0-2561

AUTOMATIC IDENTIFICATION OF INFORMATIVE SECTIONS FROM WEB PAGES

by

P.RANJITH KUMAR

Reg. No : 0720108014

of

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE – 641 006

(An Autonomous Institution affiliated to Anna University, Coimbatore)

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements for the award of the degree

Of

MASTER OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

MAY, 2009

BONAFIDE CERTIFICATE

Certified that this project report titled "**AUTOMATIC IDENTIFICATION OF INFORMATIVE SECTIONS FROM WEB PAGES**" is the bonafide work of **P. Ranjith Kumar (0720108014)** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report of dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

GUIDE
(Mr.E.A.Vimal M.E)

HEAD OF THE DEPARTMENT
(Dr.S.Thangasamy, Ph.D)

The candidate with **University Register No. 0720108014** was examined by us in the Project viva-voce examination held on 05.05.2009

5/5/09
INTERNAL EXAMINER

5/5/09
EXTERNAL EXAMINER

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(An ISO 9001:2000 Certified Institution) Accredited by NBA-AICTE

COIMBATORE - 641 008

National Conference on Computing, Communication and Information Systems **NCCCIS-2009** **February 13-14, 2009**

Organized by ***Department of Information Technology***

Certificate

It is certified that ..... P.Ranjith Kumar ..... has presented a paper titled

Automatic Identification of Informative Sections from Web pages ....., in the National Conference on Computing, Communication and Information Systems (NCCCIS-2009) held at Sri Krishna College of Engineering and Technology, Coimbatore on February 13-14, 2009.

Conference Secretaries

Organizing Secretary

Principal

Department of Information Technology
Ministry of Communications & Information Technology
Government of India

WIPRO
Applying Thought

ABSTRACT

The Web is a huge resource for people who use search engines to search for specific pages related to their specific needs. As the Internet provides millions of web pages for each and every search term, getting interesting and required results quickly from the Web becomes very difficult. Most clients and end-users search for the primary content, and largely do not seek the non primary contents such as navigational sidebars, advertisements, flash animations etc.,. Although several techniques have been developed to the problem of identifying primary contents, their use is still not spread, mostly because of the need for high human intervention and the low quality of the extraction results. Automatic extraction of web pages into relevant categories is the current research topic which helps the users to get relevant results.

The goal of this research is to extract relevant contents from web pages with the help of three independent algorithms Content Extraction algorithm, K-Feature Extraction algorithm and L-Extraction algorithm. The first step of this research is to segment the web pages into blocks and then with the help of algorithm identify the primary contents based on the blocks that do not occurs a large number of times and looking for blocks with the desired feature values. These algorithms identify relevant content blocks with lower precision and recall values, reduce the storage requirement for search engines, results in smaller indexes and thereby faster search times, and better user satisfaction. Extraction of "useful and relevant" content from web pages has many applications, including cell phone and PDA browsing, speech rendering for the visually impaired, and text summarization.

ஆய்வுச் சுருக்கம்

இனணயம் என்பது ஒரு வளமைமிக்கது, அதில் தேடுதல் பொறி கொண்டு மக்கள் அவர்களுக்கு தேவையான தகவல்களை மட்டும் தேடுகின்றனர். இனணயதளம் கோடிக்கணக்காண வலைப்பக்கங்களை கொண்டிருந்தாலும் உரிய தகவல்களை விரைவாக பெருவதற்குச் சற்று கடினமாக இருக்கிறது. இனணயத்தில் தேடுதல் பொறி கொண்டு தேடும் பொழுது தேவையற்ற தவகல்கள் அதாவது விளம்பரங்கள்,அசைவூட்டங்கள் ஆகியவைகளும் சேர்ந்து கிடைக்கிறது எனவே இவைகள் தேடுதல் நேரத்தை அதிகரிக்கிறது. தேடுதல் நேரத்தை அதிகரிக்க அதிக உத்திகள் இருந்தாலும் அது தானியங்கி ஆற்றலை பெற்றதாக இல்லை அதற்கு மனித செயற்பாடுகள் தேவைப்படுகிறது மற்றும் அந்த உத்திகள் செய்தி எடுத்தலுக்கு குறைவான தரத்தைக் கொடுக்கிறது.

இந்த ஆய்வின் நோக்கம் வலைப்பக்கத்திலிருந்து பொருத்தமான தகவல்களை மனித செயற்ப்பாடுகளின்றி விரைவாக எடுப்பதாகும். அதற்கு மூன்று தனியாகச் செயல் படுகின்ற படிமுறைகள் பயன்படுகிறது.அவைகள் தகவல் எடுத்தல் படிமுறை, கே சிறப்பியல் படிமுறை, மற்றும் எல் எடுத்தல் படிமுறை ஆகும்.இந்த ஆய்வின் முதற்ப்படி வலைப்பக்கங்களை வெவ்வேறு பகுதிகளாக பிரித்துக்கொண்டு. பின்பு தேவையான தகவல்களை அடையாளம் காட்ட ஒவ்வொரு படிமுறையும் பயன்படுத்தப்படுகிறது. இந்த படிமுறைகளின் செயல்படுகளை கணக்கிட துல்லியம் மற்றும் மீண்டும் அழை மதிப்புக்கள் பயன்படுகிறது.இந்த ஆய்வு கைப்பேசியில் இனணயம் பயன்படுத்துவதற்கும், உரை மீள்தருகை மற்றும் எழுத்துக்களை சுருக்குவதற்கும் பயன்படுகிறது.

ACKNOWLEDGEMENT

I express my profound gratitude to our esteemed Chairman **Arutselvar Dr.N.Mahalingam, B.Sc., F.I.F.,** and Vice Chairman **Prof. Dr.K.Arumugam, B.E. (Hons), MS (USA), M.I.E.,** for giving this great opportunity to pursue this course.

I thank **Dr.Joseph V.Thanikal, M.E., Ph.D., PDF, CEPIT** Principal and **Prof. R.Annamalai** Vice Principal Kumaraguru College of Technology, Coimbatore, for providing me with the necessary facilities and Infrastructure to work on this project.

I thank **Dr.S.Thangasamy, Ph.D.,** Professor and Dean, Department of Computer Science and Engineering for being my greatest of inspiration and for embedding the quest for innovative ideas.

I express my deep sense of gratitude and gratefulness to Project Coordinator **Mrs.V.Vanitha, M.E.,** Asst. Professor, Department of Computer Science and Engineering, for her supervision, tremendous patience, active involvement and guidance.

I express my sincere thanks to the Guide **Mr.E.A.Vimal M.E.,** Lecturer, Department of Information Technology, for his help and valuable support in the progress of the project.

I would like to convey my honest thanks to all Teaching staff members and Non Teaching staff members of the department for their support. I would like to thank all my classmates who gave me proper light moments and study breaks apart from extending some technical support whenever I needed them most.

My work will be incomplete without expressing my gratitude to various authors whose works were referred to carry out this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SGML	Standard Generalized Markup Language
HTML	Hyper Text Markup Language
XHTML	Extensible Hyper Text Markup Language
XML	Extensible Markup Language
DOM	Document Object Model
OMG	Object Management Group
IDL	Interface Definition Language
VIPS	Vision Based Page Segmentation
OEM	Object Exchange Model
HTTP	Hyper Text Transfer Protocol
MOBIE	Mosaic Based Information Explorer
UFRE	Uniform Free Regular Expression
FST	Finite State Transducer
L-P	Lifetime Personalization
CORBA	Common Request Broker Architecture
W3C	World Wide Web Consortium

CHAPTER 1

INTRODUCTION

1.1 Internet

There is an exponential increase in the amount of data available on the web recently. The number of pages available on the web is around 1 billion with almost another 1.5 million are being added daily. This enormous amount of data in addition to the interactive and content-rich nature of the web has made it very popular. However, these pages vary to a great extent in both the information content and quality. Moreover, the organization of these pages does not allow for easy search. So an efficient and accurate method for identifying this huge amount of data is very essential if the web is to be exploited to its full potential. This has been felt for a long time and many approaches have been tried to solve this problem.

Search engines crawl the World Wide Web to collect Web pages. These pages are either readily accessible without any activated account or they are restricted by username and password. Whatever be the way the crawlers access these pages, they are (in almost all cases) cached locally and indexed by the search engines.

An end-user who performs a search using a search engine is interested in the primary informative content of these Web pages. However, a substantial part of these Web pages especially those that are created dynamically is content that should not be classified as the primary informative content of the Web page. These blocks are seldom sought by the users of the Web site. We refer to such blocks as non content blocks. Non content blocks are very common in dynamically generated Web pages. Typically, such blocks contain advertisements, image-maps, plug-ins, logos, counters, search boxes, category information, navigational links, related links, footers and headers, and copyright information. In this project, we address the problem of identifying the primary informative content of a Web page as shown in figure 1.1.

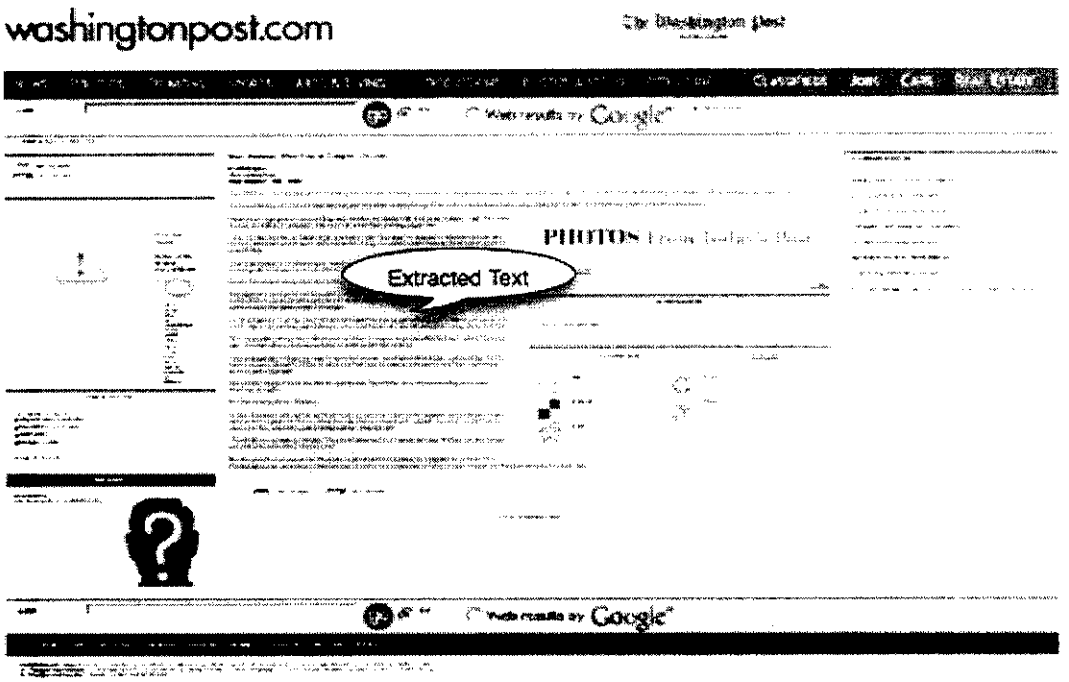


Figure 1.1 Extraction of text from a web page

1.2. Web Mining

Web mining is moving the World Wide Web toward a more useful environment in which users can quickly and easily find the information they need. It includes the discovery and analysis of data, documents, and multimedia from the World Wide Web. Web mining uses document content, hyperlink structure, and usage statistics to assist users in meeting their information needs.

The Web itself and search engines contain relationship information about documents. Web mining is the discovery of these relationships and is accomplished within three sometimes overlapping areas. Content mining is first. Search engines define content by keywords. Finding contents' keywords and finding the relationship between a Web page's content and a user's query content is content mining. Hyperlinks provide information about other documents on the Web thought to be important to another

document. These links add depth to the document, providing the multi-dimensionality that characterizes the Web. Mining this link structure is the second area of Web mining.

Finally, there is a relationship to other documents on the Web that are identified by previous searches. These relationships are recorded in logs of searches and accesses. Mining these logs is the third area of Web mining. Understanding the user is also an important part of Web mining. Analysis of the user's previous sessions, preferred display of information, and expressed preferences may influence the Web pages returned in response to a query. Web mining is interdisciplinary in nature, spanning across such fields as information retrieval, natural language processing, information extraction, machine learning, database, data mining, data warehousing, user interface design, and visualization. Techniques for mining the Web have practical application in m-commerce, e-commerce, e-government, e-learning, organizational learning, virtual organizations, knowledge management, and digital libraries.

1.3 Document Object Model (DOM)

The Document Object Model (DOM) is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents. Objects under the DOM (also sometimes called "Elements") may be specified and addressed according to the syntax and rules of the programming language used to manipulate them. The rules for programming and interacting with the DOM are specified in the DOM Application Programming Interface (API). With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model, with a few exceptions - in particular, the DOM interfaces for the XML internal and external subsets have not yet been specified.

As a W3C specification, one important objective for the Document Object Model is to provide a standard programming interface that can be used in a wide variety of

environments and applications. The DOM is designed to be used with any programming language. In order to provide a precise, language-independent specification of the DOM interfaces, we have chosen to define the specifications in Object Management Group (OMG) IDL [OMGIDL], as defined in the CORBA 2.3.1 specification [CORBA]. In addition to the OMG IDL specification, we provide language bindings for Java [Java] and ECMAScript [ECMAScript] (an industry-standard scripting language based on JavaScript [JavaScript] and JScript [JScript]).

The DOM is a programming API for documents. It is based on an object structure that closely resembles the structure of the documents it models. For instance, consider this table, taken from an HTML document:

```
<TABLE>
<TBODY>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</TBODY>
</TABLE>
```

A graphical representation of the DOM of the example table is:

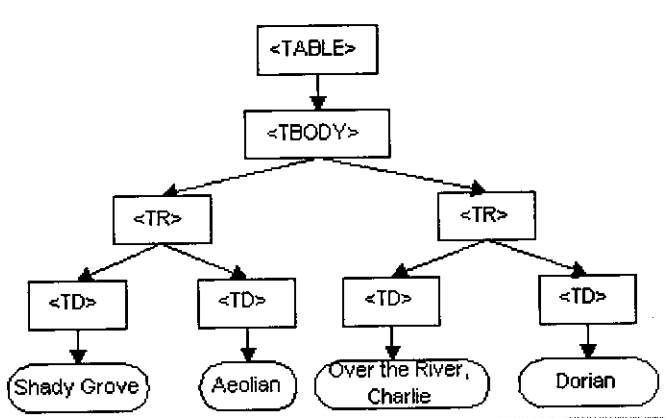


Figure 1.2 Graphical representation of the DOM of the example table

In the DOM, documents have a logical structure which is very much like a tree; to be more precise, which is like a "forest" or "grove", which can contain more than one tree. Each document contains zero or one doctype nodes, one root element node, and zero or more comments or processing instructions; the root element serves as the root of the element tree for the document. However, the DOM does not specify that documents must be implemented as a tree or a grove, nor does it specify how the relationships among objects be implemented. The DOM is a logical model that may be implemented in any convenient manner. In this specification, we use the term structure model to describe the tree-like representation of a document. We also use the term "tree" when referring to the arrangement of those information items which can be reached by using "tree-walking" methods; (this does not include attributes). One important property of DOM structure models is structural isomorphism: if any two Document Object Model implementations are used to create a representation of the same document, they will create the same structure model.

There may be some variations depending on the parser being used to build the DOM. For instance, the DOM may not contain whitespaces in element content if the parser discards them.

The structure of SGML documents has traditionally been represented by an abstract data model, not by an object model. In an abstract data model, the model is centered around the data. In object oriented programming languages, the data itself is encapsulated in objects that hide the data, protecting it from direct external manipulation. The functions associated with these objects determine how the objects may be manipulated, and they are part of the object model.

CHAPTER 2

LITERATURE REVIEW

2.1 Template Detection via Data Mining and its Applications [2]:

Template detection is done based on counting frequent items set from collection of templates. The frequent item set refers to the area of search and mining of web data. This method partition web page into various Pagelets. A pagelet is region of a web page that

1. has a single well defined topic or functionality ; and
2. is not nested within another region that has exactly the same topic or functionality.

All the HTML tags are defined as Pagelets. This approach suffers from several caveats:

1. the HTML structure is a tree structure and we would like to Flattened partitioning of the page
2. the granularity of this partitioning is too fine

A template is a collections of pages that

1. Share the same look and feel
2. Are controlled by a single authority

Templates are usually created by a master HTML page that is shared by all the pages that belongs to the site of the authority that controls the template.
The two algorithms are used to detect templates from pages such as:

1. algorithm that fits the smaller document sets
2. algorithm that fits the larger document sets

The main benefit of template elimination is that it improves the precision of search engine at all levels of recall. But this approach leads to some disadvantages such as:

1. Cannot identify the primary contents from a web pages
2. Hard to use template as a indicator of authority of web pages

2.2 Block-Based Web Search [5] :

The main purpose of this research is that the web pages are partitioned into different blocks rather than as a various templates. In this research four types of web segmentation approaches such as given below are compared:

1. Fixed length page segmentation
2. DOM based page segmentation
3. Vision based page segmentation
4. Combined approach

The Fixed length page segmentation (FixedPS) approach partition the web pages according to the fixed length passage that contains fixed number of continuous words. But the major drawback of this approach is that no semantic information is taken into account during segmentation process.

The DOM based page segmentation (DOMPS) approach partition the web pages based on the content and presentation of the page as well. The segmentation process is carried out based on the HTML tags such as <TABLE>, <TITLE>, <P>, <H1>, etc. But DOM is a linear structure so visually adjacent blocks may be far from each others in the structure and departed wrongly, this leads to the huge drawback of DOMPS approach.

The Vision based segmentation (VIPS) approach is based on the people view the web pages and get 2-D presentation which helps to distinguish different parts of the pages such as lines, blanks, images , colors etc. The structural tags such as <TABLE> and <P>

are divided with the help of visual information. VIPS can achieve better content structure for the original web pages. But VIPS do not consider the document length normalization problems.

The combined approach will integrates both fixed length page segmentation and vision based segmentation to achieve the greater retrieval performance by dealing with the multiple topics and mixed length problems of web pages. This technique can be applied to the data set close to the web scale and also for the query expansion.

2.3 The TSIMMIS Project: Integration of Heterogeneous Information Sources [3]:

The Tsimmis project is a joint project between Stanford and the IBM Almaden Research Center. The goal of this Tsimmis project is to develop tools that facilitate the rapid integration of heterogeneous information sources that may include both structured and unstructured data. This research is to extract properties from unstructured objects that translate information into a common object model, that combine information from several sources, which allow browsing information and that manage constraints across heterogeneous sites.

The Tsimmis project will provide a tool for accessing, in an integrated fashion, multiple information sources, and to ensure that the information obtained is consistent. Object Exchange Model (OEM) is used as the unifying object model for information processed. To request OEM objects from an information source, a client issues a queries in a language called OEM-QL. This language will adapts existing SQL-like languages for object-oriented models to OEM.

MOBIE (Mosaic Based Information Explorer), a graphical browsing tool is used in this project which is based on Mosaic and the World-Wide-Web for submitting Tsimmis queries and exploring the results. MOBIE lets end users connect to mediators or translators and specify queries using OEM-QL.

An important advantage of using Mosaic as the basis for user interface is its widespread use and popularity. But the OEM model does not provide the right flexibility for handling unexpected heterogeneity always and this system depend on manually provided grammar rules.

2.4 Towards Automatic Data Extraction from Large web Sites [1]:

In this research a subclass of regular expression grammar i.e. Uniform Free Regular Expression (UFRE) is used to identify the extraction rules by comparing web pages of the same class and by finding similarities or dissimilarities among them.

This research investigates techniques for extracting data from HTML sites through the use of automatically generated wrappers. To automate the wrapper generation and the data extraction process, the novel technique has been developed. This novel technique compares HTML pages and generates a wrapper based on their similarities and differences.

ACME (Align, Collapse under Mismatch, and Extract) matching technique is used to solve the mismatches between the wrapper and the samples. The matching algorithm works on two objects at a time:

1. a list of tokens, called the samples and
2. a wrapper, i.e., one uniform-free regular expression.

A mismatch happens when some token in the sample does not comply with the grammar specified by the wrapper. Mismatches are very important, since they help to discover essential information about the wrapper.

Whenever one mismatch is found, the system to solve the mismatch by generalizing the wrapper .The algorithm succeeds if a common wrapper can be generated by solving all mismatches encountered during the parsing.

In some of the cases UFREs are not sufficiently expressive to wrap the pages. This may happen either because the samples in a class form a non-regular language, or because they form a regular language which requires the use of unions to be described.

2.5 Initial results on wrapping semi structured web pages with finite-state transducers and contextual rules [4]:

The main aim of this research is try to extract information by identifying and exploiting the templates. A novel web-wrapper representation formalism has been developed during this research.

A novel web wrapper representation is based on a finite-state transducer (FST) and context rules which allow a wrapper to wrap semi structured web-pages containing missing attributes, multiple attribute values, variant attribute
permutations, exceptions and types. This wrapper can be learnt from labeled example items using a simple induction algorithm.

2.6 Automatic Detection of Fragments in Dynamically Generated Web Pages [6]:

Dividing web pages into fragments has been shown to provide significant benefits for both content generation and caching. This research proposes a novel scheme to automatically detect and flag fragments that are cost effective cache units in web sites serving dynamic content. A fragment is considered to be interesting if it is shared among multiple pages or if it has distinct lifetime or personalization characteristics. The two independent fragment detection algorithms are used:

1. one for detecting Shared fragments
2. another for detecting Lifetime Personalization based(L-P) fragments

Both these algorithms are collocated with a server-side cache or an edge cache, and work on the dynamic web page dumps from the web sites.

The algorithm for detecting shared fragments works on a collection of different dynamic pages generated from the same web site, whereas L-P fragment detection algorithm works on different versions of each web page, which cane be obtained from a single query being repeatedly submitted to the give web sites.

Shingling encoding scheme is used to identify fragments of web pages and use it to show that the storage requirements of we-caching are significantly reduced. The cost of implementing shingling algorithm is more expensive.

2.7 CoreEx: Content Extraction from Online News Articles [9]:

Uses DOM tree to segment the web pages and for every nodes in the DOM tree text count and link counts are maintained. The text count will be the sum of the non linked words. The link count is the sum of links.

Once the text and link counts are known for each node, the nodes are scored. The DOM node with the highest score is picked as the Main Content Node. The DOM sub tree rooted at this node contains the content. The question then is how to compute a score that leads to good choices for core content.

This basic algorithm has two significant drawbacks. First, it favors small nodes with no links over larger nodes with a few links. This behavior is undesirable since the latter is more likely to contain the main content . Basic scoring would pick one of the smaller nodes, which have a score of 1.0 instead of the entire root node, since the score of the root node is reduced by the single link.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1. Hardware Specification

PROCESSOR	Intel Pentium-IV
PROCESSOR CLOCK SPEED	1.80 GHz
RAM	1 GB
MAIN BOARD CHIPSET	INTEL810E CHIPSET
HARD DISK	40 GB

Table 3.1 Hardware Specification

3.2. Software Specification

OPERATING SYSTEM	MICROSOFT WINDOWS XP
SEVICE PACK	2
LANGUAGE	JDK 1.6
PROXY SERVER	TOMCAT 6.0.18
PARSER	HTML PARSER 1.6
EDITOR	ECLIPSE PLATFORM 3.4.1

Table 3.2 Software Specification

CHAPTER 4

MODULE DESCRIPTIONS AND IMPLEMENTAION

4.1. Segmenting web pages into blocks

Our algorithm uses <TABLE> as the first tag on the basis of which it partitions a Web page. After <TABLE>, it uses <TR>, <P>, <HR>, <UL>, <DIV>, and <SPAN>, etc., as the next few partitioning tags in that order. We selected the order of the tags based on our observations of Web pages and believe that it is a natural order used by most Web page designers. For example, <TABLE> comes as a first partitioning tag since we see more instances of <UL> in a table cell than <TABLE>s coming inside <LI>, an item under <UL>. Our algorithms partition a Web page based on the first tag in the list to identify the blocks, and then sub partitions the identified blocks based on the second tag and so on. It continues to partition until there is any tag left in a block in the block-set which is part of the list of tags. This ensures that the blocks are atomic in nature and no further division is possible on them.

The GetBlockSet routine takes an HTML page as input with the ordered tag-set. GetBlockSet takes a tag from the tag-set one by one and calls the GetBlocks routine for each block belonging to the set of blocks, already generated. New sub blocks created by GetBlocks are added to the block set and the generating main block is removed from the set.

The First function gives the first element (tag) of an ordered set, and the Next function gives the consecutive elements (tags) of an ordered set. GetBlocks takes a full document or apart of a document, written in HTML, and a tag as its input. It partitions the document into blocks according to the input tag as shown in the figure 4.1.

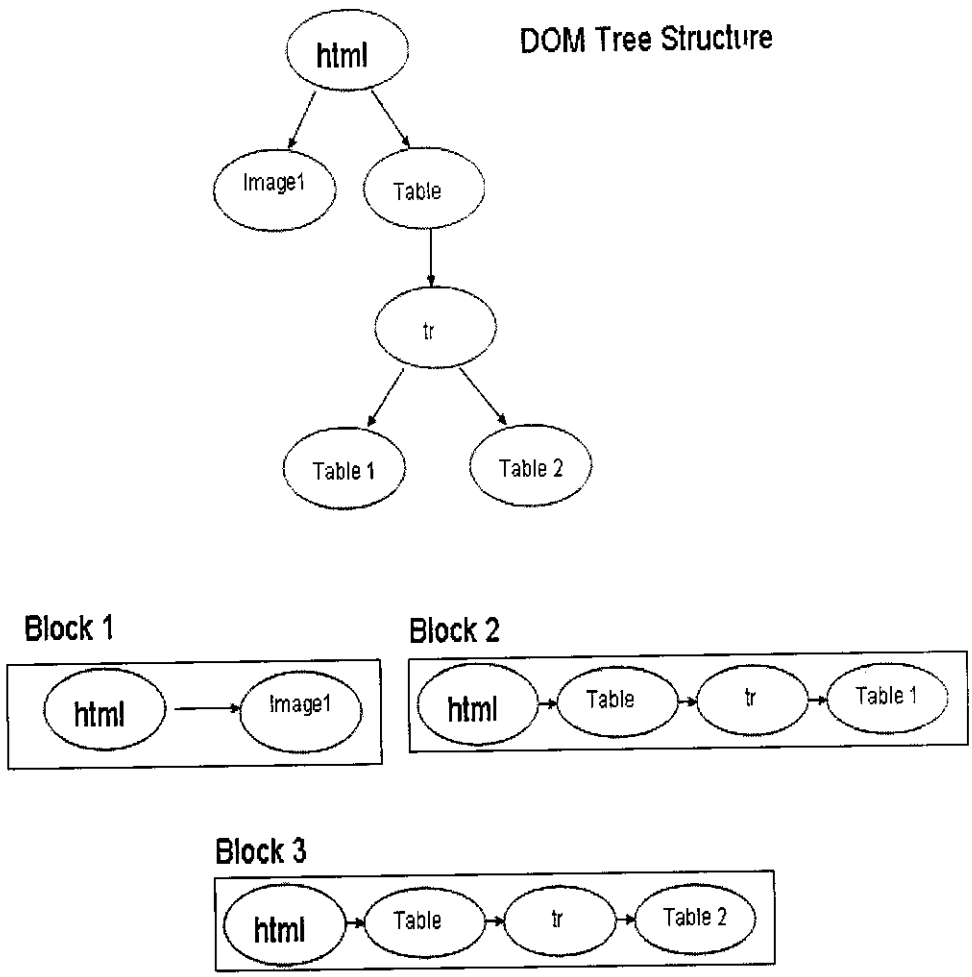


Figure 4.1 Web Page Segmentation

4.2. Implementation of Content Extraction Algorithm

The input to the algorithms is a set (at least two) of Web pages belonging to a class of Web pages. A class is defined as a set of Web pages from the same Web site whose designs or structural contents are very similar. A set of Web pages dynamically generated from the same script is an example of a class. The outputs of the algorithms are the primary content blocks in the given class of Web pages.

The content extraction algorithm uses GetBlockSet routine that takes an HTML page as input with the ordered tag-set. GetBlockSet takes a tag from the tag-set one by one and calls the GetBlocks routine for each block belonging to the set of blocks, already generated. Similarly First and Next functions are used to retrieve first block and consecutive blocks from the routine.

The blocks are given as a input to the HashSet Class and this function will remove the duplicate contents from the block set as shown in the figure 4.2. Generally HashSet will uses a hash table for storage by mechanism called hashing. In hashing, the content of a key is used to determine a unique value called hash code. The advantage of using hashing is that it allows the execution time of basic operations such as add() and remove(), to remain constant for a large sets.

The sim() function is the similarity measure function, to find the similarity between the two blocks. Given two blocks, Sim returns the cosine between their block feature vectors. We used a threshold value of ¼ 0:9. That is, if the similarity measure is greater than the threshold value, then the two blocks are accepted as identical. The threshold value can be changed according to the needs of the application and affects the precision and recall of the algorithm

Content Extraction Algorithm

1. Input to the algorithm are Set S of HTML pages, and Sorted tag-set T
2. GetBlockSet () is called to retrieve blocks from block set.
3. HashSet() is used to remove duplicates
4. Output is the primary contents without duplicates
5. Primary contents are stored in temporary location

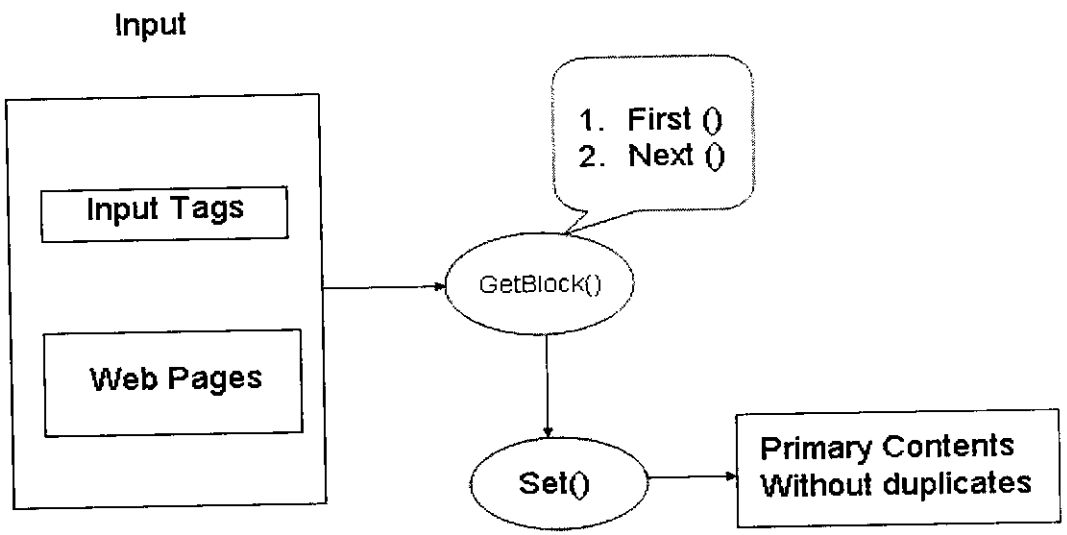


Figure 4.2 Content Extraction operations

4.3. Implementation of K-Feature extraction algorithm

The feature extraction algorithm helps to identify any informative block corresponding to any features such as text, images, links, image blocks, navigational blocks etc. In FeatureExtraction we are looking for all the text-blocks and so we need to compare the properties of a block against other blocks. This comparison is only possible if we consider all the HTML tags as the feature-set of the blocks.

We can update this list if we so desire because of changes in HTML features or because of an application's updated preferences of desirable features easily without fundamentally changing the algorithm. Unlike the ContentExtraction algorithm, the FeatureExtraction algorithm does not depend on multiple Web-pages but depends on the feature-set and the chosen feature for output.

Algorithm calculates values for each feature in each block. After calculating feature value, the blocks whose sum of desired feature value is greater than the sum of other feature values in the temporary location i.e. Winner Basket. The block with the highest value of desired value from winner basket is selected as a primary content block of Web page.

The feature value for a given feature will be calculated as ratio between Total number of features to the maximum value for the particular feature.

$$\text{Feature Value} = \frac{\text{Total Number of Features}}{\text{Maximum Number of Feature}}$$

Though FeatureExtraction performs with high precision and recall for one of our data sets, it may not do so in general and can be improved. For Web pages with multiple important text blocks, a typical reader may be interested in all the sections not just one of them (winner of FeatureExtraction). For example, an end-user may be interested in all the reviews available from a page on Amazon.com and each review is in a separate block. General shopping sites, review sites, chat forums, etc.,may all contain multiple blocks of important textual information. FeatureExtractor shows poor precision and recall as it produces only one text-block with highest probability, while other important blocks are not retrieved.

To overcome this drawback, we revised the last part of the FeatureExtraction and named the new algorithm as K-Feature Extraction. To handle more general Web pages of varied editing-styles, we improved the FeatureExtraction algorithm. Instead of taking just the winner block from the winner-basket, we apply a K-means clustering algorithm to select the best probability blocks from the basket. winner. The usual values of k taken are 2 or 3, and the initial centroids are chosen from the sorted list at equidistant index values. After the clustering is done, the high probability cluster(s) are taken and the corresponding text contents of all those blocks are taken as the output. This helps us get

high precision and recall from shopping Web sites and review Web sites and, in general, a much broader range of Web sites.

The K-means algorithm is an algorithm to cluster objects based on attributes into k partitions. It is a variant of the expectation-maximization algorithm in which the goal is to determine the k means of data generated from gaussian distributions. It assumes that the object attributes form a vector space. The objective it tries to achieve is to minimize total intra-cluster variance, or, the squared error function

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} |x_j - \mu_i|^2$$

where there are k clusters S_i, $i = 1,2,...,k$ and μ_i is the centroid or mean point of all the points $x_j \in S_i$.

K-Means clustering is an algorithm to classify or to group your objects based on attributes/features, into K number of groups. K is a positive integer number. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. Thus, the purpose of K-means clustering is to classify the data.

The algorithm starts by partitioning the input points into k initial sets, either at random or using some heuristic data. It then calculates the mean point, or centroid, of each set. It constructs a new partition by associating each point with the closest centroid. Then the centroids are recalculated for the new clusters, and algorithm repeated by alternate application of these two steps until convergence, which is obtained when the points no longer switch clusters (or alternatively centroids are no longer changed) as shown in the figure 4.3.

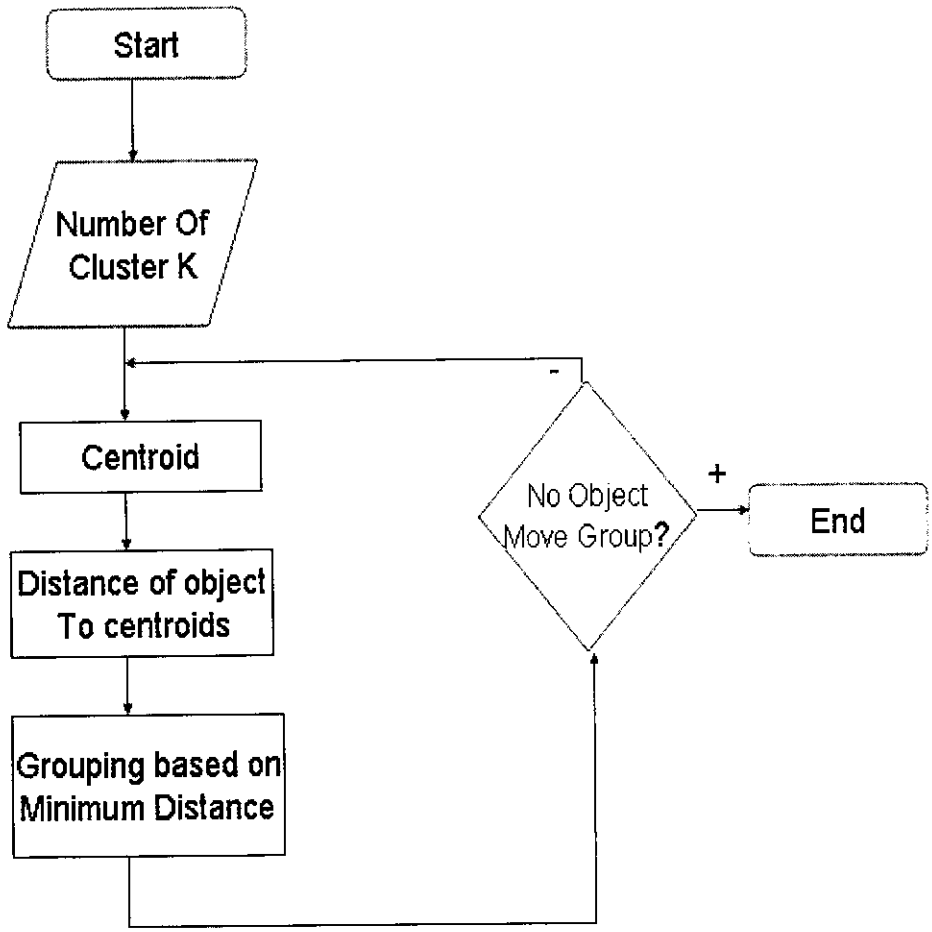


Figure 4.3.1 Functionality of k-means clustering

Algorithm:

1. Input to the algorithm are Set of HTML page H, Sorted Tag Set T, and Desired Feature F1
2. GetBlockSet () is called to retrieve blocks from block set.
3. Calculate the Feature value for each feature in the block.
4. If sum of desired feature value > sum of other feature value
    - Put the block in winner basket
5. Recompute the feature values for the block in the winner basket

6. Choose the block with the highest probability value as a primary content block for web page.

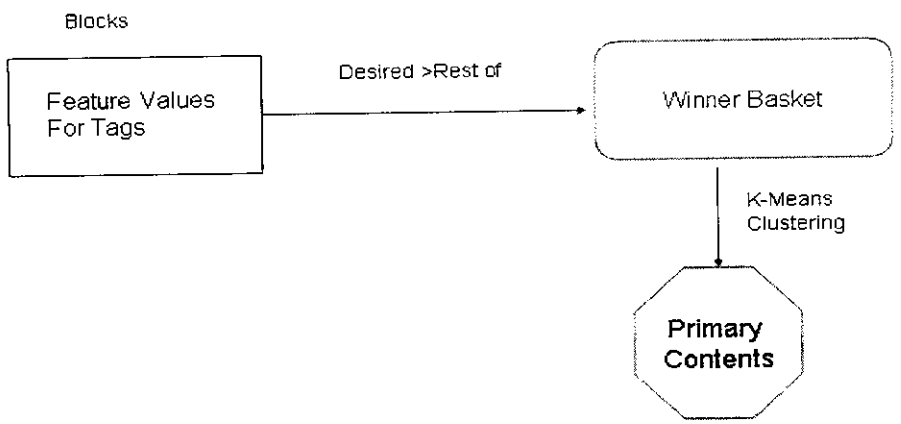


Figure 4.3.2 K-Feature Extraction Operation

4.4. Implementation of L-Extraction Algorithm

The main drawback of the K-Means algorithm is that it has to be told the number of clusters (i.e. *k*) to find. If the data is not naturally clustered, we get some strange results. Also, the algorithm works well only when spherical clusters are naturally available in data.

The purpose of machine learning algorithms is to use observations (experiences, data, patterns) to improve a performance element, which determines how the agent reacts when it is given particular inputs. The performance element may be a simple classifier

trying to classify an input instance into a set of categories or it may be a complete agent acting in an unknown environment. By receiving feedback on the performance, the learning algorithm adapts the performance element to enhance its capabilities.

Dependening on the feedback we can distinguish between the following forms of learning: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the learning algorithms receives inputs and the correct outputs, and searches for a function which approximates the unknown target function. In unsupervised learning, the agent receives only input data and uses an objective function (such as a distance function) to extract clusters in the input data or particular features which are useful for describing the data. In reinforcement learning, the agent receives an input and an evaluation (reward) of the action selected by the agent, and the learning algorithm has to learn a policy which maps inputs to actions resulting in the best performance.

C5 Machine Learning algorithm will provide a decision automatically based on the input i.e., primary content of web pages. Based on the decision given by the algorithm the output of the content are stored at the temporary location as shown in the figure 4.4

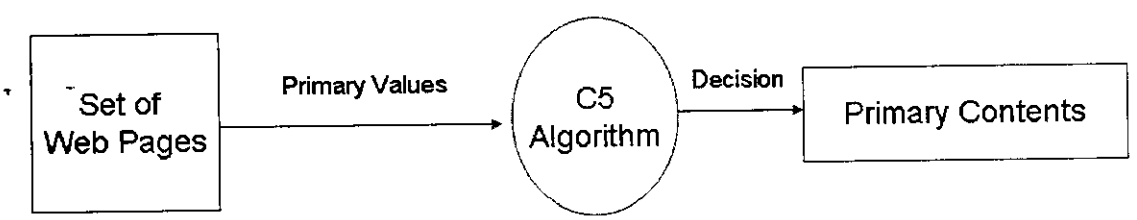


Figure 4.4 Operations in L-Extraction

4.5. Performance Comparison

Algorithms are compared in terms of b-Precision and b-Recall. Precision is defined as the ratio of the number of relevant items (actual primary content blocks) r found and the total number of items (primary content blocks suggested by an algorithm) t found. Here, we used a block level precision and so we call it b-Precision:

$$\text{b-Precision} = r/t$$

Recall has been defined as the ratio of the number of relevant items found and the desired number of relevant items. The desired number of relevant items includes the number of relevant items found and the missed relevant items m. In case of blocks, we call it as block level recall or

$$\text{b-Recall} = r/r+m$$

Similar to the way it is defned in information retrieval literature by Valter Crescenzi [4], we can refer to the F-measure here as the b-F-measure and defne it as

$$\text{b - F - measure} = \frac{2 * (\text{b - Precision}) * (\text{b - Recall})}{(\text{b - Precision}) + (\text{b - Recall})}$$

CHAPTER 5

EXPERIMENTAL RESULTS

The goal of this performance evaluation is to compare the performance of Content Extraction , K- Feature Extraction and L- Extraction algorithm with the existing algorithm such as LH algorithm and Shingling algorithm based on the metrics as follows:

- B - Precision
- B – Recall
- BF Measure

5.1 First Comparison : With LH algorithm

Our algorithms outperform LH in all news sites in all categories. The b-recall is always good since all algorithms could find most relevant blocks but the results obtained by running the LH algorithm were less precise than those obtained by ContentExtractor since the former algorithm also includes lots of other non-content blocks.

We believe that the primary reason for the poor b-precision of LH is because of the greedy approach taken by their algorithm while identifying the solution. A second reason is that the LH algorithm works at the feature level instead of the block level. LH gives high redundancy score to features that occur across Web-pages. The redundancy score of a block is proportional to the weighted sum of the redundancy scores of each feature it contains. Instead of looking at occurrences of features across Web-pages, the ContentExtractor algorithm looks at occurrences of similar blocks across pages. This fundamental difference results in better b-precision obtained by our algorithm.

The FeatureExtractor algorithm only works well on Web-pages where the primary Web-pages have one dominant feature. For example, in news Web pages, text is the dominant feature. However, if we go to a domain where the primary content is a mix of

multiple features, Feature- Extractor's b-precision suffers. If FeatureExtractor has to be deployed in such a domain, it must be modified to handle multiple features and use a weighted measure of the presence of multiple features to identify the primary content pages. Due to the dependence of FeatureExtractor on one particular feature, we expect it to perform poorer than ContentExtractor in more general cases where the dominant features in a primary content block are not known.

Property	**LH Algorithm**	**Content Extraction(CE)**	**K- Feature Extraction(KE)**	**L- Extraction (LE)**
B - Precision	Very High	High	Low	Very Low
B – Recall	Very High	High	Low	Low
Number of pages needed	All pages to calculate entropy of features	5 – 10 pages	Single HTML page	2- more HTML pages
Time of Completion	Always more than CE	Less than LH	Less than CE	Even less than KE

Table 5.1 Property Wise Comparison Table with LH Algorithm

5.2 Second Comparison : With Shingling Algorithm

In this section, we compare one of our algorithms with the Shingling algorithm proposed by Ramaswamy et.al. [6]. Regarding the way this algorithm is designed, it is the closest to our ContentExtractor algorithm, and therefore we will attempt to compare these two algorithms side-by-side. They also partition the HTML page into blocks (in their case they call them the nodes of the AF or Augmented Fragment tree). Then they characterize each individual node with different properties, such as SubtreeValue, SubtreeSize, SubtreeShingles and others.

Property	Shingling Algorithm	Content Extraction
Atomic Structure	AF Tree Node	Block
Atomic Property	Complex and expensive measurement of (N-W+1) node-ids	Simple HTML tag feature, which is perfect for measuring similarity: Inexpensive
Precision and Recall	Similar to Content Extraction	Similar to Shingling
Speed	Slow	Much Faster
Disk Space Requirements (Figure 5.1)	Similar to Content Extraction	Similar to Shingling

Table 5.2 Property Wise Comparison Table with Shingling Algorithm

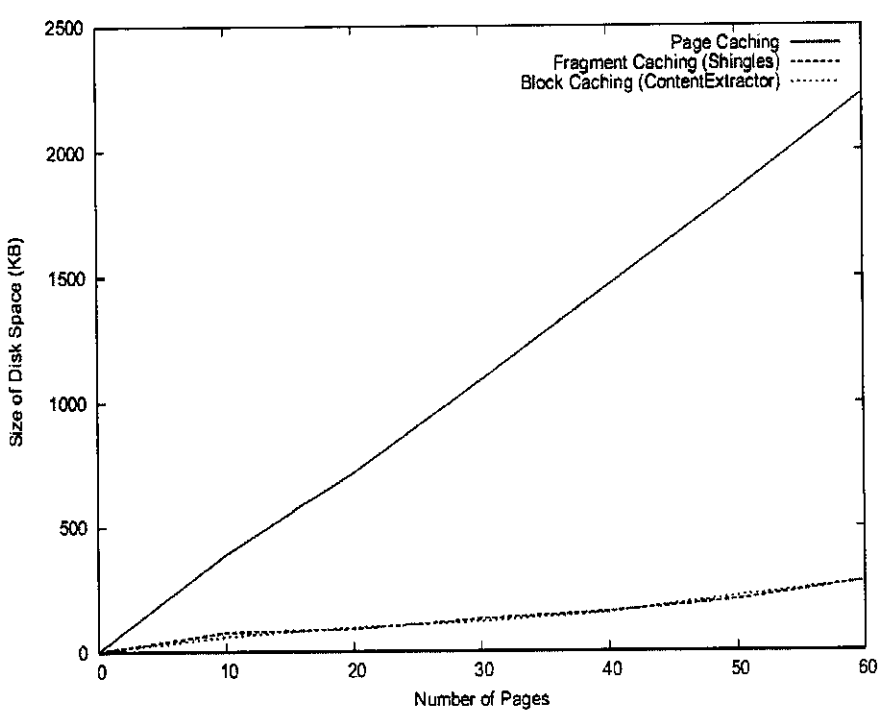


Figure 5.1 Total storage requirement of Web-caches using Content Extraction and Shingling algorithms.

CHAPTER 6

FEASIBILITY

FEASIBILIY STUDY

A feasibility study is concerned to select the best system that meets performance requirements. These entities are an identification description, an evaluation of candidate systems and the selection of the best system for the job.

- Economic feasibility
- Technical feasibility
- Behavioural feasibility

6.1. Economic feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of the candidate system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that benefits outweigh costs, and then the decision is made to design and implement the system. Otherwise, further justification or alterations in the proposed system will have to be made if it is to have an enhancement to approve.

6.2. Technical feasibility

Technical analysis centre on the existing computer system (Hardware, Software etc) and to what extend it can support the proposed addition. This involves financial considerations to accommodate technical enhancement. If the budget is a serious constraint, then the project is judged not feasible.

6.3. Behavioural Feasibility

An estimate should me made of how strong a reaction the user staff is likely to have toward the development of a computerized system. It is common Knowledge the computer installations have something to do understandable that the introduction of a candidate system requires special effort to educate, sell and train the Staff on new ways of considering business.

CHAPTER 7

SYSTEM IMPLEMENTATION

SYSTEM IMPLEMENTATION

System implementation is the important stage of project when the theoretical design is tuned into practical system. The main stages in the implementation are as follows:

- Planning
- Training
- System testing and
- Changeover Planning

7.1. Planning

Planning is the first task in the system implementation. Planning means deciding on the method and the time scale to be adopted. At the time of implementation of any system people from different departments and system analysis involve. They are confirmed to practical problem of controlling various activities of people outside their own data processing departments. The line managers controlled through an implementation coordinating committee. The committee considers ideas, problems and complaints of user department, it must also consider:

- The implication of system environment;
- Self selection and allocation for implementation tasks;
- Consultation with unions and resources available;
- Standby facilities and channels of communication

7.2. Training

To achieve the objectives and benefits from computer based system, it is essential for the people who will be involved to be confident of their role in new system. This involves them in understanding overall system and its effect on the organization and in being able to carry out effectively their specified task. So training must take place at an

early stage. Training sessions must give user staff, the skills required in their new jobs. The attendance to sort out any queries.

7.3. Testing

7.3.1. Unit testing

A program represents the logical elements of a system. For a program to run satisfactorily, it must compile and test data correctly and tie in properly with other programs. Achieving an error free program is the responsibility of the programmer. Program testing checks for two types of errors: syntax and logical. Syntax error is a program statement that violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax errors. These errors are shown through error message generated by the computer. For Logic errors the programmer must examine the output carefully.

When a program is tested, the actual output is compared with the expected output. When there is a discrepancy the sequence of instructions must be traced to determine the problem. The process is facilitated by breaking the program into self-contained portions, each of which can be checked at certain key points .The idea is to compare program values against desk-calculated values to isolate the problems.

Test case no	Description	Expected result
1	Test for application window properties	All the properties of the windows are to be properly aligned and displayed

Table 7.1 Unit Testing

7.3.2. Performance testing

Testcase no	Description	Expected result
1	This is required to assure that an application perforce adequately, having the capability to handle many identification of web pages, delivering its results in expected time and using an acceptable level of resource and it is an aspect of operational management.	Should handle large input values, and produce accurate result in a expected time

Table 7.2 Performance Testing

7.3.3. White box testing

White box testing, sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases. Using white box testing method, the software engineer can derive test cases.

Testcase no	Description	Expected result
1	Exercise all logical decisions on their true and false sides	All the logical decisions must be valid
2	Execute all loops at their boundaries and within their operational bounds.	All the loops must be finite
3	Exercise internal data structures to ensure their validity.	All the data structures must be valid

Table 7.3 White Box Testing

7.3.4. Black box testing

Black box testing, also called behavioral testing, focuses on the functional requirements of the software. That is, black testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black box testing is not alternative to white box techniques. Rather it is a complementary approach that is likely to uncover a different class of errors than white box methods. Black box testing attempts to find errors in the following categories.

Test case no	Description	Expected result
1	To check for incorrect or missing functions	All the functions must be valid
2	To check for interface errors	All the interface must function normally
3	To check for errors in a data structures or external data base access.	The database updation and retrieval must be done

Figure 7.4 Black Box Testing

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1. CONCLUSION

We devised simple, yet powerful, and modular algorithms, to identify primary content blocks from Web pages. Our algorithms outperformed the LH algorithm significantly, in b-precision as well as runtime, without the use of any complex learning technique. The K-Feature Extraction algorithm, provided a feature, can identify the primary content block with respect to that feature. The Content Extraction algorithm detects redundant blocks based on the occurrence of the same block across multiple Web pages. The algorithms, thereby, reduce the storage requirements, make indices smaller, and result in faster and more effective searches. Though the savings in file size and the precision and recall values from "Shingling Algorithm" is as good as from Content Extraction, Content Extraction outperforms the "Shingling Algorithm" by a high margin in runtime. We intend to deploy our algorithms as a part of a system that crawls Web pages, and extracts primary content blocks from it.

8.2. SCOPE FOR FUTURE WORK

The Future scope is to look at the primary content and identify heuristic algorithms to identify the semantics of the content to generate markup. The storage requirement for indices, the efficiency of the markup algorithms, and the relevancy measures of documents with respect to keywords in queries should also improve since now only the relevant parts of the documents are considered.

APPENDICES

Appendix – I

Source Code

```
package org.htmlparser;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.PrintStream;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import java.util.StringTokenizer;
import java.util.TreeMap;
import java.util.Vector;
import java.io.*;
import javax.swing.JTable;
import org.htmlparser.util.NodeList;
public class Segmentation
{
        public static Vector vecimg=new Vector(); // vector will grow automatically
        public static Vector vectable=new Vector();
        public static Vector finalimg=new Vector();
        public static String wholestr;
        public static String spacerem;
        public static int linkcount;
        public static Vector vectemp3 =new Vector();
        public static Vector vimg=new Vector();
```

```
public static void main(String input[])
{
        try
        {

        int bs=0;
        int tot=0;
        ArrayList al=new ArrayList();
        Vector finastr=new Vector();
        ArrayList a2=new ArrayList();
        String str9=input[0];
        StringTokenizer stok=new StringTokenizer(str9,"/.");
        while(stok.hasMoreTokens())
        {
                String temp=stok.nextToken();
                a2.add(temp);
                System.out.println(temp);


        }
        System.out.println(a2);
        String lastr=(String)a2.get(a2.size()-2);
        JTable table = new JTable();
        String[] tags={"img","a","string","table","script"};
        PrintStream mycoun=new PrintStream("Blocks/Total_Blocks.txt");
        PrintStream bstring=new PrintStream("Blocks/StringBlocks.txt");
        PrintStream total =new PrintStream("Blocks/total.txt");

        int colCount = table.getColumnCount();
        String[] row = new String[colCount];
```

```
Vector htmltags=new Vector();

htmltags.addElement("<html><body>");
htmltags.addElement("</body></html>");
htmltags.addElement("<script></script>");
htmltags.addElement("<body></body>");
htmltags.addElement("<font color=red>");
htmltags.addElement("</font>");
htmltags.addElement(">");
htmltags.addElement("<");
htmltags.addElement("<p></p>");
htmltags.addElement("<tr><td>");
htmltags.addElement("</td></tr>");
htmltags.addElement("</table></body></html>");
htmltags.addElement("<html><body><table>");
htmltags.addElement("</a>");
String sr="<html><body><script type="+"text/javascript"+">";
htmltags.addElement(sr);
htmltags.addElement("</script></body></html>");

////////////////////////////Start of Image Extraction//////////////////////////


int isize=0;
for(int j=0;j<input.length;j++)
{
        String[] temparr={input[j],"img"};
        Vector vectemp=org.htmlparser.Parser.main(temparr);
        vectemp.removeAll(htmltags);
        System.out.println(vectemp);
        isize=vectemp.size();
```

```
            mycoun.println("Total Number of Image Blocks = "+vectemp.size());
            mycoun.println(" ");
            total.println(" ");
            total.println(vectemp.size());
            System.out.println("Total Number of Image Blocks = "+vimg.size());
            vecimg.addElement(vectemp);
        }

//////////////////////////////////End Of image Extraction//////////////////////////////////////////

//////////////////////////////////Start of Link Extraction//////////////////////////////////////////

        Vector veca=new Vector();
        int lsize=0;
        int vall=0;
        try
        {
            Vector finala=new Vector();
            for(int j=0;j<input.length;j++)
            {
                String[] temparr1={input[j],"a"};
                Vector vectemp3=org.htmlparser.Parser.main(temparr1);
                vectemp3.removeAll(htmltags);
                Set sss=new HashSet(vectemp3);
                Vector vlink=new Vector(sss);
                if(vlink.size()==0)
                {
                }
                else
                {
                    lsize=vlink.size()+1;
```

```
                }
            mycoun.println("Total Number of Link Blocks = "+lsize);
            mycoun.println(" ");
            System.out.println("Total Number of Link Blocks = "+vlink.size());
            total.println(lsize);
            System.out.println(vectemp3);
            veca.addElement(vectemp3);
        }

}
catch(Exception ee)
{
}
////////////////////////////////End of Link Extraction////////////////////////////////

////////////////////////////////Start of String Extraction////////////////////////////

Vector vecstring=new Vector();
int val=0;
try
{
        Vector finalstring=new Vector();
        for(int j=0;j<input.length;j++)
        {
            String[] temparr3={input[j],"string"};
            Vector vectemp3=org.htmlparser.Parser.main(temparr3);
            vectemp3.removeAll(htmltags);
            Vector vimg=new Vector(vectemp3);
            System.out.println("VIMG :"+vimg.size());
            System.out.println("The Strings are :");
            bstring.println("  The String Blocks are:  ");
```

```
        for(int l=0;l<vimg.size();l++)
        {
                String tempsr=vimg.elementAt(l).toString().trim();
                if(!tempsr.equals(""))
                {
                        System.out.println(bs+" = "+vimg.elementAt(l));
                        bstring.println(bs+" = "+vimg.elementAt(l));
                        finastr.addElement(vimg.elementAt(l));
                        bs=bs+1;
                        val=bs;
                }
        }
}

        mycoun.println("Total Number of String Blocks = "+val);
        mycoun.println(" ");
        System.out.println(vectemp3);
        total.println(val);
        vecstring.addElement(vectemp3);
}

catch(Exception ee)
{
}

////////////////////////////////End of String Extraction/////////////////////////////
```

```
////////////////////////////////////Start for Script Extraction/////////////////////////////

Vector vecscript=new Vector();
int isc=0,ax=0;
try
{
        Vector finalscript=new Vector();
        for(int j=0;j<input.length;j++)
        {
                String[] temparr2={input[j],"script"};
                Vector vectemp2=org.htmlparser.Parser.main(temparr2);
                vectemp2.removeAll(htmltags);
                mycoun.println(" Number of Script Blocks = "+vectemp2.size());
                mycoun.println(" ");
                isc=vectemp2.size();
                total.println(vectemp2.size());
                System.out.println(vectemp2);
                vecscript.addElement(vectemp2);
        }

}
catch(Exception ee)
{
}

////////////////////////////////////End of Script Extraction/////////////////////////////
```

```
////////////////////////////////Table Extraction/////////////////////////////////////

Vector vectable=new Vector();
int itab=0;
try
{
        Vector finaltable=new Vector();
        for(int j=0;j<input.length;j++)
        {
                String[] temparr3={input[j],"table"};
                Vector vectemp3=org.htmlparser.Parser.main(temparr3);
                vectemp3.removeAll(htmltags);
                System.out.println(vectemp3);
                System.out.println(vectemp3);
                mycoun.println(" Number of Table Blocks  = "+vectemp3.size());
                mycoun.println(" ");
                total.println(vectemp3.size());
                itab=vectemp3.size();
        for(int n=1;n<vectemp3.size();n++)
        {
        PrintStream pstable=new PrintStream("Blocks/TableBlocks/"+n+".html");
        String tempta=(vectemp3.elementAt(n)).toString();
        pstable.println("<html><body>");
        pstable.println(tempta);
        pstable.println("</body></html>");
        }
        System.out.println(vectemp3);
        vectable.addElement(vectemp3);
        }
}
```

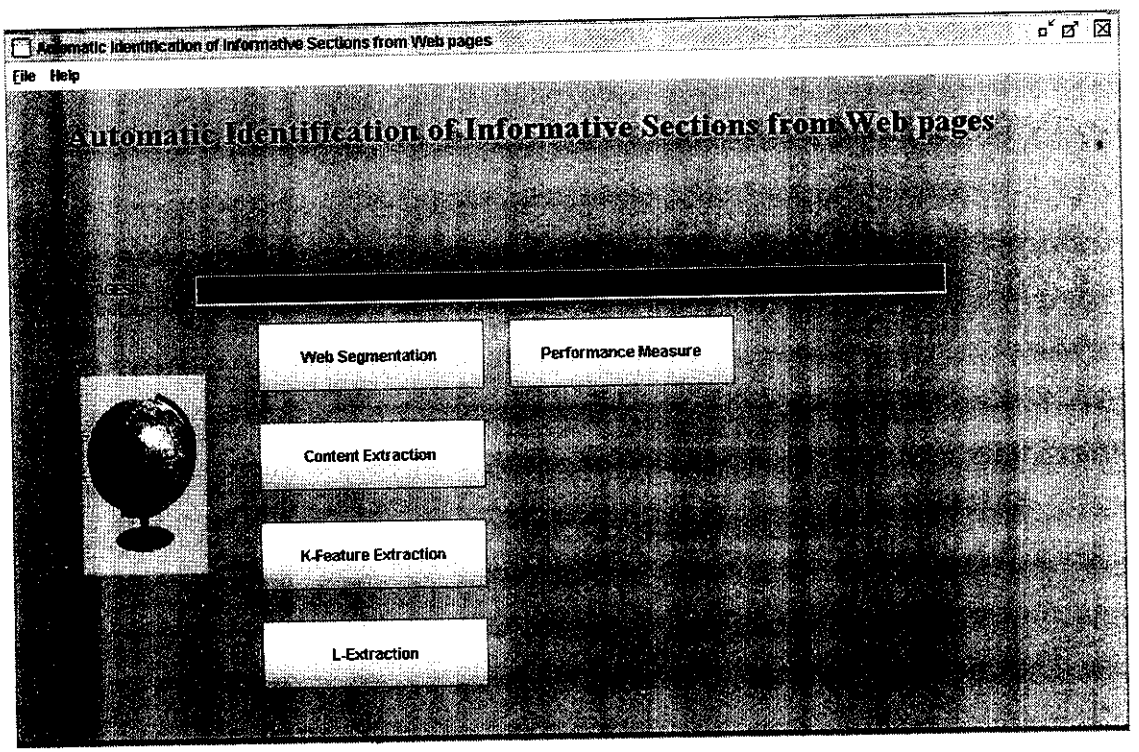
```
catch(Exception ee)    { }

FileReader fr=new FileReader("Blocks/Total_Blocks.txt");
BufferedReader br=new BufferedReader(fr);
String s;
System.out.println("");
while((s=br.readLine())!=null)
{
        System.out.println(s);
}
int i=10;
if(input.length==1)
{
        tot=isize+lsize+val+isc+itab;
        System.out.println("\nTOTAL BLOCKS :"+tot);
}
else
{
        i=(i*input.length)-10;
        tot=isize+lsize+val+isc+itab+i;
        System.out.println("\nTOTAL BLOCKS :"+tot);
}
System.out.println("\n\n****** CALCULATIONS ****** ");
System.out.println("\nTotal Images :"+initimg);
System.out.println("Total Links :"+initlink);
System.out.println("Total Strings :"+initstring);
System.out.println("Total Scripts :"+initscript);
System.out.println("Total Tables :"+inittable);

System.out.println("\nPrimary          Images :"+pimg);
System.out.println("Primary  Links :"+plink);
```

```
System.out.println("Primary  Strings :"+pstring);
System.out.println("Primary  Scripts :"+pscript);
System.out.println("Primary  Tables :"+ptable);
float ctotalsize=initimg+initlink+initscript+initstring+inittable;
float priconsize=pimg+plink+pscript+pstring+ptable;
float pt=pstring;
float cb=priconsize;
float m=initstring-pstring;

System.out.println("\n\nPrimary Text Blocks= "+pt);
System.out.println("Primary Content Blocks= "+cb);
System.out.println("Missed Text Blocks= "+m);
float Bprecision=pstring/cb;
float Brecall=(pstring)/(initstring);
float BFMeasure=(2*Bprecision*Brecall)/(Bprecision+Brecall);

System.out.println("\n\nB-Precision= "+Bprecision);
System.out.println("B-Recall= "+Brecall);
System.out.println("BFMeasure= "+BFMeasure);

////////////////////////////////End of Table Extraction//////////////////////////////////
        }
        catch(Exception e){   }
        }
}
```

Appendix – II

Screen Shots

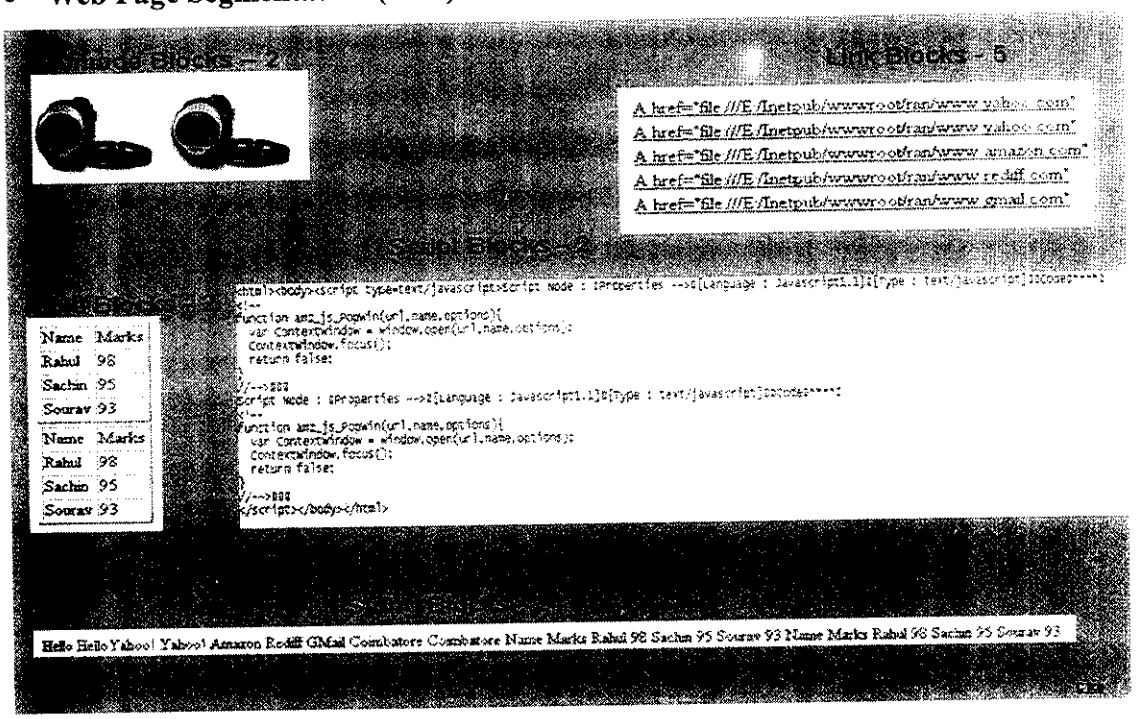
- **Graphical User Interface (GUI)**



- **Web Page Segmentation (Console)**

```
Total Number of Image Blocks = 2

Total Number of Link Blocks = 5

Total Number of String Blocks = 24

Total Number of Script Blocks = 2

Total Number of Table Blocks  = 2

TOTAL BLOCKS :35

****************************************** END OF WEB PAGE SEGMENTATION ***************
```

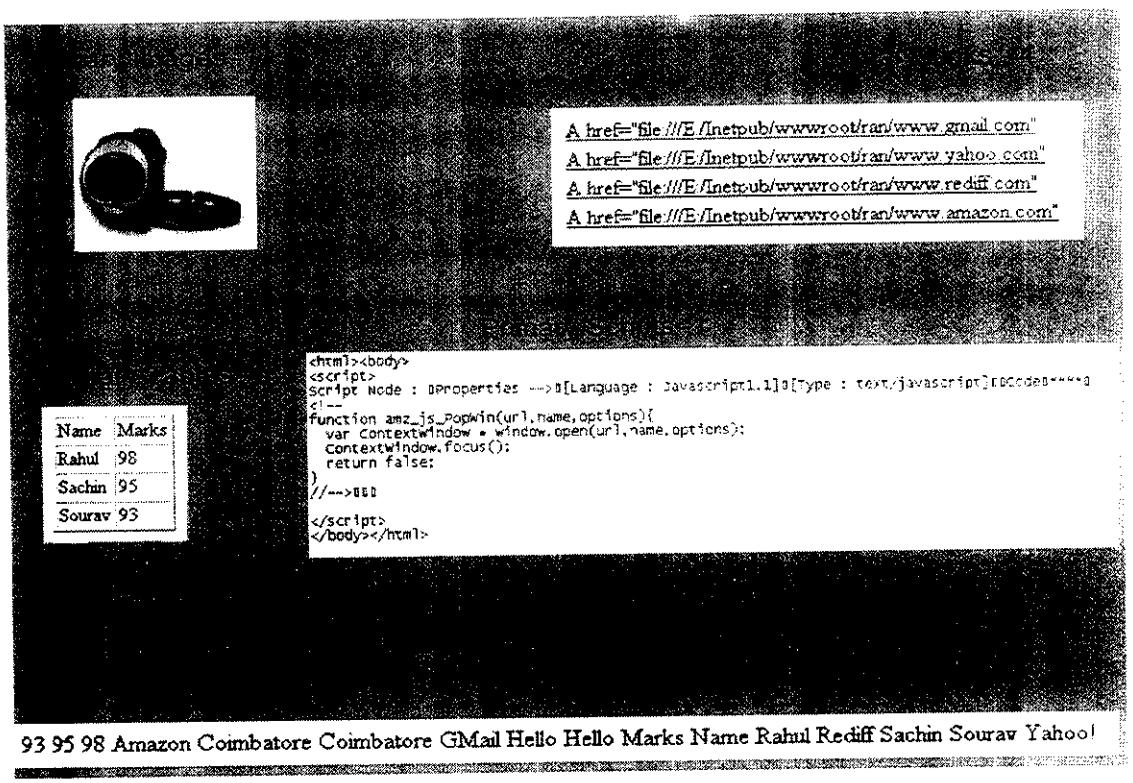
- **Web Page Segmentation (GUI)**



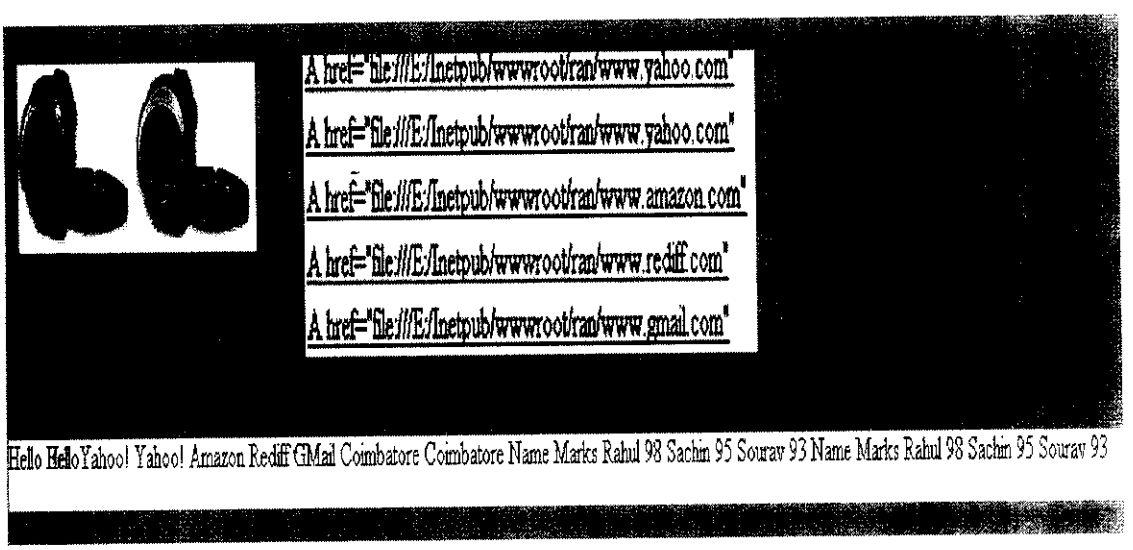
- **Content Extraction (Console)**

```
Primary Images :1.0
Primary Links :4.0
Primary Strings :15.0
Primary Scripts :1.0
Primary Tables :1.0


Primary Text Blocks= 15.0
Primary Content Blocks= 22.0
Missed Text Blocks= 9.0


B-Precision= 0.6818182
B-Recall= 0.625
BFMeasure= 0.6521739


********************************************** END OF CONTENT EXTRACTION ********************************
```

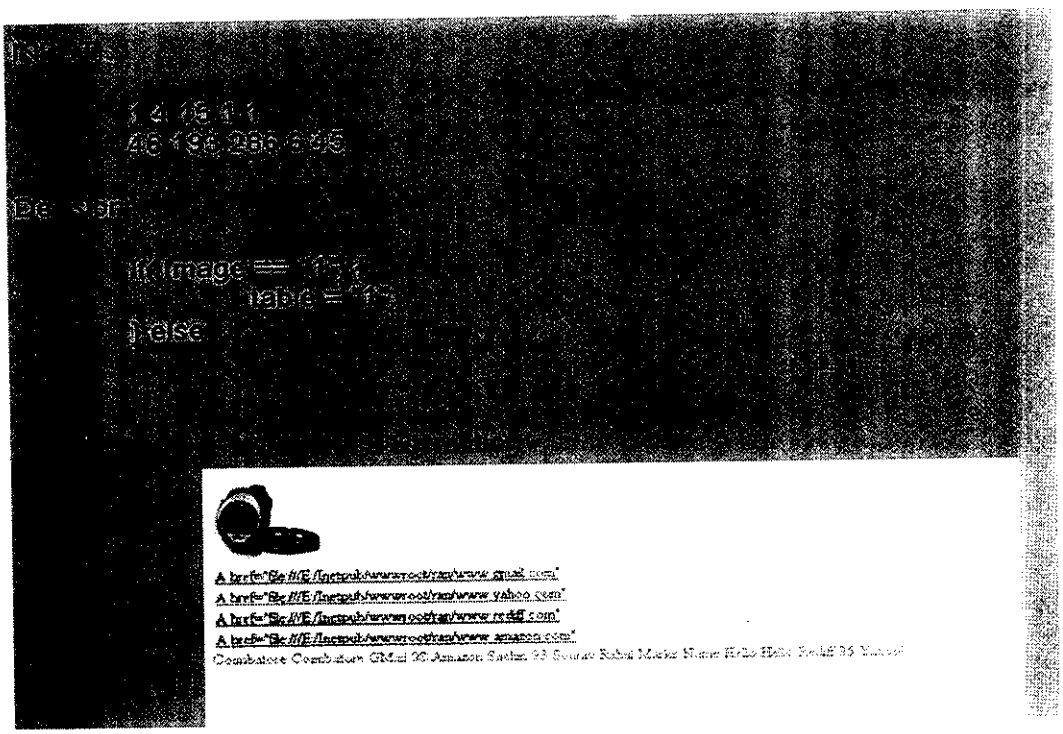
- **Content Extraction (GUI)**



- **K- Feature Extraction**



- **L- Extraction**



- **Performance Comparison**

Content Extraction	0.48263255	0.7810651	0.5966102
K-Feature Extraction	0.40995607	0.21840873	0.28498727
L-Extraction	0.1196347	0.21840873	0.15459116

REFERENCES

1. Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo, "Roadrunner: Towards automatic data extraction from large web sites", Proc. 27th Int'l Conf. Very Large Data Bases, pp. 109-118, 2001.

2. Ziv Bar-Yossef and Sridhar Rajagopalan," Template detection via data mining and its Applications", Proc. WWW 2002, pp. 580-591, 2002.

3. Shian-Hua Lin and Jan-Ming Ho," Discovering informative content blocks from web documents", Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 588-593, 2002.

4. Bing Liu, Kaidi Zhao, and Lan Yi," Eliminating noisy information in web pages for data mining", Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 296-305, 2003.

5. Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma, "Block based web search", Proc. 27th Ann. Int'l ACM SIGIR Conf., pp. 456-463, 2004.

6. Lakshmish Ramaswamy, Arun Iyengar, Ling Liu, and Fred Douglis, "Automatic detection of fragments in dynamically generated web pages", Thirteenth World Wide Web conference, 2004.

7. Ruihua Song, Haifeng Liu, Ji-Rong Wen, and Wei-Ying Ma, "Learning blocks importance models for web pages", Proc. 13th World Wide Web Conf., pp. 203-211, 2004.

8. Lakshmish Ramaswamy, Arun Iyengar, Ling Liu, and Fred Douglis, "Automatic fragment detection in dynamic a web pages and its impact on caching", IEEE Transactions on Knowledge and Data Engineering, 2005.

9. Jyotika Prasad and Andreas Paepcke: CoreEx: Content Extraction from Online News Articles. In KDD'08: Proceedings of the 11th ACM SIGKDD

10. M. Theobald, J. Siddharth, and A. Paepcke. Spotsigs:Robust and efficient near duplicate detection in large web collections. In Proceedings of the 31st Annual international ACM SIGIR Conference on Research and Development in Information Retrieval, 2008. Accessible at http://dbpubs.stanford.edu/pub/2008.

11. World Wide Web Consortium, World Wide Web Consortium Hypertext Markup Language.