Content-Rich Web Page Segmentation (CWPS)

GRADUATE PROJECT REPORT

Submitted to the Faculty of
the Department of Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the requirements for the Degree of
Master of Science in Computer Science

by

Tejashree P. Tendolkar
Fall 2006

Committee Members

Dr. Longzhuang Li
Committee Chairperson

Dr. Dulal Kar
Committee Member

Dr. David Thomas
Committee Member
ABSTRACT

The goal of this project is to calculate the visual similarity values based on different cues and segment the content in the Web page into subtopic structure. Web pages on the Internet have become an important resource for people to find information of their interest. But Web pages usually contain noise, such as advertisements and navigational bar which may easily distract users. Furthermore, a Web page may have content of different topics and a user is only interested in one topic. In this project a content-rich Web page segmentation method (CWPS) was implemented based on HTML tag structures and visual cues. Current plain text topic segmentation methods do not work well on Web pages. All the existing Web page structural segmentation algorithms employ HTML tag information to partition a Web page into a set of blocks with each containing related information. Basically, current structural segmentation algorithms mostly can only segment the text in the content–rich Web pages at paragraph level. The CWPS method consists of three contiguous procedures: structural segmentation, visual similarity calculation, topic segmentation. In structural segmentation, a page is first partitioned into information blocks using a Web page segmentation algorithm. Then noisy blocks are identified and removed, and text blocks are concatenated. In visual similarity calculation, visual cues are detected, and then the visual similarities between sentences are calculated and integrated with a plain text segmentation method to detect the subtopic boundaries. In topic segmentation, the plain text is segmented in subtopics. The CWPS Method takes into account the structure characteristics in Web pages as well as the visual similarities and lexical analysis of the extracted text blocks. The CWPS calculates the visual
similarity values based on different visual cues which gives better subtopic partitioning results than the TextTiling method.
# TABLE OF CONTENTS

Abstract .......................................................................................................................... ii
Table of Contents ............................................................................................................ iii
List of Figures ................................................................................................................ iv
List of Tables ................................................................................................................... v
1  Background and Rationale .......................................................................................... 1
  1.1 Introduction ........................................................................................................... 1
  1.2 Current Web Page Structural Segmentation Algorithms ...................................... 2
  1.3 Advantages of the New System ............................................................................ 3
2  Content-Rich Web Page Segmentation (CWPS) ....................................................... 5
  2.1 Overview ............................................................................................................. 5
  2.2 VIPS Algorithm. ................................................................................................... 6
    2.2.1 Visual Block Extraction ............................................................................... 8
    2.2.2 Visual Separator Detection ....................................................................... 11
    2.2.3 Setting Weights for Separators .................................................................. 14
    2.2.4 Content Structure Construction ................................................................... 15
    2.2.5 Structural Segmentation Analysis ................................................................ 16
  2.3 The TextTiling Algorithm .................................................................................... 20
    2.3.1 Subtopic Structure Detection via Lexical Coherence .................................. 21
    2.3.2 Topic Segmentation Analysis ...................................................................... 22
  2.4 CWPS Analysis ................................................................................................... 24
3  System Design ........................................................................................................... 25
  3.1 System Analysis ................................................................................................... 25
LIST OF FIGURES

Figure 2.1 Web Page Segmentation Method ................................................. 5
Figure 2.2 The Vision Based Page Segmentation Method .............................. 8
Figure 2.3 Sample Separator Detection Process ........................................... 13
Figure 2.4 Visual Block tree of the Web Page ............................................... 14
Figure 2.5 Sample Page Layout ................................................................. 17
Figure 2.6 DOM tree ................................................................................. 18
Figure 2.7 Screenshot of the VIPS Method for a Selected Doc Value ............. 19
Figure 2.8 Screenshot of the TextTiling Method Output ................................. 23
Figure 3.1 System Architecture ................................................................. 27
Figure 3.2 Example of HTML Tag Structure ............................................... 31
Figure 3.3 Visual Similarity Matrices ............................................................ 32
LIST OF TABLES

Table 2.1 Heuristics Rules in Block Extraction Phase……………………………………10
Table 2.2 Different rules for different DOM nodes……………………………………..11
Table 2.3 Example Text segmentation topics of sample text………………………….21
Table 4.1 Manual Segmentation of Sample Web Pages………………………………....36
Table 4.2 Comparison between methods………………………………………………..38
1. INTRODUCTION AND BACKGROUND

1.1 Introduction

In today’s world the Web is the most important source of information for people. A lot of Web pages contain main content information in the form of text, and also contain lots of unwanted information such as pop-up ads, unwanted links, images, blanks etc. This unwanted information is spread around the main content information. As a result, the user gets distracted easily. It is very important to identify and extract the main content information from the Web pages because not all of the information is useful and interesting to the users. When the users do an information retrieval on the Internet to answer their questions at hand, they only need to read the most relevant content.

Current plain text topic segmentation methods do not work well on Web pages. Most of the Web pages are differently organized from others. Also the Web page has extraneous data which is irrelevant to the main topic, e.g. advertisements, navigational links etc. There are two categories of plain text topic segmentation method: lexical cohesion methods and multi-source methods. The main characteristic of the lexical cohesion method is topic segment, which contains texts with similar words. The lexical cohesion algorithm, TextTiling [Hearst 1994], is discussed in the later sections of this report. The main idea behind lexical cohesion is to segment the written text and improve information retrieval. Multi-source methods employ both lexical cohesion and indicators of topic transition, such as cue phrases and prosodic features, to segment transcribed spoken texts and broadcast news into individual stories. To make the current plain text topic segmentation methods work on Web documents all the unwanted information such as navigational bars and advertisements etc. have to be removed. So the content-rich Web
pages segmentation (CWPS) method consists of three steps: 1) Partition the Web page into blocks using current Web structural segmentation algorithms and remove noisy blocks; 2) Calculate the visual similarity values; 3) Segment the remaining blocks into subtopics using visual cues and a plain text topic segmentation method.

1.2 Current Web Page Structural Segmentation Algorithms

The current Web page segmentation algorithms use HTML tag information to separate Web pages into different blocks that contain separate information. Three of the popular algorithms for Web page segmentation are 1) Document Object Model (DOM) based segmentation [Chen 2001], 2) Location based Page Segmentation [Kovacevic 2002] and 3) Vision-based Page Segmentation (VIPS) [Cai 2003]. In the DOM based segmentation algorithm [Chen 2001], a Web page is first represented as a DOM tree, and then tags, such as P, TABLE, and UL are used to build blocks. Because DOM is provided for not only content organization but also layout presentation, it is often not accurate enough to use tags such as TABLE and P to represent different semantic blocks in a Web page. The location based segmentation algorithm [Kovacevic 2002] is based on the layout of the Web page. The Web page is separated into 5 regions: top, down, left, right and center. But this structure can not be applied to all Web pages and the segmentation is too rough to exhibit semantic coherence. In the Vision-based Page Segmentation algorithm [Cai 2003], various cues, such as font, color, and size, are taken into account to achieve a more accurate content structure on the semantic level.

All above three algorithms work well on the structure-rich Web pages such as www.yahoo.com but they do not work on the content-rich Web pages such as
the reason is that current Web page structural segmentation algorithms can not group multiple paragraphs according to topic.

1.3 Advantages of the New System

The new system helps in information retrieval by segmenting Web pages into different blocks containing several blocks of different information by which the main content get separated and then partition it into subtopics. In this project, the TextTiling algorithm is considered, which uses lexical cohesion relations and partition the plain text into different subtopic structure faster so that it is easier for the user to extract information. The information retrieval algorithms can use subtopic structuring to return meaningful portions of the text. The new system takes into account the logic behind previous methods but instead of dividing the Web page into DOM nodes, it first partitions the Web page into visual and semantic-related blocks based on VIPS Method and only the blocks containing useful information are extracted for the further topic segmentation. Later, visual similarities are calculated and added to the lexical similarities to get the new boundaries between different topics. The experimental results show that the new method gives better results than the existing methods.

The following chapters discuss the project’s implementation from both the user’s and programmer’s view. Chapter 2 gives a detailed description of the three methods used for the Web page segmentation. Different screen-shots are used to aid in understanding the Web page segmentation process. Chapter 3 gives the system design. This section describes the analysis and implementation of the design to achieve the results. Chapter 4
describes the evaluation and results of the system. Chapter 5 discusses the possible future work and finally, Chapter 6 gives the conclusion of the project.
2. CONTENT-RICH WEB PAGE SEGMENTATION (CWPS)

2.1 Overview

Figure 2.1 describes the procedure of the content-rich Web page segmentation method (CWPS), which is composed of three procedures: structural segmentation, visual similarity calculation and topic segmentation.

In the structural segmentation, a page is partitioned into information blocks using VIPS (Vision Based Page Segmentation Algorithm). Then the noisy blocks are removed and text blocks are concatenated. The noisy blocks are generally advertisements, navigational links and blanks etc., so they can be removed automatically. Text blocks contained long text information and usually are next to each other. This project works only for Web pages containing long textual information.

![Figure 2.1 Web Page Segmentation Method](image)

Figure 2.1 Web Page Segmentation Method
In the visual similarity calculation the visual cues are detected at the beginning and the end of the paragraphs. The visual cues considered are position, color, font-size, font-weight, etc. The visual similarity values calculated based on various visual cues.

In the topic segmentation, first text blocks are grouped into paragraphs because of the assumption made that all the sentences in the same paragraph discuss the same topic. After this, all HTML tags are removed and the text blocks are segmented using a lexical cohesion method such as TextTiling to detect the subtopic boundaries.

2.2 VIPS Algorithm

In this section, the VIPS algorithm is explained. The VIPS method works for extracting the semantic structure of the Web page. The semantic structure is in the form of a hierarchical structure in which each node corresponds to a block. The Degree of Coherence (DoC) is assigned to all the nodes to measure how coherent the content in the block.

The vision based page segmentation structure of the page is a combined effect of the DOM (Document Object Model) structure and visual cues. The Document Object Model (www.w3.org/DOM) is a standard for creating and manipulating in-memory representations of HTML (and XML) content [Gupta 2003].

The basic objects in the DOM tree are defined as leaf nodes. In the VIPS method every node is called as block and a block may contain more than one leaf nodes in the DOM tree. Nodes in the DOM tree and vision based content structure do not correspond to each other. For each block, there is a degree of coherence (DoC) value to measure how coherent the block is. In the tree structure the DoC value of a child is always greater than
that of its parent. In this algorithm, DoC values are in the range of 1 to 10. This vision based content structure is to provide semantic partitioning. Every node of the DOM tree conveys certain semantics.

The VIPS page segmentation structure is shown in Figure 2.2. It takes place in three steps, initially visual block extraction, then visual separator detection, and finally the content structure construction. The VIPS method uses the page layout feature from the DOM method [Chen 2001]. After building the DOM tree the suitable blocks are extracted then the separators between these blocks are detected. Separators denote the horizontal or vertical lines in a Web page that visually cross with none of the blocks [Cai 2003]. Next the three steps are introduced one by one.

![Figure 2.2 Vision Based Page Segmentation Algorithm [Cai 2003]]
2.2.1 Visual Block Extraction

A DOM tree has several nodes that are visual blocks. Every node in the DOM tree represents a visual block. HTML tags such as <TABLE>, <P> in the DOM tree are huge nodes and they were used only for organization purpose and not appropriate to represent a single block. These nodes do not represent a single visual block because of the big structure. If the blocks are nested they are divided into sub blocks, so that the nodes are divided into their children. HTML grammar is flexible; therefore many pages do not obey HTML specification. Each node that represents a visual block, a DoC value is set according to the block’s intra visual difference. This process is repeated for all the nodes in the block. The algorithm for the block extraction process is mentioned below.

Algorithm DivideDomtree(pNode, nLevel)
{
    IF (Dividable(pNode, nLevel) == TRUE){
        FOR EACH child OF pNode {
            DivideDomtree(child, nLevel);
        }
    } ELSE {
        Put the sub-tree (pNode) into the pool as a block;
    }
}[Cai 2003]

There are some heuristic rules to check whether the node has to be expanded further. If the node has to be divided then the DoC value is updated to the block accordingly. The heuristics rules are listed in Table 2.1 (a) and Table 2.1 (b).
<table>
<thead>
<tr>
<th>Rules</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1</td>
<td>If the DOM node is not a text node and it has no valid children, then this node cannot be divided and is cut.</td>
</tr>
<tr>
<td>Rule 2</td>
<td>If the DOM node has only one valid child and the child is not a text node, then divide this node.</td>
</tr>
<tr>
<td>Rule 3</td>
<td>If the DOM node is the root node of the sub-DOM tree (corresponding to the block), and there is only one sub DOM tree corresponding to this block, divide this node.</td>
</tr>
<tr>
<td>Rule 4</td>
<td>If all of the child nodes of the DOM node are text nodes or virtual text nodes, do not divide the node.</td>
</tr>
<tr>
<td></td>
<td>1. If the font size and font weight of all these child nodes are same, set the DoC of the extracted block to 10.</td>
</tr>
<tr>
<td></td>
<td>2. Otherwise, set the DoC of this extracted block to 9.</td>
</tr>
<tr>
<td>Rule 5</td>
<td>If one of the child nodes of the DOM node is line-break node, then divide this DOM node.</td>
</tr>
<tr>
<td>Rule 6</td>
<td>If one of the child nodes of the DOM node has HTML tag <code>&lt;HR&gt;</code>, then divide this DOM node</td>
</tr>
<tr>
<td>Rule 7</td>
<td>If the sum of all the child nodes’ size is greater than this DOM node’s size, then divide this node.</td>
</tr>
</tbody>
</table>
Table 2.1 (b) Heuristics Rules in Block Extraction Phase [Cai 2003]

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 8</td>
<td>If the background color of this node is different from one of its children’s, divide this node and at the same time, the child node with different background color is not be divided in this round. Set the <strong>DoC</strong> value (6-8) for the child node based on the html tag of the child node and the size of the child node.</td>
</tr>
<tr>
<td>Rule 9</td>
<td>If the node has at least one text node child or at least one virtual text node child, and the node's relative size is smaller than a threshold, then the node cannot be divided. 1. Set the <strong>DoC</strong> value (from 5-8) based on the html tag of the node</td>
</tr>
<tr>
<td>Rule 10</td>
<td>If the child of the node with maximum size are small than a threshold (relative size), do not divide this node. 1. Set the <strong>DoC</strong> based on the html tag and size of this node.</td>
</tr>
<tr>
<td>Rule 11</td>
<td>If previous sibling node has not been divided, do not divide this node</td>
</tr>
<tr>
<td>Rule 12</td>
<td>Divide this node.</td>
</tr>
<tr>
<td>Rule 13</td>
<td>Do not divide this node. 1. Set the <strong>DoC</strong> value based on the html tag and size of this node.</td>
</tr>
</tbody>
</table>

Different HTML tags have different rules that are listed below in Table 2.2. R1 to R13 are the rules mentioned in Table 2.1 respectively. The first column in the Table 2.2 represents different HTML tags. For example, an **Inline Text Node shown** in Table 2.2 has rule 12 by which it gets divided further into sub nodes.
2.2.2 Visual Separator Detection

In page segmentation the VIPS algorithm is used for detecting the noise (navigational link, search bars, advertisements etc). After the visual block extraction, each block goes through the visual separator detection. These visual separators are horizontal or sometimes vertical lines in the Web pages that do not cross any blocks. These visual separators are good detectors for showing different semantics in the Web pages. Figure 2.3 illustrates an algorithm for visual separator detection.

1. First the separator list is initialized. The list starts with one separator whose starting pixel and ending pixel are the borders of the Web page.

2. Every block is checked in the separator and then separator was split.

3. If the block is crossing the separator, its parameters are updated.

4. If the block is covering the separator then separator is removed.

5. The separators at the border of the block are removed.
Figure 2.3 depicts an example in which dark colored blocks represents the visual blocks in the selected Web page. The process to detect the horizontal separators is as follows: consider Figure 2.3 (b), the page has only one separator, \( S_1 \) which is the whole pool. When the first visual block in Figure 2.3 (a) was put into the pool the separator get split into separator \( S_1 \) and \( S_2 \) as shown in Figure 2.3 (c). When second and third visual block in Figure 2.3 (a) are put into the pool then the above process repeats as shown in Figure 2.3 (d). When the fourth visual block is put into the pool, it crosses the separator \( S_2 \) and covers the separator \( S_3 \), which results in updating separator \( S_2 \) and removing separator \( S_3 \). At the end the separators at the border of the pool \( S_1 \) and \( S_4 \) are removed as shown in Figure 2.3 (e).
2.2.3 Setting Weights for Separators

The separator detection process is further used for setting the weights for these separators. These separators are used to distinguish blocks with different semantics. The weights of the separators are based on the visual difference between its neighboring blocks. Figure 2.4 depicts the visual block tree for the sample Web page layout. Visual blocks in the selected Web page are denoted as VB1, VB2 and VB3 etc. Blocks VB1 to VB2_3_2 contain different text information with the HTML tags. Depending on the HTML tags weights of the separators inside the blocks were determined.

![Figure 2.4 Visual Block Tree of the Web Page](image-url)
There are some rules to set up weights for the separators which are used to
differentiate blocks with different semantics. The rules are:

1. The greater the distance between blocks on different sides of the separator, the higher
   the weight of the separator.

2. If a visual separator is overlapped with some certain HTML tags (e.g., the <HR>
   HTML tag), its weight is set to be higher.

3. If background colors of the blocks on two sides of the separator were different, the
   weight is increased.

4. For horizontal separators, if the differences of font properties such as font size and
   font weight are bigger on two sides of the separator, the weight is increased.
   Moreover, the weight is increased if the font size of the block above the separator is
   smaller than the font size of the block below the separator.

5. For horizontal separators, when the structures of the blocks on the two sides of the
   separator are very similar (e.g. both are text), the weight of the separator is decreased
   [Cai 2003].

### 2.2.4 Content Structure Construction

As explained in the earlier sections the separators are detected and weights of the
separators are set. Content structure construction is based on the separators. The
construction starts with the separator with the lowest weight and the blocks beside the
separators are merged to get new blocks and this process repeats until the blocks beside
separators with highest weights are merged. The DoC for each new block gets assigned
based on the maximum weight of the separators.
In this project VIPS is used as an automated process for performing the initial structural segmentation part. The Web page gets partitioned based on the visual separators and structured as a hierarchy. When the structural segmentation is done the text blocks are extracted and given for further topic segmentation.

2.2.5 An Example of VIPS Structural Segmentation Analysis

In the Web page the noise is generally contained within one block. Figure 2.5 (a) shows a sample Web page selected for the VIPS analysis and Figure 2.5 (b) shows the page can be easily divided into four different blocks numbered 1, 2, 3 and 4 respectively. Block 1, 2 and 4 contain the noisy information such as navigational links, search bar, advertisements etc. they can be removed. Block 3 contains the textual information which is extracted out and given for the further topic segmentation. The block extraction process starts from considering the root node as block number 1. All the blocks are checked by comparing their DoC values.
Central Valley Basin

Any river is really the summation of the whole valley. To think of it as nothing but water is to ignore the greater part.

—Hal Jordan, This River This Valley

The vast Central Valley drains nearly two-thirds of California and is dominated by the state’s largest rivers, the Sacramento and San Joaquin. The Sacramento River runs north to south, and the San Joaquin runs south to north. They meet in the Delta, a maze of channels and sloughs that eventually flows into San Francisco Bay and the Pacific Ocean to the west. These two waterways have tremendous ecological and economic significance. The Central Valley hosts more than 1.5 million ducks and 75,000 geese in its seasonal marshes along the Pacific Flyway. Oak woodlands, dense riparian forests, vernal pools, extensive grasslands, and freshwater marshes once dominated the valley landscape (1).
Figure 2.6 depicts the DOM tree associated with each block of the selected Web page corresponding to the Web page in Figure 2.5. Then visual extraction process starts from the root node, shown as node P in Figure 2.6, and based on visual cues it goes on till all the nodes (node 1, 2, 3 etc.) of the DOM tree are covered. A DoC value to each block (node 1, 2, 3 etc in Figure 2.6) is assigned according to the block’s visual property. All the blocks in the same page are extracted and then put aside. The separators among these blocks identified and weight of the separator is set depending on the properties of the neighboring blocks as well as the layout hierarchy constructed (explained in later subsection). At the end, as per the hierarchy, each block’s content structure is checked and then the leaf node is treated as the sub page till all the nodes in the DOM tree are divided. The root node of the vision-based page structure is indicated by P, which is divided repeatedly with no further division is possible for any nodes.

Figure 2.6 DOM Tree
The screenshot of the structural segmentation VIPS Method analysis is shown in Figure 2.7. The method splits the Web page into different blocks and removes noise (navigational links, advertisements, search bars) eventually. For example, VB 1-2-2-1-1(5) in Figure 2.8 is the block with the main plain text. VIPS: 10 is the Degree of Coherence (DoC) for that certain page analysis. In this project, a DoC is rated on the scale of 1 to 10, 1 is the minimum and 10 is the highest. That means if the DoC is chosen as 1 then it will give whole Web page as one block and if the DoC is increased it segments the Web page into further sub blocks. The text blocks then can be identified and extracted for further segmentation. The text block with long text information and HTML nodes consisting of tags <P>, <BR>, <TEXT> etc was also extracted out for using in other module.

Figure 2.7 Screenshot of VIPS Method for a Selected Doc Value
2.3 The TextTiling Algorithm

TextTiling is an algorithm for partitioning full-length text documents into coherent multi-paragraph units that correspond to a sequence of subtopic passages. The algorithm assumes that a set of words are in use during the course of a given subtopic discussion, and when that subtopic changes, a significant proportion of the vocabulary changes as well [Hearst 1994]. TextTiling uses the lexical connectivity to approximate the subtopic structure of the document. The term *tile* is used to reflect the pattern of subtopics in the expository text. This algorithm is based on lexical analysis to classify the tiles with respect to their topic.

This algorithm discovers the subtopic structure for the plain text only. This method compares the given window size and measure the lexical similarity between different blocks. If the two blocks are similar there is no change in the subtopic structure but if the two blocks are not similar then there is necessarily a subtopic structure involved. This method automatically subdivides the text into a multi-paragraph structure that gives the subtopic structure to a Web page. In this method the structure of plain text which is discussing about one main topic is divided into series of subtopic discussions.

TextTiling works on articles, science magazines or any information which is plain text. For example, consider an article describing the main topic as “evolution of human”. The article has 25 paragraphs. Table 2.3 explains how a reader divided the main topic into a series of subtopics each discussing different information. First column represents the paragraph range which talks about the topic mentioned in the second column of Table 2.3. For example, paragraphs 1 to 3 talks about “introduction to human life”, paragraphs
4 to 6 talk about “history of man”, paragraphs 6 to 10 talk about “the evolution of the brain” etc.

Table 2.3 Example Page segmentation topics of sample plain text

<table>
<thead>
<tr>
<th>Paragraph</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>Introduction to human life</td>
</tr>
<tr>
<td>4-6</td>
<td>History of man</td>
</tr>
<tr>
<td>6-10</td>
<td>The evolution of the brain</td>
</tr>
<tr>
<td>11-16</td>
<td>The neanderthal</td>
</tr>
<tr>
<td>17-20</td>
<td>How evolution works</td>
</tr>
<tr>
<td>21-25</td>
<td>Summary</td>
</tr>
</tbody>
</table>

TextTiling works similar to the example mentioned above. The structure of an expository text is characterized as a sequence of subtopic discussions that occur in the context of one or more main topic discussions. TextTiling uses patterns of lexical coherence. TextTiling method is distributed into three parts.

1. Tokenization into terms and sentence-sized units,
2. Determination of a score for each sentence-sized unit, and
3. Detection of the subtopic boundaries [Hearst 1994].

2.3.1 Subtopic Structure Detection via Lexical Coherence

The segmented passage structure in TextTiling method means the pieces of text that describe about changes in the subtopic of the main content information. Some words occurring frequently throughout the text (for example, human, life etc.) indicate the main
topic of the text. Words that are less frequent but more uniform in distribution, such as *form* and *scientist*, tend to be neutral and do not provide much information about the divisions within the discussions. The remaining words are the most important in deciding the subtopic structure and they are grouped together. Because, the words, that do not occur frequently but are closely related conceptually, tend to work together to form a subtopic structure.

2.3.2 Topic Segmentation Analysis

Once the structural segmentation is done and the long text is extracted out it is ready for the topic segmentation. In topic segmentation, first extracted text is delimited into paragraphs because the assumption is that sentences in the same paragraph cover same subtopic and do not partition the paragraph in the middle. In this segmentation method only plain text is subdivided into series of topics. In Figure 2.8 shows output in the form of `<TILE 8 - FILE: 1.ref START: 11088 END: 12035>` where, TILE 8 is one of the segmented subtopics, FILE: 1.ref shows the plain text file given as input, the boundaries for that particular tile are shown as START: 11088 and END: 12035 (starting and ending offset of the respective subtopic). The sample screenshot is shown in Figure 2.8.
Figure 2.8 Screenshot of the TextTiling Method Output

This method works on the expository text which is not heavily stylized or structured and does not consider the visual cues such as headings, subheadings etc.
2.4 The CWPS Method

As the TextTiling algorithm only works on the plain text files and does not work well on segmenting the Web pages. The CWPS method detects various visual cues depending on different HTML tags and later the visual similarities are calculated to check the visual similarity between the paragraphs. The design process of the calculation of visual similarity values is shown in the next section.
3. SYSTEM DESIGN

3.1 System Analysis

The Design and Analysis Phase of the project to develop the content-rich Web page segmentation (CWPS) method started with the following steps:

1. Discussed with Dr. LongZhuang Li the type of features needed for the CWPS. The features discussed included user interface, and data presentation.

2. Designed a framework of what the user interface would look like, proposed, and analyzed the design with Dr. LongZhuang Li. Dr. LongZhuang Li provided me with the resource material that gave me enough idea to get started with the project, helped me with the background information for the existing system and explained to me the need for this new system and the scope of this project.

3. Designed and developed visual similarity algorithm based on rules, assumptions.

4. Researched various Web page segmentation systems to choose which software to use in the development of the CWPS.

5. Discussed with Dr. LongZhuang Li the advantages of using the latest development system, C#.NET.

6. Talked with Dr. LongZhuang Li about the current method used by the researchers to segment the topic structure.

The following steps were taken to complete the development and testing of the CPWS:

- Implemented the system using Visual C++, C#.NET and Linux.
- Installed and tested the system with real Web pages.
- Compared the results with the existing system.
3.2 System Requirements

The system requirements for the CWPS are described in this section. The application needed the Visual Studio software to implement the structural segmentation. Linux environment was used for the topic segmentation.

Microsoft Visual C++ is an integrated development environment (IDE) product for the C, C++, and C++ programming languages engineered by Microsoft. It has tools for developing and debugging C++ code, especially that written for the Microsoft Windows API, the DirectX API, and the Microsoft .NET Framework.

3.3 System Architecture and Overview

The CWPS architecture consists of three tiers: structural segmentation, visual similarity calculation and topic segmentation. Figure 3.1 shows a graphical representation of the architecture. The functional description of each tier in the architecture is given below:

- **First Tier**: This first partitions the Web page into information blocks. Then noisy blocks are identified and removed and the text blocks are concatenated. This can be automated because usually the noisy blocks contain mostly links and short irrelevant information and the text blocks contain long textual information, are contiguous and next to each other. VIPS algorithm is used to perform the structural segmentation.

- **Second Tier**: The second tier consists of calculating the visual similarity values between the paragraphs of the extracted text block. The plain text with HTML tags is given as an input and generated visual similarity values.
• **Third Tier:** The extracted text from the structural segmentation (VIPS) is delimited into paragraphs. The visual similarity values are added to the lexical similarity values calculated by the TextTiling algorithm. The plain text is given as an input to the TextTiling algorithm to get the new subtopic boundaries.

![Figure 3.1 System Architecture](image-url)
3.4 Structural Segmentation

The VIPS algorithm is adopted to conduct the initial Web page structural segmentation. To be specific, a Web page is segmented into different blocks and noisy blocks are identified and removed and text blocks are concatenated. Then only text blocks are extracted and combined for further topic segmentation. The extracted text blocks have long textual information including images, tables, lists, formula, and program source code etc. It contains nodes with tags such as <P>, <BR>, <STRONG>, <I>, <B> etc., which is used to specify the display style of the text.

3.5 Visual Similarity Method

This subsection details about extraction of the visual cues and calculation of the visual similarities between the paragraphs. The visual cues are detected based on various HTML tags such as <P>, <I>, <B> etc. Visual similarity values are calculated depending upon the visual cues. The visual similarities values are combined with the lexical cohesion method to get the similarity between the paragraphs.

3.5.1 Visual Cue Extraction

The visual similarity, sim$_{vis}$, is used to measure the similarity between two paragraphs. sim$_{vis}$ has the following properties:

1. Ranges between 0 and 1 inclusive.
2. The larger the sim$_{vis}$ value, the more the two paragraphs resemble each other.
3. The sentences within the same paragraph have sim$_{vis}$ value of 1.
In this project the paragraph is not divided in the middle, so that any changes made inside the paragraph are ignored. Some heuristics are mentioned below to measure the similarity between the paragraphs in different paragraphs.

1. `<P>` tag is used to start a new paragraph.
2. `<BLOCKQOUTE>` and `<PRE>` tags are used to format and delimit closely related topics. All the text inside these paragraphs is considered in the same paragraph.
3. If a table is inside `<P>`, then sentences inside the table are considered as same paragraph.
4. Sentences which are separated by `<BR>` with no tags in between lines are considered as same paragraph.
5. Sentences inside `<DL>` and `</DL>` tags are considered in the same paragraph.
6. Texts inside the tags `<UL>` or `<SL>` are either summary or characteristics of the same topic or may be the highlights of the Web page. All the sentences in the paragraph and above the paragraph are considered as same.

The assumption made was: A change of a page layout or style of text is a change in the topic. To enhance subtopic partition process, topic cues and pseudotopic cues were used. A topic cue indicates that it is possible the cue starts a new subtopic. In topic cues, navigational links generally refers to a subtopic so that the subtitles can be considered for calculation of the $\text{sim}_{vis}$ values.

There are some rules for the topic and subtopic cues which are discussed next.

1. Topic navigational links refer to subtopic of the document, they provide a useful guideline on the partition of the Web document. These links are known as t-links and can be identified by comparing them with the subtitles. Also they are nested.
2. Long text has their own segmentation types such as titles, subtitles. Titles have different size, font, and color from other text; also subtitles have different size or font from other text. The subtitle cues are from <H1> to <H6> and same as the t-links they are nested.

3. <HR> tag is generally used to separate different topics.

The rules for the Pseudotopic cue are explained next. Pseudotopic cue is different from the topic cues in that it starts the new subtopic but it might be just for the display purpose.

1. At the beginning of each paragraph if the font, size, color of the text is different then it may be a start of a new topic. sim\textsubscript{vis} values for these cues are increased by 0.1.

2. If the sentences are separated by more than two successive <BR> or also <P></P> then increase the sim\textsubscript{vis} value by 0.1 because spaces in the between might have two different paragraphs.

3. If the table has cell size larger than the threshold then the table might have different paragraphs. sim\textsubscript{vis} values between the cells are increased by 0.1.

An example file for HTML tag structures is shown in Figure 3.2. This file contains text plus the HTML tags. Depending on the HTML tags the visual cues are detected and later the visual similarity values between the paragraphs are calculated. The visual similarity matrix is generated from the different visual similarity values. For example, the matrix generated will be 6 x 6 if there are 6 paragraphs in a text file and so on. The example describes next considers that the text file contains 7 paragraphs. It generates (7 x7) matrix as shown in Figure 3.3.
For example, a sample text block extracted from the VIPS contains the following tag information as shown in Figure 3.2, `<H1> <P1> <HR> <H2> <P2> <B> <P3>`
At the beginning all the paragraphs have same \( \text{sim}_{vis} \) value of 0.25 as shown in Figure 3.3 (a). When the topic cue <HR> is detected the \( \text{sim}_{vis} \) values are set to 0.05 paragraphs <P1> to <P7> as shown in Figure 3.3 (b). Then topic cue <H2> and <H3> are identified then \( \text{sim}_{vis} \) values between <H2><P2><P3><P4><P5> and <H3><P6><P7> are decreased by 0.1 and \( \text{sim}_{vis} \) values inside these two subtopics are increased by 0.25 as shown in Figure 3.3 (c). Then two pseudotopic cues are found <P2><B> and <P4><B> and their \( \text{sim}_{vis} \) values are increased by 0.1 respectively as shown in Figure 3.3 (d).

Figure 3.3 Visual Similarity Matrices
The visual similarity matrix between paragraphs is calculated using the following algorithm:

Algorithm ComputeVisualSimilarity ()
{
1. organize the document into paragraphs based on the paragraph cues
2. sim_{vis} values are initialized to 0.25
3. if (HasTopicCue() == TRUE)
   {
   1. sim_{vis} values of paragraphs separated by <HR> are reduced to 0.05
   2. for each level of subtitles from the outermost to the innermost
      {
      1. sim_{vis} values of paragraphs under different subtitles are reduced by 0.1
      2. sim_{vis} values of paragraphs under the same subtitle cues are increased by 0.25.
      }
   }
4. If (HasPsuedoTopicCue() == TRUE)
   {
   1. Assign different Visual Similarity according to proposed rules.
   }
}

### 3.6 Calculating Similarity between the Paragraphs

When the visual similarity matrix is built, the visual similarity values are combined with lexical cohesion values of text to detect topic boundary. The text with the html tag information is cleared by taking away all HTML tags. The similarity between two paragraphs $p_i$ and $p_j$, $\text{sim}(p_i,p_j)$ is calculated as weighted sum of the visual similarity $\text{sim}_{vis}(p_i,p_j)$

$$\text{sim}(p_i,p_j) = \alpha \cdot \text{sim}_{vis}(p_i,p_j) + ((1- \alpha) \cdot \text{sim}_{lex}(p_i,p_j)) \quad \text{--- (3.1)}$$

In equation (3.1) $\text{sim}(p_i,p_j)$ is the similarity between the two paragraphs $p_i$ and $p_j$, $\alpha$ is a constant considered as 0.5, $\text{sim}_{vis}(p_i,p_j)$ is the visual similarity value calculated
by the ComputeVisualSimilarity algorithm and \( \text{sim}_{\text{lex}}(p_i, p_j) \) is the lexical coherence value between the two paragraphs \( p_i \) and \( p_j \).

Once the Web page is done with the structural segmentation procedure and the noisy blocks are removed from the Web page. An extracted text file was given as an input and segmented text document is obtained with the subtopic number, file name, starting and ending offsets of the subtopic.
4. EVALUATION AND RESULTS

In this chapter, the prototype, content-rich Web page segmentation (CWPS) was tested and evaluated. The evaluation had two purposes: (1) to examine how well the project was implemented and (2) to analyze the integration of the system with content-rich Web pages containing long text information.

Windows XP environment was used for the first part of the project on structural segmentation. For the topic segmentation Linux environment is used. In structural segmentation the DoC value is set to 10. The α value of Equation 1 in Chapter 3 is set to 0.5. In the TextTiling method, the default settings were as follows:

- k: number of token-sequences to compare against at each gap (default = 6)
- w: width of the token-sequences (default = 20)
- n: number of iterations of median smoothing (default = 1)
- b: width of median smoothing (default = 2)
- p: mark boundaries between token sequence gaps between paragraphs. Marking between paragraphs is the default.

In our experiments, The CWPS Method was compared with the TextTiling given the same text file as input. The method is considered better if its results are closer to the manually segmented results.

In the experiments, ten different Websites were tested and evaluated. In table 4.1 the first column represents the Urls for the ten Web pages, second column represents the narrative of the Web page content and the third column represents the number of paragraphs they contain.
Table 4.1 Sample Web Pages

<table>
<thead>
<tr>
<th>Url</th>
<th>Narrative</th>
<th>Paragraphs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) <a href="http://press.princeton.edu/chapters/s8334.html">http://press.princeton.edu/chapters/s8334.html</a></td>
<td>Biodiversity</td>
<td>26</td>
</tr>
<tr>
<td>2) <a href="http://www.tpl.org/tier3_cdl.cfm?content_item_id=9460&amp;folder_id=1685">http://www.tpl.org/tier3_cdl.cfm?content_item_id=9460&amp;folder_id=1685</a></td>
<td>Valleys</td>
<td>21</td>
</tr>
<tr>
<td>3) <a href="http://www.w3.org/WAI/intro/accessibility.php">http://www.w3.org/WAI/intro/accessibility.php</a></td>
<td>Web Standards</td>
<td>16</td>
</tr>
<tr>
<td>4) <a href="http://www.small-business-software.net/what-are-content-sites.htm">http://www.small-business-software.net/what-are-content-sites.htm</a></td>
<td>Business Software</td>
<td>18</td>
</tr>
<tr>
<td>5) <a href="http://www.fas.harvard.edu/home/academics/info.html">http://www.fas.harvard.edu/home/academics/info.html</a></td>
<td>Academic Information</td>
<td>25</td>
</tr>
<tr>
<td>6) <a href="http://www.worldtravelguide.net/country/120/country_guide/Indian-Subcontinent/India.html">http://www.worldtravelguide.net/country/120/country_guide/Indian-Subcontinent/India.html</a></td>
<td>India Guide</td>
<td>20</td>
</tr>
<tr>
<td>7) <a href="http://www.indiatravel.com/">http://www.indiatravel.com/</a></td>
<td>India Travel</td>
<td>18</td>
</tr>
<tr>
<td>8) <a href="http://www.indiaettravel.com/">http://www.indiaettravel.com/</a></td>
<td>Travel</td>
<td>22</td>
</tr>
<tr>
<td>9) <a href="http://www.indianchild.com/diwali.htm">http://www.indianchild.com/diwali.htm</a></td>
<td>Festival</td>
<td>12</td>
</tr>
<tr>
<td>10) <a href="http://www.cultureofindia.net/">http://www.cultureofindia.net/</a></td>
<td>Culture</td>
<td>16</td>
</tr>
</tbody>
</table>
Table 4.2 shows the results and comparison for different content-rich Web pages with number of subtopics found in Manual segmentation method, TextTiling method and CWPS method. The first column represents the Urls of the Web pages shown in table 4.1, the second column represents number of subtopics segmented by the Manual method, the third column represents number of subtopics segmented by the TextTiling method and the forth column represents number of subtopics segmented by the CWPS method respectively. In table 4.2

- C is the number of complete match of subtopic segments between the Manual and the TextTiling Method as well as between the Manual and the CWPS Method.
- P is the number of partial match of subtopic segments between the Manual and the TextTiling Method as well as between the Manual and the CWPS Method.
- N is the number of no match of subtopic segments between the Manual and the TextTiling Method as well as between the Manual and the CWPS Method.

The second last row of Table 4.2 shows the total number of subtopics segmented by the Manual method, the completely matched, partially matched and no matched subtopics by the TextTiling method and same for the CWPS method. The last row of Table 4.2 shows the ratio 0.64 of completely matched subtopics in the CWPS method is higher than 0.42 of the TextTiling method and the ratio 0.38 of no matched subtopics in the TextTiling method is much higher than 0.094 in the CWPS method which shows that the CWPS method get superior results than the TextTiling. The ratio 0.26 of partially matched subtopics in the CWPS method is higher than 0.18 of the TextTiling method. This may be because of the constant value of $\alpha = 0.5$ and visual similarity computation algorithm is not accurate enough.
Table 4.2 Comparison between the methods

<table>
<thead>
<tr>
<th>Url</th>
<th>Number of Manual subtopics</th>
<th>Number of TextTiling Subtopics</th>
<th>Number of CWPS Subtopics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>C = 3</td>
<td>P = 2</td>
<td>N = 7</td>
</tr>
<tr>
<td></td>
<td>C = 8</td>
<td>P = 2</td>
<td>N = 2</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>C = 3</td>
<td>P = 1</td>
<td>N = 6</td>
</tr>
<tr>
<td></td>
<td>C = 8</td>
<td>P = 1</td>
<td>N = 1</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>C = 3</td>
<td>P = 2</td>
<td>N = 3</td>
</tr>
<tr>
<td></td>
<td>C = 7</td>
<td>P = 11</td>
<td>N = 0</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>C = 4</td>
<td>P = 2</td>
<td>N = 4</td>
</tr>
<tr>
<td></td>
<td>C = 6</td>
<td>P = 2</td>
<td>N = 2</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>C = 5</td>
<td>P = 3</td>
<td>N = 4</td>
</tr>
<tr>
<td></td>
<td>C = 8</td>
<td>P = 2</td>
<td>N = 2</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>C = 8</td>
<td>P = 2</td>
<td>N = 6</td>
</tr>
<tr>
<td></td>
<td>C = 12</td>
<td>P = 3</td>
<td>N = 1</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>C = 4</td>
<td>P = 3</td>
<td>N = 3</td>
</tr>
<tr>
<td></td>
<td>C = 8</td>
<td>P = 1</td>
<td>N = 1</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>C = 6</td>
<td>P = 3</td>
<td>N = 3</td>
</tr>
<tr>
<td></td>
<td>C = 8</td>
<td>P = 3</td>
<td>N = 1</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>C = 5</td>
<td>P = 1</td>
<td>N = 2</td>
</tr>
<tr>
<td></td>
<td>C = 6</td>
<td>P = 1</td>
<td>N = 1</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>C = 4</td>
<td>P = 1</td>
<td>N = 3</td>
</tr>
<tr>
<td></td>
<td>C = 7</td>
<td>P = 1</td>
<td>N = 0</td>
</tr>
<tr>
<td>Total Manual = A =106</td>
<td>Total C = 45</td>
<td>Total P = 20</td>
<td>Total N = 41</td>
</tr>
<tr>
<td></td>
<td>C/A=0.42</td>
<td>P/A=0.18</td>
<td>N/A=0.38</td>
</tr>
</tbody>
</table>
The ten test Web pages were selected arbitrarily. The experimental results illustrate that the content–rich Web page segmentation (CPWS) Method can be used to segment any content-rich Web page that contains long text information and achieve better results than the existing best method, such as TextTiling.
5. FUTURE WORK

Content-rich Web Page Segmentation (CWPS) was successfully designed and implemented. The segmentation procedure was based on HTML tag structures and visual cues. As this system works only for content-rich Web pages that contain long text information, other features can also be added so that it works with short text paragraphs. In future Web pages with noisy information between the long textual information can be tested. The paragraphs in the input text file were separated by one or more blank lines, and in future this can be changed without any blank lines in between two paragraphs.
6. CONCLUSION

Segmentation for the content-rich Web pages was successfully designed and implemented. The segmentation procedure was based on HTML tag structures and visual cues. VIPS algorithm was adopted to conduct the initial Web page structural segmentation. Once the Web page was segmented into blocks, noisy blocks were removed to extract the useful information. All the text blocks were concatenated to form a long semi-structured file. The text was separated from the HTML tags. The text into the text blocks was segmented into different subtopics. Visual similarities were calculated and were combined with the lexical cohesion method TextTiling. The results were compared with manually segmented Web page and TextTiling method. This system works only for content-rich Web pages that contain long text information. The CWPS system was developed with utmost care. Visual Studio and C#.NET were the major tools used in this project.


APPENDIX A – SOURCE CODE

1. TILE.C

1.1 Show Text Function

```c
showtext(char *fname, TILEDOC *tdp)
{
    FILE *fp;
    int tx, c;
    long i;
    TILE *tp = tdp->tilearray;

    if ((fp = fopen(fname, "r")) == NULL) {
        fprintf(stderr, "tile: can't open %s: %s\n",
                fname, strerror(errno));
        return (1);
    }
    for (tx = 0; tx < tdp->numtiles; tx++) {
        printf("<TILE %d - FILE: %s START: %ld END: %ld>\n",
                tx, fname, tp[tx].startoff, tp[tx].endoff);
        fseek(fp, tp[tx].startoff, SEEK_SET);
        for (i = 0; i < tp[tx].endoff - tp[tx].startoff; i++) {
            c = getc(fp);
            if (c == EOF)
                break;
            (void)putc(c, stdout);
        }
        printf("</TILE %d - FILE: %s START: %ld END: %ld>\n",
                tx, fname, tp[tx].startoff, tp[tx].endoff);
    }
    fclose(fp);
    return (0);
}
```

43
1.2 Compute sim\textsubscript{vis} value Function

```c
compute_sim(int ss, int k) {
    double num, w1s, w2s, s1, s2, temp;
    int j;
    int ss2;
    AInfo *ip;
    Tcl_HashEntry *hp;
    Tcl_HashSearch hs, *sp;

    ss2 = ss + k;

    num = w1s = w2s = 0.0;
    sp = &hs;
    for (hp = Tcl_FirstHashEntry(TP, sp); hp;
         hp = Tcl_NextHashEntry(sp)) {
        s1 = s2 = 0.0;
        ip = (AInfo *)Tcl_GetHashValue(hp);
        wloc_to_sentarray(ip->wloc, ip->current_count);
        for (j = ss; j < (ss + k); j++) {
            s1 += Sentarray[j];
        }
        for (j = ss2; j < (ss2 + k); j++) {
            s2 += Sentarray[j];
        }
        num += s1 * s2;
        w1s += s1 * s1;
        w2s += s2 * s2;
    }
    temp = sqrt(w1s * w2s);
    w1[ss+k-1] = (temp) ? (num / temp) : 0.0;
}
```
1.3 Tile Offset Location Function

tile_locs()
{
    FILE * fp;
    SORTF *scores;
    int *used;
    register int i, j;
    int para, sentgap;
    long parabyte, prev;
    int count = 0;
    double avg, sd, limit;
    TILEDOC *tdp;    /* tiledoc return structure */
    TILE *tp;        /* pointer to tilearray in TILEDOC */
    long *bounds;
    double *sss;

    bounds = (long *)malloc(sizeof(long) * max_para);
    if (bounds == NULL) {
        fprintf(stderr, "Out of mammarys\n");
        exit(1);
    }
    bzero(bounds, sizeof(long) * max_para);

    sss = (double *)malloc(sizeof(double) * max_para);
    if (sss == NULL) {
        fprintf(stderr, "Out of memory\n");
        exit(1);
    }
    bzero(sss, sizeof(double) * max_para);

    scores = (SORTF *)malloc(sizeof(SORTF) * max_sent);
    if (scores == NULL) {
        fprintf(stderr, "Out of memory\n");
        exit(1);
    }
    bzero(scores, sizeof(SORTF) * max_sent);
    used = (int *)malloc(sizeof(int) * max_sent);
    if (used == NULL) {
        fprintf(stderr, "Out of memory\n");
        exit(1);
    }
    bzero(used, sizeof(int) * max_sent);

    determine_scores(w2, scores);

    qsort(scores, max_sent, sizeof(SORTF), compsf);
for (i = 0; i < max_sent; i++) {
    para = 0;
    sentgap = scores[i].sgap;
    parabyte = Paralocs[sentgap];
    for (j = 0; j < 6; j++) {
        if (Sentpara[sentgap+j] > 0) {
            para = Sentpara[sentgap+j];
            parabyte = Paralocs[sentgap+j];
            break;
        } else if (Sentpara[sentgap-j] > 0) {
            para = Sentpara[sentgap-j];
            parabyte = Paralocs[sentgap-j];
            break;
        }
    }
    if (scores[i].val > 0.0) {
        if ((used[para] != 1) && ((parabyte > startoftext) && (parabyte < endoftext))) {
            sss[para] = scores[i].val;
            bounds[para] = parabyte;
            count++;
        }
        used[para] = 1;
    } else {
        break;
    }
}

avg = 0.0;
sd = 0.0;
for (i = 0; i < max_para; i++) {
    avg += sss[i];
}
avg = avg / (double)(count ? count : 1);
for (i = 0; i < max_para; i++) {
    if (sss[i] > 0.0) {
        sd += ((sss[i] - avg) * (sss[i] - avg));
    }
}
if ((count - 1) <= 0)
    sd = 0;
else
    sd = sqrt(sd / (double)(count - 1));
limit = avg - (sd / 2.0);