Effectiveness of Template Detection on Noise Reduction and Websites Summarization

by

Derar Hasan Alassi

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE
CALGARY, ALBERTA
MAY, 2009

© Derar Hasan Alassi 2009
PARTIAL COPYRIGHT LICENCE

This information is collected under the authority of the Freedom of Information and Protection of Privacy Act. It is required to allow the University of Calgary to reproduce these for the purpose of private study or research. Questions may be directed to the University Archives as indicated above.

Last name: ALASSIT
Given names: DEPAR
Title of Thesis: Effectiveness of Template Detection on Noise Reduction and Websites Summarization
Degree for which thesis was presented: Master of Science
Year this degree was granted: 2009

♦ I hereby grant the University of Calgary a non-exclusive licence to reproduce and to distribute copies of my thesis in print and digital format.

♦ This licence is granted on the understanding that such copies shall bear a statement that the copy in question is being made available solely for the purpose of private study and research and may not be copied or reproduced except as permitted by the copyright laws without written authority of the copyright owner.

Signature: __________________________
Date: April 29, 2009
UNIVERSITY OF CALGARY
FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "Effectiveness of Template Detection on Noise Reduction and Websites Summarization" submitted by Derar Hasan Alassi in partial fulfillment of the requirements of the degree of MASTER OF SCIENCE.

Supervisor, Dr. Reda Alhajj
Department of Computer Science

Dr. John Rokne
Department of Computer Science

Dr. Jianqing Chen
Haskayne School of Business

Tuesday, April 28, 2009
Abstract

The World Wide Web is the most rapidly growing and accessible source of information. Its popularity has been largely influenced by the wide availability of the Internet in almost every modern house. Yet, pages on the Web have noisy information that does not add value. Even worse, it can harm the effectiveness of web mining techniques. Templates form one popular type of noise on the Internet. In fact, a study done in 2005 shows that 40-50% of the Web is made up of templates. In this thesis, I introduce Noise Detector (ND) as an effective approach for detecting and removing templates from web pages. ND segments web pages into semantically coherent blocks. Then it computes content and structure similarities between these blocks; a presentational noise measure is used as well. ND dynamically calculates a threshold for differentiating noisy blocks. ND can detect a template of a website with high accuracy using two pages only. Further, ND leads to website summarization. Experiments show that ND outperforms existing approaches. Furthermore, a user study emphasizes the positive impact of removing templates on information retrieval systems and web summarization tools. Finally, ND can be used as a pre-processing tool for web mining applications.
Acknowledgements

I would like to thank my supervisor Dr. Reda Alhajj for his continuous encouragement and support, without which this thesis would not have been possible. I am deeply thankful for his patience and guidance which helped produce this thesis. I am grateful to Dr. Jon Rokne and Dr. Jianqinq Chen for serving on my thesis defense committee and for their discussions and valuable inputs. I would like to thank Islam Higazi and Heather Maki for their valuable comments that improved the quality of the thesis editing. I would like also to express my love to all my lab mates who supported me all the time. Special thanks go to Abed, Ahmad, Anas, Brenan, Mohammad, Moustafa, Thaer, Wadhah, and Yaser who volunteered their time in evaluating our proposed system.

I wish to especially thank Mahmoud Jarrar, Mohammad Al-Shalalfa, and Alaa Kassab for their endless support and encouragement which they gave me every day. I cannot pay back these and the other friends who were my other family here in Calgary.

My appreciation and gratitude go to the main office staff in the Computer Science Department for their diligence and continuous help and support.

Finally, I would like to thank my family and friends in Palestine who supported me endlessly and put in me all the confidence that I needed. Special thanks go to my parents, brother, sisters, and niece.
Dedication

for

my parents Wafa' and Hasan

the souls of my martyr friends who left this life defending the land of Palestine: Abood, Fathi, and Rami

all who believe in human justice and freedom

my beloved city Nablus
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approval Page</td>
<td>ii</td>
</tr>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>iii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>v</td>
</tr>
<tr>
<td>List of Figures and Illustrations</td>
<td>ix</td>
</tr>
<tr>
<td>List of Symbols, Abbreviations and Nomenclature</td>
<td>xi</td>
</tr>
<tr>
<td><strong>CHAPTER ONE: INTRODUCTION</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem Definition and Motivation</td>
<td>2</td>
</tr>
<tr>
<td>1.2 The Proposed Approach</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Contributions</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Thesis Organization</td>
<td>8</td>
</tr>
<tr>
<td><strong>CHAPTER TWO: BACKGROUND AND RELATED WORK</strong></td>
<td>10</td>
</tr>
<tr>
<td>2.1 Data Mining</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Web Mining</td>
<td>12</td>
</tr>
<tr>
<td>2.3 HTML</td>
<td>14</td>
</tr>
<tr>
<td>2.3.1 HTML Structure</td>
<td>14</td>
</tr>
<tr>
<td>2.3.2 HTML DOM-tree</td>
<td>16</td>
</tr>
<tr>
<td>2.4 Related Work</td>
<td>17</td>
</tr>
<tr>
<td>2.4.1 Presentation-based Approaches</td>
<td>18</td>
</tr>
<tr>
<td>2.4.2 DOM-based Approaches</td>
<td>19</td>
</tr>
<tr>
<td>2.4.3 Segmentation-based Approaches</td>
<td>20</td>
</tr>
<tr>
<td>2.5 Our Proposed Approach</td>
<td>24</td>
</tr>
<tr>
<td><strong>CHAPTER THREE: THE PROPOSED SYSTEM - NOISE DETECTOR</strong></td>
<td>25</td>
</tr>
<tr>
<td>3.1 Noise Detector Architecture</td>
<td>25</td>
</tr>
<tr>
<td>3.2 Template Detection Process</td>
<td>26</td>
</tr>
<tr>
<td>3.2.1 Block versus Whole Page based Detection</td>
<td>26</td>
</tr>
<tr>
<td>3.2.2 Page Segmentation</td>
<td>29</td>
</tr>
<tr>
<td>3.2.3 Valid-Pass Filter</td>
<td>31</td>
</tr>
<tr>
<td>3.2.4 Noise Matrix</td>
<td>32</td>
</tr>
<tr>
<td>3.2.4.1 Content Similarity</td>
<td>33</td>
</tr>
<tr>
<td>3.2.4.2 Structure Similarity</td>
<td>35</td>
</tr>
<tr>
<td>3.2.4.3 Case Study</td>
<td>43</td>
</tr>
<tr>
<td>3.2.5 Block Matching</td>
<td>46</td>
</tr>
<tr>
<td>3.2.5.1 Presentational Noise Measure</td>
<td>55</td>
</tr>
<tr>
<td>3.2.6 Final Noise Measure</td>
<td>58</td>
</tr>
<tr>
<td>3.2.7 Noise Threshold Value</td>
<td>59</td>
</tr>
<tr>
<td>3.2.7.1 Further Valid Blocks Refinement</td>
<td>59</td>
</tr>
<tr>
<td>3.3 Closing Remarks</td>
<td>60</td>
</tr>
<tr>
<td><strong>CHAPTER FOUR: EXPERIMENTAL ANALYSIS</strong></td>
<td>62</td>
</tr>
<tr>
<td>4.1 The Testing Environment</td>
<td>62</td>
</tr>
</tbody>
</table>
List of Tables

Table 3.1: Content Similarity Matrix ................................................................. 43
Table 3.2: Structure Similarity Matrix ................................................................. 45
Table 3.3: Content Similarity Matrix (ContSim) .................................................. 52
Table 3.4: Structure Similarity Matrix (StructSim) .............................................. 52
Table 3.5: Combined Similarity Matrix (CombSim) ............................................. 52
Table 4.1: Number of Blocks per Page for the BBC Website ............................ 67
Table 4.2: Occurrence Percentages of Valid Blocks of Different Pages from the BBC Website ................................................................................................................. 68
Table 4.3: Occurrence Percentages of Valid Blocks of Different Pages from the CNN Website ................................................................................................................. 69
Table 4.4: Number of Blocks per Page for the CNN Website ............................ 70
Table 4.5: Occurrence Percentages of Valid Blocks of Different Pages from the J&R Website ................................................................................................................. 70
Table 4.6: Number of Blocks per Page for the CNN Website ............................ 71
Table 4.7: Average Consistency Results .............................................................. 72
Table 4.8: Number of Blocks in Different Websites ............................................ 73
Table 4.9: Output of Cross Checking $dl$ Against $D-\{dl\}$ .................................. 76
Table 4.10: PDOC Values of VIPS ........................................................................ 77
Table 4.11: Precision, Recall and F1-score Results of the Noise Detector in Identifying Noisy Blocks .............................................................................................................. 81
Table 4.12: Accuracy Measures of Noise Detector ............................................. 82
Table 4.13: Using Three Pages for Cross-Testing ............................................... 86
Table 4.14: Comparison between Different Approaches .................................... 90
Table 4.15: MPOS Gain values for Different Query Types ............................... 98
Table 4.16: MPOS Gain values for Randomly Selected Queries ...................... 98
Table 4.17: Question-based Evaluation .............................................................. 100
Table 4.18: Precision of HavenWorks ................................................................. 104
Table A.1: VIPS Rules in Sorted by their Priority ............................................. 119
List of Figures and Illustrations

Figure 1.1: Noise Detector Architecture........................................................................... 5
Figure 3.1: Similar Blocks in Different Pages Extracted from CNN ........................................ 28
Figure 3.2: VIPS Input/Output.......................................................................................... 31
Figure 3.3: Content Similarity Algorithm............................................................................... 34
Figure 3.4: Structure Similarity Example................................................................................ 35
Figure 3.5: Similarity between HTML DOM Trees................................................................. 36
Figure 3.6: General Tree Mapping Example .......................................................................... 38
Figure 3.7: Simple Tree Matching Algorithm (STM) (Yang, 1991) .......................................... 40
Figure 3.8: STM Example...................................................................................................... 41
Figure 3.9: (a) W Matrix of First-Level Subtrees; (b) M Matrix of First-Level Subtrees ............ 42
Figure 3.10: Block Matching when $m = n$ ......................................................................... 47
Figure 3.11: Block Matching when $m \neq n$ ......................................................................... 48
Figure 3.12: MatchBlocks Algorithm .................................................................................... 49
Figure 3.13: Detailed Flowchart of Noise Detector................................................................. 50
Figure 3.14: Visual Layout of the Matching Block Pairs .......................................................... 53
Figure 3.15: Unmatched blocks from p2 ............................................................................... 55
Figure 3.16: Examples on Presentational Noisy Blocks from BBC ......................................... 56
Figure 3.17: PresentationalNoise Algorithm ......................................................................... 57
Figure 3.18: UpdateWeights Algorithm ............................................................................... 58
Figure 4.1: An Example Page from BBC Business Section ..................................................... 66
Figure 4.2: An Example Page from BBC Technology Section ............................................... 67
Figure 4.3: Number of Valid Blocks for Different Websites ..................................................... 84
Figure 4.4: F1-score Measure of ND vs. FR ........................................................................... 85
List of Symbols, Abbreviations and Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ND</td>
<td>Noise Detector</td>
</tr>
<tr>
<td>LH</td>
<td>Lin &amp; Ho</td>
</tr>
<tr>
<td>SST</td>
<td>Site Style Tree</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper-Text Markup Language</td>
</tr>
<tr>
<td>IR</td>
<td>Information Retrieval</td>
</tr>
<tr>
<td>VIPS</td>
<td>Vision-based Page Segmentation</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>KDD</td>
<td>Knowledge Data Discovery</td>
</tr>
<tr>
<td>ContSim</td>
<td>Content Similarity</td>
</tr>
<tr>
<td>StructSim</td>
<td>Structure Similarity</td>
</tr>
<tr>
<td>STM</td>
<td>Simple Tree Matching</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>IBDF</td>
<td>Inverse Block-Document Frequency</td>
</tr>
<tr>
<td>PDOC</td>
<td>Permitted Degree of Coherence</td>
</tr>
<tr>
<td>P</td>
<td>Precision</td>
</tr>
<tr>
<td>R</td>
<td>Recall</td>
</tr>
<tr>
<td>MPOS</td>
<td>Mean Position of First Relevant Result</td>
</tr>
<tr>
<td>TN</td>
<td>True Negative</td>
</tr>
<tr>
<td>TP</td>
<td>True Positive</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive</td>
</tr>
<tr>
<td>std</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td>F</td>
<td>F-score</td>
</tr>
<tr>
<td>P@5</td>
<td>Precision at Rank 5</td>
</tr>
<tr>
<td>P@10</td>
<td>Precision at Rank 10</td>
</tr>
</tbody>
</table>
Chapter One: Introduction

The wide spread of the Internet has turned the World Wide Web into the primary source for people to seek information and knowledge, although it is still hard for users to retrieve all the useful information they need. The main reason could be the fact that there is no single authority that controls the Internet and, therefore, Web resources are generally scattered in an unstructured fashion. In addition, data on the Web is known to be heterogeneous and growing constantly. As a result, the field of web mining has become one of the most helpful means to achieve better user satisfaction. One of the challenges that faces web mining is that there is a lot of noisy information on the Internet.

Noisy information is any information that exists in a page and is not related to the main topic of that page, such as banners, logos, templates, and copyrights. On the other hand, non-noisy information, informative data, covers the information that is related to the main topic of a page.

In this thesis, we investigate the problem of noisy information and propose a new technique that detects and removes noisy information from Web pages. This is referred to as data cleansing. Noise detection will eventually allow us to discover the informative parts in these pages which are the non-noisy parts. This process is considered as part of what is called data pre-processing for web centric applications, like web mining.

In general, data pre-processing is a crucial step in data mining. Noise elimination is considered a pre-processing step because it is done prior to feeding the data to any mining system. The data cleansing (noise elimination) will boost the accuracy results of the mining systems as we will show in Chapter 5.
1.1 Problem Definition and Motivation

With the emergence of the Internet and the massive amount of available content, mining the Web has become an inevitable fact that researchers have focused on. Bing Liu states that the Internet is perhaps the largest source of information that anyone can access at anytime (Liu & Chen-Chuan-Chang, 2004). (World Internet Usage Statistics News and World Population Stats) shows that around 1.5 billion people use the Internet.

People have different interests when using the Internet. Some of them might be interested in searching for Web resources using the so-called search engines, such as Google™¹ or Yahoo™². Other users might want to get summaries of Web pages using summarization utilities to save their time reading the whole pages (Buyukkokten et al., 2001; Delort et al., 2003). We will show in Chapter 5 how these summaries can be integrated with other applications like search engines. More advanced users could be interested in automatically extracting information from pages for later processing. This can be achieved by the so-called information extraction systems (Papadakis et al., 2005; Ashraf et al., 2008; Chang et al., 2006). These different systems are all examples of Web Mining Systems.

A Web mining system takes the input data through three different stages to reach its final result: namely pre-processing, data mining and post-processing (Liu B., 2006). Since data is the input in a data mining system, pre-processing becomes essential to transform this raw data into an acceptable format. Pre-processing may include eliminating attributes that are irrelevant and/or cleaning the data from noisy information.

¹ www.google.com
The latter step is very important in data pre-processing since data on the Web includes a high level of noisy information (Tseng & Kao, 2006; Lin & Ho, 2002; Vieira et al., 2006).

Web noise can be defined as information present in a web page and not relevant to the main topic of the page. For example, when there is an article with a specific title, all the copyright, banners, advertisements and other redundant segments within that website are considered noise because they do not add any value to the article. If web mining systems (search engines, summarization utilities, recommender system, information extraction systems) have noisy data as their input, the results will not be accurate (Li & Ezeife, 2006; Carvalho, 2006; Debnath et al., 2005).

Two types of noise that exists on the Web have been defined in literature: *local noise* and *global noise* (Yi et al., 2003).

*Global noise*: This type of noise exists with large granularity, and includes mirror sites, duplicated web pages and near-replicas. Global noise not only tamper the quality of crawling and ranking function of web information retrieval systems, it also forces archiving applications to waste large disk space for storing the duplicated pages.

*Local noise*: This covers noisy content within a web page. Local noise is usually incoherent with the main topic of the Web page, e.g., advertisements, navigation panels, copyright announcements, etc. Local noise makes it very difficult for programs to automatically grasp the topic content of a page, i.e., local noise

---

2 www.yahoo.com
worsen the quality of web applications. The first step to discover local noise is by segmenting a page into blocks.

**DEFINITION 1.1** [Block]: A block is a semantically and visually coherent area of a Web page.

**DEFINITION 1.2** [Intra-Site Noise]: Intra-site noise is the set of blocks that are redundant over pages in the same Website.

In fact websites do incorporate some information (like advertisements) that should be displayed with each page of the website. Such information is essential according to website owners because it might be the source of their profit. However, it is considered to be noise by most visitors who are interested in the actual content of the website. In other words, a large number of web sites have predefined templates that shape the web pages of the site. A recent study by Gibson et al. (2005) shows that templates represent between 40% and 50% of the Web, and they are growing every year. To cope with noisy information, the approach proposed in this thesis segments web pages into blocks (see definition 1.1), and then processes the blocks to find out those which make up the site template. We are mainly interested in intra-site noise (templates) as specified in Definition 1.2.

This thesis proposes an integrated approach called Noise Detector which is capable of investigating and eliminating intra-site (template) noise. The extensive experiments that we have conducted show that Noise Detector achieves high precision and recall rates. Better results can be achieved by using different web mining techniques.
1.2 The Proposed Approach

Figure 1.1 shows the main modules of the proposed Noise Detector. It has 5 main modules: page segmentation, DOM-tree builder, filter, noise matrix computer, and the cut-off value calculator. The input to the Noise Detector is only two pages. The small number of pages required as input gives the proposed approach strength over the other approaches described in the literature (More details will be covered in Chapter 2).

The two input pages are segmented into blocks using a utility called Vision-based Page Segmentation (VIPS) (Cai et al., 2003). Each page is segmented on the semantic level to generate coherent areas. Then a Document Object Model tree (DOM-tree) is built.
for each block. The DOM-tree is a standard for accessing and manipulating HTML documents (W3C). (See Section 2.3 for more details).

The blocks are validated by the valid-pass filter. A valid block has content of size greater than or equal to a pre-defined threshold. This is considered an early block elimination step. The valid blocks are passed to the noise matrix module that compares the content and structure of each block in p1 against all the blocks in p2. In other words, the noise measure being used in the Noise Detector relies on the content and the structure similarity between blocks in the input pages. Essentially, this is a template detection approach since we are using the similarity between blocks to determine their noisiness.

After we compute the noise matrix, we assign to each block in the two pages its corresponding noise value. If the number of blocks in p1 and p2 are the same, then the first block in p1 will be compared against the first block in p2, and the second block from p1 against the second block in p2, and so on. In case the number of blocks in p1 is different than the number of blocks in p2, a block matcher algorithm is applied. This algorithm finds blocks from p2 that match blocks in p1. All blocks left in p1 without matching blocks from p2 will be assigned noise values using only their presentational features. Section 3.2 shows the different scenarios with detailed examples on how to compute the noise values of each block in p1 and p2. The last stage before outputting the valid blocks is to calculate the cut-off value which defines the threshold that determines the valid blocks. This value is calculated dynamically; thus it is case-dependent, i.e., a page could have different cut-off values when it is matched against different pages. More information on calculating the cut-off value is included in Section 3.2.6. After calculating the cut-off value, blocks with higher noise value than the calculated threshold are marked
as noisy. The final result of the system is the set of blocks that are not noisy. We combine all the latter blocks to create the new cleaned HTML file.

As a byproduct, Noise Detector is able to identify the sites that follow a predefined template and whether this template is consistent throughout the pages of that site or not. This is achieved by comparing the number of blocks in different pages from that site and their respective noise values. In other words, it is possible to comment on the design of a website by determining whether the same template is used across the site to specify the extra information in each of its pages. A future extension that benefits from the outcome of the Noise Detector is restructuring blocks within pages of a website by forcing all the extra information to follow a common template. Such a process would make it easier for users to surf the website and examine its useful content (assuming the extra information like advertisements is not useful and considered as noise).

1.3 Contributions

The proposed Noise Detector is a useful tool for cleaning websites before they are further processed to serve certain targets like web mining. The following points can summarize the main contributions of the work described in this thesis:

- We propose Noise Detector to detect intra-site noise. The output of this system is clean pages that can be used as input to Web mining techniques. This system is implemented and can be run in an automated way.
• Noise Detector operates with two pages; this is optimal as compared to the other approaches described in the literature. Experiments show that the precision and recall values of detecting noisy blocks are high.

• We propose a complex and robust noise measure. This measure includes content, structure and presentation of a web page. To the best of our knowledge, this is the first measure that combines these three features to detect templates.

• We have conducted a number of thorough experiments to evaluate our system and to validate the measures that we have proposed. These experiments include evaluating information retrieval systems and summarization utilities. To avoid any bias and to show that Noise Detector has not been engineered to serve specific cases, a variety of websites from different domains have been used in the testing. The reported results demonstrate the applicability and effectiveness of the Noise Detector.

1.4 Thesis Organization

The main component of this thesis is the Noise Detector as outline above. Its different components are described, tested and validated in the following chapters. The necessary background is also covered in order to have this document more self contained. The remainder of this thesis is organized as follows.

Chapter 2 discusses some of the related work. In addition, it provides an overview of the background. In this chapter, we discuss the concept of data mining in general and
web mining in particular. Since the research conducted is in the application side, we will review some web mining applications, information retrieval systems, and information extraction systems in order to lay the ground for integrating our work with these systems later. A brief introduction to HTML is also covered.

In Chapter 3 we describe the proposed Noise Detector system. The modules of Noise Detector are investigated in detail. We also cover the VIPS algorithm that we use in our work as opposed to the other algorithms that have been proposed in the literature. We discuss the building of the DOM-trees and the related steps. Then we discuss the filter that we use to eliminate the invalid blocks early. We show how to compute the noise values for all the blocks in the input pages. We have case studies to show the different practical cases we may have. Finally, we discuss the dynamically calculated threshold.

Chapter 4 investigates the applicability of Noise Detector as a noise detection system. Different measures, including precision and recall results are reported for the tested sites. Then we emphasize the importance of our work by applying it to web summarization tools and search engines. We discuss the user study that we have conducted and try to analyze the results.

Chapter 5 is summary, conclusions and future research directions.
Chapter Two: Background and Related Work

This chapter serves two purposes. I first present the basics of the background necessary for the reader to understand the details of our proposed approach. I mainly concentrate on data mining concepts and web mining in particular. Then I will cover the related work.

2.1 Data Mining

Data mining refers to the process of extracting hidden knowledge in data repositories (Dunham, 2002; Han, 2000). Knowledge discovery in databases (KDD) is another commonly used term to refer to data mining. Data mining is a multi-disciplinary field that benefits from databases, machine learning, statistics, artificial intelligence, information retrieval, information extraction, and visualization (Liu B., 2006).

Data mining tasks involve classification, clustering, estimation, prediction, association rule mining, and outlier detection, among others (Witten & Frank, 1999). The common part amongst these techniques is that their goal is to extract knowledge from the input data. However, they follow different steps to achieve this goal. Classification is a supervised mining technique in which the classes are known in advance and a model is learned using labeled data examples; the constructed model determines the class of new instances. On the other hand, clustering is an unsupervised technique where the input data is not labeled and should be classified into groups using a distance measure (Berkhin, 2006). Most clustering algorithms require the number of clusters as input. Clustering and classification techniques mostly deal with high dimensional data and hence feature reduction is essential step to preprocess the data into manageable number of dimensions.
Estimation is a technique similar to classification but the nature of output results is different. In particular, classification deals with discrete outcomes while estimation deals with continuously valued outcomes. Prediction is another process that uses existing data to predict future behavior (Dunham, 2002). Another mining technique is association rule mining, which groups the items that appear together in many transactions (Liu et al., 1998). The development of apriori by Agrawal et al. (1993) in 1993 can be considered as the event that marked the start of the tremendous interest in data mining.

The last data mining technique overviewed in this section is outlier detection. This technique discovers data points that are dissimilar, exceptional or inconsistent with respect to the whole data set (Li et al., 2006; Angiulli et al., 2006). This could be considered as a pre-processing step in the data mining process. However, it is worth mentioning that outliers are not noise. While noise refers to erroneous data and should be eliminated, outliers represent valuable information that highlights exceptional cases like credit card fraud. Some clustering techniques are capable of detecting outliers while others force outliers into existing clusters. The latter methods could lead to distorted clusters, a case that can be avoided by eliminating outlier as part of the noise in the preprocessing step.

We will evaluate the proposed system using clustering, classification, an information retrieval system, and web summarization system. The latter two systems are considered as essential tools for producing powerful web mining systems, which is at the core of this thesis work. The basics of Web mining are covered in Section 2.2.
2.2 Web Mining

Due to the rapid growth of the Web and the huge amount of available information there, web mining has attracted considerable interest in the research community. Despite this interest, it still faces many challenges. This is due to, but not limited to, the following reasons (Liu & Chen-Chuan-Chang, 2004; Kosala & Blockeel, 2000; Liu B., 2006):

- The huge amount of information available on the web; the Web is dynamic and changing constantly and rapidly.
- The information on the Web is heterogeneous, consists of structured, semi-structured and unstructured data.
- Users add on to the complexity of the Web by their reviews, blogs, and other ways of participation.
- Considerable amount of the Web information is redundant, which means that there are segments that appear to be identical in many pages or sites. This raises serious consistency issues that need to be handled.
- The Web is considered noisy by users who are not interested in the ads, copyrights notices, banners, and the redundant pieces of information that exist across the pages they visit, though such parts of a page might be the most important for the owner.

In general, web mining is classified into 3 different categories (Fürnkranz, 2005):

1. Web structure mining: This field extracts knowledge from the hyperlinks that exist on the Web. It is one of the main keys that search engines use to rank
their output results. Popular algorithms such as HITS (Kleinberg, 1999) and PageRank (Page et al., 1999) are used by Google\(^3\) and Yahoo\(^4\) search engines.

2. **Web usage mining:** This field extracts knowledge from the user access patterns and usage logs. Web masters use access patterns to re-layout websites in a more user-friendly fashion (Srivastava et al., 2000).

3. **Web content mining:** This field of web mining is intended to help users retrieve the relevant information they need. Web content mining operates on the content of web pages rather than links or user access patterns as in web structure mining, and web usage mining, respectively (Liu & Chen-Chuan-Chang, 2004).

The main theme of this thesis is mining web content to eliminate its noisy parts. Thus understanding the structure of a web page is a crucial step for the reader to comprehend the whole process of our proposed system. The structure of a web page can be described by the syntax that is used to build the web page. We focus on pages coded in Hyper Text Markup Language (HTML), which may still be considered as the main markup language on the Web, though some may argue that it is kind of legacy technology given the latest developments in website encoding techniques. The basics of HTML are covered in Section 2.3.

---

\(^3\) http://www.google.com

\(^4\) http://www.yahoo.com
2.3 HTML

Hyper Text Markup Language (HTML) has been the dominant markup language of the World Wide Web (WWW) Consortium (W3C). This is due to its ease of learning and ease of implementation. Simply, any text editor is sufficient to design a web page, which can be then displayed using any web browser.

2.3.1 HTML Structure

According to W3C\textsuperscript{5}, an HTML document is composed of three parts:

i. A line that contains HTML version information

ii. A declarative header section that uses the HEAD element

iii. A body, which contains the document’s actual content

None of these parts is mandatory and can be omitted from the HTML file; this demonstrates the high flexibility of HTML. The main part in an HTML structure is the element. An HTML element is syntactically constructed with:

i. A start tag that indicates the beginning of the element.

ii. Any number of attributes (selected from a predefined set) with their associated values that define the characteristics of the element.

iii. Some content, which is the actual material to be displayed on the page.

iv. An end tag that indicates the end of the element; every start tag should have a matching end tag.

\footnote{\url{www.w3.org}}
Figure 2.1: HTML Page

```
<html dir="ltr">
<head>
<title>HTML Example</title>
</head>
<body>
<p align="center">My Page</p>
<hr>
<p align="center">
<img border="0" src='UofC.gif' width='72' height='94'></p>
</body>
</html>
```

Figure 2.2: HTML Source Code

Figure 2.1 shows a simple example of an HTML document with its corresponding HTML source code shown in Figure 2.2.

According to the HTML source code shown in Figure 2.2, the HTML tags that are shown in purple represent HTML elements which make up the HTML document. The content enclosed between "<" and ">") other than the tag name are the attributes of the element to identify its characteristics. Each attribute's name specified in red has a value which is colored in blue. The nesting of HTML tags makes the tree-like representation
the most convenient way to represent an HTML document. In other words, HTML Document Object Model (DOM) tree is what results from representing the nested HTML tags.

2.3.2 HTML DOM-tree

The HTML Document Object Model (HTML DOM) defines a standard way for accessing and manipulating HTML documents (W3C). Figure 2.3 shows the HTML DOM-tree of the example depicted in Figure 2.2.

Nodes in the DOM-tree shown in Figure 2.3 represent the HTML tags; their corresponding attributes and content are attached to them in square boxes. Since the presence of attributes or content is optional, there are some nodes without either of them,
e.g., head, hr. Some nodes have content without attributes and some others have attributes without content, e.g., title and p, respectively. All the input pages to our system will be transformed into DOM-trees which will be processed later using trees-specific algorithms.

After building the background required for understanding the research conducted in this thesis, Section 2.4 will discuss the related work that has been proposed in the field of noise detection and noise elimination.

2.4 Related Work

The main goal of this thesis is to detect and eliminate web noise that is part of templates. Our work as described in this thesis overlaps with several research areas, including:

- Noise detection on the Web
- Noise elimination on the Web
- Template detection in Websites
- Informative segments detection on the Web
- Informative blocks detection on the Web
- Web summarization

Some of these topics are interchangeable, e.g., noise detection and template detection (see Definition 1.2). However, discovering informative blocks in a web page and noise detection on the Web are complementary techniques. Therefore, we will survey
the different pieces of work that cover these approaches. Most importantly, we will discuss how each technique tackles the problem and what shortcomings each approach has. Then, we will briefly discuss how we address these problems in our proposed approach.

A large percentage of web sites have predefined templates that shape their web pages within each site (Bar-Yossef & Rajagopalan, 2002; Lo et al., 2006; Vieira et al., 2006; Wang et al., 2008). Gibson et al. (2005) show in their study which was published in 2005, that templates represent between 40% and 50% of the Web. Moreover, this percentage is increasing every year. The information that is relevant to the main topic of the page is normally extracted from backend databases, and it is different from one page to another. In contrast, parts of a template are highly consistent in their content and structure and reoccur in different pages, even across websites. Thus, these parts are referred to as intra-site redundant parts (see Definition 1.2) (Meng et al., 2008).

Next, we will investigate the approaches that have been introduced in the literature to solve the problem of detecting templates and detecting informative blocks in web pages. We classify these approaches into three different types: presentation-based approaches, DOM-based approaches, and segmentation-based approaches.

2.4.1 Presentation-based Approaches

Gupta et al. (2003) uses a naïve approach to remove advertisements and links based on the ratio of the number of linked and non-linked words in a DOM-tree. The linkage ratio in a block is a general indication that the block is likely to be part of the template, but this is not enough. For example, a block that has copyright information at
the end of a page does not have high linkage ratio; however, it is part of the template. On the other hand, if a page has a lot of useful links, e.g., MSN⁶, most of them will be removed, which is not acceptable.

2.4.2 DOM-based Approaches

Lan Yi et al. (2003) introduced an interesting approach to eliminate the noisy information from web pages. They proposed the Site Style Tree (SST), which is discovered by processing a set of web pages from a specific web site. Initially, the SST tree is the DOM-tree of the first document. Then they build the DOM-tree of each web page and match it with the existing trees. Thus, the SST is an accumulative tree of all the mapped trees (union). While matching a new tree, if the node in the new DOM-tree exists in the SST, then the frequency of that node is incremented. Otherwise, this node will be inserted into the SST tree with initial frequency of 1. To build a "site-representative" SST, they require 400-500 pages from that site. This is a large number; especially when compared to our approach which requires two pages at a time.

After the SST tree is constructed, the frequency of nodes will identify nodes that are part of the template. Essentially, the higher frequency a node has, the more likely this node is part of the template. Nonetheless, they use information entropy to calculate node importance; they will eventually identify the noisy nodes that have low importance values. Presentational features are used to calculate the importance values as well.

⁶ http://www.msn.com/
Lo et al. (2006) proposed CF-EXALG (Collaborative Finer-EXALG), which is based on the information extraction system EXALG (EXtraction ALGorithm). CF-EXALG parses HTML pages to extract sets of words that have similar occurrence patterns in different pages. As a result, equivalent classes are constructed to represent these occurrence patterns. Then any newly unseen pages will be parsed in the same fashion to extract their tokens (frequently occurring patterns). After that, the discovered tokens are matched against the equivalent classes that were deduced from previous pages. The last stage is to output the template schema using the frequent patterns as an XML DOM tree\(^7\). This algorithm basically uses the content to extract tokens which will be used to identify the template of that site. This means that content is the only feature used and therefore, this needs a large number of pages to produce good results. Another shortcoming of their evaluation is not conducting extensive experiments to show precision and recall accuracy results. The only measure they used is the “correctness” which is not defined clearly.

2.4.3 Segmentation-based Approaches

The algorithms proposed under this category either use a previously proposed segmentation algorithm or propose a segmentation algorithm.

Lin & Ho (2002) discovers the informative content in news Web sites. They use the \texttt{<TABLE>} tag as a “container tag” to extract content strings in each block. Then, the entropy of each feature in each block is calculated. Finally, the final entropy value of a

---

\(^7\) XML DOM-tree is a standard way for accessing and manipulating XML documents. XML DOM-tree
block is defined as the summation of its features' entropy values. This work is limited to news Web sites as appeared in their experimental section. In addition, it only uses the \(<\text{TABLE}>\) tag as the “container tag” to discover the informative blocks, which is not always the case. Clearly, there are other tags that are used to layout blocks in web pages, such as using horizontal lines (\(<\text{HR}>\)), or using the division tag (\(<\text{DIV}>\)). Yet, utilizing the visual layout of a web page is more effective in the segmentation process.

Bar-Yossef and Rajagopalan (2002) do utilize the visual and functional structure of web pages in their work. They segment web pages into pagelets which are equivalent to blocks in our work. A pagelet is essentially a DOM-tree element that has at least \(k\) hyperlinks, and none of its ancestor elements is a pagelet. Therefore, a pagelet is meant to be an area with single and well-defined functionality that is not shared with another area. After pagelets have been identified, \(\text{almost-similarities}\) are computed between pagelets using a shingling\(^8\) technique.

Song et al. (2004) proposed a block importance model that automatically assigns importance weights to different blocks (regions) in a web page. VIPS (Cai et al., 2003) is used first to segment web pages. Spatial and content features are then used to calculate block importance. Spatial features include: x-coordinate of a block center, y-coordinate of a block center, block width, and block height. On the other hand, content features include image number, interaction size, form number, and form size. A set of labelled blocks that have been already extracted from different pages are used to build a vector.

\(^8\) presents XML documents with \textit{elements}, \textit{attributes} and \textit{text} as nodes same as in an HTML DOM-tree. A \textit{shingle} is a text fingerprint that is invariant under small perturbations (Template detection via data mining and its applications, 2002).
This vector represents all features that belong to the analyzed block; it is used later with its corresponding importance value to find out the best function $f$ that minimizes the expression $|f(x,y) - y|$, where $x$ is the feature vector of a block, and $y$ is its importance. They solve this problem as a regression problem by considering that the input is a continuous variable. However, it is considered as a classification problem when the input is discrete. This consideration requires labelling example data sets first, and then building these models to be applied on unseen data sets.

Li & Ezeife (2006) proposed WebPageCleaner which detects and removes noisy blocks from web pages. VIPS is used to segment a set of web pages into a set of blocks. Then, content similarity of the extracted blocks is computed as well as their linkage percentage. Text fingerprints⁹ are used to calculate the content similarity between blocks. Another measure is calculated using the position of these blocks which will be later combined into one importance measure. Blocks with the highest $N$ importance values are then used to create the final clean file. They assume $N=3$; which is not a very valid assumption. In Section 4.3.3, we show that the number of valid blocks is variable; it is website-dependent; even worse, it varies from one page to another in the same website. We also compare our approach against the approach of Li & Ezeife (2006).

Debnath et al. (2005) proposed an approach which is close to our work. Page segmentation is first done using a set of tags that are defined as potential "container tags". These tags include <TR>, <P>, <HR>, and <UL>. Consequently, this work can be

---

⁹ Let $S$ be a string of $n$ symbols over some alphabet $\Sigma$. We say that a substring $S'$ occurring within $S$ has fingerprint $\sum_{S'} \subseteq \Sigma$. Also, the fingerprint $\sum_{S'}$ is called the alphabet of $S'$. (Efficient Text Fingerprinting via Parikh Mapping, 2003)
said to be a more general approach than the approach in (Lin & Ho, 2002) where they use only the table tag (<TABLE>). A block is defined as the set of HTML tags along with their corresponding content that reside between an open tag and its corresponding closing tag. Then, FeatureExtractor and ContentExtractor modules identify importance values for these blocks. FeatureExtractor depends on heuristics and certain features that must exist to consider a block to be valid. However, it is not clear what these features are. The other ContentExtractor module marks blocks to be valid when they contain pre-defined content features, e.g., number of terms, number of images, number of java-scripts, etc. Afterwards, the Inverse Block-Document Frequency (IBDF) measure is calculated; it is inspired from the Term-Frequency-Inverse Document Frequency (TF-IDF) which is a weight measure often used in information retrieval and text mining. TF-IDF is a statistical measure used to weigh the importance of a word (term) in a document. The importance of a term increases proportionally to its frequency in a document but it is offset by the frequency of the word in the corpus. Similarly, IBDF assigns a higher value to a block that appears less frequently than another block which appears more frequently in a set of web pages extracted from a website. A threshold (like 0.9) which is provided by an expert determines whether two blocks are similar or not. The experimental results show that the approach proposed in (Debnath et al., 2005) outperforms the entropy-based approach proposed by Yi and Liu (2003). Nevertheless, (Yi et al., 2003) had been proved to outperform the template-based approach in (Bar-Yossef & Rajagopalan, 2002).

In our approach, three different features are used: content, structure, and presentation. The final noise value depends on these features with a different weight value for each one. We identify the weights of these features for each block dynamically
depending on the characteristics of the particular block. Therefore, different blocks can have different weight values even if they belong to the same page. In some cases where blocks have extremely high content similarity values, we need to bias the overall noise measure to this value. This is not the case however in (Debnath et al., 2005), where static weight values are predefined and used. More details on how these weights are calculated are provided in Chapter 3.

2.5 Our Proposed Approach

The aforementioned shortcomings of the existing approaches have motivated our work described in this thesis, which is intended to be a more comprehensive and robust approach. Our proposed approach operates on the semantic level of a web page; it segments a page into coherent blocks using VIPS. These blocks are then compared with each other using a robust measure that incorporates content, structure, and presentation. Then a dynamically computed cut-off value determines which blocks are valid. In general only two input pages are needed for our approach to give good results. Chapter 3 discusses the details of our approach, and Chapter 4 reports the experimental results along with comparisons with some of the approaches aforementioned in this chapter.
Chapter Three: The Proposed System - Noise Detector

The effectiveness of a system depends largely on its input. This means that a system with good quality input gives better results than a system with less quality input. Templates which currently make 40-50% of the Web, are considered as noise, and therefore can degrade the accuracy of a system that uses web documents as input.

This chapter investigates all the details about the proposed system, the Noise Detector. We start by presenting the general architecture of the Noise Detector. Then the different modules that the system consists of will be explained. Examples and case studies will go along to enable the reader understand the different practical cases that could happen.

3.1 Noise Detector Architecture

This thesis concentrates on identifying templates as noisy parts of web documents, and hence develops the Noise Detector as an effective approach capable of serving the target. Most of the existing approaches build a model using a training set of annotated documents (Yi et al., 2003; Debnath et al., 2005; Lin & Ho, 2002). The model is built using different machine learning techniques and similarity measures. To be able to build such a model, a relatively large set of documents is required. For example, (Yi et al., 2003) needs 400-500 pages from a site to be able to build the SST tree that will be used then to identify the parts of a template. (Vieira et al., 2006) requires more than 100 pages to give satisfactory results in detecting templates. On the other hand, Noise Detector as described in this thesis requires only two pages to detect their common template with high confidence.
The two input pages that are fed into Noise Detector go through five main modules to reach the final stage of outputting clean HTML files. The two pages are first segmented into blocks and then processed to calculate their noise values. These noise values determine which blocks are valid and which are not. The architecture of Noise Detector is depicted in Figure 1.1 (see Chapter 1).

3.2 Template Detection Process

Noise Detector uses segmentation first in its template detection process. The result blocks from the segmentation process of a page are then compared against blocks of another page. In this comparison, we find similarity of their structure and content which are later combined into one similarity measure. High combined measures of specific blocks means that these blocks are part of the template. Although Noise Detector could be classified as one of the approaches that perform the analysis at the block, other approaches zoom out to consider the whole page in the process. Section 3.2.1 discusses the different approaches described in the literature in terms of the unit used in the detection process, i.e., a block or a whole page. Section 3.2.2 discusses Noise Detector’s detection process in more detail.

3.2.1 Block versus Whole Page based Detection

Chapter 2 showed the main approaches in the literature which detect templates and consequently eliminate them from web pages. Intuitively, we can classify these approaches into two categories according to their basic processing unit:
1. Page-based approaches: this type of approaches depends on the DOM-tree of the whole page to discover patterns that recur in multiple web pages of the same site. (Yi et al., 2003; Chakrabarti, 2001) are examples on this category. For instance, (Yi et al., 2003) proved to be effective in its accuracy results, though it needs 400-500 pages from a site to detect its template. Furthermore, high complexity of the algorithm may result because of the need to process the whole DOM-tree of web pages.

2. Block-based approaches: this type considers blocks as the processing unit as opposed to a whole web page. (Lin & Ho, 2002; Bar-Yossef & Rajagopalan, 2002; Song, et al. 2004) are some examples on this category. They segment each page into a set of blocks and then process this set using different techniques (see Section 2.4.3). Processing DOM-trees of smaller sized-blocks is more efficient than processing DOM-trees of complete pages. More importantly, the visual design that every web page follow justifies the segmentation process. Nevertheless, segmentation has to make use of the visual design; and consequently operates on the semantic level of a web page.

DOM tree was initially introduced in the browser for presentation. Therefore, it does not provide much information about the semantics of a web page. That is, two nodes in the DOM tree might have the same parent, yet they might not be semantically related to each other. This is considered as a drawback of approaches that use the whole DOM tree without considering the semantics of a web page. Another drawback for the whole page-based approaches is the flexibility in the HTML syntax. Even worse, web designers do not always follow the W3C HTML specifications.
On the other hand, block-based (segment-based) approaches overcome the aforementioned drawbacks by simply utilizing the semantics of web pages. Usually, humans do not perceive a web page as one segment, but rather as a multi-segment area (Song, et al. 2004). In fact, a web designer divides a web page into semantically different areas, each of which is intended to serve a specific function in the page. For example, a page might have copyright segment, banners segment, and links to related pages segment, which have different purposes.

Figure 3.1: Similar Blocks in Different Pages Extracted from CNN

---

10 CNN is a news and information network owned by WarnerMedia (previously Turner Broadcasting System) and headquartered in Atlanta, Georgia. It is available in various countries and regions through cable, satellite, and streaming services. CNN was founded in 1980 and is known for its in-depth news coverage, particularly during major events and crises globally.
Figure 3.1 shows two pages from the CNN website. Similar blocks in both pages are enclosed inside frames that have same color. This emphasizes the importance of segmenting web pages into coherent blocks.

3.2.2 Page Segmentation

The above discussion emphasizes that segmenting a web page is a valid option. The challenge that we need to address in our work is to have a good segmentation algorithm. More precisely, a good segmentation algorithm would segment a web page as much closely as a person perceives it. Actually, the basic processing unit should look as described in Definition 1.1. We will use segment and block interchangeably in the sequel because we use them to refer to semantic and coherent areas in a web page.

There are several segmentation algorithms that have been proposed in the literature, including DOM-based segmentation (Chen et al., 2001), location-based segmentation (Kovacevic et al., 2002), and VIPS (Cai et al., 2003). The first two algorithms fail to take the visual structure of a web page into consideration. Consequently, they do not perceive a web page as it should actually be perceived. However, VIPS does utilize the visual cues of a web page and gives satisfactory results. Moreover, VIPS is technically easy to integrate into a system, as the authors provide a DLL for VIPS. Last but not least, different permitted degree of coherence (PDOC) values identify how coherent the extracted segments will be. The latter point gives VIPS flexibility and robustness, i.e., different sites have different visual structures and hence

---

10 http://www.cnn.com/
different PDOC values will be a necessity when segmenting them. Through the conducted experiments reported PDOC=6 as a good general choice for our system, still it is site-variant.

VIPS processes the HTML DOM tree of a web page to extract its set of blocks. The following are the four main cues that are being used:

1. Tag cue: there are some HTML tags that visually separate regions in a web page from each other. For instance, the <HR> tag is used to draw a horizontal line, and so VIPS uses this tag as a separator cue.
2. Color cue: VIPS divides a node in the DOM tree if its background color is different from the color of its children (see Section 2.3.2 for DOM tree example).
3. Text cue: VIPS does not divide a node if most of its child nodes are text nodes.
4. Size cue: if the standard deviation of a node is greater than a threshold, then VIPS divide that node.

These are the general visual cues that VIPS uses in its segmentation process. Accordingly, there is a set of rules that totally depend on these cues. Refer to Appendix A.1 for more details on these heuristic rules.

Figure 3.2 shows VIPS as a black box with an HTML page as the input and \( n \) blocks are the output. The number of output blocks \( n \) is dependent on the coherence value PDOC. The larger the PDOC is, the larger \( n \) will be. Intuitively, this is due to the higher coherence degree that each block must have.

VIPS adds *meta data* to each output block besides its HTML content, such as page width, page height, coordinates of the top-left corner, etc. So, each block has an
HTML content which is essentially a sub-tree in the DOM tree of the original input document, as well as the meta data that VIPS adds.

3.2.3 Valid-Pass Filter

Usually, there will be some output blocks that have small textual content. This filter will not pass blocks whose textual content's sizes are less than a predefined threshold. This is considered as an early elimination step.

Some of the conducted experiments produced 25-30 blocks per page. The valid-pass filter normally passes less than half of them (12 blocks on average). In fact, the eliminated blocks do not include valuable information and will reduce the time complexity of the algorithms employed in the next modules.

![Image](Image here)

**Figure 3.2:** VIPS Input/Output
3.2.4 *Noise Matrix*

The basic idea of this research project is to find similar blocks in multiple pages. This will, consequently, imply that these blocks are part of the template of the analyzed website. Intuitively, we will need to compare every block in the first page against all blocks in the second page. Therefore, if block $b$ from the first page has a high similarity with block $a$ in the second page, this means that $b$ and $a$ are part of the template of both pages. Thus, the similarity between blocks implies the *noisiness* of the two blocks. As mentioned in Chapter 2, the measures that have been used in the literature do not benefit from all the different aspects of HTML pages, namely content, structure, and presentation. Thus, a distinguishing feature of the proposed approach is combining all the three measures to decide on the noisiness of a block.

**DEFINITION 3.1 [Similarity Matrix]:** Let $p_1$ and $p_2$ be two pages from the same website, i.e., both pages share the same layout. By assuming that $p_1$ consists of $m$ blocks $a_1$, $a_2$, ..., $a_m$ and $p_2$ consists of $n$ blocks $b_1$, $b_2$, ..., $b_n$, a similarity matrix $S_{m \times n}$ can be computed, where $s_{i,j}$ is the similarity value between $i$th block in $p_1$ and $j$th block in $p_2$.

According to Definition 3.1, the similarity between blocks is the factor that determines the noisiness of a block in the first place. Therefore, selecting the features that we consider in our similarity measure is crucial to get good results. Considering the nature of HTML justifies the use of content and structure similarities to compute the *similarity matrix* (noise matrix).
3.2.4.1 Content Similarity

Content is an important feature that enables the discovery of templates. Bar-Yossef & Rajagopalan (2002) use shingles, which are text fingerprints that are invariant under small perturbation. In our algorithm, the content of each block is extracted. Any text in between an open HTML tag and its corresponding close tag will be part of that block’s content. The next step is to calculate the similarity between the content of every block in \( p1 \) and each other block in \( p2 \). This is the worst case in terms of the time complexity of the algorithm. In the later sections, we will discuss in more details how we refine this approach to reduce its complexity.

Cosine similarity is known to perform well as far as text analysis is concerned (Nasraoui et al., 2005; Liu B., 2006). Given 2 vectors, \( X \) and \( Y \), the cosine similarity between them is defined as follows:

\[
\text{CosSim}(X,Y) = \frac{X \cdot Y}{|X||Y|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} (x_i)^2} \sqrt{\sum_{i=1}^{n} (y_i)^2}}
\]  

(3.1)

where \( |X|, |Y| \) represent the magnitude of \( X \) and \( Y \), respectively.

To get more accurate results, it is best to remove stop words ("the", "a", "is", "will", etc) from the input vectors before applying the similarity measure. Also, applying a stemming algorithm gives much better similarity results. Porter stemming is a well-known stemming algorithm (Rijsbergen et al., 1980) that has been adapted into the Noise Detector described in this thesis. It simply removes the commoner morphological and inflexional endings from English words. For instance, the words "learning" and "learned" will be stemmed into "learn" by chopping the suffixes "ing" and "ed". Figure 3.3
illustrates the algorithm for finding the content similarity. Notice that steps 3, 4, and 5 are split into three different steps for clarity, though they can be done at the same time while extracting the content of a block, and this reduces the complexity of the algorithm.

The value returned in step 7 of the content similarity algorithm in Figure 3.3 always $\in [0,1]$. The higher the similarity value between two blocks is, the more likely the blocks are part of the template.

| Input: | b1: a tree that represents a segment in web page 1 |
|        | b2: a tree that represents a segment in web page 2 |
| Output:| Similarity: a decimal value that represents the similarity between the content of both blocks |

Algorithm: **FindCosineSimilarity(b1, b2)**

1. Extract the content of b1 into Content 1
2. Extract the content of b2 into Content 2
3. Remove stop words and punctuation from Content 1 and Content 2 and do word stemming
4. Create a vector of words for Content 1 into V1 with the count of each word
5. Create a vector of words for Content 2 into V2 with the count of each word
6. Compute Cosine Similarity as $CosSim(V1,V2) = \frac{V1 \cdot V2}{|V1| \cdot |V2|}$
7. Return $CosSim$

**Figure 3.3: Content Similarity Algorithm**

A distance measure can be derived easily from Equation 3.1 as follows:
35

\[ \text{CosDist}(X, Y) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} (x_i)^2} \cdot \sqrt{\sum_{i=1}^{n} (y_i)^2}} \]  

(3.2)

Figure 3.4: Structure Similarity Example

3.2.4.2 Structure Similarity

As we have seen in Figure 3.1, content similarity can be a strong indication that two blocks are part of a template. However, this is not the only measure that we can apply on HTML documents. In many cases, the content of two blocks may not be similar.
even if their general layout is very close. Figure 3.4 shows two blocks which have different content but similar layout. These blocks have been extracted from two different pages from the CNN\textsuperscript{11} website.

Apparently, the two blocks shown in Figure 3.4 (a) and (b) are part of the CNN template. Thus, we need another measure that exploits the layout of these blocks rather than their content, i.e., block structure. It is normal to find similar cases in other websites where blocks are the same in terms of the layout and different in their content. The layout of a block is defined by its HTML DOM tree which was introduced in Sections 2.3.1 and 2.3.2. Therefore, we need to compare the tree structures against each other to find the similarity between their layouts.

\begin{figure}[h]
\centering
\begin{tabular}{c|c}
\hline
(a) & (b) \\
\hline
\end{tabular}
\caption{Similarity between HTML DOM Trees}
\end{figure}

\textsuperscript{11} www.cnn.com
Consider Figures 3.5.a and 3.5.b which are the HTML DOM trees for the blocks illustrated in Figures 3.4.a and 3.4.b, respectively. Clearly, the two trees are identical; this is a high indication that the corresponding two blocks are part of the template. Therefore, we can determine identical blocks regardless of their content. The next sections introduce the structure measure we use and include case studies which show content and structure similarity matrices.

3.2.4.2.1 Simple Tree Matching Algorithm

The structure similarity problem is essentially a tree matching problem. This is particularly true because we intend to find the maximum number of matching nodes between the two trees. The tree edit distance was proposed to match two trees, A and B, which are labeled ordered rooted trees (Zhang, 1989). Tree edit distance is basically the cost associated with the minimum set of operations needed to transform A into B. The classic formulation of the edit tree algorithm proposed three different mapping operations: node removal, node insertion, and node replacement. A cost is assigned to each operation, and the tree edit distance is the minimum mapping cost of the two trees. Mapping of trees can be formulated as follows (Yang, 1991):

Let $X$ be a tree, and assume $X[i]$ is the $i$th node of tree $X$ in a preorder traversal of the tree. A mapping $M$ between a tree $A$ of size $n_1$ and a tree $B$ of size $n_2$ is a set of ordered pairs $(i, j)$, one from each tree, satisfying the following conditions for all $(i_1, j_1), (i_2, j_2) \in M$:

1. $i_1 = i_2 \iff j_1 = j_2$;
2. $A[i_1]$ is on the left of $A[i_2] \iff B[j_1]$ is on the left of $B[j_2]$;

Intuitively, the definition requires that each node appears no more than once in a mapping and the order among siblings and the hierarchical relation amongst nodes are preserved.

Other algorithms have been proposed to find the minimum set of operations to transform a tree into another. All formulations have complexities higher than quadratic. (Tai, 1979) proposed a solution based on dynamic programming with complexity of $O(n_1n_2h_1h_2)$, where $n_1$ and $n_2$ are the sizes of the trees and $h_1$ and $h_2$ are the heights of the trees. (Wang et al., 1998; Chen W., 2001) are two improved algorithms.

![General Tree Mapping Example](image)

**Figure 3.6: General Tree Mapping Example**

Figure 3.6 shows a general tree mapping example in which we can cross levels. For instance *title* in $A$ can be matched to *title* in $B$. Replacements are allowed as well, e.g., *body* in $A$ and *strong* in $B$. A more restricted version is being used in our system, i.e., Simple Tree Matching (STM). The main goal of STM is to find number of pairs in a
maximum matching in the two trees without doing any replacements or level crossing (Yang, 1991). The more restricted version, namely STM, is used because its time complexity is less than that of Tai (1979). Yet, it has been demonstrated that STM is capable of producing satisfactory results in information extraction techniques (Liu, 2006).

Formally, STM defines matching between two trees to be a set of pairs of nodes having one node from each tree such that:

1. Two nodes in a pair contain identical symbols
2. A node in the first tree can match at most one node in the other tree
3. The parent-child relationship as well as the order between sibling nodes should be preserved, hence, no level crossing is allowed.

Although it is a restrictive formulation of the general matching algorithm, it was found to be quite effective in Web data extraction (Liu & Chen-Chuan-Chang, 2004).

**Lemma 3.1** Let A and B be two trees. A mapping between the two trees A and B must comply with the above three points, and is defined as the number of pairs of maximum matching between the two trees, according to (Yang, 1991).

Assume \( A = R_A : (A_1, \ldots, A_k) \) and \( B = R_B : (B_1, \ldots, B_n) \) are the two trees, where \( R_A \) is the root of tree A and \( R_B \) is the root of tree B, and \( A_i \) and \( B_j \) are the \( i \)th and \( j \)th first-level subtrees of A and B, respectively. Then, \( W[i,j] \) denotes the number of pairs in a maximum matching of the subtrees \( A_i \) and \( B_j \). So, if \( R_B \) and \( R_A \) have identical symbols, i.e., similar nodes, then the maximum matching between A and B, i.e. \( W[A,B] \), is \( M((A_1, \ldots, A_k), (B_1, \ldots, B_n)) + 1 \), where \( M((A_1, \ldots, A_k), (B_1, \ldots, B_n)) \) is the number of pairs in
the maximum matching of \( \langle A_1, \ldots, A_k \rangle \) and \( \langle B_1, \ldots, B_n \rangle \). But, if \( R_A \) and \( R_B \) have distinct symbols, then \( W[A,B] = 0 \). The STM algorithm is shown in Figure 3.7.

As Figure 3.7 shows, Simple Tree Matching is a top down algorithm which uses dynamic programming to produce the maximum matching between two trees. The time complexity of it is \( O(n_1n_2) \), where \( n_1 \) and \( n_2 \) are the sizes of trees \( A \) and \( B \) respectively.

\[ \text{Algorithm: Simple-Tree-Matching}(A, B) \]
1. \textbf{if} (roots of the two trees \( A \) and \( B \) contain distinct symbols) 
2. \textbf{return} 0 
3. \textbf{m} := the number of first-level subtrees of \( A \). 
4. \textbf{n} := the number of first-level subtrees of \( B \). 
5. \textbf{Initialization:} \( M[i, 0] := 0 \) for \( i = 0, \ldots, m \) \( M[0,j] := 0 \) for \( j = 0, \ldots, n \) 
6. \textbf{for} \( i := 1 \) to \( m \) do 
7. \textbf{for} \( j := 1 \) to \( n \) do 
8. \( M[i,j] := \max(M[i-1,j], M[i-1,j-1] + W[i,j]) \) 
   //where \( W[i,j] = \text{Simple-Tree-Matching}(A_i, B_j) \), and 
   //\( A_i \) and \( B_j \) are the \( i \)th and \( j \)th first-level subtrees of \( A \) and \( B \) 
   //respectively. 
9. \textbf{return}(M[m, n] + 1) 

Figure 3.7: Simple Tree Matching Algorithm (STM) (Yang, 1991)
Figure 3.8: STM Example

Figure 3.8 is used as an example to illustrate the STM algorithm. The root nodes of $A$ and $B$, namely $N1$ and $N15$, respectively, are compared first (line 1). Since they do not contain identical symbols, the STM algorithm will recursively find the number of pairs in a maximum matching between the first-level subtrees of $A$ and $B$, i.e., the $W$ matrix. However, in case the root nodes did not contain identical symbols, the STM algorithm will return 0 as a result.

The first row and first column of $W$ are initialized to zeros (line 5). In line 8, the algorithm finds and inserts into matrix $W$ the maximum matching between the first-level subtrees of $A$ and the first-level subtrees of $B$. Dynamic programming is applied to find the number of pairs in a maximum matching between $A$ and $B$. Consequently, STM is applied on the subtrees rooted at $N2$ and $N16$ resulting with three matching pairs: $\{N2, N16\}$, $\{N6, N18\}$, and $\{N7, N19\}$. When matching $N2$ with $N17$ (the next node in the first level of $N15$), the returned value is 0 since the root nodes do not match. Iteratively,
N3 is matched against N16 yielding 0 and N3 against N17 yielding 2. N4 is matched against N16 and N17 yielding with 2 and 0, respectively. Lastly, N5 is matched against N16 and N17 yielding 0 and 3, respectively. As a result, the number of pairs of maximum matching is 7. Here, it is worth noting that line 9 in Figure 3.8 returns the content of $M[m,n]$, which is 6 in our example, and adds 1 for the root nodes. Figure 3.9.a shows the $M$ matrix and Figure 3.9.b shows the $W$ matrix of the first-level subtrees.

<table>
<thead>
<tr>
<th></th>
<th>N16</th>
<th>N17</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>N3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>N4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>N5</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1 (N16)</th>
<th>2(N16-N17)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1 (N2)</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2 (N2-N3)</td>
<td>0</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3 (n2-N4)</td>
<td>0</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4(N2-N5)</td>
<td>0</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

(b)

Figure 3.9: (a) W Matrix of First-Level Subtrees; (b) M Matrix of First-Level Subtrees

The shaded cell in Figure 3.9.b shows the number of maximum matchings between the node sets: \{N16, N17\} and \{N2, N3, N4, N5\}, which is $M[4,2]$, where 4 and 2 are the numbers of nodes in the level of A and B, respectively.

In our approach, we need a normalized similarity value in the interval [0,1] to represent the similarity between two trees. Therefore, we adapt STM to return a normalized value, which is the average value of the number of maximum matching pairs.
with respect to the size of the smaller tree. So, in line 9 we return the following value instead:

\[
\text{NormalizedMatchingScore} = \frac{M[m,n] + 1}{\min(m,n)},
\]

(3.3)

where \(m\) and \(n\) are the numbers of nodes in the 1\textsuperscript{st} level of the two trees \(A\) and \(B\), and \(\min(m,n)\) is a function that returns the minimum value of \(m\) and \(n\). In Figure 3.8, \(A\) and \(B\) have structure similarity (NormalizedMatchingScore) of 0.875 (7/8).

3.2.4.3 Case Study

In this section, we will study the effectiveness of using content and structure as the noise measures.

Given two pages \(p1\) and \(p2\), each consists of 8 blocks and let Table 3.1 be the content similarity matrix \(S_{nxm}\) (see Definition 3.1) for the blocks. For simplicity, we picked an example where \(m = n\). Later in the next sections we will show an example where \(m \neq n\).

<table>
<thead>
<tr>
<th>(P2)</th>
<th>(P1)</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
<th>b7</th>
<th>b8</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>1</td>
<td>0.119681</td>
<td>0</td>
<td>0.230464</td>
<td>0</td>
<td>0.236035</td>
<td>0.171477</td>
<td>0.262613</td>
<td></td>
</tr>
<tr>
<td>a2</td>
<td>0.113644</td>
<td>0.249805</td>
<td>0</td>
<td>0.165002</td>
<td>0.019684</td>
<td>0.058342</td>
<td>0.099385</td>
<td>0.032456</td>
<td></td>
</tr>
<tr>
<td>a3</td>
<td>0</td>
<td>0</td>
<td>0.043519</td>
<td>0</td>
<td>0</td>
<td>0.008025</td>
<td>0</td>
<td>0.125</td>
<td></td>
</tr>
<tr>
<td>a4</td>
<td>0.072556</td>
<td>0.069128</td>
<td>0</td>
<td>0.079673</td>
<td>0.014785</td>
<td>0.065733</td>
<td>\textbf{0.097704}</td>
<td>0.028441</td>
<td></td>
</tr>
<tr>
<td>a5</td>
<td>0</td>
<td>0.257844</td>
<td>0</td>
<td>0.238141</td>
<td>\textbf{0.421117}</td>
<td>0.014859</td>
<td>0</td>
<td>0.038576</td>
<td></td>
</tr>
<tr>
<td>a6</td>
<td>0.231775</td>
<td>0.047883</td>
<td>0.024692</td>
<td>0.12877</td>
<td>0.16488</td>
<td>\textbf{0.988024}</td>
<td>0.10007</td>
<td>0.078801</td>
<td></td>
</tr>
<tr>
<td>a7</td>
<td>0.171477</td>
<td>0.017102</td>
<td>0</td>
<td>0.345452</td>
<td>0.010242</td>
<td>0.09741</td>
<td>0.036589</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>a8</td>
<td>0.262613</td>
<td>0.037978</td>
<td>0.130558</td>
<td>0.052957</td>
<td>0.056857</td>
<td>0.072225</td>
<td>0.036589</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1: Content Similarity Matrix
The values in the 1st row represent the similarity between the 1st block in \( p_1 \) (\( p_1(a_1) \)) and all the other blocks in \( p_2 \) (\( b_{1,8} \)). In general, \( S[i,j] \) is the similarity value between \( a_i \) and \( b_j \). According to Definition 1.2, we need to find blocks with high similarity values, which will indicate that these blocks are part of the site’s template. The cells with the highest similarity for each \( a_i \) are shaded. The following is the set of matching pairs:

\[ \{a_1,b_1\}, \{a_2,b_2\}, \{a_3,b_3\}, \{a_4,b_7\}, \{a_5,b_5\}, \{a_6,b_6\}, \{a_7,b_7\}, \{a_8,b_8\} \]

There are two observations regarding this set:

1. The matching block of \( a_i \) is \( b_i \) \( \forall i \in [1,m], i \notin \{3,4\} \).

2. The values of \( S[1,1], S[6,6], S[7,7], \) and \( S[8,8] \), are noticeably high. On the other hand, \( S[2,2], S[3,8], S[4,7], \) and \( S[5,5] \) have relatively smaller values.

The first point assures that VIPS segments a web page spatially consistently. In fact, VIPS utilizes the spatial information of a block, and so it segments multiple pages in the same fashion in terms of the blocks’ location. Taking this fact into consideration, a block number, e.g., 1, 2, etc, provides not only a distinguishing number, but also spatial implications. Thus, block \( a_1 \) is the closest block to the upper-left corner of \( p_1 \), so is \( b_1 \) with respect to \( p_2 \). This causes \( a_1 \) to be the most similar to \( b_1 \) and \( a_2 \) to \( b_2 \) and so on, which, as a result, will cause the values along the diagonal of the similarity matrix high compared to the other values.

The second observation strongly indicates that the four pairs: \( \{a_1,b_1\}, \{a_6,b_6\}, \{a_7,b_7\}, \) and \( \{a_8,b_8\} \) are part of the site’s template. For the other blocks, namely 2, 3, 4, and 5, we need to further check their structure to decide whether they are part of the
template or not. One last observation is that $a_i$'s highest similarity value is very small compared to the others: 0.097704.

**Table 3.2: Structure Similarity Matrix**

<table>
<thead>
<tr>
<th></th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
<th>b7</th>
<th>b8</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>1</td>
<td>0.070423</td>
<td>0.068182</td>
<td>0.062992</td>
<td>0.133333</td>
<td>0.075377</td>
<td>0.027833</td>
<td>0.12766</td>
</tr>
<tr>
<td>a2</td>
<td>0.06993</td>
<td>0.992481</td>
<td>0.126582</td>
<td>0.067797</td>
<td>0.144144</td>
<td>0.030848</td>
<td>0.024291</td>
<td>0.075758</td>
</tr>
<tr>
<td>a3</td>
<td>0.06818</td>
<td>0.128205</td>
<td>1</td>
<td>0.190476</td>
<td>0.142857</td>
<td>0.041916</td>
<td>0.013667</td>
<td>0.077922</td>
</tr>
<tr>
<td>a4</td>
<td>0.07272</td>
<td>0.08</td>
<td>0.26087</td>
<td>0.447059</td>
<td>0.128205</td>
<td>0.039326</td>
<td>0.017354</td>
<td>0.080808</td>
</tr>
<tr>
<td>a5</td>
<td>0.13953</td>
<td>0.134454</td>
<td>0.123077</td>
<td>0.076923</td>
<td>0.453608</td>
<td>0.074667</td>
<td>0.020833</td>
<td>0.152542</td>
</tr>
<tr>
<td>a6</td>
<td>0.00952</td>
<td>0.009756</td>
<td>0.011236</td>
<td>0.010127</td>
<td>0.118557</td>
<td>0.006006</td>
<td>0.005188</td>
<td>0.00978</td>
</tr>
<tr>
<td>a7</td>
<td>0.02783</td>
<td>0.024341</td>
<td>0.013667</td>
<td>0.016736</td>
<td>0.02131</td>
<td>0.013351</td>
<td>1</td>
<td>0.020325</td>
</tr>
<tr>
<td>a8</td>
<td>0.12766</td>
<td>0.076336</td>
<td>0.077922</td>
<td>0.068966</td>
<td>0.165138</td>
<td>0.062016</td>
<td>0.020325</td>
<td>1</td>
</tr>
</tbody>
</table>

Structure similarity is another important feature that we can use with HTML documents. Table 3.2 is the structure similarity matrix between blocks of the two pages $p1$ and $p2$, which is computed using the Normalized Simple Matching Tree algorithm. The shaded cells in Table 3.2 show the highest similarity value for each corresponding blocks. Explicitly, the following is the set of matching pairs according to Table 3.2:

$$\{a_1, b_1\}, \{a_2, b_2\}, \{a_3, b_3\}, \{a_4, b_4\}, \{a_5, b_5\}, \{a_6, b_6\}, \{a_7, b_7\}, \{a_8, b_8\}$$

Here, three observations can be made:

1. Block $a_i$ matches block $b_i$, $\forall i \in [1, 8], i \notin \{6\}$.

2. The pairs: $\{a_1, b_1\}, \{a_2, b_2\}, \{a_3, b_3\}, \{a_7, b_7\}$, and $\{a_8, b_8\}$ have noticeably high similarity values.
3. Structure similarity values tend to be high.

The first observation emphasizes the fact that when \( m=n \) values along the diagonal of the similarity matrix tend to be the correct values for each block in both \( p1 \) and \( p2 \). This is true in Table 3.2 for all blocks of \( p1 \) except \( a6 \). The second observation shows that \( \{a1, b1, a2, b2, a3, b3, a7, b7, a8, b8\} \) are highly likely to be part of the template because of the high consistency in there structure within the two pages, \( p1 \) and \( p2 \). The third observation shows that the average of the structure similarity values is higher than the average of the content similarity values. The average of structure similarity values is 0.75, while it is 0.61 for content. We may conclude then that the analyzed site, CNN in this case, has a highly consistent layout throughout its pages.

3.2.5 Block Matching

The example in Section 3.2.4.3 considers that the two input pages have the same number of blocks. Also, we noted that we do not need to compare block \( ai \) with all blocks of \( p2 \). Rather, we need to compare \( ai \) only with \( bi \), where \( i \in [1, m] \). Thus, the noisiness of \( ai \) is the similarity between \( ai \) and \( bi \), which means that \( ai \) matches \( bi \). This assumption reduces the complexity of our approach since there is no need to do \( m^2 \) comparisons, but rather we need to do only \( m \) comparisons.

Unfortunately, this is not always true in practice; where we could have different number of blocks for pages even if they belong to the same site. In other words, pages that belong to the same site share the general layout (template), but some minor visual perturbations may occur. For example, these visual perturbations may cause VIPS to
segment a page into 10 blocks, while another page from the same site may be segmented into 8 blocks. This means that the page with the bigger number of blocks will have surplus of 2 blocks that will not have matching blocks from the other page.

**DEFINITION 3.2 [Neighboring Blocks]:** Let $p_1$ and $p_2$ be two pages from the same website, i.e. $p_1$ consists of the blocks $a_1, a_2, ..., a_m$ and $p_2$ consists of the blocks $b_1, b_2, ..., b_n$ with $m \leq n$, then the neighboring blocks of $a_i$ are $b_j \forall j \in [x, y]$ where $j \in Z, x = |(i - (n - m))|$ and $y = (i + (n - m))$. If $x = 0$ then set $x = 1$.

Notice that according to Definition 3.2 $j = i$ when $m = n$ is satisfied. In general, one of the following two scenarios may arise after segmenting two pages $p_1$ and $p_2$:

1. $m = n$
   In this case, each block in $p_1$ will have a matching block from $p_2$.

   As illustrated in Figure 3.10, the relationship between the blocks is a one-to-one relationship when $m = n$. Also, the $i$th block in $p_1$ matches the $i$th block in $p_2$ since VIPS segments a page into blocks by considering their spatial information. Chapter 4 gives more details about the number of blocks for pages of different websites that we have tested.

![Figure 3.10: Block Matching when $m = n$](image-url)
2. $m \neq n$

In this case there are blocks that do not have matching blocks from the other page.

Clearly, the number of blocks that do not have matching blocks is $|m - n|$. Figure 3.11 shows an example of two pages with 5 and 7 blocks, respectively.

Since one block in $p_1$ can match at most one block in $p_2$; there will be two blocks in $p_2$ without matching blocks from $p_1$. In Figure 3.11, $b_2$ and Block 5 in $p_2$ do not have matching blocks from $p_1$. Therefore, we will need to have a measure other than content and structure that could help in sorting out the matching; the third measure used by Noise Detector is presentation. Before we introduce the presentational noise measure, we present the MatchBlocks algorithm in Figure 3.12.
\begin{figure}
\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Input:} $S_{N \times M}$: $N \times M$ similarity matrix that has similarity values for blocks from $p1$ against $p2$ \\
\hline
\textbf{Output:} $\text{MatchingBlocks}$: an array that holds the $i$th entry the number of the matching block from $p2$ \\
\hline
\textbf{Algorithm:} $\text{MatchBlocks}(S)$ \hfill \\
1. Initialize the $i$th entry of $\text{MatchingBlocks}$ with the value $i$, $\forall i \in [1, N]$ \\
2. \textbf{for} all blocks in $p1$ \\
3. \hspace{1em} $\text{max} = S[i, i]$ \\
4. \textbf{for} all neighboring blocks of the $i$th block from $p2$ \\
5. \hspace{1em} \textbf{if} ($S[i, j] > \text{max}$) \\
6. \hspace{2em} $\text{MatchingBlocks}[i] = j$ \\
7. \hspace{1em} $\text{max} = S[i, j]$ \\
8. \textbf{Return} $\text{MatchingBlocks}$ \\
\hline
\end{tabular}
\end{center}
\caption{MatchBlocks Algorithm}
\end{figure}

For each block in $p1$, MatchBlocks looks for the most similar block from the neighboring blocks in $p2$. Note that the input to the MatchBlocks algorithm is a similarity matrix $S$ that combines content and structure similarity. Equation 3.5 explains how the combined matrix is computed. So, the MatchBlocks does not guarantee that the pairs of blocks match; it just pairs the most similar blocks with each other. Chapter 4 shows the accuracy results of this approach.

The flowchart shown in Figure 3.13 shows the details of our algorithm after the two pages are segmented using VIPS. If $p1$ and $p2$ have the same number of blocks, then content and structure similarities are computed. Most importantly, there is no need to compare all blocks from the two pages against each other; rather we compare the $i$th
block in \( p1 \) against the \( i \)th block in \( p2 \). The mapping depicted in Figure 3.10 is followed in this case.

**Figure 3.13: Detailed Flowchart of Noise Detector**
If \( p1 \) and \( p2 \) do not have the same number of blocks, i.e. \( p1 \) has fewer blocks; we find the content and structure similarities between each block in \( p1 \) and its corresponding set of neighboring blocks in \( p2 \). Another naïve approach, yet time-consuming, is to compute the whole similarity matrix \( S \) as described in Definition 3.1. Calculating the similarity between a block and only its neighboring blocks saves computation time. For example, block \( b1 \) in \( p1 \) (\( p1(b1) \)) has three neighboring blocks in \( p2 \), namely \( b1, b2, \) and \( b3 \). After calculating the similarity between \( p1(b1) \) and each of \( p2(b1), p2(b2), \) and \( p2(b3) \), the MatchBlocks algorithm is applied to find the actual matching block.

It is possible to have a block equivalent to a combination of more than one block. In some cases, two blocks in one page are equivalent to one block in another page. For example, \( b5 \) of \( p1 \) in Figure 3.11 may match with the combination of \( b6 \) and \( b7 \) in \( p2 \). This all depends on the similarity values of the neighboring blocks. In fact, Figure 4.1 and Figure 4.2 illustrate this case.

For blocks that have corresponding matching blocks, content and structure similarities are sufficient to find out if they are part of the template or not. But, blocks from \( p1 \) that do not have matching blocks in \( p2 \) can not use content and structure similarities because there are no blocks to compare them to; presentational noise measure is used instead. To illustrate the process further, we will discuss an example in which there are two pages \( p1 \) and \( p2 \) from the CNN website with 7 blocks for \( p1 \) and 9 blocks for \( p2 \). Tables 3.3, 3.4, and 4.5 are the similarity matrices for these two pages.
Table 3.3: Content Similarity Matrix (ContSim)

<table>
<thead>
<tr>
<th>a1</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
<th>b7</th>
<th>b8</th>
<th>b9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1562</td>
<td>0</td>
<td>0.0370</td>
<td>0</td>
<td>0.2317</td>
<td>0.1714</td>
<td>0.0189</td>
<td>0.2626</td>
<td></td>
</tr>
<tr>
<td>a2</td>
<td>0.1660</td>
<td>0.3590</td>
<td>0</td>
<td>0.0260</td>
<td>0.2683</td>
<td>0.0398</td>
<td>0.0142</td>
<td>0</td>
<td>0.0395</td>
</tr>
<tr>
<td>a3</td>
<td>0.0469</td>
<td>0.0901</td>
<td>0.0396</td>
<td>0.1147</td>
<td>0.1653</td>
<td>0.0281</td>
<td>0.0831</td>
<td>0.0171</td>
<td>0.0198</td>
</tr>
<tr>
<td>a4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.0160</td>
<td>0.4211</td>
<td>0.164</td>
<td>0.0102</td>
<td>0.0984</td>
<td>0.0568</td>
</tr>
<tr>
<td>a5</td>
<td>0.2365</td>
<td>0.1200</td>
<td>0.0160</td>
<td>0.0385</td>
<td>0.0148</td>
<td>0.9766</td>
<td>0.0999</td>
<td>0.0208</td>
<td>0.0723</td>
</tr>
<tr>
<td>a6</td>
<td>0.1714</td>
<td>0.0219</td>
<td>0</td>
<td>0.0320</td>
<td>0</td>
<td>1.0000</td>
<td>0</td>
<td>0</td>
<td>0.0365</td>
</tr>
<tr>
<td>a7</td>
<td>0.2488</td>
<td>0.0491</td>
<td>0</td>
<td>0.0373</td>
<td>0.0350</td>
<td>0.0752</td>
<td>0.0332</td>
<td>0.2297</td>
<td>0.9097</td>
</tr>
</tbody>
</table>

Table 3.4: Structure Similarity Matrix (StructSim)

<table>
<thead>
<tr>
<th>a1</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
<th>b7</th>
<th>b8</th>
<th>b9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.07142</td>
<td>0.08421</td>
<td>0.03669</td>
<td>0.13953</td>
<td>0.00952</td>
<td>0.02783</td>
<td>0.07272</td>
<td>0.12766</td>
<td></td>
</tr>
<tr>
<td>a2</td>
<td>0.0714</td>
<td>1</td>
<td>0.12048</td>
<td>0.04123</td>
<td>0.13675</td>
<td>0.00980</td>
<td>0.02444</td>
<td>0.12244</td>
<td>0.07751</td>
</tr>
<tr>
<td>a3</td>
<td>0.0583</td>
<td>0.08</td>
<td>0.475</td>
<td>0.29787</td>
<td>0.10526</td>
<td>0.00987</td>
<td>0.01639</td>
<td>0.14736</td>
<td>0.06349</td>
</tr>
<tr>
<td>a4</td>
<td>0.1333</td>
<td>0.14814</td>
<td>0.15873</td>
<td>0.05194</td>
<td>0.45360</td>
<td>0.11855</td>
<td>0.02123</td>
<td>0.12820</td>
<td>0.16513</td>
</tr>
<tr>
<td>a5</td>
<td>0.0753</td>
<td>0.03108</td>
<td>0.06451</td>
<td>0.01126</td>
<td>0.07466</td>
<td>0.00600</td>
<td>0.01335</td>
<td>0.04494</td>
<td>0.06201</td>
</tr>
<tr>
<td>a6</td>
<td>0.0278</td>
<td>0.02444</td>
<td>0.01793</td>
<td>0.00869</td>
<td>0.02083</td>
<td>0.00518</td>
<td>1</td>
<td>0.01735</td>
<td>0.02032</td>
</tr>
<tr>
<td>a7</td>
<td>0.1118</td>
<td>0.08053</td>
<td>0.19230</td>
<td>0.03389</td>
<td>0.17391</td>
<td>0.00932</td>
<td>0.01953</td>
<td>0.35294</td>
<td>0.86666</td>
</tr>
</tbody>
</table>

Table 3.5: Combined Similarity Matrix (CombSim)

<table>
<thead>
<tr>
<th>a1</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
<th>b7</th>
<th>b8</th>
<th>b9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.11381</td>
<td>0.04210</td>
<td>0.03685</td>
<td>0.06976</td>
<td>0.12065</td>
<td>0.09965</td>
<td>0.04584</td>
<td>0.19513</td>
<td></td>
</tr>
<tr>
<td>a2</td>
<td>0.11876</td>
<td>0.67954</td>
<td>0.06024</td>
<td>0.03361</td>
<td>0.20256</td>
<td>0.02483</td>
<td>0.01934</td>
<td>0.06122</td>
<td>0.05852</td>
</tr>
<tr>
<td>a3</td>
<td>0.052655</td>
<td>0.08508</td>
<td>0.25735</td>
<td>0.20629</td>
<td>0.13532</td>
<td>0.01901</td>
<td>0.04976</td>
<td>0.08227</td>
<td>0.04167</td>
</tr>
<tr>
<td>a4</td>
<td>0.066667</td>
<td>0.07407</td>
<td>0.07936</td>
<td>0.03398</td>
<td>0.43736</td>
<td>0.14171</td>
<td>0.01573</td>
<td>0.11334</td>
<td>0.11099</td>
</tr>
<tr>
<td>a5</td>
<td>0.15598</td>
<td>0.07556</td>
<td>0.04030</td>
<td>0.02490</td>
<td>0.04478</td>
<td>0.49131</td>
<td>0.05666</td>
<td>0.03292</td>
<td>0.06720</td>
</tr>
<tr>
<td>a6</td>
<td>0.099655</td>
<td>0.02317</td>
<td>0.00896</td>
<td>0.02061</td>
<td>0.01041</td>
<td>0.05262</td>
<td>1</td>
<td>0.00867</td>
<td>0.02845</td>
</tr>
<tr>
<td>a7</td>
<td>0.18033</td>
<td>0.06486</td>
<td>0.09615</td>
<td>0.03564</td>
<td>0.10450</td>
<td>0.04229</td>
<td>0.02640</td>
<td>0.29136</td>
<td>0.88819</td>
</tr>
</tbody>
</table>
Disney actor booted from 'Dancing'
- Story Highlights
  - "Dancing with the Stars" actor Cody Laney voted off "Dancing with the Stars"
  - Laney, along with Julianne Hough, received lowest scores from judges.
  - Brandon Banks, Lance Bass, Warren Sapp & partner in next week's finale.
- Next Article in Entertainment

'Twilight' debuts in No. 1 slot at box office
- Story Highlights
  - "Twilight" beat the previous record for biggest opening ever for a female director.
  - Summit Entertainment announced it will produce a third "New Moon".
  - Robert Pattinson and Kristen Stewart will return as their crossed lovers.
- Next Article in Entertainment

We Recommend
Stories you may be interested in based on past browsing

CNN on the Go
- Arabic
- Japanese
- Korean
- Turkish
International Edition
- CNN UK
- CNN International
- Headlines News
- Transcripts
- 2008 Cable News Network
- Turner Broadcasting System, Inc. All Rights Reserved.
Terms of Service
Privacy Guidelines
Advertise with us
About us
Contact us
help

Figure 3.14: Visual Layout of the Matching Block Pairs
The shaded cells contain the maximum similarity value in their specific rows. Using Table 3.3, the MatchBlocks algorithm will match the following pairs:

\{a_1,b_1\}, \{a_2,b_2\}, \{a_3,b_3\}, \{a_4,b_5\}, \{a_5,b_6\}, \{a_6,b_7\}, \{a_7,b_9\}

\(b_3, b_4,\) and \(b_8\) do not have matching blocks, and hence presentational noise is used to judge their noisiness. According to Table 3.3, \(a_3\) matches \(b_5\). However, after manually investigating \(a_3\) and \(b_5\), this is considered to be false because they are not equivalent. In fact, \(a_3\) should match \(b_3\).

We can also notice that the values in the two columns \(b_3\) and \(b_4\) are small compared to the other columns. This indicates that \(b_3\) and \(b_4\) are most likely to be informative blocks and not part of the template.

If we use structure similarity alone, the following set includes the result matching pairs: \(\{a_1,b_1\}, \{a_2,b_2\}, \{a_3,b_3\}, \{a_4,b_5\}, \{a_5,b_5\}, \{a_6,b_7\}, \{a_7,b_9\}\). Notice that \(b_4, b_6,\) and \(b_8\) do not match other blocks in \(pI\).

By looking at the two result sets, we notice that \(b_3\) and \(b_6\) do occur in neither solution. Table 3.5 shows the combined similarity matrix using both content and structure. This combined similarity matrix is the one used by MatchBlocks algorithm. The next figures show that combining content and structure gives better results.

The values in Table 3.5 are calculated as follows:

\[
CombSim[i,j] = CW \times ContSim[i,j] + SW \times StructSim[i,j],
\]

where \(CW\) is the content weight and \(SW\) is the structure weight. \(CW = 0.5, SW = 0.5\) (equal weight is assigned to each of the two similarities; other scenarios are also possible). Later in the algorithm when deciding on the valid blocks, different weights are dynamically calculated and used. The following set of matching pairs can be concluded.
from Table 3.5: \( \{a_1, b_1\}, \{a_2, b_2\}, \{a_3, b_3\}, \{a_4, b_4\}, \{a_5, b_5\}, \{a_6, b_6\}, \{a_7, b_7\} \). This set of pairs is different from the two previous sets generated when using content or structure separately.

Figure 3.14 shows three of the matching pairs of this example, as identified by the Noise Detector. In Figure 3.14, \( a_1 \) matches \( b_1 \), \( a_2 \) matches \( b_2 \), and \( a_3 \) matches \( b_3 \). These blocks are highly similar in their structure and even in their content. Figure 3.15 shows the two unmatched blocks in \( p_2 \). We have not depicted all blocks into Figure 3.14; though the blocks in Figure 3.15 definitely do not match other blocks in \( p_1 \).

![Figure 3.15: Unmatched blocks from \( p_2 \)](image)

### 3.2.5.1 Presentational Noise Measure

As some blocks do not have matching blocks from the other page, content and structure can not be used to judge their noisiness. Therefore, we need to use a different
measure to identify the noisiness of a block. Presentational features are used with the unmatched blocks.

By analyzing the Web, we can conclude that some presentational features add noisiness to blocks. Furthermore, the more presentational features are present in a block, the more likely the block is said to be noisy, and therefore the block is identified as not informative. Figure 3.16 supports our claim that informative blocks usually have less relative presentation than noisy blocks.

Notice that using presentational features enables us to discover the likely noisy blocks without necessarily being part of the template. In fact, these blocks are not likely to be part of the template because in general they did not have matching blocks.

Figure 3.16: Examples on Presentational Noisy Blocks from BBC\textsuperscript{12}

\textsuperscript{12} http://www.bbc.co.uk/
The presentational features include links, forms, input tags and some other tags, e.g., `<A>`, `<INPUT>`, `<FORM>`, `<LINK>`, etc. For each feature in the feature set, we calculate its relative content compared to the whole content of the unmatched block. So, if all of the content of a block is links, then the block is most likely to be a noisy block even if it is not part of the template. For instance, the two red-framed blocks in Figure 3.16 are likely to be noisy blocks because they contain high relative presentational content. On the other hand, the blue-framed block will likely be identified as an informative one since it does not contain high relative presentational content. After calculating the relative weights of all presentational features, these values are summed to calculate the total presentational noise value. Figure 3.17 presents the PresentationalNoise algorithm.

**Input:** $T$: DOM tree of a block  
**Output:** $Noise$: presentational noise value of the input block  
**Algorithm:** PresentationalNoise($T$)  
1. for each presentational feature $f_i$ find its percentage in $T$
2. for each presentational feature $f_i$
   
   $Noise = Noise + \frac{percentage(f_i)}{BlockContentSize}$
3. return $Noise$

Figure 3.17: PresentationalNoise Algorithm
3.2.6 Final Noise Measure

As we have seen so far, the combined similarity matrix incorporates both the content and structure. Also, a combined similarity value in this matrix is calculated using Equation 3.5 which assigns equal weight of 0.5 for the content and structure similarities. These values will be used to identify which blocks belong to the site template and which do not.

In the final stage, we use different weights which are biased to the measure with high value as incorporated in the UpdateWeights algorithm shown in Figure 3.18. The values 0.3 and 0.7 in Figure 3.18 are chosen after conducting some initial experiments. These values gave good results for the final noise measure. Furthermore, the trend of the similarity values shows that it is not critical to choose these values as we just need to bias the final value to either value when this value is extreme.

<table>
<thead>
<tr>
<th>Input: CS: content similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS: structure similarity</td>
</tr>
<tr>
<td>Output: CW: content weight</td>
</tr>
<tr>
<td>SW: structure weight</td>
</tr>
<tr>
<td>Algorithm: UpdateWeights(CS, SS)</td>
</tr>
</tbody>
</table>

1. \(\text{if (CS}>\text{threshold)}\)
2. \(CW = 0.7\)
3. \(SW = 0.3\)
4. \(\text{else if (SS}>\text{threshold)}\)
5. \(SW = 0.7\)
6. \(CW = 0.3\)
7. \(\text{else}\)
8. \(SW = CW = 0.5\)

Figure 3.18: UpdateWeights Algorithm
The threshold that we used in our experiments is 0.9. Blocks that do not have matching blocks will use the presentational noise measure to identify their noisiness. This measure decides whether a block is informative or noisy. The values are calculated using the PresentationalNoise algorithm which returns a value in the range [0,1].

3.2.7 Noise Threshold Value

After calculating the final noise value of a block in both pages, we need to determine which blocks are valid (informative) and which blocks are not. A dynamically calculated threshold is used to determine the cut-off value which splits the range [0,1] into two regions: valid and invalid.

\[ \text{cut\_off} = \text{min} + \frac{\text{stdev}}{\text{mean}}, \]  

(3.6)

where \( \text{min} \), \( \text{stdev} \), and \( \text{mean} \) are the minimum value, the standard deviation, and the average of the noise values of a set of blocks from the analyzed page. This cut-off value proved to work well in our experiments as it exploits different characteristics of the set of noise values. The precision and recall results that we got are high for most of the websites that have been evaluated.

3.2.7.1 Further Valid Blocks Refinement

We consider a refinement procedure in our approach through which we include blocks that have been marked as invalid (have noise value greater than the cut-off) if they satisfy a certain criterion.
The best descriptive and representative sentence for a web page is its title. Therefore, if a block has a big chunk of this title, then this block must have information related to the page topic. In other words, if the number of words in the title of a web page is greater than a certain pre-specified value, say 10 words, and there is a block that has more than 90% of the title in its content, then the latter block in marked as valid even if its noise value is greater than the cut-off value.

For this refinement procedure to work well, it is important to choose a cut-off for the number of words that the web page title must satisfy. This is because some web pages might not have a web page-topic-specific title. The 95% is a high, but yet a good threshold. For instance, if the title of a page is “BBC NEWS | Americas | Obama seeks to woo Canadians”, and a block has both “BBC NEWS” and “Americas”, then this block does not necessarily have topic-related information. On the contrary, this block is part of the BBC\textsuperscript{13} template and including it as a valid informative block will be a big mistake. This is why the refinement procedure should be strict.

3.3 Closing Remarks
In this chapter, we have introduced the different components of the Noise Detector. The presented case studies demonstrated the different novel aspects of the Noise Detector. We have illustrated via several examples how the Noise Detector computes and uses the content and structures measures. Also, we have shown the two cases when the number of blocks is the same for the two pages, and when the number of blocks is different. The

\textsuperscript{13} http://www.bbc.co.uk/
presentational noise measure has been explained along with the different corresponding cases. The methodology introduced in this chapter will be evaluated in the next chapter.
Chapter Four: Experimental Analysis

In this Chapter, I evaluate the proposed Noise Detector against the other approaches described in the literature. The set of experiments that have been conducted can be divided into two main categories:

1. An evaluation of the effectiveness of the Noise Detector in detecting and removing noisy blocks.

2. A user study that evaluates the impact of our approach by using Noise Detector’s output with different mining systems.

The first point investigates the validity and accuracy of the proposed approach when applying Noise Detector on different websites. These websites have been used by the different researchers to evaluate their approaches described in Chapter 2. The second point evaluates the applicability of our proposed approach on various tools. We used a web summarization tool and an information retrieval system in our user study. We asked users to evaluate the results of an information retrieval system that uses raw HTML documents and another information retrieval system that uses clean documents. Also, we asked them to evaluate summaries that were generated from the raw web pages and the clean web pages. The results of the user study are reported in the following sections. Before jumping to report and discuss the results, we will briefly introduce the data sets that have been used in the experiments and the testing environment.

4.1 The Testing Environment

The Noise Detector has been implemented using the C# programming language. A graphical user interface has been designed to make it more interactive and user
friendly. We have used an API of VIPS to segment pages as part of the Noise Detector's detection process. All the experiments have been conducted on a laptop with the following specifications:

- Architecture: X86-based PC
- CPU: ~1862 MHz
- Main memory: 1GB
- Operating system: Microsoft Windows XP Professional

4.2 Data Sets

In the evaluation process, we have used different websites that were used by (Lin & Ho, 2002; Vieira et al., 2006; Li & Ezeife, 2006; Debnath et al., 2005; Yi et al., 2003). These are some of the existing approaches that have been already discussed in Chapter 2. The following are the websites that have been used in the evaluation process:

1. **BBC**\(^{14}\): news website
2. **CNN**\(^{15}\): news website
3. **eJazzNews**\(^{16}\): Jazz news resource
4. **PCMag**\(^{17}\): PC Magazine for software, computers, hardware, news, reviews, and opinions
5. **CNET**\(^{18}\): website that provides product reviews, prices, software downloads, and technology news

\(^{14}\)http://www.bbc.co.uk
\(^{15}\)http://www.cnn.com
\(^{16}\)http://www.ejazznews.com
\(^{17}\)http://www.pcmag.com/
\(^{18}\)http://www.cnet.com
6. J&R (JANDR)\textsuperscript{19}: retailer of electronics

7. MemoryExpress\textsuperscript{20}: retailer of computers, laptops, monitors, and other electronics

8. Wikipedia\textsuperscript{21}: online free encyclopedia

9. Mythica Encyclopedia\textsuperscript{22}: encyclopedia of mythology, folklore, and religion

4.3 Validating the Noise Detector

The main challenge for the Noise Detector is first consistency results and then accuracy results. In other words, as described in Chapter 3, the Noise Detector uses only two pages to detect templates. Therefore, its consistency should be first questioned because it does not use a large number of pages. Once its consistency is confirmed to be high, we can proceed in the evaluation process and test its accuracy in detecting templates of the investigated websites.

The remainder of this section investigates both the consistency and accuracy of the Noise Detector. We explain the measures and techniques used in the evaluation.

4.3.1 Consistency Evaluation

As stated above, we decided to start the evaluation process by checking the consistency of the proposed Noise Detector because it uses only two pages in the process,

\textsuperscript{19} http://www.jr.com
\textsuperscript{20} http://www.memoryexpress.com
\textsuperscript{21} http://www.wikipedia.org
\textsuperscript{22} http://www.pantheon.org
while other approaches described in the literature, e.g., (Lin & Ho, 2002; Vieira et al., 2006; Li & Ezeife, 2006; Debnath et al., 2005; Yi et al., 2003) use hundreds of pages.

As discussed earlier in Chapter 3, comparing two pages, $p_1$ and $p_2$, the Noise Detector will produce the noise matrix $S_I$. The values in $S_I$ totally depend on the structure and content of both pages: $p_1$ and $p_2$. As a result, its cut-off value $c_1$ will depend on these noise values. This cut-off value identifies the valid blocks. However, comparing $p_1$ with $p_3$ will result in the noise matrix $S_2$, whose values depend solely on the content and structure of only $p_1$ and $p_3$. Its cut-off value $c_2$ will be most likely not equal to $c_1$, and will determine the valid blocks of $p_1$. As a result, two possible output valid blocks of $p_1$ are returned. In case the valid blocks of $p_1$ in both cases are different, the Noise Detector is said to be inconsistent, and therefore can not be trusted. So, our first step in the system evaluation process is to check its consistency. Detailed examples are shown in the next subsections for some of the websites in listed in Section 4.2. These examples give the reader a complete view about the different scenarios and cases, and then statistics of all the websites are included.

4.3.1.1 BBC

We downloaded around 100 pages from the BBC website. These pages were extracted from different sections: business, entertainment, health, and technology. Figure 4.1 and Figure 4.2 show two different pages from the business and technology sections of the BBC website. The page layout in both pages is highly similar, and therefore VIPS should be highly consistent in the segmentation process. Nonetheless, VIPS did not segment all pages into the same number of blocks as shown in Figures 4.1 and 4.2; these
two pages were segmented into five and six blocks, respectively. \textit{b1} and \textit{b2} in Figure 4.2 have been extracted as one block (\textit{b1}) in Figure 4.1; consequently Figure 4.1 has 1 block fewer than Figure 4.2. We report in Table 4.1 statistics of segmenting 100 pages from the BBC website.

In the segmentation process, we adjusted the \textit{permitted degree of coherence} (PDOC) of VIPS to 6. Most of the BBC pages are segmented into 6 or 7 valid pages as per Table 4.1. Though, it is worthwhile to mention that the number of blocks in these pages that we report in Table 4.1 refers to the number of blocks that pass through the valid-pass filter (initial valid blocks).

Figure 4.1: An Example Page from BBC Business Section
Table 4.1: Number of Blocks per Page for the BBC Website

<table>
<thead>
<tr>
<th>Number of Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
</tr>
<tr>
<td>Percentage</td>
</tr>
</tbody>
</table>

We picked three random pages to be cross-tested against a set of other pages from the BBC website. We tested each of the three pages against every page in the testing set, and recorded its valid block numbers. Table 4.2 depicts the results of cross checking these three pages.
Table 4.2: Occurrence Percentages of Valid Blocks of Different Pages from the BBC Website

<table>
<thead>
<tr>
<th></th>
<th>Occurrence Percentage of Valid Blocks</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B1</td>
<td>B2</td>
</tr>
<tr>
<td>p1</td>
<td>98%</td>
<td>100%</td>
</tr>
<tr>
<td>p2</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>p3</td>
<td>100%</td>
<td>64%</td>
</tr>
</tbody>
</table>

The percentages in Table 4.2 indicate the frequency of each block in p1, p2, and p3 being outputted as valid block. p2 consists of 7 blocks, while each of p1 and p3 consist of 6 blocks.

As the percentages are close to 91%, our approach is highly consistent in its output. Though, this does not mean that the output is necessarily correct. Section 4.3.2 will evaluate the accuracy of detecting the informative blocks.

4.3.1.2 CNN

A data set of around 250 pages was downloaded from the CNN website. These pages belong to five different categories: crime, entertainment, technology, travel, and world. The CNN website is highly consistent in its template design throughout its pages. Appendix B.6 illustrates a figure of a page from the CNN website along with its respective blocks produced by the segmentation.

Again, we have conducted the same consistency experiment on pages downloaded from the CNN website. Randomly selected four pages were cross tested against a test set
of around 50 pages. The valid output blocks were recorded in each test case, and the frequency of each block occurring in the output valid set is illustrated in Table 4.3.

Table 4.3: Occurrence Percentages of Valid Blocks of Different Pages from the CNN Website

<table>
<thead>
<tr>
<th></th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
<th>B8</th>
<th>B9</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>91.3%</td>
<td>95.7%</td>
<td>91.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>92.8%</td>
</tr>
<tr>
<td>p2</td>
<td>95.7%</td>
<td>97.9%</td>
<td>97.9%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>97.2%</td>
</tr>
<tr>
<td>p3</td>
<td>84.8%</td>
<td>97.8%</td>
<td>95.7%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>92.8%</td>
</tr>
<tr>
<td>p4</td>
<td>86.4%</td>
<td>100%</td>
<td></td>
<td>18.6%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>93.2%</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>93.96%</strong></td>
</tr>
</tbody>
</table>

\(p1, p2, \text{ and } p3\) were all segmented into 8 blocks, while \(p4\) was segmented into 9 blocks. The high percentages recorded in Table 4.3 indicate that the Noise Detector is consistent when operating on the CNN website. Blocks \{B2, B3, B4\} are consistently reported by the Noise Detector as the valid blocks in \(p1, p2, \text{ and } p3\). However, \(p4\) doesn’t follow the same trend; its valid blocks are B2 and B3. B6 is marked as a valid block in \(p4\) only 18.6%. In fact, outputting B6 as a valid block is false positive (i.e., it is actually not a valid block). We calculated the average using only the percentages of the frequent blocks B2 and B3.

The number of blocks a page is segmented to affects the consistency of the Noise Detector, to some extent. This was clear with \(p4\), since it consists of 9 blocks, while all
the other pages consist of 8 blocks each. Table 4.4 shows statistics about the number of blocks in the data set of the CNN pages that we used.

<table>
<thead>
<tr>
<th>Number of Blocks</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Percentage</td>
<td>10.0%</td>
<td>40.0%</td>
<td>46.7%</td>
<td>3.3%</td>
</tr>
</tbody>
</table>

4.3.1.3 J&R

J&R is an online retailer for electronics. Around 150 pages were downloaded for testing. These pages were extracted from the different categories of the J&R website, such as: MP3 players, GPS, LCD TVs, Cameras, etc. We tested the consistency of the Noise Detector on five different pages and the results are shown in Table 4.5.

<table>
<thead>
<tr>
<th>Occurrence Percentage of Valid Blocks</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
<th>B8</th>
<th>B9</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>78.9%</td>
<td>84.2%</td>
<td>84.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>81.6%</td>
</tr>
<tr>
<td>p2</td>
<td>100%</td>
<td></td>
<td></td>
<td>92.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>96.4%</td>
</tr>
<tr>
<td>p3</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100%</td>
</tr>
<tr>
<td>p4</td>
<td>82.4%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>91.2%</td>
</tr>
<tr>
<td>p5</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100%</td>
</tr>
</tbody>
</table>

Average **93.8%**
Online retailers usually have more generic templates than the templates of news websites, such as CNN and BBC. This is why there are more low values in Table 4.5 than in Tables 4.1 and 4.3. The first four pages in Table 4.5 consist of 8 blocks each, while p5 consists of 7 blocks. Clearly, p5 follows a different trend than the other pages by having a 100% consistency with only one block being outputted. Table 4.6 shows the distribution of the number of blocks in the J&R’s set of pages. Around 65% of the pages are segmented into 8 blocks. An example page from the J&R website is shown in more detail in Appendix B.7.

<table>
<thead>
<tr>
<th>Table 4.6: Number of Blocks per Page for the CNN Website</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Blocks</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>Percentage</td>
</tr>
</tbody>
</table>

4.3.1.4 The Other Websites

This section investigates the consistency of the rest of the websites without presenting much detail as in the previous sections. The consistency results of the other seven websites are shown in Table 4.7.
Table 4.7: Average Consistency Results

<table>
<thead>
<tr>
<th>Website</th>
<th>Occurrence Percentage of Valid Blocks</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
<th>B8</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNET</td>
<td>p1</td>
<td>100%</td>
<td></td>
<td>42.9%</td>
<td>64.3%</td>
<td>42.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EJAZZ</td>
<td>p1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100%</td>
</tr>
<tr>
<td>Memory Express</td>
<td>p1</td>
<td>76.0%</td>
<td></td>
<td>100%</td>
<td></td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mythica</td>
<td>p1</td>
<td>100%</td>
<td></td>
<td>100%</td>
<td>58.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>p2</td>
<td>100%</td>
<td></td>
<td></td>
<td>50.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCMag</td>
<td>p1</td>
<td>41.7%</td>
<td></td>
<td></td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wikipedia</td>
<td>p1</td>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Some of the websites listed in Table 4.7 have low consistency values. This is due to the generic layout that these websites follow. For example, CNET pages have areas where customers can post their reviews and opinions. Consequently, the segmentation process produced different number of blocks per page for pages of the CNET website. This is evident in Table 4.8 which shows that CNET pages were segmented by VIPS into 6 different classes. These pages were segmented into 6, 7, 8, 9, 10, or 11 blocks. PDOC
of VIPS, which affects how coherent the output blocks are, was adjusted to 6. Despite this variety of the number of blocks in the CNET pages, consistency is still high. The blocks that have been outputted as valid for \( p1 \) are \( B2, B4, B5, \) and \( B6 \). Block \( B2 \) seems to be a valid block since it always gets outputted (100%) (see Appendix B.2 for segmented pages from CNET). The other blocks, namely \( B4 \) and \( B6 \), are outputted as valid blocks 43% of the time. \( B5 \) is outputted 65% of the time as a valid block. These low rates compared to the CNN and BBC websites are due to the more generic template that CNET has compared to CNN and BBC.

**Table 4.8: Number of Blocks in Different Websites**

<table>
<thead>
<tr>
<th></th>
<th>Number of Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td>CNET</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.7%</td>
</tr>
<tr>
<td>EJAZZ</td>
<td>9.1%</td>
</tr>
<tr>
<td>Memory Express</td>
<td></td>
</tr>
<tr>
<td>Mythica</td>
<td>38.5%</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>100%</td>
</tr>
</tbody>
</table>
EJAZZ has notably high consistency values. In fact, one block gets always outputted (100%) as the only valid block. This high consistency is due to the high stability in its page layout. Appendix B.2 illustrates an example page from the EJAZZ website. Also, it is worth noting that EJAZZ is consistent in the number of blocks its pages are segmented to (see Table 4.8).

MemoryExpress achieved high consistency values although its page layout is somewhat generic and rich. This is due to the consistency in the HTML code that is used to design its pages. Also, it does not include a comments section that can ruin the layout consistency. An example page from the MemoryExpress website is shown in the corresponding figure in Appendix B.8. Interestingly, all of the downloaded pages from MemoryExpress have been segmented into 8 blocks. This shows how well-templated MemoryExpress is. Consequently, its consistency values are high and close to perfect. Blocks B4 and B5 have been outputted 100% of the time, and B2 has been outputted 76% of the time. This shows how consistent the Noise Detector is.

Mythica has a fairly consistent behavior. As Table 4.8 shows, its pages are segmented into 4 or 5 blocks. Having the Mythica pages segmented into only two different block numbers increases its consistency. p1 of Mythica had B2 and B4 outputted as valid blocks in all the test cases, however B5 was outputted 58% of the time as a valid block. p2 has 4 blocks and so it follows a slightly different trend than p1. Again, B2 has been consistently outputted as a valid block, and B4 has been outputted as a valid block only 50% of the time. Example pages from Mythica are shown in Appendix B.3.
PC Magazine is another rich-content website. Its pages have dense information, which causes its segmentation to generate variety of block numbers. An example page is shown in Appendix B.4. PC Mag’s pages were segmented into seven different numbers of blocks, namely 9 to 15 blocks. Despite this fact, the Noise Detector is still able to achieve high consistency results by always outputting $B4$ as a valid block. But, having $B2$ outputted only 42% of the time as a valid block degrades the consistency results.

Wikipedia is another website that has highly consistent layout. All of its pages were segmented into 3 blocks. Because of the very high consistency in the segmentation process, the consistency of the results is also perfect with a result of 100%. Appendix B.5 shows an example page that has been used from Wikipedia. Although some pages have huge size compared to other pages, VIPS is still able to segment them based on their visual cues, and consequently all of these pages are segmented into three blocks.

The values reported in Tables 4.7, 4.5, 4.3, and 4.1 show that the Noise Detector is consistent in its output over a set of different websites. Finally, it is worth reemphasizing that these same websites (from online retailers, magazines, news, and encyclopedia) have been used in testing the other approaches described in the literature (Debnath et al., 2005; Vieira et al., 2006; Yi et al., 2003).

The consistency of detecting the template of a website depends heavily on whether its layout is so generic and different from one page to another. This is particularly true because VIPS will segment two pages (according to their layout) into different number of blocks even if they belong to the same website. As a result, the different number of blocks will affect the template detection negatively. This is clear as far as the PCMag website is concerned. However, Wikipedia, MemoryExpress, CNN,
BBC, and EJAZZ have fairly consistent layout throughout their pages, which results in high consistency values.

In our testing, we do not consider an average to the consistency values of the different blocks in a page. The following example explains why it is not recommended to use the average consistency as a measure for the goodness of a website. Let us consider the following example on the EJAZZ website.

Let \( D = \{d_1, d_2, \ldots, d_{11}\} \) be a set of 11 pages downloaded from EJAZZ website, and let \( d_1 \) be checked against \( D - \{d_1\} \). Table 4.9 reports the results of this cross testing.

<table>
<thead>
<tr>
<th>Page Checked Against</th>
<th>Valid Output Blocks of ( d_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d_2 )</td>
<td>( {B_2} )</td>
</tr>
<tr>
<td>( d_3 )</td>
<td>( {B_2} )</td>
</tr>
<tr>
<td>( d_4 )</td>
<td>( {B_2} )</td>
</tr>
<tr>
<td>( d_5 )</td>
<td>( {B_2} )</td>
</tr>
<tr>
<td>( d_6 )</td>
<td>( {B_2} )</td>
</tr>
<tr>
<td>( d_7 )</td>
<td>( {B_2} )</td>
</tr>
<tr>
<td>( d_8 )</td>
<td>( {B_2} )</td>
</tr>
<tr>
<td>( d_9 )</td>
<td>( {B_2} )</td>
</tr>
<tr>
<td>( d_{10} )</td>
<td>( {B_2} )</td>
</tr>
<tr>
<td>( d_{11} )</td>
<td>( {B_1, B_2} )</td>
</tr>
</tbody>
</table>
We have considered 11 pages in this example for simplicity. Clearly, $B_2$ is a valid block since it appears in all the result instances. However, $B_1$ appears just once along with $B_2$, and this gives $B_1$ a small probability to be a valid block. The average overall consistency is 55%, and this indicates that the EJAZZ website is not consistent which should not be the case. If we consider the appearance of $B_1$ as noise and do not consider it in the overall consistency calculation, the consistency of EJAZZ will be 100%. As a result, we can conclude that calculating the average consistency does not give an insight about the website.

As we mentioned previously, the PDODC value affects the number of blocks in a page, and this in turn affects the template detection process. Table 4.10 reports the different PDODC values for the tested websites.

<table>
<thead>
<tr>
<th>Website</th>
<th>BBC</th>
<th>CNET</th>
<th>CNN</th>
<th>EJAZZ</th>
<th>J&amp;R</th>
<th>MemoryExpress</th>
<th>Mythica</th>
<th>PCMag</th>
<th>Wikipedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDODC</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

4.3.2 Accuracy Measures

This section evaluates the accuracy of the Noise Detector in identifying valid blocks. Since the consistency of the Noise Detector as reported in Section 4.3.1 is fairly high, checking its accuracy is reasonable. The statistical measures to be used in evaluating the Noise Detector are computed by the following set of equations:
\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}
\]  \hspace{1cm} (4.1)

\[
\text{Specificity} = \frac{TN}{FP + TN}
\]  \hspace{1cm} (4.2)

\[
\text{Sensitivity (Recall)} = \frac{TP}{TP + FN}
\]  \hspace{1cm} (4.3)

\[
\text{Positive Predictive Value (Precision)} = \frac{TP}{TP + FP}
\]  \hspace{1cm} (4.4)

\[
\text{Negative Predictive Value} = \frac{TN}{TN + FN}
\]  \hspace{1cm} (4.5)

\[
F_\beta = (1 + \beta^2) \frac{\text{Precision} \times \text{Recall}}{\beta^2 \text{Precision} + \text{Recall}}
\]  \hspace{1cm} (4.6)

where TN, TP, FP and FN stand for true negatives, true positives, false positives, and false negatives, respectively. These four measures could be interpreted as follows: true negatives refer to noisy blocks correctly identified as noisy; true positives refer to informative blocks correctly identified as informative; false positives refer to noisy blocks wrongly identified as informative; and false negatives refer to informative blocks wrongly identified as noisy.

The Noise Detector is capable of differentiating two classes of blocks, namely noisy and informative with the former being the target class. A high sensitivity score (recall of the target class) means that the informative blocks have been well recognized; and a high specificity score (recall of the other class) means that the noisy blocks have been recognized. Specificity evaluates the effectiveness of the system in recognizing the negative cases. Therefore, it does not tell us how well the system recognizes positive
cases. On the other hand, positive prediction rate stands for precision of the target class and negative prediction rate stands for precision of the other class. Precision is the most important measure since it reflects the probability that a positive test is capable of handling the underlying condition being tested for. $F_\beta$ is a measure that combines precision and recall; it is mostly computed with $\beta=1$ and called F-1 score. Finally, the two measures recall and precision are redefined next for the target class by renaming informative blocks as valid blocks.

**Definition 4.1** [Precision]: *Precision is the number of valid returned blocks divided by the total number of returned blocks* ($P = \frac{\text{returned valid blocks}}{\text{all returned blocks}}$).

**Definition 4.1** [Recall]: *Recall is the number of valid returned blocks divided by the total number of valid blocks (the number of blocks that should have been returned as valid)* ($R = \frac{\text{returned valid blocks}}{\text{all valid blocks}}$).

Definitions 4.1 and 4.2 (as well as Equations 4.1 to 4.6) are specific to blocks in our case, but can be generalized to fit any discrete output: documents in retrieval systems, classes in classification problems, etc.

### 4.3.2.1 Spam Filtering Example

Let us consider the following spam filtering example. We have received 100 emails, 20 of which are actual spam emails and the other 80 are non-spam. The filtering technique has identified 15 emails as spam and the other 85 emails as non-spam.
Amongst these 15 spam-identified emails, 5 have been wrongly identified. The following are the statistical measures for this scenario:

True Positives (TP) = 10

True Negatives (TN) = 80

False Positives (FP) = 5

False Negatives (FN) = 5

\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} = \frac{10 + 80}{10 + 5 + 5 + 80} = \frac{90}{100} = 0.9
\]

\[
\text{Specificity} = \frac{TN}{FP + TN} = \frac{80}{5 + 80} = \frac{80}{85} = 0.94
\]

\[
\text{Sensitivity (Recall)} = \frac{TP}{TP + FN} = \frac{10}{10 + 5} = \frac{10}{15} = 0.66
\]

\[
\text{Positive Predictive Value (Precision)} = \frac{TP}{TP + FP} = \frac{10}{10 + 5} = 0.66
\]

\[
\text{Negative Predictive Value} = \frac{TN}{TN + FN} = \frac{80}{80 + 5} = 0.94
\]

\[
F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \times \text{Recall}}{\beta^2 \text{Precision} + \text{Recall}} = 2 \cdot \frac{0.66 \times 0.66}{0.66^2 + 0.66} = 0.66
\]

### 4.3.3 Noise Detector Accuracy Evaluation

We have applied the Noise Detector on an average of 100 pages per website. After downloading a set of pages from each website, we randomly picked a page. All pages in each set are then tested against their respective randomly-picked page. As a result, the Noise Detector creates new HTML files that are made up of the valid-marked blocks, i.e., 100 new clean files will be outputted for a set of 100 original pages. These clean HTML files are checked afterwards by an evaluator who reports the number of
valid and invalid blocks. In this evaluation process, the evaluator needs to look at the
original pages to determine which blocks are valid and which are invalid. Also, the
evaluator reports how many blocks are actually valid but are missing in the set of output
blocks.

The precision, recall, and F1-score values (for the target class) of the Noise
Detector for different websites are shown in Table 4.11. These accuracy results are
computed by testing (most of the time) only two pages against each other. Our approach
still outperforms most of the other approaches described in the literature. Recall that this
is all due to two main factors:

1. The Noise Detector operates at the semantic level by segmenting web pages
into coherent blocks.

2. The noise measure used by the Noise Detector incorporates three main
features of HTML: namely content, structure, and presentation.

Table 4.11: Precision, Recall and F1-score Results of the Noise Detector in
Identifying Noisy Blocks

<table>
<thead>
<tr>
<th>Website</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBC</td>
<td>100%</td>
<td>98.9%</td>
<td>99.4%</td>
</tr>
<tr>
<td>CNET</td>
<td>87%</td>
<td>100%</td>
<td>93%</td>
</tr>
<tr>
<td>CNN</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>EJAZZ</td>
<td>96.2%</td>
<td>100%</td>
<td>98%</td>
</tr>
<tr>
<td>J&amp;R</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>MemoryExpress</td>
<td>94.8%</td>
<td>94.8%</td>
<td>94.8%</td>
</tr>
<tr>
<td>Mythica</td>
<td>63.9%</td>
<td>100%</td>
<td>78%</td>
</tr>
<tr>
<td>PCMag</td>
<td>91.3%</td>
<td>100%</td>
<td>95.4%</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>
BBC, CNN, EJAZZ, J&R, PCMag, and Wikipedia have relatively high precision/recall results. By looking at their page layout, we can see that they are more consistent compared to CNET and Mythica. The Mythica website has the lowest precision/recall results. Later on in the evaluation, we prove that Mythica’s precision/recall results could be boosted by testing against an extra third page (see Section 4.3.5 for details).

Table 4.12 shows the average values of measures computed by Equations 4.1-4.6 for by considering all the websites enumerated in Section 4.2. These high values clearly show that the Noise Detector is capable of correctly identifying both noisy and informative blocks.

**Table 4.12: Accuracy Measures of Noise Detector**

<table>
<thead>
<tr>
<th>Measure</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>92.3%</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>99.3%</td>
</tr>
<tr>
<td>Specificity</td>
<td>86.2%</td>
</tr>
<tr>
<td>Positive Predictive Value</td>
<td>95.2%</td>
</tr>
<tr>
<td>Negative Predictive Value</td>
<td>99.8%</td>
</tr>
</tbody>
</table>

The figures reported in Table 4.12 shows that ND achieves high sensitivity and this means that ND can detect the target blocks with a very high accuracy up to 99.3% on average. Also, the negative predictive rate is close to 100%, which means that ND is almost perfect in detecting the blocks that are not in the target class. The specificity of ND is 86% which is not as high as the other measures. This means that ND will misclassify 14% of the informative blocks as noisy. Finally, having the positive
predictive rate as 95% indicates that ND will be accurate up to 95% in identifying the target noisy blocks. These results as a whole demonstrate the robustness of ND in the sense that it is able to successfully detect the two types of blocks with high accuracy. Unfortunately, in this regard we are not able to comment on the other approaches described in the literature because they only reported the recall and precision of the target class. Having the latter two values high does not provide much information about the recall and precision of the other class.

4.3.4 Number of Valid Blocks

We have introduced a dynamic way to decide on the number of valid blocks. This is not applied, however, in (Li & Ezeife, 2006) where Li & Ezeife use three as a static number of valid blocks. The extensive experiments we have conducted show that the number of valid blocks is variable and page-variant. Figure 4.3 has detailed statistics about the number of valid blocks for the different websites that we used in the experiments.
4.3.5 Comparison against FR Approach

In Section 4.3.3, we have shown the accuracy of the Noise Detector in detecting the template of different websites. All of our tested websites have been taken from the set tested by Vieira et al. (2006). In this section, we will compare our results with the results that were reported by Vieira et al. In the set of websites we have used, there are 8 common sites that can be used in the comparison. For all these websites, our F1-score results were higher even by using only two pages, except for the Mythica website. Still, the approach of Vieira et al. (2006) requires more than 25 pages on average to start giving good results. We will use the abbreviation FR (Fast and Robust Method) to denote the approach of Vieira et al. (2006) and ND for our Noise Detector.

As shown in Figure 4.4, our approach outperforms FR even when only two pages are considered. However, FR outperformed ND in the case of Mythica website. When a
page is tested against two other pages rather than one, the F1-score reported by ND for Mythica has increased from 78% to 93.3%. Figure 4.5 shows the F1-score of ND and FR by considering two pages and 90 pages, respectively.

![F1-score Considering Two Pages](image)

**Figure 4.4: F1-score Measure of ND vs. FR**

We see that ND does not perform better than FR for some websites such as PCMag and CNET. The common feature between these websites is having user comments and reviews. This adds more to the complexity of these websites' structures and makes it more difficult to detect templates with high accuracy by using only two pages. But still, ND has achieved high F1-score of 93% and 95.4% for CNET and PCmag, respectively. We believe that using more than two pages with such websites will generally boost the F1-score. In fact, our claim is supported by the improvement in the F1-score of the Mythica website as reported in Table 4.13.
**4.3.6 Refinement**

To achieve even higher precision and recall results in detecting noisy blocks, a page can be tested against two other pages from the same website. The intersection of both outputs is considered as the final valid result. Mythica has the lowest precision results in detecting noisy blocks. Table 4.13 shows how the accuracy of ND increases when we make this type of extra testing.

This extra step adds to the time consumed by the process, but increases the accuracy of ND. So, if the accuracy of detecting valid noisy blocks in a website is not acceptable, then this extra step will be necessary. This may be recursively applied until
either we get to the point where the accuracy never improves or the time consumed increases beyond expected limit. A human expert has to determine how many pages a given page needs to be cross tested against. Yet, as the results show, for most of the tested websites high consistency results are achieved even when using only one page without cross testing against many pages. The websites that required ND to process more than two pages could be classified as outliers as the trend in website development is concerned. In other words, it is very common to have a template for each website and even having the same template across a number of websites is not unusual.

4.3.7 Using More Than Two Pages

As we have mentioned in the previous section, using more than two pages can give better accuracy results. We used Mythica website to find out the effect of using more pages. Figure 4.6 illustrates the behaviour of the accuracy of Noise Detector when using more than two pages.

Figure 4.6: Precision of Noise Detector Using More Than Two Pages
As Figure 4.6 shows, the precision stabilizes at 91.7% since the precision of Noise Detector when using four pages is 91.7% and its precision when using five pages is also 91.7%.

4.4 Granularity

The granularity that ND can achieve depends solely on VIPS. That is, the more coherent the blocks are, the higher the granularity will be. Recall that the basic unit that ND operates on is a block. So, a whole block will be either outputted or removed. Therefore, if a block has small noisy parts in it, these parts will be in the output if their respective block is valid. Figure 4.7 shows an example from the BBC website where a small part of Block3 is part of the BBC template, i.e., it appears in all the BBC pages, though it is part of an output valid block.

![Figure 4.7: Small Noisy Parts](image)

The granularity problem can be overcome to some extent by increasing the PDOC value of VIPS. The more the PDOC is, the more coherent the output blocks are. In VIPS, PDOC can be adjusted to a maximum of 10. But still, there will be some parts which will not be separated in a single block. Furthermore, if ND has many tiny blocks as a result of increasing PDOC, it will not give accurate results because of the confusing similarity measures amongst blocks in the two input pages. The PDOC values that we have reported
in Table 4.10 are recommended for the set of websites used in the experiments. Notice that PDOC for most of the tested sites is either 5 or 6.

4.5 General Comparison with Other Approaches

Table 4.14 shows a comparison between our approach and the other related approaches. In this table, we compare our approach with LH (Lin & Ho, 2002), SST (Yi et al., 2003), and Bar-Yossef (Bar-Yossef & Rajagopalan, 2002).

The approach that Lin and Ho (LH) proposed in (Lin & Ho, 2002) uses the <TABLE> tag to partition pages. Then, keywords are used to construct a feature-document matrix, through which they calculate the entropy of these features, i.e., content is the main feature that has been used.

The SST approach which was proposed in (Yi, et al. 2003) mainly uses the structure of web pages to construct a huge tree (SST). The entropy of its internal nodes is then calculated to decide their noise values. Presentational features are used to determine the noise values of the leaf nodes. This approach requires an average of 400-500 pages to build the SST, which identifies the noisy parts as a result.
Table 4.14: Comparison between Different Approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>Involves Segmentation</th>
<th>Incorporates structure</th>
<th>Incorporates content</th>
<th>Incorporates presentation</th>
<th>Requires a big training set of pages</th>
<th>Was tested on different types of sites</th>
<th>Was tested only on news websites</th>
</tr>
</thead>
<tbody>
<tr>
<td>LH</td>
<td>X²³</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SST</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Bar Yousef</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>FR</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ND</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Bar-Yossef and Rajagopalan (2002) proposed a template detection approach that segments a page into a set of pagelets depending on a linkage percentage. They use shingles to determine whether a block is part of the template or not. A shingle is essentially a content measure.

To detect the template in a page, our approach mostly requires only one other page to be tested against the given page. The strength of our measure that enables Noise Detector to give satisfactory results is due to combining the structure, content, and presentation measures, which are all characteristics of HTML pages. Our approach showed excellent results on a variety of websites, including news website.

4.6 User Study

This section emphasizes the positive impact of template removal by using the output of Noise Detector with an information retrieval system and a web summarization tool.
We conducted a user study to evaluate the two applications: information retrieval and web summarization. We had 10 graduate students from the computer science department and the faculty of engineering to evaluate these applications. In the following sections, we will provide details about this user study.

4.6.1 Information Retrieval System

Information retrieval (IR) systems (search engines) have become the most convenient tool for users to get the information they need. A search engine consists of the following three major components:

1. Crawler: this is the part responsible for grabbing files and any data available on the Internet. This is alternatively called spider. The target data ranges between HTML files, images, pdf files, and many other file types.

2. Indexer: this component is responsible for building an index that helps answer user queries efficiently. Most search engines use the inverted index structure for their indexes (see Figure 4.8 and Figure 4.9 which have been taken from http://apple.com/).

3. Query processor: this component is responsible for examining the query against the index that has already been built.

Search engines might have variations in their index structures, by adding extra Meta data. For example, Figure 4.8 shows a simple inverted index in which terms are mapped to their respective documents. However, Figure 4.9 shows a similar inverted

23 The segmentation used in (Lin & Ho, 2002) is based only the <TABLE> tag. This limitation made their
index with an extra piece of information, which is the position of the term in its respective document. This information can improve the relevancy score of the output set of documents because the search engine will consider not only the occurrence of terms, but also the actual position of these terms.

As we see the input to a search engine is web documents which are heterogeneous and diverse. Moreover, web documents contain noisy information such as templates. In this section, we show that cleaning documents from noise can improve the effectiveness of search engines by giving more satisfactory results to the users.

![Figure 4.8: Simple Inverted Index (adapted from http://apple.com/)](image)

testing limited mostly to news websites. Besides, not all the websites use the `<TABLE>` tag to segment their pages.
We used Indri (Trohman et al., 2005), which is a new search engine that has been built as a cooperative effort between the University of Massachusetts and Carnegie Mellon University. Indri is part of the Lemur project, which is a toolkit designed to facilitate research in language modeling and Information Retrieval (IR) (Allan et al., 2003).

4.6.1.1. Experiments Setup

We downloaded a set of around 2,000 pages from the set of websites listed in Section 4.3. This is considered as the set of raw documents since it contains the original pages. Another set that is used in the information retrieval evaluation is a clean set of documents. The original raw documents were cleaned by the Noise Detector to produce
this set of clean documents. Presumably, these documents do not include their respective template (template-free documents). Then, we built two indexes; one was built using the set of raw documents and the other one was built using the set of clean documents. As a result, we had two instances of Indri that we can evaluate. Every evaluator had to run the same query on both search engine instances, and then evaluate the reported results.

4.6.1.2 IR Evaluation Measures

In the following sections, we will examine the two instances of Indri search engine using different measures that fit information retrieval systems, i.e., we reword the definitions of recall and precision to match the investigated domain.

4.6.1.2.1 Precision

Given a user query $q$ and a collection of indexed documents $D$, the search engine retrieval algorithm will compute the relevance scores for all documents in $D$ and output them ordered according to their relevance score. Let $R_q$ be the ranking of the documents based on their relevance scores, i.e. $R_q := <d_1^q, d_2^q, ..., d_M^q>$, where $d_i^q \in D$ is the most relevant document with respect to the query $q$ in $D$. Precision at rank $i$ is the fraction of documents from $d_i^q$ to $d_i^q$ that are relevant:

$$p(i) = \frac{r_i}{i}$$

(4.7)

where $r_i$ is the number of relevant documents in $<d_1^q, d_2^q, ..., d_N^q>$. 
According to Equation 4.7, if we have 3 relevant results in the first 5 reported results, the precision at rank 5 is 3/5. Precision is a widely used measure in information retrieval systems and data mining systems in general. In the literature, researchers evaluate their IR systems using the precision of maximum the first 30 documents since most users tend not to care about results beyond the top 30 reported results. Partial average precisions at different ranking positions are used; precision for the top 5 results, precision for the top 10 results, precision for the top 15 results and so on, denoted $P@5$, $P@10$, $P@15$, respectively.

Our evaluators got an evaluation form in which they had to write the queries that they submitted on both engines. The evaluation form forces them to use 1-word, 2-word, and 3-word queries (excluding stop words). After executing each query on each search engine, the first 15 results are to be checked by the evaluator for their relevancy. Since the number of documents used to build the search engines’ indexes is relatively small, relevancy was not strict, i.e., a partly relevant document would be considered as a relevant result. This assumption will not affect the evaluation since it is applied on both search engines.

Figure 4.10 illustrates a comparison between the two instances of Indri; the instance using clean documents and the instance using original documents. The $P@5$ of the clean instance of Indri performs better than the other one. Interestingly, for the 1-word queries, the instance of Indri using clean documents has worse results than the other one. We believe that this is due to the following: some HTML tags are given certain scores in the retrieval algorithm (e.g., META, TITLE, H1, H2, etc). Some of these tags are removed by VIPS from the output valid blocks. The removal of such tags affects the
relevance scores and therefore the precision. This is clearer in the 1-word queries because there is only one word to be evaluated by the retrieval algorithm. In the case of two or three words, however, the retrieval algorithm looks for the occurrence of more than a word. In fact, Indri stores the position of the words in its index (see Figure 4.9) and uses them in its scoring algorithm.

Figure 4.10: Precision at Rank Position 5

Figure 4.11: Precision at Rank Position 10
Note that the trend is the same in both Figure 4.10 and Figure 4.11. The overall precision for all queries is illustrated in Figure 4.12. The results show that removing templates help increase the precision of search engines.

![All Queries](image)

**Figure 4.12: Overall Precision**

All evaluators marked most of the results at rank position 11 and more as irrelevant. So, we stopped at rank position 10 because our data set that was used to build the indexes is small. It is worthwhile to mention that recall is not a practical measure to be used with information retrieval system because the actual number of relevant documents needed to compute the recall for a specific query is not known. This is due to the huge size of the document collection, and therefore the difficulty of identifying them.

4.6.1.2.2 Mean Position of the First Relevant Result
The Mean Position of the first relevant results (MPOS) is another measure that is used to evaluate a search engine. It tests the rank position of the first relevant result; the closer to 1 the MPOS is, the better the search engine is. Table 4.15 shows the positive gain achieved by the search engine which uses clean documents. Again we see that 2-word queries have worse results on the search engine that uses clean documents. We believe this is due to the reasons aforementioned in Section 4.6.1.2.1.

We asked one of the evaluators to pick 5 random queries with different sizes. Table 4.16 shows these queries along with their MPOS values.

<table>
<thead>
<tr>
<th>Query Type</th>
<th>Clean Index</th>
<th>Raw Index</th>
<th>MPOS Gain [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-word query</td>
<td>2.43</td>
<td>2.57</td>
<td>5.6%</td>
</tr>
<tr>
<td>2-word query</td>
<td>5.06</td>
<td>4.31</td>
<td>-17.4%</td>
</tr>
<tr>
<td>3-word query</td>
<td>1.4</td>
<td>2.4</td>
<td>41.7%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query</th>
<th>Clean Index</th>
<th>Raw Index</th>
<th>MPOS Gain [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>All in one printer</td>
<td>1</td>
<td>4</td>
<td>300%</td>
</tr>
<tr>
<td>HP computers</td>
<td>7.34</td>
<td>10.66</td>
<td>45%</td>
</tr>
<tr>
<td>LCD screen</td>
<td>2.5</td>
<td>4.25</td>
<td>70%</td>
</tr>
<tr>
<td>Technology</td>
<td>2.33</td>
<td>2.33</td>
<td>0%</td>
</tr>
<tr>
<td>Computers</td>
<td>1.67</td>
<td>3.9</td>
<td>134%</td>
</tr>
</tbody>
</table>
4.6.2 Web Summarization

Summaries play a major role in the usability of the Web. They intend to give an overview of a document such as snippets in search engines. Although automatic summarization research has existed for more than 50 years, Web summarization has been very limited. One main reason for that is the difficulty in identifying the relevant information in a web page (Delort et al., 2003). In this section, we show the impact of our approach of removing templates to improve the quality of the summarization process.

4.6.2.1 Evaluation

Evaluating summaries is not a straightforward process because it is subjective, so there is no standard measure that we can use, though there are few ways that have been proposed in the literature to evaluate summarization systems. For instance, (Hand & Sundheim, 1998; Hovey & Lin, 1999) suggested three ways for the evaluation:

1. Shannon Game: this is a variant of Shannon’s measures in information theory. Here people are asked to reconstruct the original having seen either the full text or a summary.

2. Questions Game: the evaluators are asked to answer questions that have been previously drawn up about the original. Then the answers of the original documents are compared with the answers of the summaries.

3. Classification Game: the evaluators are asked to classify the texts into one of \( N \) categories. A good summarization system should generate summaries that are classified in the same bin as the corresponding original documents.
In our evaluation, we used the question game. We had a set of eight documents that were selected randomly to be tested. For each of the eight documents, five questions were prepared for evaluators to answer. We used Extractor\textsuperscript{24}, which is a commercial product for Web summarization. We produced a summary for each document using its clean and raw versions. These two summaries are then given to two different evaluators to answer the five questions of their respective document. Finally, we analyzed the answers of the evaluators and created the statistics reported in Table 4.17.

<table>
<thead>
<tr>
<th>Doc. No.</th>
<th>Clean</th>
<th>Raw</th>
<th>No. Of Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1.5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2.5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>2.5</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>20.5</td>
<td>18</td>
<td>38</td>
</tr>
</tbody>
</table>

The question-based evaluation does show better results for clean documents over the raw documents. This is partly due to the small number of questions that were written for each page. It was difficult for us to redo it and call the evaluators. Still, the results that are reported in Table 4.16 show the positive impact of cleaning web pages.

\textsuperscript{24} http://www.extractor.com/ provides picoFocus (intelligent search engine) with summarization service to create snippets for its results.
Another overall feedback evaluation was conducted in the same user study. For each document, we generated a summary for its clean version and its original version. We gave these pairs of summaries to the evaluators to compare. A copy of the evaluation form we used is shown in Appendix C.1. They had to evaluate both keyphrases and highlights. Keyphrases represent the key terms in a document, and highlights are the actual summary (i.e., sentences that make up the summary). An example of a summary generated by Extractor is depicted in Figure 4.13. In fact, Figure 4.13 shows a summary of web page from the CNN website and Figure 4.14 shows a summary for the cleaned version by ND of that page.

Keyphrases:
- death
- CNN
- Nichols
- Barnes
- death penalty
- husband
- Wilhelm

Highlights:
- Judge's widow testifies at death penalty hearing - CNN.com
- ATLANTA, Georgia (CNN) -- Relatives and friends of a judge and court reporter killed in a 2005 shooting at Atlanta's Fulton County Courthouse took the stand Thursday in the penalty phase of the gunman's trial.
- Claudia Barnes testified about losing her husband, the judge who was shot by Brian Nichols.
- Claudia Barnes, widow of Fulton County Superior Court Judge Rowland Barnes, recalled asking permission to hold her husband's hand one last time before his body was cremated.
- Nichols was also convicted of killing David Wilhelm, a federal customs agent, hours later at Wilhelm's home in the Buckhead section of Atlanta.

Figure 4.13: Summary of an Original Web Page from CNN
An evaluator had to read the original documents first. Then, he/she had to read both summaries for that document; the summary of the clean version and summary of the original version and evaluate each compared to the other. Figures 4.13 and 4.14 are summaries of an original document and a clean document, respectively. They had to mark which key phrases were relevant and which were not. Also, they had to give a score out of 10 for each of the two, with respect to each other. This evaluation shows clearly that summaries generated from clean documents have better user satisfaction than the other ones. In fact, their results are 20% better than the summaries of the original documents as reported in Figure 4.16.

Keyphrases:
- death
- Nichols
- Barnes
- husband
- Wilhelm
- judge
- David

Highlights:
- ATLANTA, Georgia (CNN) -- Relatives and friends of a judge and court reporter killed in a 2005 shooting at Atlanta's Fulton County Courthouse took the stand Thursday in the penalty phase of the gunman's trial.
- Claudia Barnes testified about losing her husband, the judge who was shot by Brian Nichols.
- Claudia Barnes, widow of Fulton County Superior Court Judge Rowland Barnes, recalled asking permission to hold her husband's hand one last time before his body was cremated.
- She remembered running her hands over the judge's face -- over the temple, where the bullet fired by escaped prisoner Brian Gene Nichols entered his head -- and over the judge's beard, which she always kept trimmed.
- Nichols was also convicted of killing David Wilhelm, a federal customs agent, hours later at Wilhelm's home in the Buckhead section of Atlanta.
- Jurors heard victim impact statements Thursday as part of Nichols' penalty phase, in which they will decide whether he will receive the death penalty sought by prosecutors.

Figure 4.14: Summary of a Cleaned Web Page from CNN
4.7 Storage Space

Another benefit for identifying templates is saving storage space. In the data set used in the experiments, the clean collection of documents was 2.68 times smaller than the original size. This can be beneficial especially when search engines need to store and index huge number of documents.
4.8 When Noise Detector Stops Giving Satisfactory Results

Noise Detector gives satisfactory results when it operates on well-templated websites because Noise Detector was designed from the beginning to detect templates of websites that have fairly constant layout. This assumption should be satisfied to get satisfactory results. The Amazon\(^{25}\) website is a good example on when Noise Detector stops giving good accuracy results. Table 4.18 shows the accuracy of Noise Detector in detecting the informative blocks of Amazon web pages.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td>53.6%</td>
<td>55.1</td>
<td>54.4</td>
</tr>
</tbody>
</table>

As Table 4.18 shows, Noise Detector gives poor accuracy results when the website being processed does not follow a common template. As a result, we recommend that the website being processed is well-templated. Although the Amazon website has a pre-defined template, Noise Detector has given poor accuracy results. This is due to the very generic layout in the sense that its pages are not limited to a specific rigid layout and contain information that is dependent on the page topic. The page size is another accuracy-degrading factor as its pages are lengthy and contain much related information and links.

\(^{25}\) http://www.amazon.com/
4.9 Conclusion

In this chapter we have shown that Noise Detector is accurate in detecting noisy blocks that constitute templates in websites. As one of the main contributions is reducing the analyzed set of documents to only two pages, we started the evaluation process by proving the consistency of the Noise Detector; then, we run other experiments to report its accuracy. Our data set consists of websites that were used in the literature. These websites include news, online retailers, encyclopedia, and magazines. The F1-score results for these websites were high and our approach outperformed (Vieira et al., 2006) in most of the websites we tested. However, the Mythica website results indicated a case for which the developed Noise Detector could not report good results by depending on only two pages in the analysis; hence, we extended the methodology into using an extra page when necessary, especially for websites that do not have a regular template. So, we realized that the results for Mythica have been improved by testing each page against two pages instead of one page; explicitly, this boosted the F1-score result by 15%. Four websites had F1-score results of 100%.

The user study that we have conducted ensures the importance of removing templates from websites as a preprocessing step for other applications like web mining. We have used an information retrieval system for testing. The user satisfaction for the search engine that used clean documents was much better than the search engine that uses raw documents. We have used different measures to evaluate the information retrieval systems. Another tool that has been used in our testing is a web summarization tool. After conducting the user study, we have concluded that users ranked the summaries of template-free documents higher than summaries of raw documents. The quality of
summaries generated from clean documents is better than the summaries generated from raw documents. Both tools have shown the positive impact that removing templates has on web mining research.
Chapter Five: Summary, Conclusion, & Future Work

The noise detector developed in this thesis could be seen as the result of realizing the importance of noise reduction in turning web pages into more useful source of information especially for applications dedicated to deal with the actual informative content of websites. This chapter summarizes the basic aspects of the developed approach and highlights the main lessons learned from this experience. The conducted experiments demonstrate that the approach described in this thesis could be recognized as a major step in the right direction. However, further improvements and extensions could lead to a more comprehensive product; the last section of this chapter is dedicated to address these possible extensions.

5.1 Summary
This thesis introduced the Noise Detector which is an automated system capable of detecting and removing templates from web pages intended to be used as input for certain web centric application. Noise Detector starts by using VIPS to segment pages into semantically coherent blocks. To detect a template, Noise Detector requires only two pages. Structure and content similarities are computed amongst the blocks of the first page and the blocks of the second page. If the combined similarity value between two blocks is high, then these two blocks are most likely part of the template of their common website. The similarity between two blocks is essentially a noise measure that identifies the noisiness of the two blocks. When the number of blocks in both pages is different,
there will be unmatched blocks. As a result, these blocks do not have content and structure similarities and will use presentational noise only.

Noise Detector calculates a cut-off threshold value that identifies which blocks are noisy and which blocks are assumed to be informative. This cut-off value is dynamically calculated by considering the two pages that are being tested.

The conducted experiments first examined Noise Detector as a template detector. The applicability of its results is then tested using an information retrieval system and a Web summarization tool to test its impact on web mining problems. The summarization experiments showed better results than the information retrieval process. This clearer positive impact for web summarization is due to the small number of documents used to build the information retrieval index.

5.2 Conclusion

The proposed Noise Detector applies robust noise measure to discover templates. The experiments that we have conducted show that Noise Detector is able to detect templates with high accuracy in websites from different domains and not only news websites. The experiments also show that removing noisy information, i.e., templates in our case, boosts the accuracy of web mining tasks. And since the Web contains 40-50% templated websites, removing these templates becomes a necessity for better mining results. In fact, our experiments show that the required storage space shrinks by about 3 times the original space. Also, we have shown that Noise Detector achieves better accuracy results in detecting templates; an average F1-score of 94% was achieved. This performance is achieved despite the fact that Noise Detector uses much fewer numbers of input pages.
Lin & Ho (2002) test only news websites which are usually highly consistent in their layout, and hence it easier to detect their templates. However, Noise Detector has been tested on a variety of websites with diversity in the layout, including news websites, online retailers, online encyclopedias, and online magazines. These websites have been collected from different systems that were proposed in the literature.

After showing that Noise Detector is highly accurate in detecting templates, we used its clean output files with an information retrieval system. The quality of the results of the information retrieval system that uses template-free documents (clean documents) was clearly better than the other information retrieval system that uses raw documents. We showed that the Mean Position of the First Relevant Result measure is better for the information retrieval system with template-free documents. Also, the precision results of it are better than the information retrieval system that uses raw documents.

The feedback we have got from the participants in the conducted user study shows the impact of cleaning web documents on improving the quality of their summaries. We used question-based evaluation and general-feedback evaluation. Both evaluations showed a big improvement in the results and a better user satisfaction. In fact, the general-feedback evaluation boosted the user satisfaction by 20%, which is a huge jump. Also, the question-based evaluation had better results for the clean documents than the raw documents.

These conclusions assure the importance of cleaning web documents as a pre-processing step in any web centric application like the web mining process.
5.3 Future Work

This thesis proposes and evaluates a new approach for template detection. Some technical issues are still to be worked out. Noise Detector does not have an automated crawler to automatically collect documents from a website. Rather, we had to manually download the documents that we used in the experiments. Though, with the emergence of the Internet and the huge interest it has received, finding and integrating a crawler with Noise Detector should not a major issue.

Noise Detector uses three different measures; content, structure, and presentation. These measures are combined later in the template detection process (using the UpdateWeights algorithm in Figure 3.18) with semi-automatically calculated weight for each of them. Calculating these weights using a formula that depends on the similarity values for all blocks may give better results than using a predefined set of weights.

Noise Detector assumes its input pages have a common template. If these pages do not have a common template, Noise Detector gives its results according to the computed similarity values. Noise Detector can have the ability to advise whether the two pages have the same template and then detect and remove it. This can be done using the computed similarity values e.g., standard deviation, minimum value, maximum value, etc.

Noise Detector uses a static threshold in the early filtering stage. We believe that using a dynamically calculated threshold that depends on the average size of blocks in a page will give better results.

Finally, we believe that the position of a block in the page is another important feature that can be added to the other three features. This feature can be valuable
especially with the unmatched blocks since Noise Detector uses only the presentational feature with them.

These points have been identified as future research directions that can automate Noise Detector more and make it more robust and reliable.
References


International Conference on Innovative Computing Information and Control (pp. 1-4). Kaohsiung, Taiwan: IEEE Computer Society Washington, DC, USA.


Tai, K.-C. (1979). The Tree-to-Tree Correction Problem. 26 (3).


VIPS uses four main cues: text, color, tag, and size. Based on these cues, table heuristic rules are used in the segmentation process sorted by their priority (Cai et al., 2003):

<table>
<thead>
<tr>
<th>Rule</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1</td>
<td>If the DOM node is not a text node and it has no valid children, then this node cannot be divided and will be cut.</td>
</tr>
<tr>
<td>Rule 2</td>
<td>If the DOM node has only one valid child and the child is not a text node, then divide this node.</td>
</tr>
<tr>
<td>Rule 3</td>
<td>If the DOM node is the root node of the sub-DOM tree (corresponding to the block), and there is only one sub DOM tree corresponding to this block, divide this node.</td>
</tr>
<tr>
<td>Rule 4</td>
<td>If all of the child nodes of the DOM node are text nodes or virtual text nodes, do not divide the node.</td>
</tr>
<tr>
<td></td>
<td>• If the font size and font weight of all these child nodes are same, set the DoC of the extracted block to 10.</td>
</tr>
<tr>
<td></td>
<td>• Otherwise, set the DoC of this extracted block to 9.</td>
</tr>
<tr>
<td>Rule 5</td>
<td>If one of the child nodes of the DOM node is line-break node, then divide this DOM node.</td>
</tr>
<tr>
<td>Rule 6</td>
<td>If one of the child nodes of the DOM node has HTML tag <code>&lt;HR&gt;</code>, then divide this DOM node.</td>
</tr>
<tr>
<td>Rule 7</td>
<td>If the sum of all the child nodes’ size is greater than this DOM node’s size, then divide this node.</td>
</tr>
<tr>
<td>Rule</td>
<td>Condition</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
</tr>
<tr>
<td>8</td>
<td>If the background color of this node is different from one of its children’s, divide this node and at the same time, the child node with different background color will not be divided in this round.</td>
</tr>
<tr>
<td></td>
<td>• Set the DoC value (6-8) for the child node based on the html tag of the child node and the size of the child node.</td>
</tr>
<tr>
<td>9</td>
<td>If the node has at least one text node child or at least one virtual text node child, and the node's relative size is smaller than a threshold, then the node cannot be divided</td>
</tr>
<tr>
<td></td>
<td>• Set the DoC value (from 5-8) based on the html tag of the node</td>
</tr>
<tr>
<td>10</td>
<td>If the child of the node with maximum size are small than a threshold (relative size), do not divide this node.</td>
</tr>
<tr>
<td></td>
<td>• Set the DoC based on the html tag and size of this node.</td>
</tr>
<tr>
<td>11</td>
<td>If previous sibling node has not been divided, do not divide this node</td>
</tr>
<tr>
<td>12</td>
<td>Divide this node.</td>
</tr>
<tr>
<td>13</td>
<td>Do not divide this node</td>
</tr>
<tr>
<td></td>
<td>• Set the DoC value based on the html tag and size of this node.</td>
</tr>
</tbody>
</table>
APPENDIX B: EXPERIMENTS

Each section of the following shows an example page(s) on the websites that we have tested in our experiments.

B.1. CNET

The following figure represents B2 in the cross-tested page from the CNET website (http://www.cnet.com).

![Figure B.1: Example Valid Block from a CNET Web Page](image)
Figure B.2: Segmented Page from CNET Website
Figure B.3: Segmented Page from EJAZZ Website
Figure B.4: Segmented Page from Mythica Website

Figure B.5 shows another page from Mythica website that is very close to Figure B.4.
Islamic mythology

Islam was promulgated by the Prophet Muhammad in Arabia in the 7th century AD. The term 'Islam' literally means 'surrender', as in surrender to the will of Allah, Allah (Arabic: God) is viewed as the sole God, creator, sustainer, and restorer of the world. He will, revealed through his messenger Mohammed, made his revelations through the sacred scriptures, the Qur'an (Koran).

Pre-Islamic Arab and Persian traditions, which were essentially pagan, developed a wonderful body of myth and folklore. Jinn, elves, demons, holy men and women, and great heroes played their part in sparkling collections of folkloric tales and fables. It is only natural that people, throughout history, would have been entertained and have attached their own heroes, and Islamic literature, therefore, is rich in such material.

However, we are fully aware that pure Islam is entirely monotheistic and does not encourage the creation of anthropomorphic figures, play of fantasy, or anything suggesting multiplicity of gods and idol worshiping. None of the articles or stories that appear in this section presume to be a religious discussion. Many excellent books and articles on formal Islam are available to the interested reader.

* Browse through the list of available articles in this area.

Sayed M.M. Lindermark

There are currently 155 articles in this area.

This section was last updated on July 04, 2017.

**Related link:**

**Islam 101:**

An educational site on Islam, its way of civilization and includes an entry course on Islam. It presents Islam on a contemporary basis. Descriptions of beliefs and topics to its practice.
B.4. PCMag

Figure B.6: Segmented Page from PCMag Website
Figure B.7: Segmented Page from Wikipedia Website
No charges in death of man rolled into canal 50 years ago

Figure B.8: Segmented Page from CNN Website
Figure B.9: Segmented Page from J&R Website
B.8. Memory Express

Figure B.10: Segmented Page from MemoryExpress Website
APPENDIX C: SUMMARIZATION EVALUATION FORM

Figure C.1 shows the evaluation that our evaluators had to use to evaluating the summaries they were given. This form was used in the general-feedback evaluation of summaries. For the question-based evaluation, the other form was basically a set of questions on an article that they had to answer.

<table>
<thead>
<tr>
<th>Keyphrases:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• economy</td>
</tr>
<tr>
<td>• Thai</td>
</tr>
<tr>
<td>• bank</td>
</tr>
<tr>
<td>• airports</td>
</tr>
<tr>
<td>• blockade</td>
</tr>
<tr>
<td>• political turmoil</td>
</tr>
<tr>
<td>• assistant governor Duangmanee</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Highlights:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Large cut in Thai interest rates</td>
</tr>
<tr>
<td>• Thailand's central bank has cut interest rates by the largest amount in eight years, as it aims to lift an economy hit by recent political unrest.</td>
</tr>
<tr>
<td>• Its move comes after anti-government protestors ended a week long blockade of Bangkok's two airports, grounding both passenger and cargo flights.</td>
</tr>
<tr>
<td>• &quot;Domestic political problems are likely to have greater repercussions on economic growth than previously assessed, particularly to confidence and tourism,&quot; said assistant governor Duangmanee Vongpradhip.</td>
</tr>
<tr>
<td>• The Thai government warned earlier this week that the political turmoil had shattered business confidence, and that the economy may go into recession next year, causing an increase in unemployment.</td>
</tr>
</tbody>
</table>

Figure C.1: Summary – First Page of an Evaluation Form
**Keyphrases Evaluation**

Please write down in the table below how many relevant and how many irrelevant keyphrases there were in the summary with respect to the original document.

<table>
<thead>
<tr>
<th>No. of Relevant Keyphrases</th>
<th>No. of Irrelevant Keyphrases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Comments:

**Highlights Evaluation**

Please write down in the table below how many relevant and irrelevant highlights there were in the summary according to the original document.

<table>
<thead>
<tr>
<th>No. of Relevant Highlights</th>
<th>No. of Irrelevant Highlights</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Comments:

**General Ranking:**

By comparing this summary to the original document and the other summaries of this document, write down the ranking of this summary.

*(10 is the highest score and 1 is the lowest)*

<table>
<thead>
<tr>
<th>Ranking:</th>
</tr>
</thead>
</table>

Comments:

Figure C.2: Evaluation Form – Second Page