EXTRACTION OF WEB BLOCKS FROM WEB PAGES AND ANALYSIS OF EXTRACTION ALGORITHMS

S.K.SHIRGAVE, V.B.BINAGE

Abstract: Web page can be divided in various blocks called as fragments. A fragment is a portion of a web page which has a distinct theme or functionality and is distinguishable from the other parts of the page.Dividing web pages into fragments has provided significant benefits. Good methods are needed for dividing web pages into fragments. Manual fragmentation of web pages is expensive, error prone, and un-scalable. Due to these problems, extraction of web fragments using Content extractor algorithm and DeSeA algorithm have been widely used.
The proposed work has following features:

1) Detect fragment using content extractor algorithm.
2) Extraction of fragment detected in step (1).
3) Detect fragment using DeSeA algorithm.
4) Extraction of fragment detected in step (3).
5) Analyze results of extracted fragment using above algorithms.

Index Terms: Fragment, ContentExtractor, DeSeA.

1. INTRODUCTION

The 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. Such blocks are 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 copy- right information etc. Before the content from a Web page can be used, it must be subdivided into smaller semantically homogeneous sections based on their content. Such sections are known as blocks. A block (or Web page block) B isaportionofaWebpageenclosedwithinanopen-tagand its matching close-tag, where the open and close tags belong to an ordered tag-set T that includes tags like <TR>, <P>, <HR>, and <UL>. Fig. 1, shows a Web page obtained from CNN's Web site1 and the blocks in that Web page.

- *S.K.SHIRGAVE, V.B.BINAGE*
- *Associate Professor, CSE/IT, DKTE's Textile and Engg. Institute ichalkaranji, Maharashtra, india, skshirgave@yahoo.com*
- *ME Student, CSE, D. Y. Patil College of Engg. And Technology Kolhapur, Maharashtra, India, vikbinage@yahoo.com*

We address the problem of identifying the primary informative content of a Web page. Identifying blocks involves partitioning a Web page into sections that are coherent, and that have specific functions. For example, a block with links for navigation is a navigation block. Another example is an advertising block that contains one or more advertisements that are laid out side by side. Usually, a navigation block is found on the left side of a Web page. Typically, the primary informative content block is laid out to the right of a Web page. For web block extraction we implemented two algorithms, ContentExtractor, and DeSeA which identify the primary content blocks in a Web page. An added advantage of identifying blocks in Web pages is that if the user does not require the non-content blocks or requires only a few non-content blocks, we can delete the rest of the blocks. This contraction is useful in situations where large parts of the Web are crawled, indexed, and stored. Since the non-content blocks are often a significant part of dynamically generated Web pages, eliminating them results in significant savings with respect to storage cache and indexing.

Fig. 1.A Web page from CNN.com and its blocks (shown using boxes).

Algorithms can identify similar blocks across different Web pages obtained from different Web sites. For example, a search on Google News on almost any topic returns several

syndicated articles. Popular items like syndicated columns or news articles written by global news agencies or Reuters appear in tens of newspapers. Even the top 100 results returned by Google contain only a very few unique columns related to the topic because of duplicates published at different sites. Ideally, the user wants only one of these several copies of articles. Since the different copies of the article are from different newspapers and Web sites, they differ in their non-content blocks but havesimilarcontent blocks. Byseparatingand indexingonly thecontentblocks, wecaneasilyidentifythattwoWebpages have identical content blocks, save on storage and indexing by saving only one copy of the block, and make search results better by returning more unique articles. Even search times improve because less data to search. ContentExtractor and DeSeA used to identify and separate content blocks from non-content blocksbased on the appearance of the same block in multiple Web pages. ContentExtractor and DeSeA produce excellent precision and recall values and runtime efficiency and, above all, do not use any manual input and require no complex machine learning process.

1.1 Literature Survey

The amount of information on the World Wide Web continues to grow at an astonishing speed. *Fragment*-based approach of web pages has been successfully commercialized in recent years [1]. An algorithm for identifying non content blocks (they refer to it as “noisy” blocks) of Web pages developed [5]. Their algorithm examines several Web pages from a single Web site. If an element of a Web page has the same style across various Web pages, the element is more likely than not to be marked as a non-content block. In order to identify the presentation styles of elements of Web pages, Yi and Liu’s algorithm constructs a “Style Tree”. A “Style Tree” is a variation of the DOM substructure of Web page elements. Another work that closely related was the work by Lin and Ho [5]. The algorithm proposed also tries to partition a Web page into blocks and identify content blocks. They used the entropy of the keywords used in a block to determine whether the block is redundant. Cai et al. [6] has introduced a vision-based page segmentation (VIPS) algorithm. This algorithm segments a Web page based on its visual characteristics, identifying horizontal spaces, and vertical spaces delimiting blocks much as a human being would visually identify semantic blocks in a Web page. They use this algorithm to show that better page segmentation and a search algorithm based on semantic content blocks improves the performance of Web searches. Ramaswamy et al. [8], [9] propose a Shingling algorithm to identify fragments of Web pages and use it to show that the storage requirements of Web caching are significantly reduced. Kushmerick [7] has proposed a feature-based method that identifies Internet advertisements in a Web page. It is solely geared toward removing advertisements and does not remove other non-content blocks. Although researchers have made considerable efforts to improve the performance and benefits of fragment-based caching, there has been little research on extracting cache- effective fragments in Web sites. Fragment-based caching solutions typically rely upon Web pages that have been manually fragmented at their respective Web sites by the Web administrator or the Web page designer. Manual markup of fragments from Web pages is both labor-intensive and error-prone. More importantly, identification of fragments by hand does not scale as it requires manual revision of the fragment markups in order to incorporate any new or enhanced features of content into an operational fragment-based solution framework . Furthermore, the manual approach to fragment extraction becomes unmanageable and unrealistic for edge caches that deal with multiple content providers. Thus, there is a need for schemes for extraction of fragment from web pages and that are scalable and robust for efficiently delivering web content. By “interesting”, we mean that the fragments detected are cost effective for fragment-based caching. Goal for web block extraction is to extract interesting fragments from web pages which exhibit potential benefits and, thus, are cost-effective as cache units, refer to these interesting fragments as candidate fragments. The Web documents considered here are well-formed HTML documents, although the approach can be applied to XML documents as well.

1.2 Limitations

In existing system humans can easily identify fragments with different themes or functionality based on their prior knowledge in the domain of the content. However, in order for machines and programs to automate the fragment extraction process, we need mechanisms that on the one hand can correctly identify fragments with different themes or functionality without human involvement, and on the other hand are efficient and effective for detecting and flagging such fragments through a cross-comparison of multiple pages from a web site. In past extraction of web blocks or web fragments from web pages and analysis of extraction algorithms work is done based on Comparison of Content Extractor with Feature Extractor, K-Feature Extractor, and with LH algorithm. Work in needed to compare Content Extractor with DeSeA Algorithm.

1.3 Need of Present Work

In the papers cited at references [1], [5], [6], [7], [8], [9] it shows that many researchers worked on the Extraction of web blocks from web pages and analysis of algorithms. For that purpose they used different rules, patterns and information retrieval strategies of web mining. Taking into consideration all above techniques, the work uses “content extractor algorithm”, and “DeSeA algorithm” to detect and extract fragments (web blocks) in web pages and analysis of extraction algorithms based on precision and recall values. It analyzes web pages with respect to their information sharing behavior, personalization characteristics, and the change frequencies over time. Based on this analysis, this system detects and flags the “interesting” fragments in a web site. We consider a fragment interesting if it has good share ability with other pages served from the same web site or it has distinct lifetime Characteristics. This work consists following main tasks:

1) Extraction of fragment (web blocks) from web page using Content extractor algorithm.
2) Extraction of fragment (web blocks) from web pages using DeSeA algorithm.
3) Analysis of detected fragment (web blocks) using above algorithms.

2 SYSTEM DESIGN

The previous approaches for extraction of web blocks from web pages and analysis of extraction algorithms.Different authors have compared ContentExtractor with FeatureExtractor, K-FeatureExtractor as well as LH (Lin and Ho) Algorithm Discussedas in chapter 1. This chapter highlights on the problem statement, describes the architecture of the proposed system and algorithms used for implementation of the system.

2.1 Problem statement

Extraction of web blocks from web pages and analysis of extraction algorithms

2.2 Architecture of Web block extraction

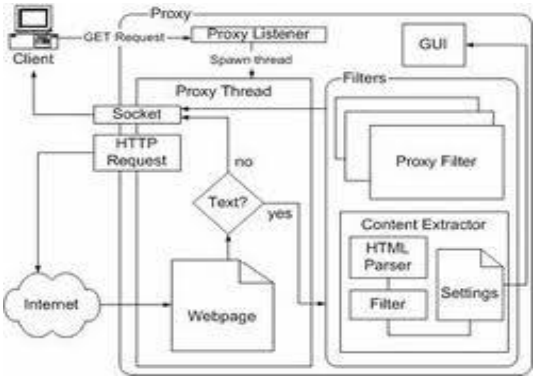


Figure 2.1 Architecture of Proposed System

The architecture of the proposed system is shown in figure2.1. The proposed system consists of different parts such as HTML Parser, Filter and Settings. In the HTML Parser convert the web page into DOM Tree. Filter block take input as a DOM tree and partition it into informative and non-informative contents. Settings block shows the web page by extracting redundant web blocks from web pages.

2.3 Processes

The proposed work consists of following processes:

i) Segmenting web pages into blocks
ii) ContentExtractor Process
iii) DeSeA Process

2.3.1 Segmenting web pages into blocks

Most Web pages on the Internet are still written in HTML [8]. Even dynamically generated pages are mostly written with HTML tags, complying with the SGML format. The layouts of these SGML documents follow the Document Object Model tree structure of the World Wide Web Consortium.2 Out of all of these tags; Web authors mostly use <TABLE> to design the layouts. ContentExtractor 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. Algorithm Select the order of the tags based on our observations of Webpages 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>. 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. In partitioning algorithm this tag-set is called the partitioning tag-set. In this process, differentHTML documents (web pages) are collected. Thesedocuments are converted into XML code in order to generate DOM Tree.Itconsists of following steps:

1) Filtering the data from web pages

This module works only on HTML pages. Normally web pages contain data such as hyperlinks, images, scripts, advertisements, noisy data etc.The main objective is to process web page and concentrate only on web blocks. So it is easy to remove such unwanted data if any, when a page is selected for processing.

2) XML conversion and DOM tree Generation

This process includes fetching the web page from specific location and converting it into XML code. This XML code is then used for the generation of DOM tree. The Document Object Model (DOM) is an application programming interface (API)for valid HTMLand well-formed XML documents.It defines the logical structure of documents and the way a document is accessed and manipulated. The XML-DOM defines a standard way for accessing and manipulating XML documents. The DOM presents an XML document as a tree-structure. Figure2.2 shows the general structure of the DOM treein which each element is separated based on Document, Root element, Element, Attribute and Text.

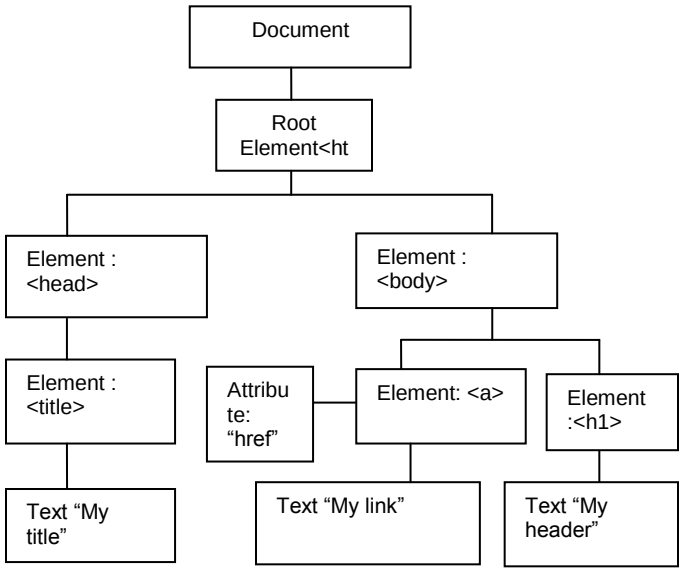


Figure 2.2 DOM Tree

2.3.2 ContentExtractor process

The input to the ContentExtractor 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 output of the algorithms are the primary content blocks in the given class of Web pages. Following functions have been used for algorithm implementation

a. GetBlockSet():-

- Takes HTML page as input with ordered tag set
- Take a tag from tag set one by one & call getBlock() routine
- New sub block created by getBlocks are added to the block set & remove main block
- First() function return first tag of ordered set
- Next() function gives consecutive tag of ordered list

b. GetBlocks():-

- Takes full document or part of document (HTML) as input
- It partition the document or part of document into blocks according to input tag
- If particular tag not present in web page (HTML) it return whole web page as single block

c. Identify Content block and separate it from non-content block.

d. ContentExtractor ():-

- Calculate Inverse Block Document Frequency (IDBF)
- Similarity function Sim use to calculate similarity between blocks

e. Similarity function & threshold

- Input is two blocks it return cosine between their block future vectors.
- Threshold value used is ε=0.9 i.e. if similarity measure value greater than 0.9 then two blocks are identical

2.3.2.1 GetBlockSet

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 (which was just partitioned) 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. Feature identification is a very important step in machine learning approach.The different usage patterns must be extracted by considering the appearance of the tables as expressed by the table tags and from the content instance type of each cell. The appropriate features are considered for distinguishing meaningful tables from decorative tables [1].These are classified into two categories such as“appearance features” and “consistency features”.

2.3.2.2 GetBlocks

GetBlocks takes a full document or a part of a document, written in HTML, and a tag as its input. It partitions the document into blocks according to the input tag. For example, in case of the <TABLE> tag given as input, it will produce the DOM tree with all the table blocks. It does a breadth-first search of the DOM tree (if any) of the HTML page. If the input tag is <TABLE> and there is no table structure available in the HTML page, it does not partition the page. In that case, the whole input page comes back as a single block. In case of other tags such as <P>, it partitions the page/block into blocks/subblocks separated by those tags. Fig. 2 shows the structure of two HTML pages. It also shows the blocks that our blocking algorithm identifies for each of these pages (under the dotted line).

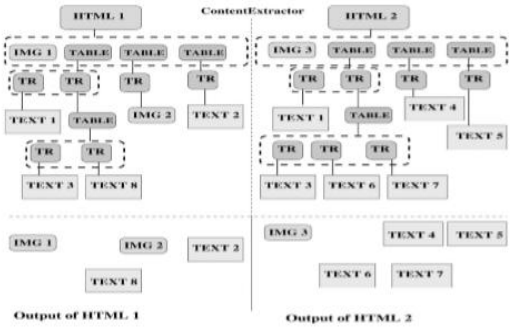


Fig. 2.3 Two Web pages' block structures as seen by GetBlockSet. The output from them is shown under the dotted line.

2.3.3 DeSeA Process

DeSeA is implemented in 2 steps. A web page is divided into coherent blocks first. Then relevant blocks are detected from them.

2.3.3.1Block Extraction

The block extraction process is divided into splitting and merging. In splitting process, a web page is segmented into blocks using level-1 delimiters first, and the hierarchical structure is recorded into a block tree. For all leaf nodes in the block tree, same process is carried out using higher-level delimiters until all leaf nodes all in block tree satisfy the granularity requirement controlled by an integer value α called window size. The experiments lead up to the fact that the accuracy reaches the highest when α is 300. For each segmentation round, the EDT is segmented using specific level delimiters. It is started from the root node of the EDT. The whole web page is put into the block tree as the root node first. From top to bottom, each node in the block tree is checked whether it forms a single block in order to be processed using rule set described below. If it forms a single block, it is put into the block tree directly as a leaf node and needn't to be segmented any more. Otherwise, it's segmented into smaller blocks based on delimiters and the EDT. Smaller blocks after segmentation are put into the block tree as leaf nodes of current round. Following predicates are defined before introducing rule set for page splitting:

Larger (n, s): Character number of node n in block tree is larger than s;

CVD (n): Node n in block tree has a different page visual attribute value compared with other child nodes of it;

CSD (n): Node n in block tree has a page structural delimiter;

CDD (n): Node n in block tree has a domain-specific delimiter;

Single (n): Node n in block tree forms a single block;

DVN (n): Divide node n according to delimiter, making nodes with the same visual attribute value in a single block;

DSN (n): Divide node n according to page structural delimiter and the EDT;

DDN (n): Divide node n according to domain-specific delimiter in the same way as the structural delimiter.

Using above predicates, we define rule set as follows:
(CSD(n)||CDD(n)) && Larger(n,s)) →Single(n)
Larger(n,s) && CVD(n) →DVN(n)
Larger(n,s) && CSD(n) → DSN(n)
Larger(n,s) && CDD(n) → DDN(n)

This splitting process is demonstrated in following code.

```
function PageSplitting(EDT)
begin
     put the whole EDT into the block tree
                    as the root node
     level=0
     while(not all leaf nodes meet the
                    granularity
                    requirement &&
                    !level<max_level)
          Process(block tree's root,
                level)
          level++
end
function Process(node, level)
begin
     if node is null
          then return
     else begin
          Visit(node, blockTree, level)
          for each children of node
               Process(child, level)
     end
end
function Visit(node, blockTree, _level)
begin
     if node doesn't contain any _level
               delimiters
          then return
     else begin
          for each _level delimiter
              divide node and insert them as
               children of node
     end
end
```

After splitting, a block tree is constructed. Sibling nodes in the block tree are processed to determine whether they should be merged based on the next-to relationship among blocks. Among above delimiters, ';', '.' and <Hx> with the same x value where x is a numerical character like '1', '2' have next-to type. Figure2.4 shows an example of this merging process, where two leaf nodes in block tree which is divided by "." are merged. Take the above page paragraph for example, whole page paragraph is processed by level-1 delimiters <head> and

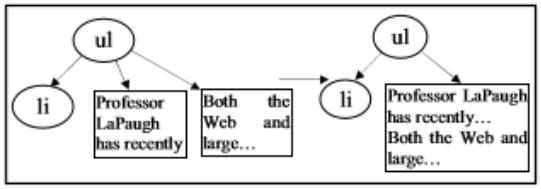
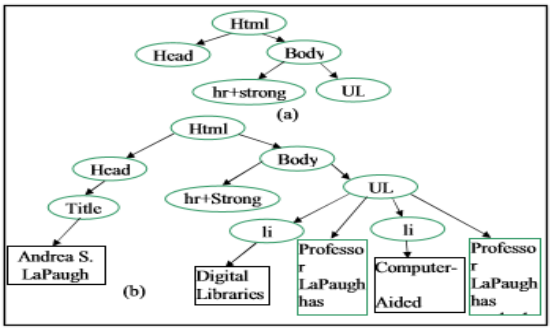


Figure2.4 An Example of Merging process

<UL> first, splitting it into 3 corser sub-blocks with <head>, <UL> and <P+Strong> as the block's root. Its splitting result is shown in Figure2.5 (a). Then each leaf block in block tree is checked whether it meet the granularity requirement. If not, the block tree is processed using higher-level delimiters similarly. Finally, extraction result is shown in Figure 2.5 (b).

Figure2.5 . An example of Block Tree

2.3.3.2 Relevant Block Detection



Relevant block means a block that contains a faculty's research interests. Each type of block has an anchor text. Take "<UL></UL>" block for example, its anchor text is defined to be 80 character before the start of block. Degree of relevant is calculated by the DoR value assigned to each block based on cue phrases appeared in the anchor text. The DoR value is calculated based on following considerations: 1) If an anchor text contains no cue phrase, its DoR value is zero; 2) If an anchor text contains one cue phrase, its DoR value is proportional to the cue phrase's priority; 3) If it contains more than one cue phrase, its DoR value is proportional to the highest cue phrase priority; 4) When block A and block B is merged together, its DoR value is the higher one between DoR of A and B. Blocks with non-zero DoR value are put into the candidate block list and sorted in descending order of DoR. The top ù blocks from the candidate block list are selected and put into the relevant block set. Through experiments, ù is assigned to 10.

3 IMPLEMENTATION

In this chapter the details of the classes used for the development of modules are described. It also focuses on the processing of a web page containing a web blocks. The proposed system shown in figure1.1 illustrates the flow of implementation. In the Segmenting web page into blocks process, HTML documents are collected from the web and their XML conversion is carried out. The DOM tree is generated for the respective pages. The operations block recognitionis performed to obtain the web blocks, which are shown in figure 3.1.

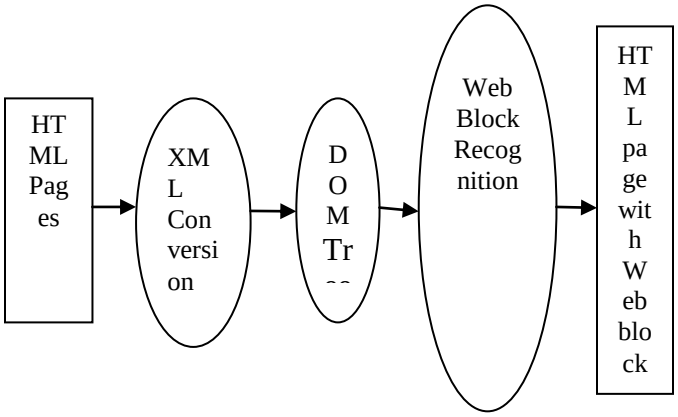


Figure 3.1 DFD for selecting web block

Aweb page without web blocks is generated by applying ContentExtractor process. Figure 3.2 shows the procedure of extraction of web blocks from web page using ContentExtractor algorithm.

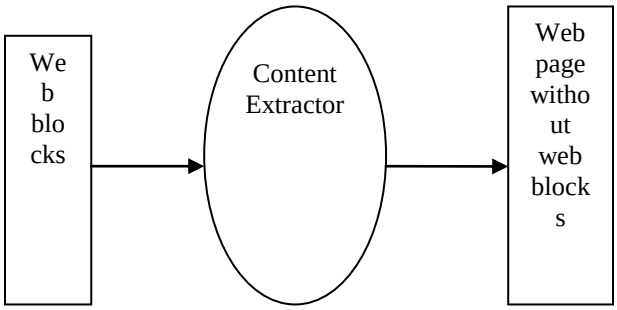


Figure 3.2 DFD for extraction of web blocks from web pages using ContentExtractor

A web page without web blocks also generated using DeSeA process. Figure 3.3 shows procedure of extraction of web blocks from web pages using DeSeA algorithm.

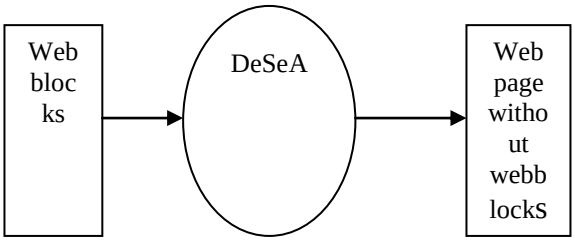


Figure 3.3 DFD for extraction of web blocks from web pages using DeSeA

3.1 Content Extraction Algorithm

Five classes are used under the package name kit. Name of the classes and there functionality is as follows.

1) ContentExtractor class :
It takes input as two web pages belonging to class of web page and gives output like

a) Page 1 Total no. of Blocks
b) Page 2 Total no. of Blocks
c) Total redundant Blocks

2) Match class :
This class is used to check the matching of tags between two web pages, which is given as input in ContentExtractor class.

3) GetBlocks class :
This class works as GetBlocks() routine present in ContentExtractor algorithm. Input is 2 html pages which is given in Main class and output is blocks retrieved from page1 & page2 in the form of ArrayList.

4) FindRedundant class :
This class performs comparison of blocks retrieved by getblocks class. Input is Files containing Blocks retrieved from web page in the form of ArrayList and output is list of Redundant blocks .Similarity function in ContentExtarctor algorithm is also implemented in this class.

5) RemoveRedundant class :
This class Arranges the page2 without redundant blocks. Input is Page2 and ArrayList of RedundantBlocks given by class FindRedundant. Output is Page2 without Redundant blocks.

3.2 DeSeA Algorithm

Following classes are used in implementation of DeSeA algorithm. Name of the classes and there functionality is as follows.

1) ParserBlogFile: main function, give requested two html file name and two resulted file names total 4 input strings.
2) ParseBlogElement: parser's object is created. two methods are there
  a. parseElement: returns NodeList object. Here whole file get parsed n stored into NodeList as well it is having html object where nodes get stored
  b. getHTMLNode: it returns Node object.
3) ParseFile: whole file get parsed with respect to their text, image and link.

3.3 HTML Parser Libraries and classes used

These java libraries provide access to the contents of local or remote HTML resources in a programmatic way. The HTML Parser distribution is composed of

- a low level laxer that converts characters from a HTML page into a linear sequence of nodes
- a high level parser that provides a hierarchical document model of a HTML page

The different classes used for development of system are as mention below.

i) NodeList(Node node)
The nodes in the NodeList can be accessed through their index number (starting from 0).The NodeList keeps itself up-to-date. If an element is deleted or added, in the node list or the XML document, the list is automatically updated.

Table 3.1 Methods of StreamReader Class

Name	Description
int getLength()	Returns the number of nodes in a NodeList
Node item (int index)	Returns the node at the specified index in a NodeList

ii) Attributes Class

The Attributes class maps Manifest attribute names to associated string values. Valid attribute names are case-insensitive, are restricted to the ASCII characters in the set [0-9a-zA-Z_-], and cannot exceed 70 characters in length. Attribute values can contain any characters and will be UTF8-encoded.

Table 3.2 Methods of Attributes Class

Name	Description
void clear()	Removes all attributes from this Map.
Object clone()	Returns a copy of the Attributes.
boolean containsKey (Object name)	Returns true if this Map contains the specified attribute name (key).
boolean containsValue (Object value)	Returns true if this Map maps one or more attribute names (keys) to the specified value.
boolean equals (Object o)	Compares the specified Attributes object with this Map for equality.
Object get (Object name)	Returns the value of the specified attribute name, or null if the attribute name was not found.
StringgetValue (String name)	Returns the value of the specified attribute name, specified as a string, or null if the attribute was not found.
boolean isEmpty()	Returns true if this Map contains no attributes.
Object put (Object name, Object Value)	Associates the specified value with the specified attribute name (key) in this Map.
Object remove (Object name)	Removes the attribute with the specified name (key) from this Map.

iii) Parser Class

The Parser provides access to the contents of the page. Parser class is having following methods.

Table 3.3 Methods of Parser Class

Name	Description
static ParsercreateParser(String html, String charset)	Creates the parser on an input string.
NodeIteratorelements()	Returns an iterator (enumeration) over the html nodes.
NodeListextractAllNodesThatMatch(NodeFilter filter)	Extract all nodes matching the given filter.
URLConnectiongetConnection()	Return the current connection.
static ConnectionManagergetConnectionManager()	Get the connection manager all Parsers use.
StringgetEncoding()	Get the encoding for the page this parser is reading from.
LexergetLexer()	Returns the lexer associated with the parser.
NodeFactorygetNodeFactory()	Get the current node factory.
StringgetURL()	Return the current URL being parsed.
NodeListparse(NodeFilter filter)	Parse the given resource, using the filter provided.
void reset()	Reset the parser to

	start from the beginning again.
void setEncoding(String encoding)	Set the encoding for the page this parser is reading from.
void setInputHTML(String inputHTML)	Initializes the parser with the given input HTML String.
void setLexer(Lexer lexer)	Set the lexer for this parser.
void setResource(String resource)	Set the html, a url, or a file.
void setURL(String url)	Set the URL for this parser.

iv) Tag Interface:-

This interface represents a tag (<xxx yyy="zzz">) in the HTML document. Adds capabilities to a Node that are specific to a tag. Tag Interfaceis having following methods.

Table 3.4 Methods of Tag Interface

Name	Description
StringgetAttribute(String name)	Returns the value of an attribute.
TaggetEndTag()	Get the end tag for this (composite) tag.
StringgetTagName()	Return the name of this tag.
boolean isEndTag()	Predicate to determine if this tag is an end tag
voidremoveAttribute(String key)	Remove the attribute with the given key, if it exists.
voidsetAttribute(String key, String value)	Set attribute with given key, value pair.
voidsetEndTag(Tag tag)	Set the end tag for this (composite) tag.
voidsetTagName(String name)	Set the name of this tag.

4 EXPERMENTAL RESULTS

This chapter highlights on the experimental results which are obtained using ContentExtractor algorithm and DeSeA algorithm.For experiments, different HTML web pages are collected from different News web sites along with number of web blocks per web site. Using this data set, experiments are carried out on: 1) Extraction of web blocks from web pages using ContentExtractor model and 2) Extraction of web blocks from web pages using DeSeA model.The algorithms are evaluated on Precision and recall values of the web pages.

4.1. Input Dataset

Table 4.1 shows the information about the input dataset and the experiments which are carried out on this dataset by using ContentExtractor model and DeSeA model. This dataset contains in total, 15 different Web sites including news, shopping, opinion posting Web sites, etc., whose designs and page-layouts are completely different.

Table 4.1 Details of input data set

Site	*Address*	*Category*	*Number*
ABC	http://www.abcnews.com	Main Page, USA, World, Business, Entertainment, Science/Tech, Politics,Living	415
BB	http://www.bloomsberg.com	Main Page, World, Market, US Top Stories, World Top Stories, Asian, Australia/New Zealand, Europe, The Americas	510
BBC	http://www.bbc.co.uk	Main Page, The Continents, Business, Health, Nature, Technology, Entertainment	890
CBS	http://www.cbsnews.com	Main Page, National, World, Politics, Technology, Health, Entertainment	370
CNN	http://www.cnn.com	Main Page, World, US, All Politics, Law, Tech(nology), Space (Technology), Health, Showbiz, Education, Specials	717
FOX	http://www.foxnews.com	Main Page, Top Stories, Politics, Business, Life, Views	476
FOX23	http://www.fox23news.com	Main Page, General, Local, Regional, National, World, In Depth, Sports, Business, Entertainment, Health	658
IE	http://www.indianexpress.com	Main Page, International, Sports, National Network, Business, Headlines	269
IT	http://www.indiatimes.com	Main Page, Main Stories, Top Media Headlines	454
MSNBC	http://www.msnbc.com	Main Page, Business, Sports, Technology an Science, Health, Travel	647
YAHOO	http://news.yahoo.com	Main Page, Top Stories, US (National), Business, World, Entertainment, Sports, Technology, Politics, Science	505
Shopping	http://www.shopping.com	Miscellaneous Products	100
Amazon	http://www.amazon.com	Book Pages	100
Barnes And Noble	http://www.bn.com	Book Pages	100
Epinion	http://www.epinions.com	Reviews	100

4.2.Experimental Results using ContentExtractor model and DeSeA model

The block level Precision rate (b-Precision), and block level Recall rate (b-Recall) are used to measure the performance. 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:

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

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 b-Recall:

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

Table 4.2 shows Block Level Precision and Recall Values for ContentExtractor Model and Table 4.3 shows Block Level Precision and Recall Values for DeSeA Model.

Table 4.2 Experimental results on the input data set by applying ContentExtractor Model

Site	b-Precision of CE	b-Recall of CE
ABC	0.90	0.90
BB	1.0	1.0
BBC	0.58	0.58
CBS	0.55	0.55
CNN	0.88	0.88
FOX	0.83	0.83
FOX23	0.92	0.92
IE	0.61	0.61
IT	0.35	0.35
MSNBC	0.28	0.28
YAHOO	1.0	1.0
Shopping	0.37	0.37
Amazon	1.0	1.0
Barnes and Noble	1.0	1.0
Epinion	1.0	1.0

Table 4.3 Experimental results on the input data set by applying DeSeA Model

Site	b-Precision of DeSeA	b-Recall of DeSeA
ABC	1.0	1.0
BB	1.0	1.0
BBC	1.0	1.0
CBS	0.16	0.16
CNN	0.84	0.84
FOX	1.0	1.0
FOX23	0.46	0.46
IE	1.0	1.0
IT	1.0	1.0
MSNBC	0.56	0.56
YAHOO	1.0	1.0
Shopping	1.0	1.0
Amazon	1.0	1.0
Barnes and Noble	1.0	1.0
Epinion	1.0	1.0

4.2 Result Analysis

Figure 4.7 shows the graph which represents the performance of the extraction of web blocks from web pages using ContentExtractor and DeSeA System. X-axis shows web site names and Y-axis shows percentage of Precision and Recall.

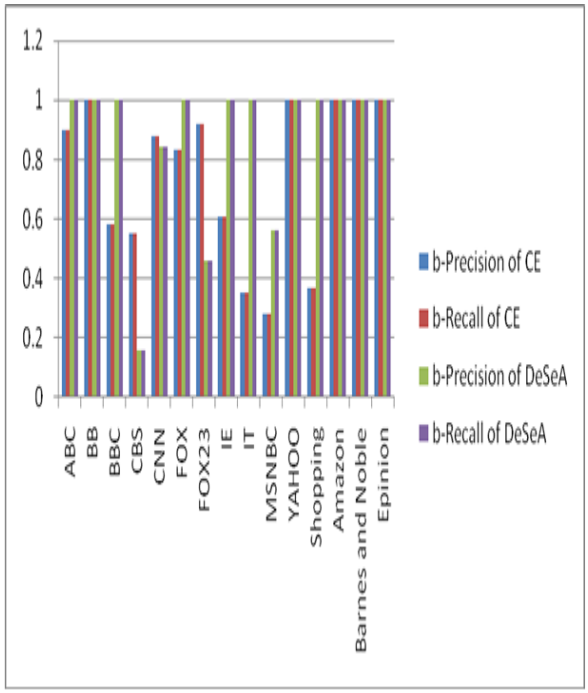


Figure 4.4 Graph of performance of the extraction of web blocks by CE and DeSeA

5 CONCLUSION

Unlike information retrieval, the goal of IE is to transform text into a structured format and thereby reducing the information in a document to a tabular structure. The ContentExtractor 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 “DeSeA Algorithm” is as good as from ContentExtractor, ContentExtractor outperforms the “DeSeA Algorithm” by a high margin in runtime. Intend to deploy ContentExtractor algorithm as a part of a system that crawls Web pages, and extracts primary content blocks from it. 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 (as we have shown briefly by caching size benefit) since now only the relevant parts of the documents are considered. It’s obvious that exact positioning of information is critical to information extraction and knowledge discovery. DeSeA algorithm divides a web page into coherent blocks, and separates relevant information from irrelevant one. The segmentation process of *DeSeA* simulates how a user understands the content of a web page. It is a reconstruction of inner structure of a web page, transforming an *EDT* to a *block tree*, based on pre-defined page delimiters and domain-specific delimiters. Compared with existing page segmentation method, DeSeA divides delimiters into different level based on their content splitting ability, with higher-level delimiters having higher priority to segment a web page. After applying ContentExtractor and DeSeA algorithms for extraction of web blocks from web pages we can remove redundant web blocks from web pages hence it will be beneficial for web page caching and for crawling the web pages. After

extraction of redundant web block web page only contains meaningful information. It is observed that b-Precision and b-Recall values for DeSeA algorithm is good compared to the ContentExtractor algorithm.

6 Further Work

As the proposed system considers only two web pages from class of web pages , the proposed system can be further enhanced to process more than two web pages for web block extraction. In Proposed system DeSeA algorithm is compared with ContentExtractor further DeSeA algorithm can be compared with Feature-Extractor and K-Feature-Extractor algorithm as well as LH algorithm. Further DeSeA algorithm extract research interests from relevant text using natural language processing techniques, and extract *cue phrase*s automatically using those manually extracted results as training data to make this algorithm scalable. Artificial Neural Network can be used for finding redundant web blocks from more than two web pages.

7 REFERENCES

[1]. Sung-Won Jung, and Hyuk-Chul Kwon, “A Scalable Hybrid Approach for Extracting Head Components from Web Tables”, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 18, NO. 2, FEBRUARY 2006.

[2]. Jeong-Woo Son, Jae-An-Lee, Seong-Bae Park, Hyun-Je Song, Song-Jo Lee, Se-Young Park, “Discriminating Meaningful Web Tables from Decorative Tables Using a Composite Ker-nel”, 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.

[3]. Chen Hong-ye, “Method of Web Information Extraction Based on Decision Tree”, 2009 International Forum on Information Technology and Applications.

[4]. H.H. Chen, S.C. Tsai, and J.H. Tsai, “Mining Tables from Large Scale HTML Texts”, Proc. 18th Int’l Conf. Computational Linguistics, July 2000.

[5]. M. Hurst, “Layout and Language: Beyond Simple Text for Information Interaction—Modeling the Table”, Proc. Second Int’l Conf. Multimodal Interfaces, 1999.

[6]. G. Ning, W. Guowen, W. Xiaoyuan, and S. Baile, “Extracting Web Table Information in Cooperative Learning Activities Based on Abstract Semantic Model”, Proc. Sixth Int’l Conf. Computer Supported Cooperative Work in Design, pp. 492-497, 2001.

[7]. Y. Wang and J. Hu, “A Machine Learning Based Approach for Table Detection on the Web”, Proc. 11th Int’l World Wide Web Conf. WWW 2002, pp. 7-11, 2002.

[8]. S. Soderland, “Learning to Extract Text-Based Information from the World Wide Web” ,Proc. Third Int’l Conf. Know-ledge Discovery and Data Mining (KDD), Aug. 1997.

[9]. M. Hurst. Layout and language: Challenges for table under-standing on the Web. In Proc. 1st WDA at 6th ICDAR, pp. 27{30, Sept. 2001}.

[10]. A. Tengli, Y. Yang, and N. L: Machine Learning table extraction from examples. In Proc. 20th COLING, pp. 987-993. COL-ING, Aug. 2004.

[11]. Margaret Dunham, Data Mining Introductory and Advanced Topics, ISBN: 0130888923, Prentice Hall, 2003