Implementation and Empirical Performance Evaluation Of Semantic Web Page Segmentation Algorithms

SUBMITTED

BY

ANASUA MITRA

EXAMINATION ROLL NUMBER: M4SWE14-17

REGISTRATION NUMBER: 121038 of 2012-2013

CLASS ROLL NUMBER: 001211002020

This THESIS SUBMITTED TO

THE FACULTY OF ENGINEERING & TECHNOLOGY OF JADAVPUR UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING

IN

SOFTWARE ENGINEERING

UNDER THE SUPERVISION

OF

DR. SAMIRAN CHATTOPADHYAY

DEPARTMENT OF INFORMATION TECHNOLOGY

JADAVPUR UNIVERSITY

2014

Department of Information Technology

Faculty of Engineering & Technology

JADAVPUR UNIVERSITY

Certificate of Submission

I hereby recommend that the thesis, entitled "**Implementation and Empirical Performance Evaluation Of Semantic Web Page Segmentation Algorithms**", prepared by **Anasua Mitra** (Registration. No.1**21038 of 2012-2013**) under my supervision, be accepted in partial fulfillment of the requirement for the degree of Master of Engineering in Software Engineering from the Department of Information Technology under Jadavpur University.

_____

(Dr. SAMIRAN CHATTOPADHYAY)

Department of Information Technology

Jadavpur University

Countersigned by:

_____

(Head of the Department)

Department of Information Technology

Jadavpur University

_____

(Dean)

Jadavpur University

Faculty Of Engineering & Technology

JADAVPUR UNIVERSITY

DEPARTMENT OF INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING & TECHNOLOGY

CERTIFICATE OF APPROVAL

The thesis at instance is hereby approved as a creditable study of an Engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve this thesis for the purpose for which it is submitted.

_____ _____

(Signature of the Examiner) **Signature of the Supervisor)**

Declaration of Compliance of Academic Ethics

I hereby declare that this thesis contains literature survey work by me, as a part of my Master Of Engineering in Software Engineering studies.

All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

_____

(Signature with Date)

Name : **ANASUA MITRA**

Roll Number : **001211002020**

Thesis Title : **Implementation and Empirical Performance Evaluation Of Semantic Web Page Segmentation Algorithms**

Acknowledgement

I would like to thank a lot of people who gave me unending support and inspiration from the beginning and without their help this thesis and project would not have been completed.

First and foremost, I would like to thank my guide Dr. **SAMIRAN CHATTOPADHYAY**, whose guidance and encouragement have helped me immensely in understanding the subject. Thank you for the many discussions we had and his valuable suggestions which made this thesis much better in the end. I would also like to thank my parents for their tireless support during my studies.

I would also like to thank all of my classmates for the constant support and encouragement they provided all the time. Finally I would like to thank my fellow project partner Sauradyuti for ideas, continued support in algorithm implementation and encouragement.

Regards,

_____

Location :

Date:

ANASUA MITRA

M.E. in Software Engineering

Class Roll No: 001211002020

Exam Roll No: M4SWE14-17

Registration No:121038 of 2012-2013

Abstract

This thesis explores the effectiveness of different semantic Web page segmentation algorithms on modern websites. We compare the **BlockFusion, Block O Matic, VIPS and the novel Gomory Hu Tree Based Clustering Algorithm**, these were implemented as part of this thesis, to each other. We introduce a new testing framework that allows to selectively run different algorithms on different datasets and that subsequently automatically compares the generated results to the ground truth. We used it to run each algorithm in four different configurations where we varied datasets, evaluation metric for a total of 32 combinations. We found that all algorithms performed better on random pages on average than on popular pages. The reason for this is most likely the higher complexity of popular pages. Of the different algorithms BlockFusion has the lowest F-score on average and Gomory Hu Tree Based Clustering Algorithm the highest. Overall there is still room for improvement as we find the best average F-score to be 0.45.

Contents

List Of Figures

List Of Tables

Chapter 1

Introduction

1.1 Topic :

In this thesis we explore *the problem of algorithmically segmenting Web pages into semantic blocks.* The problem can be described as follows: Humans can intuitively infer which parts (or blocks) on a Web page belong together simply by looking at them. It is immediately clear to them what constitutes a menu, an advertisement, an article, a headline etc. This is even true in most cases for languages which the reader does not know, by recognizing visual features such as the positioning of a block, the font size, the styling and the experience the user has with similar pages. We are now interested in the question whether it is possible to teach computers this ability that humans possess naturally.

We therefore look at a number of existing approaches that try to solve this problem in different ways. *They can be divided into vision-based (e.g – VIPS)[47,48] ones, that try to imitate the way humans do the segmentation by looking at the rendered document, structure-based [32, 33, 35] ones, which take the underlying structure of Web pages into account, given to them by the document creators, and language-based ones(e.g BlockFusion) [46][56], which only look at linguistic features of Web pages to infer the segmentation.*

We therefore took a novel web page segmentation algorithm *(Segmenting Web page with Gomory-Hu Based Clustering Tree) [45] that combines the features from each class namely - i) Vision Based ii) Structure Based iii) Text Based and analyze the performance systematically on a common dataset* using a testing framework we developed for this thesis. We present the results of our evaluation and discuss how well computers currently perform on the segmentation problem and what remains to be done.

1.2 Applications

Web page segmentation has been done to address a problem in different fields including mobile web, archiving, phishing, etc. In this section, we summarize the problems that web page segmentation addresses in different fields.

Semantic Web : We posit that a robust solution of the segmentation problem would be highly useful and could be utilized in many different applications. One of the most intriguing prospects is the application of such a solution to the Semantic Web. The hype that surrounded the Semantic Web in the early 2000s, promising to turn the Web into a giant distributed database of all human knowledge, has since died down. Outside of research projects there have been almost no applications of semantic technologies such as *RDF [2], SPARQL [8] and OWL [9, 10]*. The only technique that has gained some popularity with Web designers is the addition of some semantic tags in *HTML5 [5, 7]* such as <nav>, <article>, <footer>, <section> and a few more. While some modern websites make use of these, the vast majority of sites still does not. *On those sites a segmentation algorithm could be helpful to enable some basic reasoning and querying capabilities as were imagined in the original vision of the Semantic Web.*

Mobile Web Content Adaptation : There are a number of problems associated with browsing web pages on mobile devices. *These problems include the following: 1. the wireless bandwidth is quite limited and very expensive; 2. the screen size varies and can be very small;3. some small screen devices have very limited memory and processing capabilities ; 4. the content of a large web page cannot fit into the memory of a device;5. color schema can be limited on a small screen device; 6. mobile devices do not have keyboard or mouse ; 7. Scaling*

down for small screen devices or scaling up for large displays could be difficult with web pages that are optimized for certain size ; 8. the same content is served to all users who are interacting in different context with the mobile web ; 9. the small screen device users have to scroll down a lot to access the content (the worst case is the two-dimensional scrolling) ;10. since the web page is not properly marked up, it is very hard to personalize or customize pages for mobile devices. 11. most small screen devices cannot handle multimedia data 12. most web pages are designed to be displayed on large screens for Desktop machines . 13. the information need is very different for mobile web users, people focus more on getting direct answers to specific questions, and expect more relevant and clear results instead of browsing through large amount of data . 14. context is also very important, for example location, personal profiles, time of the day, schedule, browsing history, etc. . 15. For multi-platform development, the relationship between web page blocks and elements are not semantically explicitly specified so one cannot easily display a page on another or alternative platform .[15, 16, 17, 18, 19, 20, 32, 33, 35, 36, 37, 38, 39, 40]

> *[32] aims to extract and present only the important parts of the web page for delivery to mobile devices. [33] similarly aims to discover the importance of segments of a web page which can be used to better adapt a web page for mobile devices – block-importance information can be used to decide which part of the page need to be displayed first to the user.*

Informative Page Blocks Detection for Efficient Information Extraction :

*Recently, Web information extraction has become more challenging Due to the complexity and the diversity of Web structures and representations. The HTML structure of a Web document has also become more complicated, Making it harder to recognize the target content using the DOM tree We have to provide some clues about which content should be extracted from the page. Web page has two types of blocks: informative and redundant content blocks. Informative content blocks include content which is semantically meaningful to users and redundant content blocks include redundant data such as company logos, navigation panels, advertisement banners, etc. T*his redundant data influences the result of the data mining algorithm. Therefore it is important to identify the role

and importance of web page contents. *So the focus for efficient information retrieval is to eliminate the intra-page redundancy.[11, 12, 13, 14]* There are also many papers published on information extraction that focuses on record extraction *[27, 28, 29]*. *They mainly focus on extracting unstructured data from web pages into databases where structured information is stored. Even though they are also related to web pages segmentation their focus is different.* They mainly focus on extracting specific kind of data, for example weather forecast data, or prices, etc., however the segmentation algorithms that we refer here focus on segmenting a web page into a coherent sets of blocks.

Information Retrieval : Traditionally, l*ink analysis assume that a link represent a relationship between two web pages (A relates to B), however a link might represent a relationship between parts of web pages (part of A relates to part of B).* Furthermore,noise in a web page can cause topic drift problem. All link analysis algorithms are based on two assumptions: 1) the links convey human endorsement (if somebody adds a link to another page that means they endorse the other page) and 2) pages that are co-cited by a certain page are likely related to the same topic. Even though these are valid when the page includes a single topic, unfortunately most pages these days include a lot of sub-pages with different topics, for example the home page of Yahoo, MSN, etc. indicate two major issues: *1) if a web page is considered as a single semantic unit then it does not consider multiple topics in a web page, which means topics can be scattered at various regions of the page which can cause low retrieval precision; 2) if the page contains multiple unrelated topics then correlations among terms in a web page may be inappropriately calculated. Unrelated topics then correlations among terms in a web page may be inappropriately calculated. The redundant content in web pages such as advertisements, etc can be misleading for search engines. [48] indicates that search engines typically index whole text of a web page, and therefore they include a lot of text which is useless for processing, indexing, retrieving and extracting. Therefore, they aim to identify redundant content blocks such that the searching/information retrieval can be done in informative content blocks. [30]* propose to identify pagelets in a web page to improve the performance of information retrieval. *They mainly talk about two principles of information retrieval: 1) relevant linkage principle (web pages*

include many structural links such as menu links and these kinds of links violate the relevant linkage principle) and 2) topical unity principle (pages include mixture of topics which would negatively affect the information retrieval).

- ***Image Retrieval - (image search on the web):*** *Most search engines regard web pages as atomic units, however most web pages do not contain uniform information. They contain multiple topics with different kinds of segments such as navigation blocks, interaction elements,etc. Therefore, these affect the performance of image search algorithms on the web. [31] also indicates that traditional image search algorithms use fixed-length window size (for example, minimum 20 terms to maximum entire page) to extract the contextual information. [31] indicates that although this method is straightforward, this method tends to produce low-level accuracy as text tend to be associated with the wrong image.*

Web Page Phishing : Phishing is a criminal trick of stealing one's personal information by sending them spoofed emails urging them to visit a forged website that looks like a true one. *Identifying visual similarities between web pages is important for web page phishing detection* . Only looking at the underlying HTML source code is not enough and it is important to investigate the visual similarities (the way users' see the web page) to detect phishing web pages .Detecting Phishing web Pages, *[21]* takes an image of a web page and aims to identify the blocks in a web page to detect the visual similarity between web pages for identifying phishing web pages. *[22, 23, 24]* aims to segment a web page to identify blocks and then compare them for similarity, similarity above a threshold is accepted as phishing pages.

Duplicate Detection : Duplicate web pages effect the user-experience and also consumes a lot of resources of search engines. *Most duplicate detection algorithms are based on a concept called shigles . Shingles are extracted by moving a fixed-sized window over the text of a web page and smallest shingles (in terms of hash values) are stored as the signature of a web page.* These shingles are then used to compare web pages to identify duplicates. This approach is problematic because the noise in the web pages are also considered – there

might be pages with same content but different noisy content. Therefore, *[27]* proposes to use segmentation algorithms to identify the main content and then take the signature of web pages for comparison purposes. *[29]* focuses on identifying identical content presented using different web page layouts.

User Interest Detection / Sense making Task : web site developers usually include different topics on web pages and when the user browses a page he is usually interested in only a part of it. When they browse that page, they are usually interested in the updates of that block and if that updated is automatically shown to the user, that will save significant amount of time of browsing time and it will improve user's browsing experience. *Sense making is "sense making tasks require that users gather and comprehend information from many sources to answer complex questions"*. Even though search engines are targeting and helping users to find relevant resources the users still spend significant amount of time to find the information they are looking for in a given web page. They indicate that people spend 75% of their time in post-search phase of looking through individual web pages or sites. *Therefore, web page segmentation tasks would be useful to highlight the relevant part to the user.*

Web Page Classification / Clustering : There might be noise in the page and some links that are located in those noisy parts of the page can be misleading for the classification of pages. And One can suppose that words that belong to the central part of the page carry more information than words from the down right corner. Therefore, there should be a way to weigh words differently in different layout contexts. *[25]* indicates that the blocks in a web page that does not include content related to the main content can easily harm the data mining tasks such as web page classification and clustering. *[25]* aims to eliminate the noisy blocks from a web page to improve the performance of data mining tasks such as web page classification and categorization.

Semantic Annotation : *[26]* indicates that most web pages on the web are encoded for human consumption and they are not designed for machine processing. *[26]* aims to identify segments that correspond to semantic concepts. In this respect, the overall idea is not about segmentation of a web page into cohesive number of blocks, but rather identify blocks that include semantic concepts.

Caching : Considering fragments of web pages for caching proved to be significant benefits for caching. Especially for dynamic web pages, identifying fragments that are shared between pages and have different life time can improve the efficiency of caching.

Archiving : In order to maintain a web archive up-to-date, crawlers harvest the web by iteratively downloading new versions of documents, *however most of the time they retrieve pages with unimportant changes such as advertisements which are continually updated. Hence, web archive systems waste time and space for indexing and storing useless page versions.* Furthermore, they indicate that querying such archive can be expensive because of the large amount of useless data stored.

1.3 Thesis Structure

In Chapter 2 we give an overview of the research. We discuss the problem and present the research question. We also introduce the methods we used in our evaluation. The definitions of the basic concepts are given as well as a more in-depth overview of the different approaches to page segmentation.

Chapter 3 is a literature review where we give a short summary of the papers we found the most relevant to our research. We highlight the different approaches and methodologies used to evaluate the results.

In Chapter 4 we focus on datasets where we look at how other authors chose their samples, what was included in those samples and how they marked up the blocks. We then also give the details of the dataset we used in our performance comparison.

Chapter 5 introduces the testing framework we developed to compare the different segmentation algorithms. We talk about the requirements, the architecture and the implementation of the software.

In Chapter 6 we describe the *BlockFusion [23]*, *Block-O-Matic [30]* , *VIPS [24, 25] and Gomory-Hu Tree based Clustering [22]* algorithms. We present the different approaches each of these algorithms to tackle the problem and we talk about the implementation of the ones we implemented ourselves. We also show how they are integrated into the testing framework.

Chapter 7 is a summary of the empirical results we got for all four algorithms.

These results are discussed in Chapter 8 and

We draw conclusion in Chapter 9.

Chapter 2

Research Outline

2.1 Problem Discussion

In this thesis we have tried our level best to answer the following research questions :

How well do existing Web page segmentation algorithms work on modern Web sites and can they be improved?

Sub-questions :

The research question leads to a number of sub-questions that need to be answered in order to answer the question itself:

1. *What are the current best-performing algorithms tackling this problem?*
2. *What assumptions do those algorithms make and are they still valid?*
3. *How do they define a "semantic block"?*
4. *In which scenarios does their approach work well and in which not?*
5. *How can you evaluate each of these different approaches effectively?*
6. *How can you compare these different approaches effectively?*
7. *How can these approaches be improved?*

2.2 Our Approach

We first conducted a literature review to find the most promising looking algorithm from each of the three different approaches described in section 3.1. We analyze the theoretical advantages and disadvantages of them.

We consider each one a representative of the class of algorithms which follow that approach and we implemented many of them . For some of them we use existing code from the authors and some of the algorithms needed little bit customization too. We implemented two of the four algorithms ourselves since reference implementations for all but one were unfortunately not available.

The main contribution is then an empirical evaluation of these four algorithms on two different datasets taken from *A Quantitative Comparison of Semantic Web Page Segmentation Algorithms (Master's Thesis in Computer Science by Robert Kreuzer).*

As metrics for the performance of an algorithm *we chose precision, recall and F-score.* So we look at how many of the blocks returned by an algorithm are correct, how many of the correct blocks are recalled and at a combination of both measures.

2.3 Segmentation Vocabulary: Terms Used in the Literature

Different published work use different terms to refer to a segment in a web page. The following list shows the vocabulary used in this literature.

The acronym DOM stands for the Document Object Model, which is a tree like representation of XML and HTML documents as objects. It allows to query and manipulate the tree via a standardized application programming interface (API).

We use the term Web page when referring to a single HTML document. The term website is used for a collection of Web pages found under the same top-level URL.

When referring to the ground truth, we mean the set of semantic blocks on a page that have been manually marked up by our assessors. These are considered the true blocks to which the results of the algorithms are compared.

Web elements *refers to components of web pages* as Web elements. A web page is made up of hundreds of basic elements. *The functional role of each element is different.*

We use image as an example, an image can be a banner, advertisement or a picture of a news article.

Logical unit/section *refers to sections of web pages as semantically related group of elements.* It uses the term section to refer to segments of web pages.

Block refers to as information blocks *which is defined as closely related content, each of which forms topic within the web page*. We further classify the blocks as *atomic (non-dividable block) and composite (can contain other composite or atomic blocks).* For example - header, footer, sidebars and body are the higher-level content blocks. It defines a block as the semantic part of the web page. *It is basically a group of web contents that share a coherent topic, style and/or structure. Authors also indicate that they use the term web content to refer to the smallest indivisible content of a web page which is usually represented as a leaf node in the DOM tree.*

Semantic block *A contiguous HTML fragment which renders as a graphically consistent block and whose content belongs together semantically.*

Note that this definition allows for a hierarchy of blocks, i.e. bigger blocks can be made up of a collection of smaller blocks (e.g. the main content on a news website can be made up of a list of articles). In principle there is no limit to the level of nesting of blocks, but in practice it is rarely more than two or three levels.

Sub-page *views a web page as a collection of different small web pages structured together in a layout.* A web page can be considered as a collection of sub-pages/blocks, etc .

Segments "is a fragment of HTML, which when rendered, produces a visually continuous and cohesive region on the browse window and a has a unified theme in its content and purpose".

Granularity Of Segmentation We will call the *number of levels of the block hierarchy the granularity of the segmentation.* In the following we will use the words semantic blocks and segments interchangeably. *A segmentation with a granularity of one is a flat segmentation, since there are no blocks that are subsets of other blocks (all blocks are disjoint).*

Component refers to the blocks of web pages as web page components. Authors define web page components as "basic units for transcoding which can be extracted through syntactic analysis of the page's HTML source code".

Coherent unit refers to blocks as coherent units.

Coherent region refers to blocks as coherent region.

Objects refers to them as segments but they are used interchangeably. Authors define a basic object as "the smallest unit of a web page which cannot be further subdivided" and they define a composite object as " Authors group basic objects together to achieve a major function such group of objects is called composite objects, composite objects can then be grouped together onto a more complex one".

Page unit/data unit/context unit defines data units as web page fragments ranging from a single word to a complete page.

Fragment "a fragment is a portion of a web page which has a distinct theme or functionality and is distinguishable from the other parts of page". Web pages are constructed from simpler fragments and fragments may recursively embed other fragments.

Pagelet semantic definition- "a pagelet is 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". syntactic definition - "An HTML element in the parse tree of a page p is a pagelet if (1) none of its children contains at least k hyperlinks;(2) none of its ancestor element is a pagelet".

2.4 Overview of Approaches to Web Page Segmentation

Given short summaries of different approaches to web page segmentation and their advantages, disadvantages.

2.4.1 DOM Based Approaches (Structure Based) [40, 42]

In a DOM based approach one simply looks at the DOM (i.e. the tree built by parsing the HTML. Note that this does not include the properties added by potential external CSS files; these will only be present in the render tree) for cues how to divide a page. The idea behind this is that the HTML structure should reflect the semantics of the page, but this is of course not always the case. *The quality of these approaches thus*

depends on the quality of the underlying HTML. To do the segmentation they rely on detecting recurring patterns, such as lists and tables, and on heuristics, like e.g. headline tags working as separators and links being part of the surrounding text.

Advantages

- easy to implement, since one only needs to parse the HTML code and not render the tree
- efficient to run because no browser engine is involved, thus suitable for segmenting large numbers of pages
- take the structural information into account
- Depth and type of the node in the DOM gives an indication of how much it should be kept together.
- May capture many of the benefits of the vision based techniques, as if there is a background that is different, it can appear in the DOM.
- it is simple and scalable

Disadvantages

- based on the assumption that the HTML document reflects the semantics of the content, which is not necessarily true
- there are many different ways to build the HTML document structure while the semantics stay the same
- disregard styling and layout information by design
- do not work with pages built via Javascript (or you have to serialize the DOM in that case first)
- Examining only DOM elements may not capture obvious visual cues and may make distinctions between regions that appear similar.
- Further, this technique may require some basic computer vision technique to solve uncertainties of where to place cuts between DOM nodes.
- Some elements are very closely located in the HTML source code (for example, two cells in a table element) but they are visually displayed very far apart from each other in the visual rendering .
- indicates that since most people do not obey W3C specification, there can be a lot of mistakes in the DOM tree.
- Most authors also do not use DOM for encoding the semantic structure of the web page, for example two elements might have the same parent, but content wise (semantically) they might be very related.

• even though XHTML is introduced as an XML extension of HTML which can be used for semantic encoding, not many people use it.

• because of the heterogeneity of HTML style, algorithms are susceptible to failures.

2.4.2 Visual Approaches (Layout-Based) [43, 47, 48, 49]

Visual approaches work the most similar to how a human segments a page, i.e. they operate on the rendered page itself as seen in a browser. They have thus the most information available but are also computationally the most expensive. They often divide the page into separators, such as lines, white-space and images, and content and build a content-structure out of this information. They can take visual features such as background color, font size and type and location on the page into account.

Advantages

• take the styling and layout information into account

• work similar to how a human performs the task

• can take implicit separators such as white-space and vertical/horizontal lines into account

• Better at keeping salient portions together, especially when the portions are noticeably different colors (e.g, a region is highlighted or has a different background)

Disadvantages

• Vision based approaches naturally have a higher complexity since the layout must be rendered prior to analysis, which might be too slow to be incorporated into the web crawling and indexing cycles .

• computationally expensive because the page needs to be rendered

• requires external resources such as CSS files and images to work correctly

• Does not take into account the DOM of the web page, therefore no notion of which regions are more important to keep together, will not work on simple pages, if the regions are not visible different, this method may not work as well.

2.4.3 Text Based Approaches (Content Based) [46,56]

The text-based approaches differ from the other two in that they do not at all take the tree structure of the HTML into account. They only look at the text content and analyze certain textual features like e.g. the text-density or the link-density of parts of a page. These techniques are grounded in results from quantitative linguistics which indicate that statistically text blocks with similar features are likely to belong together

and can thus be fused together. The optimal similarity threshold depends on the wanted granularity and needs to be determined experimentally.

Advantages

- fast, since the DOM does not need to be built
- easier to implement, since no DOM access is necessary
- comparative performance to other approaches

Disadvantages

- do not work with pages built via Javascript (or you have to serialize the DOM in that case first)
- do not take structural and visual clues into account
- recursive application on sub-blocks requires (arbitrary) changes to the text-density threshold

2.4.4 Other Approaches [44, 45, 50, 52, 53]

Hybrid approach (both DOM interface and visual rendering in short) uses both structure based, content based, layout based information of a web page to effectively segment a web page.

Features

- computationally expensive because the page needs to be rendered and all structure based, content based, layout-based information need to be collected in the page preprocessing step.
- more accurate since it includes the goodness of Structure based, Content Based, Layout Based approach.

Chapter 3

Literature Review

Given a short summary of the papers we consider the most relevant to our research and pointed out the many different approaches that have been tried to solve the Web page segmentation problem. We also focus on whether authors did an empirical evaluation of their algorithms, on what kind of dataset this was done and what performance metrics were chosen.

3.1 Related Work

a) In ***Semantic Partitioning of Web Pages [40]*** the authors develop an algorithm that *uses the structural and presentation regularities of Web pages to transform them into hierarchical content structures (i.e. into "semantic blocks").* They then proceed to tag the blocks automatically using an abstract ontology consisting of Concepts, Attributes, Values and None (their approach is structural and requires no learning). Their main idea behind the labeling is that content is already organized hierarchically in HTML documents, which can be used to group things. *They test their results experimentally against the TAP knowledge base [54] (which seems to be not available anymore at this time unfortunately) and with a home-made dataset consisting of CS department websites.*Technically their approach can be split into four parts:

They first partition Web pages into flat segments (i.e. disjoint blocks without a hierarchy) by traversing the ***DOM [41]*** *and segmenting content at the leaf nodes.* They define a block as a "contiguous set of leaf nodes in a Web page, where the presentation of the content is uniform."

The second step is inference of the group hierarchy. They use an interesting solution here, where they first transform each segment into a sequence of path identifiers (using the root-to-leaf path of the tree nodes), which are then interpreted as a regular expression, in which each Kleene star represents a separate group.

They then proceed to determine labels of groups, where they use the leaf tag right above a certain group if there is only one or a learning-based approach if there is more than one.

The final step is the meta-tagging of groups, where a Concept defines the general context of a group, an Attribute corresponds to a group label and a Value is an instance of an attribute (i.e. a group member).

b) In ***Identifying Primary Content from Web Pages and its Application to Web Search Ranking [43]*** the authors also first *segment a Web page into semantically coherent blocks. They then rate the individual blocks by learning a statistical predictor of segment content quality and use those ratings to improve search results.*

The segmentation is done using a widening approach, that first considers each DOM node to be its own segment and subsequently joins it with neighboring nodes according to certain heuristic rules. The classification of blocks relies on a machine learned approach where the feature space contains both visual and content properties. The content is simply classified into content segments or noise segments.

c) In ***Recognition of Common Areas in a Web Page Using Visual Information: a possible application in a page classification [43]*** the authors present an approach to extracting semantic blocks from Web pages which is based on heuristics that take visual information into account. *To get access to information such as the pixel sizes of elements on a virtual screen and their position on the page they build their own basic browser engine.* They make some simplifications such as not taking style sheets into account and not calculating rendering information for every node in the HTML tree.

They then define a number of heuristics on the render tree and on the assumption that the areas of interest on a page are header, footer, left menu, right menu and center of the page.

The authors test their algorithm experimentally by first building a dataset where they manually label areas on 515 different pages, then run their algorithm on the dataset and subsequently compare the manual and algorithmic results. Their overall accuracy in

recognizing targeted areas is 73%.

d) In **A *graph-theoretic approach to web page segmentation [44]*** the authors first turn the DOM tree of an HTML document into a graph where every DOM node is also a node in the graph and every node has an edge to every other node in the graph. Each edge is then assigned a weight that resembles the cost of putting these two nodes into the same segment. *The weights are learned from a dataset regarded as the ground truth by looking at predefined visual and context-based features. Finally they group the nodes into segments by using either a correlation clustering algorithm or an algorithm based on energy-minimizing cuts, where the latter is performing considerably better in their empirical evaluation.*

Their evaluation is based on manually labeled data (1088 segments on 105 different Web pages). They also test their algorithm by using it for duplicate detection, where they achieve a considerable improvement over using the full text of a page for that purpose.

For defining features to determine the likelihood of two nodes belonging to the same segment they give three contexts in which DOM nodes can be viewed: *First each node can be looked at as a sub-tree containing a set of leaf nodes with certain stylistic and semantic properties. Second the visual dimensions and location of a node give cues about its semantics and relationship to other nodes. Third the particular DOM structure used by the content creators also implies semantic relationships.*

e) In ***Segmenting Web page with Gomory-Hu Tree Based Clustering [45]*** *.This is basically graph-theoretic based approach for web page segmentation that uses visual, structural and content based features of a web page.*

In this algorithm the authors formulate the problem of web page segmentation to the clustering problem on an undirected graph, in which the vertices are leaf nodes of the DOM tree, and the edges represent the neighborhood relationship between vertices when they are rendered on the screen. *The graph constructed by both vision and structure information is a planar graph*, on which most optimization problems are much easier than those on general graphs.

They deploy the Gomory-Hu Tree based algorithm to solve the underlying clustering problem in our formulation. *The use of Gomory-Hu Tree based algorithm not only because it has theoretical clustering quality guarantees, but also because it can*

benefit from the graph's planarity to gain very efficient algorithm.

To evaluate the proposed algorithm, authors did experiments on 100 pages with 2726 blocks. The dataset is crawled automatically and the blocks are manually segmented.

Experimental results show that, it outperforms both VIPS and Chakrabarti et al.'s algorithm in terms of precision and recall, and is faster than Chakrabarti et al.'s graph theoretic algorithm.

f) In ***A densitometric approach to web page segmentation [46]*** the authors use an approach that makes use of the notion of text-density as their main heuristic. Instead of analyzing the DOM tree, like many other authors, they instead focus on discovering patterns in the displayed text itself. Their key observation is that the density of tokens in a text fragment is a valuable cue for deciding where to separate the fragment.

In detail, they look at the HTML document itself and first divide it into a sequence of atomic blocks (text that does not contain HTML tags) separated by what they call gaps (the HTML tags between text blocks). They then discern so-called separating gaps from non-separating ones by analyzing the properties of the two text blocks adjacent to the gap. The deciding property is the text flow which they characterize by the density of the text in a block (defined by the number of tokens in a block divided by the number of lines in that block). If the density between two blocks does not differ significantly (less than a predefined threshold value) they will then be fused together into a new block by the proposed *BlockFusion* algorithm. This is repeated recursively until all further fusions would be above the threshold.

They evaluate their approach experimentally using a dataset consisting of 111 pages.

g) In ***Vision-based Page Segmentation (VIPS) [47]*** As indicated by the name this algorithm *is based on the rendered representation of a Web page. So it not only takes the DOM structure into account but rendered properties such as dimensions on the screen,background color etc. as well.*

The VIPS algorithm was designed to segment Web pages similarly to how humans do it. It thus analyzes the DOM after all the styling information from CSS rules have been applied and after Javascript files were executed (and potentially modified the

tree). It is tightly integrated with a browser rendering engine since it needs to query for information such as the dimensions on screen of a given element. One thus has to decide on a fixed viewport size in advance on which the page should be rendered. *Concretely the algorithm first develops a vision-based content structure,which is independent of the underlying HTML document.* This structure is built by splitting a page into a 3-tuple consisting of a set of visual blocks, a set of separators and a function that describes the relationship between each pair of blocks of a page in terms of their shared separators. Separators are for example vertical and horizontal lines, images similar to lines, headers and white-space. *This structure is built by going top-down through the DOM tree and taking both the DOM structure and the visual information (position, color, font size) into account.* Specifically they decide for each node whether it represents a visual block (i.e. the sub-tree hanging on that node) or whether it should be subdivided further by using a number of heuristics:

- *if a sub-tree contains separators like the <hr> tag, subdivide*
- *if the background color of the children of a node differ, subdivide*
- *if most of the children are text nodes, do not divide*
- *if the size of the children differs substantially, subdivide*

They detect the set of separators visually by splitting the page around the visual blocks so that no separator intersects with a block. Subsequently they assign weights to the separators, again according to certain predefined heuristic rules. From the visual blocks and the separators they can then assemble the vision-based content structure of the page.

h) In ***Vision Based Page Segmentation: Extended and Improved Algorithm [48]*** the authors improve upon the popular VIPS algorithm. They focus on improving the first part of VIPS, the visual block extraction (part 2 and 3 of VIPS are visual block separation and content structure construction respectively).

They observe that the original algorithm now has certain deficiencies due to its age (it is from 2003) and the evolving nature of the Web. *They address these issues by dividing all HTML tags (including the ones introduced by HTML 5) not into three classes but into nine instead and define new separation rules for these classes based on visual cues and tag properties of the nodes. Unfortunately they do not give an empirical*

evaluation of their updated algorithm.

i) In ***Yet Another Hybrid Segmentation Tool (Andrés Sanoja , Stephane Gançarski) [49]*** It analyzes pages based on their DOM tree information and their visual rendering. *This tool implements a modified version of VIPS with the aim of enhancing the precision of visual block extraction and the hierarchy construction, from the user perspective*.

First, the visual rendering of a page, produced by several browsers, is segmented into rectangular blocks

Then, the extracted blocks are analyzed looking for visual overlaps, which are analyzed using a adapted version of the *XY-Cut algorithm and computational geometry methods and resolve the overlap*.

As a result we may have different shapes of blocks, rectangular and non-rectangular blocks (polygons).

Finally, the visual block tree, representing the layout of the page is analyzed in order to have a more coherent layout disposition.

j) In ***Browsing on Small Screens: Recasting Web-Page Segmentation into an Efficient Machine Learning Framework [50]*** the author uses a different approach than most others, because he focuses on the application of optimizing existing Web pages for mobile phones.

He first divides a Web page into a 3x3-grid, where the user has to choose one grid he would like to see better by clicking on it. The selected part can then either be zoomed further into or viewed as rendered HTML or transcoded into a view optimized for the specific device. *Their page segmentation algorithm is based on clues from the DOM combined with a number of computer vision algorithms. Specifically they use an entropy measurement to construct a decision tree that determines how to segment the page.*

They first recursively segment the page by cutting it based on entropy (trying to reduce the entropy of split parts).

They combine this with a few heuristics e.g. favoring cuts that result in more equally sized parts or preferably cutting in nodes which are higher up in the DOM tree.

They test their approach on a number of popular websites where they achieve good results in most cases . One artificial limitation of their approach is that it is designed to divide the page into at most 9 segments (because they assumed that people

can then choose one part by clicking the respective number on their mobile phones), but it seems possible to adapt it to other grid sizes.

k) In ***Web Page Segmentation Based on Gestalt Theory [52]*** and ***Enhanced Gestalt Theory Guided Web Page Segmentation for Mobile Browsing [53]*** authors intended to simulate human's perceptive process guided by four general laws in Gestalt theory, namely: *proximity, similarity, closure and simplicity.* When perceiving a Web page, human unconsciously follow the four laws and segment it into several hierarchical parts by integrating various cues presented in the page.

The algorithm unfolds in 3 parts :

a) ***Page-preprocessing :-*** *extract out six main features basic-type, class, text, style, location, layout of each DOM node. Eliminate noisy nodes.*

b) ***Grouping :-*** *Based on the basic feature tree, group visually and semantically coherent contents together guided by similarity, closure and simplicity laws in Gestalt theory.*

c) ***Dividing :-*** *The re-constructed tree structure of Step b is processed top-down to divide spatially apart content into different nodes, guided by proximity law, and finally generate a set of blocks fitted for limited screen size.*

Authors compare segmentation results of E-GESTALT with VIPS and GESTALT . Each block in the resulting block set is manually classified into three levels: *ERROR (L1), NOT-BAD (L2) and PERFECT (L3).*

E-GESTALT produces the most PERFECT blocks and the least ERROR blocks. The count of NOT-BAD blocks is much higher in other two methods, mainly because: large blocks tend to be split into sub-blocks much smaller than screen size by VIPS and GESTALT.

l) There is also a review ***Web Page Segmentation: A Review [51]*** about all the different approaches to page segmentation attempted so far. The author systematically goes through the literature and answers the five W's (Who, What,Where, When and Why). They look at about 80 papers that are in one way or another related to page segmentation.

As applications for segmentation they list mobile web, voice web, web page phishing detection, duplicate deletion, information retrieval, image retrieval, information extraction, user interest detection, visual quality evaluation, web page

clustering, caching, archiving, semantic annotation and web accessibility.

The approaches for the segmentation itself can be broadly split into bottom-up vs top-down algorithms. Other differentiators are whether algorithms just look at the DOM or at the visual representation of the page or both. Some algorithms attempt to recognize blocks by first looking for separators such as thin lines or white space. Many algorithms are based on heuristics where the authors make assumptions about the "general layout" of a page.

The review also gives an overview of the assumptions and limitations of the different algorithms (unfortunately not in a table though).

In the evaluation of the different approaches there is also a wide variance:
Many authors use precision and recall as metrics of effectiveness, others use success rate or accuracy and some also focus on execution time or output size. One thing that the review also shows is that there seems to be no easy way to compare the different approaches to each other. There seems to be no standardized test for the effectiveness of different page segmentation algorithms. Each paper seems to use its own datasets and test procedures which often have different parameters and goals. It is thus no easy task to decide upon one particular algorithm for a practical purpose at hand.

Chapter 4

Datasets

4.1 Overview

There seems to be no easy way to compare the different approaches to each other. There seems to be no standardized test for the effectiveness of different page segmentation algorithms. Web page segmentation is a fairly well-studied problem, many authors have done an empirical evaluation of their algorithms Each paper seems to use its own datasets and test procedures which often have different parameters and goals. The datasets and methods used for this evaluation vary widely. There appears to be no standard dataset for this problem, instead every author seems to create their own dataset by first randomly sampling Web pages (sometimes taken from a directory site such as **http://dmoz.org**) and then manually marking up the semantic blocks and often also labeling the blocks according to a predefined ontology. It is thus no easy task to decide upon one particular algorithm for a practical purpose at hand.

Marking up a semantic block and subsequently labeling it are two distinct steps. *Marking up a semantic block can be thought of as taking a picture snapshot of a Web page and then drawing a rectangle around all the areas on the page that belong together semantically.*

Labeling is the subsequent step where a label, typically describing the function of a block, is applied to each one. The labels used depend on the particular ontology chosen by the authors. These range from a simple noise/no-noise classification to more involved ones containing e.g. header, footer, right- and left-menu, advertisements, content, comments etc.

While the first step is certainly less ambiguous there still remains the question

whether there is an intuitive understanding of *what constitutes a semantic block among different persons*. One do have to be s*pecific about the level of granularity he wants* (e.g. “the most high-level (i.e. biggest) blocks and their most high-level sub-blocks”), since there can be many levels. We find that in practice a block-hierarchy with two levels is sufficient for applications such as information retrieval, information extraction and mobile page-adaptation. *We will therefore restrict ourselves to hierarchies of two levels here.*

4.2 Dataset used by Authors

Paper	**Sample taken from**	**Size (no of pages)**	**Type**	**Created by**	**Available**
Semantic Partitioning of Web Pages[40]	a)TAP knowledge base[54] b)CS department websites	a) 9, 068 b) 240	a) Template driven b) non-template-driven	a) external b) 8 volunteers	No
Recognition of Common Areas in a Web Page Using Visual Information [43]	Random sites from http://dmoz.org	515	all	the authors	No
A densitometric approach to web page segmenta-tion [46]	Webspam UK-2007[55]	111	all	external	No
Graph-theoretic approach to segmentation[44] / Segmentation Based on Gestalt theory [52,53]/ machine learning based segmentation[50]/BlockOMatic [49]/Gomory[45	Random Web pages [(Yahoo, MSN), Google search , Wikipedia , Froogle, Images, Blog pages, eCommerce sites, and university home pages]	105/ 120/ 160/ 111/100 pages with 2726 blocks	all	The authors	No
VIPS and VIPS Improved[47,48]	Pages from Yahoo directory (http://www.yaho o.com)	140	all	The authors	No

Table 4.1: Datasets used in the literature

4.3 Dataset Description

We evaluate the performance of the algorithms using two different datasets taken from *A Quantitative Comparison of Semantic Web Page Segmentation Algorithms (Master's Thesis in Computer Science by Robert Kreuzer).*

4.3.1 External resources in Web pages

Web pages are not only made up of HTML documents but they can reference a number of external resources that the browser will download in order to render the page properly. The most common ones are images, videos, CSS files (which describe how a page should be styled) and Javascript files (often adding interactivity and animations to a page) but there are also lesser known ones like font files, JSON or XML files (providing raw data), favicons and vector graphic files.

A browser will first download the HTML document itself, parse it into the DOM tree (i.e. an object representation of the document) and then find all the referenced resources and download those as well. If there are one or more external style sheets they will be parsed as well, and the style rules will be applied to the DOM. Finally if there were Javascript files found they will be interpreted by the Javascript engine built into the browser and they will apply arbitrary transformations to the DOM. Finally a render tree can be built from the DOM which is then painted on the screen.

So it is clear that if you only download the HTML document itself then its rendered representation will be vastly different from the page including all resources. For this reason we decided to build a dataset consisting of HTML documents together with all their external resources (and all links rewritten accordingly so that they point to the right files).

4.3.2 Downloading the sample pages

wget -U user_agent -E -H -k -K -p -x -P folder_name -e robots=off the_url

Downloading of sample pages is done with wget using finely tuned parameters.

4.3.3 block ontology

High-level-blocks - Header, Footer, Content, Sidebar

Sub-level-blocks - Logo, Menu, Title, Link-list, Table, Comments, Ad, Image, Video, Article, Search-bar

This block ontology was chosen with the goal of being comprehensive and it was divided

into High-level blocks and Sub-level blocks (or level 1 and level 2 blocks) since Web pages can be segmented on different levels of granularity. E.g. a content-block can have a title, an image and an article as sub-blocks. While in principle there is no upper limit to how many levels of granularity you can have on a page, we found two levels to be sufficient in the majority of cases and have thus restricted ourselves to that.

4.3.4 Dataset Details

Authors include all static resources such as images, CSS files and Javascript files as well, so that they can be rendered offline as they are seen online. The links were rewritten to point to the local resources. Furthermore is each page available in three versions: One with just the basic HTML as can be obtained by a single GET request to a URL, and second as a serialized version of the DOM after all external resources were loaded. Finally there is a version of the DOM-pages which have manually marked up semantic blocks, which was done by a number of volunteers.

4.3.4.1 Random Dataset :

URL- *https://github.com/rkrzr/dataset-random*

Overview-It contains a set of random webpages that were downloaded using `wget`. *For the random websites authors made use of the web service from http://www.whatsmyip.org/random-website-machine/ to generate 100 links, which they then downloaded.* The service boasts over 4 million pages in its database and the only filtering is done for adult content, which makes it sufficiently representative for the Internet as a whole. *After removing pages that did not render properly offline they ended up with a random dataset consisting of 82 pages.*

4.3.4.2 Popular Dataset :

URL- *https://github.com/rkrzr/dataset-popular*

Overview-It contains a set of popular webpages taken from dir.yahoo.com that were downloaded using `wget`. containing randomly selected pages. *For the popular dataset authors took the top 10 pages from the 10 top-level categories from the directory site http://dir.yahoo.com/. The chosen categories were Arts & Humanities, Business & Economy, Internet, Entertainment, Government, Health, News & Media, Science .*

Chapter 5

Testing Framework

Here is an overview of the testing framework which we developed to compare different segmentation algorithms on different datasets.

5.1 Overview

Conceptually the framework takes a segmentation algorithm and a dataset as input and produces statistical results as output. The statistics are computed by running the given algorithm on each item of the dataset and comparing the resulting segmentation to the ground truth given in the dataset.

The concrete metrics used are Precision, Recall and F-Score of the segmentation.

In essence, all a segmentation algorithm should need as its input is the HTML of the page (and the external resources referenced from the HTML) it is supposed to analyze. It will then return the recognized blocks in some form which will then have to be normalized to a common structure that can be easily compared to the reference dataset. For the framework we defined the expected outcome as a HTML document which contains the blocks marked up with the additional CSS attributes data-block and data-block-type whose value is the block level (1 or 2) and the block type respectively.

It is thus necessary to provide a mapping function for each algorithm, that takes the output of the algorithm and maps it to the expected format. Furthermore it can be necessary to provide a small driver function if an algorithm is written in a different language or is a Web service for example. The driver takes the original HTML as input

and needs to interface with the algorithm and return the segmentation result. The result, together with the ground truth, is then fed to the statistics component which calculates the evaluation result, which is then presented to the user.

5.2 Architecture

DatasetGenerator :: Dataset → Iterator (HTML Pages, HTML Ground truth)
AlgorithmDriver :: HTML Page → Algorithm → Blocks
BlockMapper :: HTML Page → Blocks → Block HTML
Evaluator :: HTML Ground truth → Block HTML → Statistical Results

Figure 5.1: The pipeline components and their interfaces

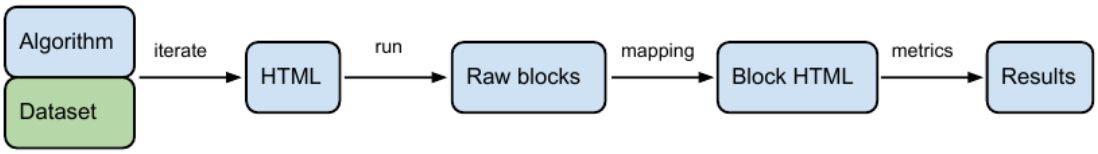


Figure 5.2: The data pipeline

The testing framework uses a pipeline as its main design pattern as pictured in Figure 5.2. As inputs it takes the algorithm and the dataset specified by the user and it generates a statistical evaluation of the performance of the algorithm on that dataset as output.

The pipeline has four distinct components which have clearly defined interfaces (Figure 5.1).

a) The first component is the DatasetGenerator which is simply a function that knows how to retrieve the original HTML documents and the HTML documents with the manually highlighted blocks (the ground truth).

b) The original HTML document is then fed to the AlgorithmDriver, which is a small function unique to each algorithm, that knows how to apply the specified algorithm to the given document. This function needs to be specific to each algorithm since algorithms can be implemented as libraries, executables or Web services, which is unknown in advance. The driver can then interface with the algorithm by some means like e.g. a sub-process or an HTTP request and return the extracted blocks.

c) Since there again is no unified format for semantic blocks and different algorithms return different formats it is necessary to normalize the data representation of

the blocks. The BlockMapper component takes care of this. It takes the raw blocks, which can for example be only the textual content that has been extracted or fragments of the HTML, and maps them back onto the original HTML document to produce our standard format. For this standard we decided to use the original HTML where the semantic blocks have been marked up with the two additional attributes data-block and data-block-type.

d) Finally there is the Evaluator component that takes the normalized Block HTML and the HTML ground truth as inputs and does the statistical evaluation of the algorithm results. It calculates Precision, Recall and F-score by getting the blocks from both documents and checking which match. It also returns the number of blocks retrieved by an algorithm, the number of correctly found blocks (the hits) and the total number of relevant blocks on a page. The equality of blocks is tested with two different metrics: an exact match metric and a fuzzy match metric. For both metrics the blocks are first serialized to text-only (i.e.all tags are ignored) and white-space and newline characters are removed as they just add noise. The exact match metric then does a string equality check to find matching blocks (the intersection of the set of found blocks and the ground truth is taken). The fuzzy match metric does a fuzzy string comparison using the SequenceMatcher algorithm in the Python difflib library . We consider strings with a matching ratio > 0.8 as equal for the fuzzy match metric.

Chapter 6

Segmentation Algorithms

Here are the details of the algorithms which we use in our performance analysis .

6.1 BlockFusion (Boilerpipe) :

The BlockFusion algorithm was introduced by Kohlschütter and Nejdl in 2008 [46]. *A Densitometric Approach to Web Page Segmentation.* The author who later wrote *Boilerplate Detection using Shallow Text Features*, which later turned into *Boilerpipe, [56] one of the best (most certainly the quickest) libraries web content extraction.* It is basically text-based densitometric approach to web page segmentation.

It is thus a relatively recent algorithm which is distinctive in that it is completely text-based. It does not rely on the DOM tree and can be implemented very efficiently, making it potentially useful for the segmentation of a large number of pages.

6.1.1 BlockFusion Overview

The BlockFusion algorithm is grounded in the observation, coming from the field of Quantitative Linguistics, that the so-called token density can be a valuable heuristic to segment text documents. The token density of a text can simply be calculated by taking the number of words in the text and dividing it by the number of lines, where a line is capped to 80 characters. A HTML document is then first preprocessed into a list of atomic text blocks, by splitting on so-called separating gaps, which are HTML tags other than the <a> tag. For each atomic block the token density can then be computed. The authors then employ a merge strategy adapted from a Computer Vision algorithm which

merges adjacent pixels, but instead they merge neighboring text blocks.

The idea is basically this:

- *walk through nodes*
- *assign a text density to each node -> number-of-tokens / number-of-'lines'*
- *merge neighbor nodes with the same densities*
- *repeat until desired granularity is reached*

They compare the token density of each atomic block to the density of its successor and if the difference between the two (the slope delta) is below a certain threshold value θmax they are merged into a bigger block. This merge strategy is done recursively until no more blocks can be merged. Due to this design this algorithm does not support multiple levels of blocks by default, but a different block granularity can of course be achieved by changing the merge threshold value. A possible extension of this algorithm to support sub-blocks is to introduce a second smaller threshold value ιmax and then call the algorithm on each (already merged) block. ι max needs to be smaller than θmax,otherwise the sub-blocks would be merged until they are identical to the main blocks.

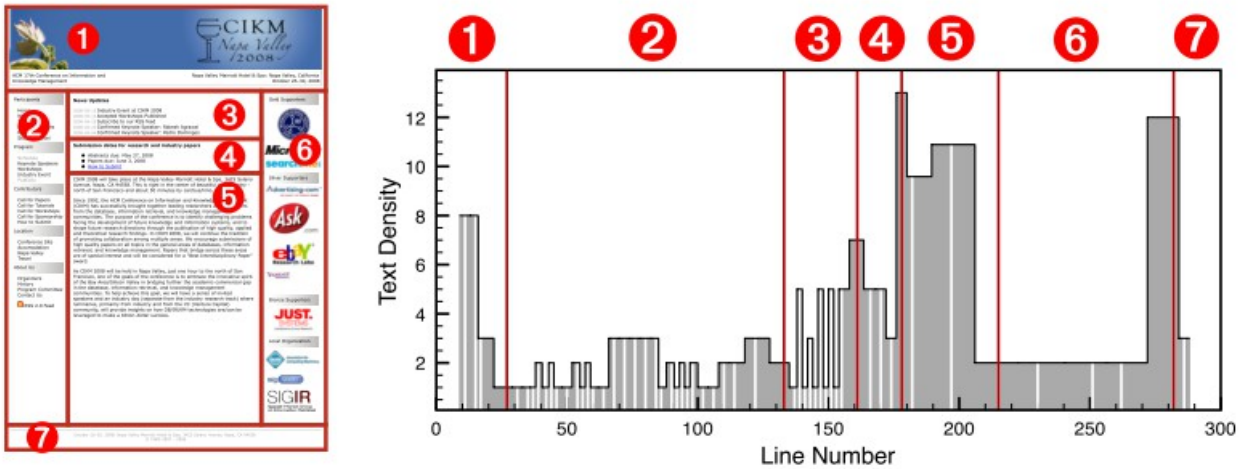


Figure 1: Visual vs. Densitometric Segmentation (expected results)

Figure 6.1 Densitometric Segmentation Of Web Page

Algorithm 6.1.2 The BlockFusion Algorithm

```
# Input : textBlocks <− set of atomic blocks which partition the lines
# Output : Set of Text blocks
thetaMax = 0 . 3 8
# Empirically determined slope threshold
def blockFusion(textBlocks) :
        if len(textBlocks)< 2:
                return textBlocks
        notDone = True
# merge b l o c k s u n t i l no more can be merged
        while notDone :
                notDone = False
                block1 = textBlocks[0]
                for block2 in textBlocks[1: ] :
                        if slopeDelta(block1,block2 ) <= thetaMax :
                                block1.merge(block2) # merge block2 into block1
                                textBlocks.remove(block2) # remove block2 fro
                                notDone = True
          else :
                block1 = block2
                return textBlocks
# The text density difference between two blocks
def slopeDelta(block1 , block2 ) :
        d1 = getTextDensity(block1)
        d2 = getTextDensity(block2)
        return abs(d1-d2) / max(d1,d2)
```

6.1.3 BlockFusion Implementation

While there is no reference implementation of the BlockFusion algorithm as it is given in [46], there is a related open source library from the same author, called boilerpipe [1], which is described in [56]. We implemented the BlockFusion algorithm (specifically the BF-plain variant as described in [56]) on top of this library, since this allowed us to stay as close to the original implementation as possible because we could

reuse many functions needed for the algorithm. For example the function that generates the atomic text blocks, which are later merged, is taken unmodified from the boilerpipe library as well as the block merging function and the function to calculate the text density of a block. We also used a text density threshold value of $\theta max = 0.38$, which the authors found to be the optimal value in their experimental evaluation.

The boilerpipe library itself is focused on retrieving the main content, like for example an article, from a Web page by removing all so called “boilerplate”, i.e. everything that is not the main content, from Web pages. It provides different extractors (strategies) to accomplish this, based on the specific extraction goal. We could therefore simply extend the library with a new BlockExtractor that implements the BlockFusion algorithm and returns the extracted text blocks.

We used an existing (minimally modified) Python wrapper for the library to interface with the algorithm[5]. As the blocks are returned as a list of strings they then still need to be mapped back onto the original HTML document. We therefore walk through the parsed HTML tree and search (fuzzily) for the text fragments. When a block is found we add the block CSS attributes (and a wrapping div element, if necessary). The thus marked up page is then passed on to the Evaluator.

6.2 VIPS :

The VIPS algorithm [47] (*Vision-based Page Segmentation*) appears to be the most popular Web page segmentation algorithm since many other papers reference it or compare their results to it. As indicated by the name this algorithm *is based on the rendered representation of a Web page. So it not only takes the DOM structure into account but rendered properties such as dimensions on the screen,background color etc. as well.* The algorithm is from 2003 and thus the oldest in our comparison.

6.2.1 VIPS Overview

The VIPS algorithm was designed to segment Web pages similarly to how humans do it. It thus analyzes the DOM after all the styling information from CSS rules have been applied and after Javascript files were executed (and potentially modified the tree). It is tightly integrated with a browser rendering engine since it needs to query for information such as the dimensions on screen of a given element. One thus has to decide on a fixed viewport size in advance on which the page should be rendered. Concretely

the algorithm first develops a vision-based content structure,which is independent of the underlying HTML document. This structure is built by splitting a page into a 3-tuple consisting of a set of visual blocks, a set of separators and a function that describes the relationship between each pair of blocks of a page in terms of their shared separators. Separators are for example vertical and horizontal lines, images similar to lines, headers and white-space. This structure is built by going top-down through the DOM tree and taking both the DOM structure and the visual information (position, color, font size) into account. Specifically they decide for each node whether it represents a visual block (i.e. the sub-tree hanging on that node) or whether it should be
subdivided further by using a number of heuristics:

- *if a sub-tree contains separators like the <hr> tag, subdivide*
- *if the background color of the children of a node differ, subdivide*
- *if most of the children are text nodes, do not divide*
- *if the size of the children differs substantially, subdivide*

They detect the set of separators visually by splitting the page around the visual blocks so that no separator intersects with a block. Subsequently they assign weights to the separators, again according to certain predefined heuristic rules. From the visual blocks and the separators they can then assemble the vision-based content structure of the page.

6.2.2 VIPS Implementation

We have not implemented this algorithm ourselves, since we could use an existing implementation from Tomas Popela instead [57] . He implemented the VIPS algorithm as part of his Master's thesis in Java using the CSSBox rendering engine. Since the testing framework is written in Python, we wrote a small driver function in Java, which we could then call as a sub-process from within Python. The sub-process runs the VIPS algorithm on the given HTML document and writes its result back .

The result of the used VIPS implementation is an XML file which contains amongst others the information about the recognized blocks. These blocks are given as plain text, i.e. they are stripped of all tags. We therefore wrote a mapping function which maps the found texts back onto the original HTML document, which is the format we need to be able to automatically apply the different evaluation metrics to the result.

6.2.3 The VIPS Algorithm

```
# Input : node <− Render tree (DOM tree with rendering information)
# level <− 0
# Output : Vision −based Content Structure
blocks = [ ]
def divideDomTree ( node , level  ) :
        if  isDividable ( node , level ) :
                for child in node.children( ) :
                        divideDomTree( child, level+1)
        else :
                blocks.append(node )
def  isDividable ( node , level ) :
        if  node.isRoot() :
                return True
        elif :
                # Check all the heuristic
        else :
                return False
```

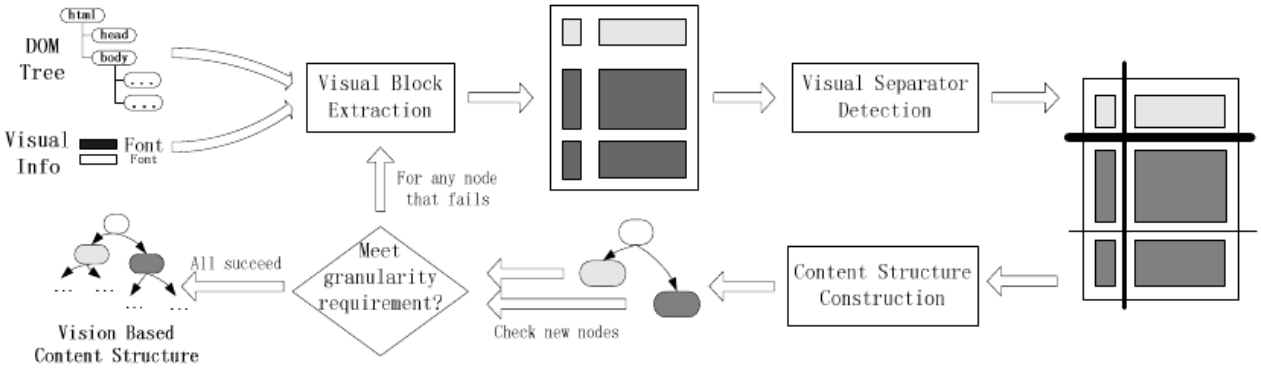


Fig. 2. The vision-based page segmentation algorithm

Figure 6.2 The vision-based page segmentation algorithm

Since we only have the plain text of the blocks, we need to check both the text content of each single node in the tree as well as the serialized plain text of the entire sub-tree of which the current element is the root. If a text node has been marked as a block we wrap it in an additional div-element containing the data-block attribute, which marks it as a block. Otherwise we just extend the tag of the found element with the block attribute. We also do a fuzzy comparison on the strings, to account for slight text anomalies due to white space and line breaks for example. As a performance optimization we walk through the tree only once when mapping the text back,since we know that the texts returned by the algorithm are in document-order (i.e. depth-first pre-order).

6.3 Block-O-Matic :

6.3.1 The Block-O-Matic Overview

Yet Another Hybrid Segmentation Tool (Andrés Sanoja , Stephane Gançarski) [49] It analyzes pages based on their DOM tree information and their visual rendering. *This tool implements a modified version of VIPS with the aim of enhancing the precision of visual block extraction and the hierarchy construction, from the user perspective*. First, the visual rendering of a page, produced by several browsers, is segmented into rectangular blocks Then, the extracted blocks are analyzed looking for visual overlaps, which are analyzed using a adapted version of the *XY-Cut algorithm and computational geometry methods and resolve the overlap*. As a result we may have different shapes of blocks, rectangular and non-rectangular blocks (polygons). Finally, the visual block tree, representing the layout of the page is analyzed in order to have a more coherent layout disposition.

6.3.2 The Block-O-Matic Algorithm

This algorithm aims to have a more precise web page segmentation using polygons to denote blocks

It is based on VIPS algorithm with some extra features:

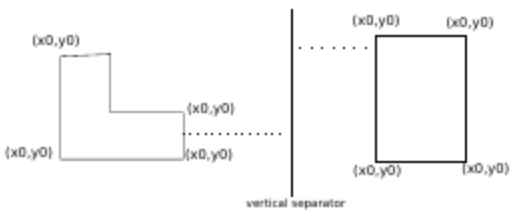
a) Polygon based model for visual blocks and separators description:

block = {(x0,y 0 ,(x 1 ,y1),..,(x n,y n))}

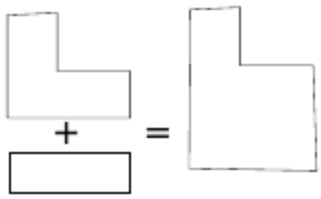
page = {block1 ,..,block m }

separator = {(xsp,y sp ,(x ep ,y ep)}

b) Computational geometric methods for visual blocks and separator detection:

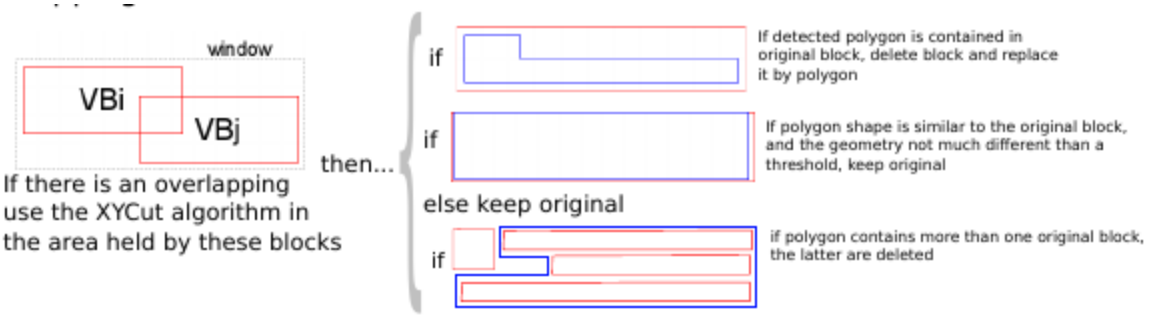


weight of separators based on the minimum distance to it

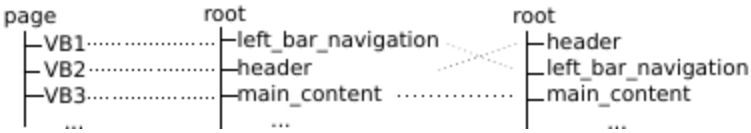


Block merge using the union convex hull

c) Overlapping detection and resolution:



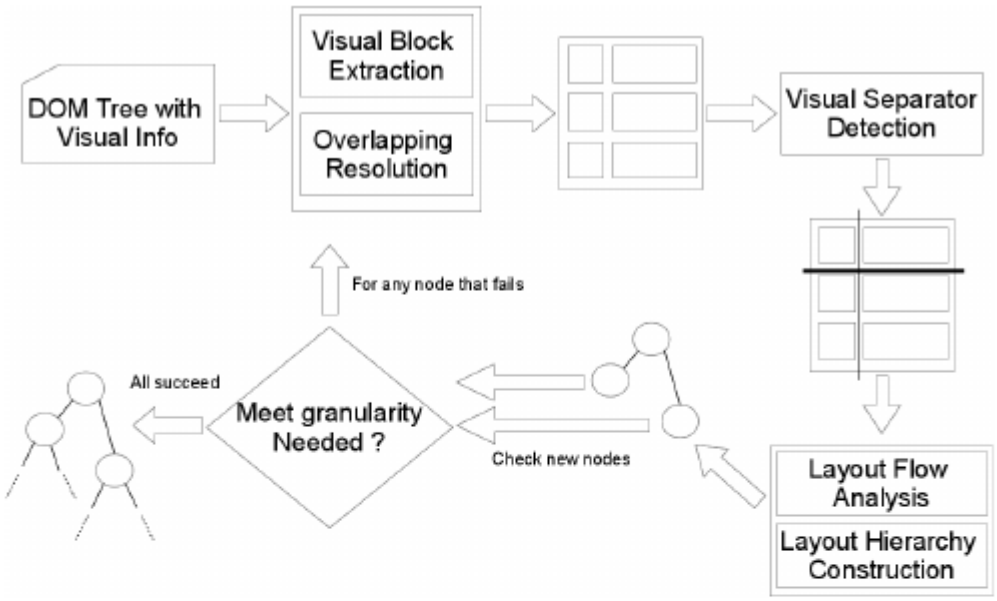
d) Page layout comparison with predefined templates for determine blocks order in a hierarchy:



Segmentation Process

```
function: segment(node)
root_block = detect_blocks(node)
root_block = detect_separators(root)
root_block = detect_overlapping(root)
root_block.merge_separators
hierarchy = root_block.detect_layout
end
```

```
function: detect_overlapping(block)
  for each pair of children of block (b1,b2)
    if detect_overlaping(b1,b2)
      window = define_window_area(b1,b2)
      new_blocks = apply_image_segmentation(window)
      for each block in new_blocks (nb)
        if b1 contains nb, delete b1 and keep nb
        if area(b1) - area(nb) > kA, delete nb and keep b1
        if nb contains b1 and b2, keeb nb and delete b1 and b2
        in other case add nb to children list
         //same procedure for b2
      end
    end
  end
```



DOM Tree with Visual Info: Is the DOM obtained from a rendering engine, after finish processing the web page

Visual Block Extraction: Is the implementation of the segmentation algorithm. It follows the general insight described by the author

Overlapping Resolution: It's objective is to resolve any overlapping between blocks an convert them to polygons

Visual Separators Detection: It is the process of determine the spaces or visual divisions between blocks. Here we applyied computation geometry to convex polygons

Layout Flow Analysis: Given a hierarchy and visual separators, the latter are weighed. A higher weight higer coherence. Those with lower weights are merged to produce a more coherent block

Layout Hierarchy Construction: Based on predefined layout templates the hierarchy pass through a process of comparison, placing each block into a taxonomy

6.3.3 The Block-O-Matic Implementation

Since the testing framework is written in Python, we wrote a small driver function in Java, which we could then call as a sub-process from within Python. The sub-process runs the algorithm on the given HTML document and writes its result back .

The result of the used Block-O-Matic implementation is an XML file which contains amongst others the information about the recognized blocks. These blocks are given as plain text, i.e. they are stripped of all tags. We therefore wrote a mapping function which maps the found texts back onto the original HTML document, which is the format we need to be able to automatically apply the different evaluation metrics to the result.

6.4 Gomory Hu Tree Based Clustering :

6.4.1 Gomory Hu Tree Based Clustering Overview

Segmenting Webpage with Gomory-Hu Based Clustering Tree (Xinyue Liu, Hongfei Lin and Ye Tian) [45].This is basically graph-theoretic based approach for web page segmentation that uses visual, structural and content based features of a web page.

In this algorithm the authors formulate the problem of web page segmentation to the clustering problem on an undirected graph, in which the vertices are leaf nodes of the DOM tree, and the edges represent the neighborhood relationship between vertices when they are rendered on the screen. *The graph constructed by both vision and structure information is a planar graph*, on which most optimization problems are much easier than those on general graphs.

They deploy the Gomory-Hu Tree based algorithm to solve the underlying clustering problem in our formulation. *The use of Gomory-Hu Tree based algorithm not only because it has theoretical clustering quality guarantees, but also because it can benefit from the graph's planarity to gain very efficient algorithm.* However, Gomory-Hu tree based algorithm tends to generate outliers. We use a simple pre processing technique to remedy this problem.

6.4.2 Gomory Hu Tree Based Clustering Algorithm

a) Vertex selection

TABLE I.
RULES FOR VERTEX SELECTION

Rules	Content
Rule 1	DOM tree nodes containing text content, link and picture information which appear in the rendered page are vertices of the graph
Rule 2	DOM tree nodes containing web page layout, text font format information are not vertices of the graph
Rule 3	DOM tree nodes containing text content, link and picture information which do not appear in the rendered page are not vertices of the graph
Rule 4	Visual nodes whose width and height on the browser screen is smaller than threshold *a* are not graph vertices
Rule 5	Visual nodes that are non-informative blank regions on the browser screen are not graph vertices

Table 6.1 Rules For Vertex Selection in Gomory Hu Tree based clustering

b) Edge Adding

(1) Obtain the location information of DOM tree nodes which are graph vertices: offsetTop, offsetBottom, offsetLeft and offsetRight.

(2) Find neighbors for each vertex. This is done by finding all nodes that are on the top, bottom, left and right of the current, and then select the nearest among them as the neighbor.

(3) Add edges to vertices with the relationship of neighbor..

c) Edge Weighting

The authors use structural information of the DOM tree to add weights to edges of the graphs. to add the edge weight between two vertices.

Given two vertices in the graph Vi and Vj, which corresponds to two nodes of the DOM tree,

Li=Di1,...DimEi and Lj=Dj1,...DjnEj , where Dix and Ei are used to denote the x tag and entities corresponding to the nodes i.

The weight, which is the similarity of the two paths, is defined as:

$$Sim(Li , L j) = 1 - EditDist ((Di1 ,LD im), (Dj1,L D jn)) / Max(D i , D j)$$

Where EditDist is the edit distance, Max(D i , D j) is the max length of the path of Di and Dj. Li and Lj represent the length of the path.

The edit distance EditDist used to calculate the min time of insert, delete and replace from the source string (s) to the target string (t). It can weigh the similarity of two strings. For example, string s1="12433",s2="1233", EditDist(s1,s2)=1, because we can insert 4 into 1233 to get 12433. The integrity algorithm can be seen in [58].

d) Gomory-Hu tree graph partition

Gomory-Hu's partition is a classic graph theoretic algorithm, which partitions the graph by computing minimum cuts and has theoretical quality guarantees.

Give an undirected weighted graph G = (V , E,W) where

V = (V 1,V2 ,...Vn) is the set of vertices,

E = { eij = (v i , v j)} is the set of edges

W = (w ij) is the set of weights.

Definition 1 (Minimum cut tree or Gomory-Hu tree)

For an undirected weighted graph G, a tree is a minimum cut tree if it follows,

(1) Nodes of the tree are vertices of the graph;

(2) Each edge in the tree has a non-negative weight wij;

(3) For each pair of nodes s and t, let eij be the edge on the path from s to t with

the minimum weight, then eij's is equal to the capacity of the minimum cut between in s and t in the graph.

The Gomory-Hu's algorithm constructs the minimum cut tree based on the maximum flow minimum cut theorem. For the graph deduced from a web page, in which the vertices represent the visual leaf nodes of the DOM tree,the edges represent neighborhood relationship between them, and edge weight represent similarity between neighbors. Then during the construction of the minimum cut tree, each SupperNode represents continuous region of the web page. The procedure of minimum cut tree construction can be stopped following a best cluttering criteria, thus we can get k SupperNodes, i.e., k sub-graphs that maps to k blocks in the web page, such that the intra block semantic similarity is maximized, and the interblock semantic similarity is minimized.

e) Gomory-HuPS Algorithm

Input: G is an undirected graph with n vertices;

Output: k parts of G;

1. vertex i is one class of G, the number of vertices reduce 1;
2 .while(vertex j in all vertices of G){
3. if(i, j satisfy formulae (3)) j is in the class of i,the number of vertices reduce 1;
4. else {
5. if(j satisfy formulae (2)){
6. j is in the class of i, the number of vertices reduce 1;
7. if(the number of vertices is 0) using Gomory-Hu algorithm to get the Gomory-Hu tree of the new graph, taking out the edges ascending, then get k parts of G;
8. else j is a new class, the number of vertices reduce 1;
9 .}
10 .else j is a new class, the number of vertices reduce 1;
11. }
12. }

6.4.3 Gomory Hu Tree Based Clustering Implementation

Vertex Selection : Since the algorithm is based on the rendered representation of a page we needed a way to get such a rendering and interact with it. First we render the web page in browser using *Python Selenium API [59]*. To get the list of initial nodes conforming to the vertex selection rule depicted in section 6.4.2 we inject a small javascript snippet to extract the nodes that content only visible text node,images,inputs etc. *Here we exclude tags like 'script', 'noscript', 'style', 'area', 'head', 'meta', 'frame', 'frameset', 'br','hr' etc.*

Filtering Initial Node List : *Now we filter out some of initial nodes based on some heuristics.* Nodes with too small area are filtered. Larger nodes are merged with smaller nodes and overlapping nodes are discarded too.

Information Extraction : Through the injection of javascript code in the webpage (*using JavascriptExecutor execute_script() functionality of Selenium API*) we extract out all the

a. Structure-based information : leaf node to parent node tagPath information

b. Layout-based information : location information of the DOM tree nodes offsetTop, offsetBottom, offsetLeft and offsetRight.

c. Content-based information : outerHTML of the leaf node .

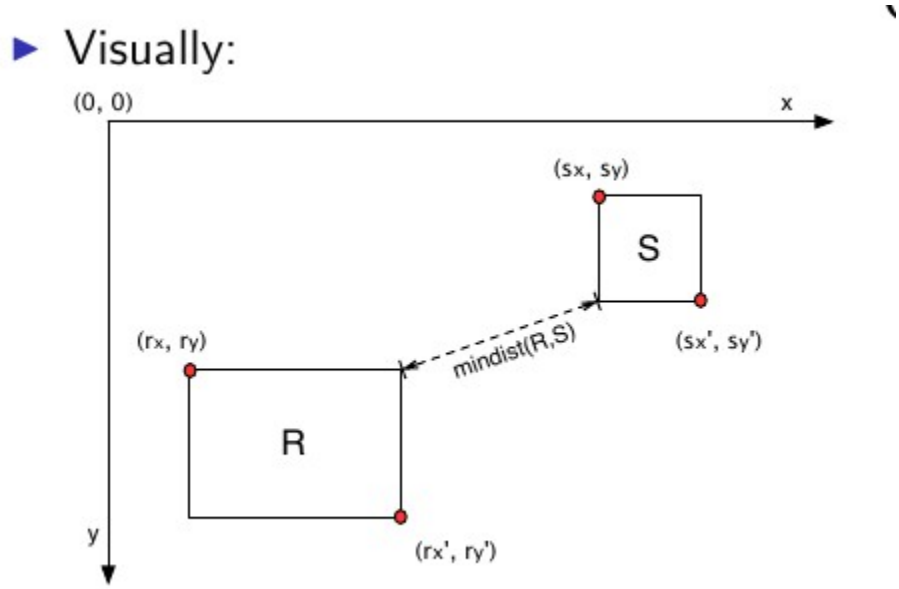
We store all the information in a class named “Vertex Object”

Graph creation and adding weights to edges :

We find neighbors for each vertex using geometric distance formula for rectangular neighbors.

- Let $R = [(r_x, r_y), (r_x', r_y')]$ and $S = [(s_x, s_y), (s_x', r_y')]$ be two bounding rectangles
- The geometric distance of R to S is given by

$$dist(R,S) = \left(\sum_{i \in x,y} {t_i}^2\right)^{\frac{1}{2}}, \text{ with } t_i = \begin{cases} r_i - s_i' & if \quad r_i > s_i' \\ s_i - r_i' & if \quad r_i' < s_i \\ 0 & if \quad otherwise. \end{cases}$$



We add weights to the edges using the the editDistance formulae depicted in section 6.4.2 c) Edge weighing portion

Sim(Li , L j) = 1 − EditDist ((Di1 ,LD im), (Dj1,L D jn)) / Max(D i , D j)

We use distance() functionality of Python Levenshtein Package [60]

Gomory Hu Tree Creation and Clustering : After the graph creation , neighbor finding and edge weighing have been done, the Gomory Hu Tree based on the graph is created using the functionality of iGraph 0.7 package and it's python binding [61, 62]. From the Gomory Hu Tree using the *Gomory-HuPS Algorithm the main clustering algorithm runs as described in section 6.4.2 and we get k parts of the original graph. When we map the k segments in original html page we can get the segmented web page.*

Chapter 7

Results

In this chapter we present the results of our evaluation of the four different segmentation algorithms described in chapter 6. We tested all algorithms in a number of different configurations using the testing framework presented in Chapter 5. First, we tested them on the two different datasets described in Chapter 5: The randomly selected dataset and the popular dataset. The first one consists of 82 random pages and the second one of 70 popular pages, marked up . We chose these two types of datasets to test whether the algorithms perform differently on random and on popular pages on average.

7.1 Output of Implemented Segmentation Algorithms

Here we present the results of the implemented, customized web page segmentation algorithms.

7.1.1 Block-O-Matic Result

A sample run of the algorithm *Block-O-Matic* for the page *http://www.iitg.ac.in/aboutus*

Original Page :

Figure 7.1 Block-O-Matic Result : Original Web Page

Segmented Page :

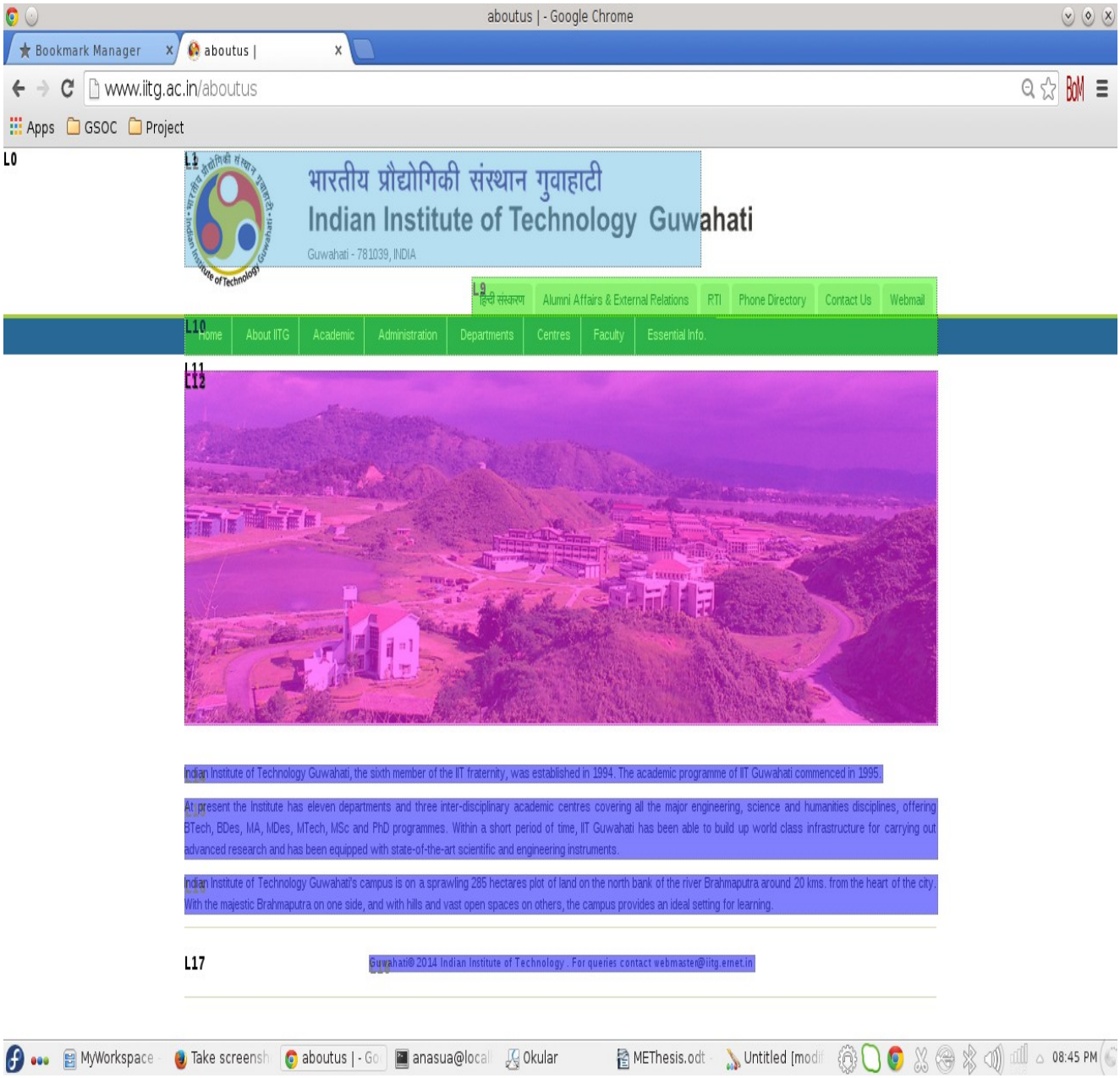


Figure 7.2 Block-O-Matic Result : Segmented Web Page

7.1.2 VIPS Result

Original WebPage

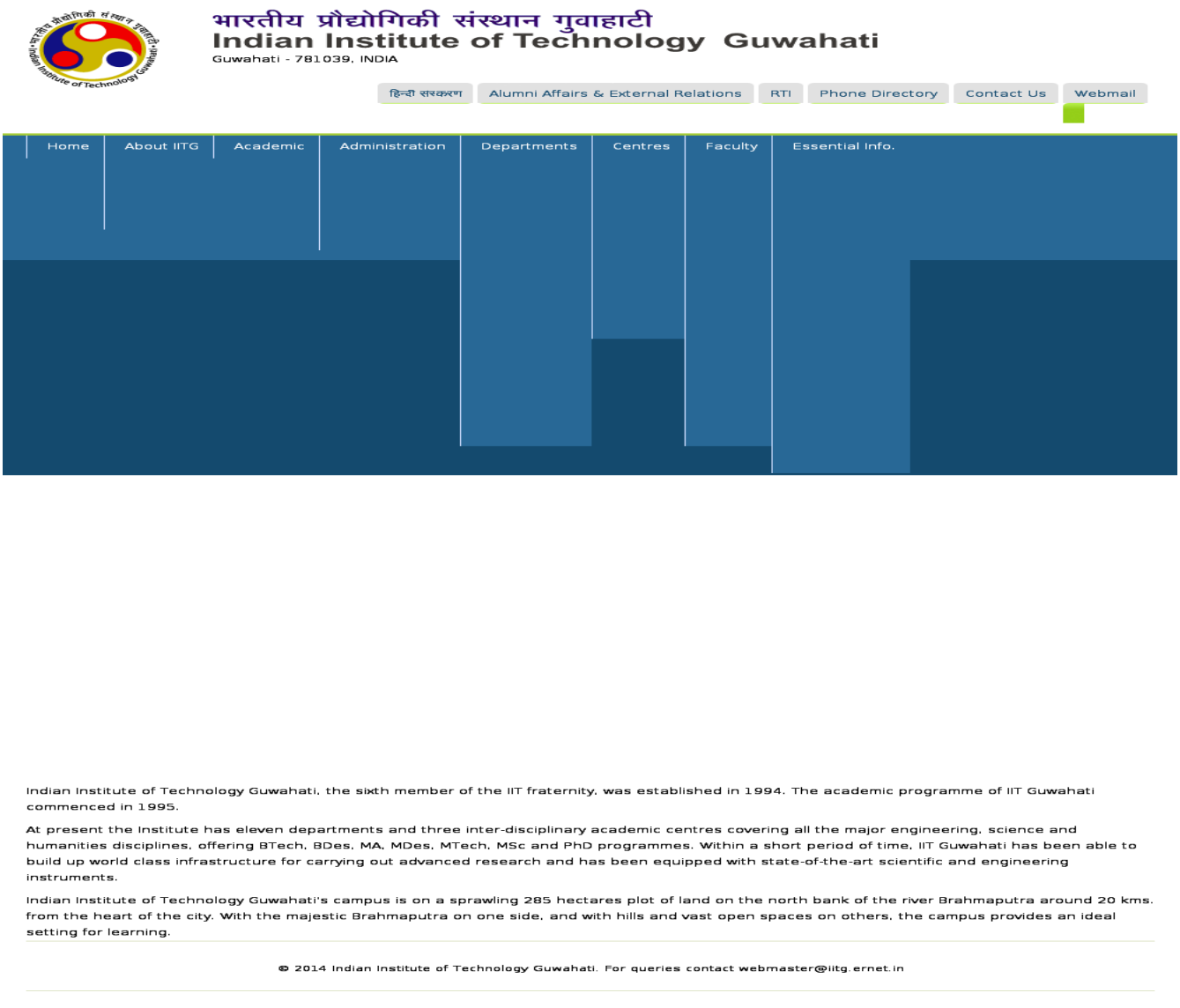


Figure 7.3 VIPS Result : Original Web Page

Segmented Page :

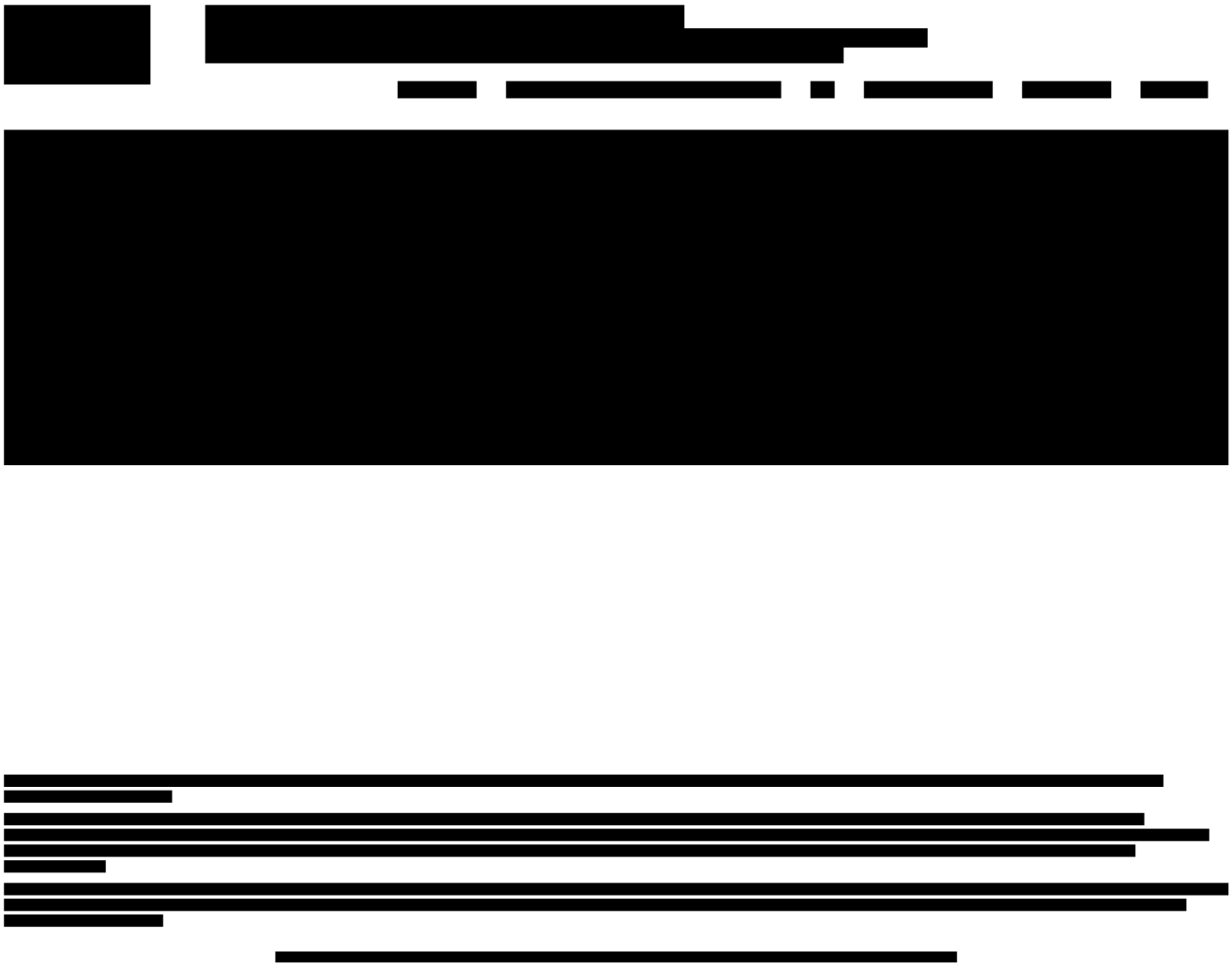


Figure 7.4 VIPS Result : Segmented Web Page

7.1.3 BlockFusion Result

Original Page :

Figure 7.5 BlockFusion Result : Original Web Page

Segmented Page :

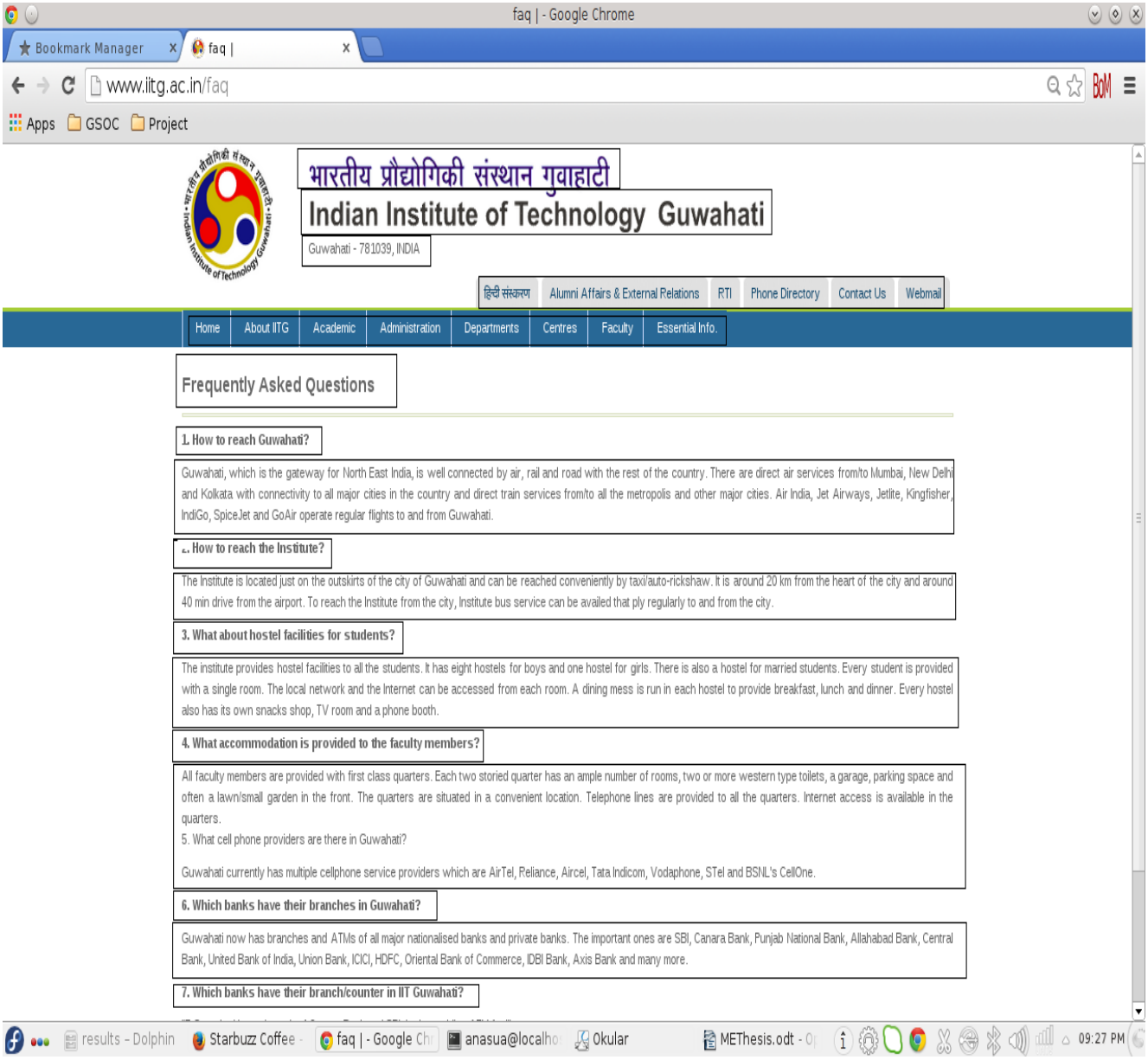


Figure 7.6 BlockFusion Result : Segmented Web Page

7.1.4 Gomory Hu Tree Based Clustering Result

Original Page :

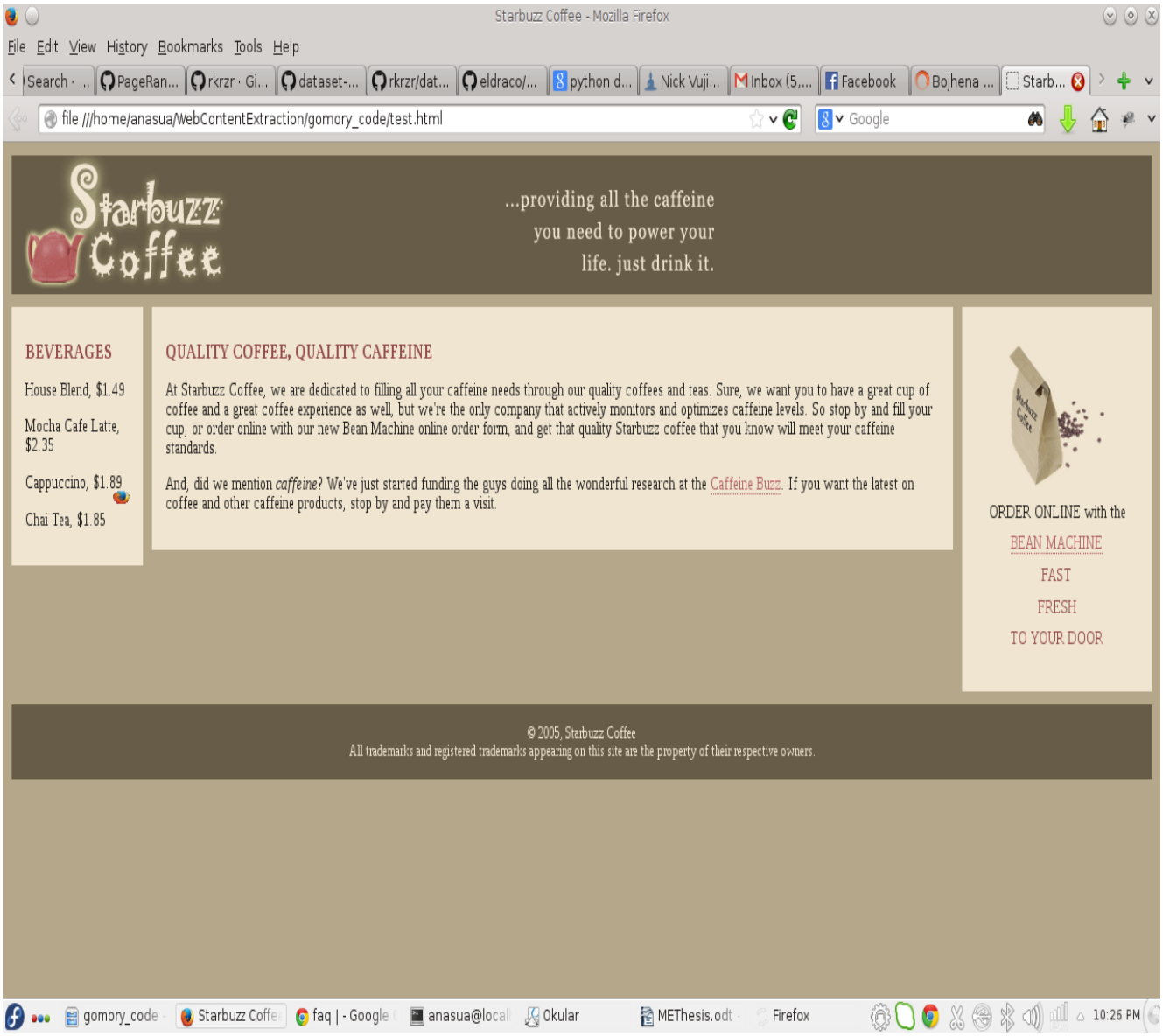


Figure 7.7 Gomory Hu Tree Based Clustering Result : Original Web Page

Two intermediate step in sample run :

a. Initial Node Selection :

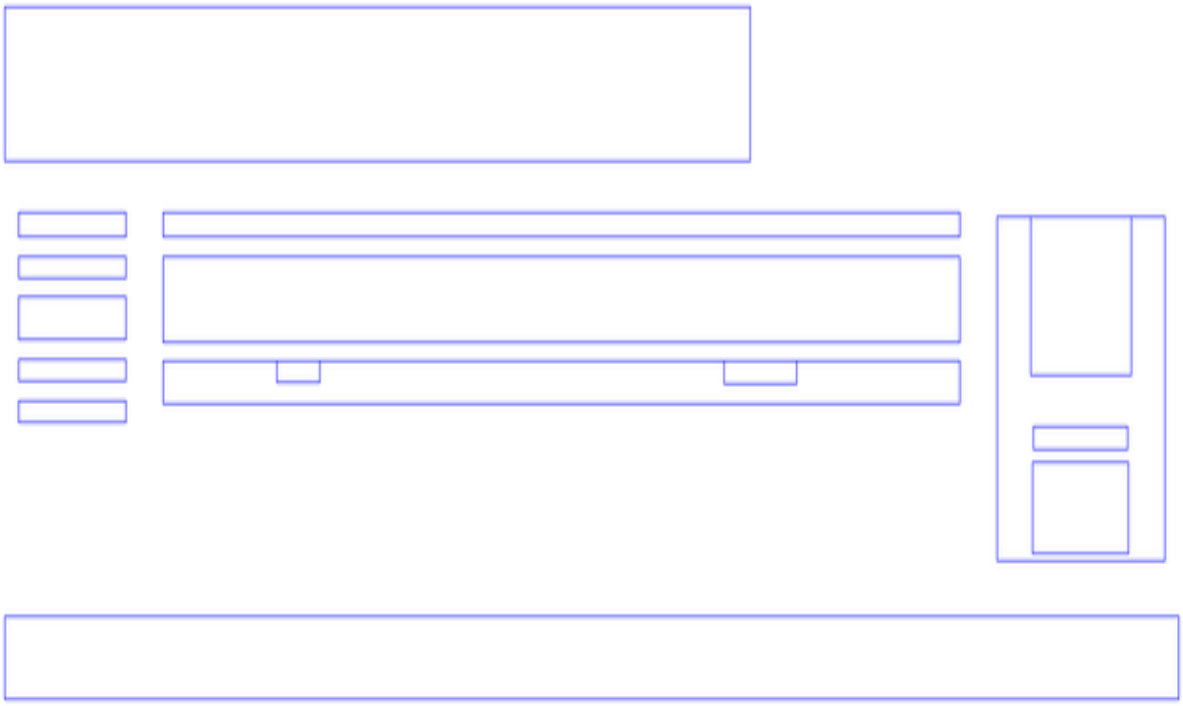


Figure 7.8 Gomory Hu Tree Based Clustering Result : Initial Node Selection

b. Graph Creation and Edge Adding :

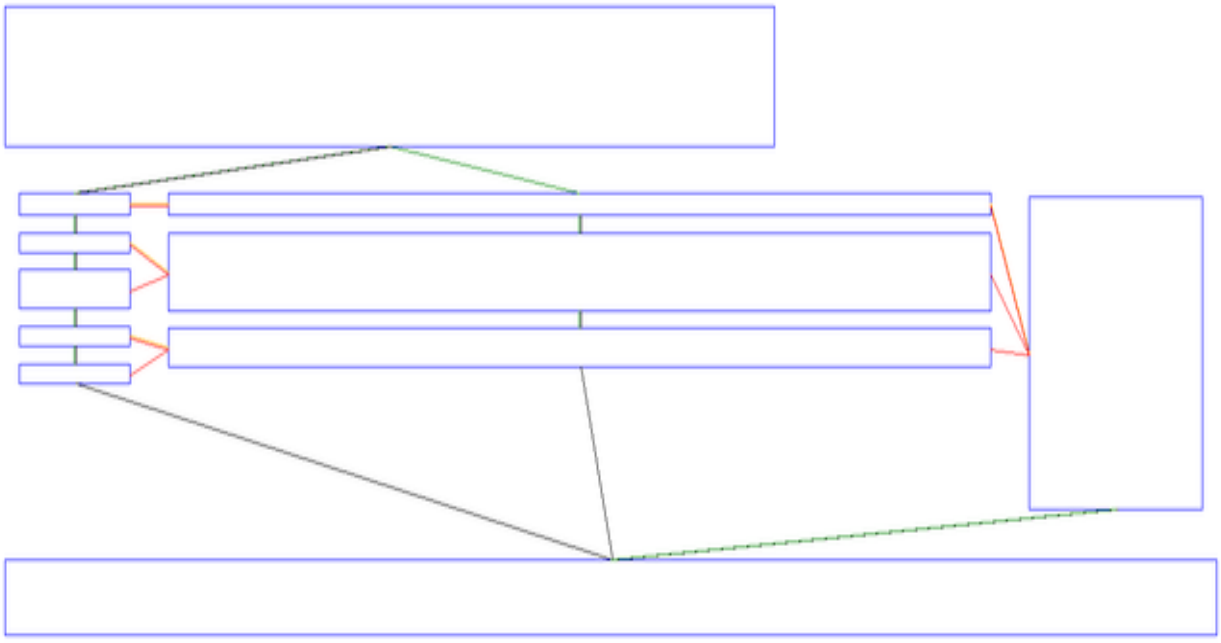


Figure 7.9 Gomory Hu Tree Based Clustering Result : Graph Creation and Edge Adding

c. After Clustering Segmented Page :

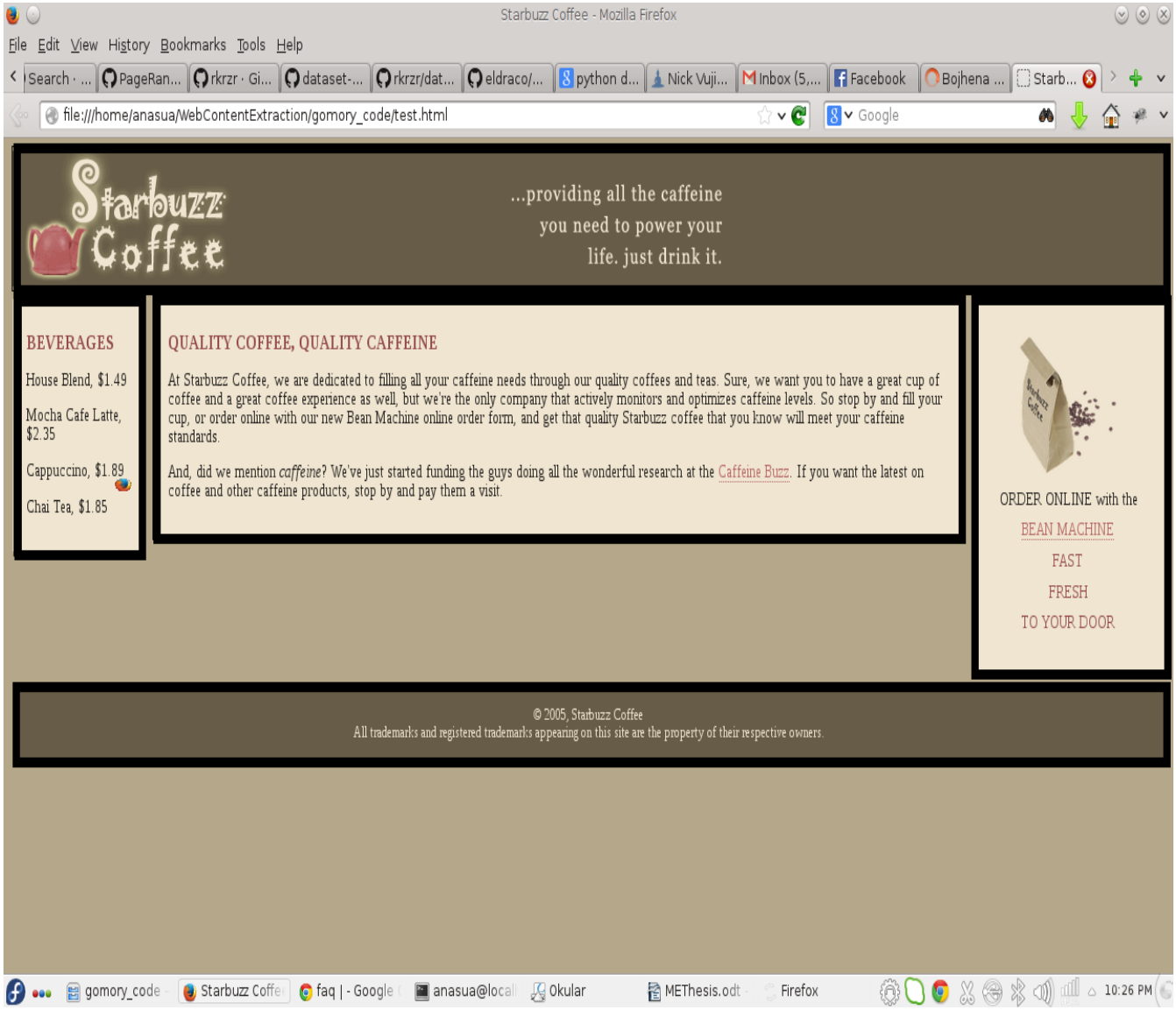


Figure 7.10 Gomory Hu Tree Based Clustering Result : Segmented Web Page

7.2 Evaluation Metric

We consider two metrics to compare the generated results to the ground truth,

a. ***the exact match metric*** *and*

b. ***the fuzzy match metric.***

Both of them compare the string contents of the blocks to each other. Each block is serialized to only text with all HTML tags removed and white-space and newlines removed as well.

7.2.1 Exact Match Metric : *For the exact match metric it then simply checks for string equality. This is of course a very strong criterion, as a minimally different string would be counted as false, while for most applications it would likely be perfectly sufficient.*

7.2.2 Fuzzy Match Metric : *For this reason we also do a fuzzy string comparison using the Python difflib library [63] .Concretely we use the SequenceMatcher class to check for a similarity ratio of better than 0.8 between strings. This class implements an algorithm similar to the "gestalt pattern matching" approach in [64].*

7.2.3 Precision : *Precision is a measure of quality that is defined as the number of relevant results out of all retrieved results.*

Precision =(a) Number of "Correct"Segments /
(b) Number of All Segments

7.2.4 Recall : *Recall is a measure of quantity that is defined as the number of retrieved results out of all relevant results.*

Recall = (a) Number of "Correct"Segments /
(c) Number of All"Correct"Segments

(a) Number of "Correct" Segments The number of positions segmented correctly.
(b) Number of All Segments. The number of all positions segmented in spite of correct or not.
(c) Number of All "Correct" Segments. The number of all positions segmented correctly.

Furthermore, in order to measure the overall accuracy of segmentation, we calculated F-measure based on the precision and recall. The mean F-measure is used to compare our results.

7.2.5 F-Score : F-Score is a combination of the two, defined as

$$F = 2 * (P * R) / (P + R)$$

where, p = Precision

R = Recall

7.2.6 Number of Retrieved Blocks : number of blocks retrieved by the algorithm.

7.2.7 Hits : number of relevant results returned by the algorithm .

7.2.8 Total Number of Relevant Result : determined by the ground truth.

7.3 The random dataset

Here we present the results of running the four different algorithms on the dataset consisting of 82 random pages. On average we have 12.24 relevant blocks on a random page. BlockFusion returns on average about twice as many blocks as there are relevant blocks, but recall is still very low (i.e. retrieved and relevant results hardly overlap). VIPS, Block O Matic and Gomory Hu Tree based algorithm returns too few blocks on average . VIPS has therefore low recall, but precision is higher (for the fuzzy match metric). Comparing the exact to the fuzzy match metric it can be seen that results are considerably better for the fuzzy match metric.

Exact match metric Precision and Recall are generally very low. BlockFusion recognize hardly anything. Precision is highest for Gomory Hu Tree based and Recall is highest for Block O Matic .

Algorithm	Precision	Recall	F-Score	Retrieved	Hits	Relevant
BlockFusion	0.03	0.06	0.04	24.99	0.99	12.24
Block O Matic	0.14	0.27	0.18	16.5	2.97	12.24
VIPS	0.07	0.06	0.06	17.96	0.93	12.24
Gomory Hu Tree based	0.23	0.2	0.21	10.62	2.23	12.24

Table 7.1 : Random-HTML-Exact

Fuzzy match metric Precision and Recall are clearly better for the fuzzy match metric with the number of hits roughly doubling. Especially VIPS improves substantially, indicating that a number of its blocks were only slightly off from the ground truth. The best F-Score (0.45 , Gomory Hu Tree based) .

Algorithm	Precision	Recall	F-Score	Retrieved	Hits	Relevant
BlockFusion	0.06	0.11	0.08	24.99	1.99	12.24
Block O Matic	0.29	0.31	0.3	16.5	4.95	12.24
VIPS	0.28	0.16	0.2	17.96	3.66	12.24
Gomory Hu Tree based	0.48	0.43	0.45	10.62	4.81	12.24

Table 7.2: Random-HTML-Fuzzy

7.4 The popular dataset

Here we present the results of running the four different algorithms on the dataset consisting of 70 popular pages. On average we have 16.1 relevant blocks on a page.

Exact match metric BlockFusion decidedly returns too many blocks on average, VIPS , Block O Matic and Gomory Hu Tree base are fairly close to the relevant number of blocks. Gomory Hu Tree having the best precision and the best recall.

Algorithm	Precision	Recall	F-Score	Retrieved	Hits	Relevant
BlockFusion	0.03	0.06	0.04	72.85	2.91	16.22
Block O Matic	0.18	0.17	0.17	18.96	3.22	16.22
VIPS	0.13	0.14	0.12	19.72	2.25	16.22
Gomory Hu Tree based	0.27	0.28	0.26	14.75	3.77	16.22

Table 7.3: Popular-HTML-Exact

Fuzzy match metric The pattern seen in the random dataset repeats: results for the fuzzy match metric are about twice better than for the exact match metric .

Algorithm	Precision	Recall	F-Score	Retrieved	Hits	Relevant
BlockFusion	0.05	0.12	0.07	72.85	5.14	16.22
Block O Matic	0.39	0.41	0.4	18.96	6.74	16.22
VIPS	0.32	0.31	0.3	19.72	5.74	16.22
Gomory Hu Tree based	0.4	0.43	0.4	14.75	5.91	16.22

Table 7.4: Popular HTML-Fuzzy

Chapter 8

Discussion

8.1 Exact and fuzzy match metric

We found that the number of recognized blocks improved significantly when using the fuzzy match metric as opposed to the exact match metric, as was to be expected. We believe that the results from the fuzzy match metric are generally more valuable since the quality of blocks will still be sufficient for most applications. Furthermore it can easily be adjusted to find more or less precise matches by adjusting the matching ratio.

8.2 The four segmentation algorithms

We found that the four algorithms in our comparison exhibited a widely differing performance. All together none of them performed well enough to be universally applicable, as the highest average F-Score was a 0.45 (Gomory Hu Tree based).

8.2.1 BlockFusion *This algorithm showed the worst performance on both datasets. Both precision and recall are very low (<0.1 and <0.2 respectively). It also returns too many blocks on average (2.5x too many for the random dataset and 5.1x too many for the popular dataset). . We conclude that a solely text-based metric is not sufficient for a good segmentation, but that it can be used to augment other approaches.*

8.2.2 Block O Matic *This algorithm exhibits low precision and (relatively) high recall . This is due to the fact that it retrieves by far the most blocks from all algorithms . The number of false positives is thus very high. . In terms of the F-Score the algorithm had the second-best result.*

8.2.3 VIPS *This algorithm showed the biggest difference between the random and the popular dataset. Precision was high and recall mediocre for the random dataset fuzzy match metric . Performance improves substantially, while both were*

low for the popular dataset . It is not clear why there is such a substantial difference. The number of retrieved results is slightly low for the random dataset, while it is slightly high for the popular dataset .

8.2.4 Gomory Hu Tree Based Clustering *This algorithm showed relatively high precision and recall for both datasets . It retrieved slightly too few blocks for both datasets. We thus find that a combination of structural , content based and rendering-based approaches enhances overall results. Future work could therefore likely improve upon these results by using more sophisticated combinations of different approaches and heuristics.In terms of the F-Score the algorithm had the best result 4.5.*

8.3 Encountered Problems

First surprising discovery was that there is no de-facto “standard” dataset on which everybody bases the evaluation of their segmentation algorithm, as is common in other fields such as spam detection (where there is the WEBSPAM-UK20071 dataset). This is surprising as it forces every author to create their own dataset,which on the one hand is a lot of work, and on the other hand makes the comparison of different algorithms much harder.

The second problem was that we could not obtain the original implementation of any of the three algorithms in our comparison (we could obtain an implementation of VIPS, but it is not from the original authors). This again leads to duplication of work, as we had to re-implement these algorithms, and it makes the results more fragile as it is impossible to prove that they were implemented exactly according to their specification. This is true as often the descriptions of algorithms (and in particular of heuristics) are not specific enough to not require some interpretation of the concrete wording.

Chapter 9

Conclusion

Going back to the first part of our research question, how well do existing Web page segmentation algorithms work on modern websites, we can now conclude that their performance in general has gotten worse over time.

While all three older algorithms, BlockFusion, Block O Matic, VIPS, Gomory Hu Tree based segmentation showed a strong performance in their original publications, this does not hold any more on our dataset using recent Web pages.

As our analysis using one dataset consisting of random pages and one consisting of popular pages shows, the main reason for this is the increasing complexity of websites and their ever more dynamic behavior due to the increasing prevalence of DOM manipulations via Javascript.

The systematic exploration and testing of the different algorithms was enabled by the testing framework we developed for this thesis. It allows to exchange datasets and algorithms and is also easily extensible with more page segmentation algorithms. It thus forms a solid basis for future work in this field.

As a last word, page segmentation algorithms seem like one possible option to a more semantic Web, but there still remains a lot of work to be done.

Bibliography

1. **boilerpipe** - boilerplate removal and full text extraction from HTML pages - *google project hosting* - ***http://code.google.com/p/boilerpipe/.*** *URL*- http://code.google.com/p/boilerpipe/**.**
2. {RDF Vocabulary Description Language 1.0: RDF Schema}.
3. **html5lib/html5lib**- python · *GitHub*.- ***https://github.com/html5lib/html5lib-python.*** *URL*- https://github.com/html5lib/html5lib-python**.**
4. **lxml/lxml**-*GitHub*.-***https://github.com/lxml/lxml/***.*URL* ***ttps://github.com/lxml/lxml/.***
5. **Python-boilerpipe**.- ***https://github.com/rkrzr/python-boilerpipe***, . *URL*- https://github.com/rkrzr/python-boilerpipe**.**
6. **Python webkit DOM bindings.** ***http://www.gnu.org/software/pythonwebkit/***, *URL*- http://www.gnu.org/software/pythonwebkit/**.**
7. Ian Hickson and David Hyatt. HTML5: a vocabulary and associated APIs for HTML and XHTML. W3C Working Draft, 19, 2010.
8. Eric Prud'Hommeaux and Andy Seaborne. SPARQL query language for RDF. W3C recommendation, 15, 2008.
9. Deborah L. McGuinness and Frank Van Harmelen. OWL web ontology language overview.W3Crecommendation,10(2004-03):10,2004.*URL* ***http://cies.hhu.edu.cn/pweb/~zhuoming/teachings/MOD/N4/Readings/5.3-B1.pdf.***
10. Mike Dean, Guus Schreiber, Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL web ontology language reference. W3C Recommendation February, 10, 2004.
11. R. Burget. Layout based information extraction from html documents. In Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on, volume 2, pages 624 –628, sept. 2007.
12. Radek Burget. Automatic document structure detection for data integration. In Proceedings of the 10th international conference on Business information systems, BIS'07, pages 391–397, Berlin, Heidelberg, 2007. Springer-Verlag.
13. Shian-Hua Lin and Jan-Ming Ho. Discovering informative content blocks from

web documents. In KDD ’02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 588–593, New York, NY,USA, 2002. ACM.

14. Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In KDD ’03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 296–305, New York, NY,USA, 2003. ACM.
15. Y. Whang, C. Jung, J. Kim, and S. Chung. Webalchemist: A web transcoding system for mobile web access in handheld devices. In Optoelectronic and Wireless Data Management, Processing, Storage, and Retrieval, pages 102–109, 2001.
16. Yunpeng Xiao, Yang Tao, and Qian Li. Web page adaptation for mobile device. In Wireless Communications, Networking and Mobile Computing, 2008.
17. Xing Xie, Gengxin Miao, Ruihua Song, Ji-Rong Wen, and Wei-Ying Ma. Efficient
18. browsing of web search results on mobile devices based on block importance model.
19. In Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, pages 17–26, Washington, DC, USA, 2005. IEEE Computer Society.
20. Yonghyun Hwang, Jihong Kim, and Eunkyong Seo. Structure-aware web transcoding for mobile devices. IEEE Internet Computing, 7(5):14–21, 2003.
21. Jiuxin Cao, Bo Mao, and Junzhou Luo. A segmentation method for web page analysis using shrinking and dividing. International Journal of Parallel, Emergent and Distributed Systems, 2010.
22. Liu Wenyin, Guanglin Huang, Liu Xiaoyue, Zhang Min, and Xiaotie Deng. Detection of phishing webpages based on visual similarity. In Special interest tracks and posters of the 14th international conference on World Wide Web, WWW ’05, pages 1060–1061, New York, NY, USA, 2005. ACM.
23. Liu Wenyin, Guanglin Huang, Liu Xiaoyue, Xiaotie Deng, and Zhang Min. Phishing webpage detection. In Proceedings of the Eighth International Conference on Document Analysis and Recognition, ICDAR ’05, pages 560–564, Washington, DC, USA,2005. IEEE Computer Society.
24. Wenyin Liu, Xiaotie Deng, Guanglin Huang, and Anthony Y. Fu. An antiphishing

strategy based on visual similarity assessment. IEEE Internet Computing, 10:58–65, March 2006.

25. Lan Yi, Bing Liu, and Xiaoli Li. Eliminating noisy information in web pages for data mining. In KDD ’03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 296–305, New York, NY,USA, 2003. ACM.

26. Saikat Mukherjee, Guizhen Yang, and I. V. Ramakrishnan. Automatic annotation of content-rich html documents: Structural and semantic analysis. In In Intl. Semantic Web Conf. (ISWC, pages 533–549, 2003.

27. D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y.-K. Ng,and R. D. Smith. Conceptual-model-based data extraction from multiple-record webpages. Data Knowl. Eng., 31:227–251, November 1999.

28. D. W. Embley, Y. Jiang, and Y.-K. Ng. Record-boundary discovery in web documents.In Proceedings of the 1999 ACM SIGMOD international conference on Management of data, SIGMOD ’99, pages 467–478, New York, NY, USA, 1999. ACM.

29. David W. Embley and Li Xu. Record location and reconfiguration in unstructured multiple-record web documents. In Selected papers from the Third International Workshop WebDB 2000 on The World Wide Web and Databases, pages 256–274, London, UK, 2001. Springer-Verlag.

30. Ziv Bar-Yossef and Sridhar Rajagopalan. Template detection via data mining and its applications. In Proceedings of the 11th international conference on World Wide Web,WWW ’02, pages 580–591, New York, NY, USA, 2002. ACM.

31. Fariza Fauzi, Jer-Lang Hong, and Mohammed Belkhatir. Webpage segmentation for extracting images and their surronding contextual information. In MM ’09: Proceedings of the seventeen ACM international conference on Multimedia, pages 649–652,2009.

32. X. Yin and W.S. Lee. Using link analysis to improve layout on mobile devices. In Proceedings of the Thirteenth International World Wide Web Conference, pages 338–344, 2004.

33. Ruihua Song, Haifeng Liu, Ji-Rong Wen, and Wei-Ying Ma. Learning block importance models for web pages. In Proceedings of the 13th international conference on World Wide Web, WWW ’04, pages 203–211, New York, NY, USA, 2004. ACM.in Technology Enhanced Learning, 3(3), 253–273.

34. Yang, S. J. H., Zhang, J., Chen, R. C. S., & Shao, N. W. Y. (2007). A UOI-based content adaptation method for improving web content accessibility in the mobile internet.ETRI Journal, 29(6), 794–807.Yang, S. J. H., Zhang, J., & Chen, I. Y. L. (2008). A JESS enabled context elicitation

35. Jeff J.S. Huang, Stephen J.H. Yang. Web content adaptation for mobile device: A fuzzy-based approach in Knowledge Management & E-Learning: An International Journal, Vol.4, No.1.

36. L.Q. Chen, X. Xie, W.Y. Ma, H.J. Zhang, H.Q. Zhou, and H.Q. Feng, DRESS A Slicing Tree Based Web Representation for Various Display Sizes, Technical Report MSR-TR-2002-126, Microsoft Research, 2002.

37. By Dongsong Zhang Web Content Adaptation for Mobile Handheld Devices in COMMUNICATIONS OF THE ACM February 2007/Vol. 50, No. 2

38. Yu Chen, Xing Xie,Wei-Ying Ma,and Hong-Jiang Zhang Microsoft Research,Asia,Adapting Web Pages for Small-Screen Devices

39. Joachim Hammer, Hector Garcia-Molina, Junghoo Cho, Rohan Aranha, and Arturo Crespo. Extracting semi structured information from the web.1997. *URL- **http://ilpubs.stanford.edu:8090/250/.***

40. Srinivas Vadrevu, Fatih Gelgi, and Hasan Davulcu. Semantic partitioning of web pages. Web Information Systems Engineering –*WISE 2005, volume 3806 of Lecture Notes in Computer Science, pages 107*-118. Springer Berlin / Heidelberg, 2005. ISBN 978-3-540-30017-5. *URL* - ***http://www.springerlink.com/content/u62r27p2720t1173/abstract/.***

41. G. Nicol, L. Wood, M. Champion, and S. Byrne. Document object model (DOM) level 3 core specification. 2001. *URL* - ***http://www.lashkul.info/books/Computers/Programming%20languages%20And%20Technologies/XML/Specifications/DOM3-Core.pdf***.

42. S. Vadrevu and E. Velipasaoglu. Identifying primary content from web pages and its application to web search ranking. In Proceedings of the 20th international conference companion on World wide web, page 135–136, 2011. *URL- **http://dl.acm.org/citation.cfm?id=1963261.***

43. M. Kovacevic, M. Diligenti, M. Gori, and V. Milutinovic. Recognition of common areas in a web page using visual information: a possible application in a page classification. In 2002 IEEE International Conference on Data Mining, 2002. ICDM 2003. Proceedings, pages 250 – 257, 2002.

doi: 10.*1109/ICDM.2002.1183910*.

44. D. Chakrabarti, R. Kumar, and K. Punera. A graph-theoretic approach to web page segmentation. In Proceeding of the 17th international conference on World Wide Web, page 377–386, 2008. *URL*- ***http://dl.acm.org/citation.cfm?id=1367549***.

45. Xinyue Liu1, Hongfei Lin and Ye Tian. Segmenting Web page with Gomory-Hu Tree Based Clustering . Journal of Software (1796217X) . Dec2011, Vol. 6 Issue 12, p2421-2425. 5p. *URL*- ***http://ojs.academypublisher.com/index.php/jsw/article/download/jsw061224212425/4043***

46. C. Kohlschütter and W. Nejdl. A densitometric approach to web page segmentation. In Proceeding of the 17th ACM conference on Information and knowledge management, page 1173–1182, 2008. *URL*- http://dl.acm.org/citation.cfm?id=1458237**.**

47. D. Cai, S. Yu, J. R. Wen, and W. Y. Ma. VIPS: a vision based page segmentation algorithm. Technical report, Microsoft Technical Report, MSR-TR-2003-79, 2003. *URL* ***ftp://ftp.research.microsoft.com/pub/tr/TR-2003-79.pdf***.

48. Elgin Akpinar and Yeliz Yesilada. Vision based page segmentation: Extended and improved algorithm. January 2012. *URL* http://cng.ncc.metu.edu.tr/content/emine.php**.**

49. Andrés Sanoja , Stephane Gançarski. Yet Another Hybrid Segmentation Tool iPRES 2012 – Proceedings of the 9th International Conference on Preservation of Digital Objects. Toronto 2012. ISBN 978-0-9917997-0-1 *URL*- ***http://www.scapeproject.eu/wpcontent/uploads/2012/11/lip6submittedIpres2012-1.pdf***

50. S. Baluja. Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In Proceedings of the 15th international conference on World Wide Web, page 33–42, 2006. *URL* - http://dl.acm.org/citation.cfm?id=1135777.1135788**.**

51. Y. Yesilada. Web page segmentation: A review. 2011. *URL* - ***http:// wel-eprints.cs.manchester.ac.uk/148/.***

52. Peifeng Xiang, Xin Yang, and Yuanchun Shi. Web page segmentation based on gestalt theory. In Multimedia and Expo 2007 IEEE International Conference (ICME), 2007. *URL*- ***http://pi.cs.tsinghua.edu.cn/publications/2007/2007ICME-***

XIANG%20Peifeng.pdf

53. Peifeng Xiang, Xin Yang, and Yuanchun Shi. Enhanced Gestalt Theory Guided Web Page Segmentation for Mobile Browsing. In Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT '09. IEEE/WIC/ACM International Joint Conferences. *URL-* ***http://pi.cs.tsinghua.edu.cn/publications/2007/2007ICME-XIANG %20Peifeng.pdf***
54. R Guha and R McCool. TAP: a semantic web test-bed. Web Semantics: Science, Services and Agents on the World Wide Web, 1(1):81–87, December 2003. ISSN 1570-8268. doi: 10.1016/j.websem.2003.07.004**.** ***URL http://www.sciencedirect.com/science/article/pii/S1570826803000064.***
55. Crawled by the Laboratory of Web Algorithmics, University of Milan, http://law.dsi.unimi.it/. Yahoo! research: "Web spam collections".***URL*** http://barcelona.research.yahoo.net/webspam/datasets//datasets/uk2007/**.**
56. Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In Proceedings of the third ACM international conference on Web search and data mining, WSDM ’10, page 441–450, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-889-6. doi: 10.1145/1718487.1718542. ***URL http://doi.acm.org/10.1145/1718487.***
57. Tomas Popela. Implementation of Algorithm for Visual Web Page Segmentation. PhD thesis, Brno University of Technology. ***URL*** http://www.fit.vutbr.cz/study/DP/DP.php?id=14163&file=t**.**
58. D. Gusfield, Algorithms on Strings, Trees and Sequences. Cambridge University Press, 1997.
59. ***Python selenium API*** https://pypi.python.org/pypi/selenium
60. ***Python Levenshtein API*** https://pypi.python.org/pypi/python-Levenshtein/
61. ***iGraph API*** https://launchpad.net/igraph/+milestone/0.7
62. ***Python iGraph API*** *https://pypi.python.org/pypi/python-igraph/0.7*
63. ***Python difflib API*** *http://docs.python.org/2/library/difflib.html*
64. John W. Ratcliff and David Metzener. Pattern matching: The gestalt approach. Dr. Dobb’s Journal, 13(7):46–72, 1988.
65. *A Quantitative Comparison of Semantic Web Page Segmentation Algorithms (Master’s Thesis in Computer Science by Robert Kreuzer).*