一、緒論

1.1 研究背景及動機

隨著網際網路各項服務及應用的蓬勃發展，網路上的資料量也隨之以爆炸性的速度在成長，依據資訊科技與創新基金會(The Information Technology & Innovation Foundation，ITIF)提出的文件[3]，全球第一個.com 通用網域名稱 Symbolics.com 在 1985 年 3 月 15 日，美國麻州正式誕生，「.com」初期的起步速度相當緩慢，創立 2 年註冊的網站數才達到 100 個，其中不乏許多本身就有投入網際網路研究學術背景像是大學等單位；直到 1995 年網站數量才暴增為 1.8 萬個，到了 1997 年則上看 100 萬大關。25 年來共有 1.13 億個網站存在過，其中有 99.9% 是在近 15 年間成立的，歷經 21 世紀初期網路泡沫化之後，目前網站數量超過 8600 萬個，每月新增的「.com」網域名多達 66 萬 8000 個。為了有效地利用這個龐大的網頁資料，各種相關的研究便因運而生，其中與一般使用者最相關的莫過於搜尋引擎了。完整的搜尋引擎幾乎擁有全世界開放的網頁資料，且網頁的異動在幾天內也會隨之更新，例如 Google[2]，只要一個新的網頁放上網際網路，在一個禮拜內大多可以透過關鍵字的搜尋而找到。因此搜尋引擎適合在未知的來源中尋找所需的資料，使用者只要輸入適當的關鍵字，很容易透過它來得到相關的資訊。另外一個相關的應用是網頁資料萃取系統，使用者可以把要抓取的網頁資料事先設定，然後定時由系統代理使用者去取回所想要的資料，並存在本地端，可以免去枯燥的等待時間。像本實驗室所開發的 BODE(Browser-Oriented Data Extraction System)[26]即是此類的系統。毫無疑問地，如果可以將網際網路各個相關網站上的資訊加以收集、分析，這些整合後的資訊可以提供使用者更專業的服務，例如使用者透過一個 Multi-Search Engine，可以收集到多個搜尋引擎上的類似資料；Shopping Agent 可以讓使用者在購買網路上商品的時候，有較好的價格選擇與產品比較等。而不管是這類的資訊整合服務或搜尋引擎或資料萃取工具，要對複雜的網頁進行分析，首要就是對網頁作區塊分類或標記，以濾除噪音 (Noise) 區域及提取各主題(Topic)區域之本文區塊， 而對網頁作區塊分類或標記，就是所謂網頁區塊分割(Page Segmentation) 。

2003 年微軟團隊發表視覺化網頁分割演算法(Vision-based page segmentation: VIPS)[11]後， 很多網頁分割研究採用了視覺化網頁區塊分割為基礎，如 OLERA[12] ，ViPER[25]。 但在近年來網頁的頁面框架設計，越來越多採用 DHTML[4,5]技術為主時，原始的 VIPS 的方法在使用上，便出現當初設計時沒有顧及的小缺陷，雖然之後的研究，出現很多組合型態的頁面分割演算法來彌補使用上的不足，如 VSDR[22] ，Browsing on small screens [8]。但因為是採用其它特性的演算法來彌補或取代 VIPS， 所以這部份切割區塊也就喪失視覺化分割的特性。本文提出一個方法，在以視覺化分割為基礎上，帶入網頁文件布局(HTML Rendering-Based) 參考，以解決視覺化分割在 DHTML 網頁上，可能找不到視覺化分割線(Separator)的問題。

1.2 論文內容概述及大綱

本文的目的在解決視覺化網頁分割演算法，在 DHTML 網頁上可能找不到視覺化分割線的問題。

本論文各章節內容及說明茲分述如下：

第二章：研究內容與方法

介紹網頁區塊分割技術與相關研究，說明視覺化區塊分割技術 VIPS 相關背景技術，及 VIPS 在 DHTML 網頁上為什麼找不到視覺化區塊分割線的問題。

第三章：理論

說明為解決代表視覺化區塊上的 Node 位置之間，發生面積上的重疊而找不到在視覺上的分割線現象；所提出的改良方法：

1. 改良的視覺區塊擷取方法。
2. 利用 DHTML 文件布局特性所提出遮蔽區塊調整策略。

第四章：實驗部份

描述所實作之系統的開發環境與各模組功能，並以此系統來實際分析幾個測試資料 ，以評估此演算法的區塊分割程度。

第五章：結論

本文的結論與相關應用及未來工作。

二、研究內容與方法

2.1 網頁區塊分割

早期網際網路的相關應用，如 Web Information Retrieval 及 Web Data Extraction，都將網頁視為基礎單元。但網頁通常都是由不同的主題區塊所組成，舉例來說，一個商業性質的網頁，可能分成數個部份，如：導覽連結（Navigational Bars）、廣告（Advertisement Bars）、著作權聲明（Copyright）、 裝飾的圖文、焦點新聞與該頁面主題相關的內容區塊…等。且越來越多的網頁，其主題內容區塊有分成各小主題內容區塊的趨勢。所以要對網頁內容作有效的擷取，並濾除不要的區塊，就必需對網頁作區塊分類或標記，也就是網頁區塊分割(Page Segmentation)。下節介紹網頁區塊分割的相關技術。

2.2 相關研究

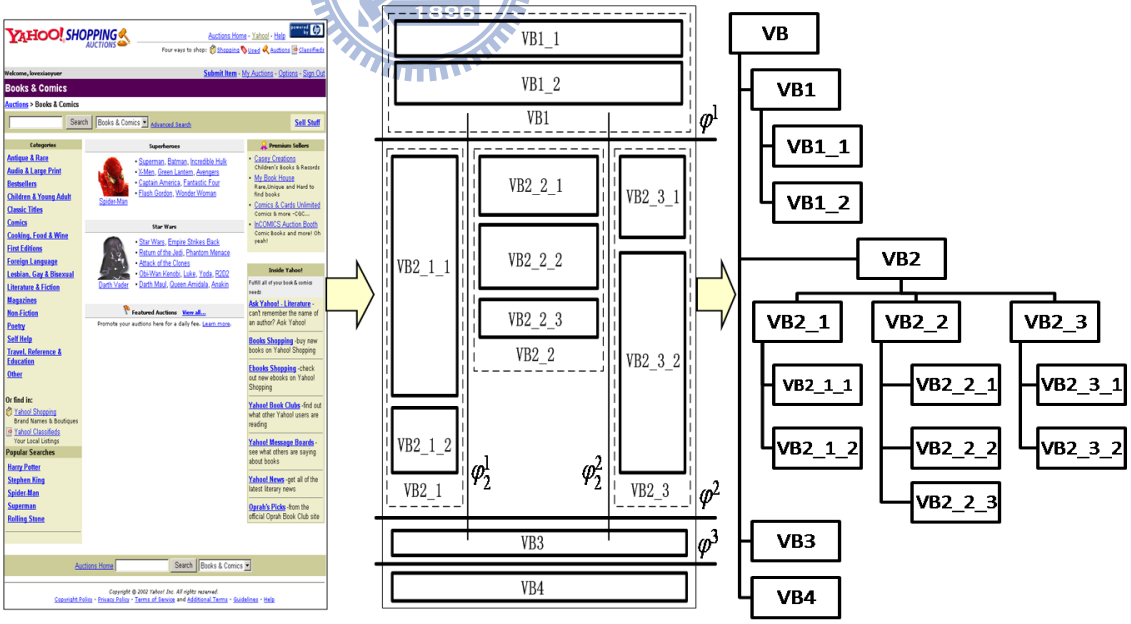
較早的網頁分割，單純的就「使用者感興趣的區域」及「使用者不感興趣的區域」來區分。如：「感興趣的區域」，以購物網站的網頁為例，列出一系列商品的名稱、價錢、商品狀況…等資訊的部份。「不感興趣的區域」則是網站上的瀏覽列、琳瑯滿目的廣告、著作權聲明…等。而網頁內容的擷取就針對 「感興趣的區域」來設計， 如 Function-based Object Mode (FOM)[14] 試圖定義一個基本物件 (Object)；如展示物件 (Presentation)、語義要素物件 (Semanteme)、 裝飾物件 (Decoration)、超連結物件 (Hyperlink)、互動物件 (Interaction)等，借由基本物件來定義物件的延伸規則(object function) ，進來找出其他同性質的物件， 如 圖 2.1 中， 紅線所框出來的區域，就是使用者可能感興趣的區域。

此類型的方法 ，主要是出資料呈現的規則，而透過擷取規則 (Extraction rule) 來擷取資料。也就是說: 依據 HTML Tag 的前後排列順序，去分辨是否為使用者所需要資訊。但此類方法需要耗費人工 (Manual) 去設定擷取規則 ，後來研究發現，網頁的內容呈現方式具有一定的規則，所以只需找出資料呈現的規則，加以歸納成擷取規則，便能讓擷取程式利用擷取規則，去擷取網頁資訊；這種方式稱為半自動 (Semiautomatic) 的方法，又稱為包覆器(wrapper)[30] 。但半自動的方法在日後漸漸不敷所求，所以後繼研究者加入 DOM-Tree[1] 的方法，希望藉著樹的語法特徵，在不同的網頁間找出相同區塊以建立擷取程式。此類型訴諸多頁間的差異來建立擷取規則，此稱為全自動(fully automatic) 的方法。隨此方法的風行，後繼研究者將 Data Mining[21] 的方法也帶入了網頁區塊分割技術中。

但 2003 年發布的 VIPS[11]分割方法，是利用單一頁面資訊且與主題內容無關聯，僅參考分割 Tag 的空間位置來進行視覺化區塊分割，可以保留分割區塊的分頁層次結構與調整區塊大小的特性，在後繼研究中開始受研究者的重視。如圖 2.2 中 (A)代表是原始網頁的概況，(B)是經由 VIPS 找出的視覺化分割線，在找出視覺化分割線的前後順序間，可得到圖 2.2 中(C)的頁面的分頁層次結構，也就是層次化的頁面區塊， 透過視覺化區塊分割，我們可以得到與主題呈現相類似的分頁層次結構，此是 VIPS 的特點。Web 頁面的層次化語義分塊，一個極大的用途就是提供移動式使用者瀏覽網際網路[9]；目前大部分的 Web 頁面都是針對桌上型而設計

的，由於移動設備通常螢幕較小，計算能力有限， 因此這些頁面並不適合移動設備直接瀏覽。 目前通常通過兩種手段來解決這種問題： 一是通過服務器進行頁面轉換，或者使用網頁縮略圖。 前者首先將使用者瀏覽的頁面進行分頁和轉換， 然後將分頁的結果提交給移動設備； 後者則是將整個 Web 頁面生成縮略頁面，整個頁面被分割為數目不等的區域， 使用者如果對特定區域感興趣，則可以再次訪問該區域的內容。

圖2.1 Example of FOM
資料來源：Function-based Object Model [14]



(A) (B) (C)

圖2.2 Web Page Hierarchical Layout
資料來源：VIPS[11]

2.3 視覺化網頁區塊分割

在現今受歡迎的網站，很明顯的網頁上資料通常不僅僅一個主題。經由觀察；特定用途的區塊通常位於某些固定的位置，而不同的主題區塊間也都會有一些視覺上明顯的分隔。 VIPS 即是利用此視覺上的線索做為分割依據的網頁區塊分割演算法，可以較為正確切割出符合網頁不同語意的區塊。

圖2.3為VIPS如何利用視覺化區塊分割線找出層次化的分頁區塊之示意圖，假設頁面的視覺上的分隔，如圖中所示，第一個回合會切出最上方的區塊，而下方三個區塊所組成的大區塊則因為內容相關程度(Degree of Coherence, DoC)。DoC值未達設定值，故進入下個區塊分割。在第二個回合，圖2.3(B) 中則會把最左邊的區塊切出來，相同的，右下方的二個區塊。因為DoC值未達設定的門檻值，進入第三個回合，圖2.3(C)，最後這兩個區塊也被切割出來。

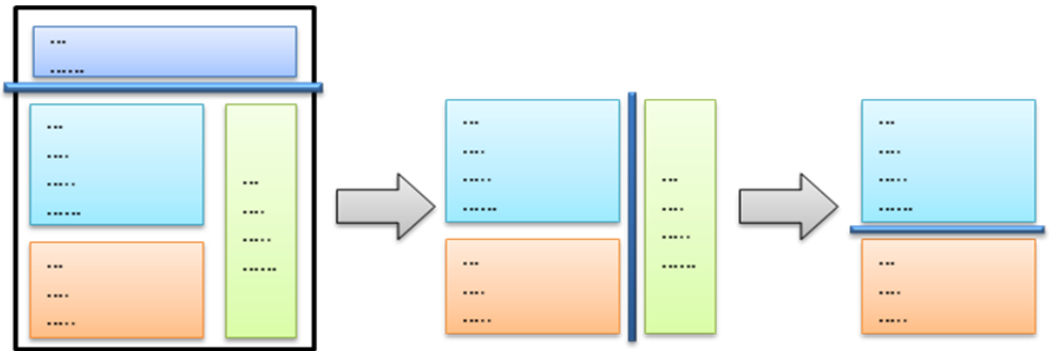
那要如何找出視覺化區塊分割線 ？VIPS主要可分為三個步驟：

VIPS的步驟一是將頁面分割為適當的區塊；依巡行 HTML DOM-Tree 找出視覺上的可分割區塊，採用人工經驗(Heuristic)的擷取規則，由頂點開始依著DOM-Tree往下做深先（ Depth first）的巡覽，每遇到一個節點（HTML Tag ）即依序比對這些規則來決定該節點是否為可分割塊，以及是否繼續往下一個節點比對，如此直至整棵樹走完，即可得出該回合所分割出的區塊。

VIPS的步驟二則是找出這些區塊在視覺上造成的水平或垂直分隔線(Separator)；依序將步驟一所得到的區塊如圖2.4(A)，逐一放入該頁面大小的空間之中，依區塊與頁面空間的占據情況執行分割、更新及刪除三種動作。如圖2.4(B)所示，先將最上方的區塊1放入，執行分割得到S1及S2；再將區塊2放入，一樣執行分割得到S1、S2及S3；再將區塊3放入，又再執行一次分割得到S1、S2、S3及S4；最後將區塊4放入，執行刪除S3，並更新S2的邊界，得到S1、S2及S3；最後再將最上方及最下方的分隔線S1、S4刪除，剩下來的分隔線S2即是本回合所得到的視覺上分隔線。

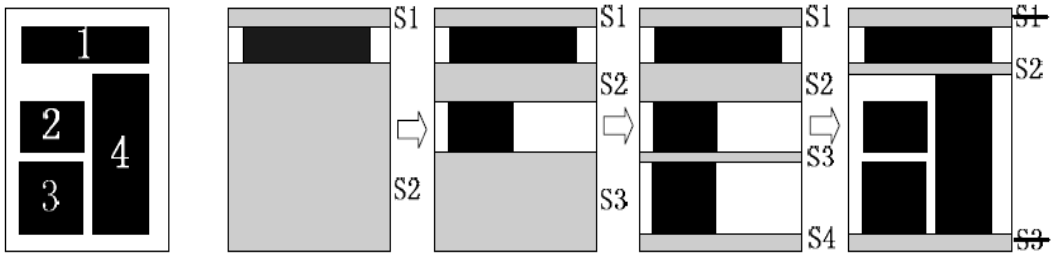
VIPS的步驟三是先依步驟二所得出的分隔線建立頁面區塊結構，再設定分隔線的權重。分隔線的權重是依視覺上的差異做為衡量的標準，例如水平分隔線高比較大的，權重比高比較小的來得大；又如分隔線上面字體小，下面字體大的權重也比上下字體一樣大的來得大。

最後只保留權重最大的分隔線為此回合切割區塊的依據，再依其DoC是否符合設定的門檻值，判斷所分割的區塊是否要再進行下一個回合的分割步驟。



(A)　　(B)　　(C)

圖2.3　VIPS演算法示意圖



(A) (B)

圖2.4 簡單的頁面分隔線偵測流程圖
資料來源： VIPS [11]

2.4 VIPS的區塊分割提早停止問題

基本上 VIPS 的分割區塊擷取 (Block Extraction) 是使用巡覽 DOM-Tree 的方法，只不過配合 HTML Tag 的空間位置來找出視覺化區塊分割線；在後續網頁設計採用新的設計方法下，使用上還是遭遇一些問題。 如圖 2.5 是 VIPS 作者之一所提供的 Demo Program，可看到在此位置下網頁完全無法分割 。

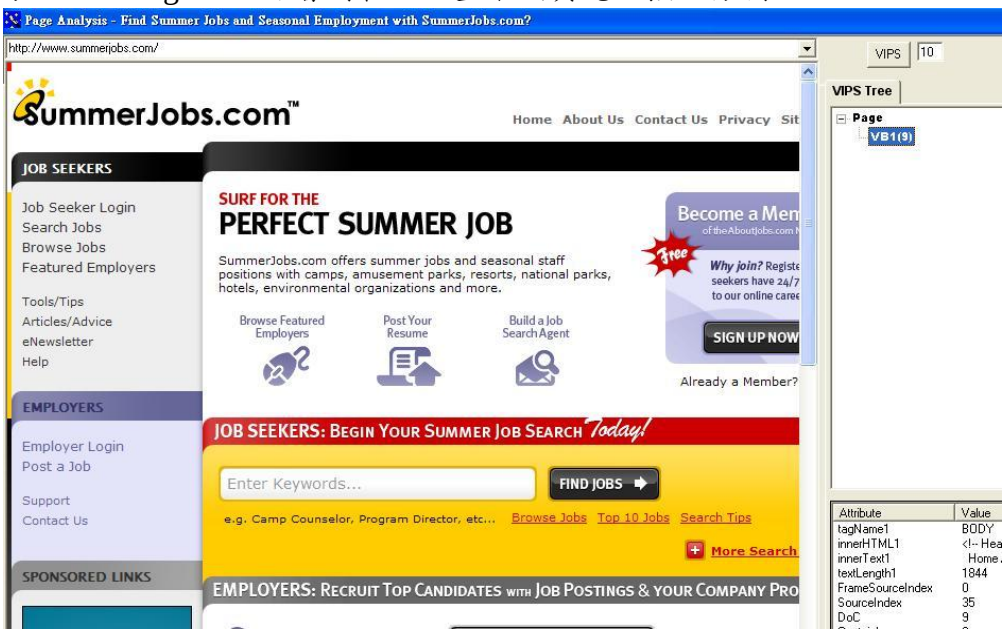
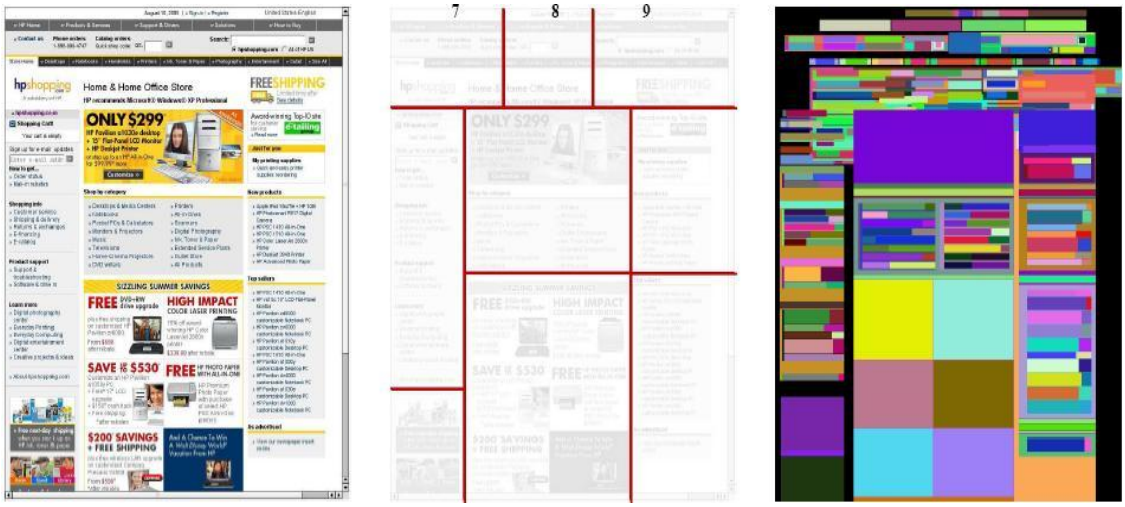


圖2.5 VIPS Demo Program
資料來源： www.summerjobs.com

而 Browsing on small screen [8]提出在一些簡單的頁面或視覺上很接近的頁面，VIPS 都會產生無法找出視覺化區塊分割線的狀況，所以他們提出以資訊密度的方法來進行區塊分割 。 圖 2.6 中(A)是一個範例網頁，將該網頁的 DOM-Tree 中每個結點放到 2 維空間座標上即形成圖(C)，借由演算法的縮減程序找出其密度較低區域當作分割線而得到圖(B)。但圖 (B) 中我們可看出在上端原本是一整塊的區域切割成三塊 (7,8,9) ，所以 Browsing on small screen[8]的方法雖然切開了整個區域， 但符合分頁層次結構的語義分塊卻沒有保留下來。

VSDR[22]將前面提到找不到視覺化區塊分割線的狀態，稱為提早停止問題(Pre-maturity problem)， 他們的解決方案即是將 VIPS與類似資訊密度的方法(Data region clustering)來搭配， 讓視覺化區塊分割可以進行分割 。 但同樣的因非採用VIPS的視覺化區塊擷取方法，那切割出來的區塊就不具其分頁層次結構的特性。雖然很多研究指出此現象如[8] [22]， 但並沒有提出具體的原因，所以底下我們提出幾個例子來分析此現象 。



(A) (B) (C)

圖2.6 Entropy Reduction

資料來源：Browsing on small screen[8]

2.5 區塊分割提早停止現象的分析

圖2.7 Empty Separator

資料來源：http://tw.yahoo.com

區塊分割提早停止，主要是當前區塊中的子結點發生重疊或面積異常導致找不到視覺化分割線。 圖2.7是一個找不到視覺化分割線範例，原先從紅線所框出來的視覺區塊 (Visual Cues)， 很明顯的可看出，應有兩條垂直的視覺化分割線，但中間橫軸的零碎文字所形成的視覺區塊，卻造成右邊的垂直視覺化分割線不存在，此視覺化分割線兩邊的視覺區塊就被當成一個區塊。這種狀況只要垂直與水平各發生一次，即造成區塊分割提早停止現象。下面幾個狀況都是可能找不到視覺化區塊分割線的原因。

2.5.1 代表視覺區塊之結點的選擇位置

圖 2.8 是一個範例，紅色線標示出其分割區塊的範圍，分割區塊對應的 DOM-Tree 結點在圖 2.10 Partial DOM-Tree 中的 <Table> Tag。圖 2.9 是同一範例，但紅色線標示出其分割塊的範圍是遠小於圖 2.8， 原因在於分割塊對應的 DOM-Tree 結點在圖 2.10 Partial DOM-Tree 中的 <B>Tag 。

顯然的當 HTML 本文包含了一些框架的設計時，如果區塊擷取沒有濾除此多餘框架，那整個視覺化分割區塊的大小會產生一些偏差。當此網頁又使用 DHTML 設計時，那很可能會造成在區塊間找不到視覺上分割線 。但此現象可以單純經由修改區塊擷取方法來避免 。

圖2.8 Zone of <Table> Tag

資料來源：https://www.google.com/accounts/ServiceLogin?

圖2.9 Zone of <B> Tag

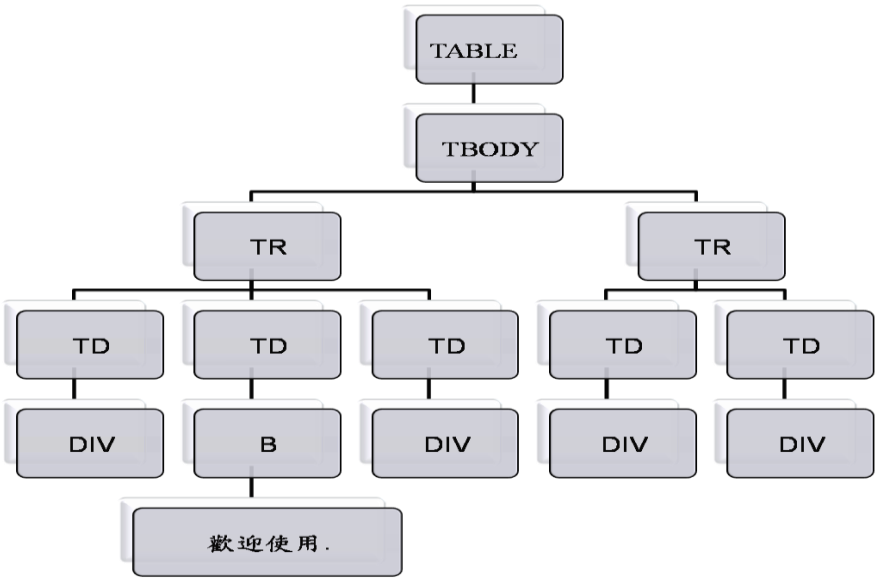


圖2.10 Partial DOM-Tree of figure 2.8,2.9

2.5.2 隱藏的結點

圖 2.11 範例中，紅色線標示出左上方一堆分割塊的範圍，而這些分割塊對應的 DOM-Tree 結點，在圖 2.12 Partial DOM-Tree 中的路徑 <HTML>-<BODY>-<DIV> 之下所有結點。也就是說此<DIV> Tag 之下的 Tag 被賦予隱藏的屬性，在頁面上不顯示出來，但其作標與大小是有效的，而在頁面上，還有其它顯示資料疊覆其上。如果區塊擷取沒有濾除隱藏屬性的 Tag ，那當然會發生沒有分割線的狀態。同樣的可以經由修改區塊擷取方法來避免此現象。

圖2.11 隱藏的結點

資料來源：http://www.janis.idv.tw/

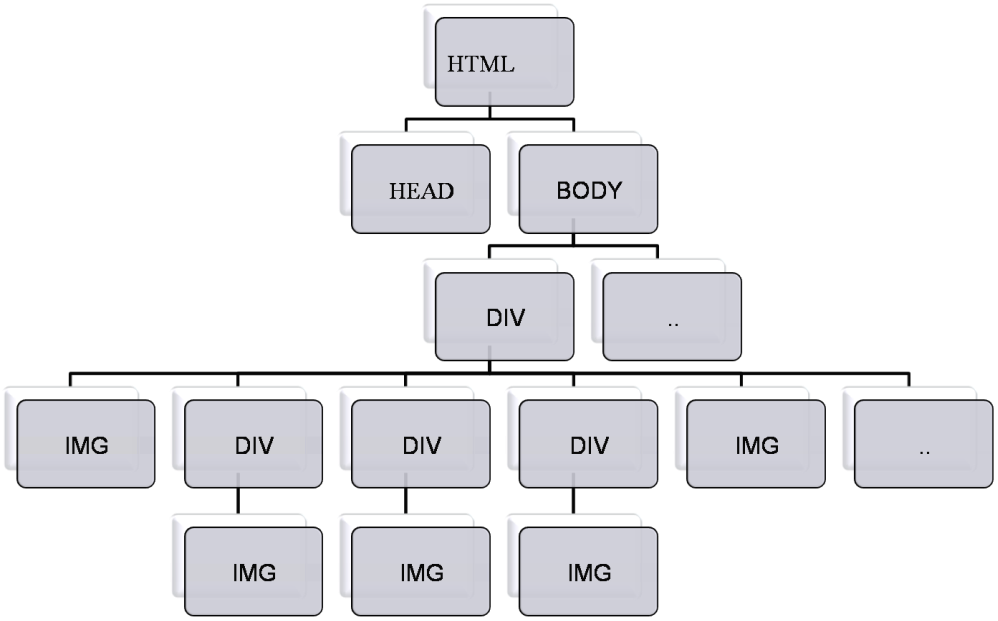


圖2.12 Partial DOM Tree of figure 2.10

2.5.3 重疊結點的位置

圖2.13 是 Javascript[5] 控制網頁內容的例子；圖中原本是一個自動捲軸的即時新聞，紅色線標示出相同架構的子結點資料所佔的區塊位置，圖2.14 是它的 Partial DOM-Tree 概略架構。也就是說頁面借由定時觸發 Javascript 來更改Tag 的屬性使其顯示出來，但進行區塊擷取得出來的子結點資料所佔的區塊位置，卻是緊鄰在一塊，這使的原區塊大小變大了，所以對後續的視覺化區塊分割就產生影響。此現象也可以經由修改區塊擷取方法來避免 。

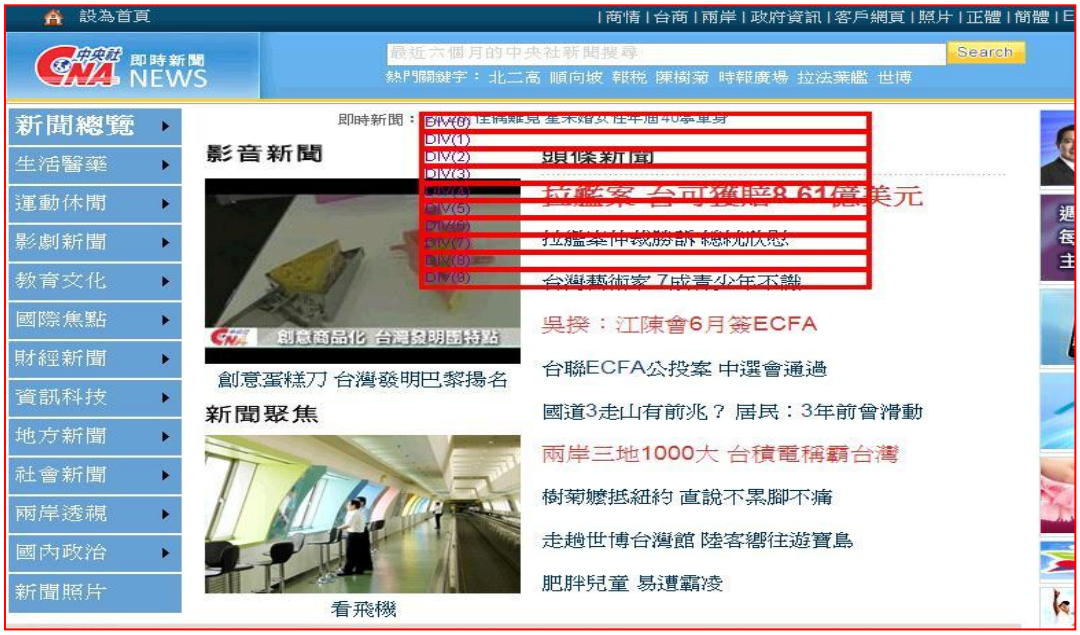


圖 2.13 重疊的結點

資料來源：http://www.cna.com.tw

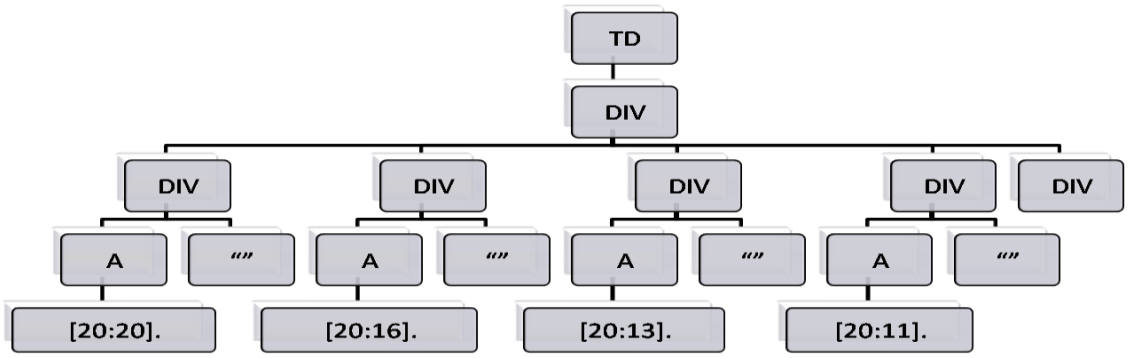


圖 2.14 Partial DOM-Tree of figure 2.12
資料來源：http://www.cna.com.tw

2.5.4 瀏覽器本身的佈局問題

圖 2.15 是網頁內容顯示不正確的例子；紅色線標示出其分割塊位置跑到右方去了，並且連本文都沒顯示出來，這種現象在新版的 Browser 已經沒有呈現此現象(如 IE 7.0)。

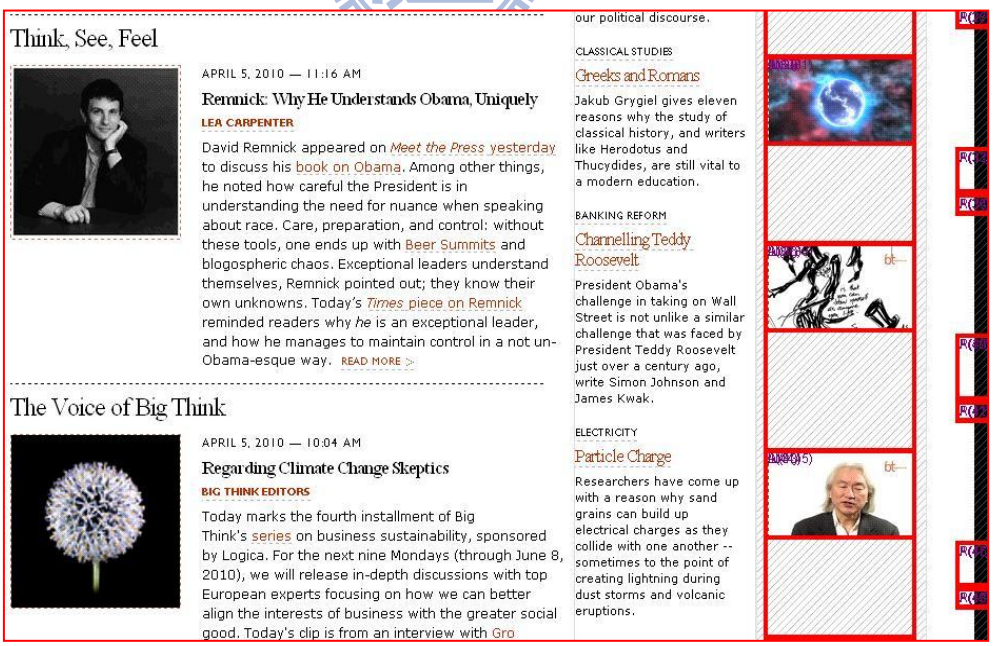


圖 2.15 Browser Render Problem
資料來源： http://www.bigthink.com

2.5.5 代表資料的DOM-Tree出現在不只一處

正常下每個子DOM-Tree只代表一組資料，但因為Browser本身的問題， 卻使的這些結點出現在不只一處。圖2.16是DOM-Tree出現重覆子樹的現象(出現兩個 <TABLE>)，紅色線標示出其正確的兩個分割塊範圍(<P> 與 <TABLE>)，對應於圖2.17 路徑 <TR>-<TD>-<P>-<FONE>-#Text，其中 <TABLE>所代表的子樹應不存在，但讀取 <P> 的結點資訊，它的大小卻是橘色線標示出的範圍，雖然 HTML本文沒有此<TABLE>子樹，但透過 DOM的API卻讀到Tag與重覆位置。因為<P>的大小包含另一個兄弟樹<TABLE>的大小，當然是找不到視覺化分割區塊。此現象只有調整<P>結點為正常大小，才可能產生視覺上分割線 。

圖 2.16 Repeater Elements
資料來源： http://www.bigthink.com

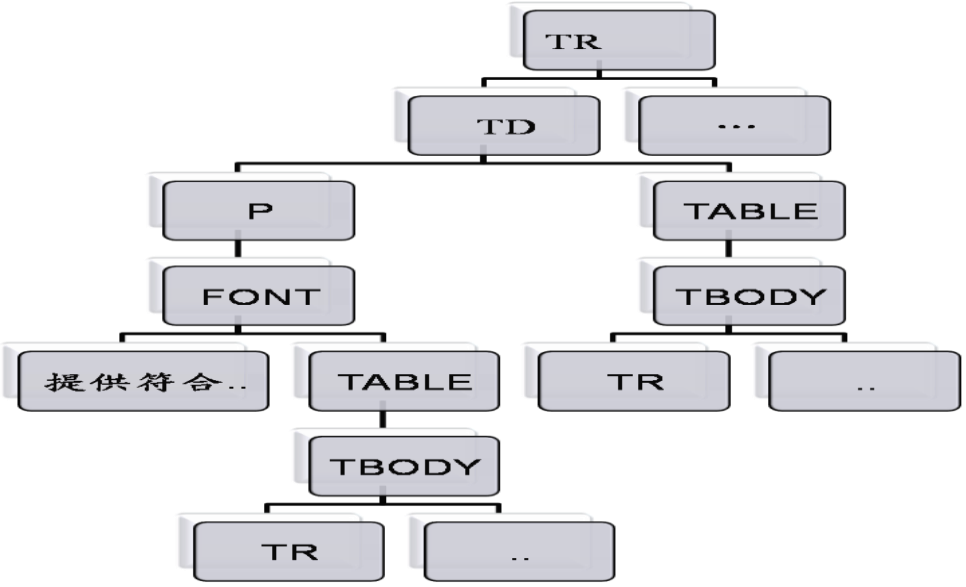


圖 2.17 **Partial DOM-Tree of figure 2.15**

2.5.6 由小變大的區塊

圖 2.18 為使用 DIV 的特性來設計網頁的的影響；紅色線標示出在 DOM-Tree 中，中間區塊的所有結點的大小與範圍，我們可看到中間區塊的結點，有部份跑到兩邊的區塊，會發生此情況，很明顯的是頁面設計採多層 DIV 方法，每個子代的 DIV 都顯示一些資料，因後繼的 DIV 區塊已經大於原先設計的父代 DIV 的大小，所以跑到兩邊的區塊，但因為重疊的原因，Browser 按 HTML 文件布局的原則來呈現頁面，所以看不到這些凸出區塊。那兩邊多凸出去的結點，很可能會造成在區塊間找不到視覺上分割線 。 只有找出重疊區塊的顯示順序，將重疊結點大小加以調整，才能製造出視覺上分割線 。

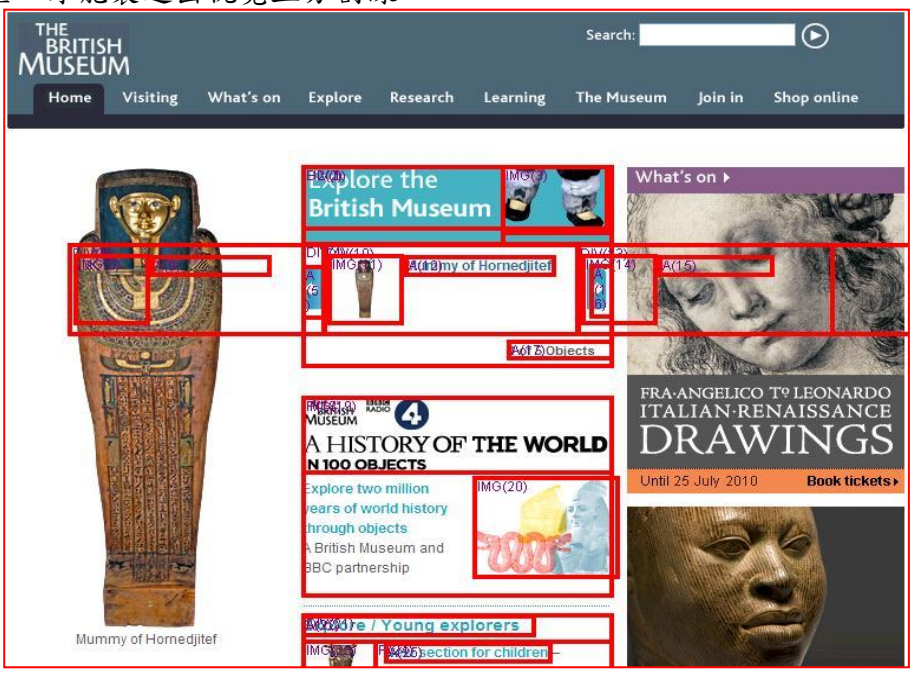


圖 2.18　由小變大的區塊

資料來源：http://www.britishmuseum.org/default.aspx

2.5.7 使用重疊設計的區塊

圖 2.19 為 CSS[4]的影響，對照其圖 2.20 Patial DOM-Tree 圖，紅色線標示出最大的 Tag <DIV>包含了其他 Tag。我們知道因 CSS 設計可以讓區塊放在 DOM-Tree 的任何位置，而且讓它呈現在空間同一位置，所以此頁面的設計，使一開始巡行 DOM-Tree 便發生區塊重疊現象。同樣的只有找出重疊區塊的顯示順序，將重疊結點大小加以調整，才能製造出視覺上分割線 。

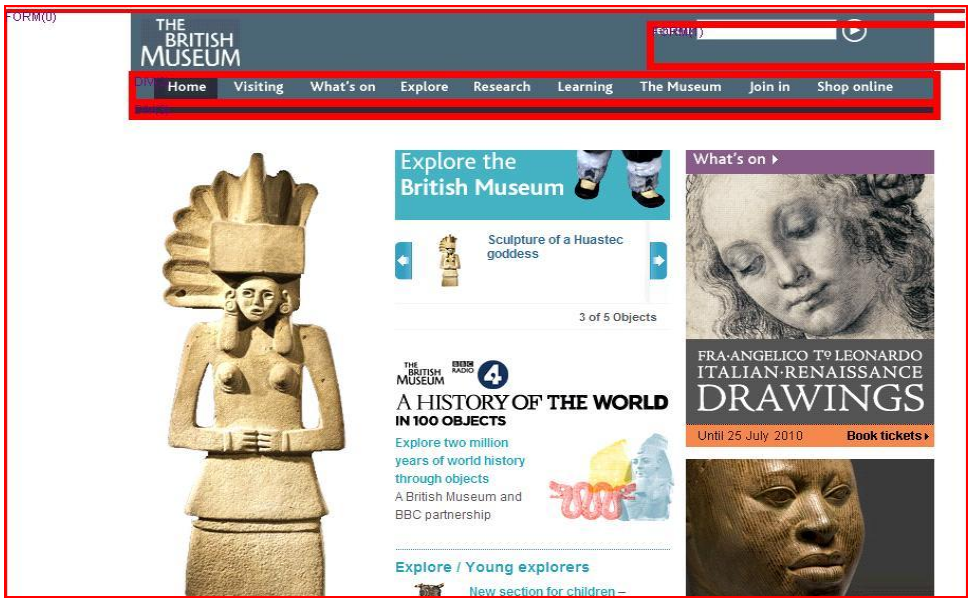


圖 2.19　使用重疊設計的區塊

資料來源：http://www.britishmuseum.org/default.aspx

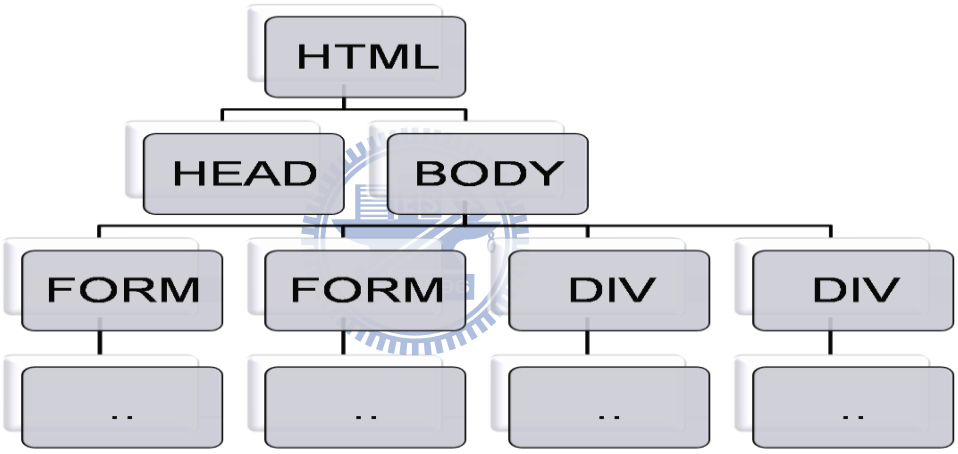


圖 2.20 Partial DOM-Tree of figure 2.18

2.6 DHTML 頁面設計的效應總結

由上面舉例可知，經由修改區塊擷取方法可以避免因 DHTML 的額外特性造成的現象(如 2.5.1、2.5.2、2.5.3)，但 2.5.5、2.5.6、2.5.7 的頁面設計方法，卻可能發生區塊重疊而找不到在視覺上的分割線；而現代網頁設計傾向將網頁架構(Web Page Layout)與內容設計(Content)分開，更加重此現象的發生。雖然 VIPS 在後來發布的程式中，使用了某種校正方法後，在上述的範例中有部分可分割到最小區塊，但分頁層次結構的語義區塊是不正確。正常的狀況是分割區塊的總面積是小於頁面面積 ，反之，當原頁面面積小於分割區塊的總面積時，可確定分頁層次結構的語義區塊特性沒有保留下來。本文提出的方法即是按照 HTML 布局的特性 (HTML Rendering-based) 來修改區塊重疊區域的大小， 進而讓視覺化分割區塊可以分割到最小結點區塊。

三、理論

3.1 改良的區塊擷取方法

會發生視覺化區塊無法分割到最小區塊，基本上即代表視覺化區塊上的 Node 位置之間，發生面積上的重疊而找不到在視覺上的分割線。為避免 Node 代表的所在位置空間，發生重疊而找不到在視覺上的分割線的現象，最有效的方法就是修改原 Node 的大小，讓視覺上的分割線可以呈現出來。本文提出的方法即是改善區塊擷取的方法並修改區塊重疊區域的大小，讓視覺化區塊分割可以進行到單一結點區塊，並保留分頁層次結構的語義分塊特性。

3.1.1 DOM-Tree的巡行問題

DOM-Tree 最早引入是為了在瀏覽器中進行布局顯示，而不是進行 Web 頁面的區塊空間結構描述。比如，即使 DOM-Tree 中兩個結點具有同一個父結點，那麼這兩個結點在空間上也不一定就是有關聯的。反之，兩個在空間上有關係的結點卻可能分布在 DOM 樹的不同之處。因此僅僅通過分析 DOM-Tree 並不能正確獲取 Web 頁面中各區塊分頁層次結構。針對兩個在空間上有關係的 Node 卻可能分布在 DOM-Tree 的不同處之特性，顯而易見不能使用巡行 DOM-Tree 來做區塊擷取 (Block Extraction)。所以我們提出基於網頁文件布局(HTML Rendering-based) 的區塊擷取方法，簡稱為 HRPS (HTML Rendering-Based Page Segmentation)。

整個方法的重點即將位置空間鄰近的 Node 預先挑選出來，而後修改有發生重疊的區塊的大小，讓視覺上的分割線可以呈現出來。以下各節分別就原理上來做介紹。

3.1.2 視覺區塊 的選擇方法

首先我們知道在 DOM-Tree 上每各 Node (即 HTML Tag的 Element) 皆有位置與大小的資訊，而 Text Node (即非 HTML Tag的Element)則沒有此資訊。所以每個 Node 都可代表一個區塊，而區塊間就位置空間的關係可分為兩類，一為 Inline Elements ，一為 Block Elements。圖3.1 表示Inline Elements 的特性，在DOM-Tree 上的每個 Node 將其顯示在位置空間，形成的是連續區域，也就是說每個 Node 所代表的區塊都是緊鄰在一塊。當子區塊排列至邊界時，此區塊就放到下邊的位置，但它的上邊還是緊鄰著上邊的區塊。圖3.2 表示在DOM-Tree上的每個Node皆是 Block Elements，也就是說由此類 Node 開始的 Sub-Tree 皆成一獨立區塊。其 Sub-Tree 的 Node 所代表的區塊，也成了 Root Node 的子區塊，此為Block Elements 的特性，如 <P>、<DIV>、<BLOCKQUOTO>、<TABLE>、
、<OL>、<FORM>、<H1>to<H6> 都是Block Elements。 圖3.3 顯示當 Inline Elements 緊接著一個 Block Elements時，此後繼的 Node 與其 Sub-Tree 成了一個子區塊。所以在區塊擷取上，我們只要挑選 Block Elements ，而其後繼的 SUB-Tree 含有要顯示的資料，如 Image 、Text、 UI-Form 等，就足以代表此視覺區塊(Visual Cues)，不需原 VIPS 的

人工規則 。而為修正巡行 DOM-Tree 的區塊擷取上的問題，下節介紹 Offset-DOM 的建立方法。

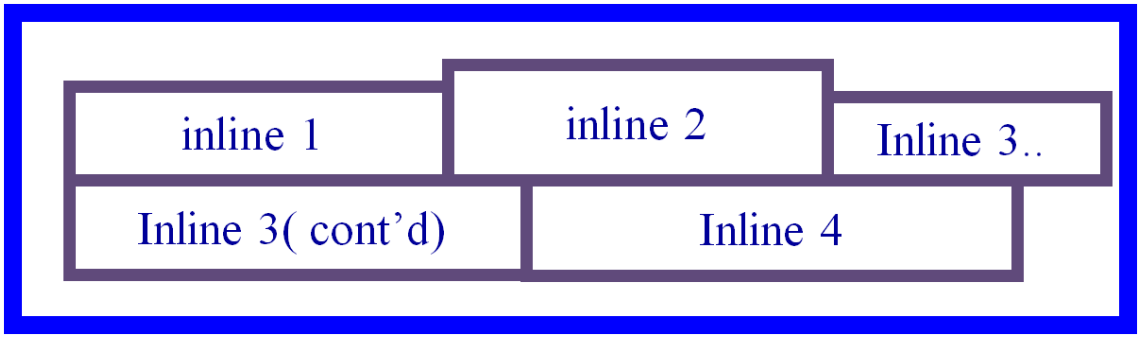


圖3.1 Inline Elements

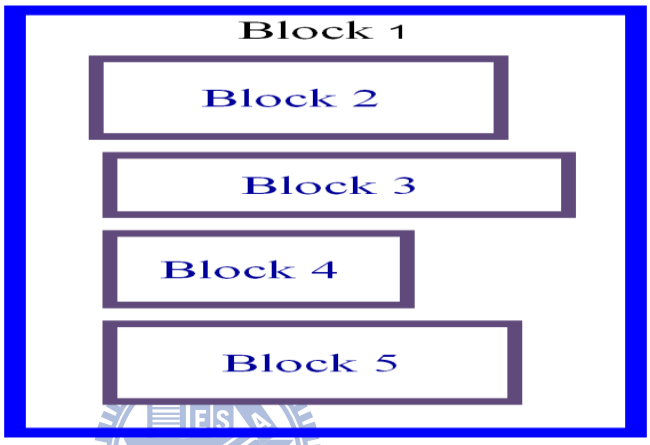


圖3.2 Block Elements

圖3.3 Block Elements as Carriage return

3.1.3 Offset DOM

HTML文件的DOM-Tree，是提供給使用者的 API 。在 HTML API中有一個附屬的API， 稱為 OffsetParent，它是個指向另一個父代結點的API， 功用如同圖3.4中所示parentNode，但它不是 DOM-Tree 的父代結點，而是 HTML Rendering 功用的父代結點；藉由此 API 可以得到每個 Node的空間位置與大小，但要取得 Node 的位置與大小，是藉由此連結來重覆計算得來。如圖3.5範例中我們可看到，<TD> 需經由 API 得到 <TABLE>的Node，再得到<BODY>的Node，把所經過Node 的位置

(OffsetLeft 、 OffsetTop)疊加起來，才是此 Node 的真正位置。由觀察透過連結計算每個 Node的位置與大小的方式中，我們發現此方式，實際上就代表同群資料的連結。代表可能分布在DOM-Tree的不同處的 Node ，但空間位置上有關係的 Node 卻此連結樹中。因為 Browser 也是經由此資訊，來改變對應的 HTML Node。經由比對 IE 與 FireFox 兩個 Browser，其 OffsetParent 算法是大同小異，所以算是一個一般化的作法。把此連結建立起來，稱為 Offset DOM 以別於 HTML DOM(如果 OffsetParant 是 Null值，即放在樹根之下。) 。Offset DOM中每個Sub-Trees即代表在區塊中同一群跟視覺改變上有關的所有Elements的集合，所以此圖已把空間上同一群鄰近的 Node挑選出來了。

在圖3.8是圖3.6 的 Partial Offset-DOM ，表示由一個傳統的<TABLE>來表示的一個區塊。而<TABLE>的下一層<TBODY>、<TR>都沒有後繼的有效Node，故可忽略，而<TD>即代表<TABLE>之下又一個區塊，其第一個子區塊是由 <DIV> 與 <FONT> 來表示，因為跟隨了一個 <TABLE>所以又有一個子區塊，而下面兩個
 又切割出第三個子區塊，但因<FONT>、<A>都是 inLine Elements。所以經由 Offset-DOM 來挑出後繼的Node是否為 Block Elements ，即可找出代表區塊的 Node。而其它 Inline Elements 可經由彼此間的位置關係與 HTML DOM的判斷，來取出代表此 SubTree 最頂端的 Node。所以實務上只要在 Offset-DOM上的有效Node (有後繼的有效Node，如 Text Image 等)，就是文件上要呈現的Node ，找到代表此 DOM-Tree 最頂端的Node，即是想要取得的視覺化區塊的代表 Node。

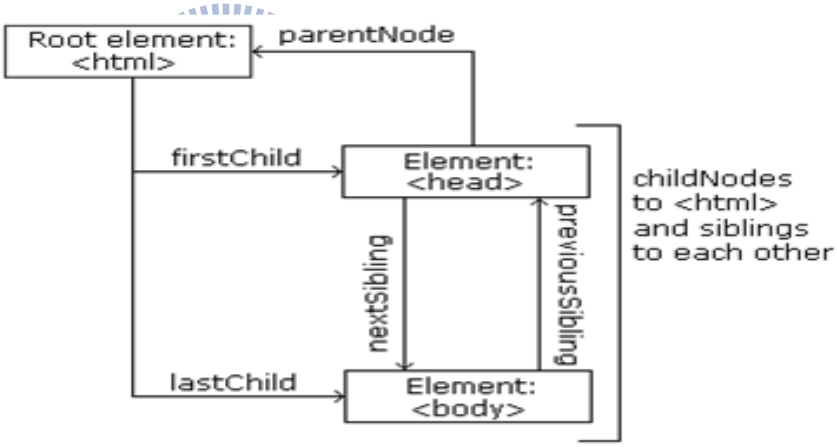


圖 3.4 HTML/XML Application Program Interface

資料來源：http://www.w3schools.com/htmldom/dom_nodetree.asp

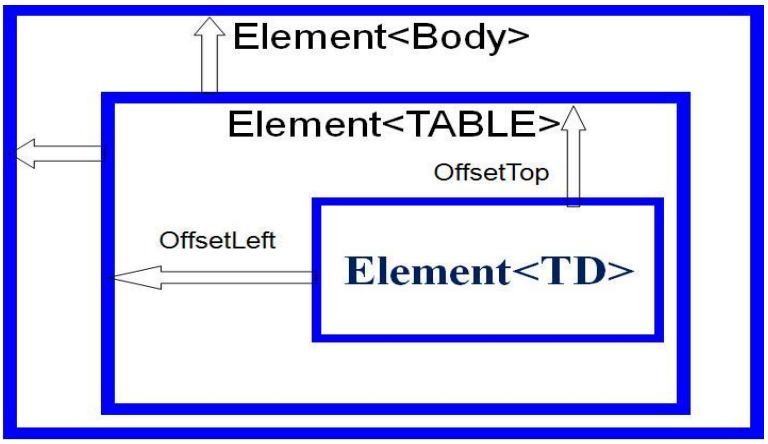


圖 3.5 Element Position

圖3.6 Partial Graph of figure 2.15

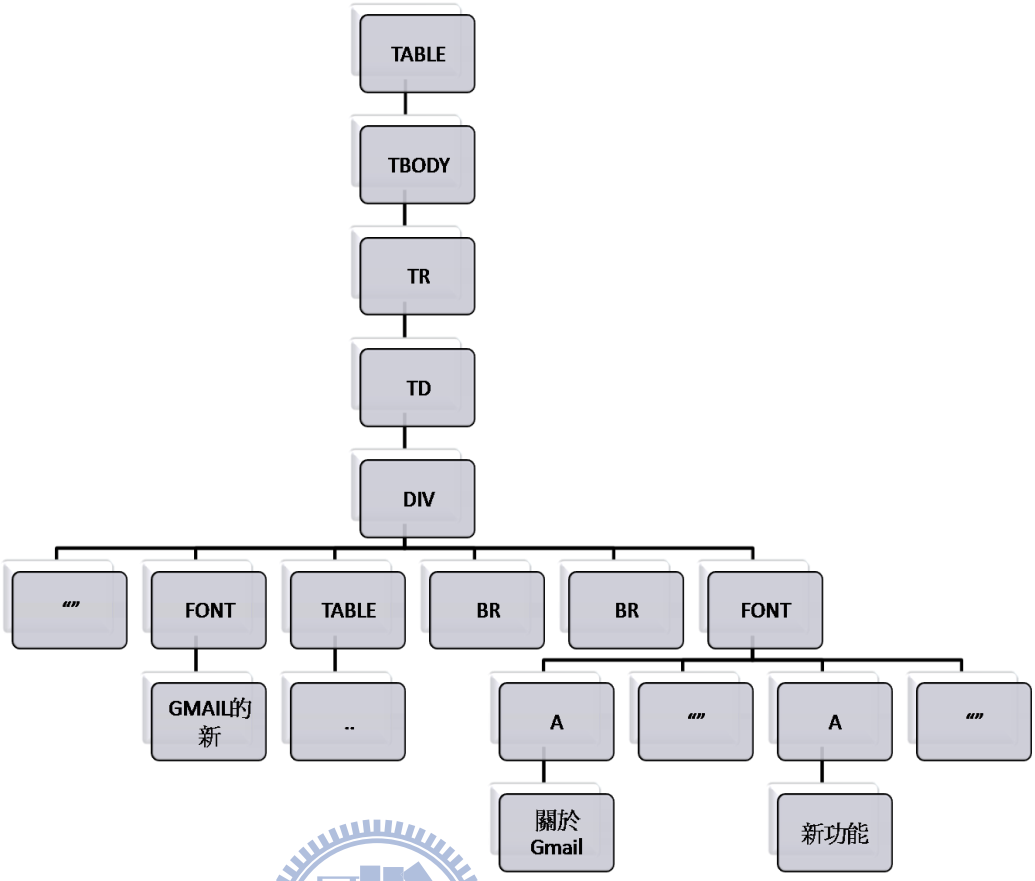


圖3.7 HTML DOM

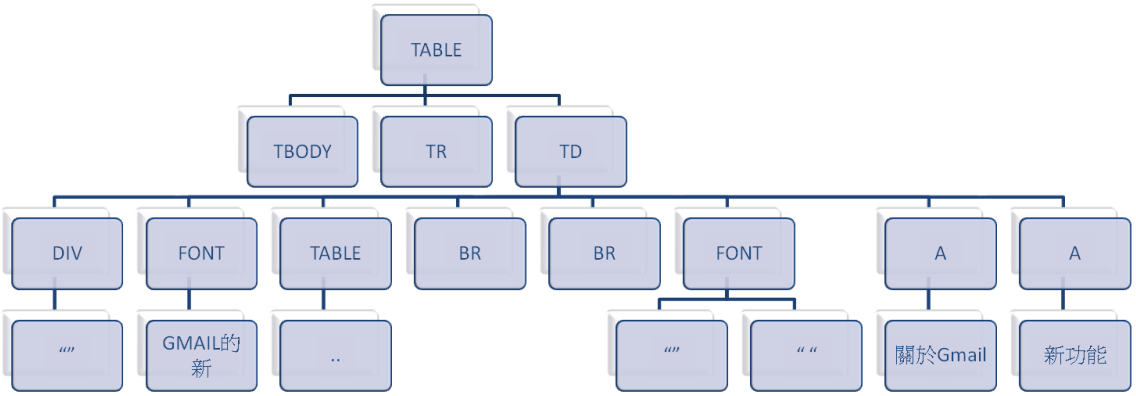


圖3.8 Offset DOM

3.2 遮蔽區塊調整策略

當取得視覺區塊的代表Node時，下個步驟就要了解，當Node間發生面積上的重疊而找不到在視覺上的分割線時，區塊之間的前後順序，進而來決定調整依據。

3.2.1 DHTML文件布局

透過[30]的了解，可知 DHTML 文件的呈現階層(Hierarchical layout)。圖3.9 中被疊覆的 Node，最後呈現的是那一個 Node，所以這給了一個調整 疊覆Node基準。因為被遮蔽的部分，絕對不是原設計者想要呈現的狀況。但純粹由 Node 之 Z-Index 屬性與文件布局，來參考疊覆方向是不夠的， 後面會介紹疊覆基準的輔助測量策略 。

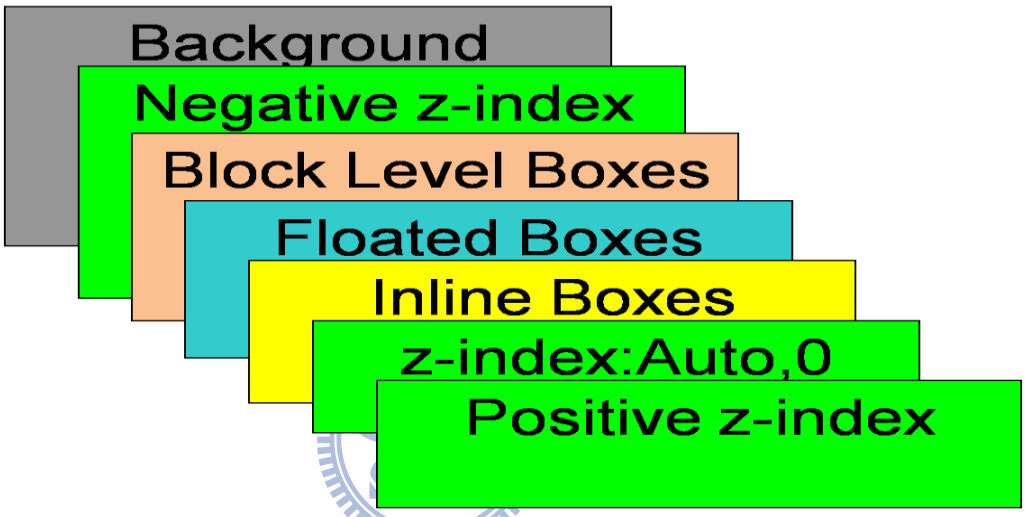
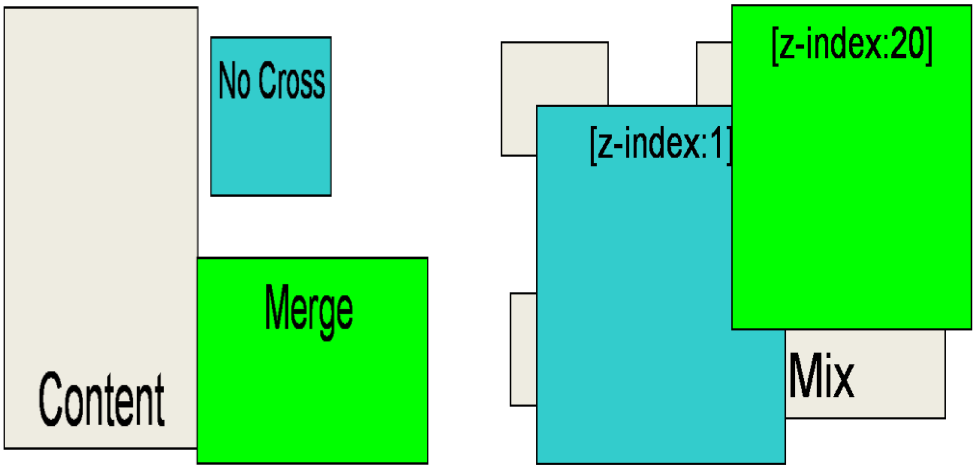


圖3.9 Hierarchical layout

3.2.2 遮蔽區塊的分裂調整

那些疊覆狀況需要調整 Node大小 ？ 由圖3.10中 (A) 可知(A)是表示不需調整的，甚至緊接的Node也是不需調整的，因為原設計者就是要彼此緊臨。而(B)中代表的是要進行調整Node大小，但某些狀況，如圖3.11所示，它可能是代表子區塊而非疊覆的區塊，就不需調整， 但要把此Node 放到下個子區塊去比對疊覆狀況。最後要調整的區塊是如圖3.12中看到的疊覆狀況，如果區塊上層有另一個區塊遮蔽 ，即將遮蔽區塊分割，所以產生兩個區塊 。而這兩個區塊，都是可視區塊，因為其它部份被遮蔽了。經由底層往上的調整遮蔽區塊分割，可能得到多個代表區塊，最後留下面積最大的區塊，由此區塊的位置大小為依據，反過來調整此對應 Node 的大小，即產生了視覺上的分割線。

由底到頂的調整遮蔽區塊分割，讓最後能產生視覺上的分割線，但只藉由圖3.9的關係來看區塊前後順序，可能會有錯誤，因為區塊間有可能原始設計者是把它們放在同一層，那排序後可能遮蔽訊息量較大區塊，所以需要一個輔助的測量訊息量的方法。



(A) (B)

圖3.10 (A).No Effect (B).Overlay

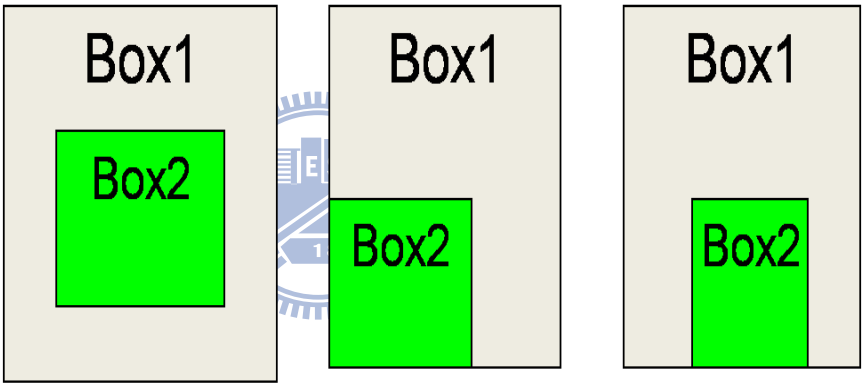


圖3.11 Included Block

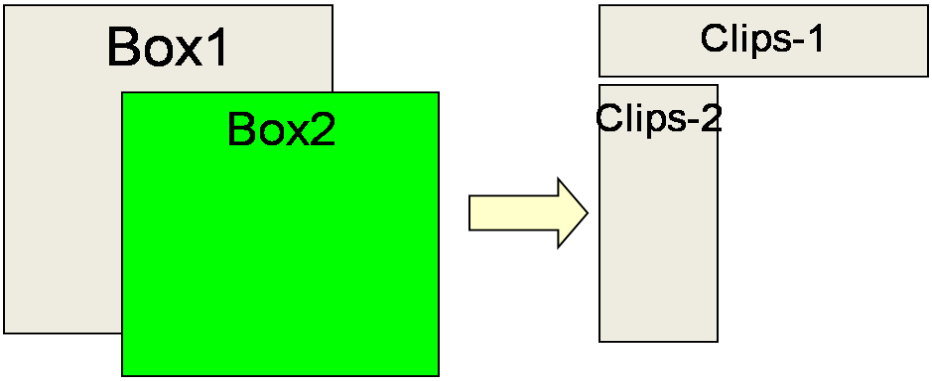


圖3.12 Intersection Block

3.2.3 資訊測量 (Information Measuring)

針對區塊間的疊覆排序，本文提出一個資訊測量方法來當作疊覆排序的輔助因子。因為是對頁面的分頁間作彼此比較，所以分頁間必須以原頁面為比較基準，也就是說直接以分頁資訊與原頁面資訊的接近度作為輔助因子，可能是比較好的選擇。

所以資訊測量度（Degree of Information, DoI ）如下式:

$$D(i) = f(WebPage, Blocki)$$

把每個分頁與原頁面的關係拿來當比較的因子；在數學上可採用歐幾理德距離計算，即滿足分頁資訊與原頁面資訊接近度的量測。針對遮蔽區塊大小的判別，很明顯的，區塊大小是比較因子的首選，而考慮本文字數多的，應該是要設計者想要表示區塊。所以取這兩個因子，作歐幾理德距離計算做為測量結果 。

所以最後方程式為

$$Di = \sqrt{(1-ri)^2 + (1-pi)^2}$$

式子中 r = 分頁中的總字數與頁面總字數的比率，

p = 分頁的大小與頁面大小的比率 。

如果 DoI越大代表此分頁與原頁面的接近度也越小。

DoI越小代表此分頁與原頁面也越接近。

3.2.4 遮蔽區塊調整

此節以一個真實範例，其原始網頁是圖3.14 所示。

圖3.13 來解釋是如何調整遮蔽區塊，可看到經由 Z-Index 與 DoI 排序後，有六個疊覆區塊。首先由編號0的區塊由底層往上逐層比對 ，如果出現上層遮蔽區塊，即將被遮蔽區塊分割 。 每個分割區塊都要保留下來，再與更上層遮蔽區塊比對，重覆此步驟，最後得到的最大的分割區塊，即是此區塊的答案。 此範例是 View 之下的編號0的區塊。重覆其他五個區塊，執行遮蔽區塊調整，得到最大的分割區塊，即完成此分頁區塊調整。

看圖3.14 中(A)圖代表最上層有一被遮蔽區塊，如紅線所示其大小與位置，在其上有一個近似大小的遮蔽區塊，如橘線所示，經由調整策略，紅線所示區塊經分頁區塊調整後， 縮減成一個可視區塊， (B)圖即代表最後的分頁區塊結果。

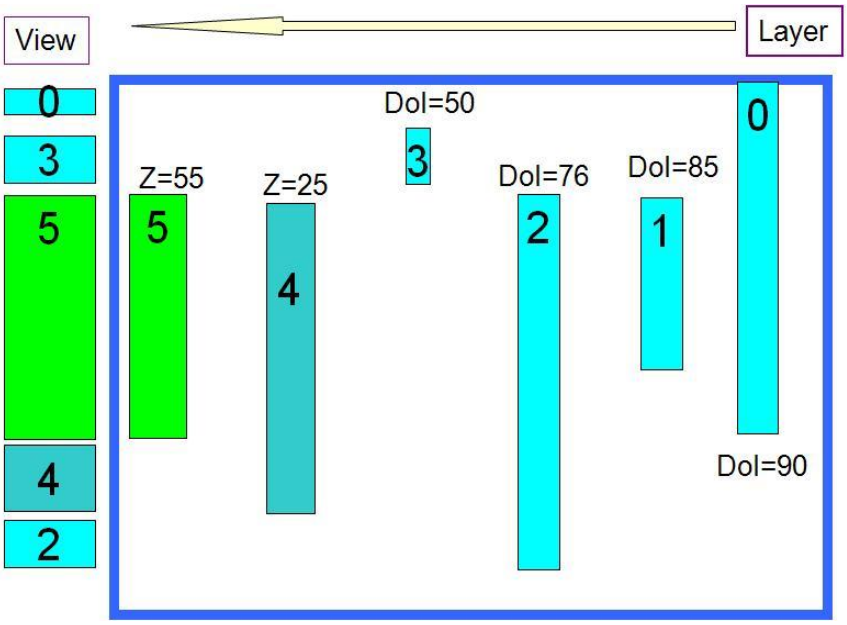


圖3.13 Measuring Strategy

(A) (B)

圖3.14 Example of Segmentation

資料來源：http://www.24ways.org/

3.3 HRPS 程序流程圖

最後把上述方法，以程序流程圖表示。圖3.15可看到 HRPS 流程基本上與 VIPS 相仿，只不過區塊擷取是由 Offset DOM 開始。且流程中多了個 Included Pools，以儲存區塊被完全包含的Node，以放入下一回合的視覺上的分割線的判斷程序(Separator Detection)。在Separator Detection前端的遮蔽區塊調整策略(Measuring Strategy)，是整個演算法的核心程序。借由此區塊調整程序，可保證每個分頁區塊存在視覺上的分割線。

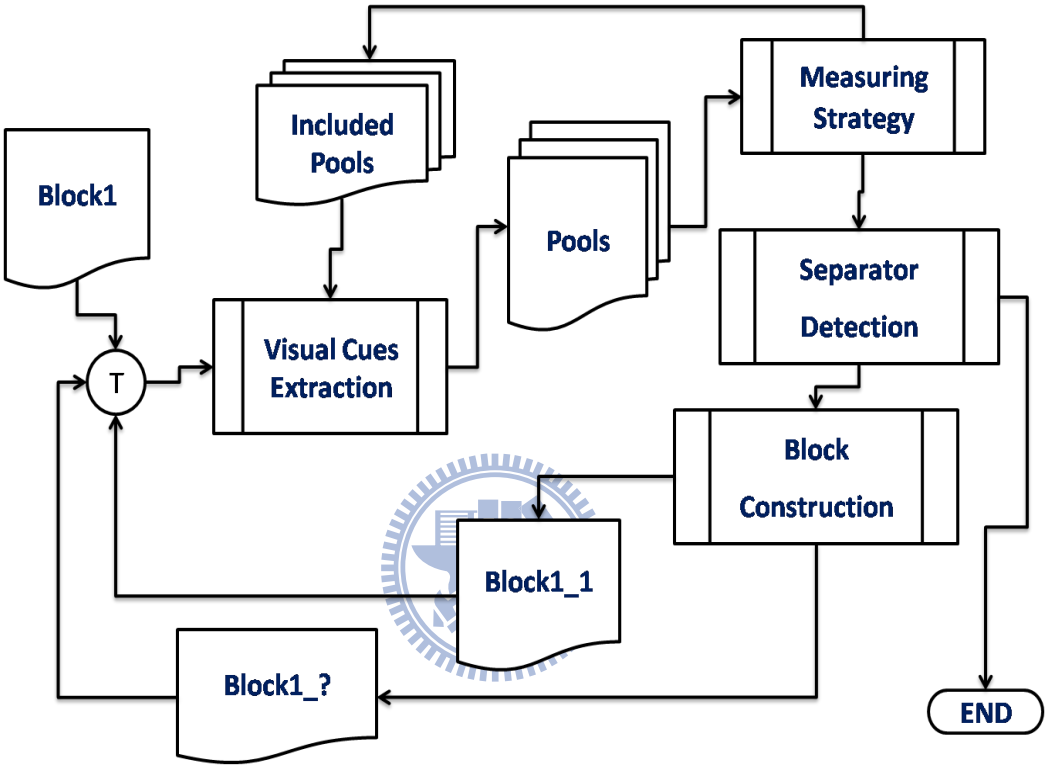


圖3.15 HRPS Process Flow

3.4 分割區塊的大小調整

在原來的 VIPS 上，有調整分割區塊大小的方法。只不過它要指定實際的最小區塊大小與 DoC 值，而 DoC 值又可能跟網頁頁面設計有關，不是一個恆定的參考值。本文的方法僅用一個 Doi 來調整分割區塊大小 ，且是一個接近恆定的參考值。如圖3.16 經由設定 DoI 的門檻，就能動態調整分頁區塊大小。

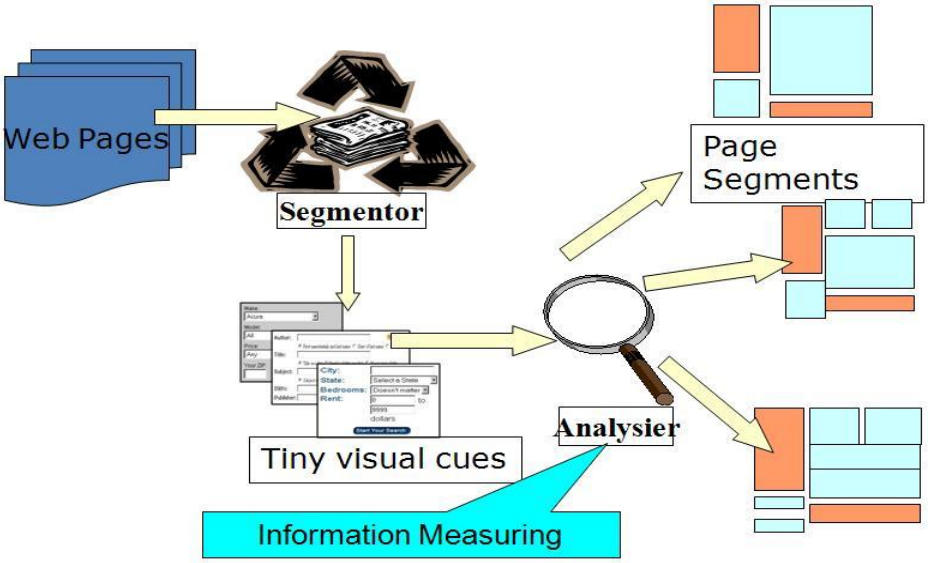


圖3.16 Data Regions Adjustment