
23	https://www.caltech.org/map/history	-1	11
24	https://www.caltech.org/contact	-1	10
25	https://www.caltech.org/academics/resources/academic-calendar	-1	10
26	https://www.caltech.org/research/jpl	-1	9
27	https://www.caltech.org/campus-life-events/master-calendar	-1	8
28	https://www.caltech.org/admissions-aid	-1	6
29	https://www.caltech.org/	-1	5
30	https://www.caltech.org/about	-1	4
31	https://www.caltech.org/research	-1	4
32	https://www.caltech.org/academics	-1	4
33	https://www.caltech.org/campus-life-events	-1	4
34	https://www.caltech.org/research/centers-institutes	-1	4
35	https://www.caltech.org/research/research-facilities	-1	4
36	https://www.caltech.org/research/faculty-listing	-1	4
37	https://www.caltech.org/campus-life-events/caltech-today	-1	4
38	https://www.caltech.org/campus-life-events/emergency-information	-1	4
39	https://www.caltech.org/quick-links-students	-1	2
40	https://www.caltech.org/quick-links-staff	-1	2
41	https://www.caltech.org/quick-links-alumni	-1	2
42	https://www.caltech.org/rssfeeds	-1	2
43	https://www.caltech.org/claimed-copyright-infringement	-1	2
44	https://www.caltech.org/privacy-notice	-1	2

Table 7.2: Order obtained from a Caltech's web page

Figure 7.5 shows the news web page from the Caltech University website and the result of computing a complete subdigraph of size 4 (4-CS). The links that form the 4-CS are:

- https://www.caltech.org/about/at-a-glance
- https://www.caltech.org/about/offices-departments
- https://www.caltech.org/about/legacy/history-milestones
- https://www.caltech.org/about/visit/plan-your-visit

Note that the links are located at the top positions in Table 7.2, concretely they are located at the first 6 positions. This means that it has been possible to build a 4-CS with the first 6 links in Table 7.2. There are 2 links in those first 6 positions that do not form a complete subdigraph with the others. Despite the fact that from the web pages pointed by those

Algorithm 9 Extract an n-CS from a website

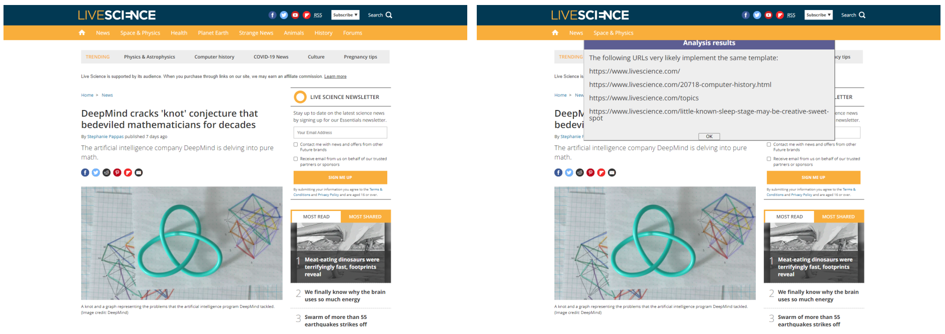
```
Input: An $initialLink$ that points to a web page and the expected size $n$ of the CS.
Output: A set of links to web pages that together form an n-CS.
        If an n-CS cannot be formed, then they form the biggest m-CS with m < n.

begin
  $keyPage = loadWebPage(initialLink)$;
  $reachableLinks = getLinks(keyPage)$;
  $processedLinks = \emptyset$;
  $connections = \emptyset$;
  $bestCS = \emptyset$;
  $sortedLinks = sortLinks(reachableLinks, initialLink)$;
  foreach ($link$ in $sortedLinks$)
      $webPage = loadWebPage(link)$;
      $existingLinks = getLinks(webPage) \cap reachableLinks$;
      $processedLinks = processedLinks \cup \{link\}$;
      $connections = connections \cup \{(link \to existingLink) \mid existingLink \in existingLinks\}$;
      $CS = \{ls \in \mathcal{P}(processedLinks) \mid link \in ls \land \forall l, l' \in ls . (l \to l'), (l' \to l) \in connections\}$;
      $maximalCS = cs \in CS$ such that $\forall cs' \in CS . |cs| \geq |cs'|$;
      if $|maximalCS| = n$ then return $maximalCS$;
      if $|maximalCS| > |bestCS|$ then $bestCS = maximalCS$;
  return $bestCS$;
end
```

2 links all the web pages in the 4-CS can be reached, the web pages in the 4-CS do not contain any link to them, so they can not form a CS.

7.3 Implementation

As with the rest of the techniques presented in this thesis, this technique for candidates selection (including all its algorithms) has been implemented as a WebExtension. The extension is implemented in JavaScript and it contains about 2500 LOC.

Section 12.2 introduces a workbench for *Template Detection*. It provides four common modules that can be used by any template detection



Analysis results

The following URLs very likely implement the same template:

https://www.livescience.com/

https://www.livescience.com/20718-computer-history.html

https://www.livescience.com/topics

https://www.livescience.com/little-known-sleep-stage-may-be-creative-sweet-spot

OK

LIVESCIENCE
DeepMind cracks 'knot' conjecture that bedeviled mathematicians for decades
The artificial intelligence company DeepMind is delving into pure math.
LIVE SCIENCE NEWSLETTER
SIGN ME UP
MOST READ
MOST SHARED

Figure 7.5: Key page (left) and a CS set of web pages (right) automatically identified with the tool

algorithm if needed. Two of those modules are the *hyperlink analysis* and *complete subdigraph extraction* algorithms described in this chapter. As we can observe in Figure 12.2, the algorithms presented in this chapter form the *detection of candidates* phase of the workbench. The input of this phase is the key page and the web pages pointed by its hyperlinks, while the output is the *complete subdigraph*.

7.3.1 Empirical evaluation

This chapter presents the candidates selection technique in an abstract way. Some ideas and definitions such as DOM distance, hyperlink distance and the influence of the main menu links in sharing the template can be demonstrated empirically. In addition, other features, such as the size of the CS, are parameters of some algorithms. For instance, the computation of a CS (Algorithm 9) can be done of any specified size if there exist enough web pages mutually linked. This section empirically validates some premises used to develop the algorithms. Moreover, the optimal size of the CS is determined based on the empirical evaluation.

It should be highlighted that some restrictions have been applied to the domain boundaries of the analyzed websites. It is usual to find web pages of different domains, from the same or different organizations, mutually linked. In this case, the templates are often different. Note that the algorithm cannot find the same template across different domains. Hence, external domains have been omitted when computing the CS.

To validate the proposed statements and determine the parameters the training subset of the TeCo benchmark suite (formed by 105 websites) has been used. As Chapter 13 describes and Table 5.3 shows, this training subset consists of 21 randomly selected benchmarks from 5 different categories.

Main menu links influence

Idea 7.2.1 was validated using the training dataset to compute the similarity between the key page's template and the template of each web page pointed and not pointed by the main menu. The total sample was formed from 1838 web pages pointed from the menu and 2944 web pages not pointed from the menu of their respective key page. It should be highlighted that we selected a maximum of 40 web pages of each type (pointed or not pointed by the menu) per benchmark. The achieved results are shown in Figure 7.6.

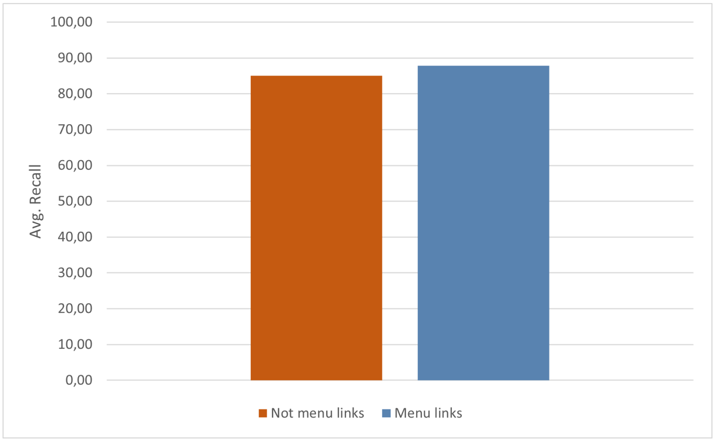


Figure 7.6: Similarity between the key page and the web pages pointed and not pointed by the menu

The chart illustrates that the web pages pointed by the main menu more likely share the template with the key page. The bars represent the recall, which is the average number of DOM nodes in the key page's template that appear in the explored web pages. The average recall obtained by the web pages pointed by the menu was 87.88%, while the average recall obtained by the web pages not pointed by the menu was 85.09%.

Table 7.3 shows the results grouped by benchmark type. In the table, column `Benchmark type` shows the corresponding benchmark type. Column `Recall` is the average recall computed for all the pointed web pages of each benchmark type. Finally, column `Avg. web pages` shows the average number of web pages analyzed for each benchmark.

	Menu		Not menu	
Benchmark type	Recall	Avg. web pages	Recall	Avg. web pages
Institutions / Associations	93.67 %	22.05	90.19 %	13.77
Media / Communication	82.44 %	22.59	87.51 %	26.67
Forums /Social	79.78 %	22.00	80.12 %	108.90
Personal websites / Blogs	90.98 %	11.82	81.01 %	10.00
Companies / Shops	91.83 %	23.05	88.80 %	19.77

Table 7.3: Menu links influence by benchmark type

It can be observed that for the *Media / Communication* and *Forums / Social* benchmark types, the web pages not pointed by the main menu more likely share the template with the key page. On the other side, for the rest of the benchmark types, the web pages pointed by the main menu more likely share their template with the key page.

Web page location influence

Idea 7.2.4 was also validated using the TeCo training dataset. A maximum of 40 hyperlinks per benchmark were selected, and then, the mapping between the key page of each website and the web pages pointed by its links was computed. In Table 7.4 and Figure 7.7 we can observe the relation between the hyperlink distance and the recall. In the table, column `URL distance` ranges between -7 and 5, however, URL distances lower than -4 are not representative because very few benchmarks contain web pages with such URL distances. For each URL distance value, column `Recall` represents the average recall computed for all the web pages with such value. Finally, column `#Benchmarks` shows the number of benchmarks that contains analyzed hyperlinks of each URL distance.

We can observe in the chart (Figure 7.7) that the recall is higher for the positive values than for the negative ones, except for the URL distance value of -3. This fact validates the Idea 7.2.4 which asserts that web pages located in the same folder probably share the same template. The case of the URL distance value of -3 is not as representative as the URL distances that range between -2 and 3, because it is computed only with web pages from 6 benchmarks, while the values that range between -2 and 3 are based on at least 2.5 times more web pages.

DOM distance influence

The TeCo training dataset was also used to validate Idea 7.2.7. We selected a maximum of 40 hyperlink DOM nodes from each key page in

URL distance	Recall	#Benchmarks
-7	0.64%	1
-6	78.00%	1
-5	45.40%	2
-4	74.86%	4
-3	96.17%	6
-2	68.45%	21
-1	82.07%	43
0	89.17%	65
1	82.15%	31
2	86.36%	23
3	83.81%	16
4	83.99%	4
5	86.38%	5

Table 7.4: Relationship between hyperlink distance and recall

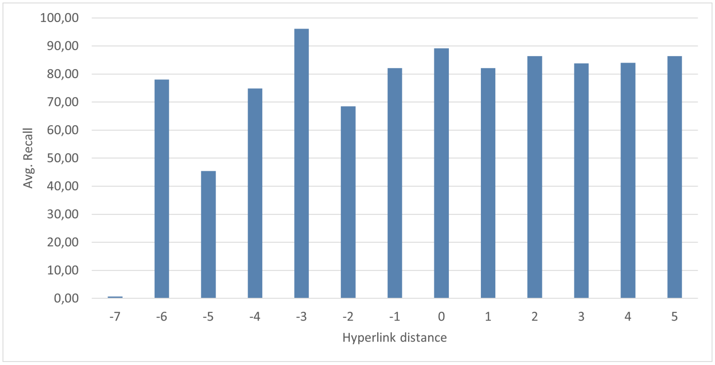


Figure 7.7: Graphical representation of Table 7.4

the dataset, and for each benchmark, we measured the `Recall` obtained by mapping its key page with each web page pointed by each hyperlink.

Table 7.5 and Figure 7.8 present the results of the experiment. In the table, column `Distance` shows the DOM distance. Column `Recall` is the average recall computed for all the pointed web pages of each benchmark type. Finally, column `#Benchmarks` shows the total number of benchmarks where we found hyperlink DOM nodes with that DOM distance. Note that, as commented above, a maximum of 40 hyperlink DOM nodes per key page were selected.

The chart on Figure 7.8 shows that, despite there are no relevant differences between the obtained recall values, higher DOM distance values have slightly higher recall values than lower DOM distance values. Therefore, the similarities between the template implemented by the web pages pointed by two hyperlinks are accentuated by the DOM distance between them.

Distance	Recall	#Benchmarks
2	63.87%	2
4	89.09%	11
5	96.07%	6
6	96.93%	9
7	97.57%	6
8	99.55%	8
9	81.46%	7
10	97.10%	11
11	93.85%	7
12	88.71%	10
13	83.80%	6
14	72.76%	8
15	89.89%	11
16	84.72%	7
17	84.23%	5
18	95.98%	2
19	98.00%	1
20	82.37%	2
21	83.08%	4
22	98.00%	1
23	98.00%	1
27	100.00%	1
29	98.00%	1
34	98.00%	1

Table 7.5: Relationship between DOM distance and template distance

Comparison with other candidates selection methods

As stated in Section 10.1, there are several approaches to build the set of web pages from which the site-level techniques extract the necessary information to achieve their objective. For instance, some programmers input the web pages manually ([83, 54]), other techniques use random web pages as input ([114, 120]), etc. In addition, we have demonstrated that the web pages linked from the main menu of the web page more likely implement the template (Idea 7.2.1). Therefore, we implemented several different methods for selecting the set of web pages in order to compare with them the hyperlink analysis technique described in this chapter. The implemented methods are:

- Selecting the candidate web pages randomly.
- Selecting as candidates only web pages that belong to the main menu.

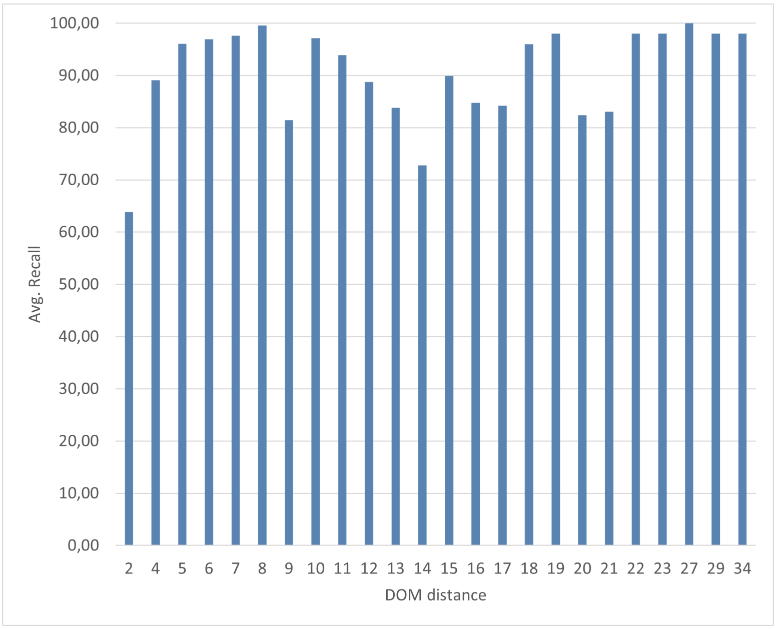


Figure 7.8: Graphical representation of Table 7.5

- Selecting as candidates web pages that do not belong to the main menu.
- Selecting the candidates using our hyperlink analysis algorithm.

We evaluated the different candidate selection methods with our template detection algorithm (see Chapter 9) using the 105 benchmarks of the TeCo benchmark suite (see Chapter 13). The obtained results are presented in Table 7.6. We can observe that our hyperlink analysis method obtains the best recall and the best F1 value, while selecting random links obtain the best precision. From the table we can also infer that a candidates selection technique has to combine both, links from the main menu and links that do not belong to it, because selecting only links that belong to the main menu or only links that do not belong to it are the methods that obtain the worse results.

Determining the size of the complete subdigraph

It is important to determine the optimal size of the CS computed by Algorithm 9. On the one hand, we could think that the bigger the CS

Method	Recall	Precision	F1
Random links	87.64 %	90.74 %	85.31 %
Links from menu	87.04 %	87.49 %	83.69 %
Links not from menu	86.10 %	88.11 %	82.56 %
Hyperlink analysis	92.16 %	90.21 %	88.66 %

Table 7.6: Comparison of different candidate selection methods

is, the better because it provides more information since it contains more web pages. On the other hand, the cost of computing the maximal CS is exponential in the worst case[5], so it is important to determine an optimal CS size. In addition, experiments reveal that an increase in the CS size does not imply obtaining better precision or recall. Therefore, it is more convenient to find a CS of a big enough size to ensure good precision, but small enough to ensure good performance.

It should be highlighted that the optimum CS size also depends on the block detection algorithm that uses it, e.g. a template detection algorithm can obtain better results with a CS of size 3, while another can obtain better results with a CS of size 4.

We performed several experiments to determine the optimum CS size for the template detection algorithm described in Chapter 9, which implements this technique to select the candidate web pages. In order to determine the best value, the experiments were repeated with different CS sizes (from 1 to 8). Table 7.7 presents the obtained results.

Size	Recall	Precision	F1	Loads	Runtime
1	83.69 %	82.59 %	76.21 %	2.00	1303 ms.
2	91.92 %	83.69 %	84.45 %	8.86	2305 ms.
3	91.14 %	94.37 %	91.33 %	12.63	3378 ms.
4	91.67 %	93.51 %	91.15 %	18.11	4555 ms.
5	91.23 %	94.13 %	91.25 %	26.74	6467 ms.
6	91.39 %	93.92 %	91.22 %	28.66	6998 ms.
7	90.57 %	94.52 %	91.06 %	30.11	7596 ms.
8	91.99 %	93.06 %	91.37 %	32.71	7851 ms.

Table 7.7: Determining the size of the complete subdigraph

To perform the experiments, we selected 35 benchmarks (7 from each category) from the training subset of the TeCo benchmark suite (see Chap-

[5]Even though the worst case complexity is of exponential order; in practice, most web pages contain menus that form CSs and that are easy to discover. For this reason, in Table 7.7 we can observe a linear increment in the performance.

ter 13). In Table 7.7, each row is the average of 35 template extractions from the selected training dataset with a different value for n in the n-CS computed by Algorithm 9.

The meaning of each column is:

`Size:` indicates the size of the CS computed by Algorithm 9 in the websites of the training dataset. It is possible that a CS of the searched size does not exist. In this case, as described in Algorithm 9, the algorithm computes the largest CS with a size under the specified size.

`Recall:` shows the result of dividing the number of correctly retrieved nodes by the number of nodes in the gold standard.

`Precision:` shows the result of dividing the number of correctly retrieved nodes by the number of retrieved nodes.

`F1:` shows the F1 metric.

`Loads:` represents how many web pages were loaded in average to build the n-CS. This value also includes the load of the key page.

`Runtime:` shows the runtime of the template extraction algorithm with that complete subdigraph.

We can observe that the F1 grows until it gets stabilized at a value of 91% approximately for a CS of size 3. The F1 values for a CS of size greater than 3 are very similar, so the increment of the size of the CS does not necessarily increase the F1 value, but contrarily, it increases the number of web pages loaded to build the CS and thus, the computation time, as we can observe in the table. Consequently, the optimal size for a complete subdigraph is 3 because it obtains almost the best F1 value while its computation time is efficient. Note that the algorithm needs to load potentially fewer web pages than in a size bigger than 3. Note that for a CS size of 8 the F1 is 0.04 % higher, but the runtime is also more than 2 times higher so a CS of size 3 is more convenient.

7.4 Conclusions

Site-level block detection algorithms are widely used. This chapter presents a new technique for detecting the candidate web pages that should be analyzed to extract the desired information. A set of web pages from a website sharing the same template is a very valuable source of information not only for template extraction techniques but also for content extraction or another kind of block detection techniques. Given a web page, it is important to select a set of web pages that very likely implement the same template. The described technique proposes a new method to build a set of candidate web pages that provides the relevant information needed by site-level block detection techniques. In addition, for performance reasons, the number of loaded web pages to build the set of candidate web pages must be reduced as much as possible. The technique was evaluated with the TeCo training set of benchmarks (see Chapter 13). The empirical evaluation concluded that building a complete subdigraph of size 3 is optimal for the template detection algorithm in Chapter 9 since a greater complete subdigraph does not obtain significantly better metric values and has a higher runtime.

For future work, a strategy to further reduce the number of web pages loaded with the technique could be investigated. The combination of this technique with the menu detection technique presented in Chapter 5 is an interesting research opportunity. Despite we have demonstrated that a candidates selection strategy based only on the links from the main menu is not the best option, a combination of both techniques could achieve good results with a reduced computational cost since the presented menu detection technique is page-level while the obtained set of web pages is a complete subdigraph.

7.5 Contributions

The candidate selection algorithms presented in this chapter provide several contributions that can be included in many systems, especially in site-level block detection techniques.

This chapter describes the hyperlink distance and DOM distance algorithms, which have been demonstrated to be accurate to establish an order of relevance of the links of a web page (the key page). Namely, the order of relevance sorts the links of a web page based on the probability of the pointed web pages implementing the same template of the key page. This has been evaluated with an empirical evaluation of both algorithms, hyperlink distance and DOM distance.

The implementation of the proposed technique as an independent module is also an important contribution. The module is part of a block detection architecture, so it can be used by many site-level block detection techniques.

Chapter 8

Equal Top-Down Mapping

As described in Chapter 7, in site-level block detection techniques, the information identified by canditates selection algorithms needs to be analyzed to extract the desired block, normally the template or the main content. This information consists of a set of web pages from a website that implement the same template.

The obtained set of web pages is processed in order to identify the desired block (normally the template implemented by them). Usually, the analysis of the set of web pages is done by comparing their DOM nodes to identify those which are repeated in several web pages. It should be highlighted that both techniques (the algorithm described in Chapter 7 and this one) are not always sequential, indeed, they are often interlaced.

This chapter defines a mapping algorithm that compares the DOM trees of the web pages identified by a candidates selection algorithm to identify the template.

8.1 Related Work

Site-level block detection techniques in literature implement several ways to infer the desired information block from the set of web pages obtained by the candidates' selection algorithm. However, not all of them are based on comparing the DOM nodes of the web pages in the set. For instance, some of them analyze the DOM tree with heuristics [19], while others analyze the DOM trees of a collection of web pages in the website to detect common subtrees [120, 114].

Authors in [120] build a data structure called Site Style Tree (SST) to analyze a set of DOM trees obtained from several web pages from the same website. An SST collects data from all the DOM nodes of the analyzed DOM trees. The SST is similar to a DOM tree; but, in contrast, it contains all the DOM nodes of the analyzed web pages. A node in the SST includes counters to identify repeated nodes in the DOM trees. In a nutshell, it is a kind of summary of the set of DOM trees. As the SST contains information about the most repeated nodes in a website, and the most repeated nodes in all the analyzed web pages most likely belong to the template, the most repeated nodes in the SST form the noisy information to be removed.

The described technique is similar to [114]. They perform mappings between several DOM trees corresponding to a set of web pages from a website. The main purpose of their mappings is to identify repeated nodes in several web pages. They use an algorithm called RTDM-TD that computes a mapping called *restricted top-down mapping* [94] between several web pages from the same website. There exist some differences between their technique and the technique presented in this chapter. The main difference is that their mapping does not force all nodes that form pairs in the mapping to be equal, so it is less restrictive.

The evolution of [114] is a mapping called RBM-TD proposed in [113] by the same authors. This mapping is similar to RTDM-TD, but the main difference is that it is a bottom-up mapping and RTDM-TD is top-down. One of the new features of this mapping is that it considers that the subtrees which are repeated in all web pages have to be exactly in the same position, that is, the DOM path from the root node to them must be the same for all web pages.

8.2 Comparing DOM nodes

Chapter 7 describes a technique that, given a web page called key page, analyzes a website and builds a set of web pages that share the whole template or part of it. This set of web pages forms a complete subdigraph.

Site-level block detection techniques obtain valuable information by comparing the DOM nodes of the set of selected web pages. For instance, a DOM node from the key page that appears in all web pages of the set probably belongs to the template of the website. In contrast, a node from the key page that only appears on it has a high probability of being a main content node.

8.2.1 Template extraction from a complete subdigraph

The DOM nodes in the DOM trees that form the CS must be compared in order to identify which of them appear in several DOM trees. This comparison is based on the notion of mapping. A mapping states a correspondence between the nodes of two DOM trees.

The following mapping, named *equal top-down mapping* (ETDM) (see Figure 8.1), is defined to perform the comparison of the DOM nodes in the DOM trees.

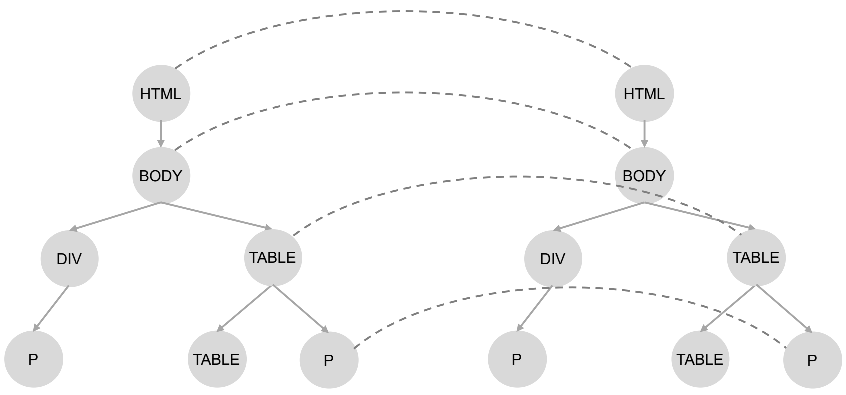


Figure 8.1: Equal top-down mapping between DOM trees

Definition 4.2.7 corresponds to the ETDM. It should be noted that the definition is parametric with respect to the equality relation $\triangleq$ because the relation is open to cover any possible implementation. Thus, this relation is not a simple standard equality (=). This technique introduces a complex notion of node equality that performs the comparison of two DOM nodes by considering the following properties:

- HTML *tagName*: The algorithm checks if both DOM nodes contain the same HTML *tagName*. If they share the same HTML *tagName*, the rest of the properties can be evaluated. Otherwise, both nodes can not be mapped.

- Node class names: The classes (HTML attribute *class*) of both DOM nodes are analyzed to compute the percentage they both share. For instance, if the node n_1 shares half of its classes with n_2, the value of this property is 0.5.

- Node position: The position of both nodes, n_1 and n_2, in the DOM tree is compared. The value of this property is 1 if both nodes are located at the same position and it decreases to 0 the further they are in the DOM tree.

Example 8.2.1 *Figure 8.2 shows two DOM nodes, A and B, with the same amount of children. The position computation algorithm for* *their child nodes will obtain the values shown in Table 8.1.*

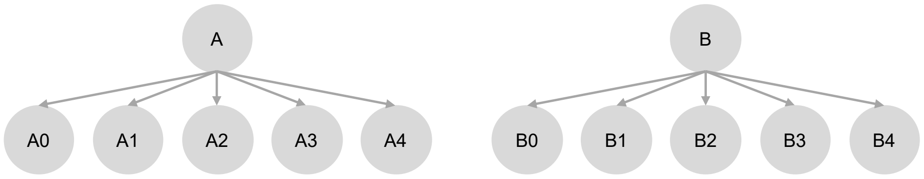


Figure 8.2: Nodes with the same amount of children

Node	A0	A1	A2	A3	A4
B0	1	0.8	0.6	0.4	0.2
B1	0.8	1	0.8	0.6	0.4
B2	0.6	0.8	1	0.8	0.6
B3	0.4	0.6	0.8	1	0.8
B4	0.2	0.4	0.6	0.8	1

Table 8.1: Position value when nodes have the same number of siblings

Example 8.2.2 *Figure 8.3 shows two DOM nodes, A and B. DOM node A has 5 children while DOM node B has 4. The position computation algorithm for their child nodes will obtain the values shown in Table 8.2.*

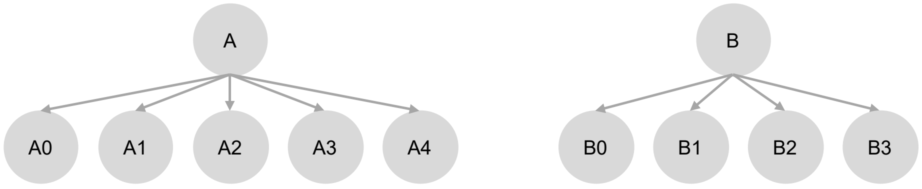


Figure 8.3: Node A with more children than B

Example 8.2.3 *Figure 8.4 shows two DOM nodes, A and B. DOM node B has 4 children while DOM node B has 5. The position computation algorithm for their child nodes will obtain the values shown in Table 8.3.*

Node	A0	A1	A2	A3	A4
B0	0.8	0.8	0.8	0.6	0.4
B1	0.6	0.6	0.8	0.8	0.6
B2	0.4	0.4	0.6	0.8	0.8
B3	0.2	0.2	0.4	0.6	0.8

Table 8.2: Position value when DOM node A has more children

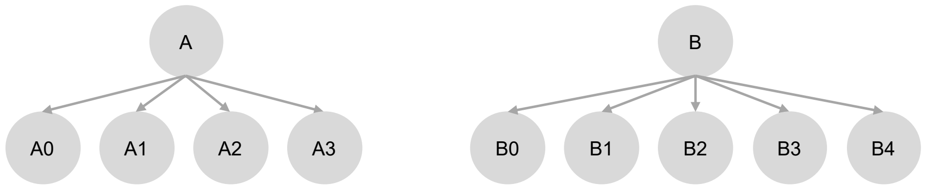


Figure 8.4: Node B with more children than A

Node	A0	A1	A2	A3
B0	1	0.75	0.5	0.25
B1	1	1	0.75	0.5
B2	0.75	1	1	0.75
B3	0.5	0.75	1	1
B4	0.25	0.5	0.75	1

Table 8.3: Position value when DOM node B has more children

- HTML attributes: The percentage of HTML attributes shared by both DOM nodes is computed. For instance, if all the attributes of node n_1 are contained by node n_2, the value of this property is 1.
- Node children: The algorithm counts the number of children of both nodes to compute a relation between them. For instance, if n_1 has 4 children and n_2 has 2 children, as n_2 has half of the children of n_1 the value of this property is 0.5.

The value of the equality relation $\triangleq$ between two DOM nodes is computed using the combination of the values obtained for each property, except for the HTML *tagName*, which has to be the same in both DOM nodes. However, it is possible that not all the properties have the same weight in the relation $\triangleq$, a ponderation based on experimentation should be established for each of the 4 last properties (all except HTML *tagName*). Then, the sum of the pondered values produces a value between 0 and 1 for the relation $\triangleq$, which is also a parameter. Therefore, in the comparison of two DOM nodes, n_1 and n_2, a threshold should be established for the value of the relation $\triangleq$. The template extraction experiments described in Section 8.3 show the values obtained for the 4 properties and for the relation $\triangleq$.

Compared to other mapping algorithms such as, e.g., the *restricted top-down mapping* (RTDM) described in [94], this mapping is more restrictive. For example, while RTDM can map different nodes (e.g., a node with the tagName *div* with a node with the tagName *section*), the pairwise mapped nodes with ETDM are forced to have the same tagName.

For instance, the example in Figure 8.1 shows an ETDM which uses: $n \triangleq n'$ based only on the tagName of n and n'.

Once the CS is built, the algorithm identifies an ETDM between the key page and the web pages from the CS. This process considers that the template is initially empty. Then, the ETDM is iteratively computed between the key page and v web pages in the CS, being v the number of votes which are needed to consider a node as part of the template. Note that the maximum number in the CS is n, that is, $n \leq v$. Finally, the obtained DOM tree is a template that includes all those nodes of the key page which appear in at least v other web pages of the CS. The described process is formalized in Algorithm 10, which computes the biggest ETDM between a set of DOM trees through the function *ETDM*. The algorithm contains a loop (`foreach` $(\{p_1 \dots p_v\}$ `in` $P)$) that iterates over all the partitions of P that can be formed with v pages. Then, the algorithm computes an ETDM between the key page and those web pages. Note that the recursive function *ETDM* traverses the DOM trees top-down selecting all those nodes computed as equal using the equality relation $\triangleq$. It should be highlighted that, given two web pages, function *ETDM* maps only one node from the first web page with one node of the second. That is, given two web pages $p_1 = (N_1, A_1)$, $p_2 = (N_2, A_2)$, only one node $n_1 \in N_1$ satisfies $n_1 \triangleq n_2$ for a given $n_2 \in N_2$. 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 implement an additional mechanism to ensure the selection of only one node (either n_1 or n_1').

As stated above and in Definition 4.2.7, the equality relation $\triangleq$ in Algorithm 10 has been left as a parameter. Therefore, researchers can establish the relation of equality between two DOM nodes depending on their needs. For instance, a researcher can establish that two nodes are the same if they share the same *tagName* and position, while another researcher can en-

Algorithm 10 Extract a template from a set of web pages

```
Input: A key page $p_k = (N, A)$, a set $P$ of $n$ web pages, 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
  $template = (N_t, A_t) = (\emptyset, \emptyset)$;
  foreach ($\{p_1 \dots p_v\}$ in $P$)
      if $root(p_k) \triangleq root(p_1) \triangleq \dots \triangleq root(p_v)$
          $(N', A') = ETDM(p_k, p_1, \dots, p_v)$;
          $(N_t, A_t) = (N_t \cup N', A_t \cup A')$;
          $template = (N_t, A_t)$;
  return $template$;
end

function $ETDM$(tree $T_0 = (N_0, A_0)$, tree $T_1 = (N_1, A_1)$, ..., tree $T_v =$
$(N_v, A_v)$)
  $r_0 = root(T_0)$; $r_1 = root(T_1)$; ...; $r_v = root(T_v)$;
  nodes = $\{r_0\}$;
  edges = $\emptyset$;
  foreach $n_0 \in N_0$, ..., $n_v \in N_v$ . $n_0 \triangleq \dots \triangleq n_v, (r_0, n_0) \in A_0$, ...,
$(r_v, n_v) \in A_v$
      $(nodes\_st, edges\_st) = ETDM(subtree(n_0), \dots, subtree(n_v))$;
      $nodes = nodes \cup nodes\_st$;
      $edges = edges \cup edges\_st \cup \{(r_0, n_0)\}$;
  return $(nodes, edges)$;
end function
```

hance the restriction by considering also the *className* of the DOM node. Besides, the configuration of the equality relation $\triangleq$ has an impact on the recall and the precision of the technique. In fact, the more restrictive the equality relation $\triangleq$ is, the more precision (and less recall).

8.3 Implementation

This mapping algorithm has been implemented as a module of the block detection architecture described in Section 12.2, which includes the building of a CS, the ETDM algorithm, etc. The mapping module can be used by any block detection technique to infer the desired block. For instance, it is used by the template detection technique described in Chapter 9 (TemEx), and the content extraction technique explained in Chapter 10 (ConEx).

As shown by the following 2 chapters, both techniques (TemEx and ConEx) include the ETDM and have been implemented as a WebExtension.

The empirical evaluation of the ETDM in this chapter refers to the template detection algorithm in Chapter 9. The content extraction algorithm in Chapter 10 also includes the empirical evaluation of the ETDM (see Section 10.3.1).

8.3.1 Empirical evaluation

The theoretical formalization of the algorithm reveals some parameters that have been left open (the size of the CS, the number of votes, the weight of the properties of the equality relation $\triangleq$, and the threshold of the value of the equality relation $\triangleq$). This section computes the value of all these parameters based on experimental analysis.

First of all, the parameter n as the optimal size of the CS, and the parameter v as the number of votes needed to state a node as part of the template should be computed. Once the values n and v are computed, the weight of the different properties that form the equality relation $\triangleq$ has to be estimated. Finally, a value is needed to determine the threshold t of the equality relation $\triangleq$. Then, if $n_1 \triangleq n_2 \geq t$ both nodes are equal.

Determining the values of n and v

Algorithm 9 in Chapter 7 explores the sorted list of hyperlinks on the key page (output of Algorithm 8) to extract an n-CS containing n mutually linked web pages. As stated in Chapter 7, it is important to determine the optimal size of n because the larger the CS is, the more time is needed to compute the CS and execute the ETDM. In addition, experiments in Section 8.3 of Chapter 7 reveal that a larger CS does not guarantee better template extraction results.

On the other hand, the number of web pages that must contain a DOM node to consider it as part of the template (parameter v) has to be computed. The value of v is highly related to n. That is, a high value of v, near to n is probably excessively restrictive due to it forces a node to appear in almost all web pages of the CS to consider it as a template node. However, a low value of v could be insufficient due to it can consider almost all nodes in all web pages of the CS as template nodes.

Determining the values of the properties of the equality relation

Section 8.2 considers 5 properties of the DOM nodes whose combination produces a value for the equality relation $\triangleq$. Not all these properties have the same value. For instance, HTML *tagName* is a required property. That is, both nodes must share their tagName in order to be mapped. The rest of the properties are ponderated through experimentation to determine their weight in the equality relation $\triangleq$. That is, the sum of the 4 ponderated values establishes another value corresponding to the equality relation $\triangleq$.

Determining the equality relation threshold parameter

Function $ETDM$ in Algorithm 9 computes the equality relation $\triangleq$ of the nodes of the DOM trees that belong to the CS. It explores top-down the DOM trees of the web pages in the CS and compares their nodes with the nodes in the DOM tree. The equality relation $\triangleq$ of two nodes n_1 and n_2 considers that both nodes are equal if $n_1 \triangleq n_2 \geq t$, being t a threshold parameter, which takes a value between 0 and 1. If t is not a parameter, both nodes would need to have exactly the same properties to be equal. For instance, to be equal, two nodes n_1 and n_2 would need to have the same HTML *tagName*, node position, node class names, number of children and HTML attributes (considering that all properties are relevant for the algorithm).

Computing the parameters

The best combination of values was computed for all the parameters of the algorithm: the values of n, v and t, and the weight of the properties that form the equality relation $\triangleq$. For this, the training method followed these steps:

i. First, the training subset of 105 web pages of the TeCo benchmark suite (see Chapter 13) was selected as input. Concretely the same subset that was used in other algorithms (see, e.g., Chapters 5, 6, and 10). Note that the benchmark suite is prepared for template detection, among others. Then, the system was executed for several combinations of parameters.

ii. Next, we proved the ETDM by performing several experiments with the template extractor described in Chapter 9 which implements this technique. The recall and precision were measured for each different possible combination of values for the thresholds and properties.

iii. Finally, the best combination of thresholds and properties was selected and evaluated against the evaluation subset of 45 web pages of the benchmark suite.

As the benchmark suite is prepared for template detection, each benchmark is labelled with some HTML classes that indicate which parts of the web page do not belong to the template. Therefore, any technique can automatically validate its recall and precision.

First of all, the ideal size of the CS and the number of web pages that must contain a DOM node to consider it as part of the template were evaluated empirically. As in Section 8.3 of Chapter 7, we selected 35 benchmarks (7 from each category) from the 105 benchmarks of the training subset of the TeCo benchmark suite (see Chapter 13). Results are shown in Table 8.4. Note that it is the same table as Table 7.7, but it also includes the `Votes` value.

`Size`	`Votes`	`Recall`	`Precision`	`F1`	`Loads`	`Runtime`
1	1	83.69 %	82.59 %	76.21 %	1	1303 ms.
2	2	91.92 %	83.69 %	84.45 %	8.86	2035 ms.
3	2	91.14 %	94.37 %	91.33 %	12.63	3378 ms.
4	2	91.67 %	93.51 %	91.15 %	18.11	3309 ms.
5	3	91.23 %	94.13 %	91.25 %	26.74	6467 ms.
6	3	91.39 %	93.92 %	91.22 %	28.66	6998 ms.
7	4	90.57 %	94.52 %	91.06 %	30.11	7596 ms.
8	4	91.99 %	93.06 %	91.37 %	32.71	7851 ms.

Table 8.4: Determining the optimum size of the complete subdigraph

Each row in the table shows the average of the repetition of all the experiments in the training subset of the benchmark suite using a different value for n in the n-CS obtained by the algorithm, and using a different value for all $v < n$. Thus, it summarizes the evaluation of all possible combinations. Column `Size` is the parameter n (the size of the CS), and column `Votes` is the best parameter v computed for each CS size. It is possible that a CS of the searched size does not exist for some websites, in that case the algorithm uses the biggest CS that can be built with a size under the specified size. Column `Loads` is the average number of web pages that need to be loaded to extract the template. It should be noted that it includes the load of the key page. Column `Runtime` shows the runtime of the algorithm relative to the selected options.

Table 8.4 shows that the best value for the size of the complete subdigraph is 3 (parameter n), because it gets almost the best F1 value while it is very efficient (it only needs to load 10 web pages in average to extract the template). This result was also obtained in the empirical evaluation of the candidates selection algorithms (see Section 7.3.1). For a n value of 3, the best result is obtained with v equal to 2.

Then, the weight of each property of the equality relation $\triangleq$ was computed ($\triangleq = A * \textit{Node position} + B * \textit{Node class names} + C * \textit{Node children} + D * \textit{HTML attributes}$), where $A + B + C + D = 1$). All the experiments were repeated with the following possible values for the weightings used:

Node position:	[0.00 – 1.00] in steps of 0.1.
Node class names:	[0.00 – 1.00] in steps of 0.1.
Node children:	[0.00 – 1.00] in steps of 0.1.
HTML attributes:	[0.00 – 1.00] in steps of 0.1.

In addition, the threshold of the equality relation $\triangleq$ (t) was also evaluated for each possible weighting with the following values:

$\triangleq$ *t:* [0.10 – 1.00] in steps of 0.10.

Finally, it should be highlighted that the computation of the *Node position* value can be done in four ways:

- Option 1: Once the mapping algorithm maps two nodes, the *Node position* value of their sibling nodes is computed again. In addition, this option considers the position of the matched nodes in the computation of the *Node position* values for their sibling nodes, namely, it two nodes n_0 and n_1 have been paired, it is not possible to pair a sibling node located on the left side of n_0 with a sibling node located on the right side of n_1 (and vice versa). This process is repeated until there is not possibility of match with the remaining nodes.

- Option 2: This option is similar to *Option 1*, that is, once two DOM nodes are mapped, the *Node position* value of their sibling nodes is computed again. In this case, the position of the matched nodes is considered in the computation of the *Node position* values for their siblings even if it is not restrictive. Hence, if two nodes n_0 and n_1 have been paired, it is possible to pair a sibling node located on the left side of n_0 with a sibling node located on the right side of n_1 (and vice versa), but their *Node position* value is equal to 0. This process is also repeated until it is not possible to obtain a match with the remaining nodes.

- Option 3: In this option, after the mapping of two nodes, the algorithm recomputes the *Node position* value for their sibling nodes, but in this case, their *Node position* value is not influenced by their position with respect to the previously mapped nodes. For instance, if two nodes n_0 and n_1 have been paired, the *Node position* value of their sibling nodes is normally computed, regardless their relative position to n_0 and n_1. This process is repeated after each mapping until there is not possibility of match with the remaining nodes.

- Option 4: The last option does not consider the previously mapped nodes in the computation of the *Node position* value. In this case, when the algorithm starts the mapping process, it computes the *Node position* value of the nodes, and the obtained number is not changed during the mapping process despite the obtained pairs. For instance, if the algorithm has to map the children of a DOM node n_0 with the children of another node n_1, it will initially compute the *Node position* value that will not be recalculated during the mapping process.

Note that the attribute HTML *tagName* has not been included in the list because both nodes have to share the same HTML tag name to be mapped. Table 8.5 presents the best 20 computed combinations after evaluating all possibilities against the training subset of the TeCo benchmark suite (see Chapter 13). It summarizes many experiments, since each row in the table corresponds to the average of 105 template extractions from 105 different web pages. The first 4 columns correspond to the weight of the 4 previously described properties (*Node position*, *Node class names*, *Node children*, and *HTML attributes*). Column $\triangleq$ `thres.` shows the threshold of the equality relation $\triangleq$; column `Option` represents how the *Node position* value is computed; columns `Recall`, `Precision`, and `F1` are the recall, precision, and F1 respectively. Finally, column `Time` indicates the average runtime of the algorithm for this combination of parameters.

The combination of all parameters produced a total of 1201200 experiments, which were performed to build the table with a total computing time of approximately 63 days using an Intel i9 9900k.

The first row in the table was selected as the optimum combination of parameters for the equality relation $\triangleq$, as it is the fastest combination that obtains the best F1 metric.

Finally, the following table summarizes the optimal parameters obtained empirically:

Node class.	Node pos.	HTML att.	Node ch.	$\triangleq$ thres.	Option	Recall	Precision	F1	Time
0.1	0.1	0.5	0.3	0.7	2	92.16 %	90.21 %	88.66 %	6870 ms.
0.1	0.4	0.0	0.5	0.9	3	90.44 %	92.07 %	88.64 %	9154 ms.
0.1	0.5	0.0	0.4	0.9	3	90.60 %	91.80 %	88.56 %	7955 ms.
0.1	0.5	0.1	0.3	0.9	3	90.60 %	91.72 %	88.47 %	8628 ms.
0.1	0.4	0.0	0.5	0.9	4	90.53 %	91.71 %	88.45 %	2865 ms.
0.1	0.5	0.0	0.4	0.9	4	90.69 %	91.45 %	88.38 %	2534 ms.
0.1	0.1	0.0	0.8	0.8	4	90.70 %	91.39 %	88.32 %	2510 ms.
0.1	0.1	0.2	0.6	0.8	4	90.70 %	91.41 %	88.31 %	2607 ms.
0.1	0.0	0.0	0.9	0.8	4	90.48 %	91.54 %	88.30 %	2798 ms.
0.1	0.5	0.1	0.3	0.9	4	90.67 %	91.38 %	88.30 %	3170 ms.
0.1	0.1	0.0	0.8	0.8	3	90.70 %	91.37 %	88.30 %	6609 ms.
0.1	0.0	0.0	0.9	0.8	2	90.48 %	91.54 %	88.30 %	7124 ms.
0.1	0.0	0.0	0.9	0.8	3	90.48 %	91.54 %	88.30 %	7996 ms.
0.1	0.1	0.2	0.6	0.8	3	90.69 %	91.37 %	88.29 %	6273 ms.
0.0	0.6	0.0	0.4	0.9	4	90.62 %	91.44 %	88.25 %	2581 ms.
0.1	0.0	0.4	0.5	0.8	4	90.47 %	91.47 %	88.23 %	3348 ms.
0.1	0.0	0.1	0.8	0.8	4	90.48 %	91.47 %	88.23 %	3353 ms.
0.1	0.0	0.4	0.5	0.8	3	90.47 %	91.47 %	88.23 %	6429 ms.
0.1	0.0	0.1	0.8	0.8	3	90.48 %	91.47 %	88.23 %	6780 ms.
0.1	0.0	0.4	0.5	0.8	2	90.47 %	91.47 %	88.23 %	6986 ms.

Table 8.5: Determining the best values of the equality relation $\triangleq$ properties

CS size (n):	3.
Number of votes (v):	2.
Node class names:	0.10.
Node position:	0.10.
HTML attributes:	0.50.
Node children:	0.30.
$\triangleq$ *threshold (t):*	0.70.
Option:	2.

Note that these parameters were obtained for both techniques (candidates selection and ETDM) applied to template extraction. For content extraction or other block detection techniques, as demonstrated in Section 10.3.1, they could be different.

8.4 Conclusions

Site-level block detection algorithms obtain valuable information from comparing DOM nodes from different web pages from the same website. This Chapter presents a new mapping called Equal Top-Down Mapping (ETDM) useful to check whether two DOM nodes in two different web pages are the same or not. Two DOM nodes are equal if they meet two conditions: they are equal regarding the equality relation $\triangleq$, and all their ancestors are equal. In addition, it should be highlighted that the notion of node equality is based in some properties of the DOM nodes, such as their HTML tagName, their position in the DOM tree, their class names, the number of children they have, and their HTML properties. The combination of these properties, as a sum of pondered values, produces a value for the equality relation $\triangleq$, which is also a parameter.

8.5 Contributions

The Equal Top-Down Mapping described in this chapter provide several contributions that can be included in site-level block detection systems.

The chapter introduces the concept of ETDM, where two nodes are considered equal if they and their ancestors meet the equality relation $\triangleq$.

Another contribution of this chapter is the equality relation $\triangleq$, which is not a simple equality relation (=). The equality relation $\triangleq$ is based on the following properties: the HTML *tagName*, the class names of the DOM node, the node position, the HTML attributes, and the number of children a node has.

The functional implementation of this mapping as a module is also an important contribution. The module is part of a block detection architecture, so it can be used by many site-level block detection techniques.

Chapter 9

Site-level Template Detection

Template detection is a topic highly related to web development, web mining, indexing, and searching. Over the last 15 years, many approaches have tried to address this challenging problem. Gibson et al. [44] studied and measured web templates and concluded that they account for between 40% and 50% of data on the Web, which means that templates include approximately 30% of the visible terms and hyperlinks. This fact gives relevance to template removal techniques [120, 114] for tasks such as web mining, searching, indexing, etc.

Some boilerplate removal techniques were developed in the context of CleanEval [20], which was a periodical competition that proposed a suite of benchmarks and their corresponding gold standard that provided a framework to assess the techniques.

As stated in Chapter 2, *Content Extraction* and *Template Detection* are very close disciplines. They both belong to a more general discipline called *Block Detection* which attempts to isolate the pagelets in a web page. On the one hand, content extraction attempts to detect and isolate the pagelets with the main content of the web page. On the other hand, template detection tries to detect (or detect and remove) the template of the web page, which, in most cases corresponds to the part of the web page that is not the main content (they are often complementary). There are many content extraction techniques (see, e.g., [46, 117, 24, 51]), which are all closely related to *Template Detection.*

This chapter describes a site-level technique (TemEx) that detects and isolates the template of a web page.

9.1 Related work

Chapter 2 describes the three main approaches in the *Block Detection* area, concretely, (i) using the textual information provided by the web page, (ii) based on the rendered image of the web page, and (iii) analyzing the DOM tree of the web page.

This technique is based on the third approach, namely, it obtains the information needed to infer the template from the DOM tree. Some works in literature try to identify pagelets using heuristics to analyze the DOM tree [19], while others are based on the detection of common subtrees in the DOM trees of a set of web pages from the website [120, 114]. The technique described in this chapter is similar to these last two works.

The technique presented in [120] is not a template extraction algorithm; even so, its aim is to remove redundant parts of a website. As explained in Section 8.1, their algorithm analyzes a set of DOM trees obtained from several web pages from the same website, and it builds a data structure called Site Style Tree (SST). An SST gathers information about all the DOM nodes from the analyzed DOM trees so that it uses counters to identify repeated nodes in the DOM trees. The resulting structure is a kind of summary of the set of DOM trees. Once the SST is built, it contains data about the most repeated nodes on the website. As the most repeated DOM nodes in all the analyzed web pages most likely belong to the template, the most repeated DOM nodes in the SST form the noisy information to be removed.

Other approaches like [114], as this technique, perform mappings between several DOM trees obtained from a set of web pages from a website. The final goal of these mappings is to identify repeated nodes in several web pages. Their technique uses an algorithm called RTDM-TD that computes a mapping called *restricted top-down mapping* [94] between several web pages of the same website. Even though their objective is also template extraction, there exist significant differences with the technique presented in this chapter. On the one hand, their mapping is less restrictive because all paired nodes in the mapping are not required to be equal. On the other hand, their technique selects the web pages that should be mapped randomly, limited by a predefined threshold. They define such threshold as a few dozen of web pages. Therefore, they assume that the same template is shared by all the web pages that form the website, which is a severe limitation for many websites, particularly for large websites.

The authors of [114] proposed an algorithm called RBM-TD in [113], which is an approach similar to RTDM-TD that uses a bottom-up alternative of their mapping (contrary to the top-down variant they use in [114]). One of the innovations of this approach is that it describes a constraint to classify a common subtree between several DOM trees as part of the template: those subtrees that can be found in all web pages must be located exactly in the same position (i.e., the path in the DOM tree from the root to them must be exactly the same for all web pages).

Other approaches like [116] analyze the web pages and divide their DOM trees into several subtrees. The root nodes of the obtained subtrees are DOM nodes related to concrete HTML tags (i.e., DIV, UL, TABLE, etc.). Hence, the algorithm extracts the DOM nodes of type #text (text segments) of the subtrees and compares them with those of all web pages.

Authors of [60] propose a quantitative analysis of Web content based upon techniques from the field of *Quantitative Linguistics*. They argue that the text corpus exposes two fuzzy classes of text: covering full text and navigational information. Template text can be divided into these classes: noisy (short navigational text), and frequently used (full-text).

Sivakumar [103] presented a main content extraction algorithm based on removing several kinds of noises from the web page. The noises are removed by performing three operations. First, the algorithm removes primary noises and partitions the useful text contents into blocks. Then, the duplicate blocks are removed to obtain the distinct blocks. Finally, the algorithm computes the importance of each block based on three parameters. The important blocks are selected based on a threshold value. Saravanan and Bama [100] developed a technique similar to [103] that extracts useful content from web pages by removing noise and duplicates. The technique is also divided into three steps. First, the algorithm divides the web page into several blocks and the block that is considered as noise is removed. Then, the algorithm performs the elimination of redundant blocks. Finally, the algorithm computes several parameters from the remaining blocks to decide whether a block is main content or not.

Aslam et al. [18] extended the Boilerpipe algorithm [63] with their Web-AM algorithm. The authors detected that Boilerpipe has very good performance in main content extraction, but that performance decreased while filtering the noise in the main article. First, their algorithm extracts the main article from the web page using Boilerpipe. Then, they remove the noise retrieved by Boilerpipe using a rule-based algorithm.

Most recent techniques are based on *Machine Learning*. Vogels et al. [115] use a technique based on convolutional neural networks to classify all text blocks in an HTML page as either boilerplate or main content. In [66], authors propose a machine learning model that removes the boilerplate by taking as input only the HTML tags and words that appear on a web page. Zhang and Wang [123] developed SemText, a hierarchical neural network model that detects web page boilerplate using a novel semantic representation of text blocks.

9.2 Template detection

The presented template detection approach (TemEx) takes as input a webpage (the key page) and outputs its template. As it is a site-level technique, it identifies, loads, and analyzes several web pages from the website to infer the template. The main ideas introduced by this technique are:

- To increase the performance, the number of web pages loaded and analyzed should be minimum. This technique analyzes the links in the key page and identifies a *complete subdigraph* in the website topology (see Chapter 7).
- The conflicts between web pages implementing different templates should be solved efficiently. For this, the algorithm establishes a *voting system* for the web pages in the complete subdigraph.
- The template is detected and extracted by comparing the web pages in the complete subdigraph. For that, it uses the mapping described in Chapter 8, called *equal top-down mapping*.

The technique consists of a three-step approach:

i. Based on the candidates selection algorithms described in Chapter 7, a set of web pages from the same website of the key page is selected.

ii. Then, an algorithm explores each web page in the set and computes an equal top-down mapping (see Chapter 8) between its DOM nodes and the DOM nodes of the key page. When it finds that a DOM node of the key page is repeated in any web page of the set, it updates a counter that reflects the number of times each node is repeated in other web pages.

iii. Finally, an algorithm explores the DOM tree and those nodes whose counter is higher than a specified threshold are selected as template nodes.

9.2.1 The web page's template

When we observe the rendered image of a web page it is trivial for us, humans, to identify the template with accuracy despite not comparing the web page with other web pages from the same website. However, we can infer the template with more accuracy by comparing the web page with other web pages from the same website. Then, we can observe the part of the web page repeated in other web pages, so we can state that the repeated parts of the web page belong to the template with high probability.

The same reasoning may be applied to automatic template detection. The web page can be represented as a DOM tree that can be compared with other DOM trees from other web pages from the same website. The DOM nodes repeated in several web pages can be considered part of the template.

A definition of a template, independent of any detection method, can be provided based on the repetition of a set of DOM nodes through different web pages of the same website. The formal definition of the main content (see Definition 4.3.2) is based on the definition of a web page (see Definition 4.1.1) and the definition of a website (see Definition 4.1.7). This definition of the template of a web page $P = (N, A)$ is based on the following assumptions:

- There exist other web pages from the same websites different to P.
- The set of DOM nodes considered as the template is repeated in other web pages from the same website.

Example 9.2.1 *In Figure 9.1, we can observe part of three DOM trees from three different web pages that belong to the same website. The DOM tree on the left belongs to the key page while the other two DOM trees belong to two other web pages. Note that the DOM nodes with a dark background are repeated in the three web pages. Based on the definition of template, those DOM nodes would form the template of the key page because they are equal and repeated in several different web pages. Note that in this example we have considered a DOM node as part of the template if it appears on the three web pages.*

9.2.2 Building a complete subdigraph

First, a set of web pages sharing their template should be identified. This phase is an independent process described in Chapter 7 which can be used in combination with any block detection technique.

It can be observed, by analyzing the menu of several websites, that mutually linked web pages from the same domain very likely share a common template. This phase identifies a set of mutually linked web pages which share their template with the key page. The web pages in this set are pairwise and mutually linked. Therefore, they form an n-complete subdigraph (n-CS), consisting of n nodes.

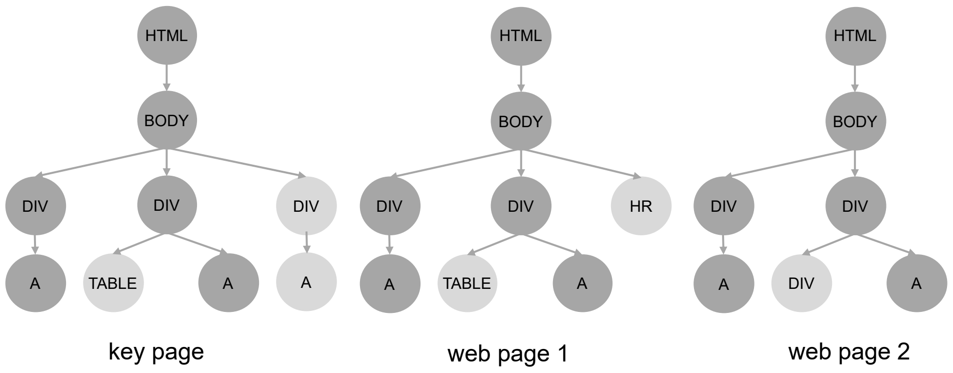


Figure 9.1: Example of DOM nodes repeated in several web pages

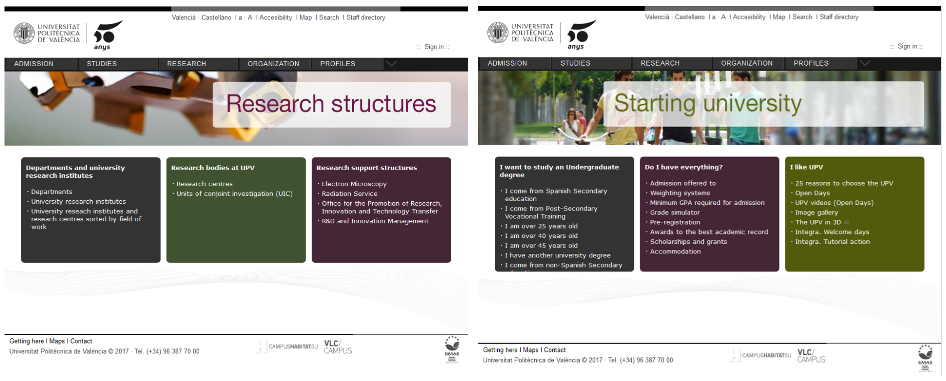

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Valencià · Castellano I a · A I Accessibility I Map I Search I Staff directory
:: Sign in ::
ADMISSION
STUDIES
RESEARCH
ORGANIZATION
PROFILES
Research structures
Departments and university research institutes
Research bodies at UPV
Research support structures
Getting here I Maps I Contact
Starting university
I want to study an Undergraduate degree
Do I have everything?
I like UPV

Figure 9.2: Web pages of Polithecnical University of Valencia sharing a template

Example 9.2.2 *In Figure 9.2, we can observe two web pages of the Polytechnical University of Valencia that implement the same template. The website on the left can be reached from the menu option "Research". The web page on the right can be reached from the menu option "Admission". In both cases, the main content is located at the bottom of the web page. Both web pages share their header, menu, and footer. In addition, they form a 2-CS. Likewise, the rest of the web pages from the same domain linked by the menu located at the top form a complete subdigraph, and they all share the whole template. The candidates selection technique identifies this set of web pages as candidates.*

As explained in Chapter 7, this candidate selection technique improves the quality of the candidates because they share more template nodes with the key page. Moreover, the performance is also improved because only a reduced subset of web pages linked from the key page is analyzed. Chapter 7 also describes other approaches for selecting the candidate web pages [83, 120, 114]. Some of them use random web pages obtained from the website, while others take as input a set of web pages directly provided by the programmer.

Additionally, Chapter 7 describes several steps to build an n-CS. The hyperlinks in the key page are analyzed to select those that produce a complete subdigraph with high probability. The hyperlink analysis establishes an order of relevance using two algorithms: hyperlink distance, and DOM distance. Once the order of relevance is established, it is iteratively explored until the web pages pointed by the links in it form an n-CS. 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 farther from the already selected links are collected.

9.2.3 Web pages implementing several templates

Most of the techniques found in the literature assume that a website has a unique template. Nevertheless, it should be noted that, usually, a website implements several templates, including subsets of different templates. The following example describes that issue.

Example 9.2.3 *Figure 9.3 shows the DOM tree of three web pages: the key page and two web pages that provide information to extract its template. Both web pages that provide information about the template extraction process implement different templates, and they are disjoint except for their root node. Therefore, if we suppose that the same template is implemented by all web pages in the website, the template extraction process will only extract the root node because it is the only node that the three web pages share. On the other hand, if we assume the existence of several templates, we can observe that the template of the key page is built with a part of the template of one web page, and a part of the template of the other web page, being the template represented with the gray nodes. Thus, despite being disjoint, both web page candidates can contribute to the template.*

Example 9.2.3 shows that it is not necessary that all web pages share a node to consider it as part of the template. Based on experimentation, we

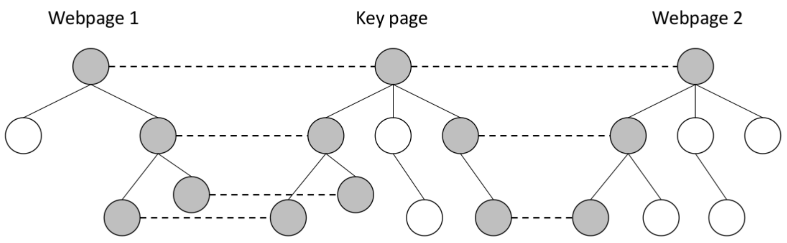


Figure 9.3: Template extracted from web pages implementing different templates

can state that the number of web pages that must share a node to consider it as part of the template depends on the size of the CS. Table 8.4 in Section 8.3.1 summarizes the experiments conducted to obtain the optimal number of web pages that must share a node for each CS size. For instance, for a CS of size 7, it is optimal to consider the nodes repeated in 4 web pages as part of the template to get the best F1. Consequently, the algorithm takes both, the size of the CS, and the number of web pages that must share a node, in the following votes, as parameters. Therefore, it can be configured for different CS sizes and different number of votes, if needed.

9.2.4 Template detection from a complete subdigraph

The DOM nodes in the DOM trees that form the CS must be compared in order to identify which of them appear in several DOM trees. This comparison is based on the *equal top-down mapping* (ETDM) described in Chapter 8, which, given two DOM trees, establishes a correspondence between their nodes.

The definition of ETDM is parametric with respect to the equality relation $\triangleq$ (see Section 8.2.1). Therefore, this notion of equality is more general than the standard equality (=). It considers several DOM node properties in order to compare two nodes, such as their class names, HTML *tagName*, number of children, HTML attributes, and their relative position in the DOM tree.

Once the *complete subdigraph* is built, an ETDM is computed between the key page and each web page in the complete subdigraph. The DOM trees are compared by traversing them top-down starting from the root. As shown in Figure 8.1 and described in Section 8.2.1, the ETDM is computed iteratively between the key page and the web pages that belong to the CS. Two nodes n_1 and n_2 can be mapped if they are equal ($n_1 \triangleq n_2$). Then, the algorithm recursively continues by trying to map the children of both mapped nodes. It should be noted that when it is not possible to map a node, the algorithm does not explore its descendants. A DOM node is considered to belong to the template when it appears in v web pages of the CS, so v is the number of votes needed to consider a node as part of the template. Note that $n \leq v$, being n the number of web pages in the complete subdigraph (n-CS). When the process finishes, the template is the obtained DOM tree, which is formed by DOM nodes from the key page that, at least, appear in v other web pages of the CS.

9.3 Implementation

This technique, as the rest of the techniques in this thesis, has been implemented as a WebExtension compatible with Firefox and Chromium-based browsers. As in the rest of the techniques, users browse on the Internet as usual, and then, when they want to extract the template of the web page loaded in the browser, they only have to press the "Extract Template" button. Then, the add-on automatically does the required actions and shows the template of the web page. Therefore, the nodes of the web page that do not belong to the template are hidden.

Example 9.3.1 *Figure 9.4 shows a real example of the use of the template detection tool with a web page. The image on the left is a web page of the* ***www.jdi.org.za*** *website. The image on the right is the output of extracting its template.*

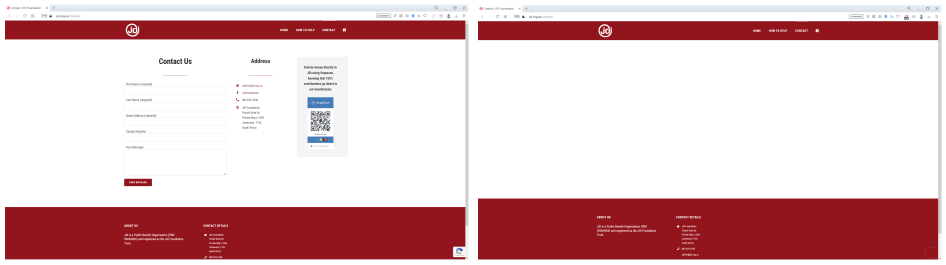


Figure 9.4: Example of the detection of a web page template

9.3.1 Empirical evaluation

The theoretical formalization of the algorithms in Chapter 7 and Chapter 8 reveals some parameters that have been left open (size of the CS, number of votes, the weight of the properties of the equality relation $\triangleq$, and the threshold of the value of the equality relation $\triangleq$).

The value of these parameters is computed based on an experimental analysis in the empirical evaluation section of both chapters (Chapter 7 and Chapter 8). The computation of the parameters was done using this technique, that is, combining both techniques (candidates selection and ETDM) for template detection.

As described in Chapter 7 and Chapter 8, the parameter n is the optimal size of the CS and the parameter v is the number of votes needed to mark a node as part of the template should be computed. Once the values n and v were computed, the weight of the different properties that form the equality relation $\triangleq$ was estimated. Then, a final value was computed to determine the threshold t of the equality relation $\triangleq$. Therefore, if $n_1 \triangleq n_2 \geq t$ both nodes are equal.

As stated in previous chapters, it is important to determine the optimal size of n because the CS size is directly proportional to the time needed to compute it. In addition, Section 7.3.1 of Chapter 7 concludes that larger CS sizes do not always obtain better template detection results.

Algorithm evaluation

To measure the technique, several experiments were performed using the 45 evaluation benchmarks of the TeCo benchmark suite (see Chapter 13). Once the template was detected, it was compared real template to compute the precision, recall and F1 scores of the algorithm.

Table 9.1 shows the results obtained for all the executed benchmarks with the optimal parameters obtained empirically in Chapter 8:

CS size (n):	3.
Number of votes (v):	2.
Node class names:	0.10.
Node position:	0.10.
HTML attributes:	0.50.
Node children:	0.30.
$\triangleq$ *threshold (t):*	0.70.
Option:	2.

The first column of the table shows the URLs of the website domains. For each benchmark, column `Nodes` shows the total number of DOM nodes contained in the DOM tree of the key page; column `Templ.` represents the number of DOM nodes in the gold standard (template); column `Retr.` indicates the number of DOM nodes detected by the tool as template nodes;

Benchmark	Nodes	Templ.	Retr.	Correct	Recall	Precision	F1	Load	Rt.	Rt. opt.
www.jdi.org.za	619	394	566	394	100.00 %	69.61 %	82.08 %	6	253 ms.	274 ms.
www.premiere-urgence.org	480	438	443	430	98.17 %	97.07 %	97.62 %	4	202 ms.	213 ms.
www.indiangaming.org	575	201	199	199	99.00 %	100.00 %	99.50 %	4	57 ms.	64 ms.
hispalinux.es	501	345	363	345	100.00 %	95.04 %	97.46 %	4	126 ms.	152 ms.
www.gktw.org	767	637	682	637	100.00 %	93.40 %	96.59 %	5	431 ms.	463 ms.
www.apnic.net	598	453	517	453	100.00 %	87.62 %	93.40 %	4	201 ms.	236 ms.
www.unicef.org	1037	656	656	656	100.00 %	100.00 %	100.00 %	5	223 ms.	264 ms.
www.klimabuendnis.org	851	525	495	491	93.52 %	99.19 %	96.27 %	4	238 ms.	264 ms.
www.isoc-es.org	259	159	164	159	100.00 %	96.95 %	98.45 %	6	112 ms.	128 ms.
biztechmagazine.com	1892	1053	1794	1053	100.00 %	58.70 %	73.98 %	6	771 ms.	845 ms.
www.eeo.com.cn	834	626	648	596	95.21 %	91.98 %	93.57 %	13	369 ms.	442 ms.
www.wishtv.com	2167	1811	1535	1535	84.76 %	100.00 %	91.75 %	5	5940 ms.	5846 ms.
news.mit.edu	2117	1041	1853	1041	100.00 %	56.18 %	71.94 %	4	1755 ms.	1790 ms.
asia.nikkei.com	869	662	661	660	99.70 %	99.85 %	99.77 %	6	540 ms.	526 ms.
www.rcnky.com	1738	1425	1635	1425	100.00 %	87.16 %	93.14 %	4	629 ms.	773 ms.
news.discovery.com	2826	1161	2424	1156	99.57 %	47.69 %	64.49 %	5	1241 ms.	1377 ms.
www.kathimerini.gr	1825	1541	1259	1259	81.70 %	100.00 %	89.93 %	12	837 ms.	983 ms.
news.un.org	1726	1252	1175	1095	87.46 %	93.19 %	90.23 %	6	530 ms.	569 ms.
frances.forosactivos.net	785	290	164	163	56.21 %	99.39 %	71.81 %	26	156 ms.	157 ms.
www.wysiwygwebbuilder.com	3936	735	315	287	39.05 %	91.11 %	54.67 %	143	4036 ms.	4066 ms.
www.3dprintforums.com	1040	276	267	267	96.74 %	100.00 %	98.34 %	6	271 ms.	264 ms.
www.strangehorizons.com	631	146	164	146	100.00 %	89.02 %	94.19 %	7	2238 ms.	125 ms.
communities.apple.com	3136	368	2351	368	100.00 %	15.65 %	27.06 %	5	930 ms.	1016 ms.
www.sloweurope.com	4193	514	498	491	95.53 %	98.59 %	97.04 %	5	216 ms.	242 ms.
community.ricksteves.com	2057	384	2057	384	100.00 %	18.67 %	31.47 %	7	3568 ms.	3702 ms.
hackercombat.com	1711	794	1006	792	99.75 %	78.73 %	88.00 %	16	1120 ms.	1138 ms.
www.scbwi.org	876	216	202	202	93.52 %	100.00 %	96.65 %	37	303 ms.	305 ms.
johngardnerathome.info	395	176	37	29	16.48 %	78.38 %	27.23 %	4	413 ms.	388 ms.
www.annmalaspina.com	392	182	228	182	100.00 %	79.82 %	88.78 %	4	73 ms.	73 ms.
foodsense.is	330	100	122	100	100.00 %	81.97 %	90.09 %	4	54 ms.	57 ms.
sites.google.com	372	287	334	286	99.65 %	85.63 %	92.11 %	4	140 ms.	123 ms.
whatever.scalzi.com	1648	1405	1423	1405	100.00 %	98.74 %	99.37 %	13	190421 ms.	5444 ms.
www.javiercelaya.es	740	668	444	433	64.82 %	97.52 %	77.88 %	4	143 ms.	133 ms.
diarium.usal.es	604	80	79	76	95.00 %	96.20 %	95.60 %	4	855 ms.	779 ms.
www.jameslovelock.org	653	458	498	458	100.00 %	91.97 %	95.82 %	5	2309 ms.	2118 ms.
www.cipri.info	933	377	375	375	99.47 %	100.00 %	99.73 %	4	199 ms.	225 ms.
naranjascarcaixent.com	290	148	151	147	99.32 %	97.35 %	98.33 %	5	61 ms.	74 ms.
www.technicalbookstoreonline.com	2959	386	367	367	95.08 %	100.00 %	97.48 %	32	430 ms.	472 ms.
www.floridarealestatecollege.com	1023	528	545	528	100.00 %	96.88 %	98.42 %	4	135 ms.	159 ms.
www.basf.com	827	762	792	762	100.00 %	96.21 %	98.07 %	4	211 ms.	246 ms.
www.mcphersonoil.com	831	600	624	600	100.00 %	96.15 %	98.04 %	4	464 ms.	427 ms.
www.thirteenhou.com	1217	133	136	132	99.25 %	97.06 %	98.14 %	4	41 ms.	51 ms.
www.embalajesterra.com	2342	1677	1783	1677	100.00 %	94.05 %	96.93 %	4	505 ms.	546 ms.
www.crypto.ch	338	248	258	248	100.00 %	96.12 %	98.02 %	4	194 ms.	208 ms.
www.shopbookshop.com	1727	1310	1313	1310	100.00 %	99.77 %	99.88 %	4	5500 ms.	5040 ms.
Average	1281.49	613.96	746.71	573.31	93.09 %	87.75 %	87.54 %	10.36	5099 ms.	951 ms.

Table 9.1: Experimental evaluation results

column `Correct` shows the number of correct DOM nodes detected by the tool; column `Recall` represents (in percentage) the number of DOM nodes correctly retrieved divided by the number of DOM nodes in the gold standard; column `Precision` shows (in percentage) the number of DOM nodes that have been retrieved correctly divided by the number of retrieved DOM nodes; column `F1` reveals the F1 metric; column `Load` indicates the total number of web pages loaded by the technique; column `Rt.` contains the total time used to compute the template (in milliseconds); finally, column `Rt. opt.` shows the total time used to compute the template implementing the runtime improvement described subsequently in this section.

It can be observed in Table 9.1 that experiments obtain an average recall higher than 93%, and an average precision and F1 higher than 87%. On average, the algorithm needed to load 10 web pages approximately from each website. However, as we can observe in the table, in almost half of the benchmarks the algorithm only needed to load 4 web pages (the key

Benchmark category	Recall	Precision	F1	Load	Rt.	Rt. opt.
Institutions / Associations	98.97 %	93.21 %	95.71 %	4.67	205 ms.	229 ms.
Media / Communication	94.27 %	81.64 %	85.42 %	6.78	1401 ms.	1461 ms.
Forums / Social	86.76 %	76.80 %	73.25 %	28.00	1426 ms.	1224 ms.
Personal websites / Blogs	86.16 %	90.03 %	85.18 %	5.11	21623 ms.	1038 ms.
Companies / Shops	99.29 %	97.07 %	98.15 %	7.22	838 ms.	803 ms.

Table 9.2: Experimental evaluation results grouped by category

page and 3 web pages to build the complete subdigraph). Regarding the runtime of the algorithm, the table shows an average runtime of more than five seconds. However, when the runtime improvement is used, the average runtime is reduced from 5 to less than 1 second. We can observe that, for most of the benchmarks, the runtime is similar whether using the runtime improvement or not. Nevertheless, there is one benchmark that reduces its runtime from more than 3 minutes to almost 5 seconds.

Table 9.2 shows the obtained results grouped by benchmark category (see Chapter 13). We can observe that `Institutions / Associations` and `Companies / Blogs` categories obtain very high F1 values (higher than 95%), while the `Forums / Social` category obtains a lower F1 value (about 73%). This result obtained for the `Forums / Social` category is consistent with the technique since we observed that in most cases the complete subdigraph for websites of this category includes web pages quite different from each other. For instance, in most forums, the complete subdigraph is formed by a web page with a list of discussions, a web page with a discussion thread, and another with a single message. This fact makes it difficult to obtain a good outcome when mapping the key page with the pages of the complete subdigraph.

Runtime improvement

Table 9.1 shows 3 benchmarks with a runtime higher than 5 seconds. Moreover, it should be noted that the runtime of one of the benchmarks is almost 190 seconds. This phenomenon occurs in some web pages and it is due to the runtime of the mapping phase, which is its main bottleneck.

We observed that the mapping of two nodes when they have a large number of children takes excessive time. This is due to the fact that when the position property value for two nodes with a large number of children is computed, each child of one node has to be compared with all the children of the other node. When two children are mapped, the position property value of the remaining children has to be computed, and so on. It should be highlighted that for more than 200 children, the runtime of the algorithm grows exponentially.

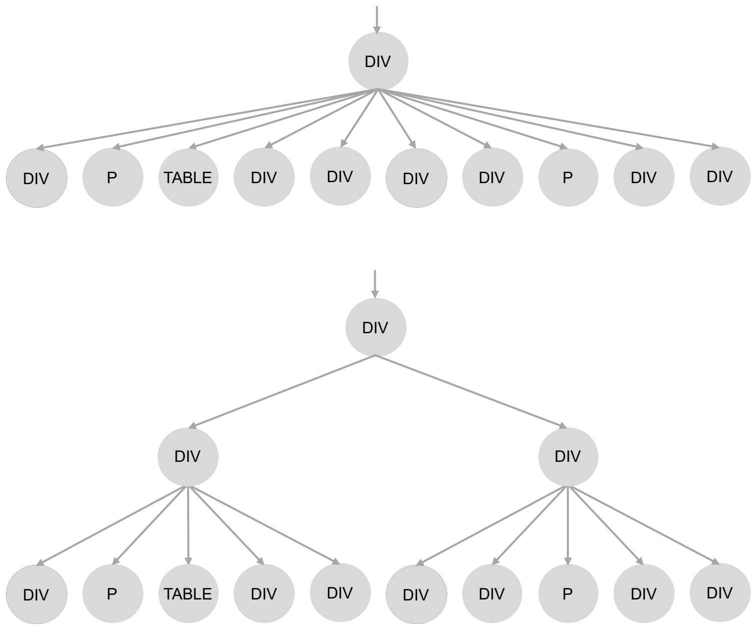


Figure 9.5: Transformation of a DOM node for runtime improvement

The reduction of the number of children of some nodes and thus, the reduction of the mapping runtime, is done by inserting some artificial (and temporary) nodes between the parent and the children. These temporary nodes group the children in a way that the number of children of a node is limited. For instance, if a node has 200 children, we can insert 4 temporary nodes between the parent and the children and each temporary node will have 50 children. This way, we have 4 parents with 50 children each instead of one parent with 200 children. It should be noted that the addition of the temporary DOM nodes does not affect the layout of the web pages.

Example 9.3.2 *Figure 9.5 shows, at the top, a "DIV" DOM node with 10 children. At the bottom, we can find the same "DIV" node and the same 10 children, but between them, there are 2 temporary "DIV" nodes. These nodes group the original children into two groups of 5 children each. This way, the maximum number of children of that DOM subtree is 5.*

To measure the gain obtained by applying the runtime improvement algorithm, we defined two parameters:

- Group: It determines the maximum number of child nodes of the temporary DOM nodes. In other words, it is the maximum size of the groups formed with the child nodes of the original node.
- Childnodes: It defines a threshold that corresponds to the number of children a DOM node should have for applying the runtime improvement algorithm.

It should be highlighted that, in some cases, the results obtained by the runtime improvement algorithm can be slightly different than the expected results. This is because the mapping of two temporary nodes could be different than expected if their parent nodes did not have exactly the same children. However, the impact of the runtime improvement is minimum. In fact, we used 3 decimal places for the `Group` and `Childnodes` values because for most of the combinations the obtained results are the same with 2 decimal places.

To test the algorithm, we performed several experiments using the 105 training benchmarks of the TeCo benchmark suite (see Chapter 13). We computed the `Recall`, `Precision`, `F1`, and `Runtime` of the mapping phase for `Group` values ranging from 25 to 125, and `Childnodes` values ranging from 100 to 350.

Table 9.3 shows the results obtained by different combinations of `Group` and `Childnodes` values. First, it can be observed that the `Runtime` of applying the improvement algorithm is significantly lower in all cases. Then, we can observe that the `Recall` is roughly the same for all combinations, but the `Precision` varies. For a `Childnodes` value equal or higher to 200, the results are repeated depending on the `Group` value while the `Runtime` grows. For this reason, we selected a `Group` value of 100, and a `Childnodes` value of 200, since the variation of the F1 score is less than 0.001 % and it is the combination with less `Runtime` that obtains such `F1`.

Runtime analysis

Figure 9.6 shows the relation between the runtime of the algorithm and the number of DOM nodes of the key page. It can be observed that despite the highest runtimes correspond to web pages with a higher number of DOM nodes, there is no direct clear relationship between these two variables.

In order to draw conclusions about the variables directly related to the runtime of the algorithm, we decided to perform a statistical analysis of some variables that we thought could be directly related to the runtime. The analyzed variables were:

Group	Childnodes	Recall	Precision	F1	Runtime
25	100	90.297 %	90.321 %	87.101 %	896 ms.
50	100	90.953 %	90.248 %	87.591 %	950 ms.
75	100	91.204 %	90.244 %	87.729 %	1017 ms.
100	100	89.850 %	90.178 %	86.549 %	979 ms.
125	100	89.896 %	90.173 %	86.613 %	1190 ms.
25	150	91.463 %	90.307 %	88.165 %	1038 ms.
50	150	92.120 %	90.227 %	88.653 %	1087 ms.
75	150	92.126 %	90.228 %	88.657 %	1137 ms.
100	150	92.149 %	90.230 %	88.669 %	1018 ms.
125	150	92.494 %	90.312 %	88.206 %	1164 ms.
25	200	92.159 %	90.205 %	88.660 %	1215 ms.
50	200	92.159 %	90.209 %	88.662 %	1285 ms.
75	200	92.159 %	90.210 %	88.663 %	1295 ms.
100	**200**	**92.159 %**	**90.211 %**	**88.663 %**	**1133 ms.**
125	200	92.159 %	90.211 %	88.663 %	1258 ms.
25	250	92.159 %	90.205 %	88.660 %	1334 ms.
50	250	92.159 %	90.209 %	88.662 %	1357 ms.
75	250	92.159 %	90.210 %	88.663 %	1405 ms.
100	250	92.159 %	90.211 %	88.663 %	1261 ms.
125	250	92.159 %	90.211 %	88.663 %	1347 ms.
25	300	92.159 %	90.205 %	88.660 %	1329 ms.
50	300	92.159 %	90.209 %	88.662 %	1372 ms.
75	300	92.159 %	90.210 %	88.663 %	1418 ms.
100	300	92.159 %	90.211 %	88.663 %	1289 ms.
125	300	92.159 %	90.211 %	88.663 %	1370 ms.
25	350	92.159 %	90.203 %	88.659 %	1840 ms.
50	350	92.159 %	90.208 %	88.661 %	1749 ms.
75	350	92.159 %	90.209 %	88.662 %	1830 ms.
100	350	92.159 %	90.210 %	88.663 %	1872 ms.
125	350	92.159 %	90.211 %	88.663 %	2050 ms.
No improvement		92.159 %	90.213 %	88.664 %	6870 ms.

Table 9.3: Results obtained for different `Group` and `Childnodes` parameters

- The size of the template measured in `number of template DOM nodes`. The choice of this variable is based on the assumption that the technique performs a mapping between several web pages to infer the template of the key page. Therefore, the size of the real template with high probability is related to the runtime.

- The `number of loaded web pages`. This variable is selected because the technique needs to load several web pages to build the complete subdigraph. The more web pages it loads, the more runtime.

- The `standard deviation of the number of children of the element DOM nodes`. The choice of this variable is due to the fact that, as stated above, the mapping process of DOM nodes with a high amount of children has a direct influence on the runtime, making it

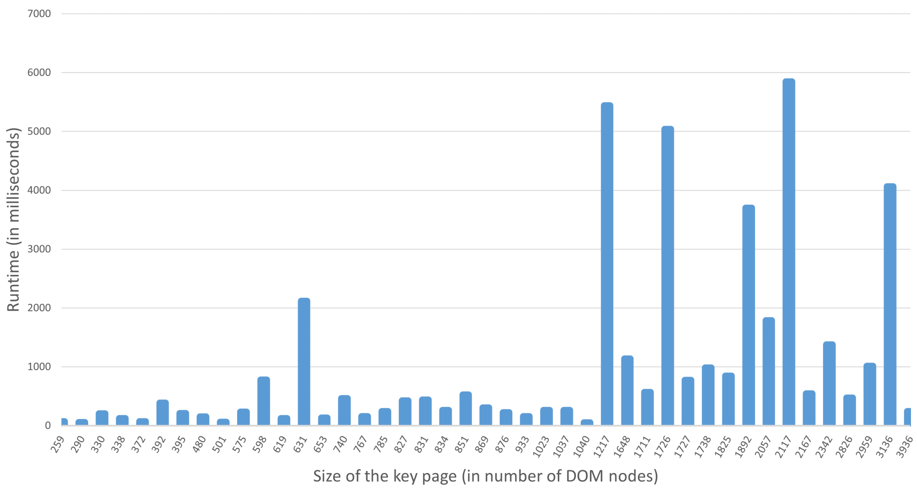


Figure 9.6: Relation between the size of the web page and the runtime

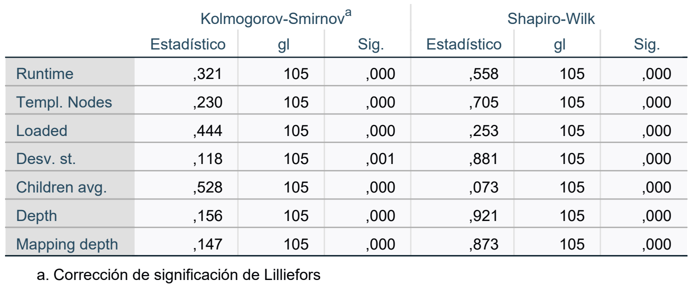
grow exponentially. The standard deviation of the number of children can indicate if there are DOM nodes with a high amount of children.

- The `average number of children of the element DOM nodes`. This variable is selected because the runtime of the algorithm grows exponentially due to DOM nodes with a high amount of children.

- The `depth of the DOM tree of the key page`. The choice of this variable is based on the assumption that, with high probability, the mapping of deeper DOM trees takes more runtime.

- The `maximum depth reached by the mapping`. In this case, the variable is selected because probably the more depth reached by the mapping involves more runtime.

It should be noted that this statistical analysis was performed to the technique using the `runtime improvement` algorithm, therefore, the DOM tree of the key page and the web pages that belong to the complete subdigraph are optimized for the mapping. As in Chapters 5 and 6 , we used IBM SPSS Statistics to analyze the relationship between these variables. We performed the analysis using the 105 benchmarks of the test subset of the TeCo benchmark suite (see Chapter 13). First, we had to check if the data were normally distributed, therefore, we computed Table 9.7. We



	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Runtime	,321	105	,000	,558	105	,000
Templ. Nodes	,230	105	,000	,705	105	,000
Loaded	,444	105	,000	,253	105	,000
Desv. st.	,118	105	,001	,881	105	,000
Children avg.	,528	105	,000	,073	105	,000
Depth	,156	105	,000	,921	105	,000
Mapping depth	,147	105	,000	,873	105	,000

a. Corrección de significación de Lilliefors

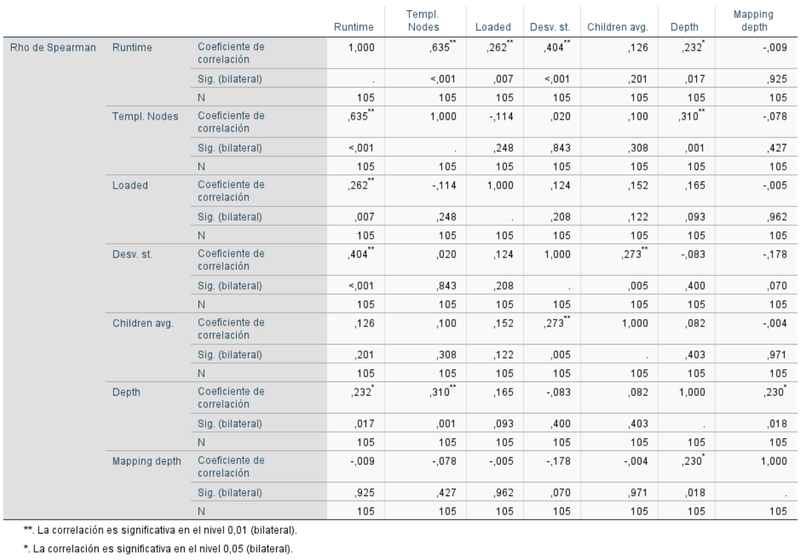
Figure 9.7: Result of the normality test for TemEx

can observe in the table that the `sample size` (column gl) is 105, so the appropriate test is Kolmogorov-Smirnov.

When the `significance` (column Sig.) is less than 0.05, the variable is not distributed normally, hence, the correlation coefficient can be computed through the Spearman test. We can observe in Figure 9.8 the result of the Spearman test for all variables. The first row shows the correlation coefficient between all the variables and the runtime. The correlation coefficient of the `number of template DOM nodes` variable is higher than 0.6, while the correlation coefficient of the `standard deviation of the number of children` variable is higher than 0.4. These values denote a clear relationship between these variables and the `runtime` of the algorithm. The correlation coefficient value for the variable `maximum depth reached by the mapping` is close to 0, therefore, it is a fact that this variable has no relationship with the `runtime`. On the other hand, Figure 9.8 also shows low values of the correlation coefficient for the variables `average number of children of the element DOM nodes`, `depth of the DOM tree of the key page`, and `number of loaded web pages`. These low values indicate that there is not a clear relationship between them and the `runtime` of the algorithm.

9.4 Conclusions

This chapter describes a new site-level template extraction technique. It sorts the hyperlinks in the key page using a hyperlink analysis technique. Once sorted, the technique selects a set of web pages pointed by those hyperlinks that form a CS and, consequently, they probably implement the same template. The DOM trees of the web pages in the CS are compared with the DOM tree of the key page using a mapping called ETDM



			Runtime	Templ. Nodes	Loaded	Desv. st.	Children avg.	Depth	Mapping depth
Rho de Spearman	Runtime	Coeficiente de correlación	1,000	,635**	,262**	,404**	,126	,232*	-,009
		Sig. (bilateral)	.	<,001	,007	<,001	,201	,017	,925
		N	105	105	105	105	105	105	105
	Templ. Nodes	Coeficiente de correlación	,635**	1,000	-,114	,020	,100	,310**	-,078
		Sig. (bilateral)	<,001	.	,248	,843	,308	,001	,427
		N	105	105	105	105	105	105	105
	Loaded	Coeficiente de correlación	,262**	-,114	1,000	,124	,152	,165	-,005
		Sig. (bilateral)	,007	,248	.	,208	,122	,093	,962
		N	105	105	105	105	105	105	105
	Desv. st.	Coeficiente de correlación	,404**	,020	,124	1,000	,273**	-,083	-,178
		Sig. (bilateral)	<,001	,843	,208	.	,005	,400	,070
		N	105	105	105	105	105	105	105
	Children avg.	Coeficiente de correlación	,126	,100	,152	,273**	1,000	,082	-,004
		Sig. (bilateral)	,201	,308	,122	,005	.	,403	,971
		N	105	105	105	105	105	105	105
	Depth	Coeficiente de correlación	,232*	,310**	,165	-,083	,082	1,000	,230*
		Sig. (bilateral)	,017	,001	,093	,400	,403	.	,018
		N	105	105	105	105	105	105	105
	Mapping depth	Coeficiente de correlación	-,009	-,078	-,005	-,178	-,004	,230*	1,000
		Sig. (bilateral)	,925	,427	,962	,070	,971	,018	.
		N	105	105	105	105	105	105	105

**. La correlación es significativa en el nivel 0,01 (bilateral).

*. La correlación es significativa en el nivel 0,05 (bilateral).

Figure 9.8: Result of the Spearman test for TemEx

that identifies the blocks that are common to the DOM trees. Some of the parameters have been approximated empirically: the size of the CS, the number of votes needed by a DOM node to be considered template, the properties that form the equality relation $\triangleq$, and the equality relation threshold. To the best of our knowledge, using a hyperlink analysis technique to select the web pages from the website that more likely implement the same template is new, and it quickly allows us to find a set of reliable web pages from which the template can be extracted. This is a key benefit for performance, since loading and analyzing web pages is expensive, and the presented technique minimizes this process. With the optimal parameters, this technique only loads 10 pages on average to extract the template. As a consequence, the runtime of the overall template extraction process is less than 1 second on average.

9.5 Contributions

The template detection technique described in this chapter provides several contributions that can be exploited by many systems, especially indexers and crawlers.

It describes the combination of two techniques in order to infer the template of a web page. The technique combines building a complete subdigraph with the web pages that more likely implement the template, with the comparison of those web pages through a mapping called equal top-down mapping.

Another important contribution of the chapter is a functional implementation of the technique as a WebExtension[1], which is compatible with Mozilla-based and Chromium-based browsers and is also officially published by Mozilla in their Firefox browser add-ons website[2].

[1]http://personales.upv.es/josilga/retrieval/Web-TemEx/index.html
[2]https://addons.mozilla.org/es/firefox/addon/template-extractor/

Chapter 10

Site-level Content Extraction

When we surf the Web, our attention is mainly focused on the main content of the web pages, which contain the useful information. Besides, extracting information from web pages is not only useful for humans, but also for many different systems. In a web page, the pagelet that contains the most relevant information is the main content. Nevertheless, this pagelet is often surrounded by other noisy elements such as main menus, footers, advertisements, banners, etc. Extracting the main content of a web page involves the isolation of the useful information from other elements by removing the useless elements (see, e.g., Figure 10.1).

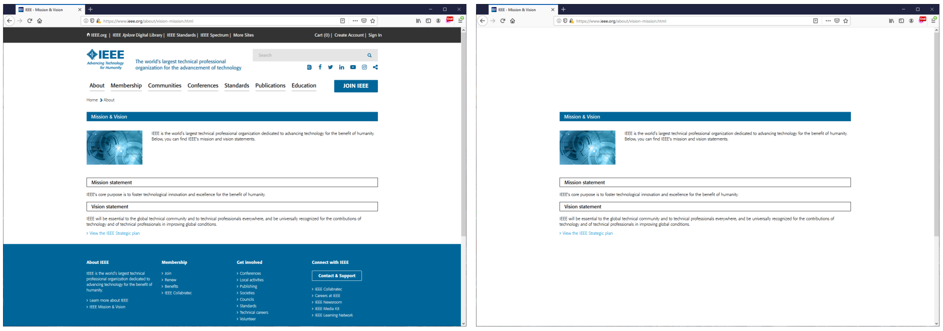
The main content is important for indexers and crawlers because processing the boilerplate of web pages may cause a waste of resources, such as bandwidth, storage, and time. Hence, indexers and crawlers preprocess the web pages to extract the main content and isolate it from the boilerplate. Moreover, the main purpose of indexers and crawlers is to provide users with only the relevant information. Therefore, extracting the main content is an essential task in order to preprocess that information. Gibson et al. [44] determined that boilerplate represents between 40% and 50% of all the data on the Web. As a result, it is mandatory to use techniques, such as main content extraction [117, 115] or template detection [120, 114], as a preprocessing method.

This chapter defines a site-level technique (site-level ConEx) that detects and extracts the main content of a web page.

10.1 Related work

As stated in previous chapters, content extraction, just as template detection, menu detection, and other block detection techniques, is an interesting topic due to its relation to web mining, searching and indexing.

Many content extraction techniques can be found in the literature (see,


IEEE - Mission & Vision
https://www.ieee.org/about/vision-mission.html
IEEE.org | IEEE Xplore Digital Library | IEEE Standards | IEEE Spectrum | More Sites
Cart (0) | Create Account | Sign In
IEEE Advancing Technology for Humanity
The world's largest technical professional organization for the advancement of technology
Search
About Membership Communities Conferences Standards Publications Education
JOIN IEEE
Home > About

Mission & Vision

IEEE is the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity. Below, you can find IEEE's mission and vision statements.

Mission statement

IEEE's core purpose is to foster technological innovation and excellence for the benefit of humanity.

Vision statement

IEEE will be essential to the global technical community and to technical professionals everywhere, and be universally recognized for the contributions of technology and of technical professionals in improving global conditions.

> View the IEEE Strategic plan

About IEEE

IEEE is the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity.

> Learn more about IEEE
> IEEE Mission & Vision

Mission & Vision

IEEE is the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity. Below, you can find IEEE's mission and vision statements.

Mission statement

IEEE's core purpose is to foster technological innovation and excellence for the benefit of humanity.

Vision statement

IEEE will be essential to the global technical community and to technical professionals everywhere, and be universally recognized for the contributions of technology and of technical professionals in improving global conditions.

> View the IEEE Strategic plan

Figure 10.1: Main content of IEEE's 'mission and vision' web page extracted with the site-level content extraction tool

e.g., [118, 46, 117, 24, 106, 51]). Even a web cleaning competition [20], called CleanEval, was proposed. This competition included a collection of examples to be analyzed, prepared for content extraction and boilerplate removal, and their respective gold standard.

As described in Chapter 2, there are three main approaches in the area of block detection, namely, (i) using the textual information provided by the web page, (ii) based on the rendered image of the web page, and (iii) analyzing the DOM tree of the web page.

Several techniques analyze a rendered image of the web page on the browser. In [22], authors propose a technique based on the assumption that the main content of a web page is usually visible without scrolling and located in the central part of the web page. The authors of [61] analyzed this kind of technique and concluded that they are not widely used because rendering web pages for classification is computationally expensive.

Other techniques use the textual information of the website by analyzing its HTML code. For instance, the technique proposed in [24] extracts structural properties and visual presentation information from the web pages. Then the extracted information is used to train a machine-learning model that classifies the DOM nodes in main content or noise. The authors of [46] propose an algorithm called *content code blurring* to decide whether a token (word or tag) is part of the main content or code. The technique described in [117] compute a ratio called *CETR* for all the lines of the HTML code. The group of lines where the higher CETR is concentrated is the main content of the web page. Nevertheless, the code distribution between the lines of a web page is not always what the user expects. HTML code can be written in many ways, so it can be completely unbalanced (i.e., without tabulations, spaces or even carriage returns), mostly when it is automatically generated. A typical example of this is the source code of the main Google's web page. At the moment of writing these lines, the whole code of the web page takes up only a few lines, and it does not have any legible structure. It should be noted that CETR is useless on this kind of web pages. Li et al. [68] proposed an algorithm called NBCE that uses the HTML source code to build a tree structure (different from the DOM tree). It performs the extraction of triples from the HTML, which are used to construct a graph based on neo4j database. Finally, an algorithm decides whether a node is the main content node or not.

Some approaches, based on DOM trees, develop multiple DOM node features utilizing the DOM tree node properties. Moong [80] presented a site-level content extraction algorithm that matches DOM trees to classify which nodes are contents and which are noises and, after classification, they are clustered into their group respectively. Finally, the algorithm only extracts the content group from the web page. Yang et al. [119] combine page-level with the site-level knowledge to extract structured data from web forum websites. Authors use Markov logic networks (MLNs) to integrate all useful evidences by learning their importance automatically. In [118], authors propose to compute several DOM node features to train a machine learning model that obtains the node or nodes that contain the main content. There are also techniques based on DOM trees that are based on computing a node ratio. For instance, [51] computes a ratio called WLR for all DOM nodes in the web page. This ratio is based on dividing the number of words in the DOM node between its number of leaves. The subtree whose root is the best node is selected as the main content of the web page. In [106], authors define a DOM node ratio called *Text Density* which is computed by dividing the number of chars under a node between the number of tags under it. Then, a threshold is used to infer whether a DOM node is part of the main content or noise. Saravanan and Bama [100] developed a 3-phase method for content extraction. The first 2 phases remove the primary and secondary "noises", while the third phase extracts the main content using a weighted block score mechanism. Gong et al. [45] developed a text extraction technique that combines a site-level noise reduction based on hashtree with a page-level noise reduction based on linked clusters. This combination eliminates noise in web articles.

In recent years, most of the techniques are based on *Machine Learning*. Vogels et al. [115] use a convolutional neural networks-based technique to classify all text blocks in an HTML page as either main content or boilerplate. Leonhardt et al. [66] propose a neural sequence labelling method for boilerplate removal that only takes as input the HTML tags and words. Morbieu et al. developed an unsupervised learning method divided in three stages that extracts the textual main content of a web page. First, it clusters text blocks. Then, it selects the clusters associated with the main content, and finally it performs a classification phase whose input is the labeled data from the two previous steps. Yu et al. [121] propose a web page text extraction algorithm based on multi-feature fusion. The algorithm establishes a small neural network that takes several features of DOM nodes as input, and then predicts whether the nodes contain text information.

10.2 Main content extraction

The proposed content extraction technique (site-level ConEx) takes as input an arbitrary web page (the key page) and outputs a set of DOM nodes that correspond to the main content. The technique is site-level, that is, it loads and analyzes several web pages from the same website to infer the main content. As with the rest of the techniques in this thesis, the technique works at the level of DOM, and because of the DOM tree properties, the main content of a web page can be identified with one or more DOM nodes.

The technique consists of a three-step approach:

i. A set of web pages from the same website of the key page is selected.

ii. An algorithm explores each web page in the set and computes a mapping between its DOM nodes and the DOM nodes of the key page. When it finds that a DOM node of the key page is repeated in any web page of the set, it updates a counter that reflects the number of times each node is repeated in other web pages.

iii. A set of *candidate nodes* is built with all the DOM nodes in the key page that do not appear on other web pages. Then, this set of nodes is reduced in the following way:

    - Those DOM nodes that do not have ancestors in the set are removed from it. The remaining nodes form the *reduced set of candidate nodes.*
    - At this step, if the *reduced set of candidate nodes* only contains one DOM node, the main content of the web page is formed by that DOM node and all its descendants. Nevertheless, if the set includes more than one node:

– An algorithm analyzes each candidate node in order to infer the branch of the DOM tree that contains the main content with a higher probability.
– Then, the algorithm selects all the DOM nodes that belong to both, the set of candidate nodes and the main content branch.

The three phases above are detailed in the following sections.

10.2.1 The web page's main content

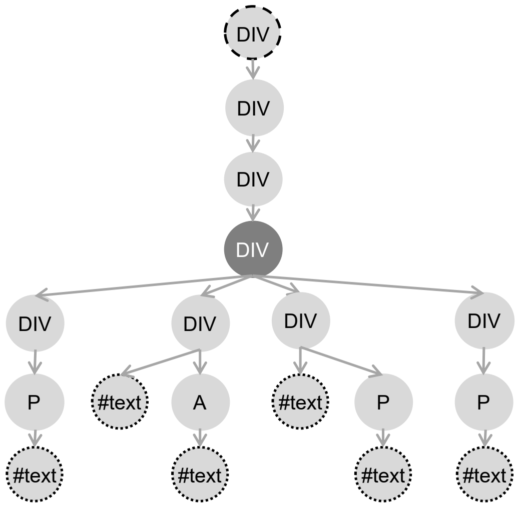


Figure 10.2: Main content DOM nodes example

It is often trivial for humans to identify the main content given a rendered image of a web page. However, when a web page is represented as a DOM tree, usually we can identify several nodes whose subtree includes the main content. Frequently, DOM nodes in a DOM tree form a complex hierarchy and, one node and some of its ancestors contain the same text and/or other content. So, which node should be chosen as the main content? This question can be answered according to a design policy.

On a web page, the "interesting" is clearly subjective. Unfortunately, even if we know that the main content can be defined as the information that can be found on a web page excluding template data, side information like comments or advertisements, and metadata like publication date, the main content may be also subjective. A clear example happens in a web page that displays a news article: the comments of the readers are considered main content by some people (thus, they should be extracted together with the new), while others consider that this part does not belong to the new (and thus they should not be extracted). Therefore, providing a definition of main content is controversial.

An objective definition of the main content, independent of any detection method, can be provided based on the structure of the web page if we assume that a labelling exists indicating the relevant and irrelevant content of the web page. For a web page $P = (N, A)$, it can be assumed the existence of the following labelling:

- $relevant(n)$, which identifies those leaf DOM nodes that should belong to the main content of the web page.
- $irrelevant(n)$, which identifies the rest of the leaf DOM nodes, because they do not belong to the main content.

Some techniques (e.g., [51, 17]) represent the main content of the webpage as a single DOM node whose subtree is the smallest containing all the relevant nodes in a web page. However, this definition can be inaccurate because sometimes the relevant nodes are mixed with irrelevant nodes and boilerplate elements. For instance, consider a DOM subtree whose root node has three children nodes. The first and the third children contain relevant content (i.e. some paragraphs from a blog post), and the second one contains irrelevant content (i.e. a block of advertisements). Hence, the main content of a web page can be defined as a set of DOM nodes where the union of their subtrees contain all the relevant content, and they do not contain irrelevant content.

The formal definition of the main content (see Definition 4.3.3) is based on the definition of a web page (see Definition 4.1.1) and the definition of a website (see Definition 4.1.7). This definition is based on the following assumptions:

- All relevant nodes belong to the subtrees of the main content nodes.
- All nodes that belong to the subtrees of the main content nodes are relevant.
- The set of main content nodes is minimal.

Mainly, the main content is the minimal set of DOM nodes that contain all the relevant nodes.

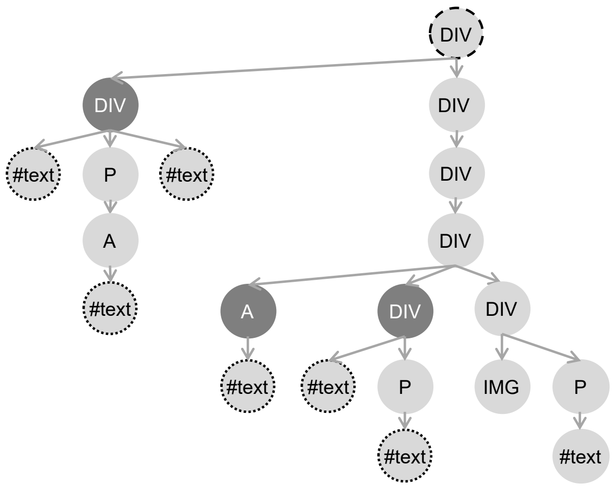


Figure 10.3: Main content with irrelevant DOM nodes example

Example 10.2.1 *Consider the DOM tree in Figure 10.2. The relevant nodes are the leaf nodes represented with a dotted border. According to definition 4.3.3, the main content is the "DIV" node with a dark background because it represents the smallest set of nodes that contains all the relevant nodes. We should not select one of its ancestors as the main content node, e.g, the "DIV" node with a dashed shape because in such a case we break the third condition: the set of main content nodes would not be minimal.*

Example 10.2.2 *Consider the DOM tree in Figure 10.3. The relevant nodes are the leaf nodes represented with a dotted border. Contrary to example 10.2.1 where all the leaves are relevant nodes, in this case, not all the text nodes are relevant. According to definition 4.3.3, the main content is the union of nodes with a dark background because it represents the smallest set of nodes that contains all the relevant nodes.*

Definition 4.3.3 can be used to unify criteria in which DOM nodes should be selected when providing the *relevant* labelling; for instance, when building a suite of benchmarks (see Chapter 13). However, in automatic content extraction tasks, the *relevant* labelling is not available, therefore it must be approximated. The following subsections provide a technique to automatically identify the relevant nodes.

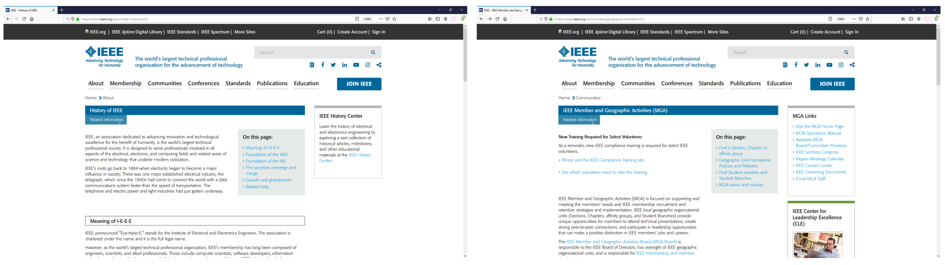
10.2.2 Set of web pages selection

Similarly to the template detection technique presented in Chapter 9, a set of web pages from the same website of the key page that share their template with high probability should be identified. This phase, proposed in [5], is described in Chapter 7 as an independent process. Note that it can be used by any site-level block detection technique.

As stated in Chapter 9, it can be observed that mutually linked webpages from the same domain very likely share their template, and consequently, they very likely share the main content location. This phase identifies a set of mutually linked web pages that implement the same template as the key page. The web pages of this set are pairwise and mutually linked. Therefore, they form an n-complete subdigraph (n-CS), made up of n nodes.

Example 10.2.3 *Figure 10.4 shows two web pages of IEEE's website that implement the same template. The web page on the left can be reached from the menu option "About". The web page on the right can be reached from the menu option "Membership". On both web pages, the main content is located at the bottom-left of the web page. Both web pages share several elements: a header, a menu, a footer, and a right panel with complementary information. As they are mutually linked they form a 2-CS. In the same way, most of the web pages from the same domain linked by the menu located at the top form a complete subdigraph and implement the entire template. The candidates selection technique identifies them as candidate web pages.*

As outlined in previous chapters, other candidate selection techniques use random web pages from the website [114] or take web pages provided by the programmer as input [120]. However, the technique described in Chapter 7 obtains higher quality candidates because the amount of template implemented shared by the candidates is maximized. Therefore, as


IEEE - History of IEEE
https://www.ieee.org/about/ieee-history.html
IEEE.org | IEEE Xplore Digital Library | IEEE Standards | IEEE Spectrum | More Sites
Cart (0) | Create Account | Sign In
IEEE
Advancing Technology for Humanity
The world's largest technical professional organization for the advancement of technology
Search
About
Membership
Communities
Conferences
Standards
Publications
Education
JOIN IEEE
Home > About
History of IEEE
Related information
On this page:
Meaning of I-E-E-E
Foundation of the AIEE
Foundation of the IRE
The societies converge and merge
Growth and globalization
Related links
IEEE History Center
Meaning of I-E-E-E
IEEE - IEEE Member and Geogr
https://www.ieee.org/communities/geographic-activities.html
Home > Communities
IEEE Member and Geographic Activities (MGA)
New Training Required for Select Volunteers
Please visit the IEEE Compliance Training site
See which volunteers need to take the training
Find a Section, Chapter, or affinity group
Geographic Unit Formations Policies and Petitions
Find Student activities and Student Branches
MGA vision and mission
MGA Links
Visit the MGA Home Page
MGA Operations Manual
Available MGA Board/Committee Positions
IEEE Sections Congress
Region Meetings Calendar
IEEE Contact Center
IEEE Governing Documents
Email MGA Staff
IEEE Center for Leadership Excellence (CLE)

IEEE, an association dedicated to advancing innovation and technological excellence for the benefit of humanity, is the world's largest technical professional society. It is designed to serve professionals involved in all aspects of the electrical, electronic, and computing fields and related areas of science and technology that underlie modern civilization.

IEEE's roots go back to 1884 when electricity began to become a major influence in society. There was one major established electrical industry, the telegraph, which since the 1840s had come to connect the world with a data communications system faster than the speed of transportation. The telephone and electric power and light industries had just gotten underway.

Learn the history of electrical and electronics engineering by exploring a vast collection of historical articles, milestones, and other educational materials at the IEEE History Center.

IEEE, pronounced "Eye-triple-E," stands for the Institute of Electrical and Electronics Engineers. The association is chartered under this name and it is the full legal name.

However, as the world's largest technical professional organization, IEEE's membership has long been composed of engineers, scientists, and allied professionals. These include computer scientists, software developers, information

As a reminder, new IEEE compliance training is required for select IEEE volunteers.

IEEE Member and Geographic Activities (MGA) is focused on supporting and meeting the members' needs and IEEE membership recruitment and retention strategies and implementation. IEEE local geographic organizational units (Sections, Chapters, affinity groups, and Student Branches) provide unique opportunities for members to attend technical presentations, create strong peer-to-peer connections, and participate in leadership opportunities that can make a positive distinction in IEEE members' jobs and careers.

The IEEE Member and Geographic Activities Board (MGA Board) is responsible to the IEEE Board of Directors, has oversight of IEEE geographic organizational units, and is responsible for IEEE membership and member

Figure 10.4: Web pages of IEEE's website sharing the template

the amount of template is maximized, the main content can be identified more accurately.

Roughly, the technique described in Chapter 7 analyzes the hyperlinks in the key page in order to select those that with high probability produce a complete subdigraph. An order of relevance is established by the hyperlink analysis by using two algorithms: hyperlink distance and DOM distance. Finally, the order of relevance is iteratively explored until the web pages pointed by the links in it form a n-CS.

10.2.3 Web pages mapping

The identification of the DOM nodes of the key page repeated in other web pages of the n-CS is done using the equal top-down mapping algorithm (ETDM) defined in 8.2.1. This mapping compares two DOM trees establishing a correspondence between their nodes (see Figure 8.1).

As stated in Section 8.2.1, the definition of ETDM is parametric with respect to the equality relation $\triangleq$. This notion of equality is more general than the standard equality (=). It makes the comparison of two nodes by considering their HTML *tagName*, HTML attributes, class names, number of children, and their relative position in the DOM tree.

Once the *set of web pages* (n-CS) is built, an ETDM is computed between each web page in the n-CS and the key page. The algorithm compares the DOM trees by traversing them top-down. It starts at the root and begins mapping each node of the key page with the nodes located at the same depth in the other web pages. The algorithm maintains an attribute called *occurrences* that acts as a counter on each node of the key page. That attribute stores the number of times a node from the key page is found in the web pages of the n-CS. When two nodes can be mapped, they are equal ($n_1 \triangleq n_2$), and therefore the occurrences attribute of the node in the key page is updated by incrementing it in one unit. Then, the algorithm recursively continues trying to map the children of both mapped nodes. It should be noted that when it is not possible to map a node, the algorithm does not explore its descendants.

Algorithm 11 takes as input a key page and a set of web pages from the same website (n-CS), and it outputs the same key page including the value of the *occurrences* attribute for each node. This attribute contains the number of times a node from the key page is found in the web pages of the n-CS. Then, nodes that belong to the template of the key page are identified using this attribute in the following way: those nodes whose *occurrences* attribute is higher than zero (because they appear in other web pages of

Algorithm 11 Compute the number of occurrences of each node in the key page

```
Input: A key page $p_k = (N_1, A_1)$ and a set of $n$ web pages $P$.
Output: The key page $p_k$ equipped with a variable occurrences for each node.
Initialization: $\forall n \in N_1 \ . \ n.occurrences = 0$.

begin
  $r_1 = root(p_k)$;
  foreach ($p = (N_2, A_2)$ in $P$)
    $r_2 = root(p)$;
    if ($r_1 \triangleq r_2$)
      $r_1.occurrences = r_1.occurrences + 1$;
      $assignOccurrences(r_1, r_2)$;
  return $p_k$;
end

procedure $assignOccurrences$(node $r_1 \in N_1$, node $r_2 \in N_2$)
  foreach ($n_1 \in N_1, n_2 \in N_2 \ . \ n_1 \triangleq n_2, (r_1, n_1) \in A_1$ and $(r_2, n_2) \in A_2$)
    $n_1.occurrences = n_1.occurrences + 1$;
    $assignOccurrences(n_1, n_2)$;
end procedure
```

the website) will probably belong to the template of the web page. On the other hand, the nodes with *occurrences* $= 0$ will high probability form the main content of the web page.

When it finishes, the algorithm returns a set of *candidate nodes* as output. For instance, the grey nodes in Figure 10.5 are the candidate nodes. As the candidate nodes only appear on the key page and do not appear in other web pages of the n-CS, therefore all of them or just a subset corresponds to the main content.

10.2.4 Candidate set reduction

The set of candidate nodes obtained in the previous subsection contains DOM nodes that are ancestors of other nodes from the set. This is due to the fact that when a node in the key page does not belong to any web page from the n-CS, its ancestors neither belong to any web page from the n-CS, so all of them are included in the set of candidate nodes. As a result,

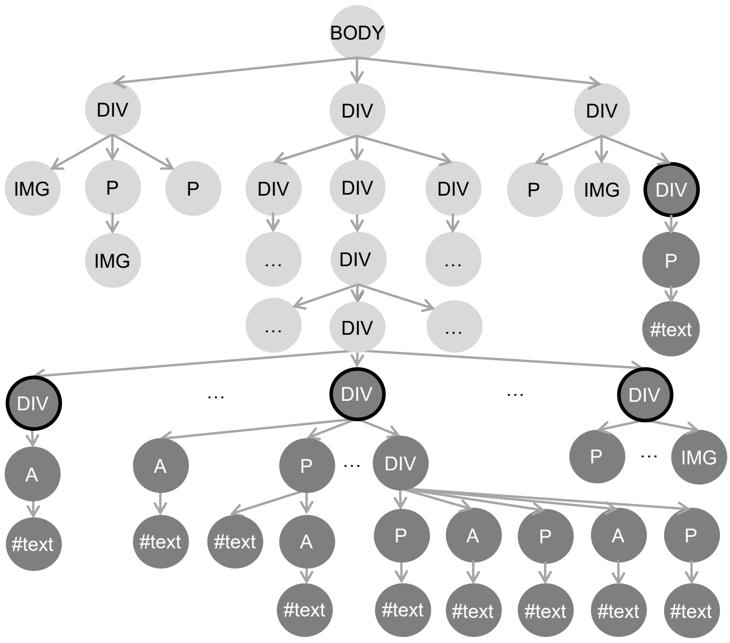


Figure 10.5: Set of candidate nodes

a set of candidate nodes can be represented using the roots of the different subtrees of candidate nodes. It can be observed in Figure 10.5, where all the nodes from the set of candidate nodes are represented with a grey background, that the four subtrees of candidate nodes can be simplified with the four nodes with a bold border. Then, the resulting set of nodes, which computation is based on Theorem 10.2.4, is called the *reduced set of candidate nodes.*

Theorem 10.2.4 (parent-child relation of candidate nodes) *Let $P = (N, A)$ be a web page and let $C \subseteq N$ be the set of all candidate nodes of P. Then,*

$$n \in C \implies \forall n', (n, n') \in A^* \; . \; n' \in C$$

Proof. First, if $|descendants(n)| = 0$ the claim follows trivially. We prove the case when $|descendants(n)| \geq 0$ by contradiction. We assume that $n \in C \wedge \exists n', (n, n') \in A^* \; . \; n' \notin C$. Because $n' \notin C$, there must exist a web page P' with an ETDM mapping M and $(n', n'') \in M$ (for some n''). Moreover, according to the top-down property of Definition 4.2.7, all ancestors of n' also belong to the ETDM mapping M, and therefore, all ancestors of n' (including n) are not candidates: $n \notin C$. But this is a contradiction with the premise $n \in C$. $\square$

Algorithm 12 computes the *reduced set of candidate nodes* based on Theorem 10.2.4. The algorithm gets a set of candidate nodes as input, and, for each node, it examines its ancestors. Then, the algorithm adds to the *reduced set of candidate nodes* the ancestors with lower depth on the DOM tree that belong to the set of candidate nodes. Figure 10.6 shows the result of applying this algorithm to the example in Figure 10.5. The dark gray nodes are the root nodes of all the nodes in the set of candidates. These root nodes must belong to the original set of candidates.

When the reduced set of candidates is only formed by one node, that node corresponds to the main content of the web page. In this case, the algorithm finishes and the following phases are not executed. The node is returned as the main content.

Algorithm 12 Candidate set reduction

```
Input: A set of DOM nodes $candidatesSet$
Output: A reduced set of DOM nodes $reducedSet$
        that includes only the nodes at a higher level.
Initialization: $reducedSet = \{\}$.

begin
  foreach ($node$ in $candidatesSet$)
    $candidate = node$;
    while ($parent(candidate) \in candidatesSet$)
      $candidate = parent(candidate)$;
      $candidatesSet = candidatesSet \setminus$
                  $subtree(candidate)$;
    $reducedSet = reducedSet \cup candidate$;
  return $reducedSet$;
end
```

10.2.5 Main content branch detection

In the case that the *reduced set of candidate nodes* contains more than one node, the DOM tree should be analyzed in order to remove those nodes not belonging to the main content.

For each node in the *reduced set of candidate nodes*, an algorithm analyzes its parent and counts its number of descendants, storing the obtained

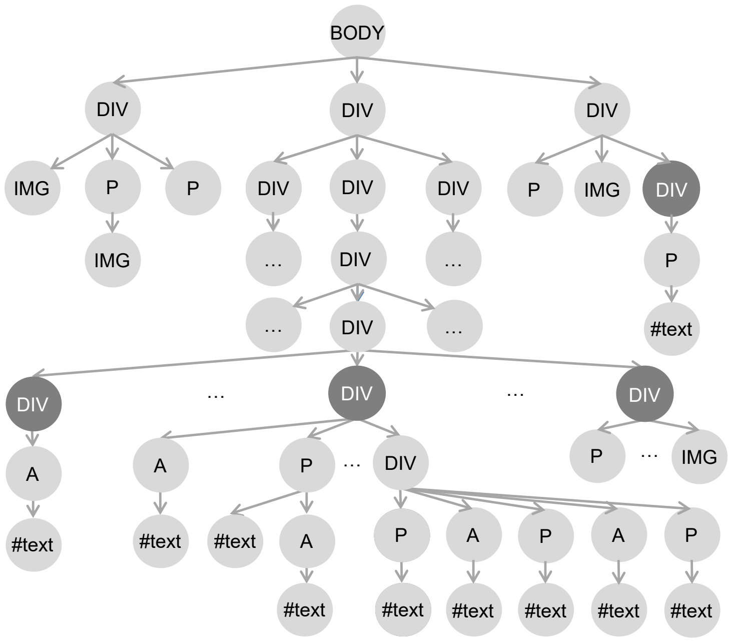


Figure 10.6: Candidate set reduction

value. Then, the node with more descendants is set as the root node of the branch of the DOM tree that corresponds to the main content. It is very difficult (but possible) to obtain a draw. If it happens, the algorithm selects as the root node of the main content branch the first one in a deep first traversal. The reason for this is that the first node in a deep traversal appears on the screen without scrolling with a higher probability.

This phase explores and selects the parent of each DOM node in the *reduced set of candidate nodes* because they are also found on other web pages, for that reason they probably belong to the template of the website or at least they are located in the boundary between the main content and the template. It should be noted that Theorem 10.2.4 states that all the descendants of the selected parent nodes are candidate nodes and thus, they can not be found on other web pages.

The process of selecting the root node corresponding to the main content branch of a web page is performed by Algorithm 13. The root node of the main content branch indicates the subtree (branch) of the DOM tree containing the main content. Hence, all nodes located outside the main content branch have to be discarded, as described in the following subsection. Often, the selected main content branch includes several candidates, and therefore they have to be processed in order to obtain the set of main content nodes, as detailed below.

Algorithm 13 Main content branch detection

```
Input: A key page $p_k$, and a set $reducedSet$
       of DOM nodes in $p_k$
Output: A DOM node $branch$.

begin
    $count = 0$;
    foreach ($n$ in $reducedSet$)
        $node = parent(n)$;
        if $|subtree(node)| > count$
            $branch = node$;
            $count = |subtree(node)|$;
    return $branch$;
end
```

Example 10.2.5 *In Figure 10.7, the "DIV" with a dashed shape is the root node of the main content branch. This "DIV" node has 3 children that belong to the* reduced set of candidate nodes.

10.2.6 Discarding candidates

Once the algorithm founds the branch containing the main content, it has to remove the nodes not belonging to it from the *reduced set of candidate nodes.* The process is simple, an algorithm checks whether the main content branch contains each node in the *reduced set of candidate nodes.* The nodes that do not belong to the main content branch are removed from the set.

Algorithm 14 explores the *reduced set of candidate nodes* and removes from it the nodes that do not belong to the main content branch.

Example 10.2.6 *In Figure 10.8, Algorithm 14 discards the grey node at the top-right of the tree (and removes it from the* reduced set of candidate nodes*) because it is not a descendant of the branch node.*

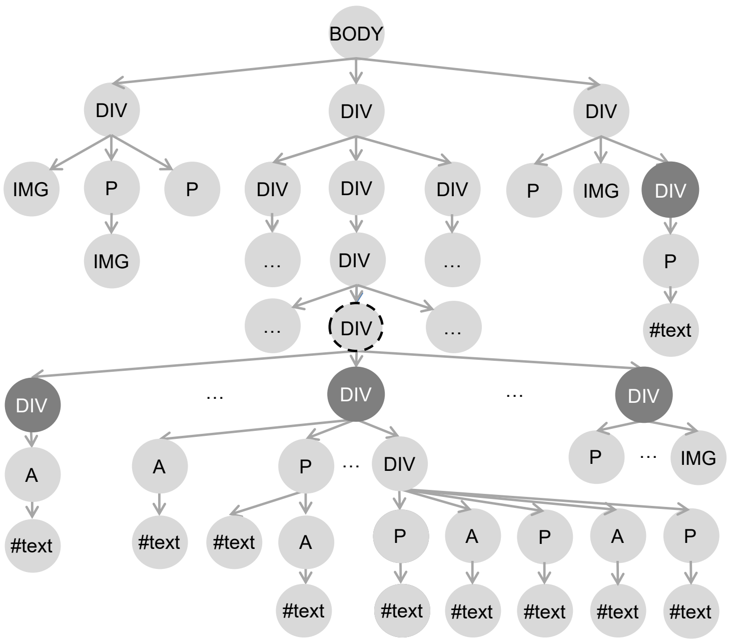


Figure 10.7: Main content branch detection

Algorithm 14 Candidate set reduction

```
Input: A set of DOM nodes reducedSet, and
       the branch node branch
Output: A set of DOM nodes finalReducedSet
        that does not include nodes not
        belonging to the main content branch

begin
  foreach (node in reducedSet)
    if ($branch \notin ancestors(node)$)
      $reducedSet = reducedSet \setminus \{node\}$
  return reducedSet;
end
```

10.2.7 Main content selection

Once Algorithm 14 has discarded the candidate nodes not belonging to the main content branch, the remaining candidate nodes are considered main content. Nevertheless, often these nodes can be grouped in the following

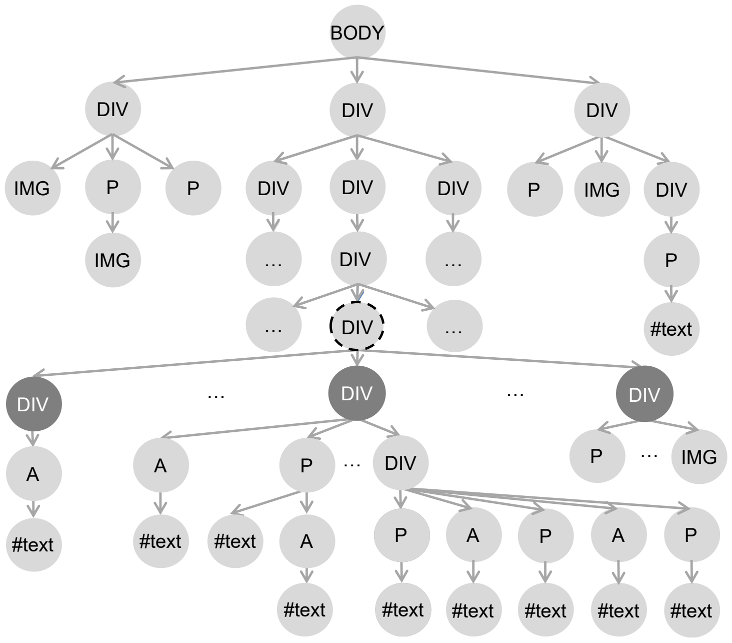


Figure 10.8: Discarding candidates

way: if two or more candidates are sibling nodes (as the "DIV" nodes with gray background in Figure 10.8), they are replaced recursively by their parent. Algorithm 15 computes this process.

Algorithm 15 Main content selection

```
Input: A set of DOM nodes $reducedSet$
Output: A set of DOM nodes $mainContent$.
begin
    $mainContent = reducedSet$;
    foreach $(n_1, n_2$ in $mainContent$ with
    $parent(n_1) == parent(n_2))$
        $mainContent = (mainContent \setminus \{n_1, n_2\})$
        $\cup\{parent(n_1)\}$
    return $mainContent$;
end
```

Example 10.2.7 *In Figure 10.8, after the removal of the grey node located at the top right, the* reduced set of candidates *is formed from only*

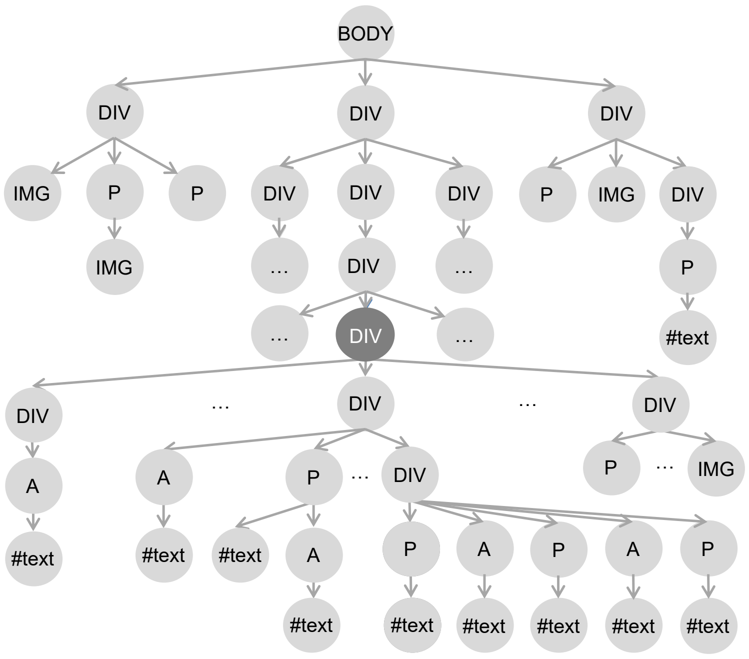


Figure 10.9: Discarding candidates

three nodes: the three grey background sibling nodes. In Figure 10.9, based on Algorithm 15, the “DIV” node with grey background is the final main content of the web page, because it replaces its three children.

10.3 Implementation

This technique, as the rest of the techniques described in this thesis, has been implemented as a WebExtension compatible with Mozilla-based and Chromium-based browsers. When users browsing on the Internet want to extract the main content of the web page loaded in the browser, they press on the “Extract Content” button and the add-on automatically does the required actions and shows the main content of the web page. Therefore, the nodes that do not belong to the main content of the web page are hidden.

10.3.1 Empirical evaluation

Section 8.3 estimates the value of the open parameters relative to the CS and the equality relation $\triangleq$. These parameters were the size of the CS, the number of votes, the weight of the properties of the equality relation $\triangleq$, and the threshold of the value of the equality relation $\triangleq$).

Moreover, Section 8.3 also determined the values of the parameter n as the optimal size of the CS and the parameter v as the number of votes needed to consider a node as part of the template should be computed. Since this content extraction technique, as described in Section 10.2, states that a node belongs to the set of candidate nodes if it is not repeated in several web pages, the parameter v should be equal to zero. Therefore, the only parameter that needs to be computed is n (the optimal size of the CS), because it may occur that a value is optimal for template detection but not for content extraction, thus it has to be determined for content extraction.

It should be underlined that, as stated in Chapter 7, it is important to determine the optimal size of n because the larger the CS is, the more time is needed to compute the CS and to execute the ETDM. Moreover, experiments taken in Section 7.3.1 of Chapter 7 reveal that larger CS sizes do not always obtain better template detection results, so maybe the same could happen in content extraction.

Therefore, the size of the set of web pages (the n value of the n-CS) needed by Algorithm 11 had to be determined. The parameter n (the optimum size of the CS) was determined by measuring the recall, precision and F1 of the retrieved text words and DOM nodes for different CS sizes.

Table 10.1 shows the obtained results of the performed evaluation experiments, with an n-CS size from 1 to 8, and with the training subset of TeCo benchmark suite (see Chapter 13). Each row is the average of repeating all the experiments for the 105 benchmarks with a different value for n in the n-CS. Column `Size` represents the size of the n-CS. Moreover, the table shows, for the retrieved DOM nodes and the retrieved text words, the average `Recall`, `Precision`, and `F1`. The last column shows the `Runtime` in milliseconds needed to obtain the main content.

We can observe that a set of web pages of size 4 (4-CS) is the best option because it obtains the best F1 value in retrieved words (91.71%), while it obtains one of the best results in retrieved DOM nodes (84.52%). Table 10.1 shows that sets containing 1 web page (1-CS) obtain the lowest F1 values (around 75%), and sets of web pages containing 5 or more web pages obtain similar F1 values. In addition, as stated in Chapter 7, the size of the set directly affects the performance. The greater the set size,

	DOM nodes			Words				
Size	Recall	Precision	F1	Recall	Precision	F1	Loads	Runtime
1	90.02 %	74.46 %	75.17 %	92.39 %	83.28 %	84.82 %	1	303 ms.
2	94.54 %	78.36 %	81.99 %	96.50 %	84.53 %	88.73 %	7.86	654 ms.
3	92.80 %	84.00 %	83.50 %	96.17 %	88.52 %	89.84 %	11.63	874 ms.
4	95.33 %	82.16 %	84.52 %	98.50 %	88.32 %	91.71 %	17.11	1213 ms.
5	94.68 %	82.50 %	84.48 %	97.95 %	88.47 %	91.53 %	25.74	1738 ms.
6	94.56 %	83.23 %	84.50 %	97.86 %	88.72 %	91.56 %	27.66	2545 ms.
7	94.56 %	83.34 %	84.55 %	97.86 %	88.76 %	91.58 %	29.11	2805 ms.
8	94.56 %	83.34 %	84.55 %	97.86 %	88.76 %	91.58 %	31.71	3280 ms.

Table 10.1: Determining the optimal size of the n-CS

the more web pages must be loaded, and more ETDM mappings must be calculated. This is also a compelling reason to select the 4-CS.

The parameters relative to the equality relation $\triangleq$ are used by the ETDM algorithms to discern whether two nodes that belong to two different web pages are the same node or not, so that if $n_1 \triangleq n_2 \geq t$ both nodes are equal. Section 8.3.1 of Chapter 8 computes the weight of each property of the equality relation $\triangleq$ for the site-level template detection algorithm (TemEx) with a complete subdigraph of size 3. In order to obtain the optimal results, we computed again the parameters for the content extraction algorithm described in this chapter (site-level ConEx). As in Section 8.3.1 of Chapter 8, the weight of each property of the equality relation $\triangleq$ was computed ($\triangleq = A * \textit{Node position} + B * \textit{Node class names} + C * \textit{Node children} + D * \textit{HTML attributes}$), where $A + B + C + D = 1$). Therefore, all the experiments were repeated with the following possible values for the weightings used:

Node position:	$[0.00 - 1.00]$ in steps of 0.1.
Node class names:	$[0.00 - 1.00]$ in steps of 0.1.
Node children:	$[0.00 - 1.00]$ in steps of 0.1.
HTML attributes:	$[0.00 - 1.00]$ in steps of 0.1.

Moreover, the threshold of the equality relation $\triangleq$ t was also evaluated for each possible weighting with the following values:

$\triangleq$ *threshold (t):* $[0.10 - 1.00]$ in steps of 0.10.

Finally, it should be noted that the computation of the *Node position* value can be done in four ways. They are described in Section 8.3.1 of Chapter 8.

We also want to highlight that, as in Chapter 8, the attribute HTML *tagName* has not been included in the list because to map two nodes it is mandatory that both share the same HTML tag name. Table 10.2 shows the best 20 computed combinations after evaluating all possible combinations against the 105 benchmarks training subset of the TeCo benchmark suite (see Chapter 13). Specifically, it summarizes the repetition of the experiments with all the possible combinations of the properties that form the equality relation $\triangleq$ (first 4 columns of Table 10.2), the threshold $\triangleq$ (t) (fifth column of the table), and the *Node position* computation method (sixth column of the table). Columns `Recall`, `Precision`, and `F1` are the recall, precision, and F1 respectively for both, retrieved DOM nodes and retrieved words. Finally, column `Runtime` indicates the average runtime of the algorithm for this combination of parameters. It shoud be noted that, each row in the table corresponds to the average of 105 template extractions from 105 different web pages.

To build the table, all the necessary combinations of the parameters involved the computation of 1201200 experiments, which were performed with a total computing time of approximately 57 days using an Intel i9 9900k.

						DOM nodes			Words			
`Class.`	`Pos.`	`Attr.`	`Child.`	$\triangleq$ `thres.`	`Opt.`	`Recall`	`Precision`	`F1`	`Recall`	`Precision`	`F1`	`Runtime`
0.1	0.0	0.2	0.7	0.9	4	89.92 %	76.80 %	79.04 %	92.34 %	85.31 %	87.14 %	2417 ms.
0.1	0.0	0.2	0.7	0.9	3	89.92 %	76.80 %	79.04 %	92.34 %	85.31 %	87.14 %	5076 ms.
0.1	0.0	0.2	0.7	0.9	2	89.92 %	76.80 %	79.04 %	92.34 %	85.31 %	87.14 %	5765 ms.
0.2	0.0	0.1	0.7	0.9	4	89.01 %	76.38 %	78.82 %	91.09 %	84.67 %	86.28 %	2195 ms.
0.2	0.0	0.1	0.7	0.9	3	89.01 %	76.38 %	78.82 %	91.09 %	84.67 %	86.28 %	4752 ms.
0.2	0.0	0.1	0.7	0.9	2	89.01 %	76.38 %	78.82 %	91.09 %	84.67 %	86.28 %	5712 ms.
0.0	0.0	0.5	0.5	0.8	4	89.52 %	77.58 %	78.74 %	91.45 %	84.87 %	86.23 %	2547 ms.
0.0	0.0	0.5	0.5	0.8	3	89.52 %	77.58 %	78.74 %	91.45 %	84.87 %	86.23 %	5678 ms.
0.0	0.0	0.5	0.5	0.8	2	89.52 %	77.58 %	78.74 %	91.45 %	84.87 %	86.23 %	5991 ms.
0.0	0.0	0.2	0.8	0.9	4	89.13 %	77.23 %	78.70 %	90.88 %	84.58 %	85.90 %	2326 ms.
0.0	0.0	0.2	0.8	0.9	2	89.13 %	77.23 %	78.70 %	90.88 %	84.58 %	85.90 %	4852 ms.
0.0	0.0	0.2	0.8	0.9	3	89.13 %	77.23 %	78.70 %	90.88 %	84.58 %	85.90 %	5352 ms.
0.1	0.1	0.2	0.6	0.8	2	88.80 %	77.57 %	78.64 %	90.65 %	85.50 %	85.66 %	5140 ms.
0.1	0.2	0.2	0.5	0.9	4	89.32 %	76.02 %	78.58 %	91.64 %	84.54 %	86.45 %	2613 ms.
0.2	0.1	0.1	0.6	0.9	4	88.97 %	75.90 %	78.53 %	90.96 %	84.14 %	85.95 %	2763 ms.
0.2	0.1	0.1	0.6	0.9	3	88.97 %	75.90 %	78.53 %	90.96 %	84.14 %	85.95 %	5335 ms.
0.0	0.0	0.4	0.6	0.8	4	88.20 %	76.79 %	78.15 %	90.25 %	84.17 %	85.30 %	2848 ms.
0.0	0.0	0.4	0.6	0.8	2	88.20 %	76.79 %	78.15 %	90.25 %	84.17 %	85.30 %	4806 ms.
0.0	0.0	0.4	0.6	0.8	3	88.20 %	76.79 %	78.15 %	90.25 %	84.17 %	85.30 %	5514 ms.
0.1	0.0	0.1	0.8	0.9	4	88.15 %	76.82 %	78.07 %	90.06 %	85.10 %	85.41 %	2409 ms.

Table 10.2: Determining the best values of the equality relation $\triangleq$ properties for site-level ConEx

The first row in the table was selected as the optimum combination of parameters for the equality relation $\triangleq$ because it is the fastest combination that obtains the best F1 metric. Note that, for most combinations of the properties that form the equality relation $\triangleq$ and the $\triangleq$ *threshold (t)*, the results obtained for different *Node position* computation methods (column `Option` of the table) are equal. In that case, the rows are ordered by column `Runtime`, placing the faster combinations first.

Finally, the following table summarizes the optimal parameters obtained empirically:

CS size (n):	4.
Node class names:	0.10.
Node position:	0.00.
HTML attributes:	0.20.
Node children:	0.70.
$\triangleq$ *threshold (t):*	0.90.
Option:	4.

Note that these parameters were obtained for content extraction, and they are different from the parameters obtained for template detection (See Chapter 8).

Algorithm evaluation

In order to evaluate the technique, several experiments were performed with the 45 evaluation benchmarks of the TeCo benchmark suite (see Chapter 13). For all the benchmarks in the evaluation subset, we computed the `Recall`, `Precision`, and `F1` of the retrieved DOM nodes and the retrieved words. Additionally, we computed the `Runtime` in milliseconds. The results, that were computed with a 4-CS, are shown in Tables 10.3 and 10.5.

In Table 10.3, column `Total` shows the total number of DOM nodes of the key page; column `Gold` indicates the number of DOM nodes of the gold standard; column `Retr.` is the number of retrieved DOM nodes; column `Correct` represents the number of retrieved DOM nodes that belong to the gold standard; columns `Rec.`, `Prec.`, and `F1` are the recall, precision, and F1 respectively. In Table 10.5, column `Gold` shows the total number of words on the key page; column `Retr.` is the number of retrieved words; column `Corr.` indicates the number of retrieved words that belong to the gold standard; columns `Rec.`, `Prec.`, and `F1` are the recall, precision, and F1 respectively. In both tables, column `Load` shows the number of web pages loaded by the technique; column `Rt.` represents the runtime used to compute the main content (in milliseconds); and column `Rt. opt.` represents the runtime used to compute the main content implementing the runtime improvement algorithm described subsequently in this section.

It can be observed that the algorithm obtains an average F1 over 80% for retrieved DOM nodes and an average F1 over 90% for retrieved words. In addition, for retrieved words, one third of the benchmarks obtain an F1 of 100%, that is, the algorithm extracts exactly the text that forms the main content. As it is a site-level algorithm, it needs to load on average

	Number of nodes				DOM nodes		
Benchmark	Total	Gold	Retr.	Corr.	Rec.	Prec.	F1
www.jdi.org.za	619	199	207	137	68.84 %	66.18 %	67.49 %
www.premiere-urgence.org	480	32	32	32	100.00 %	100.00 %	100.00 %
www.indiangaming.org	575	148	375	148	100.00 %	39.47 %	56.60 %
hispalinux.es	501	144	138	138	95.83 %	100.00 %	97.87 %
www.gktw.org	767	130	86	86	66.15 %	100.00 %	79.63 %
www.apnic.net	598	79	69	69	87.34 %	100.00 %	93.24 %
www.unicef.org	1037	381	386	381	100.00 %	98.70 %	99.35 %
www.klimabuendnis.org	851	134	352	134	100.00 %	38.07 %	55.14 %
www.isoc-es.org	259	56	100	56	100.00 %	56.00 %	71.79 %
biztechmagazine.com	1892	454	235	29	6.39 %	12.34 %	8.42 %
www.eeo.com.cn	834	119	119	119	100.00 %	100.00 %	100.00 %
www.wishtv.com	2167	343	356	343	100.00 %	96.35 %	98.14 %
news.mit.edu	2117	128	355	128	100.00 %	36.06 %	53.00 %
asia.nikkei.com	869	116	206	116	100.00 %	56.31 %	72.05 %
www.rcnky.com	1738	112	103	103	91.96 %	100.00 %	95.81 %
news.discovery.com	2826	791	791	791	100.00 %	100.00 %	100.00 %
www.kathimerini.gr	1825	117	562	117	100.00 %	20.82 %	34.46 %
news.un.org	1726	59	57	57	96.61 %	100.00 %	98.28 %
frances.forosactivos.net	785	495	632	495	100.00 %	78.32 %	87.84 %
www.wysiwygwebbuilder.com	3936	3201	3686	3201	100.00 %	86.84 %	92.96 %
www.3dprintforums.com	1040	748	754	748	100.00 %	99.20 %	99.60 %
www.strangehorizons.com	631	403	436	400	99.26 %	91.74 %	95.35 %
communities.apple.com	3136	1306	81	0	0.00 %	0.00 %	0.00 %
www.sloweurope.com	4193	2789	3684	2786	99.89 %	75.62 %	86.08 %
community.ricksteves.com	2057	1177	1858	1177	100.00 %	63.35 %	77.56 %
hackercombat.com	1711	698	707	698	100.00 %	98.73 %	99.36 %
www.scbwi.org	876	506	662	506	100.00 %	76.44 %	86.64 %
johngardnerathome.info	395	188	353	188	100.00 %	53.26 %	69.50 %
www.annmalaspina.com	392	84	105	68	80.95 %	64.76 %	71.96 %
foodsense.is	330	192	217	184	95.83 %	84.79 %	89.98 %
sites.google.com	372	85	44	41	48.24 %	93.18 %	63.57 %
whatever.scalzi.com	1648	243	225	225	92.59 %	100.00 %	96.15 %
www.javiercelaya.es	740	57	305	57	100.00 %	18.69 %	31.49 %
diarium.usal.es	604	524	513	513	97.90 %	100.00 %	98.94 %
www.jameslovelock.org	653	174	174	174	100.00 %	100.00 %	100.00 %
www.cipri.info	933	556	559	556	100.00 %	99.46 %	99.73 %
naranjascarcaixent.com	290	141	145	141	100.00 %	97.24 %	98.60 %
www.technicalbookstoreonline.com	2959	2002	2453	2002	100.00 %	81.61 %	89.88 %
www.floridarealestatecollege.com	1023	65	502	65	100.00 %	12.95 %	22.93 %
www.basf.com	827	62	62	62	100.00 %	100.00 %	100.00 %
www.mcphersonoil.com	831	225	231	225	100.00 %	97.40 %	98.68 %
www.thirteenhou.com	1217	1073	1080	1073	100.00 %	99.35 %	99.67 %
www.embalajesterra.com	2342	470	470	470	100.00 %	100.00 %	100.00 %
www.crypto.ch	338	68	50	50	73.53 %	100.00 %	84.75 %
www.shopbookshop.com	1727	387	414	387	100.00 %	93.48 %	96.63 %
Average	1281.49	476.91	554.02	432.80	91.14 %	77.48 %	80.43 %

Table 10.3: Evaluation of the precision, recall, F1, and runtime for retrieved DOM nodes

around 12 web pages from the same website to infer the main content. However, in approximately 40% of the benchmarks, the algorithm only loaded

Benchmark	Load	Rt.	Rt. opt.
www.jdi.org.za	9	135 ms.	140 ms.
www.premiere-urgence.org	5	225 ms.	241 ms.
www.indiangaming.org	5	65 ms.	68 ms.
hispalinux.es	5	153 ms.	169 ms.
www.gktw.org	6	351 ms.	324 ms.
www.apnic.net	5	279 ms.	258 ms.
www.unicef.org	6	310 ms.	309 ms.
www.klimabuendnis.org	5	1474 ms.	1611 ms.
www.isoc-es.org	7	98 ms.	94 ms.
biztechmagazine.com	7	700 ms.	723 ms.
www.eeo.com.cn	14	1275 ms.	1185 ms.
www.wishtv.com	7	3474 ms.	3286 ms.
news.mit.edu	5	1491 ms.	1555 ms.
asia.nikkei.com	8	575 ms.	608 ms.
www.rcnky.com	5	829 ms.	798 ms.
news.discovery.com	6	917 ms.	946 ms.
www.kathimerini.gr	14	957 ms.	932 ms.
news.un.org	8	803 ms.	801 ms.
frances.forosactivos.net	39	170 ms.	170 ms.
www.wysiwygwebbuilder.com	143	1757 ms.	1754 ms.
www.3dprintforums.com	7	259 ms.	380 ms.
www.strangehorizons.com	8	291 ms.	281 ms.
communities.apple.com	8	476 ms.	475 ms.
www.sloweurope.com	6	231 ms.	234 ms.
community.ricksteves.com	8	112 ms.	96 ms.
hackercombat.com	17	774 ms.	858 ms.
www.scbwi.org	37	149 ms.	124 ms.
johngardnerathome.info	6	40 ms.	36 ms.
www.annmalaspina.com	5	90 ms.	88 ms.
foodsense.is	5	60 ms.	57 ms.
sites.google.com	5	149 ms.	149 ms.
whatever.scalzi.com	14	106781 ms.	3916 ms.
www.javiercelaya.es	5	160 ms.	164 ms.
diarium.usal.es	5	37 ms.	44 ms.
www.jameslovelock.org	6	1816 ms.	1482 ms.
www.cipri.info	5	242 ms.	249 ms.
naranjascarcaixent.com	6	71 ms.	73 ms.
www.technicalbookstoreonline.com	33	370 ms.	407 ms.
www.floridarealestatecollege.com	5	182 ms.	197 ms.
www.basf.com	5	230 ms.	237 ms.
www.mcphersonoil.com	5	400 ms.	384 ms.
www.thirteenhou.com	5	54 ms.	59 ms.
www.embalajesterra.com	6	770 ms.	885 ms.
www.crypto.ch	5	168 ms.	159 ms.
www.shopbookshop.com	6	4319 ms.	3301 ms.
Average	11.82	2984 ms.	673 ms.

Table 10.4: Evaluation of the precision, recall, F1, and runtime for retrieved DOM nodes (cont.)

5 web pages (the key page and 4 web pages more to form the complete subdigraph). Regarding the runtime, we can observe that for approximately 80% of the benchmarks it is less than 1 second, but there is a benchmark whose runtime is higher than 100 seconds. The average runtime is close

	Number of words			Words		
Benchmark	Gold	Retr.	Corr.	Rec.	Prec.	F1
www.jdi.org.za	313	300	284	90.73 %	94.67 %	92.66 %
www.premiere-urgence.org	134	134	134	100.00 %	100.00 %	100.00 %
www.indiangaming.org	145	294	145	100.00 %	49.32 %	66.06 %
hispalinux.es	514	514	514	100.00 %	100.00 %	100.00 %
www.gktw.org	478	461	461	96.44 %	100.00 %	98.19 %
www.apnic.net	245	242	242	98.78 %	100.00 %	99.38 %
www.unicef.org	120	123	120	100.00 %	97.56 %	98.77 %
www.klimabuendnis.org	258	334	258	100.00 %	77.25 %	87.16 %
www.isoc-es.org	69	97	69	100.00 %	71.13 %	83.13 %
biztechmagazine.com	815	201	121	14.85 %	60.20 %	23.82 %
www.eeo.com.cn	43	43	43	100.00 %	100.00 %	100.00 %
www.wishtv.com	786	786	786	100.00 %	100.00 %	100.00 %
news.mit.edu	1000	1105	1000	100.00 %	90.50 %	95.01 %
asia.nikkei.com	642	701	642	100.00 %	91.58 %	95.61 %
www.rcnky.com	935	935	935	100.00 %	100.00 %	100.00 %
news.discovery.com	767	767	767	100.00 %	100.00 %	100.00 %
www.kathimerini.gr	737	833	737	100.00 %	88.48 %	93.89 %
news.un.org	303	303	303	100.00 %	100.00 %	100.00 %
frances.forosactivos.net	169	256	169	100.00 %	66.02 %	79.53 %
www.wysiwygwebbuilder.com	2115	2343	2115	100.00 %	90.27 %	94.89 %
www.3dprintforums.com	347	347	347	100.00 %	100.00 %	100.00 %
www.strangehorizons.com	3559	3700	3559	100.00 %	96.19 %	98.06 %
communities.apple.com	608	157	0	0.00 %	0.00 %	0.00 %
www.sloweurope.com	804	1041	804	100.00 %	77.23 %	87.15 %
community.ricksteves.com	589	866	589	100.00 %	68.01 %	80.96 %
hackercombat.com	383	384	383	100.00 %	99.74 %	99.87 %
www.scbwi.org	247	343	247	100.00 %	72.01 %	83.73 %
johngardnerathome.info	1375	1469	1375	100.00 %	93.60 %	96.69 %
www.annmalaspina.com	114	152	114	100.00 %	75.00 %	85.71 %
foodsense.is	442	458	407	92.08 %	88.86 %	90.44 %
sites.google.com	54	50	48	88.89 %	96.00 %	92.31 %
whatever.scalzi.com	1151	1141	1141	99.13 %	100.00 %	99.56 %
www.javiercelaya.es	359	402	359	100.00 %	89.30 %	94.35 %
diarium.usal.es	869	863	863	99.31 %	100.00 %	99.65 %
www.jameslovelock.org	1308	1308	1308	100.00 %	100.00 %	100.00 %
www.cipri.info	1281	1282	1281	100.00 %	99.92 %	99.96 %
naranjascarcaixent.com	73	73	73	100.00 %	100.00 %	100.00 %
www.technicalbookstoreonline.com	873	999	873	100.00 %	87.39 %	93.27 %
www.floridarealestatecollege.com	310	728	310	100.00 %	42.58 %	59.73 %
www.basf.com	128	128	128	100.00 %	100.00 %	100.00 %
www.mcphersonoil.com	319	319	319	100.00 %	100.00 %	100.00 %
www.thirteenhou.com	1337	1337	1337	100.00 %	100.00 %	100.00 %
www.embalajesterra.com	111	111	111	100.00 %	100.00 %	100.00 %
www.crypto.ch	185	181	181	97.84 %	100.00 %	98.91 %
www.shopbookshop.com	241	241	241	100.00 %	100.00 %	100.00 %
Average	614.56	641.16	583.18	95.07 %	88.06 %	90.41 %

Table 10.5: Evaluation of the precision, recall, F1, and runtime for retrieved words

Benchmark	Load	Rt.	Rt. opt.
www.jdi.org.za	9	135 ms.	140 ms.
www.premiere-urgence.org	5	225 ms.	241 ms.
www.indiangaming.org	5	65 ms.	68 ms.
hispalinux.es	5	153 ms.	169 ms.
www.gktw.org	6	351 ms.	324 ms.
www.apnic.net	5	279 ms.	258 ms.
www.unicef.org	6	310 ms.	309 ms.
www.klimabuendnis.org	5	1474 ms.	1611 ms.
www.isoc-es.org	7	98 ms.	94 ms.
biztechmagazine.com	7	700 ms.	723 ms.
www.eeo.com.cn	14	1275 ms.	1185 ms.
www.wishtv.com	7	3474 ms.	3286 ms.
news.mit.edu	5	1481 ms.	1555 ms.
asia.nikkei.com	8	575 ms.	608 ms.
www.rcnky.com	5	829 ms.	798 ms.
news.discovery.com	6	917 ms.	946 ms.
www.kathimerini.gr	14	957 ms.	932 ms.
news.un.org	8	803 ms.	801 ms.
frances.forosactivos.net	39	170 ms.	170 ms.
www.wysiwygwebbuilder.com	143	1757 ms.	1754 ms.
www.3dprintforums.com	7	259 ms.	380 ms.
www.strangehorizons.com	8	291 ms.	281 ms.
communities.apple.com	8	476 ms.	475 ms.
www.sloweurope.com	6	231 ms.	234 ms.
community.ricksteves.com	8	112 ms.	96 ms.
hackercombat.com	17	774 ms.	858 ms.
www.scbwi.org	37	149 ms.	124 ms.
johngardnerathome.info	6	40 ms.	36 ms.
www.annmalaspina.com	5	90 ms.	88 ms.
foodsense.is	5	60 ms.	57 ms.
sites.google.com	5	149 ms.	149 ms.
whatever.scalzi.com	14	106781 ms.	3916 ms.
www.javiercelaya.es	5	160 ms.	164 ms.
diarium.usal.es	5	37 ms.	44 ms.
www.jameslovelock.org	6	1816 ms.	1482 ms.
www.cipri.info	5	242 ms.	249 ms.
naranjascarcaixent.com	6	71 ms.	73 ms.
www.technicalbookstoreonline.com	33	370 ms.	407 ms.
www.floridarealestatecollege.com	5	182 ms.	197 ms.
www.basf.com	5	230 ms.	237 ms.
www.mcphersonoil.com	5	400 ms.	384 ms.
www.thirteenhou.com	5	54 ms.	59 ms.
www.embalajesterra.com	6	770 ms.	885 ms.
www.crypto.ch	5	168 ms.	159 ms.
www.shopbookshop.com	6	4319 ms.	3301 ms.
Average	11.82	2984 ms.	673 ms.

Table 10.6: Evaluation of the precision, recall, F1, and runtime for retrieved words (cont.)

to 3 seconds. However, implementing the runtime improvement algorithm described subsequently, the average runtime decreases up to nearly 700 ms. especially due to the reduction of the runtime of one benchmark which is reduced from 106 seconds to 4 seconds.

Other similar techniques that also use heterogeneous websites declare the following results: Shanchan et al. obtain an F1 of 82% [118], Gottron et al. 77% [46], and Insa et al. 74% [51]. However, other techniques are based on evaluating prepared datasets such as Cleaneval [20], MSS [86], L3S-GN1 [61], etc. Even some techniques evaluate RSS feeds, or prepared websites (collections of automatically generated web pages that share the same template). For that reason, they usually obtain high F1 values: Pasternack et al. obtain an F1 value of 95% [86], Qureshi et al. 94% [92], Adam et al. 93% [1], and Zhao et al. 88% [69]. Hence, different techniques should not be compared using different datasets because we may obtain inaccurate conclusions.

Chapter 12 contains different comparisons of different content extraction and template detection techniques using the same benchmark suites. Unfortunately, the suites of benchmarks used to compare the main content extraction techniques in Chapter 12 are only prepared for page-level techniques. However, as both techniques have been evaluated with the TeCo benchmark suite, this site-level content extraction technique can be compared with our page-level content extraction technique described in Chapter 6. We can observe by comparing Tables 10.3, 10.4, 10.5, and 10.6 with Tables 6.3 and 6.4 that the page-level content extraction technique is better for both, retrieved DOM nodes and retrieved words. The differences for the `F1` are not significant, about 1% for retrieved DOM nodes and about 2% for retrieved words. However, the `Recall` is higher in the site-level technique for both, retrieved DOM nodes and retrieved words. Therefore, in scenarios where the extracted main content should be maximized, it is preferable to use the site-level technique. Regarding the `Runtime`, as expected, the page-level technique is faster than the site-level technique, 133 ms. versus 673 ms. in average.

Table 10.7 shows the obtained results grouped by benchmark category (see Chapter 13). We can observe that the `Companies / Shops` category obtains very high F1 values (almost 88% for retrieved DOM nodes), while the `Media / Communications` category obtains the lowest F1 value (about 73% for retrieved DOM nodes). Regarding the optimized runtime, for the `Media / Communication` category the technique is about 4 times slower than for the `Institutions / Associations` category.

	DOM nodes			Words		
Benchmark type	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**F1**
Institutions / Associations	90.91 %	77.60 %	80.12 %	98.44 %	87.77 %	91.71 %
Media / Communication	88.13 %	69.10 %	73.35 %	90.54 %	92.31 %	89.81 %
Forum / Social	88.79 %	74.47 %	80.60 %	88.89 %	74.39 %	80.47 %
Personal websites / Blogs	90.61 %	79.35 %	80.15 %	97.71 %	93.63 %	95.41 %
Companies / Shops	97.06 %	86.89 %	87.90 %	99.76 %	92.22 %	94.66 %

Benchmark type	**Load**	**Rt.**	**Rt. opt.**
Institutions / Associations	5.89	343 ms.	357 ms.
Media / Communication	8.22	1223 ms.	1204 ms.
Forum / Social	30.33	469 ms.	486 ms.
Personal websites / Blogs	6.22	12153 ms.	687 ms.
Companies / Shops	8.44	729 ms.	634 ms.

Table 10.7: Results of the performed experiments grouped by category

Runtime improvement

As in the site-level template detection technique (TemEx) described in Chapter 9, the runtime can be improved by inserting temporary DOM nodes in order to reduce the number of children of some nodes. As explained in Section 9.3.1, the mapping of two nodes when they have a large number of children takes excessive time. When the position property value for two nodes with a large number of children is computed, each child of one node needs to be compared with all the children of the other node. When two children are mapped, the position property value of the remaining children has to be computed, and so on. This involves a significant growth of the runtime for nodes with more than 200 children.

As described in Section 9.3.1, the number of children of some DOM nodes and thus, the mapping runtime, can be reduced by inserting some temporary nodes between the parent and the children. These temporary nodes group the children in a way that the maximum number of children of a node is limited. For instance, in the case of a node with 320 children, we can insert 8 temporary nodes between the parent and the children, so each temporary node will have 40 children. This way, we obtain 8 parents with 40 children each instead of one parent with 320 children. The addition of those temporary DOM nodes does not affect the layout of the web pages.

As in Section 9.3.1, two parameters were defined to measure the gain obtained by applying the runtime improvement algorithm:

- Group: It represents the maximum number of child nodes of the temporary DOM nodes. Namely, it is the maximum size of the groups formed with the child nodes of the original node.

- Childnodes: It establishes the number of children a DOM node should have for applying the runtime improvement algorithm.

It should be noted that, in some cases, the results achieved by the runtime improvement algorithm might differ from the expected results. The reason is that the mapping of two temporary nodes could be different than expected if their parent nodes did not have exactly the same children. Nevertheless, this only happens in a few cases, and it depends on the `Group` and `Childnodes` values.

To test the runtime improvement, we performed several experiments using the 105 training benchmarks of the TeCo benchmark suite (see Chapter 13). As in Section 9.3.1, we computed the `Recall`, `Precision`, `F1`, and `Runtime` of the mapping phase for `Group` values ranging from 25 to 125, and `Childnodes` values ranging from 100 to 350.

Table 10.8 shows the results obtained by different combinations of `Group` and `Childnodes` values. On the one hand, we can observe that the `Runtime` of applying the improvement algorithm is substantially lower in all cases. On the other hand, the table shows that the obtained results for retrieved words are the same for all combinations with a `Childnodes` value higher or equal to 200. Finally, we selected a `Group` value of 100, and a `Childnodes` value of 300, since the `Recall` and `F1` score for both, retrieved DOM nodes and retrieved words, are exactly the same as without runtime improvement, and the `Runtime` is more than two and a half times faster than without this improvement. Note that the `Precision` is 0.001 lower than without the improvement for retrieved DOM nodes, however, this is not significative since the obtained `F1` values are equal. It should be highlighted that other combinations of `Group` and `Childnodes` obtain the same results, but their `Runtime` is higher. Note that the `Recall`, `Precision`, and `F1` values have been expressed with three decimal places because of the similarity of the obtained results for almost all combinations.

Runtime analysis

Figure 10.10, based on Column `Rt. opt.` in Table 10.4, shows the relation between the total time needed to extract the main content and the size of the key page, for the benchmarks in the evaluation subset. We can observe that more than 80% of the benchmarks took less than 1 second. On the other hand, only 3 benchmarks took more than 3 seconds. It should be noted that for around 65% of the benchmarks the runtime took less than half a second. By analyzing Figure 10.10, we can infer that larger (considering the number of DOM nodes) web pages usually take the larger runtime, but sometimes there are large websites whose runtime is really low.

		DOM nodes			Words			
Group	Childnodes	Recall	Precision	F1	Recall	Precision	F1	Runtime
25	100	90.556 %	76.100 %	78.988 %	92.711 %	85.069 %	87.237 %	643 ms.
50	100	89.945 %	76.280 %	78.772 %	92.129 %	84.895 %	86.681 %	679 ms.
75	100	89.945 %	76.784 %	79.119 %	92.129 %	85.301 %	86.949 %	718 ms.
100	100	90.518 %	75.719 %	78.633 %	92.711 %	84.825 %	87.092 %	844 ms.
125	100	89.557 %	75.097 %	77.875 %	91.758 %	84.140 %	86.295 %	897 ms.
25	150	90.556 %	76.066 %	78.916 %	92.711 %	85.070 %	87.236 %	697 ms.
50	150	89.935 %	76.727 %	79.040 %	92.129 %	85.306 %	86.951 %	729 ms.
75	150	89.935 %	76.750 %	79.045 %	92.129 %	85.306 %	86.951 %	767 ms.
100	150	90.558 %	76.801 %	79.515 %	92.711 %	85.306 %	87.370 %	923 ms.
125	150	90.559 %	76.070 %	78.922 %	92.711 %	85.070 %	87.236 %	1011 ms.
25	200	89.919 %	76.797 %	79.034 %	92.337 %	85.306 %	87.139 %	813 ms.
50	200	89.921 %	76.799 %	79.038 %	92.337 %	85.306 %	87.139 %	820 ms.
75	200	89.921 %	76.800 %	79.038 %	92.337 %	85.306 %	87.139 %	864 ms.
100	200	89.922 %	76.800 %	79.039 %	92.337 %	85.306 %	87.139 %	1038 ms.
125	200	89.922 %	76.801 %	79.040 %	92.337 %	85.306 %	87.139 %	1039 ms.
25	250	89.923 %	76.799 %	79.039 %	92.337 %	85.306 %	87.139 %	889 ms.
50	250	89.923 %	76.800 %	79.040 %	92.337 %	85.306 %	87.139 %	869 ms.
75	250	89.923 %	76.801 %	79.040 %	92.337 %	85.306 %	87.139 %	908 ms.
100	250	89.923 %	76.801 %	79.041 %	92.337 %	85.306 %	87.139 %	961 ms.
125	250	89.923 %	76.801 %	79.041 %	92.337 %	85.306 %	87.139 %	1088 ms.
25	300	89.923 %	76.799 %	79.039 %	92.337 %	85.306 %	87.139 %	734 ms.
50	300	89.923 %	76.800 %	79.040 %	92.337 %	85.306 %	87.139 %	743 ms.
75	300	89.923 %	76.801 %	79.040 %	92.337 %	85.306 %	87.139 %	793 ms.
100	**300**	**89.923 %**	**76.801 %**	**79.041 %**	**92.337 %**	**85.306 %**	**87.139 %**	**946 ms.**
125	300	89.923 %	76.801 %	79.041 %	92.337 %	85.306 %	87.139 %	1087 ms.
25	350	89.923 %	76.799 %	79.039 %	92.337 %	85.306 %	87.139 %	1125 ms.
50	350	89.923 %	76.800 %	79.040 %	92.337 %	85.306 %	87.139 %	1120 ms.
75	350	89.923 %	76.801 %	79.040 %	92.337 %	85.306 %	87.139 %	1101 ms.
100	350	89.923 %	76.801 %	79.041 %	92.337 %	85.306 %	87.139 %	1361 ms.
125	350	89.923 %	76.801 %	79.041 %	92.337 %	85.306 %	87.139 %	1496 ms.
`No improvement`		89.923 %	76.802 %	79.041 %	92.337 %	85.306 %	87.139 %	2984 ms.

Table 10.8: Results obtained for different `Group` and `Childnodes` parameters

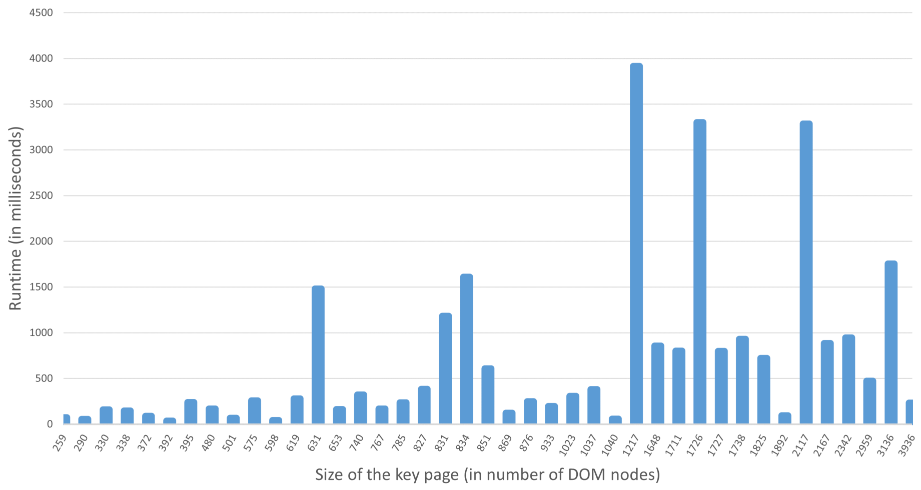
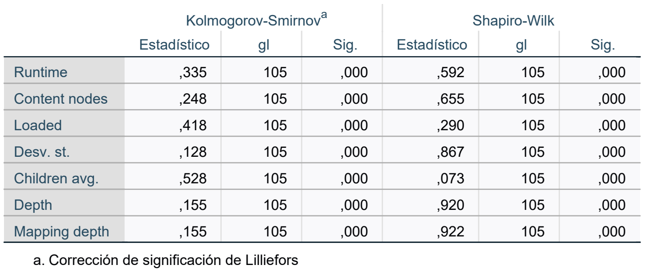


Figure 10.10: Relation between the size of the web page and the runtime

Hence, there is not a clear relationship between the size of the key page and the runtime of the algorithm, so we decided to explore other variables to check which are really related to the runtime of the algorithm. As in Chapter 9, the technique is site-level, and it is also based on the candidates selection and ETDM algorithms. Therefore, we computed the statistical analysis using essentially the same variables we used in that chapter, which are:

- The size of the main content measured in `number of content DOM nodes`. The choice of this variable is based on the assumption that the technique performs a mapping between several web pages to infer the main content of the key page. Therefore, the size of the real main content with high probability is related to the runtime.
- The `number of loaded web pages`. The selection of this variable is based on the fact that the technique needs to load several web pages to build the complete subdigraph. The more web pages it loads, the more runtime.
- The `standard deviation of the number of children of the element DOM nodes`. This variable was selected because, as stated above, the mapping process of DOM nodes with a high amount of children has a direct influence on the runtime, making it grow exponentially. This variable can evidence if there are nodes with a high amount of children.
- The `average number of children of the element DOM nodes`. As in the previous variable, this variable is also selected because the runtime of the algorithm grows due to the DOM nodes with a high amount of children.
- The `depth of the DOM tree of the key page`. The choice of this variable is based on the assumption that probably the mapping of deeper DOM trees takes more runtime.
- The `maximum depth reached by the mapping`. In this case, the variable is selected because probably the more depth reached by the mapping involves more runtime.

It should be highlighted that this statistical analysis was performed to the technique using the `runtime improvement` algorithm, therefore, the DOM tree of the key page and the web pages that belong to the complete



	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Runtime	,335	105	,000	,592	105	,000
Content nodes	,248	105	,000	,655	105	,000
Loaded	,418	105	,000	,290	105	,000
Desv. st.	,128	105	,000	,867	105	,000
Children avg.	,528	105	,000	,073	105	,000
Depth	,155	105	,000	,920	105	,000
Mapping depth	,155	105	,000	,922	105	,000

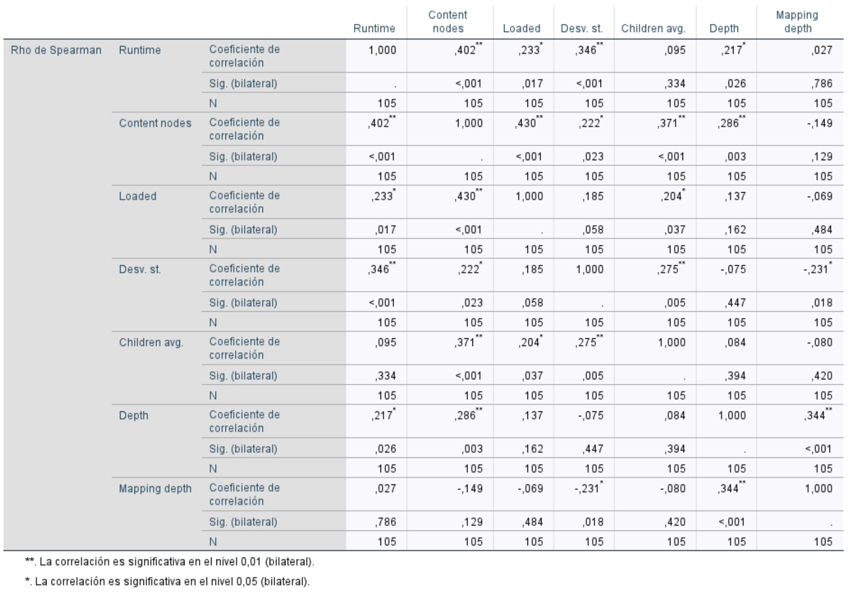
a. Corrección de significación de Lilliefors

Figure 10.11: Result of the normality test for site-level ConEx

subdigraph are optimized for the mapping. As in previous chapters, we used IBM SPSS Statistics to analyze the relationship between these variables. We also conducted the analysis using the test subset of the TeCo benchmark suite, formed by 105 web pages (see Chapter 13). First of all, we computed Table 10.11 to check whether the data were normally distributed. As we can observe in the table, the `sample size` (column gl) is 105, in consequence, the appropriate test to check the normality of the variables is Kolmogorov-Smirnov.

The table shows that the `significance` (column Sig.) of all variables is less than 0.05, therefore, the correlation coefficient can be computed through the Spearman test because the variables are not distributed normally. Figure 10.12 shows the result of the Spearman test for these variables. We can observe the correlation coefficient between all the variables and the runtime in the first row of the table.

The correlation coefficient of the `number of content DOM nodes` variable is higher than 0.4, while the correlation coefficient of the `standard deviation of the number of children of the element DOM nodes` is close to 0.35. These values state a clear relationship of these variables with the `runtime` of the algorithm. On the other hand, the correlation coefficient value for the variables `maximum depth reached by the mapping` and `average number of children of the element DOM nodes` are close to 0, therefore, these variables have no relationship with the `runtime`. Figure 9.8 also shows low values of the correlation coefficient for the variables `depth of the DOM tree of the key page`, and `number of loaded web pages`. These low values indicate that there is not a clear relationship between them and the `runtime` of the algorithm.



			Runtime	Content nodes	Loaded	Desv. st.	Children avg.	Depth	Mapping depth
Rho de Spearman	Runtime	Coeficiente de correlación	1,000	,402**	,233*	,346**	,095	,217*	,027
		Sig. (bilateral)	.	<,001	,017	<,001	,334	,026	,786
		N	105	105	105	105	105	105	105
	Content nodes	Coeficiente de correlación	,402**	1,000	,430**	,222*	,371**	,286**	-,149
		Sig. (bilateral)	<,001	.	<,001	,023	<,001	,003	,129
		N	105	105	105	105	105	105	105
	Loaded	Coeficiente de correlación	,233*	,430**	1,000	,185	,204*	,137	-,069
		Sig. (bilateral)	,017	<,001	.	,058	,037	,162	,484
		N	105	105	105	105	105	105	105
	Desv. st.	Coeficiente de correlación	,346**	,222*	,185	1,000	,275**	-,075	-,231*
		Sig. (bilateral)	<,001	,023	,058	.	,005	,447	,018
		N	105	105	105	105	105	105	105
	Children avg.	Coeficiente de correlación	,095	,371**	,204*	,275**	1,000	,084	-,080
		Sig. (bilateral)	,334	<,001	,037	,005	.	,394	,420
		N	105	105	105	105	105	105	105
	Depth	Coeficiente de correlación	,217*	,286**	,137	-,075	,084	1,000	,344**
		Sig. (bilateral)	,026	,003	,162	,447	,394	.	<,001
		N	105	105	105	105	105	105	105
	Mapping depth	Coeficiente de correlación	,027	-,149	-,069	-,231*	-,080	,344**	1,000
		Sig. (bilateral)	,786	,129	,484	,018	,420	<,001	.
		N	105	105	105	105	105	105	105

**. La correlación es significativa en el nivel 0,01 (bilateral).

*. La correlación es significativa en el nivel 0,05 (bilateral).

Figure 10.12: Result of the Spearman test for site-level ConEx

10.4 Conclusions

This chapter describes a new technique for content extraction from heterogeneous websites. It is a site-level technique, so it loads several web pages and uses their information to extract the main content. Concretely, given the key page, the technique analyzes it and extracts its hyperlinks. Then, the hyperlinks are sorted in order to select the web pages that are more likely to provide valuable information about the main content of the key page. As demonstrated by empirical evaluation, a set of 4 web pages (4-CS) obtains the best values of F1 for both, retrieved words and retrieved DOM nodes. The comparison of the DOM nodes from the set of web pages is done using an ETDM mapping. The comparison of the different web pages provides information about the number of times each DOM node is repeated in them. The technique considers that a DOM node that only appears on the key page and it is not repeated in others more likely belongs to its main content.

In essence, the algorithm maps the DOM nodes of the key page with several web pages from the same website in order to infer the branch of the DOM tree that probably contains the main content. The main idea is that usually the main content of a web page is not repeated on other web pages of the same website, that is, finding the non-repeated nodes may lead us to find the main content. In addition, the main content is usually condensed in the same part of the DOM tree, so the identification of the main content branch relies on this idea.

10.5 Contributions

The site-level content extraction technique described in this chapter provides several contributions that can be exploited by both, final users and many systems, such as indexers and crawlers.

The main contribution of the technique is that it is based on the combination of two techniques in order to infer the main content of a web page. The technique combines building a complete subdigraph with several web pages from the same website, with the comparison of those web pages through a mapping called equal top-down mapping. The nodes that only appear on the key page are considered possible main content nodes because they are not repeated in other web pages (set of candidate nodes).

Other important contributions are the algorithms that infer the template from the set of candidate nodes: first, an algorithm reduces the set of nodes to its minimal equivalent expression; finally, another algorithm detects the main content branch of the DOM tree, which corresponds to the main content.

As with the rest of the algorithms in this thesis, we implemented the technique as a WebExtension, which is compatible with Mozilla-based and Chromium-based browsers and is also officially published by Mozilla in their Firefox browser add-ons website.

Chapter 11

Hybrid Technique for Template Detection

Previous chapters introduce several block detection techniques (menu detection, template detection, and content extraction). As stated while describing those techniques, the main content of a web page is usually a pagelet which contains the most relevant information and is always surrounded by other noisy elements. This common distribution of web pages is really relevant for indexers and crawlers, because processing the noisy information may produce a waste of resources, such as storage, bandwidth, and time. Therefore, indexers and crawlers preprocess the web pages in order to extract the relevant information (main content) and isolate it from the noisy information (template or boilerplate). Moreover, there could be elements in a web page that are not part of the template because they are only repeated in a few web pages, but they can be detected as part of the main content. Adding a preprocess to remove the main content of the web page before computing the template is an interesting idea, because these elements that do not belong to the template, in some situations, can be detected as part of the template because sometimes they appear in several web pages.

Hence, as we had developed several template detection and content extraction techniques, and the main content and the template are not always complementary, we decided to implement a hybrid technique which combined two of our techniques. The key idea is to add a preprocess to our site-level template detection technique in Chapter 9 (TemEx) by removing the main content inferred by our page-level content extraction technique in Chapter 6 (page-level ConEx). Therefore, page-level ConEx works as a preprocess for TemEx. Obviously, adding this new phase to the TemEx algorithm would increase its runtime, however, it would not be especially significant because page-level ConEx is a page-level technique. The rationale behind this idea is that removing the main content would facilitate the template detection process because the web page would have none or less relevant information that can appear on several web pages, and therefore, be considered as a template. In conclusion, the technique described in this chapter (HybEx) would be more accurate than TemEx.

11.1 Related work

The template elements in a web page were measured by Gibson et al. [44] which estimated that they represent between 40% and 50% of all the data on the Web. Templates are important for the block detection discipline for many reasons. For instance, template detection is a discipline that depends on block detection. In addition, menu detection techniques are related to templates because the main menu of a web page is always located in the template. Furthermore, main content extraction techniques have to focus on the relevant data (main content) removing the non-relevant data.

As stated in previous chapters, it is possible to find in literature many block detection techniques (see, e.g., [120, 114, 116, 113, 24, 115, 66, 123, 90, 121]), especially main content extraction techniques (see, e.g., [46, 117, 106, 51, 118, 110, 78, 11, 81, 121]). However, despite that many researchers have been working in the field of template detection for the last 15 years, it is difficult to find hybrid algorithms that combine several block detection techniques, such as template detection and content extraction. Not even the latest block detection techniques (see, e.g., [115, 110, 124, 66, 123, 121, 53, 81]) implement another block detection phase as a preprocess. Many techniques implement simple preprocess methods such as removing nodes that surely do not have any content to extract (see, e.g., [115, 90, 110]) or standardizing the HTML code and precleaning it (see, e.g., [105, 79]).

However, Aslam et al. proposed a boilerplate removal technique called Web-AM [18] that combines the main content extraction algorithm Boilerpipe [63] with an algorithm that removes the noise of the web page. First, Boilerpipe extracts the main content of the web page. As the output given by Boilerpipe usually contains noise, then, the algorithm selects a seed-node and some cluster-nodes to detect the main content of the web page. Finally, the content extraction is performed by extracting text from the seed-node and the cluster-nodes. On the other hand, despite not combining several block detection techniques, some authors based their techniques on the combination of several methods or several kinds of information. For instance, Song et al. [104] propose a hybrid content extraction approach based on the combination of their measure of the text density (called textual information), and a visual measure for the evaluation of tags in web pages (called visual importance). Uzun et al. [111] proposed a hybrid method for extracting relevant content which is divided into two phases: the first one uses a machine learning method to discover informative content from the web page, while the second step extracts the relevant content using the rules obtained in the first step. This chapter presents a novel technique for template detection that combines the page-level ConEx content extractor (see Chapter 6) and the TemEx template detector (see Chapter 9).

11.2 Hybrid template detection

As TemEx, described in Chapter 9, this technique (HybEx) takes as input an arbitrary web page (the key page) and outputs a DOM tree that corresponds to its template. The technique is hybrid, that is, it combines two block detection techniques in order to detect the template. The basis of this technique is to use our page-level content extraction technique (page-level ConEx) as a preprocess of our template detection technique (TemEx). Inasmuch as page-level ConEx is a page-level technique, executing it as a preprocess of TemEx does not significantly affect its performance.

As with the rest of the techniques described in this thesis, this technique works at the level of DOM, and because of the DOM tree properties, the template of a web page can be identified with one or several DOM nodes.

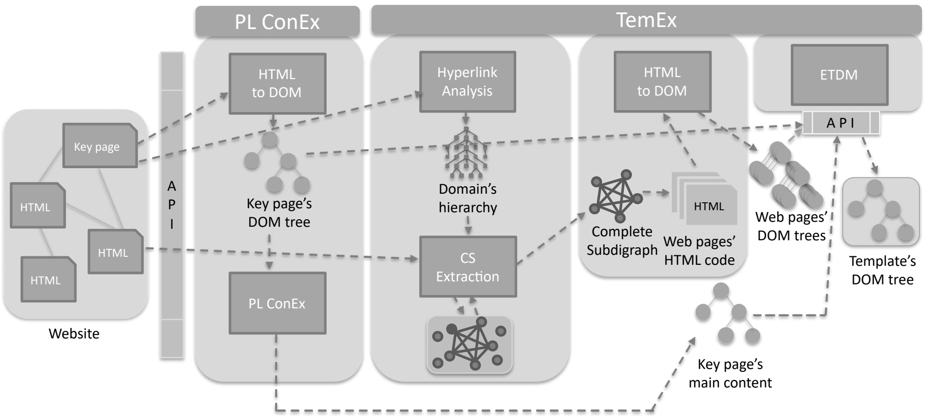


Figure 11.1: Hybrid template detection technique scheme

The technique, as Figure 11.1 shows, consists in a six-step approach:

i. An algorithm converts the HTML of the key page into its corresponding DOM tree.

ii. The page-level ConEx algorithm gets as input the DOM tree of the key page and performs all the steps described in Section 6.2. The output of this algorithm is the DOM tree of the main content.

iii. The hyperlink analysis algorithm selects a set of web pages from the same key page's website.

iv. A complete subdigraph is computed from the selected set of web pages.

v. An algorithm converts the HTML of all the web pages in the complete subdigraph into their corresponding DOM tree.

vi. Finally, the DOM tree of the key page is modified by removing the main content detected in step 2. Then, each web page in the complete subdigraph is explored by an algorithm that computes a mapping between its DOM nodes and the DOM nodes of the modified key page. When it finds that a DOM node of the modified key page is repeated in any web page of the complete subdigraph, it updates a counter that reflects the number of times each node is repeated in other web pages. When the number of times a node is repeated is equal to a specified threshold, the node belongs to the template. Finally, the algorithm returns the template.

The first 2 steps belong to page-level ConEx, while the last 4 steps correspond to the TemEx algorithm. It should be highlighted that step 1 is part of step 5 because step 5 converts the key page and other web pages from HTML to DOM. In addition, steps 1 and 2, and steps 3, 4, and 5 can be executed concurrently because the output of step 2 is the input of step 6. The 6 steps are described in the following sections.

11.2.1 HTML to DOM corresponding to page-level ConEx

All the techniques described in this thesis are based on DOM trees. As stated in Chapter 2, the representation of a web page as a DOM tree is the most extended approach (see e.g., [19, 120, 114, 80, 10, 125, 124, 90]).

As detailed in previous chapters, using DOM trees to represent webpages has many benefits for block detection techniques. In particular, it provides two main benefits to this hybrid content extraction technique:

- Due to the properties of DOM trees, the algorithm can return as output one or several DOM nodes. This is because a DOM node contains all its descendants and their information.
- Returning one or several DOM nodes as output (and not just text as most of the algorithms) allows the technique to output other kinds of content, such as images, animations, videos, etc.

This phase converts the HTML file corresponding to the key page into its DOM tree, which is used as input for both, page-level ConEx and TemEx algorithms.

11.2.2 Content extraction

This phase corresponds to the page-level content extraction algorithm (page-level ConEx) described in Chapter 6. The input of this phase is the DOM tree of the key page. As described in Chapter 6, page-level ConEx algorithm is divided into four stages:

i. The algorithm selects some DOM nodes of the DOM tree of the key page (which meet the criteria in Section 6.2) and, for each one, it computes several weights described in Definition 6.2.1: position ratio, children ratio, word ratio, and hyperlink ratio.

ii. Then, the value of the weights computed in the previous stage is standardized using the formula in Definition 6.2.2.

iii. Once the value of the weights has been standardized, the algorithm considers each node as a point in $\mathbb{R}^4$. Then, these DOM nodes (points in $\mathbb{R}^4$) are explored by an algorithm that computes the centroid. Once the centroid is computed, an algorithm builds the set of *candidate nodes* which includes the DOM nodes (points in $\mathbb{R}^4$) located farther than the centroid. Note that the distance between two points in $\mathbb{R}^4$ is computed using the Euclidean distance, shown in Definition 6.2.4.

iv. Finally, the set of *candidate nodes* is analyzed by an algorithm in the following way:

    - Those nodes which are descendants of other nodes in the set are removed if they share exactly the same text nodes as their ancestors.

- Then, for each remaining node in the set of *candidate nodes*, the algorithm computes the ratio between its words and its tags. It selects as the main content the node with a higher ratio together with its siblings that belong to the set of *candidate nodes*.

In addition to these stages, the page-level ConEx algorithm includes a final post-process that removes remaining groups of links that do not belong to the main content, such as links to other sections of the website, breadcrumbs, etc. This algorithm is detailed in Section 6.2. The output of these stages is one or several DOM nodes that represent the main content of the key page.

11.2.3 Hyperlink analysis

The first step of the template detection algorithm (TemEx), described in Chapter 9, is to identify a set of web pages from the same website of the key page that share their template with high probability. This phase is described in Chapter 7 as an independent process and can be used by any site-level block detection technique.

As it is detailed in Idea 7.2.4, the rationale behind the `hyperlink analysis` is that, in a website, those web pages located in the same folder probably share their template. The development of this idea leads to the need for establishing an order to sort the links of a web page, based on the `link relevance` (see Definition 4.2.5) and the `DOM relevance` (see Definition 4.2.6) orders.

As stated in Chapter 7, `link relevance` establishes an order to the links of a given web page based on the `hyperlink distance`, which is a metric that assigns a value corresponding to the distance from one directory of the website to another. See e.g., Example 7.2.6 which illustrates the process to compute the `hyperlink distance`. With high probability, there are many draws in the `link relevance` order. Therefore, another order is required to solve these draws. This order is called `DOM relevance` and it is based on the idea that hyperlinks located near others maybe contain repeated information apart from the template. This is not recommended for template detection because an algorithm could detect main content information as part of the template. Hence, in case of a draw, it is preferable to first select the links further away from the already selected links in the DOM tree.

Once `link relevance` and `DOM relevance` have been defined, an order for the links in a web page can be established. It promotes the links that should be explored by the template detection algorithm. This order combines both orders. First, the link relevance order ($\leq^{h}_{link}$), which uses the link distance algorithm to order the links of the web page. In case of a draw, the links with the same link relevance are ordered with the DOM relevance order ($\leq^{N}_{DOM}$), which uses the DOM distance algorithm.

In conclusion, the combination of both sorting algorithms produces the final order of the links. For instance, all the links with a hyperlink distance equal to 0 are then ordered based on their DOM distance (in order from highest to lowest). Then, this is repeated for the links with positive hyperlink distance, and finally for those with negative hyperlink distance.

11.2.4 Complete subdigraph extraction

As stated in Section 7.2, a complete subdigraph (CS) is a set of web pages that are pairwise and mutually linked. Therefore, an n-complete subdigraph (n-CS) is a CS formed by n nodes (web pages).

Once the hyperlinks of the web page have been ordered using the `link relevance` and the `DOM relevance` orders, Algorithm 9 explores them following the resulting order to find a CS in the website. The algorithm iteratively explores the links. When an n-CS is found the algorithm stops. It should be highlighted that the algorithm stops loading web pages when the n-CS is completed.

The output of this phase is a set of web pages that belong to the same website as the key page, and with high probability share the same template because all of them are pairwise and mutually linked.

11.2.5 HTML to DOM corresponding to TemEx

As described in phase 11.2.1, this algorithm converts an HTML web page into its corresponding DOM tree. In this case, it converts the HTML files corresponding to the web pages of the CS into DOM trees that are the input for the TemEx algorithm. Note that it is not necessary to convert the HTML file corresponding to the key page in its DOM tree because it has been converted in phase 11.2.1.

11.2.6 Template detection

As can be observed in Figure 11.1, the input of this phase is the DOM tree of the key page, the main content of the key page (computed in phase 11.2.2), and the DOM trees of the web pages that form the complete subdigraph.

In this case, it should be added a previous additional stage to remove the main content detected in phase 11.2.2 from the DOM tree of the key page. Hence, the TemEx algorithm will compute the template from the key page excluding its main content. This fact produces that the key page has no or less relevant content, thus the algorithm can compute the template with more accuracy.

Then, the DOM trees of the web pages that form the CS must be compared to the DOM tree of the key page (without the main content inferred in phase 11.2.2) in order to identify which DOM nodes they share. This comparison is carried out by the *equal top-down mapping* described in Chapter 8, which establishes a correspondence between the nodes of two given DOM trees.

As stated in Chapter 8.2.1, the definition of equal top-down mapping (ETDM) is parametric with respect to the equality relation $\triangleq$. Hence, it is more general than the standard equality (=). Concretely, it compares two DOM nodes by considering several properties, such as their HTML *tagName*, class names, HTML attributes, number of children, and their relative position in the DOM tree.

As mentioned above, an ETDM is computed between the DOM tree of the key page (excluding the main content inferred in phase 11.2.2) and each DOM tree of the web pages in the CS. The comparison is performed by traversing the DOM trees top-down starting from the root. Two DOM nodes n_1 and n_2 can be mapped if they are equal ($n_1 \triangleq n_2$). If so, the algorithm recursively continues by trying to map the children of both mapped nodes. Note that when the algorithm cannot map a DOM node, it does not continue exploring its descendants. As stated in Chapter 8, we consider that a DOM node belongs to the template when it appears in v web pages of the CS. It should be highlighted that $n \leq v$, being n the size of the complete subdigraph (n-CS).

The output of this phase is the DOM tree that corresponds to the template of the key page. Note that this is the last phase of the hybrid template detection algorithm.

11.3 Implementation

As with all the techniques presented in this thesis, this hybrid template detection technique has been implemented as a WebExtension, which is compatible with Mozilla-based and Chromium-based browsers, among others (see Chapter 14). The add-on consists of a single button located on the upper right of the browser window. When it is pressed, it performs the actions described in the previous section and extracts the template of the web page loaded by the browser, which is automatically displayed. If the button is pressed again, the original web page is displayed.

The evaluation of the technique, as the rest of the techniques in this thesis, has been performed using the Template detection and Content extraction Benchmark Suite (TeCo), described in Chapter 13. For the evaluation, we used the evaluation subset of TeCo, formed by 45 benchmarks. We computed the recall, precision, and F1 for retrieved DOM nodes. In addition, we also measured the runtime.

Example 11.3.1 *Figure 11.2 shows a real example of the use of the hybrid template detection tool with a web page. The image on the left is a web page of the* ***www.bbc.com*** *website. The image on the right is the output of extracting its template.*

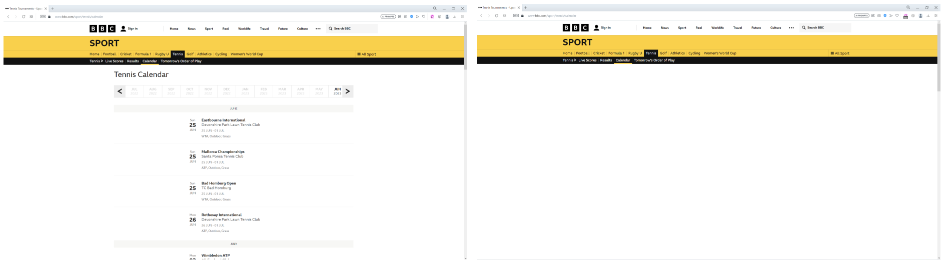

BBC
SPORT
Home Football Cricket Formula 1 Rugby U Tennis Golf Athletics Cycling Women's World Cup
Tennis Live Scores Results Calendar Tomorrow's Order of Play
Tennis Calendar
Eastbourne International
Mallorca Championships
Bad Homburg Open
Rothesay International
Wimbledon ATP
SPORT
Home Football Cricket Formula 1 Rugby U Tennis Golf Athletics Cycling Women's World Cup
Tennis Live Scores Results Calendar Tomorrow's Order of Play

Figure 11.2: Example of the detection of a web page template

11.3.1 Empirical evaluation

As stated in previous chapters, both algorithms (page-level ConEx and TemEx) contain some parameters that have been left open. On the one hand, in the case of page-level ConEx these parameters are $c\text{–}SET$ *size*, and *max. words*. On the other hand, the open parameters of TemEx are the size of the CS, the number of votes, the weight of the properties of the equality relation $\triangleq$, and the threshold of the value of the equality relation $\triangleq$).

With respect to the page-level ConEx technique, as described in Chapter 6, the $c\text{–}SET$ parameter corresponds to the optimal size of the set of candidate nodes while, for the hyperlink nodes in the main content, the *max.words* parameter corresponds to the maximum number of words of their descendants.

Regarding TemEx technique, as described in Chapters 7, 8, and 9, the parameter n corresponds to the optimal size of the CS and the parameter v is the number of votes required by a node to be considered as part of the template. In addition, Chapter 8 estimates the weight of the different properties that form the equality relation $\triangleq$. Finally, the threshold t of the equality relation $\triangleq$ was estimated (if $n_1 \triangleq n_2 \geq t$ both nodes are equal).

The value of these parameters is computed based on an experimental analysis in the empirical evaluation section of Chapters 6, 7, and 8. The optimal values are:

c-SET:	3.
max. words:	3.
CS size (n):	3.
Number of votes (v):	2.
Node class names:	0.10.
Node position:	0.10.
HTML attributes:	0.50.
Node children:	0.30.
$\triangleq$ *threshold (t):*	0.70.
Option:	2.

Algorithm evaluation

As stated above, to measure the technique several experiments were performed with the 45 web pages of the evaluation set of the TeCo benchmark suite (see Chapter 13). Once the template was detected, it was compared to the real template to compute the recall, precision, and F1 scores of the algorithm.

Table 11.1 shows the results obtained for all the executed benchmarks with the optimal parameters obtained empirically in Chapter 6, Chapter 7, and Chapter 8:

The first column of the table contains the URLs of the website domains. For each benchmark, column `Nodes` shows the total number of DOM nodes contained by the DOM tree of the key page; column `Templ.` contains the number of DOM nodes in the gold standard (template); column `Retr.` shows the number of DOM nodes detected by the tool as template nodes; column `Correct` contains the number of DOM nodes detected correctly by the tool; column `Recall` shows (in percentage) the number of DOM nodes retrieved correctly divided by the number of DOM nodes in the gold standard; column `Precision` contains (in percentage) the number of

Benchmark	Nodes	Templ.	Retr.	Correct	Recall	Precision	F1	Load	Runtime
www.jdi.org.za	619	394	388	388	98.48 %	100.00 %	99.23 %	6	259 ms.
www.premiere-urgence.org	480	438	439	429	97.95 %	97.72 %	97.83 %	4	173 ms.
www.indiangaming.org	575	201	199	199	99.00 %	100.00 %	99.50 %	4	56 ms.
hispalinux.es	501	345	355	343	99.42 %	96.62 %	98.00 %	4	119 ms.
www.gktw.org	767	637	682	637	100.00 %	93.40 %	96.59 %	5	424 ms.
www.apnic.net	598	453	516	452	99.78 %	87.60 %	93.29 %	4	196 ms.
www.unicef.org	1037	656	648	648	98.78 %	100.00 %	99.39 %	5	237 ms.
www.klimabuendnis.org	851	525	499	491	93.52 %	98.40 %	95.90 %	4	238 ms.
www.isoc-es.org	259	159	164	159	100.00 %	96.95 %	98.45 %	6	124 ms.
biztechmagazine.com	1892	1053	1757	1053	100.00 %	59.93 %	74.95 %	6	804 ms.
www.eeo.com.cn	834	626	647	596	95.21 %	92.12 %	93.64 %	13	1436 ms.
www.wishtv.com	2167	1811	1535	1535	84.76 %	100.00 %	91.75 %	5	6112 ms.
news.mit.edu	2117	1041	1828	1041	100.00 %	56.95 %	72.57 %	4	1660 ms.
asia.nikkei.com	869	662	608	607	91.69 %	99.84 %	95.59 %	6	556 ms.
www.rcnky.com	1738	1425	1627	1419	99.58 %	87.22 %	92.99 %	4	645 ms.
news.discovery.com	2826	1161	2424	1156	99.57 %	47.69 %	64.49 %	5	1267 ms.
www.kathimerini.gr	1825	1541	1257	1257	81.57 %	100.00 %	89.85 %	12	796 ms.
news.un.org	1726	1252	1175	1095	87.46 %	93.19 %	90.23 %	6	540 ms.
frances.forosactivos.net	785	290	140	140	48.28 %	100.00 %	65.12 %	26	159 ms.
www.wysiwygwebbuilder.com	3936	735	290	287	39.05 %	98.97 %	56.00 %	143	4160 ms.
www.3dprintforums.com	1040	276	267	267	96.74 %	100.00 %	98.34 %	6	263 ms.
www.strangehorizons.com	631	146	151	146	100.00 %	96.69 %	98.32 %	7	133 ms.
communities.apple.com	3136	368	1830	368	100.00 %	20.11 %	33.49 %	5	866 ms.
www.sloweurope.com	4193	514	498	491	95.53 %	98.59 %	97.04 %	5	224 ms.
community.ricksteves.com	2057	384	880	384	100.00 %	43.64 %	60.76 %	7	3840 ms.
hackercombat.com	1711	794	793	792	99.75 %	99.87 %	99.81 %	16	1156 ms.
www.scbwi.org	876	216	202	202	93.52 %	100.00 %	96.65 %	37	298 ms.
johngardnerathome.info	395	176	37	29	16.48 %	78.38 %	27.23 %	4	421 ms.
www.annmalaspina.com	392	182	228	182	100.00 %	79.82 %	88.78 %	4	67 ms.
foodsense.is	330	100	122	100	100.00 %	81.97 %	90.09 %	4	55 ms.
sites.google.com	372	287	125	125	43.55 %	100.00 %	60.68 %	4	123 ms.
whatever.scalzi.com	1648	1405	1405	1405	100.00 %	100.00 %	100.00 %	13	9106 ms.
www.javiercelaya.es	740	668	443	432	64.67 %	97.52 %	77.77 %	4	126 ms.
diarium.usal.es	604	80	76	76	95.00 %	100.00 %	97.44 %	4	868 ms.
www.jameslovelock.org	653	458	467	458	100.00 %	98.07 %	99.03 %	5	2282 ms.
www.cipri.info	933	377	375	375	99.47 %	100.00 %	99.73 %	4	212 ms.
naranjascarcaixent.com	290	148	144	144	97.30 %	100.00 %	98.63 %	5	70 ms.
www.technicalbookstoreonline.com	2959	386	367	367	95.08 %	100.00 %	97.48 %	32	458 ms.
www.floridarealestatecollege.com	1023	528	521	504	95.45 %	96.74 %	96.09 %	4	136 ms.
www.basf.com	827	762	792	762	100.00 %	96.21 %	98.07 %	4	228 ms.
www.mcphersonoil.com	831	600	620	596	99.33 %	96.13 %	97.70 %	4	407 ms.
www.thirteenhou.com	1217	133	135	131	98.50 %	97.04 %	97.76 %	4	44 ms.
www.embalajesterra.com	2342	1677	1774	1671	99.64 %	94.19 %	96.84 %	4	524 ms.
www.crypto.ch	338	248	248	248	100.00 %	100.00 %	100.00 %	4	217 ms.
www.shopbookshop.com	1727	1310	1313	1310	100.00 %	99.77 %	99.88 %	4	5822 ms.
Average	1281.49	613.96	688.69	566.60	91.20 %	90.70 %	88.29 %	10.36	1065 ms.

Table 11.1: Experimental evaluation results of the hybrid template detection algorithm

DOM nodes that have been retrieved correctly divided by the number of retrieved DOM nodes; column `F1` reveals the F1 metric and R the recall; column `Load` shows the number of DOM web pages loaded to compute the complete subdigraph; finally, column `Runtime` contains the total time used to compute the template (in milliseconds).

As Table 11.1 shows, the experiments obtain an average precision higher than 90%, an average recall higher than 91%, and an average F1 of about 87%. Compared to the results of TemEx (see Table 9.1), this hybrid algorithm improves its precision but it decreases its recall. With respect to the F1, the hybrid template detection technique obtains a value 0.75% higher than the TemEx technique.

Benchmark category	Recall	Precision	F1	Load	Runtime
Institutions / Associations	98.55 %	96.74 %	97.58 %	4.67	217 ms.
Media / Communication	93.32 %	81.88 %	85.12 %	6.78	1854 ms.
Forums / Social	85.87 %	84.21 %	78.39 %	28.00	1441 ms.
Personal websites / Blogs	79.69 %	92.86 %	82.20 %	5.11	3792 ms.
Companies / Shops	98.37 %	97.79 %	98.05 %	7.22	1128 ms.

Table 11.2: Experimental evaluation of the hybrid template extraction algorithm grouped by category

It can be observed in both tables that the *CS Extraction* algorithm needed to load 10.36 web pages on average to build the complete subdigraph. However, this value is due to a few benchmarks. Note that in 37 benchmarks the algorithm needed to load less than 10 web pages, therefore only in 8 benchmarks it needed to load more than 10. In particular, there is 1 benchmark where the algorithm needed to load more than 100 web pages to build the complete subdigraph.

With respect to the runtime, the hybrid template detection technique takes about 110 ms. more on average. This is because of the execution of the page-level ConEx algorithm as a preprocess of the template detection algorithm. However, it should be noted that the runtime of both algorithms is very similar. On the one hand, the runtime of the page-level content extraction algorithm implemented as a preprocess is low because it is a page-level algorithm. On the other hand, as the mapping is performed with fewer DOM nodes than the original TemEx technique (see Chapter 9), the runtime of this mapping is lower on average than the runtime of TemEx for the same benchmarks. These two factors explain the similarity between the runtime of this hybrid algorithm and the runtime of TemEx.

Table 11.2 shows the results of the evaluation grouped by benchmark category. It can be observed that *Institutions / Associations* and *Companies / Shops* obtain high F1 values. Compared to the values obtained by the TemEx algorithm in Chapter 9 (see Table 9.1), we can observe that the hybrid algorithm obtains higher F1 values for the categories *Institutions / Associations*, and *Forums / Social*, while the TemEx algorithm obtains better F1 values for *Personal websites / Blogs*, *Media / Communication*, and *Companies / Blogs*. However, the obtained results for *Media / Communication* and *Companies / Blogs* are nearly the same for both algorithms. It should be highlighted that the precision obtained with the hybrid algorithm is higher than the precision obtained by TemEx for all benchmark categories. Likewise, the recall obtained by TemEx is higher than the recall obtained by the hybrid algorithm for all benchmark categories.

11.4 Conclusions

This chapter describes a new technique for template detection from heterogeneous websites. As it is a site-level technique, it has to load several web pages from the same website to infer the template. The technique is a combination of the page-level content extraction technique (page-level ConEx) described in Chapter 6, and the site-level template detection technique (TemEx) presented in Chapter 9. Therefore, it is a hybrid template detection technique.

First, given the key page, the page-level ConEx algorithm extracts its main content. Then, the algorithm continues with the TemEx algorithm as usual. The technique analyzes the key page and extracts its hyperlinks. Once the hyperlinks are sorted, the algorithm computes a complete subdigraph of size 3 (3-CS). Then, the web pages from the complete subdigraph are converted from HTML to DOM trees, and they are the input of the ETDM. Subsequently, the algorithm removes the detected main content from the DOM tree of the key page and computes the ETDM between this modified key page with the DOM trees of the web pages in the complete subdigraph. Thus, it identifies the blocks that are common to those DOM trees, which correspond to the template of the key page. The parameters of both algorithms (page-level ConEx and TemEx) were computed empirically in their corresponding chapters.

As we can observe in the results of the empirical evaluation, this hybrid template detection algorithm improves the results obtained by the original TemEx algorithm (see Section 9.3.1). The F1 value obtained by this hybrid template detection algorithm is 0.75% higher than the value obtained by the original TemEx algorithm. By contrast, this algorithm is, on average, about 110 ms. slower than TemEx.

11.5 Contributions

The hybrid template detection technique described in this chapter provides several contributions that can be exploited by many systems, especially indexers and crawlers.

The main contribution of the technique is that it combines two block detection techniques in order to detect the template of a web page. The key idea is to combine a main content extraction technique (page-level ConEx) with a template detection technique (TemEx). The main content inferred by the main content extraction technique is removed from the key page.

Then, the key page is compared with other web pages from the same website in order to infer the template.

We implemented this technique as a WebExtension, which is compatible with Chromium-based and Mozilla-based browsers and is also officially published by Mozilla in their Firefox browser add-ons website.

Part V

Comparison with the State of the Art

Chapter 12

Comparison with the State of the Art

The specifications of a template detector or a content extractor depend on the intended use and the system where it will be inserted. While some systems require high recall, others require high precision, and others demand high efficiency. An analysis of the state of the art reveals that there exist several approaches in the literature. Each technique reports its own results, but it is not possible to find a fair comparison of techniques because each of them used different metrics and different benchmarks in their research. A process of literature review can produce a table containing the precision, recall, F1, efficiency, runtime, asymptotic cost, etc. reported by the authors of each technique. However, unfortunately, this information is not useful because the comparison would be unfear, imprecise, biased, and inaccurate. On the one hand, each technique has been implemented using a different technology which affects the efficiency. On the other hand, each author has evaluated its technique using different evaluation criteria (e.g., counting retrieved words [114, 113] vs. characters [61] vs. DOM nodes [10, 6] vs. text blocks [101, 115]), and using a different set of benchmarks. Using different collections of benchmarks to compare template detectors or content extractors is highly inaccurate because some techniques used artificial benchmarks [25] (automatically generated web pages that share exactly the same template) while others used real heterogenous web pages implemented by different designers [114, 10, 6]. In the same way, some authors selected the web pages randomly [120, 116, 50] possibly implementing different templates, while others manually provided web pages that implement exactly the same template [114, 113]. Finally, other authors used well-known benchmark suites such as CleanEval [20] benchmark suite [115, 99], MSS (Myriad 40 and Big 5) [85], L3S-GN1 [61], etc. In addition, it should be noted that for some techniques, the language of the web page can affect the performance. For instance, Jung et al. [53] demonstrated that some techniques such as web2text [115] may vary their performance depending on the language of the web page.

Nowadays there is no objective evidence, empirical data, or a widely accepted (subjective) consensus about template extraction or content detection tools (regarding the recall, precision, F1, performance, accuracy, scalability, and accepted technologies; i.e., HTML, CSS, JavaScript, etc.). Therefore, the results reported by different authors are not comparable.

Nevertheless, we can find several comparisons in the literature (e.g., [106, 115]) for content extraction techniques using the same metrics and the same sets of benchmarks. Unfortunately, however, we could not find in the literature fair comparisons for template detection techniques using the same metrics and benchmarks. The only way to fairly compare TemEx (described in Chapter 9) with other state-of-the-art techniques was to evaluate all of them using the same dataset, the same metrics, and in the same context (to properly compare runtimes). Therefore, we made a systematic review to select several template detection techniques and then we compared them. Once the techniques were selected [120, 114, 116, 113, 8], we tried to find their implementation, but it was not possible to compare them using the same benchmarks and the same accuracy measures, because it was not possible to access the implementation of any tool even though some of them were reported to be free. In addition, authors were asked to share their tools with very few positive answers. To solve this, we decided to reimplement them from scratch[1].

In order to integrate the implementations of the selected detection systems, a workbench for template detection has been created. This workbench is able to work offline (using a repository of websites) and online (it is integrated into the browser as a WebExtension that allows us to extract the template of a given website). One interesting property is that the loading of the web pages, the transformation from HTML to DOM trees, the renderization, etc. is orthogonal to the analysis of the web pages. Hence, all these features can be shared by different template detection algorithms. The use of this workbench to compare template detection algorithms produces a common evaluation criterion, so elements such as technology or loading time do not affect the comparison.

This chapter describes the process used to select, implement, and compare the selected template extraction and content detection techniques, as well as the obtained results.

[1]All of them are published, so they are open-source and publicly available at `http://personales.upv.es/josilga/retrieval/Web-TemEx/`

Contrarily to template detection, we could find a WebExtension based framework for assessing main content extraction methods [52]. The extension has several functionalities, such as creating a dataset, curating data, executing JavaScript/TypeScript extraction algorithms, and evaluating those algorithms using multiple measures. Additionally, we found in the literature several systematic reviews [98, 89] and comparisons [106, 115] of content extraction algorithms. We used the datasets and metrics proposed by those comparisons to evaluate our page-level content extraction algorithm so we could compare the obtained results with them.

12.1 Selection and description of web template detectors

12.1.1 Methodology for the selection of template detectors

This section describes the process followed to identify and select the template detectors that we compared to the template detection algorithm described in Chapter 9 (TemEx).

The process starts with the formulation of the research questions and the definition of the inclusion and exclusion criteria. Then, the processes of searching and screening primary studies are described.

i. *Research questions.* The identification of the current state of the art in boilerplate removal and template detection methods was achieved by the formulation of two research questions.

   - RQ1: What methods for template detection and boilerplate removal have been developed? The purpose of this research question is to obtain an overall perspective of the existing boilerplate removal and template detection methods, focusing on those that have been developed in the last 20 years.
   - RQ2: Which are the main characteristics of each template detection and boilerplate removal method? This question is an enhancement of the previous one since it broadens the knowledge of boilerplate removal and template detection methods.

ii. *Search process.* The main purpose of the conducted search process was to assess the body of knowledge related to template extraction and systematically answer to the research questions. The process was strict and unbiased, and it was carried out using the most relevant databases in the computer science area: Web of Science, Scopus, Science Direct, ACM Digital Library, IEEE Explore, Springer Computer Science, Citeseer X, Google Scholar, and Arxiv.

The following search string was created for the search:

(template OR boilerplate OR noise)
AND (detection OR extraction OR removal OR cleaning)
AND ("web page" OR webpage)

The creation of the search string was based on the analysis of several keywords obtained from relevant literature, which was found by exploring relevant articles and by reviewing their related bibliography.

Since the query resulted extremely wide (e.g., ScienceDirect returned nearly 9000 documents), the results were filtered by refining the search string:

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

Finally, 297 studies were obtained as the result of the search process. Once we discarded unavailable, duplicated, and non-related to the topic results, 50 papers were obtained.

iii. *Inclusion and exclusion criteria.* The following inclusion and exclusion criteria were defined to deal with the research questions:

- IC1: Those papers where the web pages are represented as DOM trees.
- IC2: Those papers that have been published in a conference with at least an A rating in the *GGS Conference Rating*[2].
- EC1: Those papers with less than 15 cites.[3]
- EC2: Those papers that report a F1 less than 50%.

iv. *Quality assessment.* We evaluated each paper using our own adaptation of a quality checklist that is used across multiple study types, which was proposed in [58] (see Table 7.3). We defined our 6 assessment criteria based on the 11 original quality assessment questions proposed by the authors:

[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.

- 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 collected in a way that addressed the research issue?
- AC5: Is there a clear statement of findings?
- AC6: Is the study of value for research or practice?

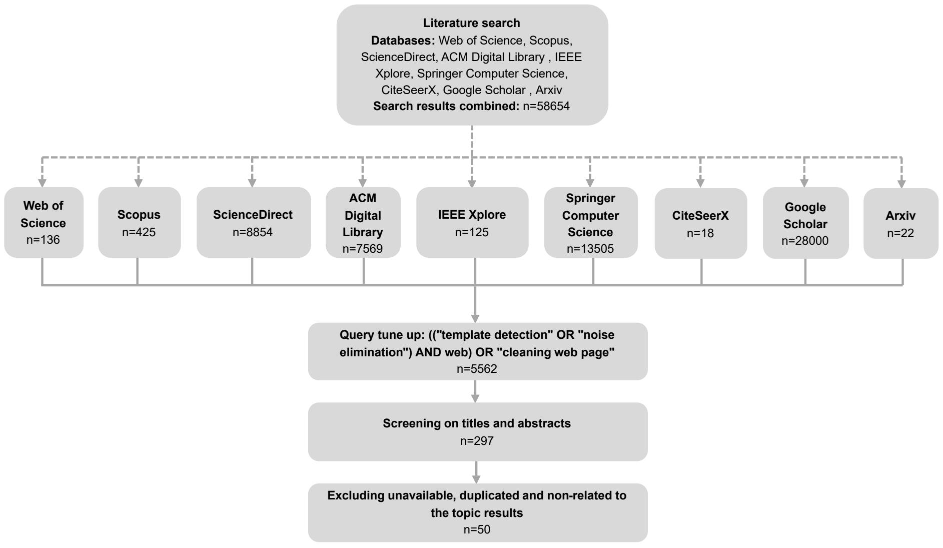

Literature search
Databases: Web of Science, Scopus, ScienceDirect, ACM Digital Library , IEEE Xplore, Springer Computer Science, CiteSeerX, Google Scholar , Arxiv
Search results combined: n=58654
Web of Science n=136
Scopus n=425
ScienceDirect n=8854
ACM Digital Library n=7569
IEEE Xplore n=125
Springer Computer Science n=13505
CiteSeerX n=18
Google Scholar n=28000
Arxiv n=22
Query tune up: (("template detection" OR "noise elimination") AND web) OR "cleaning web page" n=5562
Screening on titles and abstracts n=297
Excluding unavailable, duplicated and non-related to the topic results n=50

Figure 12.1: QUORUM flow chart

12.1.2 Search results

After the primary screening process, a total of 50 papers were selected and 247 were excluded from the 9 sources. Figure 12.1 shows the QUORUM flow chart of the reviewing process. The list of selected papers, including the metadata analyzed in the selection process, is shown in Table 12.1. The full text of each selected paper was read in order to decide if it had to be included in our study (and, thus, implemented it). Then, we selected the papers that met the inclusion criteria without meeting the exclusion criteria. Finally, a total of 4 papers were selected. Additionally, we added the template detection method described in Chapter 9 to compare it with the selected techniques. The result of the selection is shown as the rows with grey background in Table 12.1. All the selected techniques are described below:

Article	**Year**	**Venue**	**Core**	**LS**	**MA**	**JCR**	**Cites**	**DOM**	**P-L/S-L**
[19]	2002	WWW	A++	A++	A++	-	438	No	S-L
[120]	2003	SIGKDD	A++	A++	A++	-	581	Yes	S-L
[114]	2006	CIKM	A	A++	A+	-	131	Yes	S-L
[72]	2006	CSCWD	B	B	C	-	5	No	S-L
[67]	2006	DEXA	B	-	B	-	30	Yes	S-L
[55]	2006	IKE	C	-	-	-	11	No	S-L
[27]	2006	SAC	B	-	-	-	78	Yes	S-L
[25]	2007	WWW	A++	A++	A++	-	134	Yes	P-L
[59]	2007	PKDD	A	A	A+	-	12	No*	S-L
[77]	2007	WAC5	-	-	-	-	43	No	P-L
[36]	2008	LREC	C	A	A	-	64	No	P-L
[20]	2008	LREC	C	A	A	-	149	No	N/A
[116]	2008	WWW	A++	A++	A++	-	25	Yes	S-L
[60]	2009	WWW	A++	A++	A++	-	41	No	P-L
[83]	2009	KSE	B	-	-	-	11	No	S-L
[113]	2009	WWW	A++	A++	A++	-	28	Yes	S-L
[61]	2010	WSDM	A++	A+	A+	-	643	No	P-L
[43]	2011	IC3K	C	C	-	-	2	No	S-L
[91]	2011	ICCSIT	-	-	C	-	4	Yes	P-L
[57]	2011	TKDE	-	-	-	Q1	83	No	S-L
[54]	2012	IJCSE	-	-	-	-	8	Yes	S-L
[74]	2012	IJACR	-	-	-	-	4	Yes	S-L
[16]	2013	Inf. sci.	-	-	-	Q1	8	Yes	S-L
[111]	2013	Inf. proc. & man.	-	-	-	Q1	76	No	P-L
[93]	2013	IJCA	-	-	-	-	6	No	P-L
[84]	2013	IJRTE	-	-	-	-	16	Yes	P-L
[75]	2013	ICGCE	-	-	-	-	3	Yes	S-L
[65]	2014	App. mech. & mat.	-	-	-	Q4	1	Yes	S-L
[103]	2014	Wir. per. comm.	-	-	-	Q3	16	No	P-L
[41]	2014	ICDM	A++	A++	A	-	5	Yes	S-L
[34]	2014	ICACNI	-	-	-	-	7	Yes	S-L
[33]	2014	ICACCI	-	-	-	-	9	No	S-L
[50]	2014	IJCA	-	-	-	-	6	Yes	S-L
[108]	2015	IJCA	-	-	-	-	7	Yes	S-L
[35]	2015	IJCA	-	-	-		9	No	P-L
[32]	2015	IJETCR	-	-	-	-	3	Yes	S-L
[8]	2015	WWW	A++	A++	A++	-	10	Yes	S-L
[49]	2017	GJPAM	-	-	-	Q4	11	Yes	S-L
[109]	2018	Cluster computing	-	-	-	Q2	11	No	P-L
[15]	2018	WISE	A	B	B	-	2	Yes	S-L
[115]	2018	ECIR	A	A	B	-	33	Yes	P-L
[125]	2018	WISE	A	B	B	-	3	Yes	P-L
[18]	2019	FIT	-	-	-	-	3	Yes	P-L
[112]	2019	ICWE	B	B	C	-	2	No	P-L
[45]	2020	ACCESS	-	-	-	Q1	1	Yes	Both
[66]	2020	WWW	A++	A++	A++	-	15	No	P-L
[100]	2020	JESTR	-	-	-	Q3	2	Yes	P-L
[40]	2021	ICAIS	C	C	-	-	0	Yes	S-L
[11]	2021	TKDD	-	-	-	Q1	4	Yes	S-L
[12]	2022	WWW	A++	A++	A++	-	0	Yes	S-L

Table 12.1: Selected papers that describe web template extraction tools

SST (2003) [120]: This site-level technique describes a structure called *Site Style Tree* (SST) to represent a set of web pages. In essence, the SST is the union of all DOM trees from all web pages it represents. DOM nodes of a web page are represented in the SST in the same position as they appear, and also the brotherhood relations are kept so that the SST represents groups of sibling nodes explicitly. When groups of nodes are repeated on several web pages, they are represented in the SST with a counter. Therefore, the SST stores the information about the number of times a group of nodes is repeated. The most repeated groups of nodes belong to the template with a higher probability. Those groups of nodes are selected using a threshold. The authors used 5 commercial websites to carry out the evaluation experiments producing an F1 of 75.1 %. They do not provide any information about the measurement unit used.

- *Main goal*: Removing noisy blocks from web pages. Authors define 'noisy blocks' as not main content blocks, such as advertisements, navigation pans, copyright and privacy notices, etc.
- *Technologies used*: Authors do not give specific details on the programming language used, technologies, or layouts accepted by the tool.
- *Benchmarks used in their evaluation*: They used real web pages from 5 commercial websites: Amazon, CNet, J&R, PCMag, and ZDnet.
- *Limitations/Problems*: The main constraint of this technique is that it needs a large number of web pages (authors specify 500) to build the SST of each website. In addition, the technique was evaluated using homogeneous websites.

RTDM-TD (2006) [114]: This site-level algorithm takes as input a set of web pages and identifies which parts of their DOM trees are exactly equal. The template corresponds to those DOM nodes repeated in all web pages. The technique uses a top-down variant of the tree edit distance (TED) algorithm to compare a set of DOM trees. The technique randomly selects two web pages and computes their TED. All mapped nodes (they appear in both DOM trees) represent the current template. Then, the algorithm iteratively computes the TED between the current template and another randomly selected web page from the same website until a predefined number of web pages have been processed. The authors carried out the evaluation experiments using "a few dozen" manually selected web pages. They report an F1 with 10 websites "higher than 95%" in 9 out of 10 websites, and "above 85%" in the other one. The computation of the metric was based on the number of correctly retrieved words from the template.

- *Main goal*: Removing templates from collections of web pages in order to enhance web information retrieval and web mining methods. Initially, the template of a small set of sample web pages is detected. Then, the algorithm removes the previously computed template from the remaining web pages in the collection.
- *Technologies used*: Not reported by authors.
- *Benchmarks used in their evaluation*: Authors used real webpages obtained from 10 websites: the 5 websites used in [120] (Amazon, CNet, J&R, PCMag, and ZDnet), and other 5 well-known websites (CNN, E-Jazz, Encyclopedia Mythica, UBL, and Wikipedia).
- *Limitations/Problems*: The main limitation of the technique is the number of web pages that have to be analyzed to achieve a high F1 value. To obtain an F1 value near 95% they used 25 web pages. The computation of a top-down mapping between the DOM trees of 25 or more web pages can take a long time, although it depends on the amount of DOM nodes of the web pages. Moreover, the evaluation of the technique was done using only 10 websites.

IWPTD (2008) [116]: This site-level technique splits the DOM tree of the web pages into several subtrees whose root nodes correspond to DOM nodes associated with concrete HTML tags (i.e., UL, DIV, TABLE, etc.). Then, it extracts the text segments (DOM nodes of type `#text`) from the subtrees of all web pages and compares them. If a text segment belongs to 5 or more web pages, it is regarded as a template segment. Finally, the algorithm analyzes all the subtrees. It considers a subtree as part of the template if the quotient between the length of all its template segments and all its text segments is higher than 0.7. The authors carried out the evaluation experiments using 5 websites and 400 random web pages from each website. They obtained a precision of 98% and a recall of 80%. They do not provide any information about how their measures are computed.

- *Main goal*: Detecting templates in which a web page is processed as soon as it has been crawled. A framework detects the templates incrementally.
- *Technologies used*: Not reported by authors.
- *Benchmarks used in their evaluation*: Authors selected 5 websites from the dataset used in [73] and sampled 400 web pages from each. They do not mention which web pages from the dataset were selected.
- *Limitations/Problems*: The main constraint of the technique is that the evaluation was done using only 5 websites.

RBM-TD (2009) [113]: This technique is broadly similar to the RTDM-TD approach, but the authors introduce a bottom-up variant of the TED algorithm (instead of the top-down variant). During the DOM trees comparison, the algorithm introduces a restriction that allows us to classify a common subtree as part of the template: the position of a subtree repeated in all web pages must be exactly the same. Note that, for example, the position of a subtree can be computed as the path from the root to it, which has to be the same on all web pages. The authors carried out the evaluation experiments using 10 websites and manually selecting 24 web pages that implemented the same template from each. They report an F1 close to 90%, computed with 10 websites. The computation of the metric was based on the number of correctly retrieved words from the template.

- *Main goal*: Removing the template of the web page in order to improve its indexing and processing.
- *Technologies used*: Not reported by authors.
- *Benchmarks used in their evaluation*: Authors selected the same 10 websites used in [114]: Amazon, CNet, CNN, E-Jazz, Encyclopedia Mythica, J&R, PCMag, UBL, Wikipedia, and ZDnet.
- *Limitations/Problems*: The main constraint of the technique is that the evaluation was done using only 10 websites. In addition, the 24 web pages from each website used were not randomly selected, but all of them implemented the template, which is the easiest scenario for a template extractor.

TemEx (2015) [8]: As described in Chapter 9, this algorithm is in line with RTDM-TD and RBM-TD regarding the use of a mapping to determine what nodes belong to all web pages. In the case of TemEx, a template DOM node does not have to belong necessarily to all webpages, only to a subset, as in SST. Therefore, it is a more democratic algorithm since it uses a threshold (number of votes) to determine whether a DOM node is repeated in enough web pages to be considered from the template. This fact allows us to classify a set of DOM nodes as template nodes even if they are web pages that do not contain them. In some cases, this can reduce precision, however, it often increases recall, which uses to be the main handicap of all techniques. Chapter 9 reports an F1 with 45 websites of 87.57 %.

- *Main goal*: The technique allows website developers to reuse templates.
- *Technologies used*: Implemented with JavaScript and distributed as a WebExtension.
- *Benchmarks used in their evaluation*: 45 real heterogeneous websites. In addition, 105 additional heterogeneous websites were used to train the algorithm.
- *Limitations/Problems*: The main problem of the algorithm was the execution time of the mapping, which was its bottleneck. The problem has been solved with the runtime improvement algorithm described in Section 9.3.1.

It should be noted that it is not possible to compare the different F1 values reported by each tool. On the one hand, they were evaluated using different evaluation sets. On the other hand, some techniques do not specify the metric they used. Section 12.3 provides a fair comparison of these techniques.

12.2 A workbench for template detection

As stated at the beginning of the chapter, we had to implement the techniques selected in Section 12.1 [120, 114, 116, 113] from scratch. The implementation of the algorithms is included in a workbech[4] which provides several features for template detection.

The main benefit of this workbench is that it is parametric with respect to the algorithm used, namely, it contains an API so that a template de-

[4]It is available at `http://personales.upv.es/josilga/retrieval/Web-TemEx/`.

tection algorithm can access all the resources provided by the workbench. The architecture of the workbench is shown in Figure 12.2.

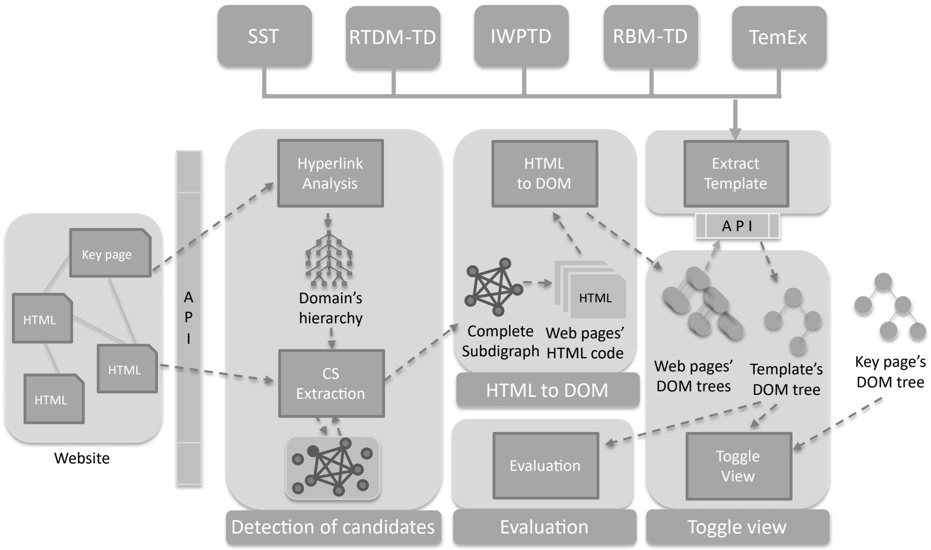


Figure 12.2: Workbench architecture

As can be noted from the figure, each module is represented using a squared dark gray shape, and its input and output are connected to it using dashed arrows. For instance, the `HTML to DOM` module inputs the web page's HTML code, and it outputs the DOM trees of those web pages.

It can be observed that modules are organized in four light gray areas which implement a particular functionality. Those areas are: `Detection of candidates`, `HTML to DOM`, `Toggle View`, and `Evaluation`. The `Extract Template` module implements the template extraction functionality, therefore, it has to be replaced with one concrete template detection algorithm, for example, one of the five already implemented that appear at the top of the figure. The lines below describe each area separately:

Detection of candidates: As stated in Chapter 7, a site-level template detector typically receives a web page as input and it outputs its template because it performs a comparison between that web page and other web pages from the same website that implement the same template. Thus, the template detection process is divided into two different stages: (1) Exploring the website and identifying web pages that probably implement the same template (web page candidates), and (2) comparing the web page with the web page candidates identified.

However, none of the algorithms selected and described in Section 12.1 (except for TemEx) takes account of this first phase. The description of their experiments shows that web page candidates were selected randomly from the website (IWPTD [116]), randomly from a predefined set of web pages that implemented the same template (SST [120]), or manually, all of them implementing the same template (RBM-TD [114], RTDM-TD [113]).

In contrast, the template detection workbench uses the hyperlink analysis algorithm described in Chapter 7 to automatically identify web pages that implement the web template with a high probability.

When comparing hyperlink analysis with a random selection of web pages, hyperlink analysis significantly improves the precision and recall of the second phase (for all algorithms). In addition, when compared to the manual selection, it has the advantage of being automatic because it just needs an URL to follow the links and automatically explore the website. It should be highlighted that the performance is significantly improved in this phase because the number of candidates needed to find the template is reduced. Another interesting aspect is that it is orthogonal to the second phase, so it can be used by all algorithms, thus the workbench performs this phase for all template detection algorithms.

HTML to DOM: Two main problems appear when researchers want to compare the results from different tools. On the one hand, almost every technique is evaluated with its own criteria. On the other hand, the authors do not provide any information about the evaluation criterion [120, 116]. Both situations make difficult a comparison between different techniques. Authors from papers in template detection literature usually evaluate how good its algorithm is using two measures: text or DOM nodes retrieved. The precision, recall, etc. can be significantly different if it is measured using the number of extracted template chars, the number of text words, the number of HTML tags, etc. Hence, it is essential to determine the evaluation criterion in order to compare the tools.

In addition, those papers that measure their techniques (precision, recall, etc.) using text can be further classified: those that measure chars, those that measure text words, and those that measure paragraphs. On this basis, it is reasonable to think that words are better than paragraphs, and chars are better than words because, e.g., if a technique measures words, it can detect that a word that should have been retrieved is missing in a paragraph, but this situation is impossible if the technique is measuring paragraphs (complete paragraphs would be marked as retrieved or not). The same situation would happen with words and chars.

Nevertheless, usually, extracting a subset of the chars in a word does not make sense. Besides, the measurement of retrieved chars is strongly dependent on the number of chars a word has. For instance, if the template word "publications" is not retrieved, it would produce a higher penalty than if the template word "about" is not retrieved, since "publications" has more chars. This may distort the measures.

The problems described above related to text can usually be solved by using DOM nodes instead of text to evaluate the techniques because that way each individual block of text that appears in the HTML code is represented with a `#text` DOM node. Thus, the use of DOM nodes instead of text to evaluate the template detection techniques is especially interesting for template extraction because templates reuse HTML labels and their blocks of text. It is not usual to find a template that does not reuse the whole text in a `#text` DOM node. For that reason, it is more appropriate to use DOM nodes to evaluate template detectors. Therefore, the workbench automatically transforms every web page to its associated DOM tree.

Toggle View: Before extracting the web page template, the workbench shows it directly on the browser and allows us to toggle the view by swapping from the extracted template to the original web page and vice versa. The template detection architecture in Figure 12.2 has a module that allows us to swap between the original DOM tree and the template's DOM tree. In our WebExtensions, when the template is extracted, a button allows the user to swap between the original web page and the template.

Evaluation: The workbench also includes a module for the evaluation of the produced template (precision, recall, F1, and runtime). Once the algorithms have generated the template, the workbench compares that template with the gold standard and shows the results in a report.

Runtime of each module

We computed the average runtime of each module using the test subset of the TeCo benchmark suite, formed by 105 web pages (see Chapter 13). Note that the module `Detection of candidates` has been divided into `Hyperlink analysis` and `CS extraction`. Table 12.2 shows the average runtime obtained by each module. Analyzing the obtained results, we observed that the runtimes of `Hyperlink analysis` and `CS extraction` modules are far greater than the runtimes of those modules for the evaluation subset of the TeCo benchmark suite, formed by 45 web pages. This phenomenon occurs when the module `Detection of candidates` has to load and analyze a large number of web pages in order to build the Complete Subdigraph. For instance, for 7 benchmarks in the test subset of the TeCo benchmark suite, the algorithm loads and analyzes more than 50 web pages. If those 7 benchmarks are removed from the experiment, the average runtime of the `Hyperlink analysis` module is 53 milliseconds, and the runtime of the `CS extraction` module is less than half millisecond. In fact, the computation of the Complete Subdigraph for a benchmark that has to load 392 web pages takes almost 380 seconds.

Despite the split of the TeCo benchmark suite (into a test subset and an evaluation subset) was performed randomly, the evaluation subset has only one benchmark for which the module `Detection of candidates` has to load more than 50 web pages (see e.g., Table 9.1). This means that the average runtime of the module `Detection of candidates` is significantly lower for the evaluation subset than for the test subset of the TeCo benchmark suite.

Module	Runtime
Hyperlink Analysis	129 ms.
CS Extraction	4668 ms.
HTML to DOM	0 ms.
Evaluation (TemEx)	1133 ms.

Table 12.2: Runtime of each module

12.3 Comparison of template detectors

The comparison of the selected template detectors and TemEx (described in Chapter 9) was carried out using the workbench for template detection described in Section 12.2). As detailed in the previous section, the workbench provides several features that allow researchers to fairly compare template detection techniques.

To ensure the heterogeneity of the web pages used in the comparison, we used the 45 evaluation web pages from the TeCo benchmark suite (described in Chapter 13). Table 12.3 shows the obtained comparison results of the accuracy and performance with the evaluation set of benchmarks. Column `Algorithm` indicates the algorithm used in the experiments; column `Recall` represents (in percentage) the number of DOM nodes correctly retrieved divided by the number of DOM nodes in the gold standard; column `Precision` shows (in percentage) the number of DOM nodes that have been retrieved correctly divided by the number of retrieved DOM nodes; column `F1` reveals the F1 metric which is computed as $(2 * P * R)/(P + R)$ where P the precision and R the recall; column `Load` indicates the total number of web pages loaded by the technique; column `Runtime` contains the total time used to compute the template (in milliseconds) As can be observed, our template detection algorithm (TemEx) achieves the best `F1` values. It also obtains the best precision of all algorithms. In addition, its performance is quite good compared to the other algorithms. It also obtains the best average runtime of all algorithms, just ahead of IWPTD ([116]).

Algorithm	Recall	Precision	F1	Runtime
SST (2003) [120]	37.54 %	58.03 %	41.90 %	1725 ms.
RTDM-TD (2006) [114]	15.94 %	98.15 %	16.53 %	2795 ms.
IWPTD (2008) [116]	75.80 %	65.65 %	65.17 %	**601 ms.**
RBM-TD (2009) [113]	40.68 %	**100.00 %**	51.52 %	1181 ms.
TemEx (2015) [10]	**93.09 %**	87.75 %	**87.54 %**	951 ms.

Table 12.3: Empirical evaluation and comparison with five site-level web template detection algorithms

A fair comparison of runtimes was ensured because the experiments were performed with the same computer, software configuration, and load. As all the techniques are integrated into the template detection workbench, they all use the same technology (WebExtensions). It should be noted that the first iteration was always discarded in order to provide more independence to the experiments, trying to avoid the impact of aspects such as the influence of dynamically loaded libraries persisting in physical memory, data persisting in the disk cache, etc. Moreover, for each technique and benchmark, we repeated the experiments until a standard deviation under

10% of the sample average was obtained in a window of ten executions. The returned statistic value was the average of the window.

TemEx clearly obtains the best `F1` value. The rest of the algorithms obtain noticeably lower values: between 16% and 65%. It should be noted that RBM-TD and RTDM-TD are focused on precision. RBM-TD obtained 100% precision in all experiments, while RTDM-TD obtained an average precision of 98.15%. In consequence, when it is required to retrieve only the template without noisy elements, both algorithms are valid. However, RBM-TD is better due to its higher F1. The table shows that the lowest `F1` values are obtained by RTDM-TD. With respect to RBM-TD and SST, they achieve similar `F1` values. Finally, IWPTD obtains a `F1` value of 65%, while SST obtains an F1 value close to 42%.

Based on the obtained results, TemEx should be used if precision or both precision and recall are crucial. On the other side, RBM-TD or RTDM-TD should be used if the recall needs to be maximized.

12.3.1 Computation time

Despite that all the algorithms use the same workbench, computation times must be analyzed in detail. The computation time of the `Candidates selection` phase varies depending on the algorithm because not all algorithms need the same number of web pages as input nor a complete subdigraph.

On the other hand, the algorithm's computation time is substantially different in all tools. While IWPTD and TemEx are the quickest algorithms, SST and RTDM-TD are significantly slower. Table 12.3 shows the mean of the execution time of each algorithm for the 45 evaluation benchmarks. IWPTD has the lowest runtime, which is about 600 milliseconds. TemEx only takes an average runtime of about 1 second per benchmark, while RBM-TD takes about 1.2 seconds per benchmark. However, SST is about 3 times slower than IWPTD, while RTDM-TD is significantly slower, nearly 5 times slower than IWPTD.

12.3.2 Scalability

The evaluation of the scalability has been performed by measuring the evolution of the runtime with regard to the increase in the number of nodes of the DOM trees. Figure 12.3 shows that TemEx and IWPTD are significantly better than RBM-TD, SST, and RTDM-TD. The figure draws the runtime trendline of each algorithm with respect to the number of DOM

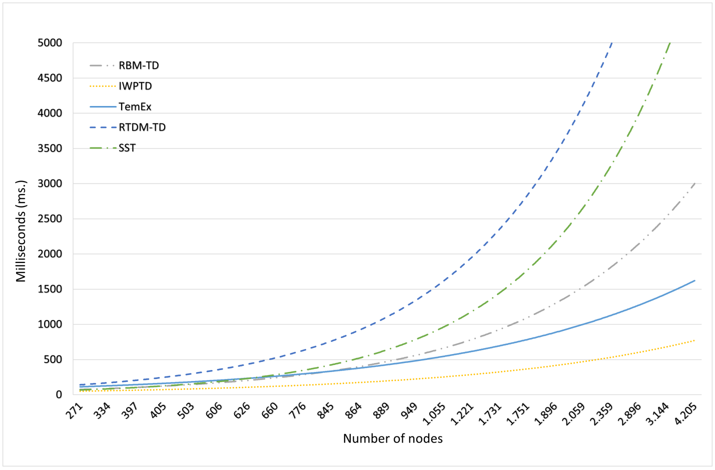


Figure 12.3: Runtime trendlines associated with DOM tree sizes

nodes on the key page. It can be observed that the trendlines of RBM-TD, SST, and particularly RTDM-TD are quadratic, while the trendlines of TemEx and IWPTD are linear with a slight incline. Hence, regarding scalability, TemEx and IWPTD are better than the rest of the algorithms.

12.3.3 Asymptotic costs

From a theoretical perspective, as we reimplemented the algorithms, we can study their scalability. Namely, we analyzed their asymptotic costs based on their source codes. We obtained the following measures:

- SST [120] $\in \mathcal{O}(W * (n^2 + T))$, being n the number of DOM nodes of the key page, W the maximum width (in the number of nodes) of the Site Style Tree and T the size (in the number of nodes) of the Site Style Tree.

- RTDM-TD [114] $\in \mathcal{O}(n^2)$, being n the number of DOM nodes of the key page.

- IWPTD [116] $\in \mathcal{O}(n + T \log T)$, being n the number of DOM nodes of the key page and T the amount of text segments extracted.

- RBM-TD [113] $\in \mathcal{O}(n^2)$, being n the number DOM of nodes of the key page.

- TemEx [10] $\in \mathcal{O}(n * W)$, being n the number of DOM nodes of the key page and W the maximum width of the DOM tree.

The analysis of the asymptotic costs confirms the obtained empirical results. While TemEx and IWPTD have linear growth, RBM-TD, RTDM-TD, and SST have quadratic growth. The best cost is the cost of IWPTD. It is practically $\mathcal{O}(n)$ due to the fact that the amount of obtained text segments is always significantly lower than the number of DOM nodes. The growth of TemEx is also linear. The cost of the rest of the algorithms (RBM-TD, RTDM-TD, and SST) is quadratic, being the asymptotic cost of RTDM-TD and RBM-TD $\mathcal{O}(n^2)$, which is better than the asymptotic cost of SST.

12.4 Comparison of content extractors

In contrast to template detection techniques, many content extraction algorithms have been compared using publicly available datasets (i.e., Cleaneval [20]).

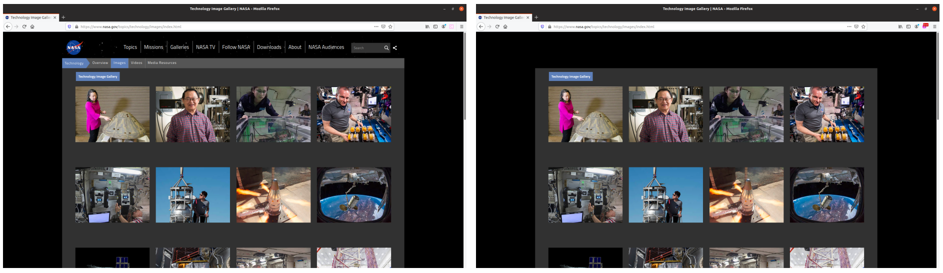


Figure 12.4: Image gallery from NASAS's website extracted with our web content extraction tool

Unfortunately, these datasets are only prepared for page-level techniques, so they facilitate the comparison of our page-level content extraction technique described in Chapter 6 (page-level ConEx). In contrast, we were unable to compare our site-level content extraction technique (site-level ConEx) with theirs.

It should be noted that, usually, each technique uses its own metric to measure the retrieved content. Therefore, we used the metrics and datasets introduced by the authors of those techniques in order to fairly compare our technique with the results reported by others.

First of all, we used the datasets and metrics proposed in [106]. Authors use 3 well-known and publicly available datasets (CleanEval; Big

5, which contains sets of web pages from Ars Technica, BBC, New York Times, Yahoo, and Wikipedia; and Chaos, which contains web pages from Blogger, Google News, and WordPress). First, we evaluated our page-level content extraction technique using their evaluation datasets and then, we used their metrics to compute the recall, precision, and F1. We compared our page-level ConEx algorithm with the algorithms included in [106] plus CEHTD-DS (which is a CETD variant). Table 12.4 reveals that CECTD-DS obtains the best F1 for most of the datasets. It also achieves the best precision and recall from several datasets and the best average F1. Moreover, other variants of the CETD algorithm (CETD-DS, CECTD-S and CEHTD-DS) achieve high average F1 values (over 90%). Our page-level algorithm achieves the best recall for the Yahoo dataset. The remaining algorithms (BTE, DSC, FE, K-FE, LQF and CCB) obtain lower F1 values, between 68% and 86% on average, except for FE, which achieves an average F1 of around 9%.

	CleanEval			NYTimes		
Algorithm	Prec.	Rec.	F1	Prec.	Rec.	F1
BTE (2001) [39]	88.87%	95.83%	92.22%	62.22%	**98.38%**	76.23%
DSC (2002) [88]	91.94%	62.01%	74.07%	98.58%	85.67%	91.68%
FE (2005) [30]	73.87%	9.97%	17.56%	97.51%	3.62%	6.98%
K-FE (2005) [31]	79.28%	69.61%	74.13%	73.82%	71.35%	72.56%
LQF (2005) [76]	88.60%	94.02%	91.23%	90.02%	97.10%	93.42%
CCB (2008) [46]	80.61%	92.71%	86.24%	57.61%	96.09%	72.03%
CETR (2010) [117]	91.26%	86.08%	88.59%	85.19%	90.58%	87.80%
CETD-DS (2011) [106]	92.96%	94.52%	93.73%	98.38%	95.84%	97.09%
CECTD-S (2011) [106]	90.35%	92.60%	91.46%	96.72%	96.56%	96.64%
CECTD-DS (2011) [106]	**95.87%**	**97.15%**	**96.51%**	99.69%	98.16%	**98.92%**
CEHTD-DS (2015) [104]	94.97%	94.07%	94.52%	**99.72%**	95.96%	97.80%
Page-level ConEx [11]	92.79%	92.35%	92.57%	98.55%	87.68%	92.79%

	Yahoo			Wikipedia		
Algorithm	Prec.	Rec.	F1	Prec.	Rec.	F1
BTE (2001) [39]	54.94%	95.06%	69.64%	83.91%	81.60%	82.74%
DSC (2002) [88]	96.54%	73.14%	83.23%	81.67%	34.61%	48.62%
FE (2005) [30]	**99.08%**	4.94%	9.41%	98.79%	1.48%	2.91%
K-FE (2005) [31]	69.49%	56.97%	62.61%	73.76%	44.60%	55.59%
LQF (2005) [76]	64.54%	90.65%	75.40%	83.60%	76.41%	79.85%
CCB (2008) [46]	46.90%	93.45%	62.46%	63.22%	73.14%	67.82%
CETR (2010) [117]	69.36%	77.65%	73.27%	94.69%	72.77%	82.30%
CETD-DS (2011) [106]	83.16%	85.90%	84.51%	98.31%	97.22%	97.77%
CECTD-S (2011) [106]	80.33%	93.34%	86.35%	98.02%	**97.61%**	**97.81%**
CECTD-DS (2011) [106]	84.59%	93.99%	89.04%	98.25%	92.77%	95.43%
CEHTD-DS (2015) [104]	91.99%	88.59%	**90.26%**	96.58%	90.41%	93.39%
Page-level ConEx [11]	67.16%	**97.29%**	79.47%	**98.90%**	94.61%	96.71%

It should be highlighted that all the CETD variants are only based on the text contained by the DOM nodes. Hence, in contrast to our techniques, all those algorithms ignore main content elements such as animations, images, video, and other media.

	BBC			Ars Technica		
Algorithm	Prec.	Rec.	F1	Prec.	Rec.	F1
`BTE` (2001) [39]	69.09%	97.09%	80.73%	68.25%	97.91%	80.44%
`DSC` (2002) [88]	89.27%	78.89%	83.76%	95.82%	90.52%	93.09%
`FE` (2005) [30]	**98.95%**	3.71%	7.15%	0.01%	0.00%	0.00%
`K-FE` (2005) [31]	63.84%	65.02%	64.43%	81.35%	82.73%	82.03%
`LQF` (2005) [76]	77.03%	92.17%	83.93%	88.40%	98.43%	93.15%
`CCB` (2008) [46]	53.52%	92.19%	67.72%	64.05%	96.27%	76.92%
`CETR` (2010) [117]	68.93%	86.58%	76.76%	83.06%	93.93%	88.16%
`CETD-DS` (2011) [106]	84.39%	95.21%	89.48%	97.81%	98.85%	98.33%
`CECTD-S` (2011) [106]	82.55%	93.77%	87.80%	94.61%	93.56%	94.08%
`CECTD-DS` (2011) [106]	86.15%	**97.95%**	91.67%	98.04%	**99.51%**	**98.77%**
`CEHTD-DS` (2015) [104]	95.55%	96.46%	**96.00%**	**98.12%**	98.83%	98.48%
`Page-level ConEx` [11]	93.15%	91.42%	92.28%	97.81%	97.03%	97.42%

	Chaos			Average		
Algorithm	Prec.	Rec.	F1	Prec.	Rec.	F1
`BTE` (2001) [39]	76.36%	92.80%	83.78%	71.95%	94.10%	80.83%
`DSC` (2002) [88]	94.45%	80.27%	86.79%	92.61%	72.16%	80.17%
`FE` (2005) [30]	72.59%	6.22%	11.46%	77.26%	4.28%	8.97%
`K-FE` (2005) [31]	73.97%	66.04%	69.78%	73.64%	65.19%	68.73%
`LQF` (2005) [76]	82.76%	93.98%	88.01%	82.14%	91.82%	86.42%
`CCB` (2008) [46]	64.45%	91.05%	75.47%	61.48%	90.70%	72.66%
`CETR` (2010) [117]	78.75%	86.92%	82.63%	81.61%	84.93%	82.78%
`CETD-DS` (2011) [106]	93.59%	94.99%	93.59%	92.66%	94.65%	93.50%
`CECTD-S` (2011) [106]	89.64%	91.22%	91.22%	90.33%	93.97%	92.24%
`CECTD-DS` (2011) [106]	**96.21%**	96.10%	**96.15%**	94.11%	**96.52%**	**95.21%**
`CEHTD-DS` (2015) [104]	94.74%	**96.42%**	95.57%	**95.95%**	94.39%	95.15%
`Page-level ConEx` [11]	95.01%	91.88%	93.42%	91.91%	93.18%	92.09%

Table 12.4: Empirical evaluation with CETD's metrics

To compare our page-level content extraction algorithm with the algorithms included in [106] we had to use the metrics proposed by them. They use the longest common subsequence algorithm (LCS): if a represents the text extracted and b represents the text in the gold standard, they compute the precision as the length of the LCS between a and b divided by the length of a, and they compute de recall as the length of the LCS between a and b divided by the length of b. As stated in Section 4.4, F1 is computed as $(2*P*R)/(P+R)$ where P is the precision and R is the recall. It is a fact that the use of these metrics (proposed by themselves and used in Table 12.4) favours all CETD variants because using the largest subsequence of text that is common to two strings only considers the text content of the web page, and not any other type of content.

Furthermore, we also evaluated our page-level content extraction technique with the metrics proposed in [115], which uses the CleanEval dataset. This allowed us to compare our algorithm with the algorithms included in their paper. The authors added one stage before computing their metrics. A dynamic programming algorithm finds the optimal alignment between the original HTML web page and the CleanEval gold standard. In addition, they align the obtained main content text with the text from the

CleanEval gold standard. Finally, they compute their metrics performing a comparison between both aligned texts. In order to decide if an obtained DOM node belongs to the main content or not, they use heuristics: if 2/3 of the retrieved node's content also belongs to the gold standard at the same location, the node is marked as "content". It should be noted that the authors use two datasets to evaluate their technique and to compare with others. On the one hand, they use the original CleanEval dataset. On the other hand, they use a reduced version of the CleanEval dataset which includes only 148 web pages selected by them.

	Cleaneval test				Web2text's test				
Method	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Runtime
`BTE (2001)` [39]	79%	79%	89%	83%	75%	76%	84%	80%	0.08 s.
`CRF (2008)` [105]	82%	87%	81%	84%	82%	88%	81%	84%	0.13 s.
`Default-ext (2010)` [63]	80%	89%	75%	81%	79%	89%	74%	81%	0.05 s.
`Article-ext (2010)` [63]	72%	**91%**	59%	71%	67%	89%	50%	64%	0.05 s.
`Largest-ext (2010)` [63]	60%	83%	36%	52%	59%	**93%**	33%	48%	0.05 s.
`Unfluff (2014)` [42]	71%	90%	57%	70%	68%	90%	51%	65%	0.52 s.
`Web2text (2018)` [115]	**84%**	88%	85%	86%	**86%**	87%	90%	**88%**	0.05 s.
`Page-level ConEx [11]`	83%	83%	**90%**	**87%**	84%	83%	**92%**	87%	0.27 s.

Table 12.5: Empirical evaluation with Web2text's metrics

The comparison results can be observed in Table 12.5. Page-level ConEx algorithm obtains the highest F1 value for the CleanEval dataset (87%), and the highest recall for both datasets, the original CleanEval test dataset (90%), and the CleanEval web2text's test dataset (92%). The best accuracy for both datasets is obtained by Web2text. In addition, the best precision for both datasets is obtained by Boilerpipe. We also obtained the publicly available implementation of the algorithms proposed in [115] and evaluated their performance. There are significant differences between the runtimes, as can be observed in Column *Runtime* of Table 12.5. This occurs because the algorithms are implemented with different technologies, such as Java, Python, Perl, Scala, etc.

It should be highlighted that all the content extraction algorithms in this section focus on text extraction. This is evidenced by the fact that most authors use metrics that only consider the extracted text. On the other hand, as Figure 12.4 shows, our both content extraction algorithms extract the main content of the web pages regardless of its type (it not only extracts text but also animations, images, videos, etc.). Therefore, the metrics used in this section to evaluate the algorithms (those proposed by the other techniques) influence our technique, because the non-textual information extracted by our algorithms (such as images, videos, animations, etc.) is not considered.

In addition, it should be highlighted that many block detection techniques compute the average F1 as $(2 * P * R)/(P + R)$, being R the average recall, and P the average precision of all benchmarks (see e.g. [115, 104, 117, 106, 66]). In contrast, we measured the average F1 of our techniques in the same way as the average recall and the average precision, computing the average of the F1 obtained by each benchmark (see e.g. Table 5.2, Table 6.3, Table 6.4, etc.). Computing the average F1 using the average recall and the average precision distorts the obtained results. As for each benchmark, the F1 is always less or equal to the average of the recall and the precision, computing the average F1 of all benchmarks using the average recall and the average precision will most likely lead to a higher value than the real average F1.

12.5 Conclusions

The evaluation and comparison of our template detection and content extraction algorithms with other state-of-the-art techniques revealed that they obtain the best results in various datasets.

On the one hand, we compared our site-level template detection algorithm (TemEx) with several well-known template detection techniques. As we could not find in the literature fair comparisons for template detection techniques using the same metrics and benchmarks, we made a systematic review to select several template detection techniques and then compared them. It was not possible to access the implementation of any of the selected techniques, so we had to implement them from scratch (now they are open-source and publicly available). We included the implementation of the algorithms in a workbench that includes several features for template detection.

On the other hand, we compared our page-level content extraction technique (page-level ConEx) with other well-known content extraction techniques using different datasets and metrics. In this case, we did not have to make a systematic review to select the techniques, nor implement the techniques from scratch. We compared our technique with several well-known techniques from the state-of-the-art using their datasets and metrics. As a result, to the best of our knowledge, we offered a general view of the strong and weak points of several content extraction techniques. In addition, we identified the best tool for different possible scenarios. From the comparison, we concluded that most techniques are only focused on text extraction. In contrast, our two content extraction techniques are able to extract the main content regardless of its type, so they not only extract text, but also animations, images, videos, etc.

12.6 Contributions

This chapter provides several contributions that can be exploited by researchers in order to evaluate and compare their block detection techniques with the techniques in the state of the art.

The main contribution of the chapter is the workbench for template detection. It is parametric with respect to the algorithm used, so any template detection algorithm can access to all resources it provides. It implements four common modules that can be used by any template detection algorithm if needed: `Detection of Candidates`, `HTML to DOM`, `Toggle View`, and `Evaluation`. The `Extract Template` module has to be implemented depending on the template extraction algorithm, so it corresponds to de algorithm itself.

Another important contribution of the chapter is the implementation of 5 template extraction algorithms. In addition to our template detection algorithm (TemEx [10]), we implemented from scratch 4 well-known template extractors: SST [120], RTDM-TD [114], IWPTD [116], and RBM-TD [113]. As with the rest of the algorithms in this thesis, the template detection workbench and the techniques were implemented as a WebExtension, which is compatible with Mozilla-based and Chromium-based browsers.

Finally, the chapter provides several comparisons of template detection and content extraction algorithms using the same metrics. This contribution allows researchers to fairly compare their techniques with some well-known block detection techniques in the literature.

Part VI

Implementations

Chapter 13

TeCo Benchmark Suite

As stated in previous chapters, block detection disciplines such as template detection, content extraction, menu detection, etc. are important tools for website developers, website analyzers such as crawlers, and also for many other information processing tasks applied to web pages. In the last fifteen years, there have been important improvements that produced several techniques for block detection disciplines. For the tasks of testing, comparing and tuning these techniques, researchers need:

- sets of heterogeneous benchmarks (that guarantee generality of the techniques) and
- a gold standard (that ensures the same evaluation criteria).

Benchmark suites are extremely significant to measure the performance of block detection techniques, as well as to compare them with previous approaches. They are used both in the testing phase and in the evaluation phase. First, the testing phase is where parameters are adjusted in order to optimize the techniques. Then, the evaluation phase obtains the performance of the technique using objective measures. Nevertheless, one cannot overlook the fact that the same set of benchmarks cannot be used in the testing phase and in the evaluation phase, that is, they need disjoint sets of web pages.

This chapter presents a benchmark suite together with a gold standard that is useful for several block detection techniques, such as menu detection, template detection, and content extraction. The key page of every benchmark has been labelled so that a block detection technique can find out whether a DOM node represents the main menu, whether it should be classified as part of the template or not, or whether it should be classified as main content or not.

TeCo benchmark suite is the result of a research project. It started from a technique for content extraction [51] and another technique for tem-

Date	May 14	Jun 16	Nov 17	Dec 18	Jun 21
Version	1.0	2.0	3.0	4.0	5.0
Benchmarks	40	50	100	130	150

Table 13.1: Number of benchmarks of each TeCo version

plate detection [14]. At first, in the evaluation phase, we tried to use a public benchmark suite. CleanEval [20] seemed a good option because it has been widely used in literature, but unfortunately, it is not prepared for site-level techniques nor it is not prepared for template detection. Then, we contacted the authors of other techniques that had used benchmarks of heterogeneous web pages to evaluate them. Nevertheless, we could not use those benchmarks for several reasons, such as privacy issues (they belonged to a project or a company whose results were private), unavailability (they had been lost), and copyright (they were not publicly available). In other cases, when the benchmarks were available, additional problems arose. For instance, some benchmark suites were only prepared for page-level techniques, others included only a few web pages, etc. So, finally, we built our own benchmark suite which we made free and publicly available. The first version [7] contained 40 benchmarks. Later, with the development of a new menu detection technique [3], we updated the benchmark suite by labelling the DOM nodes that represent the main menu of the website. So we added 10 new benchmarks to the suite, for a total of 50. Then, with the publication of [13], we doubled the number of benchmarks, for a total of 100. Later, we added 30 more benchmarks for a total of 130. Finally, 20 more benchmarks were added, thus now our benchmark suite is formed by 150 websites. Table 13.1 shows the approximate date of each TeCo version and the number of benchmarks included in it.

13.1 Benchmark suite's structure

TeCo (Template detection and Content extraction benchmark suite) was specifically created as a benchmark suite for block detection techniques, specifically for template detection, menu detection, and content extraction. It can be used in both, the testing and the evaluation phase of these techniques. It contains 150 real websites downloaded from the Internet. We selected and downloaded real heterogeneous websites with different layouts and page structures such as blogs, shops, company websites, forums, newspapers, personal websites, sports websites, etc.

The language of the websites is also heterogeneous. Despite the dominant language being English, benchmarks include many other languages such as Spanish, French, German, Italian, Japanese, Chinese, etc. Some of them are well-known websites, like the FIFA website or the BBC website. On the other hand, others are less known like personal blogs or websites from small companies. The downloading of the web pages was done using the Linux command `wget`.

The Linux terminal command used to download the websites using `wget` was the following:

```
    $ wget --convert-links --no-clobber --random-wait -r -l 2
-p -E -e robots = off -U mozilla --reject pdf, mp4, zip
http://www.example.org
```

The meaning of the flags used is:

- `--convert-links`: Converts links so they can work locally.
- `--no-clobber`: Do not overwrite any existing file.
- `--random-wait`: Random waits between downloads.
- `-r`: Recursive downloading.
- `-l 2`: Limit the recursive downloading up to 2 levels of links.
- `-p`: Downloads everything.
- `-E`: Get the right file extension.
- `-e robots = off`: Act as not being a robot.
- `-U mozilla`: Identify as a Mozilla browser.
- `--reject pdf, mp4, zip`: Rejects files with these extensions.

The composition of each benchmark is:

- A main web page, called *key page*. This is the target web page from which the block detection techniques should extract the desired block (main menu, template, or main content). It should be noted that this is not necessarily the main web page of the website (e.g., index.html), it could be any web page from the website.
- A set of web pages that are reachable recursively (up to 2 levels of links) from the key page. Moreover, they belong to the same website

as the key page. Note that, usually, this set of web pages does not contain all the web pages from the website, only in the case of small websites.

13.2 Producing the gold standard

TeCo benchmark suite includes a gold standard that serves as a reference to compare different block detection techniques. For each key page, the gold standard specifies what parts form the template, what parts form the main content, or which DOM node represents the main menu. This attribute is included in the own web page through the use of HTML classes that indicate, for instance, which elements are classified as *mainContent*, which are classified as *notTemplate*, and which are classified as *mainMenu*. This gold standard has been produced manually by several people who carefully inspected the websites.

Specifically, before downloading all the websites (the key page and recursively two levels of linked web pages that belong to the same website), four different engineers did the following procedures independently:

- They manually inspected the key page and the web pages reachable from it to decide which part of the web page is the main content, which part is the template, and which is the main content.

- They printed a snapshot of the blocks they defined in the previous step (main content, template, and main menu).

Subsequently, the four engineers had a meeting and performed again these two procedures, but now sharing their individual opinions. With the conclusions of this agreement, each website was prepared for template detection, content extraction, and menu detection. On the one hand, an HTML class called *TECO_notTemplate* was used to include all the DOM nodes from the key page that did not belong to the template. This allows a template extraction tool to perform a comparison between its output and the nodes not belonging to the *TECO_notTemplate* class. On the other hand, an HTML class called *TECO_mainContent* was used to include all the DOM nodes belonging to the main content of the key page. Hence, a content extraction tool can easily perform a comparison between its output and the nodes that belong to that class. In addition, an HTML class called *TECO_mainMenu* was used to include the DOM node that represented the main menu of the key page. In consequence, a menu extraction tool can perform a comparison between its output and the nodes that belong to that class.

13.3 Benchmark details

TeCo benchmark suite also provides a classification of the benchmarks, which could be useful depending on the application and the technique that is being tested or evaluated. Different classifications are provided according to the properties and purpose of the benchmarks. All the benchmarks are classified into five categories:

- Companies / Shops.
- Forums / Social.
- Personal websites / Blogs.
- Media / Communication.
- Institutions / Associations.

This classification and the URLs of the websites that belong to each category can be observed in the following tables:

- Table 13.2 for the benchmarks that belong to the Institutions / Associations category.
- Table 13.3 for the benchmarks that belong to the Media / Communication category.
- Table 13.4 for the benchmarks that belong to the Forums / Social category.
- Table 13.5 for the benchmarks that belong to the Personal / Blogs category.
- Table 13.6 for the benchmarks that belong to the Companies / Shops category.

We randomly selected 9 benchmarks of each category as evaluation benchmarks, while the remaining 21 benchmarks of each category form the training subset of benchmarks. Therefore, considering the 5 categories

Id	Original URL of the web page
1	web.mit.edu/institute-events/visitor/
2	www.museodelprado.es
3	www.u-tokyo.ac.jp/en/about/history.html
4	www.savethechildren.net/what-we-do/our-humanitarian-work/
5	college.harvard.edu/financial-aid/
6	www.linuxfoundation.org/about/
7	clinicaltrials.gov/ct2/search/index/
8	cordis.europa.eu/fp7/ict/fire.html
9	parents.berkeley.edu/advice/babies/laundry.html
10	www.mit.edu/campus-life
11	cpoepalencia.es/federaciones-y-asociaciones-confederadas-asociaciones/
12	www.icann.org/history.html
13	www.gip-jci-justice.fr/en/about-us/support-council/
14	www.einstein.yu.edu/leadership/
15	www.americanacademy.de/about/
16	www.mensa.es/cms/pages/%C2%BFqu%C3%A9-es-mensa.html
17	www.bcrf.org/breast-cancer-research.html
18	www.ielts.org/what-is-ielts/ielts-introduction.html
19	fr.unesco.org/about-us/introducing-unesco.html
20	www.ccbe.eu/about/who-we-are/
21	www.fraud.org/get_involved.html
22	www.jdi.org.za
23	www.premiere-urgence.org/qui-sommes-nous/
24	www.indiangaming.org
25	hispalinux.es/QuienesSomos
26	www.gktw.org/about/
27	www.apnic.net/about-apnic/organization/vision-mission-objectives/
28	www.unicef.org/where-we-work.html
29	www.klimabuendnis.org/home.html
30	www.isoc-es.org

Table 13.2: Sources of the Institutions / Associations benchmarks

of benchmarks, the evaluation subset is formed by a total of 45 benchmarks while the training subset is formed by the remaining 105 benchmarks. These 2 subsets of the TeCo benchmark suite have been used to train and evaluate all the algorithms in this thesis.

Table 13.7 shows some properties of each benchmark. Here, column **Benchmark's domain** is the domain of the website's key page, column **Type** indicates whether the benchmark is a training benchmark (T) or an evaluation benchmark (E), column **Nodes** shows the total number of DOM nodes of the key page, column **T. Nodes** refers to the number of DOM nodes of the template, column **M.C. Nodes** indicates the total number of

Id	Original URL of the web page
31	edition.cnn.com
32	www.neoteo.com/star-wars-the-force-awakens-el-regreso-de-viejos-personajes/
33	riotimesonline.com
34	www.turfparadise.com
35	www.cleanclothes.org
36	www.afp.com/es/contact/
37	www.history.com
38	detroit.cbslocal.com/2018/12/04/high-school-newspaper-suspended-after-publishing-disruptive-investigation/
39	www.rocklists.com/91x-1983.html
40	www.lashorasperdidas.com
41	www.journalism.org/2014/03/13/social-search-direct/
42	www.socialmediatoday.com/news/facebook-adds-new-features-for-instant-articles-including-links-to-more-pu/569786/
43	www.diariodeburgos.es/Noticia/Z1C5D6DE9-D1E6-B03A-61236AF21520B8B2/202002/Un-programa-verde-dedicado-a-Felix-Rodriguez-de-la-Fuente.html
44	wordofmouthmendo.com/word-of-mouth-stories/2018/5/31/travellers-fare.html
45	www.usine-digitale.fr/article/la-start-up-americaine-clearview-ai-illustre-deja-les-derives-de-la-reconnaissance-faciale.N921119.html
46	1015fm.com.au/2020/02/steve-mickenbecker-interest-rates-on-hold-2020-02-07/
47	www.dw.com/de/lebron-james-vom-pflegekind-zum-basketball-superstar/a-52088565.html
48	www.theday.com/movies–tv/20200203/super-bowl-ads-dialed-up-fun-as-antidote-to-politics.html
49	nltimes.nl/2019/12/16/chocolate-spread-babies-wins-misleading-product-award.html
50	www.bbc.co.uk/news/
51	techcrunch.com/gadgets/
52	biztechmagazine.com/article/2019/12/why-byod-makes-endpoint-security-crucial-small-businesses.html
53	www.eeo.com.cn/2022/0506/533366.shtml
54	www.wishtv.com/news/flu-is-widespread-across-the-us/
55	news.mit.edu/2021/grand-decoding-data-0909.html
56	asia.nikkei.com/Spotlight/Sharing-Economy/New-Tokyo-homes-ditch-parking-spaces-but-offer-car-sharing
57	www.rcnky.com/articles/2021/09/12/ft-mitchell-reflects-life-age-104.html
58	news.discovery.com/tech/robotics/artificial-intelligences-hawkings-fears-stir-debate-141206.htm
59	www.kathimerini.gr/society/561833251/koronoios-arsi-metron-i-megali-prova-kanonikotitas-enopsei-toy-kalokairioy/
60	news.un.org/en/content/navigate-news

Table 13.3: Sources of the Media / Communication benchmarks

Id	Original URL of the web page
61	es.sharelatex.com/learn/Uploading_a_project
62	github.com/DawidStankiewicz/forum
63	en.citizendium.org
64	www.filmaffinity.com/es/
65	www.meneame.net/faq-es/
66	www.accountkiller.com/en/delete-activision-account
67	study.com/learn/science-questions-and-answers.html
68	c.mi.com/it/
69	alumni.harvard.edu/help/message-board/
70	www.spacetimestudios.com/forumdisplay.php?29-Websites-and-Forum-Discussion
71	www.gimpforum.de
72	www.emaildiscussions.com
73	forums.debian.net/viewforum.php?f=5
74	forums.mozillazine.org/viewforum.php?f=23
75	forums.tomsguide.com/forums/laptop-general-discussion.15/
76	forums.mysql.com/list.php?21
77	lawstudents.ca/forums.html
78	www.japanesepod101.com/forum/viewforum.php?f=26
79	forum.skyscraperpage.com
80	forums.opera.com
81	forums.linuxmint.com/viewforum.php?f=72
82	frances.forosactivos.net
83	www.wysiwygwebbuilder.com/forum/viewforum.php?f=10
84	www.3dprintforums.com
85	www.strangehorizons.com/2004/20040906/greenglass-f.shtml
86	communities.apple.com/es/community/mac_os/os_x_el_capitan.html
87	www.sloweurope.com/community/
88	community.ricksteves.com/travel-forum/spain.html
89	hackercombat.com/forum/
90	www.scbwi.org/boards/index.php?board=62.0

Table 13.4: Sources of the Forums / Social benchmarks

DOM nodes of the main content, and column **Links** represents the number of links contained in the main menu. Note that, as stated in previous chapters, the number of template nodes and the number of main content nodes are not necessarily complementary. I.e., usually, we can find DOM nodes in many web pages that do not belong to the template nor to the main content. For instance, web pages typically contain irrelevant (not main content) information that only appears on that web page, such as blocks from social networks, submenus, sliders, etc.

Id	Original URL of the web page
91	www.cocinaconmarta.com/2015/04/empanadillas-chinas-de-gambas-y-verduras.html
92	www.trendencias.com
93	googleblog.blogspot.com.es
94	www.robyncarr.com/qa/
95	users.dsic.upv.es/∼jsilva/wwv2013/index2.html
96	www.folj.com/puzzles/difficult-logic-problems.htm
97	oneminutelist.com/16-browser-alternatives-to-desktop-programs/
98	artsonline.uwaterloo.ca/jburbidg/index.html
99	benjamincongdon.me/blog.html
100	michael.tsikerdekis.com
101	www.beeorganisee.com/reprendre-en-main-le-nettoyage/
102	www.danielgrindrod.com/about.html
103	ofdollarsanddata.com
104	blog.mint.com/updates/enter-our-newdecadenewyou-meme-sweepstakes-for-a-chance-to-win-5000/
105	elainesir.com/best-korean-beauty-blogs-bloggers-follow/
106	www.vindame.com.br/semana-riesling/uva-riesling/
107	www.rosamontero.es/obra-rosa-montero.html
108	www.almezzer.com/libros/literatura-infantil/a-partir-de-4-anos/
109	markahall.blogspot.com.es
110	johnboyne.com/about/
111	users.dsic.upv.es/∼dinsa/en/
112	johngardnerathome.info
113	www.annmalaspina.com
114	foodsense.is/a-list.html
115	sites.google.com/a/ciencias.unam.mx/pagina-ana-meda/
116	whatever.scalzi.com/about/interviews-appearances-articles-and-etc/
117	www.javiercelaya.es
118	diarium.usal.es/lguich/pagina-personal-de-luis-arturo-guichard/
119	www.jameslovelock.org/scientific-papers/
120	www.cipri.info

Table 13.5: Sources of the Personal / Blogs benchmarks

13.4 Guidelines for using the suite

13.4.1 Downloading and configuring the suite

TeCo is distributed free and can be downloaded from the following URL:

http://personales.upv.es/josilga/retrieval/teco/

The suite is distributed in 5 separated zip files, one for each benchmarks category. Each zip file, once downloaded and decompressed, creates 30

Id	Original URL of the web page
121	today.java.net/pub/a/today/2004/07/06/3ddesktop.html
122	clotheshor.se
123	www.raspberrypi.org/resources/teach/
124	doodle.com/online-calendar/
125	www.newprosoft.com/web-content-extractor.htm
126	worryfreelabs.com/about/
127	www.intelligencetest.com
128	www.ikea.com/gb/
129	www.nubbeo.com.ar
130	www.mulberry.com/es/shop/sale/sale-mens-accessories.html
131	www.tous.com/es-es/novedades/relojes/c/59.html
132	preferenceweb.com/collections/all-sneakers.html
133	www.trekbikes.com/us/en_US/bikes/mountain-bikes/electric-mountain-bikes/c/B512/
134	addons.prestashop.com/es/2-modulos.html
135	us.pandora.net/en/charm-bracelets/pandora-moments/pandora-moments-bracelets/
136	kawaiipenshop.com
137	www.vam.ac.uk/shop/lindsay-philip-butterfield-blue-flower-silk-scarf.html
138	shop.fendt.com/kids-toys/clothing/shirts.html
139	www.euroholds.com/it/29-prese-arrampicata.html
140	www.emmaclothes.com
141	www.arduino.cc/en/Main/Software/
142	naranjascarcaixent.com/tienda.html
143	www.technicalbookstoreonline.com/new-arrivals.php
144	www.floridarealestatecollege.com
145	www.basf.com/nl/nl/who-we-are/BASF-in-Nederland.html
146	www.mcphersonoil.com
147	www.thirteenhou.com/menu.php
148	wwww.embalajesterra.com/precintadoras-manuales-168
149	www.crypto.ch/en/about
150	www.shopbookshop.com

Table 13.6: Sources of the Companies / Shops benchmarks

folders, one for each benchmark. The name of each folder corresponds to the domain name of its key page. Additionally, the compressed file also contains a text file in its root folder with some useful information, such as the path to the key page of each benchmark. Table 13.8 includes the path to the key page of each benchmark as well as the zip file that contains that benchmark.

Id	Benchmark's domain	Type	Nodes	T. Nodes	M.C. Nodes	Links
1	web.mit.edu	T	424	252	141	9
2	www.museodelprado.es	T	639	148	168	7
3	www.u-tokyo.ac.jp	T	614	499	97	30
4	www.savethechildren.net	T	763	690	54	21
5	college.harvard.edu	T	1098	669	397	5
6	www.linuxfoundation.org	T	597	534	38	118
7	clinicaltrials.gov	T	545	424	101	37
8	cordis.europa.eu	T	980	335	164	19
9	parents.berkeley.edu	T	287	99	180	8
10	www.mit.edu	T	1290	472	809	13
11	cpoepalencia.es	T	719	644	73	51
12	www.icann.org	T	492	397	90	46
13	www.gip-jci-justice.fr	T	887	680	137	28
14	www.einstein.yu.edu	T	1168	815	187	29
15	www.americanacademy.de	T	746	670	14	37
16	www.mensa.es	T	422	354	37	10
17	www.bcrf.org	T	917	587	294	6
18	www.ielts.org	T	761	605	150	43
19	fr.unesco.org	T	957	615	308	73
20	www.ccbe.eu	T	1003	783	177	33
21	www.fraud.org	T	558	321	61	16
22	www.jdi.org.za	E	661	401	199	10
23	www.premiere-urgence.org	E	502	457	32	30
24	www.indiangaming.org	E	594	209	148	7
25	hispalinux.es	E	515	347	144	32
26	www.gktw.org	E	793	646	130	8
27	www.apnic.net	E	650	461	79	59
28	www.unicef.org	E	1057	671	381	4
29	www.klimabuendnis.org	E	892	536	134	75
30	www.isoc-es.org	E	279	171	56	17
31	edition.cnn.com	T	3980	192	877	15
32	www.neoteo.com	T	1051	636	388	18
33	riotimesonline.com	T	2115	1094	743	23
34	www.turfparadise.com	T	1072	838	205	98
35	www.cleanclothes.org	T	1358	266	932	7
36	www.afp.com	T	1208	404	789	16
37	www.history.com	T	1324	673	260	12
38	detroit.cbslocal.com	T	1261	1004	96	65
39	www.rocklists.com	T	783	533	184	6
40	www.lashorasperdidas.com	T	1924	554	683	12
41	www.journalism.org	T	830	459	86	10
42	www.socialmediatoday.com	T	1288	666	149	8
43	www.diariodeburgos.es	T	606	384	69	9
44	wordofmouthmendo.com	T	916	668	26	26
45	www.usine-digitale.fr	T	994	259	124	18
46	1015fm.com.au	T	1041	835	65	28
47	www.dw.com	T	2593	1596	470	135
48	www.theday.com	T	2147	933	456	86
49	nltimes.nl	T	588	115	164	10
50	www.bbc.co.uk	T	3029	573	1195	22
51	techcrunch.com	T	2612	1891	586	35
52	biztechmagazine.com	E	1950	1057	454	97
53	www.eeo.com.cn	E	936	676	119	11
54	www.wishtv.com	E	2380	1993	343	77
55	news.mit.edu	E	2122	1045	128	8

Id	Benchmark's domain	Type	Nodes	T. Nodes	M.C. Nodes	Links
56	asia.nikkei.com	E	886	671	116	42
57	www.rcnky.com	E	1771	1435	112	17
58	news.discovery.com	E	2926	1209	791	68
59	www.kathimerini.gr	E	1897	1606	117	83
60	news.un.org	E	1809	1258	59	42
61	es.sharelatex.com	T	1100	877	214	6
62	github.com	T	1242	453	783	5
63	en.citizendium.org	T	1092	415	633	35
64	www.filmaffinity.com	T	1337	352	972	32
65	www.meneame.net	T	769	207	423	11
66	www.accountkiller.com	T	510	222	273	8
67	study.com	T	7328	1897	5433	74
68	c.mi.com	T	3506	2949	541	37
69	alumni.harvard.edu	T	2026	1785	219	40
70	www.spacetimestudios.com	T	5049	1387	3500	47
71	www.gimpforum.de	T	2058	457	1300	6
72	www.emaildiscussions.com	T	1129	239	674	7
73	forums.debian.net	T	2766	150	2327	7
74	forums.mozillazine.org	T	2023	235	1411	4
75	forums.tomsguide.com	T	7911	992	5064	22
76	forums.mysql.com	T	4493	430	3950	10
77	lawstudents.ca	T	3563	949	1935	11
78	www.japanesepod101.com	T	1574	924	434	36
79	forum.skyscraperpage.com	T	3410	146	1676	6
80	forums.opera.com	T	1456	617	829	7
81	forums.linuxmint.com	T	5079	327	3872	7
82	frances.forosactivos.net	E	814	318	495	9
83	www.wysiwygwebbuilder.com	E	3941	739	3201	7
84	www.3dprintforums.com	E	1125	312	748	8
85	www.strangehorizons.com	E	643	149	406	23
86	communities.apple.com	E	3144	375	1306	10
87	www.sloweurope.com	E	4208	526	2789	29
88	community.ricksteves.com	E	2061	386	1177	9
89	hackercombat.com	E	1828	828	698	6
90	www.scbwi.org	E	892	219	506	6
91	www.cocinaconmarta.com	T	4154	3404	307	9
92	www.trendencias.com	T	2503	1139	1040	7
93	googleblog.blogspot.com.es	T	5096	3574	1494	343
94	www.robyncarr.com	T	292	92	200	4
95	users.dsic.upv.es	T	207	170	34	14
96	www.folj.com	T	567	176	384	4
97	oneminutelist.com	T	503	276	211	5
98	artsonline.uwaterloo.ca	T	413	164	240	4
99	benjamincongdon.me	T	329	55	274	4
100	michael.tsikerdekis.com	T	577	124	95	7
101	www.beeorganisee.com	T	840	494	303	18
102	www.danielgrindrod.com	T	424	395	29	4
103	ofdollarsanddata.com	T	1075	365	651	5
104	blog.mint.com	T	872	442	129	44
105	elainesir.com	T	1383	523	636	26
106	www.vindame.com.br	T	667	546	104	20
107	www.rosamontero.es	T	808	89	717	9
108	www.almezzer.com	T	1121	516	416	23
109	markahall.blogspot.com.es	T	3144	697	2437	22
110	johnboyne.com	T	690	214	185	8

Id	Benchmark's domain	Type	Nodes	T. Nodes	M.C. Nodes	Links
111	users.dsic.upv.es	T	243	75	160	5
112	johngardnerathome.info	E	397	176	188	21
113	www.annmalaspina.com	E	403	190	84	8
114	foodsense.is	E	339	104	192	5
115	sites.google.com	E	405	320	85	32
116	whatever.scalzi.com	E	1693	1434	243	11
117	www.javiercelaya.es	E	763	682	57	12
118	diarium.usal.es	E	625	82	524	3
119	www.jameslovelock.org	E	679	465	174	20
120	www.cipri.info	E	955	387	556	27
121	today.java.net	T	733	342	354	6
122	clotheshor.se	T	465	232	228	8
123	www.raspberrypi.org	T	398	143	209	14
124	doodle.com	T	580	491	82	5
125	www.newprosoft.com	T	833	151	679	6
126	worryfreelabs.com	T	514	321	190	7
127	www.intelligencetest.com	T	595	323	263	18
128	www.ikea.com	T	1556	407	985	10
129	www.nubbeo.com.ar	T	1642	605	975	7
130	www.mulberry.com	T	8506	3943	4203	152
131	www.tous.com	T	5109	3010	1056	216
132	preferenceweb.com	T	2279	725	1322	17
133	www.trekbikes.com	T	5698	1924	3760	15
134	addons.prestashop.com	T	8062	2333	3382	5
135	us.pandora.net	T	6375	2011	3376	142
136	kawaiipenshop.com	T	1237	821	403	26
137	www.vam.ac.uk	T	1595	1186	392	61
138	shop.fendt.com	T	2278	1433	640	55
139	www.euroholds.com	T	4923	843	3490	89
140	www.emmaclothes.com	T	1088	374	705	8
141	www.arduino.cc	T	854	490	336	26
142	naranjascarcaixent.com	E	321	172	141	6
143	www.technicalbookstoreonline.com	E	2971	391	2002	12
144	www.floridarealestatecollege.com	E	1069	556	65	29
145	www.basf.com	E	845	776	62	12
146	www.mcphersonoil.com	E	891	628	225	34
147	www.thirteenhou.com	E	1226	137	1073	4
148	www.embalajesterra.com	E	2360	1694	470	106
149	www.crypto.ch	E	346	249	68	3
150	www.shopbookshop.com	E	1743	1314	387	9

Table 13.7: Benchmark properties

13.4.2 Rules for using the suite and report

As stated in previous sections of this chapter, TeCo was created to provide researchers with a wide collection of heterogeneous benchmarks useful for several block detection disciplines. When we tried to find collections of benchmarks used in previously published techniques, we encountered several problems. Some of the benchmark suites, such as CleanEval [20] and

Id	Zip file	Path to the key page
1	Institutions.zip	web.mit.edu/institute-events/visitor
2	Institutions.zip	www.museodelprado.es/index.html
3	Institutions.zip	www.u-tokyo.ac.jp/en/about/history.html
4	Institutions.zip	www.savethechildren.net/what-we-do/our-humanitarian-work.html
5	Institutions.zip	college.harvard.edu/financial-aid.html
6	Institutions.zip	www.linuxfoundation.org/about.1.html
7	Institutions.zip	clinicaltrials.gov/ct2/search/index/index.html
8	Institutions.zip	cordis.europa.eu/fp7/ict/fire.html
9	Institutions.zip	parents.berkeley.edu/advice/babies/laundry.html
10	Institutions.zip	www.mit.edu/campus-life.1.html
11	Institutions.zip	cpoepalencia.es/federaciones-y-asociaciones-confederadas-asociaciones/index.html
12	Institutions.zip	www.icann.org/history.html
13	Institutions.zip	www.gip-jci-justice.fr/en/about-us/support-council/index.html
14	Institutions.zip	www.einstein.yu.edu/leadership/index.html
15	Institutions.zip	www.americanacademy.de/about/index.html
16	Institutions.zip	www.mensa.es/cms/pages/¿qué-es-mensa.html
17	Institutions.zip	www.bcrf.org/breast-cancer-research.html
18	Institutions.zip	www.ielts.org/what-is-ielts/ielts-introduction.html
19	Institutions.zip	fr.unesco.org/about-us/introducing-unesco.html
20	Institutions.zip	www.ccbe.eu/about/who-we-are/index.html
21	Institutions.zip	www.fraud.org/get_involved.html
22	Institutions.zip	www.jdi.org.za/index.html
23	Institutions.zip	www.premiere-urgence.org/qui-sommes-nous/index.html
24	Institutions.zip	www.indiangaming.org/index.html
25	Institutions.zip	hispalinux.es/QuienesSomos.html
26	Institutions.zip	www.gktw.org/about/index.html
27	Institutions.zip	www.apnic.net/about-apnic/organization/vision-mission-objectives/index.html
28	Institutions.zip	www.unicef.org/where-we-work.html
29	Institutions.zip	www.klimabuendnis.org/home.html
30	Institutions.zip	www.isoc-es.org
31	Media.zip	edition.cnn.com/index.html
32	Media.zip	www.neoteo.com/star-wars-the-force-awakens-el-regreso-de-viejos-personajes/
33	Media.zip	riotimesonline.com/index.html
34	Media.zip	www.turfparadise.com/index.html
35	Media.zip	www.cleanclothes.org/index.html
36	Media.zip	www.afp.com/es/contact.html
37	Media.zip	www.history.com/index.html
38	Media.zip	detroit.cbslocal.com/2018/12/04/high-school-newspaper-suspended-after-publishing-disruptive-investigation/index.html
39	Media.zip	www.rocklists.com/91x-1983.html
40	Media.zip	www.lashorasperdidas.com/index.html

Id	Zip file	Path to the key page
41	Media.zip	www.journalism.org/2014/03/13/social-search-direct/index.html
42	Media.zip	www.socialmediatoday.com/news/facebook-adds-new-features-for-instant-articles-including-links-to-more-pu/569786/index.html
43	Media.zip	www.diariodeburgos.es/Noticia/Z1C5D6DE9-D1E6-B03A-61236AF21520B8B2/202002/Un-programa-verde-dedicado-a-Felix-Rodriguez-de-la-Fuente.html
44	Media.zip	wordofmouthmendo.com/word-of-mouth-stories/2018/5/31/travellers-fare.html
45	Media.zip	www.usine-digitale.fr/article/la-start-up-americaine-clearview-ai-illustre-deja-les-derives-de-la-reconnaissance-faciale.N921119.html
46	Media.zip	1015fm.com.au/2020/02/steve-mickenbecker-interest-rates-on-hold-2020-02-07/index.html
47	Media.zip	www.dw.com/de/lebron-james-vom-pflegekind-zum-basketball-superstar/a-52088565.html
48	Media.zip	www.theday.com/movies–tv/20200203/super-bowl-ads-dialed-up-fun-as-antidote-to-politics.html
49	Media.zip	nltimes.nl/2019/12/16/chocolate-spread-babies-wins-misleading-product-award.html
50	Media.zip	www.bbc.co.uk/news/index.html
51	Media.zip	techcrunch.com/gadgets
52	Media.zip	biztechmagazine.com/article/2019/12/why-byod-makes-endpoint-security-crucial-small-businesses.html
53	Media.zip	www.eeo.com.cn/2022/0506/533366.shtml.html
54	Media.zip	www.wishtv.com/news/flu-is-widespread-across-the-us/index.html
55	Media.zip	news.mit.edu/2021/grand-decoding-data-0909.html
56	Media.zip	asia.nikkei.com/Spotlight/Sharing-Economy/New-Tokyo-homes-ditch-parking-spaces-but-offer-car-sharing.html
57	Media.zip	www.rcnky.com/articles/2021/09/12/ft-mitchell-reflects-life-age-104.html
58	Media.zip	news.discovery.com/tech/robotics/artificial-intelligences-hawkings-fears-stir-debate-141206.htm
59	Media.zip	www.kathimerini.gr/society/561833251/koronoios-arsi-metron-i-megali-prova-kanonikotitas-enopsei-toy-kalokairioy/index.html
60	Media.zip	news.un.org/en/content/navigate-news.html
61	Forum.zip	es.sharelatex.com/learn/Uploading_a_project
62	Forum.zip	github.com/DawidStankiewicz/forum.1
63	Forum.zip	en.citizendium.org/index.html
64	Forum.zip	www.filmaffinity.com/es/main.html
65	Forum.zip	www.meneame.net/faq-es.html
66	Forum.zip	www.accountkiller.com/en/delete-activision-account.html
67	Forum.zip	study.com/learn/science-questions-and-answers.html
68	Forum.zip	c.mi.com/it/index.html
69	Forum.zip	alumni.harvard.edu/help/message-board.html

Id	**Zip file**	**Path to the key page**
70	Forum.zip	www.spacetimestudios.com/forumdisplay.php?29-Websites-and-Forum-Discussion.html
71	Forum.zip	www.gimpforum.de/index.html
72	Forum.zip	www.emaildiscussions.com/index.html
73	Forum.zip	forums.debian.net/viewforum.php?f=5.html
74	Forum.zip	forums.mozillazine.org/viewforum.php?f=23.html
75	Forum.zip	forums.tomsguide.com/forums/laptop-general-discussion.15/index.html
76	Forum.zip	forums.mysql.com/list.php?21.html
77	Forum.zip	lawstudents.ca/forums.html
78	Forum.zip	www.japanesepod101.com/forum/viewforum.php?f=26.html
79	Forum.zip	forum.skyscraperpage.com/index.html
80	Forum.zip	forums.opera.com/index.html
81	Forum.zip	forums.linuxmint.com/viewforum.php?f=72.html
82	Forum.zip	frances.forosactivos.net/index.html
83	Forum.zip	www.wysiwygwebbuilder.com/forum/viewforum.php?f=10.html
84	Forum.zip	www.3dprintforums.com/index.html
85	Forum.zip	www.strangehorizons.com/2004/20040906/greenglass-f.shtml.html
86	Forum.zip	communities.apple.com/es/community/mac_os/os_x_el_capitan.html
87	Forum.zip	www.sloweurope.com/community/index.html
88	Forum.zip	community.ricksteves.com/travel-forum/spain.html
89	Forum.zip	hackercombat.com/forum/index.html
90	Forum.zip	www.scbwi.org/boards/index.php?board=62.0.html
91	Personal.zip	www.cocinaconmarta.com/2015/04/empanadillas-chinas-de-gambas-y-verduras.html
92	Personal.zip	www.trendencias.com
93	Personal.zip	googleblog.blogspot.com.es
94	Personal.zip	www.robyncarr.com/qa.html
95	Personal.zip	users.dsic.upv.es/∼jsilva/wwv2013/index2.html
96	Personal.zip	www.folj.com/puzzles/difficult-logic-problems.htm
97	Personal.zip	oneminutelist.com/16-browser-alternatives-to-desktop-programs/index.html
98	Personal.zip	artsonline.uwaterloo.ca/jburbidg/index.html
99	Personal.zip	benjamincongdon.me/blog.html
100	Personal.zip	michael.tsikerdekis.com/index.html
101	Personal.zip	www.beeorganisee.com/reprendre-en-main-le-nettoyage/index.html
102	Personal.zip	www.danielgrindrod.com/about.html
103	Personal.zip	ofdollarsanddata.com/index.html
104	Personal.zip	blog.mint.com/updates/enter-our-newdecadenewyou-meme-sweepstakes-for-a-chance-to-win-5000/index.html
105	Personal.zip	elainesir.com/best-korean-beauty-blogs-bloggers-follow/index.html
106	Personal.zip	www.vindame.com.br/semana-riesling/uva-riesling/index.html
107	Personal.zip	www.rosamontero.es/obra-rosa-montero.html

Id	Zip file	Path to the key page
108	Personal.zip	www.almezzer.com/libros/literatura-infantil/a-partir-de-4-anos/index.html
109	Personal.zip	markahall.blogspot.com.es
110	Personal.zip	johnboyne.com/about/index.html
111	Personal.zip	users.dsic.upv.es/∼dinsa/en/index.html
112	Personal.zip	johngardnerathome.info/index.htm
113	Personal.zip	www.annmalaspina.com/index.html
114	Personal.zip	foodsense.is/a-list.html
115	Personal.zip	sites.google.com/a/ciencias.unam.mx/pagina-ana-meda/index.html
116	Personal.zip	whatever.scalzi.com/about/interviews-appearances-articles-and-etc/index.html
117	Personal.zip	www.javiercelaya.es/index.html
118	Personal.zip	diarium.usal.es/lguich/pagina-personal-de-luis-arturo-guichard
119	Personal.zip	www.jameslovelock.org/scientific-papers/index.html
120	Personal.zip	www.cipri.info/index.html
121	Companies.zip	today.java.net/pub/a/today/2004/07/06/3ddesktop.html
122	Companies.zip	clotheshor.se/index.html
123	Companies.zip	www.raspberrypi.org/resources/teach/index.html
124	Companies.zip	doodle.com/online-calendar.html
125	Companies.zip	www.newprosoft.com/web-content-extractor.htm
126	Companies.zip	worryfreelabs.com/about.1.html
127	Companies.zip	www.intelligencetest.com/index.htm
128	Companies.zip	www.ikea.com/gb/en.html
129	Companies.zip	www.nubbeo.com.ar/index.html
130	Companies.zip	www.mulberry.com/es/shop/sale/sale-mens-accessories.html
131	Companies.zip	www.tous.com/es-es/novedades/relojes/c/59.html
132	Companies.zip	preferenceweb.com/collections/all-sneakers.html
133	Companies.zip	www.trekbikes.com/us/en_US/bikes/mountain-bikes/electric-mountain-bikes/c/B512/index.html
134	Companies.zip	addons.prestashop.com/es/2-modulos.html
135	Companies.zip	us.pandora.net/en/charm-bracelets/pandora-moments/pandora-moments-bracelets/index.html
136	Companies.zip	kawaiipenshop.com/index.html
137	Companies.zip	www.vam.ac.uk/shop/lindsay-philip-butterfield-blue-flower-silk-scarf.html
138	Companies.zip	shop.fendt.com/kids-toys/clothing/shirts.html
139	Companies.zip	www.euroholds.com/it/29-prese-arrampicata.html
140	Companies.zip	www.emmaclothes.com/index.html
141	Companies.zip	www.arduino.cc/en/Main/Software.html
142	Companies.zip	naranjascarcaixent.com/tienda.html
143	Companies.zip	www.technicalbookstoreonline.com/new-arrivals.php.html
144	Companies.zip	www.floridarealestatecollege.com/index.html
145	Companies.zip	www.basf.com/nl/nl/who-we-are/BASF-in-Nederland.html
146	Companies.zip	www.mcphersonoil.com/index.html
147	Companies.zip	www.thirteenhou.com/menu.php.html

Id	Zip file	Path to the key page
148	Companies.zip	www.embalajesterra.com/precintadoras-manuales-168.html
149	Companies.zip	www.crypto.ch/en/about.html
150	Companies.zip	www.shopbookshop.com/index.htm

Table 13.8: Path to the key page of each benchmark

L3S-GN1 [61], were only prepared for page-level techniques or were only prepared for content extraction. Others, such as MSS [85], were restricted to only one of the 5 benchmark categories we defined, or their number of different domains was poor. Finally, other benchmark suites were not publicly available due to privacy restrictions, copyright, unavailability, etc. Therefore, we decided to create a publicly available benchmark suite for block detection techniques.

For the reasons given above, we urge all researchers and developers that use TeCo to follow two basic principles:

i. They must publish their results so that they are publicly available.

ii. They must provide enough information so that anyone can easily duplicate their experiments.

13.5 Conclusions

This chapter describes a benchmark suite (TeCo) composed of 150 heterogeneous websites. This benchmark suite can be used to evaluate or test any block detection technique, but it is especially useful for menu detection, content extraction, and template detection because the included gold standard is prepared for them. Specifically, the gold standard identifies the main menu, the template, and the main content of each benchmark. Therefore, it can be used to test, evaluate, and compare techniques and implementations of these block detection disciplines. It is important to note that, unlike other benchmark suites, TeCo can be used by both, page-level and site-level techniques.

The effectiveness of TeCo has been widely demonstrated because it has been used to test and evaluate all the block detection techniques described in this thesis.

Finally, it should be highlighted that TeCo is publicly available and free.

13.6 Contributions

This chapter provides several contributions that can be exploited by researchers in order to train and evaluate their block detection techniques.

The main contribution of the chapter is a benchmark suite labelled for template detection, content extraction, and menu detection. The suite is formed by 150 heterogeneous websites from 5 main categories (Companies / Shops, Forums / Social, Personal websites / Blogs, Media / Communication, and Institutions / Associations.

Another important contribution of the chapter is the information about the creation of the benchmark suite. The chapter details which tool was used and how the set of benchmarks was created. In addition, it describes the benchmark's labelling method.

Chapter 14

Implementation

In this chapter, we describe the current state of the art in the implementation of the different WebExtensions. As commented in several chapters, all the techniques explained in this thesis have been implemented as WebExtensions, and in addition, those WebExtensions have been officially published by Mozilla in their Firefox browser add-ons website.

14.1 WebExtensions' implementation

All the described techniques (MenEx, page-level ConEx, TemEx, site-level ConEx, and Hybrid technique) share most of their features, i.e., they are based on DOM trees, they all have been implemented using JavaScript, they share the same architecture, they are compatible with the same browsers, etc. Hence, all of them have been implemented as WebExtensions using a common architecture.

14.1.1 Architecture

As all the proposed techniques are based on DOM trees, they share the basis of their architecture, which is very similar to the architecture explained in Chapter 12, in the description of the workbench for template detection. Figure 14.1 shows a common architecture scheme for all the techniques' WebExtensions. The diagram outlines the 5 phases into which each technique can be divided. However, all the phases are not required for all the techniques, i.e., the `Detection of candidates` phase only makes sense for site-level techniques, so it is not useful for MenEx and page-level ConEx. This architecture is shared by all the WebExtensions published by Mozilla in their Firefox browser add-ons website, as well as the WebExtensions used for training and evaluating each technique. The main modules of the WebExtensions' architecture are:

- Detection of candidates: The main difference between the architecture of the various implementations is this module. Since the page-level techniques do not need to extract any information from other web pages from the website, it does not make sense to perform the `Detection of candidates` phase. Therefore, this phase is optional depending on the technique to be applied. As previously detailed in Chapter 7, this module is divided into two different stages: (1) detecting the links in the key page that most likely produce a complete subdigraph (hyperlink analysis), and (2) sorting the links detected in the previous phase to choose the links that must be explored first in order to find an optimal CS (CS extraction). Note that, as stated in Section 9.2, not all the links that form a CS produce equally good CS. The input of the `Hyperlink analysis` stage is the key page, while its output is a set of links that can form a CS. This set of links is the input of the `CS extraction` stage, while its output is the CS.

- HTML to DOM: As can be observed in Figure 14.1, this module converts the web pages' HTML code into their corresponding DOM tree. This is a trivial conversion since it is the process that web browsers perform in order to represent an HTML web page. The input of this module is a set of web pages corresponding to the links that form the CS. Once processed, the output of the module is a set of DOM trees corresponding to those web pages. Note that the execution of this module is mandatory since all the presented techniques are based on DOM trees.

- Block detection: The block detection module implements the desired technique (MenEx, page-level ConEx, TemEx, site-level ConEx, and Hybrid algorithm). Depending on the implemented technique and the operation mode (evaluation or test), this module calls the required modules and performs the operation. As described in Section 12.2, other block detection algorithms can be added to this architecture, i.e., SST [120], RTDM-TD [114], IWPTD [116], RBM-TD [113], etc. This module receives as input one or several DOM trees (depending on whether it corresponds to a page-level or a site-level technique), and returns as output the DOM tree corresponding to the detected block.

- Evaluation: As described in Section 12.2, this module allows us to obtain an evaluation (precision, recall, F1, and runtime) of the executed algorithm, since it performs a comparison between the obtained block

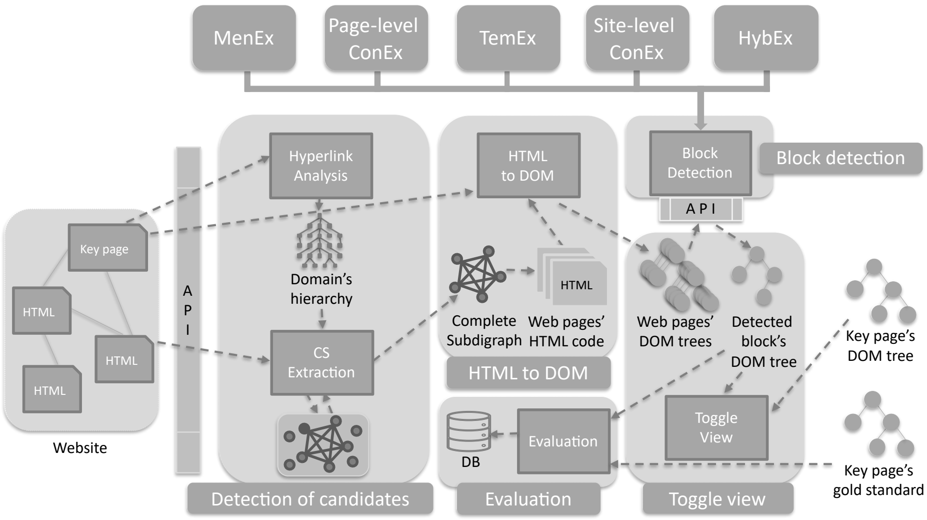


Figure 14.1: Architecture of the WebExtensions

and a previously provided gold standard. This module is completely optional, it makes sense when one is evaluating or testing a technique with a benchmark suite. On the other hand, when one is extracting a block from a single random web page this module makes no sense because the gold standard from that website with high probability is not available. This module receives as input the detected block's DOM tree and the gold standard of the web page. Then, it uses a REST web service to store the evaluation result in a MySQL database.

Figure 14.2 represents the entity-relationship model of the MySQL database. It can be observed that there are 3 entities. An *algorithm* at least executes 1 *experiment*, and a maximum of n *experiments*, while an *experiment* is executed only by one algorithm. Accordingly, an *experiment* uses only 1 *benchmark*, while a *benchmark* could be used by between 0 and n *experiments*. This entity-relationship model has been implemented as a MySQL database which contains 3 tables, one for each of the three entities. However, the REST web service that performs the insertion in the database only needs to insert in the "Experiment" table.

Figure 14.3 shows the scheme of the REST web service. It uses a *PUT* request to send the data to the server.

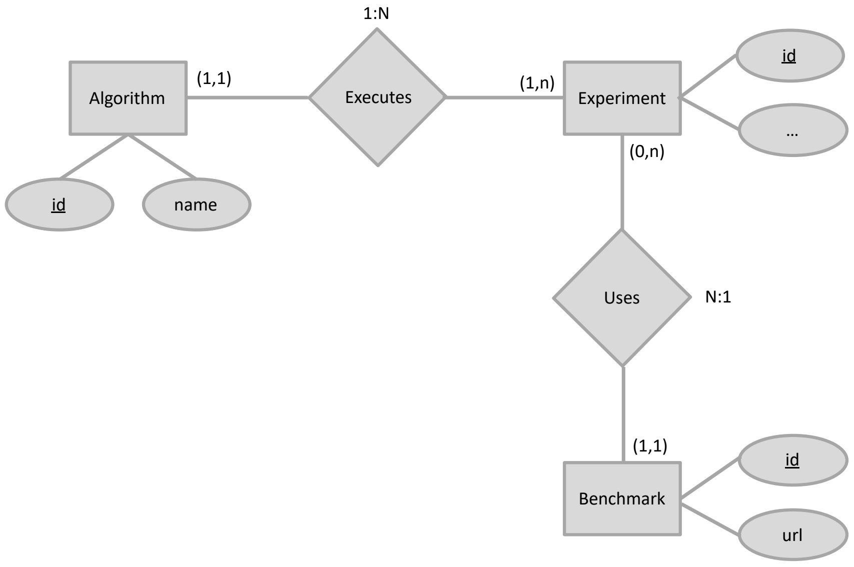


Figure 14.2: Entity-relationship model of the database

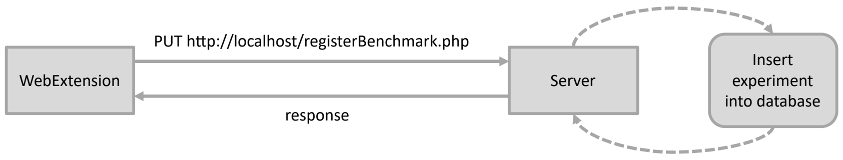

PUT http://localhost/registerBenchmark.php
WebExtension
Server
Insert experiment into database
response

Figure 14.3: REST web service scheme

The process of inserting the experiments into the database is the following:

i. The WebExtension organizes the data in a two-dimensional array with 200 experiments (if possible).

ii. The WebExtension creates an XMLHttpRequest object, opens the URL of the web service, and sends the *PUT* request including a JSON string which corresponds to the array of the experiments stringified[1]. Figure 14.4 show the JavaScript source code of this step. The first line of the code creates a new *XMLHttpRequest* object. Then, in line 2, the *open()* method initializes a newly-created request, or re-initializes an existing one. The

[1]The JSON.stringify() method converts a JavaScript object or value to a JSON string.

HTTP request method is "POST", the URL of the web service is "http://localhost/registerBenchmark.php", and the third parameter indicates that the request is asynchronous. In the third line, the *setRequestHeader()* method sets the header "Content-type" to "application/x-www-form-urlencoded". Finally, the request is sent to the server with the JSON string that corresponds to the stringified array.

```
xmlhttp = new XMLHttpRequest;
xmlhttp.open("POST", "http://localhost/registerBenchmark.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
xmlhttp.send('experiments =' + JSON.stringify(data));
```

Figure 14.4: REST web service request from the WebExtension

iii. The web service decodes the JSON string and converts it to the original two-dimensional array that contained the experiments.

iv. An algorithm explores iteratively the array and inserts the experiments into the database.

It should be highlighted that the data was organized in arrays of 200 experiments because it is more efficient to make requests with blocks of several experiments than one by one. For instance, Chromium-based browsers had memory problems if they request the web service with one experiment at a time despite closing the connection and destroying the object.

- Toggle view: This module is also described in Section 12.2. Contrary to the evaluation module, this module makes sense when one is not testing or evaluating a technique. This module receives as input the key page's DOM tree and the detected block's DOM tree. When one wants to extract a block from a web page, the result of the extraction is shown in the user's browser. The user can use the `Toggle view` function to switch the web browser content between the original web page and the extracted block.

14.1.2 Structure

Browser extensions or add-ons are used to enhance or modify the capability of a web browser. Actually, Mozilla-based extensions are implemented using the WebExtensions API cross-browser technology. This technology

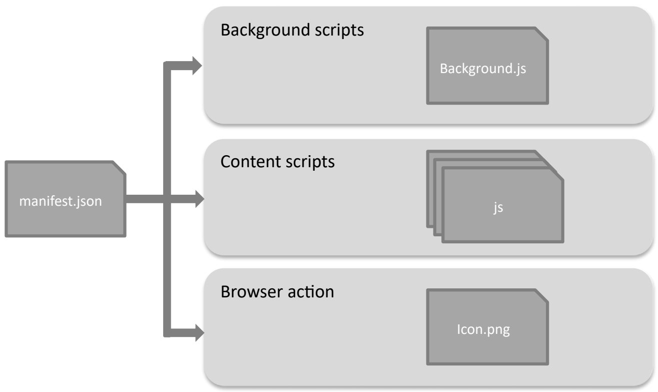


Figure 14.5: Structure of the WebExtensions

is largely compatible with the extension API[2] supported by Chromium-based browsers (such as Google Chrome, Microsoft Edge, Vivaldi, Opera, etc.). In addition, WebExtensions are also compatible with the W3C draft community group report[3].

A WebExtension consists of a set of files, packaged in a particular way for its distribution and installation. Figure 14.5 shows the structure of our WebExtensions, which is an instance of a more complex scheme (the general structure for WebExtensions). It can be observed that a file called "manifest.json" is mandatory. That file must be present in any WebExtension. It contains basic metadata related to the extension, such as its name, version, required permissions, default locale, etc. The file also contains pointers to other files in the WebExtension:

- Background scripts: Usually, WebExtensions need to maintain a long-term state or execute long-term operations regardless of the lifetime of any particular loaded web page or browser window. That long-running logic is defined in background scripts, which are loaded as soon as the extension is loaded and they remain loaded until the extension is disabled or uninstalled. In the case of our WebExtensions, this section of the "manifest.json" file contains a pointer to the background script, called "background.js", which includes functions such as the initial listener that starts to work when the user press the extension's button, error treatment functions, the operations that have to be done once the block is extracted, etc.

[2]https://developer.chrome.com/docs/extensions/reference/
[3]https://browserext.github.io/browserext/

- Content scripts: This section from the "manifest.json" file contains pointers to the scripts that contain the WebExtension operation. Those scripts can access and manipulate web pages because they are loaded into the web pages and they run in the context of them. In the case of our block detection WebExtensions, this section contains pointers to all the JavaScript files needed to implement the modules described in the previous subsection and Figure 14.1. In addition to the JavaScript classes corresponding to the techniques (block detection module of Figure 14.1), it includes many other JavaScript classes, such as the classes that implement all the operations of the rest of the modules in Figure 14.1, classes for time measuring, hash table implementations, etc.

- Browser action: This part from the "manifest.json" file includes pointers to the icons used by the WebExtension. In the case of our WebExtensions, it points to the "png" file that contains both icons, the primary icon and the "toggle view" icon.

14.1.3 Evaluation environment

In previous chapters, mainly in Chapters 5, 8, and 10, we conducted millions of experiments to evaluate the algorithms. Those experiments were performed using the architecture described in Section 14.1.1.

To execute the WebExtensions that conducted the experiments, we prepared a dedicated server in the following way:

- We hired a dedicated server in a data centre. The server was equipped with an 8-core Intel i9 9900k processor and 64 Gb of DDR4 RAM. It was running Ubuntu Desktop[4]. As the testing phase of all algorithms was going to take several months, we chose a server in an external data centre in order to maximize the availability.

- Then, we connected the server using SSH and installed MySQL and Apache. Once installed, we also installed phpMyAdmin to handle the administration of the database over the Web.

- Subsequently, we also installed Anydesk[5] to remotely access the operating system GUI.

[4]https://ubuntu.com
[5]https://anydesk.com

- Finally, we installed the web browsers that executed the WebExtensions. As the server had an 8-core processor, we decided to execute 8 instances of the WebExtension at the same time. Therefore, we installed 8 Chromium-based browsers. The installed web browsers were: Chromium[6], Google Chrome[7], Brave[8], Opera[9], Microsoft Edge[10], Vivaldi[11], Slimjet[12], and SRWare Iron[13].

Once the environment was ready, we connected the server via Anydesk. Then, we uploaded and executed the different WebExtensions. We could follow the execution by connecting to the server at any time (via Anydesk or by browsing the database with phpMyAdmin). Once the experiments of an algorithm were finished, we executed the necessary SQL sentences to obtain the results.

More than 11.5 million experiments were conducted to evaluate the algorithms. The computing time was approximately 227 days using an Intel i9 9900k.

14.2 Usage scenario

This section describes a typical usage scenario of any of the WebExtensions, since all of them are used likewise. A scenario in which the user decides to extract a block from a web page is described.

Step 1: The first step is to load into the web browser the web page from which the user wants to extract the desired block (i.e., the main menu, the main content, or the template). Figure 14.6 shows the main web page of the VRAIN's website loaded into Mozilla Firefox.

Step 2: Once the web page has been loaded, the user has to press the button located on the upper right of the browser window. The button is different for each WebExtension. Concretely, it is a "T" letter in a square for the TemEx WebExtension, a "M" letter for the MenEx Webextension, a "C" letter for both ConEx WebExtensions, and a

[6]https://www.chromium.org
[7]https://www.google.com/intl/es˙es/chrome
[8]https://brave.com
[9]https://www.opera.com
[10]https://www.microsoft.com/es-es/edge
[11]https://vivaldi.com
[12]https://www.slimjet.com
[13]https://www.srware.net/iron

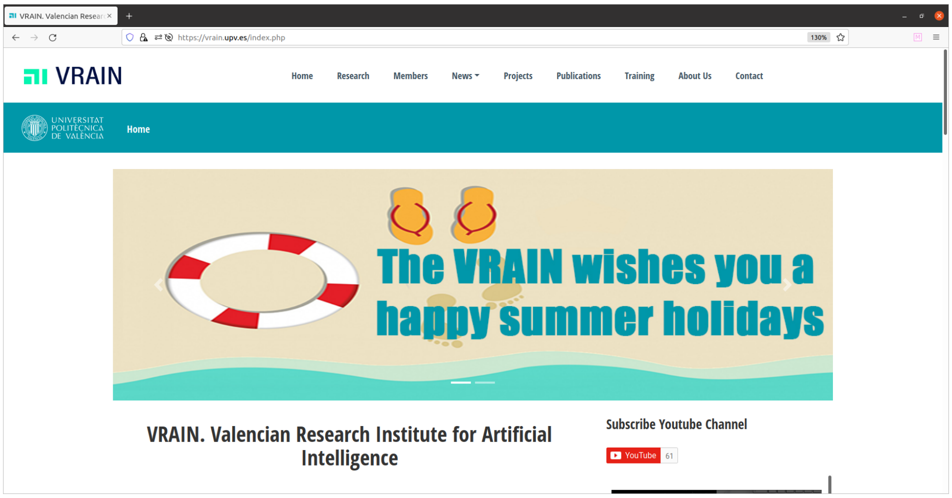

VRAIN. Valencian Resear
https://vrain.upv.es/index.php
VRAIN
Home
Research
Members
News
Projects
Publications
Training
About Us
Contact
UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Home
The VRAIN wishes you a happy summer holidays
VRAIN. Valencian Research Institute for Artificial Intelligence
Subscribe Youtube Channel
YouTube 61

Figure 14.6: VRAIN's web page loaded into the browser

"H" letter for the Hybrid technique WebExtension. For instance, Figure 14.7 shows the MenEx button.

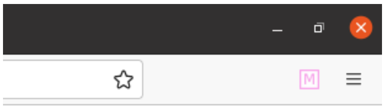


Figure 14.7: MenEx button

Step 3: After the user has pressed the button, the WebExtension performs all the required operations to extract the desired block and shows it in the browser window. The nodes that do not belong to the desired block are properly hidden by changing their *visibility* and *display* attributes to *hidden* or *none*, respectively. Hence, the result is the isolation of the block, which appears in the same place as on the original web page. Figure 14.8 shows the browser window after extracting the main menu of VRAIN's website[14] with MenEx.

Step 4: As stated in Section 14.1, once the block has been extracted, the user can switch the web browser between the original web page and the extracted block. This can be done using the button shown in Figure 14.9.

[14]https://vrain.upv.es/

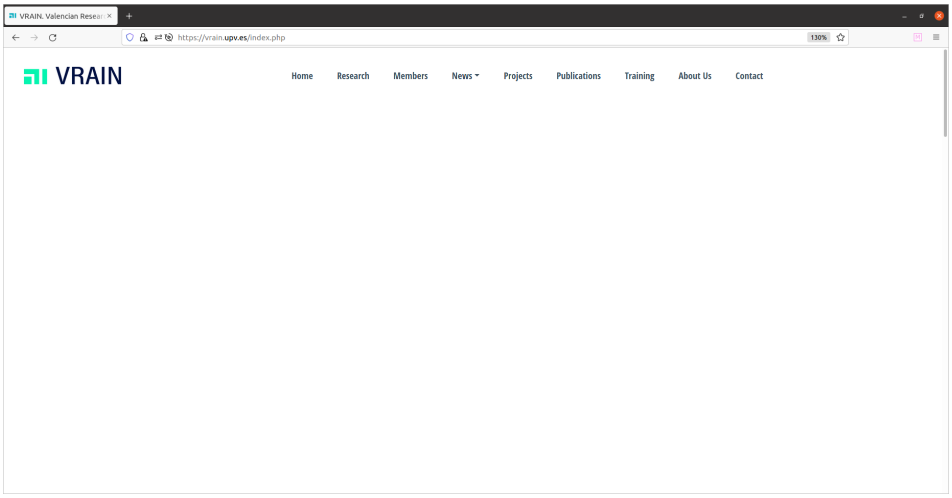


Figure 14.8: VRAIN's web page main menu extracted with MenEx

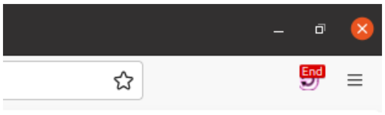


Figure 14.9: Toggle view button

14.3 Tools information

As stated previously in this chapter, we have implemented 5 different WebExtensions, one for each technique: MenEx, page-level ConEx, TemEx, site-level ConEx, and HybEx. Each WebExtension contains between 2500 and 4200 LOC and they are fully implemented in JavaScript. As they are implemented using the WebExtensions API cross-browser technology, they are compatible with Mozilla-based, Chromium-based browsers, and other browsers compatible with the W3C draft community group report. The last release of all the WebExtensions is distributed in English, Spanish, French, and German.

We also implemented the template detection workbench described in Section 12.2. The WebExtension includes the 5 template detection algorithms described in Section 12.1.

It should be highlighted that all the functionalities described in this thesis are completely implemented in the last releases of the WebExtensions.

These tools and the template detection workbench described in Chapter 12 are open and publicly available at:

`http://personales.upv.es/josilga/retrieval/`

14.3.1 Differences between different browsers

Despite we could think that a WebExtension works in exactly the same manner in all browsers, this is not always true. We observed that there are significant differences in two aspects:

- The number of DOM nodes created by the HTML to DOM module. The DOM trees created by the Chromium-based browsers and the Mozilla-based browsers are slightly different because the number of nodes they use to represent the HTML web page is different.

			DOM nodes			**Words**		
Browser	Kp nodes	Mc nodes	Recall	Precision	F1	Recall	Precision	F1
Chromium-based	1613.65	714.71	90.29 %	77.01 %	79.46 %	93.16 %	86.13 %	88.12 %
Mozilla-based	1613.37	714.78	90.29 %	77.01 %	79.46 %	93.16 %	86.13 %	88.12 %

Table 14.1: Results of Mozilla-based and Chromium-based browsers

We performed a comparison experiment using the 150 benchmarks of the TeCo benchmark suite (see Chapter 13). The result of the experiment can be observed in Table 14.1, which shows a comparison between Chromium-based browsers and Mozilla-based browsers for the site-level ConEx algorithm in Chapter 10. Column `Browser` shows the browser (Chromium-based or Mozilla-based); column `Kp nodes` corresponds to the nodes of the key page computed by the browser; column `Mc nodes` shows the number of nodes of the main content computed by the browser. the rest of the columns correspond to the `Recall`, `Precision`, and `F1` for both, retrieved DOM nodes and retrieved text words. As Table 14.1 shows, Chromium-based browsers created more key page's DOM nodes on average, while Mozilla-based browsers created more main content DOM nodes on average. With respect to the `Recall`, `Precision`, and `F1`, we can observe that both algorithms obtain the same values. However, if we use 4 decimal places or more, the `Recall`, `Precision`, and `F1` values for retrieved DOM nodes are slightly different due to the fact that browsers use a different number of DOM nodes to represent the DOM tree. This phenomenon does not occur for retrieved text words because the obtained text is the same in both browser types.

- The runtime of the algorithms is also significantly different. Some techniques are more than two times faster when executed in a Chromium-based browser than in a Mozilla-based browser.

Browser	TemEx	SL ConEx	PL ConEx	HybEx	MenEx
Chromium-based	1326 ms.	909 ms.	289 ms.	1406 ms.	784 ms.
Mozilla-based	2973 ms.	2392 ms.	3473 ms.	2952 ms.	1751 ms.

Table 14.2: Runtimes of Mozilla-based and Chromium-based browsers

When comparing the amount of DOM nodes created by Chromium-browsers and Mozilla-browsers, we performed another comparison experiment using the 150 benchmarks of the TeCo benchmark suite (see Chapter 13) to check the runtime of the 5 block detection techniques described in this thesis on different browsers. Table 14.2 shows the results of the comparison. Each column shows, for both browser types, the average runtimes of each technique for the 150 benchmarks. We can observe that, in general, the runtime of Chromium-based browsers is two times faster than the runtime of Mozilla-based browsers. However, it should be noted that the runtime of page-level ConEx is about 12 times faster for Chromium-based browsers.

14.4 Conclusions

This chapter describes the implementation and usage scenario of the WebExtensions created to evaluate and test the different algorithms presented in this thesis.

The main functional blocks of the architecture are described, relating them to the algorithms detailed in previous chapters. In particular, the chapter emphasizes the evaluation module developed to train the different algorithms. For the evaluation phase of the algorithms, more than 11 million experiments were conducted. We also developed a web service to store the obtained results in a MySQL database. In addition, the evaluation environment created to perform the training experiments is also described.

The chapter also defines the structure of the WebExtensions, the technology used to develop them, and their compatibility with different browsers.

Finally, the chapter shows a usage scenario of one WebExtension and presents the differences observed running the WebExtensions in different browsers.

14.5 Contributions

This chapter provides several contributions that can be exploited by researchers in order to implement or evaluate their block detection techniques.

The chapter introduces an architecture proposal for DOM-based block detection WebExtensions. It is divided into several phases and it can be adapted to any block detection algorithm. It also includes an `Evaluation` phase to train the techniques which store the experiment results in a MySQL database using a REST web service.

Another contribution of the chapter is the description of the structure of the developed WebExtensions. It also outlines the environment created to test the techniques and the statistics about the performed experiments.

The chapter also describes a usage scenario which is representative for any of the WebExtensions in the thesis.

Part VII

Conclusions and Future Work

Chapter 15

Conclusions

Block detection techniques are a Web Mining discipline used to extract information from web pages. Depending on the purpose, there are several different blocks of information that we can extract, e.g., the template, the main content, the menu, comments, etc. Given that the web pages on the World Wide Web are extremely heterogeneous, despite the existence of several content management systems (CMS) used by many websites, the task of extracting information pagelets (main content, menu, template, etc.) is a challenging task. These Web Mining techniques are useful for humans and for many different systems.

The most important block of information in a web page is the main content because it contains really relevant information. However, the main content is almost always next to other elements such as banners, footers, main menus, advertisements, etc. The task of extracting the main content from a web page consists in isolating the useful information and removing those elements that do not contain useful knowledge for the user. However, it should be highlighted the importance of extracting other blocks that do not contain relevant information, such as the template of the web page or the main menu.

On the one hand, block detection techniques provide interesting benefits for humans. Block detection techniques provide numerous advantages to people with disabilities. For instance, visually impaired people can take benefit from menu detection techniques or content extraction techniques since web pages are simplified and they can have a better experience surfing the Web. In addition, among other things, template detection techniques can help developers and designers reuse web templates or check easily the template differences among several web pages.

On the other hand, block detection techniques are also important for many systems. Today, the terms in a web page are treated differently by crawlers and indexers depending on their functionality. Usually, they preprocess the web pages in order to identify the template or the main content. This is due to the fact that template extraction allows crawlers and indexers the identification of those pagelets that only include noisy information such as banners and advertisements. Likewise, the isolation of the main content helps indexers and crawlers to focus on the most relevant information. The main purpose of indexers and crawlers is to provide users with only relevant information. Therefore, extracting the main content is an essential task to preprocess that information. Gibson et al. [44] determined that template elements represent between 40% and 50% of all the data on the Web. This fact justifies the importance of using techniques such as main content extraction, or template extraction [120, 114] as a preprocessing method. Hence, the irrelevant content should not be indexed similarly to the relevant content, because the indexation of the non-content part of templates usually affects accuracy and performance; and it also can lead to a waste of time, bandwidth, storage space, etc. In addition, for crawlers and indexers, identifying the main menu plays an important role in inferring the structure of the website. The basis of the structure of the website can be inferred from the main menu since it always contains hyperlinks to the most important web pages of the website.

This thesis is focused on HTML-structured web pages, which nowadays are by far the great majority. From an engineering point of view, a web page is a set of Document Object Model (DOM) nodes [29]. All the techniques described in this thesis use the representation of a web page as a DOM tree. The use of DOM trees has many advantages for block detection techniques. For instance, as the main content is not only formed by text, it allows content extraction techniques to extract not only text content but also other information such as images, videos, animations, etc. Another important benefit is that the representation of web pages as DOM trees allows us to easily carry out operations with DOM nodes, e.g., insert, delete, traverse, search, etc. Besides, the representation of web pages as DOM trees is a significant advantage for the implementation of block detection techniques as browser extensions, because the extracted blocks are displayed on the users' browser keeping their format and style. However, not all block detection techniques internally represent web pages as their corresponding DOM trees. In fact, researchers use three main different approaches to solve the problem:

i. Using the textual information of the web page (i.e., the HTML code). These techniques are based on the idea that the main content on a web page has more density of text while it has fewer labels [38, 117, 62, 60].

ii. Using the rendered image of the web page in the browser. The main idea of these techniques is that the main content of a web page is typically located in the central part of the screen and it is often visible without scrolling [22].

iii. Using the DOM tree of the web page. Most of the current block detection techniques are based on the representation of the web pages as DOM trees [19, 120, 114, 10]. As pointed out above, this approach provides several benefits to the techniques.

Block detection techniques can be also classified depending on the number of web pages they can access:

- *Page-level techniques.* They only use the information contained on the target web page. These techniques are usually faster than the site-level block detection techniques. The menu detection technique described in Chapter 5, and the content extraction technique described in Chapter 6 are examples of page-level block detection techniques.

- *Site-level techniques.* They use the information contained on several web pages (often from the same website). These techniques compare the information extracted from the set of web pages in order to infer the desired block. The template detection techniques in Chapters 9 and 11, and the content extraction technique in Chapter 10 are examples of site-level block detection techniques.

On the one hand, the main advantage of page-level techniques against site-level techniques is that page-level techniques are faster because they only have to load and analyze one web page, whereas site-level techniques have to load and analyze more web pages to extract the desired information. On the other hand, site-level techniques are usually more accurate because they can obtain more information due to the fact that they load and compare several different web pages from the website.

In this thesis, we have developed some block detection techniques for template detection, content extraction, and menu detection; with the aim of producing usable implementations of them. We also compared these techniques with several well-known techniques in the state-of-the-art. In addition, we built a template detection workbench, and a set of benchmarks to test and evaluate any technique. The main contributions of this thesis have been summarized below:

i. **Block detection techniques.**
Since the early 2000s, many block detection techniques have been proposed. The main purposes of almost all the techniques have been template detection and content extraction. Researchers have developed their techniques in many different ways. Some techniques have been designed as page-level, others as site-level, some of them represent web pages as DOM trees, others are based on textual information, etc. Our site-level block detection techniques (site-level template detection in Chapter 9, and site-level content extraction in Chapter 10), are based on two key ideas: the construction of a complete subdigraph (CS), and the use of an equal top-down mapping of the DOM trees for their comparison. The combination of these two key ideas is the basis of both site-level block detection algorithms. Regarding our page-level block detection techniques (page-level menu detection in Chapter 5, and page-level content extraction in Chapter 6), despite they are rather different, both techniques are based on the ponderation of several properties related to the DOM nodes. In addition, our hybrid template detection technique (see Chapter 11) was developed by combining page-level content extraction with site-level template detection techniques.

ii. **Comparison of techniques.**
The comparison of block detection techniques is an important issue, not only to check how good are our techniques but also to help researchers choose among several techniques. We reviewed the state-of-the-art to find fair block detection technique comparisons. Unfortunately, for template detection, we did not find any valid comparison, and it was impossible to compare several techniques with the results published by the authors. First, each technique was implemented using a different technology which affects the efficiency. Then, each author evaluated her technique using different evaluation criteria, such as counting retrieved words [114, 113], characters [61], DOM nodes [10], text blocks [101, 115], etc. In addition, not all techniques used the same set of benchmarks. We decided to perform a systematic review to select several template detection techniques and, thus, implement the selected techniques from scratch. Once implemented, we compared them with our site-level template detection and hybrid template detection techniques. For content extraction we found two interesting comparisons [106, 115], despite the compared algorithms being only prepared for extracting text. Hence, we eval-

uated our page-level content extraction algorithm and compared it to the algorithms evaluated in [106, 115] using the same benchmarks and metrics.

iii. **Benchmark suite.**
When we tried to compare our block detection techniques with those in the state-of-the-art, we found that, in most cases, it was not possible because using different sets of benchmarks to compare different techniques is highly inaccurate. Regarding the template detection techniques, we found in the literature, some of them used artificial benchmarks [25], while others used real heterogeneous web pages [114, 10]. Similarly, some authors selected the input web pages randomly [120, 116, 50], while others provided the input web pages manually [114, 113]. Finally, regarding the block detection techniques, we found authors that used well-known benchmark suites such as CleanEval [20] benchmark suite [115, 99], MSS [85], L3S-GN1 [61], etc. However, most of these benchmark suites are more than 15 years old, and we consider they have a lack of heterogeneity. To solve these problems, we built our TeCo suite of benchmarks (see Chapter 13). TeCo is formed of 150 heterogeneous benchmarks and it is prepared for several block detection disciplines, such as template detection, content extraction, and menu detection. TeCo is an important resource for researchers to evaluate, test, and compare their block detection techniques.

iv. **Implementations.**
We had to implement several template detection algorithms from scratch. In order to compare them with our template detection techniques, we developed a workbench for template detection. This workbench can integrate any template detection technique based on DOM. Hence, it allows us to easily test, evaluate, and compare different template detection techniques. In addition to the template detection workbench, as stated in previous chapters, all the techniques developed in this thesis have been implemented as WebExtensions. Moreover, they have been published by Mozilla in their Firefox browser add-ons website. These WebExtensions are completely integrated into the user's browser, so when the user presses the extension's button, the algorithm extracts the corresponding block. It should be noted that all these implementations are free and available for download.

As a result of this thesis, researchers can access several block detection techniques in order to enhance their own techniques by reusing some of our algorithms, definitions, metrics, or ideas. Moreover, a fair comparison of both, template detection and content extraction techniques with several important techniques in the state-of-the-art is provided. Researchers are also able to test, evaluate, and compare their block detection techniques using a heterogeneous set of benchmarks formed by 150 real websites. Finally, the implementation of all the developed techniques and the template detection workbench are publicly available for download. More than 11 million experiments were needed to train the algorithms, therefore we hired a dedicated server in a data centre. The server was running Ubuntu Desktop, and was equipped with an 8-core Intel i9 9900k processor and 64 Gb of DDR4 RAM. We executed 8 instances of the algorithms simultaneously using 8 different web browsers. The computing time was approximately 227 days using an Intel i9 9900k.

Chapter 16

Open Lines of Research

There exist several aspects in which *Block Detection* can be improved. We list some of them in the following points:

Techniques The *Menu Detection* technique shown in Chapter 5 assigns a weight to each node depending on the value of some node properties. However, the distinction between two DOM nodes cannot be computed by comparing the sum of their respective ratios because different combinations produce the same value for the sum. This algorithm can be enhanced with a mechanism to distinguish between any combination of ratios, which can be achieved by using the *Euclidean Distance* as in the *Page-level Template Detection* technique described in Chapter 6. Concerning the *equal top-down mapping* used by all the site-level block detection techniques in the thesis, the evaluation of the position of two nodes represents almost all the runtime of the mapping. Despite the *Runtime improvement* algorithm described in sections 9.3 and 10.3 significantly reduces the runtime of the mapping with a minimum impact on the evaluation results, it is necessary to develop a DOM tree transformation algorithm that does not affect the evaluation results of the algorithm. Regarding the *Template Detection* algorithms implemented from scratch (see Chapter 12), to obtain the fairest and most reliable comparison possible, runtime optimization processes are necessary. TemEx (see Chapter 9) has been optimized for years to obtain the lowest runtime possible. However, to be fairly comparable with TemEx, the rest of the *Template Detection* algorithms should be also optimized. We are also working on a page-level *Template Detection* algorithm based on the page-level ConEx algorithm described in Chapter 6. As it is a page-level algorithm, we expect that it will significantly reduce the runtime of TemEx. Moreover, we plan to develop algorithms for other *Block Detection* disciplines, such as *Comments Extraction* and *Advertisement*

Detection. On the one hand, *Comments Extraction* techniques are very valuable for webshops and manufacturers since they allow them to collect information about their products from several websites. On the other hand, *Advertisement Detection* techniques are interesting for users when they surf the Internet as the advertisements are sometimes excessive.

Workbench The *Template Detection Workbench* presented in Section 12.2 implements several well-known *Template Detection* techniques. Moreover, the process to integrate another *Template Detection* technique is easy because it provides several modules useful for *Template Detection*, such as *Detection of Candidates*, *HTML to DOM*, *Evaluation*, and *Toggle View*. However, to provide more possibilities to researchers, it must be updated adding the possibility of integrating other *Block Detection* techniques, such as *Content Extraction* and *Menu Detection* techniques.

Benchmark suite The *TeCo Benchmark Suite* described in Chapter 13 is formed by 150 heterogeneous benchmarks, and it is ready for *Template Detection*, *Content Extraction*, and *Menu Detection*. The first TeCo version (1.0) was formed of 30 benchmarks, while the current version (5.0) is formed by 150 benchmarks. Moreover, the first TeCo version was only prepared for *Template Detection*. For future versions, we plan to increase the number of benchmarks and to include other *Block Detection* disciplines, e.g., *Advertisement Detection* and *Comments Extraction*. In addition, *TeCo Benchmark Suite* should be updated by removing the older websites and adding more up-to-date ones. This will make *TeCo* more reliable since it will represent the Web more accurately.

Implementations All the *Block Detection* techniques in the thesis have been implemented as WebExtensions and are publicly available for download. Moreover, they have been officially published by Mozilla in their Firefox browser add-ons website. However, it would be interesting to integrate all the techniques in a unique WebExtension. The user just might have to choose the desired technique and press the "Extract" button. Then, the extracted block would appear on the browser's screen and the rest of the web page would be hidden. This integration would allow users to easily check different techniques on the same website and compare the obtained results. Moreover, a unique WebExtension is easier to maintain. Additionally, our future

WebExtensions will be based on Manifest v3[1]. In fact, they are Manifest v3 ready but, at the time of writing these lines, Mozilla does not support it in Firefox.

[1]Manifest v3 is the new API developed by Google for Chromium-based browsers. It establishes how WebExtensions interact with the browser.

Bibliography

[1] George Adam, Christos Bouras, and Vassilis Poulopoulos. “CUTER: An Efficient Useful Text Extraction Mechanism”. In: *2009 International Conference on Advanced Information Networking and Applications Workshops*. 2009, pp. 703–708. DOI: 10.1109/WAINA.2009.60.

[2] Julián Alarte, David Insa, and Josep Silva. “Page-Level Webpage Menu Detection”. In: *XVI Jornadas sobre Programación y Lenguajes (PROLE 2016)*. 2016, pp. 134–147.

[3] Julián Alarte, David Insa, and Josep Silva. “Webpage Menu Detection Based on DOM”. In: *SOFSEM 2017: Theory and Practice of Computer Science - 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Limerick, Ireland, January 16-20, 2017, Proceedings*. 2017, pp. 411–422. DOI: 10.1007/978-3-319-51963-0_32.

[4] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. “Automatic detection of webpages that share the same web template”. In: *Electronic Proceedings in Theoretical Computer Science* 163 (2014), pp. 2–15.

[5] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. “Automatic Detection of Webpages that Share the Same Web Template”. In: *Proceedings of the 10th International Workshop on Automated Specification and Verification of Web Systems (WWV 14)*. Ed. by Maurice H. ter Beek and António Ravara. Vol. 163. Electronic Proceedings in Theoretical Computer Science. Vienna (Austria): Open Publishing Association, 2014, pp. 2–15. DOI: 10.4204/EPTCS.163.2.

[6] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. “Site-Level Template Extraction Based on Hyperlink Analysis”. In: *XIV*

Jornadas sobre Programación y Lenguajes (PROLE 2014). 2014, pp. 23–35.

[7] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. "A Collection of Website Benchmarks Labelled for Template Detection and Content Extraction". In: *XV Jornadas sobre Programación y Lenguajes (PROLE 2015)*. 2015. DOI: `11705/PROLE/2015/009`.

[8] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. "TeMex: The Web Template Extractor". In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15 Companion. Florence, Italy: ACM, 2015, pp. 155–158. ISBN: 978-1-4503-3473-0. DOI: `10.1145/2740908.2742835`.

[9] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. "Web template extraction based on hyperlink analysis". In: *Electronic Proceedings in Theoretical Computer Science* 173 (2015), pp. 16–26.

[10] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. "Site-Level Web Template Extraction Based on DOM Analysis". In: *Perspectives of System Informatics*. Ed. by Manuel Mazzara and Andrei Voronkov. PSI 2015. Cham: Springer International Publishing, 2016, pp. 36–49. ISBN: 978-3-319-41579-6. DOI: `10.1007/978-3-319-41579-6_4`.

[11] Julián Alarte and Josep Silva. "Page-Level Main Content Extraction From Heterogeneous Webpages". In: *ACM Transactions on Knowledge Discovery from Data* 15.6 (2021). ISSN: 1556-4681. DOI: `10.1145/3451168`.

[12] Julián Alarte and Josep Silva. "HybEx: A Hybrid Tool for Template Extraction". In: *Companion Proceedings of the Web Conference 2022*. WWW '22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, 205–209. ISBN: 9781450391306. DOI: `10.1145/3487553.3524242`.

[13] Julián Alarte, Josep Silva, and Salvador Tamarit. "What Web Template Extractor Should I Use? A Benchmarking and Comparison for Five Template Extractors". In: *ACM Transactions on the Web* 13.2 (Mar. 2019). ISSN: 1559-1131. DOI: `10.1145/3316810`.

[14] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. "Template Extraction Based on Menu Information". In: *Proceedings of the 9th International Workshop on Automated Specification and Verification of Web Systems (WWV 13)*. Ed. by Josep Silva and António

Ravara. Electronic Proceedings in Theoretical Computer Science. Firenze (Italy): Open Publishing Association, 2013, pp. 71–80. DOI: 10.4204/EPTCS.123.

[15] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. "Main Content Extraction from Heterogeneous Webpages". In: *Web Information Systems Engineering – WISE 2018*. Ed. by Hakim Hacid, Wojciech Cellary, Hua Wang, Hye-Young Paik, and Rui Zhou. Cham: Springer International Publishing, 2018, pp. 393–407. ISBN: 978-3-030-02922-7. DOI: 10.1007/978-3-030-02922-7_27.

[16] Derar Alassi and Reda Alhajj. "Effectiveness of template detection on noise reduction and websites summarization". In: *Information Sciences* 219 (2013), pp. 41 –72. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2012.07.022.

[17] Mohsen Asfia, Mir Mohsen Pedram, and Amir Masoud Rahmani. "Main content extraction from detailed web pages". In: *International Journal of Computer Applications* 4.11 (2010), pp. 18–21. DOI: 10.5120/869-1219.

[18] Naseer Aslam, Bilal Tahir, Hafiz Muhammad Shafiq, and Muhammad Amir Mehmood. "Web-AM: An efficient boilerplate removal algorithm for Web articles". In: *2019 International Conference on Frontiers of Information Technology (FIT)*. IEEE. 2019, pp. 287–2875. DOI: 10.1109/FIT47737.2019.00061.

[19] 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)*. Honolulu, Hawaii, USA: ACM, 2002, pp. 580–591. ISBN: 1-58113-449-5. DOI: 10.1145/511446.511522.

[20] 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)*. Marrakech, Morocco: European Language Resources Association, 2008, pp. 638–643.

[21] Jan Berg. *Improve content extraction in web pages for browser reader modes*. English. Bachelor Thesis: University of Stuttgart, Institute of Architecture of Application Systems. Bachelor Thesis. 2020. DOI: 10.18419/opus-11033.

[22] 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)*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 67–72. ISBN: 978-0-7695-3580-7. DOI: 10.1109/ACIIDS.2009.71.

[23] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. “Extracting Content Structure for Web Pages Based on Visual Representation”. In: *Web Technologies and Applications*. Ed. by Xiaofang Zhou, Maria E. Orlowska, and Yanchun Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 406–417. ISBN: 978-3-540-36901-1. DOI: 10.1007/3-540-36901-5_42.

[24] 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)*. Mountain View, California, USA: ACM, 2011, pp. 121–128. ISBN: 978-1-4503-0863-2. DOI: 10.1145/2034691.2034720.

[25] Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. “Page-level Template Detection via Isotonic Smoothing”. In: *Proceedings of the 16th International Conference on World Wide Web*. WWW ’07. Banff, Alberta, Canada: ACM, 2007, pp. 61–70. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242582.

[26] 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)*. Hong Kong, Hong Kong: ACM, 2001, pp. 211–220. ISBN: 1-58113-348-0. DOI: 10.1145/371920.372054.

[27] Liang Chen, Shaozhi Ye, and Xing Li. “Template Detection for Large Scale Search Engines”. In: *Proceedings of the 2006 ACM Symposium on Applied Computing*. SAC ’06. Dijon, France: ACM, 2006, pp. 1094–1098. ISBN: 1-59593-108-2. DOI: 10.1145/1141277.1141534.

[28] Yu Chen, Wei-Ying Ma, and Hong-Jiang Zhang. “Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices”. In: *Proceedings of the 12th International Conference on World Wide Web*. WWW ’03. Budapest, Hungary: Association for Computing Machinery, 2003, 225–233. ISBN: 1581136803. DOI: 10.1145/775152.775184.

[29] W3C Consortium. *Document Object Model (DOM).* Available from URL: `https://dom.spec.whatwg.org/`. 2019.

[30] Sandip Debnath, Prasenjit Mitra, and C. Lee Giles. “Automatic Extraction of Informative Blocks from Webpages”. In: *Proceedings of the 2005 ACM Symposium on Applied Computing.* SAC ’05. Santa Fe, New Mexico: Association for Computing Machinery, 2005, 1722–1726. ISBN: 1581139640. DOI: `10.1145/1066677.1067065`.

[31] Sandip Debnath, Prasenjit Mitra, and C. Lee Giles. “Identifying Content Blocks from Web Documents”. In: *Foundations of Intelligent Systems.* Ed. by Mohand-Said Hacid, Neil V. Murray, Zbigniew W. Raś, and Shusaku Tsumoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 285–293. ISBN: 978-3-540-31949-8. DOI: `10.1007/11425274_30`.

[32] R. Deepa and D.R. Nirmala. “Noisy elimination for web mining based on style tree approach”. In: *International Journal of Engineering Technology and Computer Research* 3.2 (2015). ISSN: 2348 -2117. URL: `https://ijetcr.org/index.php/ijetcr/article/view/126`.

[33] Amit Dutta, Sudipta Paria, Tanmoy Golui, and Dipak K. Kole. “Structural analysis and regular expressions based noise elimination from web pages for web content mining”. In: *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI).* 2014, pp. 1445–1451. DOI: `10.1109/ICACCI.2014.6968377`.

[34] Amit Dutta, Sudipta Paria, Tanmoy Golui, and Dipak Kumar Kole. “Noise Elimination from Web Page Based on Regular Expressions for Web Content Mining”. In: *Advanced Computing, Networking and Informatics- Volume 1.* Ed. by Malay Kumar Kundu, Durga Prasad Mohapatra, Amit Konar, and Aruna Chakraborty. Cham: Springer International Publishing, 2014, pp. 545–554. ISBN: 978-3-319-07353-8. DOI: `10.1007/978-3-319-07353-8_63`.

[35] Hassan F. Eldirdiery and A.H. Ahmed. “Detecting and removing noisy data on web document using text density approach”. In: *International Journal of Computer Applications* 112.5 (2015). DOI: `10.5120/19663-1328`.

[36] Stefan Evert. “A Lightweight and Efficient Tool for Cleaning Web Pages”. In: *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC’08)*. Ed. by Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, and Daniel Tapias. Marrakech, Morocco: European Language Resources Association (ELRA), 2008. ISBN: 2-9517408-4-0. URL: http://www.lrec-conf.org/proceedings/lrec2008/summaries/885.html.

[37] Junlan Feng, Patrick Haffner, and Mazin Gilbert. “A learning approach to discovering Web page semantic structures”. In: *Eighth International Conference on Document Analysis and Recognition (ICDAR’05)*. 2005, 1055–1059 Vol. 2. DOI: 10.1109/ICDAR.2005.19.

[38] 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)*. 2008, pp. 47–54. URL: https://www.researchgate.net/profile/Cedrick-Fairon/publication/200823227_GlossaNet_2_a_linguistic_search_engine_for_RSS-based_corpora/links/0fcfd50be620a1a614000000/GlossaNet-2-a-linguistic-search-engine-for-RSS-based-corpora.pdf#page=53.

[39] Aidan Finn, Nicholas Kushmerick, and Barry Smyth. “Fact or Fiction: Content Classification for Digital Libraries”. In: *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*. Dublin (Ireland), 2001, pp. 1–6. URL: http://www.ercim.org/publication/ws-proceedings/DelNoe02/AidanFinn.pdf.

[40] S. Ganeshmoorthy and R. Priya. “Eliminating the Web Noise by Text Categorization and Optimization Algorithm”. In: *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*. 2021, pp. 586–593. DOI: 10.1109/ICAIS50930.2021.9396020.

[41] Bo Gao and Qifeng Fan. “Multiple Template Detection Based on Segments”. In: *Advances in Data Mining. Applications and Theoretical Aspects*. Ed. by Petra Perner. Cham: Springer International Publishing, 2014, pp. 24–38. ISBN: 978-3-319-08976-8. DOI: 10.1007/978-3-319-08976-8_3.

[42] Albert Geitgey. “Unfluff - an automatic web page content extractor for node.js!” In: *GitHub repository* (2014). URL: https://github.com/ageitgey/node-unfluff.

[43] Filippo Geraci and Marco Maggini. “A multi-sequence alignment algorithm for Web template detection”. In: *KDIR 2011 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (Jan. 2011), pp. 121–128. URL: https://www.scitepress.org/Papers/2011/37128/37128.pdf.

[44] David Gibson, Kunal Punera, and Andrew Tomkins. “The volume and evolution of web page templates”. In: *Proceedings of the 14th International Conference on World Wide Web (WWW’05).* Ed. by Allan Ellis and Tatsuya Hagino. Chiba (Japan): ACM, 2005, pp. 830–839. ISBN: 1-59593-051-5. DOI: 10.1145/1062745.1062763.

[45] Jibing Gong, Hekai Zhang, Weixia Du, Huanhuan Li, and Hongnian Wen. “VB-PTC: Visual Block Multi-Record Text Extraction Based on Sensor Network Page Type Conversion”. In: *IEEE Access* 8 (2020), pp. 167900–167913. DOI: 10.1109/ACCESS.2020.3024194.

[46] Thomas Gottron. “Content Code Blurring: A New Approach to Content Extraction”. In: *2008 19th International Workshop on Database and Expert Systems Applications.* 2008, pp. 29–33. DOI: 10.1109/DEXA.2008.43.

[47] Ilya Grigorik. *Render-tree Construction, Layout, and Paint.* Available from URL: https://web.dev/critical-rendering-path-constructing-the-object-model/. 2018.

[48] Ilya Grigorik. *Constructing the Object Model.* Available from URL: https://web.dev/critical-rendering-path-constructing-the-object-model/. 2019.

[49] Gaurav Gupta and Indu Chhabra. “Optimized Template Detection and Extraction Algorithm for Web Scraping of Dynamic Web Pages”. In: *Global Journal of Pure and Applied Mathematics* 13.2 (2017), pp. 719–732. ISSN: 0973-1768. URL: https://www.ripublication.com/gjpam17/gjpamv13n2_43.pdf.

[50] Kulkarni A. H. and Patil B. M. “Template Extraction from Heterogeneous Web Pages with Cosine Similarity”. In: *International Journal of Computer Applications* 87.3 (2014), pp. 4–8. ISSN: 0975-8887. URL: https://research.ijcaonline.org/volume87/number3/pxc3893546.pdf.

[51] David Insa, Josep Silva, and Salvador Tamarit. “Using the words/leafs ratio in the DOM tree for content extraction”. In: *The Journal of Logic and Algebraic Programming* 82.8 (2013), pp. 311–325. ISSN: 1567-8326. DOI: 10.1016/j.jlap.2013.01.002.

[52] Geunseong Jung and Jaehyuk Cha. “A Webextension Based Framework for the Assessment of Main Content Extraction Methods from Web Pages”. In: *Available at SSRN 4127824* (2022). URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4127824.

[53] Geunseong Jung, Sungjae Han, Hansung Kim, Kwanguk Kim, and Jaehyuk Cha. “Don’t read, just look: Main content extraction from web pages using visual features”. In: *arXiv preprint arXiv:2110.14164* (2021).

[54] Vidya Kadam and Prakash R. Devale. “A Methodology for Template Extraction from Heterogeneous Web Pages”. In: *Indian Journal of Computer Science and Engineering (IJCSE)* 3.3 (2012). ISSN: 0976-5166. URL: http://www.ijcse.com/docs/INDJCSE12-03-03-101.pdf.

[55] Byeong Ho Kang and Yang Sok Kim. “Noise Elimination from the Web Documents by Using URL Paths and Information Redundancy”. In: *Proceedings of the 2006 International Conference on Information & Knowledge Engineering, IKE 2006, Las Vegas, Nevada, USA, June 26-29, 2006*. Ed. by Hamid R. Arabnia and Ray R. Hashemi. CSREA Press, 2006, pp. 135–141. URL: https://eprints.utas.edu.au/723/.

[56] Jeremy Keith. *DOM Scripting: Web Design with JavaScript and the Document Object Model*. friends of ED, 2005. ISBN: 9781590595336.

[57] C. Kim and K. Shim. “TEXT: Automatic Template Extraction from Heterogeneous Web Pages”. In: *IEEE Transactions on Knowledge and Data Engineering* 23.4 (2011), pp. 612–626. ISSN: 1041-4347. DOI: 10.1109/TKDE.2010.140.

[58] Barbara Ann Kitchenham, David Budgen, and Pearl Brereton. *Evidence-Based Software Engineering and Systematic Reviews*. Chapman & Hall/CRC, 2015. ISBN: 1482228653, 9781482228656.

[59] Aleksander Kocz and Wen-tau Yih. “Site-Independent Template-Block Detection”. In: *Knowledge Discovery in Databases: PKDD 2007*. Ed. by Joost N. Kok, Jacek Koronacki, Ramon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron.

Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 152–163. ISBN: 978-3-540-74976-9. DOI: `10.1007/978-3-540-74976-9_17`.

[60] Christian Kohlschütter. "A densitometric analysis of web template content". In: *Proceedings of the 18th International Conference on World Wide Web (WWW'09)*. Ed. by Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl. Madrid (Spain): ACM, 2009, pp. 1165–1166. ISBN: 978-1-60558-487-4. DOI: `10.1145/1526709.1526909`.

[61] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. "Boilerplate detection using shallow text features". In: *Proceedings of the 3th International Conference on Web Search and Web Data Mining (WSDM'10)*. Ed. by Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu. New York (New York / USA): ACM, 2010, pp. 441–450. ISBN: 978-1-60558-889-6. DOI: `10.1145/1718487.1718542`.

[62] Christian Kohlschütter and Wolfgang Nejdl. "A densitometric approach to web page segmentation". In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*. Ed. by James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury. Napa Valley (California / USA): ACM, 2008, pp. 1173–1182. ISBN: 978-1-59593-991-3. DOI: `10.1145/1458082.1458237`.

[63] Christian Kohlschütter et al. "Boilerpipe–boilerplate removal and fulltext extraction from HTML pages". In: *GitHub repository* (2010). URL: `https://github.com/kohlschutter/boilerpipe`.

[64] Anurendra Kumar, Keval Morabia, William Wang, Kevin Chang, and Alex Schwing. "CoVA: Context-aware Visual Attention for Webpage Information Extraction". In: *Proceedings of the Fifth Workshop on e-Commerce and NLP (ECNLP 5)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 80–90. DOI: `10.18653/v1/2022.ecnlp-1.11`.

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

[66] Jurek Leonhardt, Avishek Anand, and Megha Khosla. "Boilerplate Removal Using a Neural Sequence Labeling Model". In: New York, NY, USA: Association for Computing Machinery, 2020, 226–229. ISBN: 9781450370240. DOI: `10.1145/3366424.3383547`.

[67] Jing Li and C. I. Ezeife. “Cleaning Web Pages for Effective Web Content Mining”. In: *Database and Expert Systems Applications.* Ed. by Stéphane Bressan, Josef Küng, and Roland Wagner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 560–571. ISBN: 978-3-540-37872-3. DOI: 10.1007/11827405_55.

[68] Xiaoyan Li, Mengming Li, Rongfeng Zheng, Anmin Zhou, and Liang Liu. “NBCE: A Neo4j-Based Content Extraction Algorithm in Threat Intelligence Web Pages”. In: *2020 International Conference on Computer, Network, Communication and Information Systems (CNCI 2020).* 2020. ISBN: 978-1-989348-56-7. DOI: 10.23977/CNCI2020040.

[69] Zhao Li, Wee Keong Ng, and Aixin Sun. “Web data extraction based on structural similarity”. In: *Knowledge and Information Systems* 8.4 (2005), pp. 438–461. ISSN: 0219-3116. DOI: 10.1007/s10115-004-0188-z.

[70] Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 3540378812.

[71] Jiaying Liu, Xiangjie Kong, Xinyu Zhou, Lei Wang, Da Zhang, Ivan Lee, Bo Xu, and Feng Xia. “Data Mining and Information Retrieval in the 21st century: A bibliographic review”. In: *Computer Science Review* 34 (2019), p. 100193. ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2019.100193.

[72] Lawrence Lo, Vincent To-yee Ng, Patrick Ng, and Stephen Cf Chan. “Automatic Template Detection for Structured Web Pages”. In: *2006 10th International Conference on Computer Supported Cooperative Work in Design.* 2006, pp. 1–6. DOI: 10.1109/CSCWD.2006.253257.

[73] Ling Ma, Nazli Goharian, Abdur Chowdhury, and Misun Chung. “Extracting Unstructured Data from Template Generated Web Documents”. In: *Proceedings of the Twelfth International Conference on Information and Knowledge Management.* CIKM ’03. New Orleans, LA, USA: ACM, 2003, pp. 512–515. ISBN: 1-58113-723-0. DOI: 10.1145/956863.956961.

[74] Trupti B. Mane and Girish P. Potdar. “Template extraction from heterogeneous Web pages”. In: *International Journal of Advanced Computer Research* 2.4 (2012), p. 197. ISSN: 2277-7970. URL: https://www.accentsjournals.org/PaperDirectory/Conference/ICETT-2012/35.pdf.

[75] R. Manjula and A. Chilambuchelvan. “Extracting templates from Web pages”. In: *2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*. 2013, pp. 788–791. DOI: 10.1109/ICGCE.2013.6823541.

[76] Constantine Mantratzis, Mehmet Orgun, and Steve Cassidy. “Separating XHTML Content from Navigation Clutter Using DOM-Structure Block Analysis”. In: *Proceedings of the Sixteenth ACM Conference on Hypertext and Hypermedia*. HYPERTEXT ’05. Salzburg, Austria: Association for Computing Machinery, 2005, 145–147. ISBN: 1595931686. DOI: 10.1145/1083356.1083384.

[77] Michal Marek, Pavel Pecina, and Miroslav Spousta. “Web page cleaning with conditional random fields”. In: *Building and Exploring Web Corpora: Proceedings of the Fifth Web as Corpus Workshop, Incorporationg CleanEval (WAC3), Belgium*. 2007, pp. 155–162. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=91454cc9f2833f9bc4957a001441b69c99fe5d10.

[78] Sangita S. Modi and Sudhir B. Jagtap. “Multimodal Web Content Mining to Filter Non-learning Sites Using NLP”. In: *Proceeding of the International Conference on Computer Networks, Big Data and IoT (ICCBI - 2018)*. Ed. by A.Pasumpon Pandian, Tomonobu Senjyu, Syed Mohammed Shamsul Islam, and Haoxiang Wang. Cham: Springer International Publishing, 2020, pp. 23–30. ISBN: 978-3-030-24643-3. DOI: 10.1007/978-3-030-24643-3_3.

[79] Mahdi Mohammadi, Mohammad Javad Shayegan, and Nima Latifi. “Web Content Extraction by Weighing the Fundamental Contextual Rules”. In: *2021 7th International Conference on Signal Processing and Intelligent Systems (ICSPIS)*. 2021, pp. 01–08. DOI: 10.1109/ICSPIS54653.2021.9729342.

[80] Nang Kham Line Moong. “Constructing and Implementing a New DOM-based Content Extraction Algorithm”. PhD thesis. MERAL Portal, 2009.

[81] Stanislas Morbieu, Guillaume Bruneval, Mohamed Lacarne, Mohamed Kone, and François-Xavier Bois. “Main Content Extraction from Web Pages”. In: *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2020, pp. 1002–1005. DOI: 10.1109/ICMLA51294.2020.00162.

[82] Ranjani Murali. "An intelligent web spider for online e-commerce data extraction". In: *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)*. 2018, pp. 332–339. DOI: 10.1109/ICGCIoT.2018.8753071.

[83] 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. IEEE Computer Society, 2009, pp. 232–236. DOI: 10.1109/KSE.2009.39.

[84] Alpa K Oza and Shailendra Mishra. "Elimination of noisy information from web pages". In: *International Journal of Recent Technology and Engineering* 2.1 (2013), pp. 115–117. ISSN: 2277-3878. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=be66ef066f05ef213f01487129e40d79076e0684.

[85] Jeff Pasternack and Dan Roth. "Extracting Article Text from the Web with Maximum Subsequence Segmentation". In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: ACM, 2009, pp. 971–980. ISBN: 978-1-60558-487-4. DOI: 10.1145/1526709.1526840. URL: https://doi.acm.org/10.1145/1526709.1526840.

[86] Jeff Pasternack and Dan Roth. "Extracting Article Text from the Web with Maximum Subsequence Segmentation". In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: Association for Computing Machinery, 2009, 971–980. ISBN: 9781605584874. DOI: 10.1145/1526709.1526840.

[87] Gregory Piatetsky-Shapiro and William Frawley, eds. *Knowledge discovery in databases*. en. AAAI Press. London, England: MIT Press, Dec. 1991.

[88] David Pinto, Michael Branstein, Ryan Coleman, W. Bruce Croft, Matthew King, Wei Li, and Xing Wei. "QuASM: A System for Question Answering Using Semi-Structured Data". In: *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*. JCDL '02. Portland, Oregon, USA: Association for Computing Machinery, 2002, 46–55. ISBN: 1581135130. DOI: 10.1145/544220.544228.

[89] Manjunath Pujar and Monica R Mundada. “A Systematic Review Web Content Mining Tools and its Applications”. In: *International Journal of Advanced Computer Science and Applications* 12.8 (2021). URL: `https://www.academia.edu/download/69949991/Paper_86-A_Systematic_Review_Web_Content_Mining_Tools.pdf`.

[90] Gusti Lanang Putra Eka Prismana. “Automatic Web News Content Extraction”. In: *Journal Research of Social, Science, Economics, and Management* 1.7 (2022), 785–794. DOI: `10.36418/jrssem.v1i7.107`.

[91] Xin Qi and JianPeng Sun. “Eliminating Noisy Information in Webpage through Heuristic Rules”. In: *2011 International Conference on Computer Science and Information Technology.* 2011.

[92] Pir Abdul Rasool Qureshi and Nasrullah Memon. “Hybrid Model of Content Extraction”. In: *Journal of Computer and System Sciences* 78.4 (July 2012), pp. 1248–1257. ISSN: 0022-0000. DOI: `10.1016/j.jcss.2011.10.012`.

[93] Neeraj Raheja and VK Katiyar. “A Noise Reduction Approach based on NX 1 Table and XSL Display Method for Efficient Web Data Extraction”. In: *International Journal of Computer Applications* 64.11 (2013). DOI: `10.5120/10677-5552`.

[94] 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).* New York, NY, USA: ACM, 2004, pp. 502–511. ISBN: 1-58113-844-X. DOI: `10.1145/988672.988740`.

[95] Tom Rowlands, Paul Thomas, and Stephen Wan. “Web indexing on a diet: Template removal with the sandwich algorithm”. In: *Proceedings of the 14th Australasian Document Computing Symposium.* Sydney, Australia, 2009. URL: `http://es.csiro.au/adcs2009/proceedings/poster-presentation/06-rowlands.pdf`.

[96] Lassri Safae, Benlahmar El Habib, and Tragha Abderrahim. “A Review of Machine Learning Algorithms for Web Page Classification”. In: *2018 IEEE 5th International Congress on Information Science and Technology (CiSt).* 2018, pp. 220–226. DOI: `10.1109/CIST.2018.8596420`.

[97] Gerard Salton and Donna Harman. “Information Retrieval”. In: *Encyclopedia of Computer Science*. GBR: John Wiley and Sons Ltd., 2003, 858–863. ISBN: 0470864125.

[98] Makinde Opeyemi Samuel, Afolabi Ibukun Tolulope, and Oladipupo Olufunke Oyejoke. “A Systematic Review of Current Trends in Web Content Mining”. In: *Journal of Physics: Conference Series* 1299.1 (2019), p. 012040. DOI: 10.1088/1742-6596/1299/1/012040.

[99] Pan Ei San. “Boilerplate Removal and Content Extraction from Dynamic Web Pages”. In: *International Journal of Computer Science, Engineering and Applications* 4.6 (2014), p. 27. URL: https://meral.edu.mm/record/4270/files/Boilerplate%20removal%20and%20content%20extraction(ijren).pdf.

[100] A Saravanan and S Sathya Bama. “Extraction of Core Web Content from Web Pages using Noise Elimination”. In: *Journal of Engineering Science & Technology Review* 13.4 (2020). URL: http://www.jestr.org/downloads/Volume13Issue4/fulltext171342020.pdf.

[101] Roland Schäfer. “Accurate and efficient general-purpose boilerplate detection for crawled web corpora”. In: *Language Resources and Evaluation* 51.3 (2017), pp. 873–889. ISSN: 1574-0218. DOI: 10.1007/s10579-016-9359-2.

[102] Dipali Shete, Sachin Bojewar, and Ankit Sanghvi. “Survey Paper on Web Content Extraction & Classification”. In: *2021 6th International Conference for Convergence in Technology (I2CT)*. 2021, pp. 1–6. DOI: 10.1109/I2CT51068.2021.9417947.

[103] P. Sivakumar. “Effectual Web Content Mining using Noise Removal from Web Pages”. In: *Wireless Personal Communications* 84.1 (2015), pp. 99–121. ISSN: 1572-834X. DOI: 10.1007/s11277-015-2596-7.

[104] Dandan Song, Fei Sun, and Lejian Liao. “A hybrid approach for content extraction with text density and visual importance of DOM nodes”. In: *Knowledge and Information Systems* 42.1 (2015), pp. 75–96. ISSN: 0219-3116. DOI: 10.1007/s10115-013-0687-x.

[105] Miroslav Spousta, Michal Marek, and Pavel Pecina. “Victor: the web-page cleaning tool”. In: *4th Web as Corpus Workshop (WAC-4)-Can we beat Google*. 2008, pp. 12–17. URL: https://www.academia.edu/download/1435992/57hiyya85c1m7631.pdf#page=18.

[106] Fei Sun, Dandan Song, and Lejian Liao. "DOM Based Content Extraction via Text Density". In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '11. Beijing, China: ACM, 2011, pp. 245–254. ISBN: 978-1-4503-0757-4. DOI: `10.1145/2009916.2009952`.

[107] Karthikeyan T, Sekaran K, Ranjith D, Vinoth kumar V, and Balajee J.M. "Personalized Content Extraction and Text Classification Using Effective Web Scraping Techniques". In: *International Journal of Web Portals* 11.2 (2019), pp. 41 –52. ISSN: 1938-0194. DOI: `10.4018/IJWP.2019070103`.

[108] Rashmi D Thakare and Manisha R Patil. "Extraction of Template using Clustering from Heterogeneous Web Documents". In: *International Journal of Computer Applications* 119.11 (2015). DOI: `10.5120/21112-3906`.

[109] R. Uma and B. Latha. "Noise elimination from web pages for efficacious information retrieval". In: *Cluster Computing* (2018). ISSN: 1573-7543. DOI: `10.1007/s10586-018-2366-x`.

[110] Nichita Utiu and Vlad-Sebastian Ionescu. "Learning Web Content Extraction with DOM Features". In: *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP).* 2018, pp. 5–11. DOI: `10.1109/ICCP.2018.8516632`.

[111] Erdinç Uzun, Hayri Volkan Agun, and Tarik Yerlikaya. "A hybrid approach for extracting informative content from web pages". In: *Information Processing and Management* 49.4 (2013), pp. 928 –944. ISSN: 0306-4573. DOI: `10.1016/j.ipm.2013.02.005`.

[112] Roberto Panerai Velloso and Carina F. Dorneles. "Web Page Structured Content Detection Using Supervised Machine Learning". In: *Web Engineering.* Ed. by Maxim Bakaev, Flavius Frasincar, and In-Young Ko. Cham: Springer International Publishing, 2019, pp. 3–18. ISBN: 978-3-030-19274-7. DOI: `10.1007/978-3-030-19274-7_1`.

[113] Karane Vieira, André Luiz da Costa Carvalho, Klessius Berlt, Edleno S. de Moura, Altigran S. da Silva, and Juliana Freire. "On Finding Templates on Web Collections". In: *World Wide Web* 12.2 (2009), pp. 171–211. ISSN: 1386-145X. DOI: `10.1007/s11280-009-0059-3`.

[114] 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).* Arlington, Virginia, USA: ACM, 2006, pp. 258–267. ISBN: 1-59593-433-2. DOI: `10.1145/1183614.1183654`.

[115] Thijs Vogels, Octavian-Eugen Ganea, and Carsten Eickhoff. “Web2Text: Deep Structured Boilerplate Removal”. In: *Advances in Information Retrieval.* Ed. by Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury. Cham: Springer International Publishing, 2018, pp. 167–179. ISBN: 978-3-319-76941-7. DOI: `10.1007/978-3-319-76941-7_13`.

[116] Yu Wang, Bingxing Fang, Xueqi Cheng, Li Guo, and Hongbo Xu. “Incremental Web Page Template Detection”. In: *Proceedings of the 17th International Conference on World Wide Web (WWW ’08).* Beijing, China: ACM, 2008, pp. 1247–1248. ISBN: 978-1-60558-085-2. DOI: `10.1145/1367497.1367749`.

[117] Tim Weninger, William Henry Hsu, and Jiawei Han. “CETR: Content Extraction via Tag Ratios”. In: *Proceedings of the 19th International Conference on World Wide Web (WWW’10).* Ed. by Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti. Raleigh (North Carolina / USA): ACM, 2010, pp. 971–980. ISBN: 978-1-60558-799-8. DOI: `10.1145/1772690.1772789`.

[118] Shanchan Wu, Jerry Liu, and Jian Fan. “Automatic Web Content Extraction by Combination of Learning and Grouping”. In: *Proceedings of the 24th International Conference on World Wide Web.* WWW ’15. Florence, Italy: International World Wide Web Conferences Steering Committee, 2015, pp. 1264–1274. ISBN: 978-1-4503-3469-3. DOI: `10.1145/2736277.2741659`.

[119] Jiang-Ming Yang, Rui Cai, Yida Wang, Jun Zhu, Lei Zhang, and Wei-Ying Ma. “Incorporating Site-Level Knowledge to Extract Structured Data from Web Forums”. In: *Proceedings of the 18th International Conference on World Wide Web.* WWW ’09. Madrid, Spain: Association for Computing Machinery, 2009, 181–190. ISBN: 9781605584874. DOI: `10.1145/1526709.1526735`.

[120] 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). Washington, D.C.: ACM, 2003, pp. 296–305. ISBN: 1-58113-737-0. DOI: 10.1145/956750.956785.

[121] Bowen Yu, Junping Du, and Yingxia Shao. *Web Page Content Extraction Based on Multi-feature Fusion.* 2022. DOI: 10.48550/ARXIV.2203.12591.

[122] Xin Yu and Zhengping Jin. "Web content information extraction based on DOM tree and statistical information". In: *2017 IEEE 17th International Conference on Communication Technology (ICCT).* 2017, pp. 1308–1311. DOI: 10.1109/ICCT.2017.8359846.

[123] Hao Zhang and Jie Wang. "Boilerplate Detection via Semantic Classification of TextBlocks". In: *2021 International Joint Conference on Neural Networks (IJCNN).* 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9534308.

[124] Shengnan Zhang, Jiawei Wu, and Kun Yang. "A Webpage Segmentation Method Based on Node Information Entropy of DOM Tree". In: *Journal of Physics: Conference Series* 1624.3 (2020), p. 032023. DOI: 10.1088/1742-6596/1624/3/032023.

[125] Chenxu Zhao, Rui Zhang, and Jianzhong Qi. "Web Page Template and Data Separation for Better Maintainability". In: *Web Information Systems Engineering – WISE 2018.* Ed. by Hakim Hacid, Wojciech Cellary, Hua Wang, Hye-Young Paik, and Rui Zhou. Cham: Springer International Publishing, 2018, pp. 439–449. ISBN: 978-3-030-02922-7. DOI: 10.1007/978-3-030-02922-7_30.

Appendices

Appendix A

Glossary of Acronyms

AC (Assessment Criteria): Criteria used to check the quality of a paper based on several quality assessment questions.

ACM (Association for Computing Machinery): A global scientific and educational organization which aims to advance the art, engineering, science, and application of computing, serving public and professional interests (definition extracted from the official website).

API (Application Programming Interface): A way to enable two or more software components to communicate with each other using a set of predefined definitions and protocols (definition extracted from Amazon AWS).

CMS (Content Management System): An application that allows us to manage the creation, update, and publication of digital content (definition extracted from Wikipedia).

CRP (Critical Rendering Path): The step sequence performed by the browser to convert the HTML code, CSS code, and JavaScript into a rendered image of the web page on the screen.

CS (Complete Subdigraph): A set of nodes for which all the nodes are connected to each other.

CSS (Cascade Style Sheet): The language used to provide style to an HTML document. It describes how to display the HTML elements.

CSSOM (Cascade Style Sheet Object Model): A set of APIs for the manipulation of CSS. It is similar to the DOM, but for the CSS rather than the HTML (definition extracted from Mozilla).

DDR (Double Data Rate): A form of computer memory that uses a computer bus operating with a double data rate, which transfers data on both, the rising and falling edges of the clock signal (definition extracted from Wikipedia).

DHTML (Dynamic HyperText Markup Language): A term that describes the combination of HTML, style sheets and client-side scripts (VBScript, JavaScript, etc.) to enable the development of interactive and animated documents (definition extracted from Wikipedia).

DOM (Document Object Model): A programming API for XML and HTML documents. It defines the documents with a tree structure whose nodes are objects representing a part of the document (definition extracted from Wikipedia).

EC (Exclusion Criteria): Criteria used to exclude a paper from the selection results of a systematic review.

ETDM (Equal Top-Down Mapping): A mapping between two DOM trees that is used in several site-level block detection techniques.

GGS (GII-GRIN-SCIE): A committee formed by GII (Group of Italian Professors of Computer Engineering), GRIN (Group of Italian Professors of Computer Science), and SCIE (Spanish Computer-Science Society). They publish the GGS Conference Rating.

GUI (Graphical User Interface): A kind of user interface that allows users to interact with a program using graphical icons and visual indicators (definition extracted from Wikipedia).

HTML (HyperText Markup Language): The standard markup language to produce documents designed to be opened in a web browser. Hence, it defines the meaning and structure of web content (definition extracted from Wikipedia).

HTTP (HyperText Transmission Protocol): An application layer protocol that allows information transfer through the World Wide Web (definition extracted from Wikipedia).

IC (Inclusion Criteria): Criteria used to include a paper into the selection results of a systematic review.

IEEE (Institute of Electrical and Electronics Engineers): An organization dedicated to advancing technology for the benefit of humanity (definition extracted from the official website).

JS (JavaScript): A programming language which, combined with HTML and CSS, forms the core technologies of the World Wide Web (definition extracted from Wikipedia).

JSON (JavaScript Object Notation): An open standard format for both, files and data interchange, that is based on human-readable text and consists of attribute-value pairs and arrays (definition extracted from Wikipedia).

KDD (Knowledge Discovery in Databases): A subfield of Machine Learning related to discovering information from large amounts of possible uncertain data.

LCS (Longest Common Subsequence): The longest subsequence which is common to all sequences in a set of sequences.

LOC (Lines Of Code): A software metric that measures the size of a computer program through the number of lines in the text of its source code (definition extracted from Wikipedia).

PL (Page-Level): A block detection technique that only uses the key page to infer the block.

QUORUM (QUality Of Reporting Of Meta-analyses): An international conference convened in 1996 to establish standards for improving the report of meta-analyses of clinical randomized controlled trials. The result of the conference was the QUORUM checklist and a flow diagram, which define the sections of a report of a systematic review or a meta-analysis (definition extracted from Wikipedia).

RAM (Random Access Memory): A kind of computer memory which can be read and changed in any order (definition extracted from Wikipedia).

REST (REpresentational State Transfer): An API between software components that uses HTTP to obtain or manipulate data.

RQ (Research Question): A specific question that the research intends to answer.

RSS (Really Simple Syndication): An XML format that provides a standardized format to users and applications to access website updates (definition extracted from Wikipedia).

SEO (Search Engine Optimization): The process of improving a website in order to increase its visibility in search engines. The visibility is directly related to the *pagerank* of a website in a search engine. The pagerank determines the importance of this web page in a search.

SL (Site-Level): A block detection technique that uses the key page and several web pages from the same website to infer the block.

SQL (Structured Query Language): A domain-specific language used to manage data from relational databases.

SSH (Secure SHell): A network protocol that operates network services with security over an unsecured network (definition extracted from Wikipedia).

TED (Tree Edit Distance): A sequence of operations with the minimum cost that transforms one tree into another.

URI (Uniform Resource Locator): A unique string that allows us to identify a resource, physical or logical, used in a network.

URL (Uniform Resource Identifier): A reference to a given unique web resource on the Web (definition extracted from Mozilla).

VRAIN (Valencian Research Institute for Artificial Intelligence): A research institute of the Universitat Politècnica de València, which is composed of researchers belonging to 7 research groups: Language Engineering and Pattern Recognition (ELiRF), Automata, Formal Languages and its Applications (ALFA), Extensions of Logic Programming (ELP), Machine Learning and Language Processing (MLLP), Computer Technology and Artificial Intelligence (GTIIA) and Multiparadigm Software Technology (MIST), Interactive Technologies Lab (VertexLit), and the Research Center on Software Production Methods (PROS) of the Universitat Politècnica de València (definition extracted from the official website).

W3C (World Wide Web Consortium): An international community which is responsible for developing Web standards (definition extracted from the official website).

XML (eXtensible Markup Language): A machine-readable and human-readable markup language which defines a set of rules for encoding arbitrary data (definition extracted from Wikipedia).

Appendix B

Scientific Contributions

This appendix presents all the contributions related to this thesis where the author has actively participated. The author has contributed in all the mentioned research papers in different ways: participating actively in the brainstorming sessions where the algorithms and models were defined, collaborating in the definition and proof of formal aspects like lemmas or theorems, and as an active part of the implementation and empirical evaluation of all the WebExtensions and tools described in every paper.

B.1 Conference papers

- Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. Template Extraction Based on Menu Information. 9th International Workshop on Automated Specification and Verification of Web Systems (WWV 13). Proceedings of WWV 2013, pages 71-80, 2013.

- Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. Automatic Detection of Webpage Candidates for Site-Level Web Template Extraction. 10th International Workshop on Automated Specification and Verification of Web Systems (WWV 14), 2014.

- Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. Site-Level Template Extraction Based on Hyperlink Analysis. XIV Jornadas sobre Programación y Lenguajes (PROLE 2014). Proceedings of PROLE 2014, pages 23-36, 2014.

- Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. TeMex: The Web Template Extractor. 24th International Conference on World Wide Web (WWW 2015). Companion: Proceedings of the 24th International Conference on World Wide Web, pages 155–158, 2015.

- Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. A Collection of Website Benchmarks Labelled for Template Detection and Content Extraction. XV Jornadas sobre Programación y Lenguajes (PROLE 2015). Proceedings of PROLE 2015, 2015.

- Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. Site-Level Web Template Extraction Based on DOM Analysis. 10th International Andrei Ershov Informatics Conference (PSI 2015). Published on Springer Lecture Notes in Computer Science Vol 9609, pages 36-49, 2016.

- Julián Alarte, David Insa, and Josep Silva. Page-Level Webpage Menu Detection. XVI Jornadas sobre Programación y Lenguajes (PROLE 2016). Proceedings of PROLE 2016, pages 134-147, 2016.

- Julián Alarte, David Insa, and Josep Silva. Webpage Menu Detection Based on DOM. 43rd International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2017). Published on Springer Lecture Notes in Computer Science Vol 10139, pages 411-422, 2017.

- Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. Main Content Extraction from Heterogeneous Webpages. Web Information Systems Engineering (WISE 2018). Published on Springer Lecture Notes in Computer Science Vol 11233, pages 393-407, 2018.

- Julián Alarte and Josep Silva. HybEx: A Hybrid Tool for Template Extraction. The Web Conference 2022 (WWW 2022). Companion: Proceedings of the Web Conference 2022, pages 205–209, 2022.

B.2 Journal Publications

- Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. Automatic Detection of Webpages that Share the Same Web Template. Electronic Proceedings in Theoretical Computer Science 163, pages 2-15, 2014.

- Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. Web Template Extraction Based on Hyperlink Analysis. Electronic Proceedings in Theoretical Computer Science 173, pages 16-26, 2015.

- Julián Alarte, Josep Silva, and Salvador Tamarit. What Web Template Extractor Should I Use? A Benchmarking and Comparison for Five Template Extractors. ACM Transactions on the Web Vol. 13 Iss. 2 Num. 9: 1-19, 2019.
- Julián Alarte and Josep Silva. Page-Level Main Content Extraction From Heterogeneous Webpages. ACM Transactions on Knowledge Discovery from Data Vol. 15 Iss. 6 Num. 105: 1-21, 2021.

B.3 List of derived artifacts

Resource Name	Type	URL
Web information retrieval	Web page	`http://personales.upv.es/josilga/retrieval/`
TeCo	Benchmark suite	`http://personales.upv.es/josilga/retrieval/teco/`
TemEx	Web page	`https://personales.upv.es/josilga/retrieval/Web-TemEx/`
	Firefox add-ons version	`https://addons.mozilla.org/en-US/firefox/addon/template-extractor/`
MenEx	Web page	`https://personales.upv.es/josilga/retrieval/Web-MenEx/`
	Firefox add-ons version	`https://addons.mozilla.org/es/firefox/addon/menex/`
Site-level ConEx	Web page	`https://personales.upv.es/josilga/retrieval/Web-ConEx/`
	Firefox add-ons version	`https://addons.mozilla.org/es/firefox/addon/conex-web-content-extractor/`
Page-level ConEx	Web page	`https://personales.upv.es/josilga/retrieval/Web-ConEx/`
	Firefox add-ons version	`https://addons.mozilla.org/es/firefox/addon/page-level-content-extractor/`
HybEx	Web page	`https://personales.upv.es/josilga/retrieval/Web-HybEx/`
	Firefox add-ons version	`https://addons.mozilla.org/es/firefox/addon/hybrid-template-extractor/`
SST	WebExtension	`http://personales.upv.es/josilga/retrieval/Web-TemEx/downloads/Extractors/SST.zip`
RTDM-TD	WebExtension	`http://personales.upv.es/josilga/retrieval/Web-TemEx/downloads/Extractors/RTDMTD.zip`
RBM-TD	WebExtension	`http://personales.upv.es/josilga/retrieval/Web-TemEx/downloads/Extractors/RBMTD.zip`
IWPTD	WebExtension	`http://personales.upv.es/josilga/retrieval/Web-TemEx/downloads/Extractors/Incremental.zip`