國立交通大學

資訊管理研究所

碩　士　論　文

一個以賽局理論為基礎的網頁主題區塊擷取演算法

A Game-Theory-Based Algorithm

for Extracting Theme-Block from A Web Page

研 究 生：陳 昌 民

指導教授：羅 濟 群 博士

中華民國　九十八　年　六　月

一個以賽局理論為基礎的網頁主題區塊擷取演算法
A Game-Theory-Based Algorithm for Extracting Theme-Block from A Web Page

研 究 生：陳 昌 民　　Student： Chang-Min Chen

指導教授：羅 濟 群　　Advisor： Chi-Chun Lo

國立交通大學

資訊管理研究所

碩士論文

A Thesis

Submitted to Institute of Information Management

College of Management

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Information Management

June 2009

Hsinchu, Taiwan, the Republic of China

中華民國　九十八 年 六 月

一個以賽局理論為基礎的網頁主題區塊擷取演算法

研究生：陳昌民　　指導教授：羅濟群 博士

國立交通大學

資訊管理研究所

摘要

隨著資訊與網路科技的快速蓬勃發展，網際網路已成為目前最龐大的資料體，由於科技的進步以及使用者人數爆增，每天有數以萬計的網頁產生。而網際網路也成為使用者最大的資訊來源。資訊爆炸的現代，要在這麼龐大的資料當中找尋特定主題的相關資料，變成是一件相當重要的研究課題。

因此，本論文提出一個以賽局理論為基礎的網頁主題區塊擷取演算法(a **G**ame-theo**R**y-based **A**lgorithm for extracting theme-**B**lock from a web page, **GRAB**)，能夠自動地將使用者有興趣的主題區塊自動地辨識出來，並轉換成易於儲存、檢索與分析的結構化資料，提供不同平台(e.g.手機、PDA)應用的方便性。

本論文針對提出的 GRAB 演算法設計一雛型系統，並設計了兩個實驗，以實際網頁資料測試驗證演算法的效能。實驗一的結果證明 GRAB 演算法在 10 種主題的 HTML 網頁具有 70%到 90%的效能，對於新聞網頁的效果特別好。實驗二從實驗一挑選效能最好的 3 種主題做為資料集，並與現有三種網頁區塊擷取法比較。實驗結果證明 GRAB 的整體效能優於三種現有方法，其中處理結構化網頁的效能雖與現有方法差不多，但對於新聞網頁的效能仍優於現有方法；在處理非結構化網頁則具有 87%到 95%的效能，都優於現有的三種網頁區塊擷取方法。

關鍵字：網頁擷取、資料擷取、主題區塊、賽局理論

A Game Theory-based algorithm for extracting Theme Block from web pages

Student：Chang-Min Chen Advisor：DR. Chi-Chun Lo

Nation Chiao Tung University

Institute of Information Management

Abstract

Because of the rapid development of Information and network technology, Internet has become the largest body of information. The progress of technology and the explosion in the number of users, so there are produced tens of thousands pages every day. Internet has become the largest source of information. In the information explosion times, searching specific topic data in such a huge data is becoming an important topic.

For this reason, this paper presents a **G**ame-theo**R**y-based **A**lgorithm for extracting theme-**B**lock from a web page(GRAB). It can automatically identify user interested topic blocks, and then converted to easy storage, retrieval and analysis of structured data for providing different platforms (eg. mobile phone, PDA) to facilitate the application.

This paper provides a prototype system based on GRAB algorithm. I design two experiments, and verify the effectiveness of algorithms through the actual web page data. The first experiment results prove that GRAB algorithm in ten topics of HTML web page can achieve 70% to 90% accuracy rate, especially in news web page. The second experiment selects the best three topics from the first experiment results for data sets, and compare with three existing web blocks retrieving methods. The results prove GRAB overall performance is better than three existing methods. Although processing structured web performance is similar to existing methods, but news web performance is better than existing methods. Processing unstructured web performance is can achieve 87% to 95% accuracy rate, and all is better than three existing web blocks retrieving methods

Keywords： web extraction, information retrieval, Theme-based block, game theory

誌謝

首先我要感謝指導教授羅老師在研究所兩年內的細心指導，羅老師做事方法與態度影響我很多，讓我要求自己做事要更講求效率、具有正確邏輯。老師開放而不放縱的個性，讓我能自由選擇有興趣的題目，開會時也提供鞭辟入理的建議，使本論文更加嚴謹。

感謝實驗室的同學、學長姐與學弟妹，特別是要感謝鼎元學長與同儕志華，給我學術研究上眾多的支持與建議。大蓉、湘婷每次出現總是能帶來歡笑，讓實驗室氣氛和樂許多。學弟們寫程式認真的態度，也點醒我不少。雖然只相處短短的時間，大家卻像是一家人。

感謝我的女朋友-韋君，在我心情低落的時候能給我勇氣，在我開心的時候能與我共享，在我煩悶的時候能逗我開心。全世界就屬妳最瞭解我，妳以後也會是我的好老婆，我會一直把妳留在身邊。

最後我要感謝一位生命中最重要的人-我的父親。在撰寫論文最後的時刻，父親因重病住院，因此那陣子時常兩地來回跑，假日就到醫院照顧父親。沒想到父親還是沒能撐過去，在五月的時候去世了。也因為如此，這篇論文有一部份是在醫院陪父親的時候寫的，也有一部份是半夜守靈的時候寫的。父親走得突然，對我的打擊很大，但也同時讓我成熟許多，對人生有更多體悟。

能從交大畢業，是我父親、母親最大的期待與驕傲。我終於沒有辜負您們的期待，將從這個殿堂離開，到未來另一個殿堂打拼。

目錄

圖目錄

表目錄

第一章　緒論

1.1 研究背景與動機

網際網路是目前最龐大的資料環境，科技的進步加上使用者人數爆增，每天有數以萬計的網頁產生，而網際網路也成為使用者最大的資訊來源。資訊爆炸的現代，要在這麼龐大的資料當中找尋特定主題的相關資料，是相當不方便的。因此，近年來網頁資料擷取技術的研究變得相當熱門。

目前的網頁資料擷取技術主要有三種 [26]：

(1) Web Context Mining

從網路頁的內容中擷取出有用的資訊，然後將資料傳送給使用者，內容包含文字、圖片、影音資訊等。

(2) Web Structure Mining

用來發現網頁間結構的資訊，或以超連結(Hyperlinks)的方式連結兩個相關的網頁之間的關係。

(3) Web Usage Mining

是以 Data Mining 的技術從網頁中去發現使用者對網頁的使用行為，進而了解與提升所需提供的網頁服務內容。

一般網頁資料探勘技術(Web Mining)，主要探勘的資料皆是以 HTML 的網頁內容為主。但是以 HTML 建構的網頁內容大多都是缺乏組織且結構鬆散，也因此提升了網頁資料擷取的難度。網頁資料擷取的相關研究包括資訊檢索、資料擷取、資料探勘、資訊整合等。這些技術皆是要從大量資料中，擷取出有用的資訊，並可進一步用來決策推論。

人類在閱讀網頁時，會透過許多視覺化線索(visual cues)，例如背景顏色、文字大小、表格排版等線索，來辨認網頁中不同的區域[10]。而且網頁的設計者在建置網頁時，為了使版面更容易閱讀，通常會把一些相關的內容放在一起，形成一個區域，不同區域之

間就用視覺化分隔(visual separators)來區隔。例如用水平線、空白、框線、對比顏色來做區隔。這些區域就稱為網頁區塊(block)[9]。

網頁是由多個區塊所構成，一個網頁可以切割成許多個區塊。每個區塊是由特定主題以及內容所組合成[13, 14]，若該區塊可以提供使用者有用的資訊(useful)，且對讀者是有意義的(relevant)，那麼就稱之為資訊內容區塊(Informative Content Block)[21, 24]。

以圖 1 為例，方框中的區塊是主題是“頭條新聞”，區塊中每一個元件都是在表達同一個新聞事件，包括新聞圖片、新聞標題、新聞摘要、新聞全文、相關新聞連結等。從使用者的觀點來看，在瀏覽一個網站時，是以區塊為單位，先找到要閱讀的主要區塊後，再進一步進行詳讀。以圖 1 的頭條新聞區塊為例，使用者在閱讀時，心中想著要瀏覽『頭條新聞』的區域，因此網頁一呈現出來之後，目光會先掃描『頭條新聞』，然後在心中形成大致的範圍，如圖 1 框起來的部份[10]。接著使用者才認知到這個區塊，是由頭條新聞標題、新聞圖片、新聞標題、新聞摘要、及相關新聞等元件所組成的[9]。

由此可知，每個網頁是由多個不同主題的資訊內容區塊所組成，這些區塊所包含的元件皆有一定的相關性、都是在講同一件事、且具有較緊密的排版關係。綜合相關文獻，本研究將這些具有特定主題的資訊內容區塊，定義為主題區塊(Theme Block)。

從網頁中擷取出主題區塊(Theme Block)，具有下列三個好處：

- 第一、解決資訊過載問題

網頁是由許多不同主題的區塊組成，若能夠自動地將這些區塊辨識出來，使用者就可以只閱讀感興趣的主題區塊。

- 第二，應用在各種平台

將主題區塊轉換成易於儲存、檢索與分析的結構化資料，例如 XHTML，就可以把各個區塊應用於手機或 PDA 等視窗較小的智慧型手持設備，只顯示出使用者需要的那一塊資訊。

- 第三，個人化網頁

應用於使用者個人化自訂網頁，即使用者可以從各個網頁中取出他需要的區塊，將所有區塊整合在一起，組成一個對使用者最有資訊價值的個人化網頁。

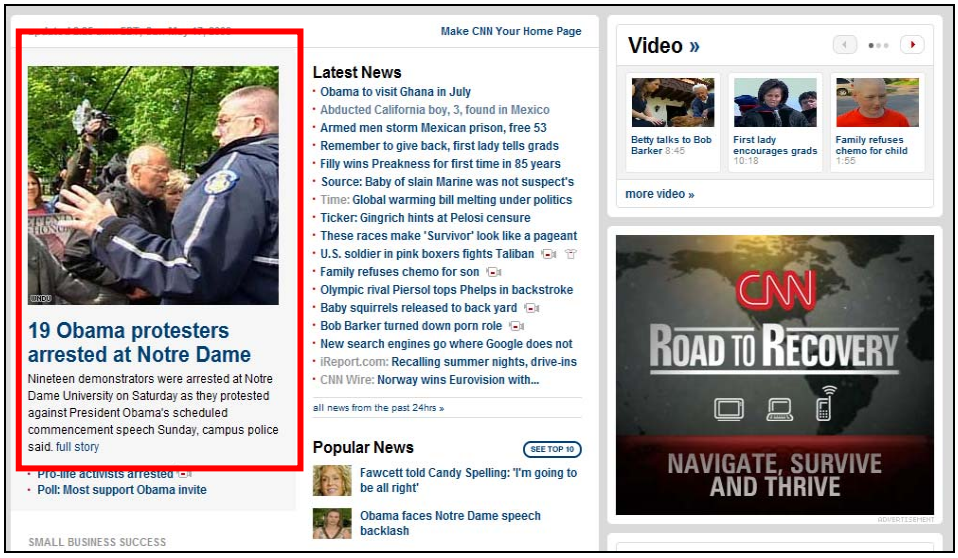


圖 1、由眾多主題區塊所組成的網頁

來源：www.cnn.com

1.2 研究目的

本論文提出一個以賽局為基礎的網頁主題區塊擷取方法，以『主題區塊』為首要考量，來處理各種網頁，包括結構化與非結構化網頁，以及含有一筆到多筆記錄的單一網頁，使得擷取出來的每個『主題區塊』都在描述同一件事，並且是已經將同網頁中相同主題的區塊做過整合。

本論文之目的有三點：(1)處理結構化及非結構化網頁、(2)處理含有一到多筆記錄的單一網頁、以及(3)整合內容相似的區塊。這三個研究目的說明如下：

(1) 處理結構化及非結構化網頁

目前現有的網頁資料擷取方法，皆是針對結構化網頁[26]來做處理。所謂結構化網頁，是指網頁內容係由網頁伺服器在後端抓取資料庫的記錄之後，依照固定的樣板格式，再透過動態網頁技術顯示在頁面上，如圖 2 所示，網頁中每一筆書籍資料的記錄都是由程式動態產生，因此每筆記錄的樣版格式是相同的。而裡面的圖片與資料則是從後端資料庫抓取出來。

結構化網頁有兩個特性：(1)具有相同的樣式(pattern)、(2)區塊間彼此相鄰。目前的網頁區塊擷取技術，大多皆是針對結構化的網頁[26]。如圖 2 中每筆書籍記錄的部分，

就是網頁區塊。由此可知，結構化的網頁區塊具有固定的資料型態與架構，擷取結構化區塊的技術，是先把網頁轉成 DOM Tree 架構[33]，再從 DOM Tree 架構中找出多筆相似的子樹(sub-tree)，經由計算子樹間的相似度之後，就可以推斷出哪幾棵子樹是具有相同樣版格式(pattern)，而這些子樹就是網頁中重複出現的重要區塊了。因此現有方法對於結構化區塊的處理正確率都很高。

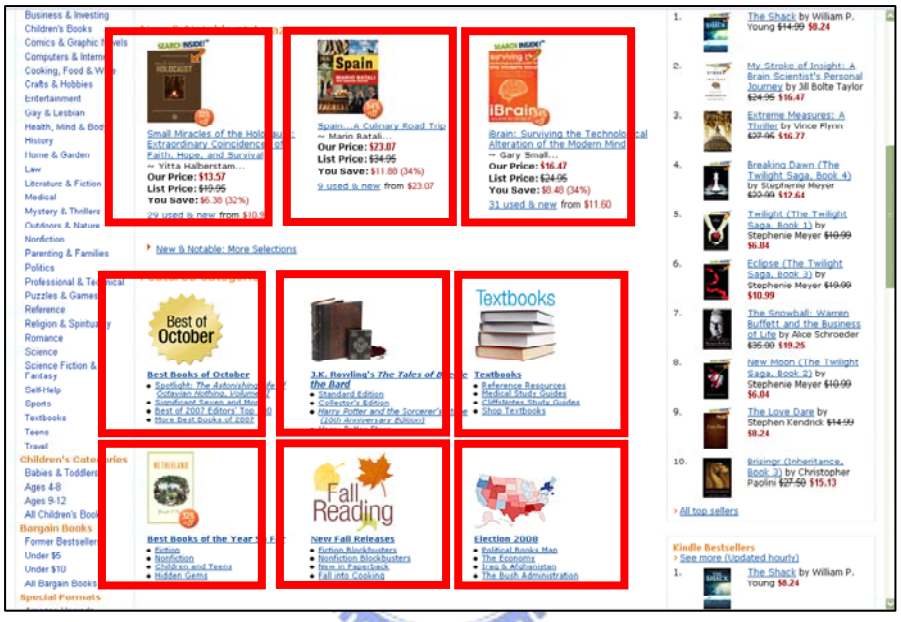


圖 2、結構化網頁

來源：http：//www.amazon.com/

然而，只針對結構化網頁的擷取技術，並無法完全適用於真實情況。如圖 3 所示，網頁工程師通常會為了排版好看、容易閱讀等理由，依照使用者的觀感，自行設計各種大小不一的樣版格式，而分成各種樣版格式不同的區塊，因此目前仍有許多網頁是非結構化，而現有的研究在處理非結構化網頁的效能並不高。

(2) 處理含有一到多筆記錄的單一網頁

現有網頁擷取方法，如[17,7]，是經由計算網頁裡每筆記錄的相似度，找出共同的樣式，再建構成要擷取的網頁區塊。這樣的方法在含有多筆記錄的單一網頁有顯著的效果，但是若遇到只含有一筆或極少記錄的網頁，由於無法跟其他記錄做比較，因此就很難準確的擷取出區塊。

(3) 整合內容相似的區塊

網頁的眾多主題區塊當中，可能有幾個包含相同主題，若使用者要瀏覽同主題的區塊，就會希望把同主題的區塊整合在一起。例如一個網站的上方有體育相關的主題區塊，在下方亦有一個體育相關的主題區塊，這兩個區塊具有相似的主題，但卻是位在相隔甚遠的兩個地方，可能造成瀏覽者在閱讀時的不方便。因此本論文透過計算區塊間的相似度，將在 HTML 上不連續、在視覺線索上不相鄰，但區塊內容相似的區塊整合在一起。

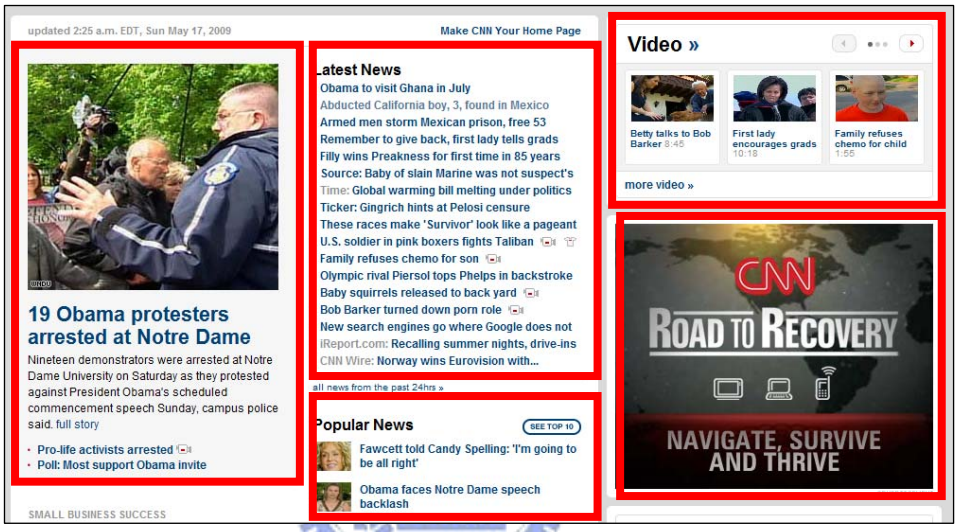


圖 3、非結構化網頁

來源：www.cnn.com

1.3 章節規劃

本論文的章節規劃如下：第一章緒論闡述研究背景、動機、目的，並概括描述研究的整體架構。第二章文獻探討，概述 DOM Tree 及現有研究對於網頁處理的方法，包括網頁區塊化、網頁資料擷取，針對各方法之優劣進行比較。第三章為本論文所提出的 GRAB 演算法，可分成(1)建構 DOM Tree、(2)建構主題區塊樹(Theme-based Tree)、(3)主題區塊樹之整併及分割等三大步驟。第四章為系統實作與分析，概述雛形系統之架構、程序運作流程、及實際畫面。接著設計了兩個實驗來驗證 GRAB 擷取主題區塊出來的效果，並說明實驗結果與討論。第五章為本論文做總結，進一步描述未來尚可研究的方向。

第二章　文獻探討

本章首先介紹了 DOM Tree 架構，再介紹現有三大類的網頁資料擷取技術，包括：經由人工或自動學習的 Machine Learning 方法、找尋重複記錄的 Automatic Extraction、以及自行定義擷取規則的 Rule-based Extraction。

2.1 DOM Tree 架構

近年來已有許多對於網頁資料處理與擷取方面的研究，絕大多數研究都採用 DOM Tree 架構來表示一個網頁的結構。DOM 的全名是 Document Object Model，由 W3C[33]所定義的一種規範，將網頁的標籤結構以樹狀資料結構來表示，目的在於表現單一網頁的結構性。使用 DOM Tree 架構可表現網頁的初步結構，且能夠清楚辨識出標籤之間的階層關係，在處理標籤、追遡標籤、或搜尋標籤上都極為方便。圖 4 是一個簡單的 HTML 語法範例，以及其對應的 DOM Tree 架構。

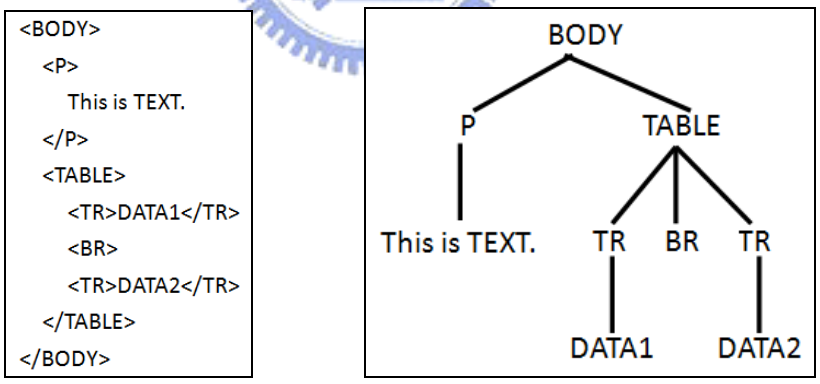


圖 4、HTML 語法範例與對應的 DOM Tree 結構

透過樹狀的表示方式，能清楚看出網頁結構中的上下層次關係，以及節點的歸屬關係。DOM Tree 的基本概念如下[29]：

(1)被 Non-empty Tag(例如 ：<html>、</html>、<table></table>等)所涵蓋的內容，都以樹狀結構中的子節點來表示，並置於 Non-empty Tag(父節點)之下。

(2)Empty Tag(例如：
、<hr>、<p>等)部份與其它位於同一個 Non-empty 下的標籤，以兄弟點節方式，存放在同一個父節點之下。

2.2 現有網頁區塊擷取方法

網頁資料擷取的方法，可歸納為三大類： (1)Machine Learning、(2)Automatic Extraction、(3)Rule-based Extraction。

2.2.1 Machine Learning(Wrapper Induction)擷取方法

此方法又稱為 wrapper induction。Line Eikvil[15]對 Wrapper 的定義為：由一組擷取規則，加上應用這些規則來進行擷取動作的程式碼所構成的。所以 Wrapper[28]是一種自動化的資訊擷取程式，能自動的將文件包含的資訊擷取出來，並透過單一格式顯示出來。Wrapper induction 則是透過監督式機器學習的方法，讓系統能夠自動根據使用者所標記的擷取部份來產生擷取規則，再進行內容的擷取，如 SoftMealy[7,8]和 STALKER[12]的系統。

產生擷取規則的方式可歸納為兩種方式：(1)透過學習、(2)手動直接建立規則。採取學習的機制來訓練擷取程式，稱為學習法(Learning Rules)，事先為系統設計一套學習演算法，讓程式自動從網頁中學習，進而產生規則，讓系統依循規則進行擷取工作。另一種方式則稱為建立擷取規則法(Building Rules)，是透過定義擷取規則來指示擷取程式所要進行擷取工作的內容，只要設定好擷取規則，擷取程式就可以工作。

WIEN(WRAPPER Induction ENvironment)系統是由 Kushmerick[18,19]等人提出，能透過學習方式產生擷取規則。WIEN 由六種模組所組成，模組之間能互相搭配，判斷標籤的結構，分別處理不同的標籤樣式，產出擷取規則。這六種模組分別是判斷標籤及左右兩方標籤的 LR-Wrapper(Left-Right)模組、擷取標籤前後左右資訊的 HLRT-Wrapper(Head-Left-Right-Tail)模組、以資訊的前後界定標籤來做判別的 OCLR-Wrapper(Open-Close-Left-Right)模組、結合 HLRT 與 OCLR 的 HOCLRT-Wraper 模組、以及能處理巢狀網頁的 N-HLRT Wrapper 模組。

Chidlovskii[1]等人提出人 CRR，一開始需要以手工方式來標記要擷取的部份，以便讓擷取程式做學習，之後再讓程式透過歸納法來產生擷取規則。但無論是經由學習方式還是手動方式，Wrapper Induction 的效果皆取決於產生出來擷取規則的正確性，若訓練

資料選得不好，或手動標記有誤，皆會影響規則正確性。再者，透過餵養訓練資料(training data)來學習，或者經由手動標記來學習，都非常耗費時間，效率不彰。

2.2.2 Automatic Extraction 擷取方法

此方法則是屬於不須標註(Non-Labeling)的方法，藉由找尋相似或重複資料樣式，自動找出應該要擷取的部份。其中又可概分成兩類：(1)在同一個網頁內互相比較，比較同一個網頁內所有資料記錄是否具有相同樣式。(2)在新舊網頁之間比較，例如新網頁與三天前的舊網頁比較，若有某些記錄的資料有更新或異動，那麼這些記錄就可能是重要的資訊[30]。

IEPAD[6]、DeLa[15]、MDR[2]、DEPTA[3,4]等系統，皆是屬於 automatic extraction 第一類的方法。其做法是認為所謂重要的資料區塊，是指多筆重複相似的資料記錄(Data Records)，因此主要目標，就是設法找出這些重複資料記錄[32]。

IEPAD[6]是 2001 年由張嘉惠博士等人所開發的系統，係以自動化的方式產生 Wrapper 藉以擷取資料。IEPAD 認為每一個網頁中皆含有多筆重複出現的資料，具有相同的 pattern，若能找到網頁中重複出現的記錄中具有相同的 Prefix 的字串，即是所謂的 Repetitive Pattern。下一步再從這些 Repetitive Pattern 中找到一個滿足使用者所給的 Maximal Repeat 條件，接著計算每個 Pattern 的變異性與密度來決定此 Pattern 是否為真實的資料記錄，並從剩餘的 Pattern 推導出擷取規則。此方法對於多筆記錄的網頁擷取能力相當好，但對於網頁中只有單一筆記錄的網頁，則無法進行擷取。

DeLa[15]是在 2003 年由 Wang 等人所發表的系統，主要包含兩大功能：資料擷取與資料標識。Wang 認為網頁中的資料記錄大多是連續且重複的，若能將網頁中所有標籤建立出一棵 Suffix-tree 來找出連續重複的樣式(C-Repeated Patterns)。有了連續重複的樣式，就可以用來產生 Wrapper。圖 5 為 DeLa 系統的架構圖。

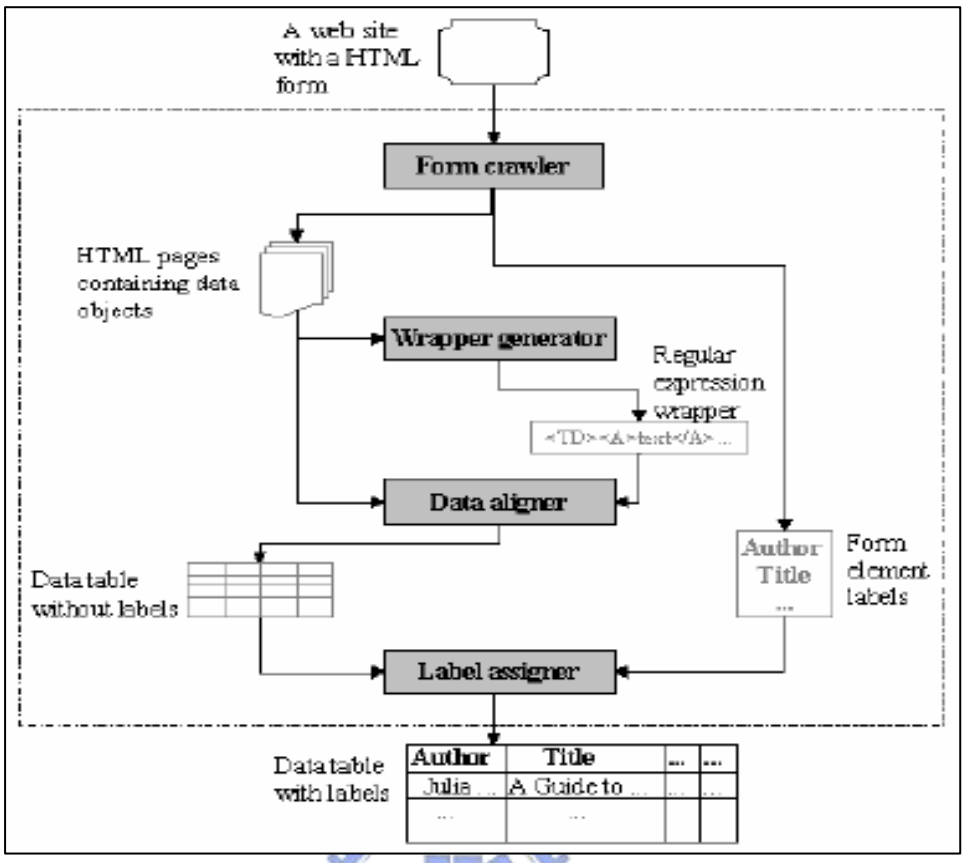


圖 5、DeLa 資訊擷取系統的架構圖

來源：Data Extraction and Label Assignment for Web Databases [15]

MDR[2]為 Liu 等人在 2003 年所發表的系統，Liu 提出兩個概念：一般化節點(Generalized Nodes)和資料區域(Data Region)。一般化節點用來表示包含多個節點的集合，此集合裡所有的節點皆具有相同父節點，且所有節點皆左右相鄰；資料區域表示一個或多個一般化節點之集合，如圖 6 中，灰色網底的節點即為一般化節點，其中節點 5、6 較相似，同屬於資料區域 1；節點 8、9、10 同屬於資料區域 2;節點 14、15 的一般化節點，跟節點 16、17 的一般化節點，同屬於資料區域 3。

DEPTA[3]是 Liu 等人在 2005 年所發表的系統，它改善了 MDR 的擷取功能，利用 MDR 所找出來的資料記錄中，透過 Tree Alignment 的方法，將資料轉換成結構化的格式，能夠存入資料庫中。為了更進一步改善計算 Tree node 相似度的 STM (Simple Tree Matching)演算法，Liu 等人在 2006 提出 ESTM (Enhanced Simple Tree Matching)演算法[4]，在計算兩棵樹的最大符合(maximum matching)的時候，把節點的內容也一併加入考量。學者 Y.Kim 等人則是在 2007 年提出 HTML Tree Matching 演算法[27]，認為每個節

點的內容皆不同，其權重也應該有所不同。Y.Kim 等人藉由節點值(Node value)的計算，改善原本的 STM 方法。圖 7 為整個 DEPTA 系統之架構與流程

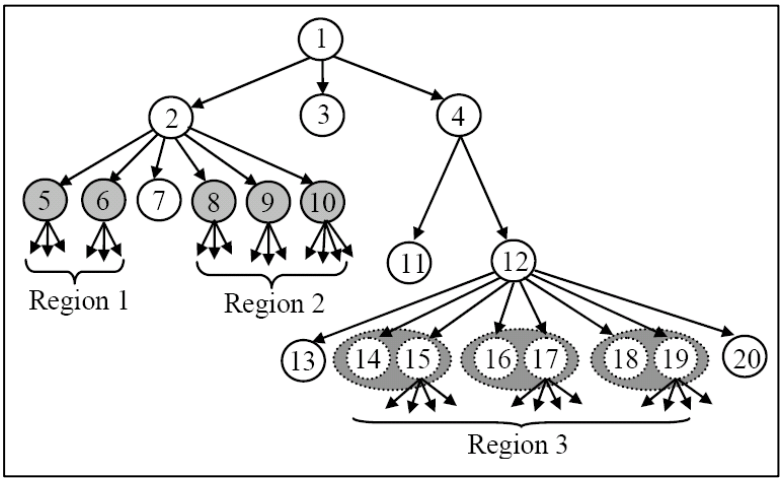


圖 6、一般化節點與資料區域

來源：Mining Data Records in Web Pages [2]

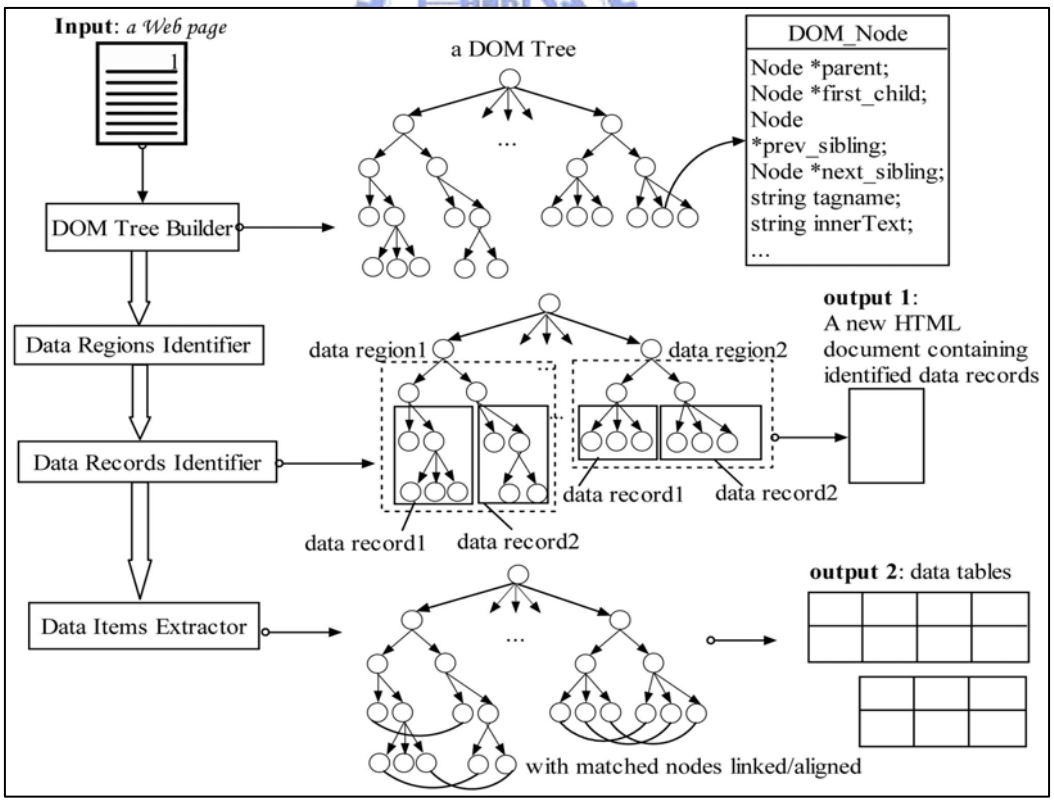


圖 7、DEPTA 系統架構

來源：Web Data Extraction Based on Partial Tree Alignment [3]

S.J. Lim 與 Y.K. Ng 提出 CDA(Change Detection Algorithm)，是屬於 automatic extraction 第二類方法[20]，將新網頁與舊網頁的網頁結構轉換成分支(Branch)，每個分支分別給予權重值，接著針對分支兩兩進行比較，計算新舊分支的差異度。差異度最大的分支，就代表新網頁跟舊網頁不同之處。此方法的優點是能將網頁結構轉成易儲存的表格形式，能節省網頁儲存空間；而缺點是僅能比對出新舊網頁結構化表格的變動，對於非結構化的異動則無法處理。

2.2.3 Rule-based Extraction 擷取方法

以規則為基礎的擷取方法是最直覺方式。其做法是先定義一組規則，然後以深度為優先搜尋(Depth First Search, DFS)針對每個節點去做 IF-THEN 的判斷，若判斷之後認為該節點及其底下的所有子節點可以獨立成為一個主題區塊，則以該節點為樹根，擷取出整棵子樹，所得到的一棵子樹就是一個主題區塊[10,25]。

Jobbot[32]是范綱岷等人在 2001 年發表的系統，透過樹狀結構來表示網頁的結構，然後由使用者決定網頁中所要擷取的資訊位置，在樹中建立一條路徑，做為規則來源。當系統要進行擷取的時候，就會依照先前產生的規則來執行，將指定路徑上的資料擷取回來，並能透過資訊融合的規則進行資訊整合。

VIPS (Vision-based Page Segmentation)演算法[10,23]，是 Yu、Ma 等人在 2003 年提出，其概念是先把整個網頁以 DOM Tree 架構表示，DOM Tree 中的每個節點，都代表一個虛擬區塊(Visual Block)。但因為有些區塊太大，必須進一步做切割，分成更小的區塊。切割的方法是以背景顏色、文字數目、區塊大小等視覺上的線索來做依據，若一個區塊內的這些特徵差異性太大，就必須將區塊切割成更小。切割出許多虛擬區塊之後，接下來偵測這些區塊之間的分隔線，依照其權重值來判斷該分隔線是不是重要的。決定出分隔線之後，所有在分隔線同一邊的區塊都將合併成一個區塊，再針對這個區塊計算顆粒值(Granularity)，又稱為 DoC。若區塊的 DoC 小於某個門檻值，就重複進行切割，直到所有區塊都滿足門檻值為止。

以規則為基礎的方法，具有系統與邏輯分離的優點，在定義完規則之後，若有要修

改規則，只需要修改規則庫即可，核心的擷取程式不必重新撰寫。

然而這樣的方式的缺點是太過於主觀，容易因為規則不足或例外情況等因素，造成誤判，且網頁結構往往相當複雜，難以制定通用的規則。

這三大類網頁擷取方法共通的優點，是在處理結構化網頁的效能都很高。因為結構化網頁具有重複的 Pattern，要針對結構化網頁定義規則或從中學習擷取規則，也比較容易。但這三種方法仍有可以改善的空間，分別整理如表 1 所示：

表 1、文獻探討整理與比較

網頁擷取方法	現有研究	方法概述	改善空間
Machine Learning (Wrapper Induction)	SoftMealy STALKER WIEN CRR	透過手動或自動方式，讓系統能夠自動學習，產生擷取規則，再進行內容擷取	(1) 若訓練資料選得不好，或手動標記有誤都會影響正確性 (2) 非常耗費時間，效率不彰
Automatic Extraction	IEPAD Dela MDR DEPTA	認為網頁中的資料記錄大多是連續且重複的，若從中找出重複的 Pattern，就可以用來產生 Wrapper	(1)對於網頁中只有單一筆記錄的網頁，則無法進行擷取 (2)對於非結構化或較複雜的網頁，計算出來的樹相似度可能效果不彰
Rule-based Extraction	Jobbot VIPS	事先定前一組規則，或在使用者操作時建立規則，等到要進行擷取時再逐一做判斷	(1)可能因規則不足或例外情況等因素，造成誤判 (2)網頁結構往往相當複雜，難以制定通用的規則

第三章 一個以賽局理論為基礎的網頁主題區塊擷取演算法

這一章首先定義本論文要解決的問題，並說明 GRAB 演算法為何能解決這些問題。接著再介紹 GRAB 演算法的每一個步驟，以及各步驟的計算方法。

3.1 問題定義

從一棵 DOM Tree 之中判別出哪些節點可以直接擷取出來，形成一個主題區塊，是最核心也最重要的部份。在第二章 2.2 節的部份，整理了現有的三大類網頁區塊擷取方法，以及它們各自的缺點：

- 採用手動標記的方式，非常耗費時間，效率不彰
- 非結構化網頁或只含一筆記錄的網頁，難以找出重複相似的資料記錄
- 定義擷取規則，可能因規則不足而有誤判或例外情況出現

因此，本論文提出一個以賽局為基礎的網頁主題區塊擷取方法(GRAB)，透過計算資訊含量的方式來判斷是否要擷取。資訊含量的計算方式，是站在貼近使用者的觀點，同時考慮了網頁元件所在的視覺位置，也考慮了不同網頁元件的特性，給予不同權重值，因此不須要人工建立標記、進行學習、或自行定義規則，也可以針對非結構化網頁，計算出每個網頁元件的資訊含量，進而判斷是否要組成主題區塊。除了能避免上述三種方法的缺點外，更利用賽局理論求取最適策略的概念，考慮節點與其他節點的關係，以及做決策時彼此的影響，讓每個節點做出最適的決策。

3.2 GRAB 演算法

本論文所提出的 GRAB 演算法可分成三個步驟(Step)：

- **Step 1 轉換(Transformation)**

此步驟的目的是將原始網頁轉換成 DOM Tree 的樹狀架構，概略的表現整個網頁結構及階層性，並且對於網頁也有初步的區隔。

Input：HTML 網頁

(1) 利用 Tidy 套件，清除不必要的 Tag

(2) 修復有錯的 Tag

(3) 轉換成 XHTML 格式，即代表一棵 DOM Tree 的架構

Output：XHTML 檔案(DOM Tree)

■ **Step 2 篩選(Selection)**

此步驟的目的，是在 Step 1 產生的 DOM Tree 中，逐一計算 DOM Tree 裡面每一個節點的資訊含量，再透過正規式的賽局(Normal Form Game)的方式，判斷該節點是否可以自己獨立成為一個主題區塊。在此步驟適合獨立成為主題區塊的節點，將逐一加入主題區塊樹(Theme-based Tree)，漸漸形成一棵樹。步驟 2 又可細分成兩個小步驟：

Step 2.1 挑選主題區塊

Input：DOM Tree

透過以深度為優先搜尋(DFS)方式，遞迴的處理每個節點

(1) 計算節點的資訊含量(Information Content, IC)

(1-1)計算所在位置權重(Position Weight, PW)

(1-2)計算元件權重(Component Weight, CW)

(1-3)經由 PW、CW、以及子節點資訊含量的和，算出自己的資訊含量

(2) 節點進行賽局決策

(2-1)自己節點的報酬和下面所有子節點的報酬進行賽局

(2-2)找出雙方均衡點，決定要獨立成主題區塊的節點，就暫存起來

Output：所有決定要獨立的主題區塊

Step 2.2 建立主題區塊樹

Input：所有決定要獨立的主題區塊

(1) 從最後要獨立的主題區塊開始，逐一建構成一棵樹

Output：主題區塊樹(Theme-based Tree)

■ **Step 3 精煉(Refinement)**

此步驟的目的，是調整上個步驟所產生的主題區塊樹，同主題的區塊進行合併，主題性不足的區塊，也就是小於門檻值的節點，將放入步驟二，繼續分割。最後，會得到一棵調整過後的主題區塊樹，其中的每一個樹葉節點，就是一個主題區塊。

Input：Theme-based Tree

逐一處理 Theme-based Tree 的每個節點

(1) 若資訊含量大於門檻值 H，代表主題性已足夠，不須要再精煉

(2) 若資訊含量小於門檻值 L，代表主題性不夠，要放入 Step 2 再處理一次

(3) 若資訊含量介於 H 與 L 之間，則先標記起來。

(4) 所有已標記的節點，用 TSTM 演算法計算相似度，相似的節點合併成一個一般化節點(Generalized Node)

Output：Refined Theme-based Tree

以下就用三個小節來詳細說明 GRAB 演算法的三個步驟。

3.2.1 GRAB 演算法步驟 1：轉換(Transformation)

一般的 HTML 網頁，可以先經由轉成 XML 格式之 XHTML，處理掉 HTML Tag 的缺漏問題，再清除不必要的 tag，例如 JavaScript、註解等資訊後，轉換成 DOM Tree 的樹狀架構。如圖 8 所示：

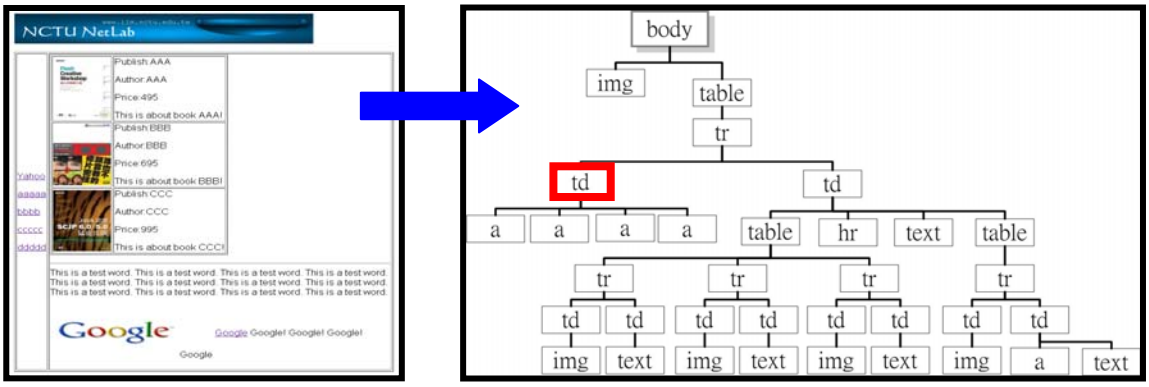


圖 8、網頁轉換成 DOM Tree 架構

DOM Tree 架構就代表網頁的階層架構，其中每一個節點都是一個 HTML Tag。但光是從 Tag 無法正確的取出網頁的資料，而是需要取出完整的子樹出來，才能得到一塊

有意義的網頁區塊資訊。如圖 8 所示，若只取出<TD>這一個節點的內容，並無其他意義；若是取出包含<TD>之下的整棵子樹，亦即取出<TD> <A …> <A …> <A …> <A …> <A …> </TD>就相當於擷取出這個表格的一個儲存格的內容，這就有意義了。因此在本論文中，要取出某個節點的內容，指的就是擷取出該節點之下的整棵子樹。

在 W3C 的 HTML 規格書中，HTML DOM node 可分成三種：

(1)Insignificant node

無法在瀏覽器上顯示出來的節點。此類的節點包括只含有空白的節點、換段符號、 等等，例如註解文字、隱藏的 tag(例如<INPUT type=hidden>)等等。值得注意的是
並不是 Insignificant node，它是屬於 line-break node。

(2)Inline node

會影響文字顯示外觀的 html tag，包括：<A>、<ACRONYM>、<ABBR>、<B>、<BIG>、<CITE>、<CODE>、<DEL>、<DFN>、<EM>、<FONT>、<I>、<INPUT>、<INS>、<NOBR>、<KBD>、<Q>、<SAMP>、<SMALL>、<STRONG>、<SUP>、<SUB>、<TT>、<U>、<VAR>等。

(3)Line-break node

既不屬於 Insignificant node，也不屬於 Inline node 的節點，就歸類為 Line-break node。

在本論文中，為了更明確地分辨節點的屬性及階層關係，因此將 W3C 定義的 Line-break node 擴充為以下兩種節點：

(1)Visible node：可以直接在瀏覽器上看到的節點，其下沒有子節點。包括以下幾個：

- Text node：僅包含文字內容的節點
- Url node：具有超連結功能的節點，如：<a href="…">
- Image node：其內容為圖片、或 Flash 等能夠直接在瀏覽器上看見的物件
- separate node：HTML tag 本身就具有分隔效果，如：<hr>

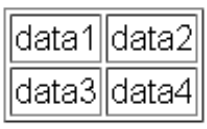
(2)Format node：節點內包含多個子節點，並由特定格式來呈現。包括：

- Free format node：包含多個 Visible node，且 Visible node 之間不相似。例如同時包含圖片和文字的節點。
- Area format node：本身具有區隔性，能將內容做表格式的呈現，如：<table>、 、、<form>等
- List format node：包含多個 Visible node 或 Format node ，且是以相似格式重複出現
- Combined format node：包含多個 Visible node 或 Format node

圖 9 描述了將網頁元件處理成 XHTML 的過程。首先，一個網頁會包含著各種元件，其中註解、JavaScript、CSS 樣式等非必要的資訊，要先在這個步驟中拿掉，僅留下 HTML Tag。接下來，將 HTML 轉換成符合 XML 規範的 XHTML 格式。本論文是利用 Tidy 套件來處理，Tidy 可在各種語言上執行，能夠自動判斷及修復容易寫錯的 HTML 標籤，不論是標籤的順序，或是標籤是否有結尾等等，接著可以透過標準的轉譯方式，將合法的 HTML 轉換成為 XML 格式的檔案。

此階段產出的合法 XHTML 檔，就是一個 DOM Tree 的架構。一個合法的 XHTML 檔，會有一個唯一的根節點，其下有數個子節點，每個子節點之下還有其他子節點。相對於一棵 DOM Tree，就如同樹根、子樹節點、樹葉節點等概念。

在轉換成 XHTML 的過程中會為每個節點加入兩個屬性：ccmID(Check Content for Mark Identify)及 nodeType。ccmID 的值從 1 開始遞增，並且是從<BODY>標籤開始寫入，因此所有 DOM Tree 的根節點都是<BODY>。而 nodeType 則是記錄該節點的型態是上述 HTML DOM node 的哪一種。



(a)

```
<table border=1>
  <tr>
    <td>data1
    <td>data2</td>
  <tr>
    <td>data3</td>
    <td>data4</td>
  </tr>
</table>
```

(b)

```
<?xml version="1.0" encoding="UTF-8"?>
 <html xmlns="http://www.w3.org/1999/xhtml">
  <head>
   <meta content="HTML Tidy, see www.w3.org" name="generator"></meta>
    <title></title>
  </head>
  <body>
   <table border="1">
    <tr><td>data1</td> <td>data2</td> /tr>
    <tr><td>data3</td> <td>data4</td></tr>
   </table>
  </body>
</html>
```

(c)

圖 9、原始網頁與 HTML/XHTML 格式之對應

3.2.2 GRAB 演算法步驟 2：篩選(Selection)

演算法的第二步驟，是從 DOM Tree 中挑選出適合的主題區塊，並建構成一棵主題區塊樹(Theme-based Tree)。DOM Tree 架構就代表網頁的階層架構，取出來的一棵子樹，就相當於擷取出網頁當中的某塊內容。每一個 DOM Tree 節點都要去判斷：它是不是具有足夠主題性的節點？若不是，是否要繼續往下找？

(1) 挑選主題區塊 (GRAB Step 2.1)

- **計算資訊含量(Information Content, IC)**

學者 R.Song 等人在[16]提出了 Block Importance Model，並在研究中指出，在一個網頁當中，依照區塊的位置、區域大小、內容量的不同，區塊所擁有的重要性也有所不同。本論文採取計算資訊含量的方式來判斷 DOM node 是否可以獨立成為主題區塊，所謂資訊含量，是指每個主題區塊有包含足夠豐富的元件。以圖 10 為例，主題區塊(a)包含了標題文字、新聞圖片、新聞摘要等資訊，其資訊含量就比較高；而主題區塊(b)只包含了一張圖片，其資訊含量相對而言就比較低。

當一個節點要判斷自己是否要獨立擷取成為主題區塊時，首先要計算自己節點本身的資訊含量，若高於門檻值，代表這個區塊的內容夠豐富了，就將這個節點擷取出來。

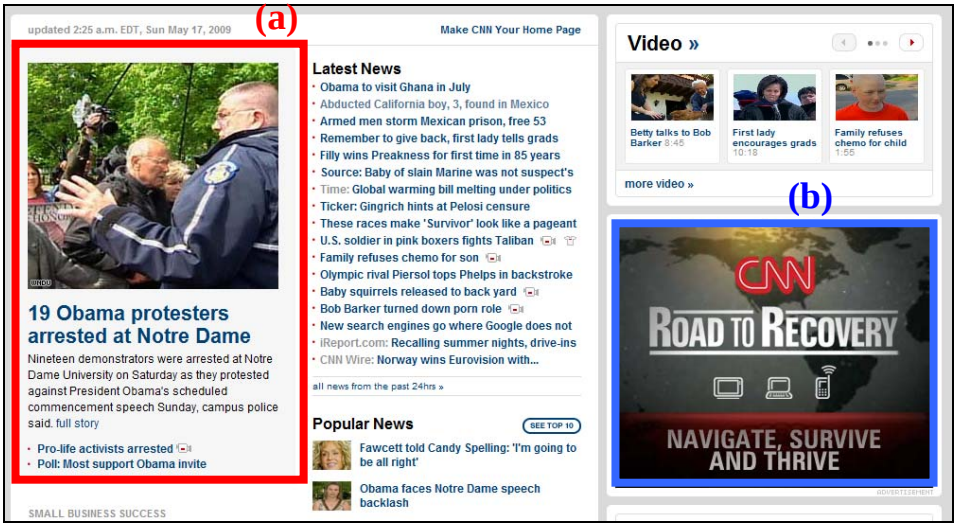


圖 10、資訊含量不同之主題區塊

來源：www.cnn.com

計算資訊含量的方法，是以每個網頁元件的特性來做判斷，網頁設計者在放置這些元件時，必然有他的目的及考量，事先決定好在這一塊要顯示出哪些資訊內容。因此本方法更能客觀地貼近使用者的觀點，擷取出對使用者有意義的主題區塊。

資訊含量的採用遞迴計算方式，首先計算每個區塊的所在位置權重(Position Weight, PW)、區塊元件權重(Component Weight, CW)之後，再乘上所有子節點資訊含量的和，所得到的結果；但若要得知子節點的資訊含量，就要繼續遞迴往下做，直到傳回樹葉節點的資訊含量。

- **區塊所在位置權重(Position Weight, PW)**

使用者在瀏覽網頁時，眼睛注視網頁位置順序，決定了網頁區塊所在位置的權重。學者 George[5]針對網頁設計是否影響其使用性進行調查，發現使用者閱讀網頁的順序是由上到下，由左到右，並且期待在頁面上方找到常用連結，如功能表單、網頁地圖等。

美國 Poynter[22]新聞研究院於 2004 年公佈「第三次眼球追蹤」研究報告，經由實驗發現了一般使用者在瀏覽新聞網頁時的眼球移動模式也是由上到下，由左至右。如圖 11 所示，使用者在瀏覽新聞網頁時，眼球首先會停留在頁面左上角，然後在左上角附近稍微瀏覽之後，接下來目光在右邊及左邊來回瀏覽之後，最後才移到右邊及右上角。

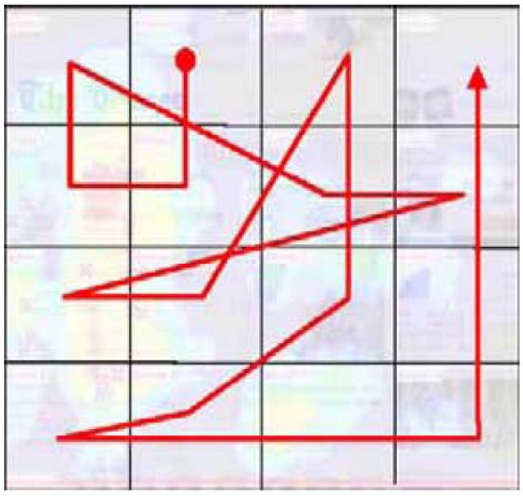


圖 11、瀏覽新聞網頁的眼球移動模式

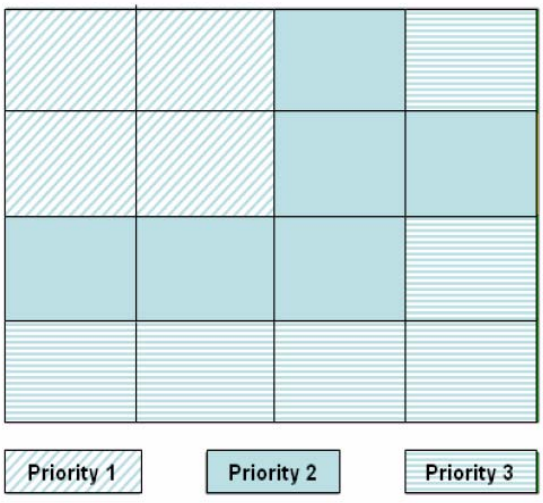


圖 12、網頁區塊位置之優先順序

來源：The Best of Eyetrack III [22]

因此，可以將網頁劃分成4×4的區域，如圖 12 所示，最左上四塊的優先順序最高，其次是中間的部份六塊區域，再來是下面及右上角的六塊區域，接著再依照其優先順序給定不同的權重 ρ ，如表 2 所示。

表 2、網頁位置權重

優先順序	優先順序權重公式 $\rho(\,)$
Priority 1	$\rho(priority\_1) = 0.8$
Priority 2	$\rho(priority\_2) = 0.5$
Priority 3	$\rho(priority\_3) = 0.3$

將整個網頁的高(h)與寬(w)平均分成四等分，則可以計算出每個位置區塊(Position Block)所佔的座標範圍，如圖 13 所示。

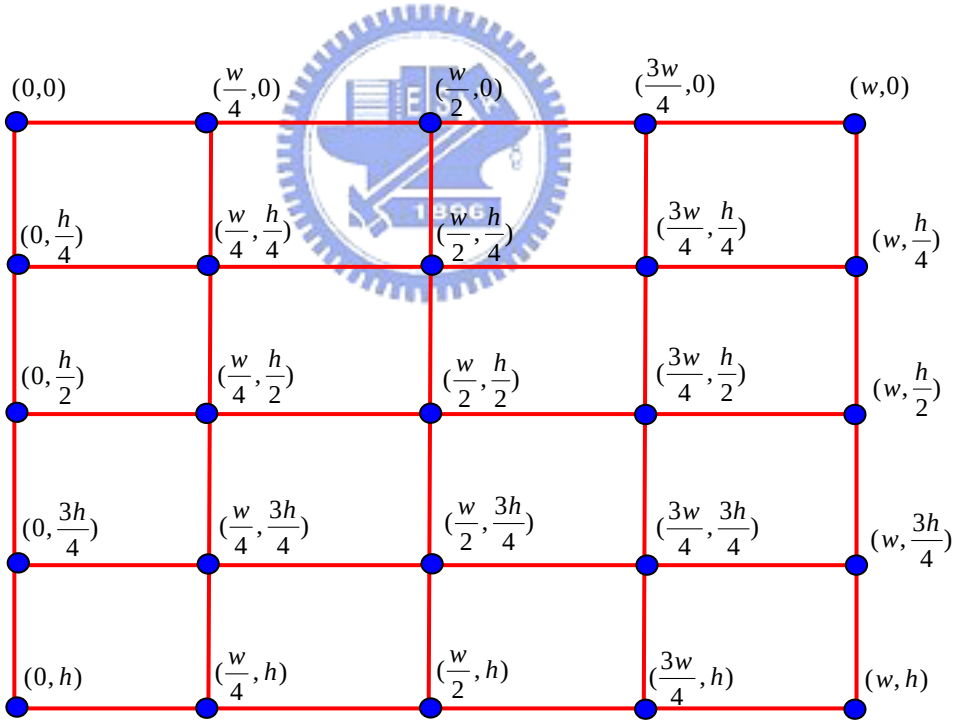


圖 13、各個頂點的座標

接下來，每一個位置區塊利用左上及右下座標來定義其範圍，並以 B(i,j)表示，其中 $i, j \in \{0,1,2,3\}$ ，B(i,j)代表第 i 行第 j 列的位置區塊座標範圍。因此 16 個位置區塊座標範圍如圖 14 所示。

	0	1	2	3
0	$B(0,0)=\left\{(0,0),(\frac{w}{4},\frac{h}{4})\right\}$	$B(1,0)=\left\{(\frac{w}{4},0),(\frac{w}{2},\frac{h}{4})\right\}$	$B(2,0)=\left\{(\frac{w}{2},0),(\frac{3w}{4},\frac{h}{4})\right\}$	$B(3,0)=\left\{(\frac{3w}{4},0),(w,\frac{h}{4})\right\}$
1	$B(0,1)=\left\{(0,\frac{h}{4}),(\frac{w}{4},\frac{h}{2})\right\}$	$B(1,1)=\left\{(\frac{w}{4},\frac{h}{4}),(\frac{w}{2},\frac{h}{2})\right\}$	$B(2,1)=\left\{(\frac{w}{2},\frac{h}{4}),(\frac{3w}{4},\frac{h}{2})\right\}$	$B(3,1)=\left\{(\frac{3w}{4},\frac{h}{4}),(w,\frac{h}{2})\right\}$
2	$B(0,2)=\left\{(0,\frac{h}{2}),(\frac{w}{4},\frac{3h}{4})\right\}$	$B(1,2)=\left\{(\frac{w}{4},\frac{h}{2}),(\frac{w}{2},\frac{3h}{4})\right\}$	$B(2,2)=\left\{(\frac{w}{2},\frac{h}{2}),(\frac{3w}{4},\frac{3h}{4})\right\}$	$B(3,2)=\left\{(\frac{3w}{4},\frac{h}{2}),(w,\frac{3h}{4})\right\}$
3	$B(0,3)=\left\{(0,\frac{3h}{4}),(\frac{w}{4},h)\right\}$	$B(1,3)=\left\{(\frac{w}{4},\frac{3h}{4}),(\frac{w}{2},h)\right\}$	$B(2,3)=\left\{(\frac{w}{2},\frac{3h}{4}),(\frac{3w}{4},h)\right\}$	$B(3,3)=\left\{(\frac{3w}{4},\frac{3h}{4}),(w,h)\right\}$

圖 14、位置區塊座標範圍

圖 14 列出了 16 個位置區塊的座標範圍，可以看出每一個位置區塊的長與寬都佔網頁長與寬的$\frac{1}{4}$。因此可推導出下列位置區塊座標公式(B)，而每個座標所對應到的優先順序則如表 3 所示。

$$B(i,j)=\left\{(\frac{iw}{4},\frac{jh}{4}),\left(\frac{(i+1)w}{4},\frac{(j+1)h}{4}\right)\right\},where\quad i,j\in\{0,1,2,3\}$$

表 3、優先順序權重公式對應表

優先順序權重公式$\rho(\,)$	對應到的優先順序	優先順序權重值
$\rho(B(0,0))$、$\rho(B(1,0))$、$\rho(B(0,1))$、$\rho(B(1,1))$	$priority\_1$	0.8
$\rho(B(0,2))$、$\rho(B(1,2))$、$\rho(B(2,2))$、$\rho(B(2,0))$、$\rho(B(2,1))$、$\rho(B(3,1))$	$priority\_2$	0.5
$\rho(B(0,3))$、$\rho(B(1,3))$、$\rho(B(2,3))$、$\rho(B(3,3))$、$\rho(B(3,2))$、$\rho(B(3,0))$	$priority\_3$	0.3

由於網頁每個元件的大小不一，可能落在其中一塊座標範圍內，也可能落在好幾塊座標範圍內。以圖 15 為例，右下角的廣告區塊就同時落在 4 個座標範圍(a、b、c、d)之內。由此可知，每個網頁元件可能只會落在 1 格內、可能落在同一列內、可能落在同一行內、也可能落在多行與多列的範圍內。

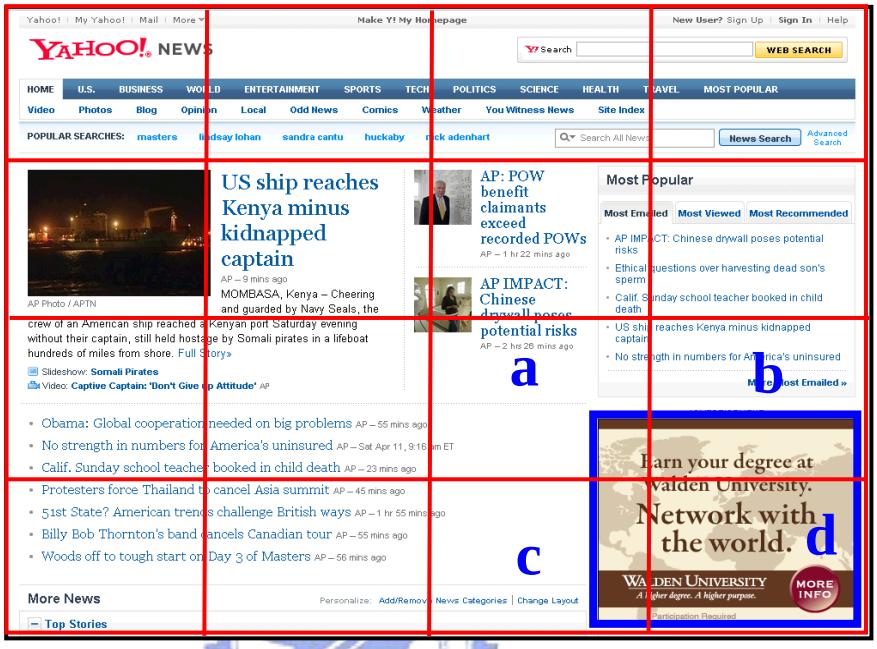


圖 15、落在不同座標範圍的元件

來源：www.yahoo.com

以下分別概述這四種情況：(1)元件落在同一格內、(2)元件落在同一列內、(3)元件落在同一行內、(4)元件落在多行及多列之內，並用較粗的藍色方框表示網頁元件，較細的紅色方框代表位置座標範圍，分別推導元件落在位置區塊的面積權重計算方法。計算的概念是要設法找出網頁元件落在 16 個位置區塊中的哪幾個?面積有多少?再各自乘上優先順序權重，就可以算出區塊所在位置權重(*PW*)。

假設網頁元件的長為 m，寬為 n，左上角座標為(x, y)，右上角座標為$(x+m, y)$，左下角座標為$(x, y+n)$，右下角座標為$(x+m, y+n)$，且(x, y)是落在 $B(i,j)=\left\{(\frac{iw}{4},\frac{jh}{4}),\left(\frac{(i+1)w}{4},\frac{(j+1)h}{4}\right)\right\}$座標範圍內，則：

◆**(情況一)元件落在同一格內：**

若元件僅落在同一格內，如圖 16，其計算方式就非常簡單，只要算出元件面積，再乘上所在位置權重即可。

➢ **公式推導：**

元件落在位置區塊的面積權重(PW)

= 元件面積×所在權重

= $\left((m \cdot n) \times \rho(B(i,j))\right)$

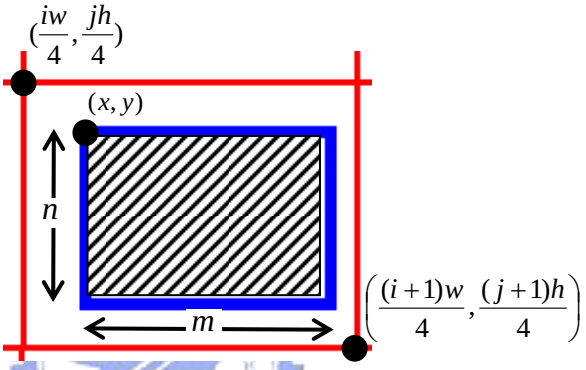


圖 16、落在同一格的元件

◆**(情況二)元件落在同一列內：**

元件落在同一列，可能會橫跨好幾條直軸線。假設元件橫跨了 k 條直軸線(k≧1)：

➢ 若元件只橫跨 1 條直軸線(k=1)，如圖 17，則只需要計算以 $X = \frac{(i+1)w}{4}$ 為直軸線所劃分出來的左右兩段面積即可。

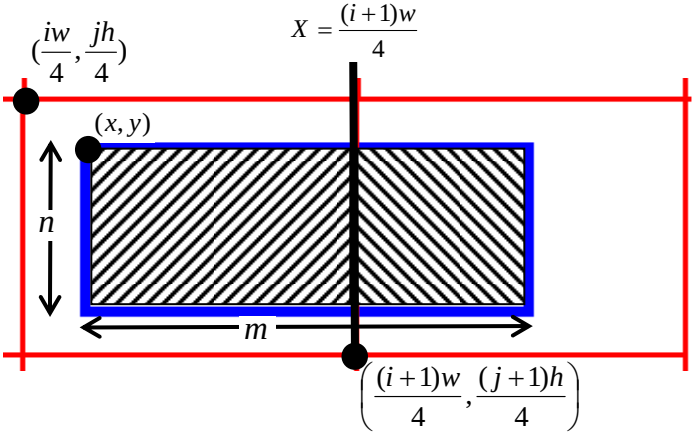


圖 17、橫跨 1 條直軸線的元件

➢ 若元件橫跨 2 條以上直軸線(k≧2)，如圖 18，就要計算 3 段面積：

(1) $X=\frac{(i+1)w}{4}$ 直軸線以左的 1 塊面積

(2) $X=\frac{(i+1)w}{4}$ 直軸線到 $X=\frac{(i+k)w}{4}$ 直軸線之間，共有(k-1)塊面積

(3) $X=\frac{(i+k)w}{4}$ 直軸線以右的 1 塊面積

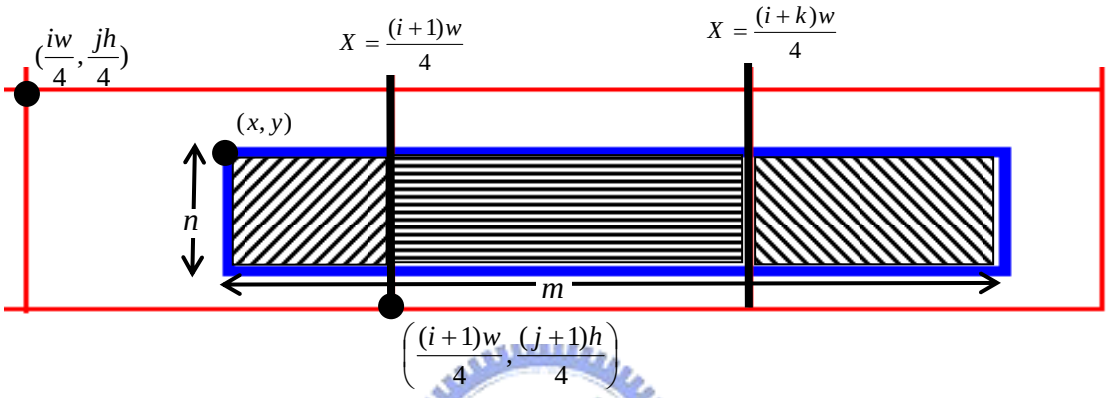


圖 18、橫跨 k 條直軸線的元件

➢ **公式推導：**

元件上邊線與直軸線相交於 $\left(\frac{(i+g)w}{4},y\right)$ 點，元件下邊線與直軸線相交於 $\left(\frac{(i+g)w}{4},(y+n)\right)$，$g\in[1,2,..,k]$，則：

元件落在位置區塊的面積權重(PW)

=(元件左段面積×所在權重)+(元件中段面積×所在權重)+(元件右段面積×所在權重)

$$=\left(\left(\frac{(i+1)w}{4}-x\right)\times n\times\rho(B(i,j))\right)+\left(\sum_{g=1}^{k-1}\left(\frac{w}{4}\times n\times\rho(B(i+g,j))\right)\right)+\left(\left((x+m)-\frac{(i+k)w}{4}\right)\times n\times\rho(B(i+k,j))\right)$$

◆**(情況三)元件落在同一行內：**

元件落在同一行，則可能會直跨好幾條橫軸線。假設元件直跨了 t 條橫軸線(t≧1)：

- 若元件只直跨 1 條橫軸線(t=1)，如圖 19，則只需要計算以 $Y=\frac{(j+1)h}{4}$ 為橫軸線所劃分的上下兩段面積即可。

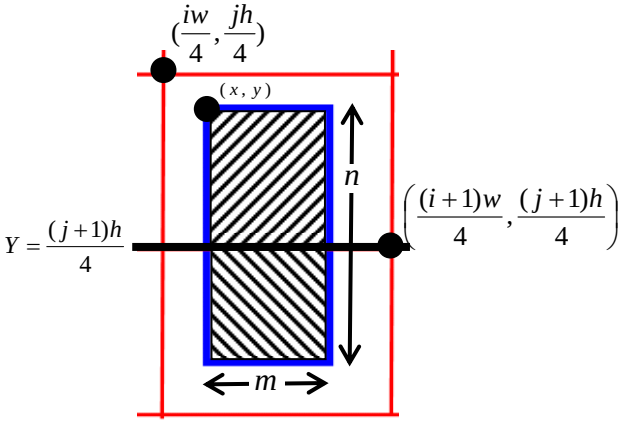


圖 19、直跨 1 條橫軸線的元件

- 若元件直跨 2 條以上橫軸線(t≧2)，如圖 20，就要計算 3 段面積：

$Y=\frac{(j+1)h}{4}$ 橫軸線以上的 1 塊面積

$Y=\frac{(j+1)h}{4}$ 橫軸線到 $Y=\frac{(j+t)h}{4}$ 橫軸線之間，共(t-1)塊面積

$Y=\frac{(j+t)h}{4}$ 橫軸線以下的 1 塊面積

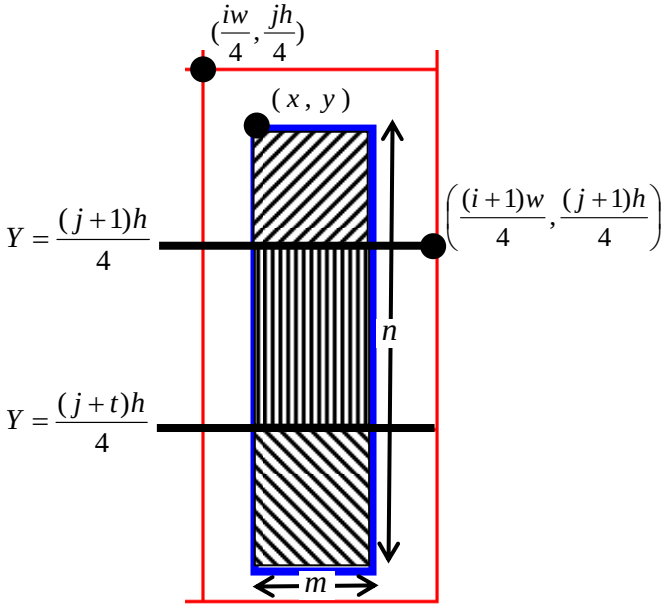


圖 20、直跨 t 條橫軸線的元件

➢ **公式推導：**

元件左邊線與橫軸線相交於 $\left(x,\frac{(j+q)h}{4}\right)$ 點，元件右邊線與橫軸線相交於 $\left((x+m),\frac{(j+q)h}{4}\right)$，$q\in[1,2,..,t]$，則：

元件落在位置區塊的面積權重(PW)

=(元件上段面積×所在權重)+(元件中段面積×所在權重)+(元件下段面積×所在權重)

$$=\left(\left(\frac{(j+1)h}{4}-y\right)\times m\times\rho(B(i,j))\right)+\left(\sum_{q=1}^{t-1}\left(\frac{h}{4}\times m\times\rho(B(i,j+q))\right)\right)+\left(\left((y+n)-\frac{(j+t)h}{4}\right)\times m\times\rho(B(i,j+t))\right)$$

◆(情況四)元件落在多行及多列之內：

元件落在多行及多列之內，則可能會同時跨過好幾條軸線。假設元件橫跨了 k 條直軸線，同時直跨了 t 條橫軸線(k,t≧1)：

➢ 若元件只橫跨 1 條直軸線與 1 條橫軸線，如圖 21，則只需要計算以 $X=\frac{(i+1)w}{4}$，$Y=\frac{(j+1)h}{4}$ 兩條軸線所劃分的四塊面積即可。

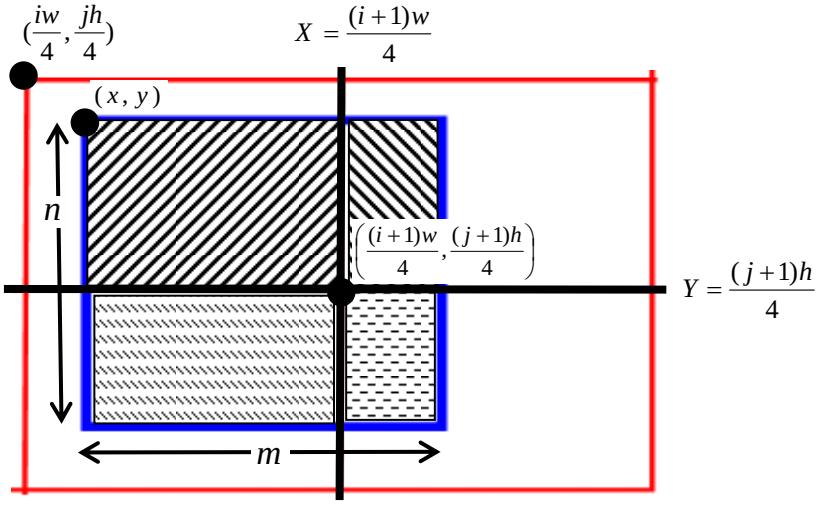


圖 21、分別跨過 1 條直軸線及橫軸線的元件

➢ 若元件橫的、直的皆跨過 2 條以上，就要計算九塊面積，如圖 22 所示：

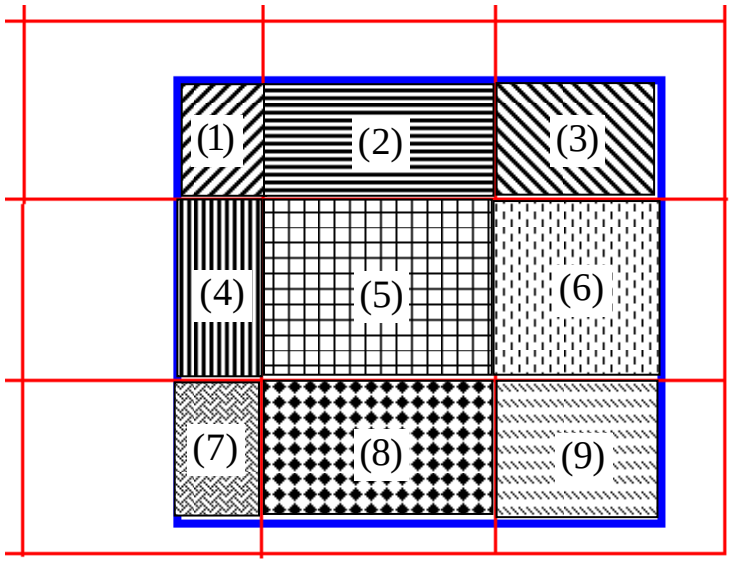


圖 22、跨過多條直軸線及橫軸線的九塊面積

(1) $X=\frac{(i+1)w}{4}$ 之左，$Y=\frac{(j+1)h}{4}$ 之上，共 1 塊面積

(2) $X=\frac{(i+1)w}{4}$ 到 $\frac{(i+k)w}{4}$ 之間，$Y=\frac{(j+1)h}{4}$ 之上，共(k-1)塊面積

(3) $X=\frac{(i+k)w}{4}$ 之右，$Y=\frac{(j+1)h}{4}$ 之上，共 1 塊面積

(4) $X=\frac{(i+1)w}{4}$ 之左，$Y=\frac{(j+1)h}{4}$ 到 $\frac{(j+t)h}{4}$ 之間，共(t-1)塊面積

(5) $X=\frac{(i+1)w}{4}$ 到 $\frac{(i+k)w}{4}$ 之間，$Y=\frac{(j+1)h}{4}$ 到 $\frac{(j+t)h}{4}$ 之間，共(k-1)(t-1)塊面積

(6) $X=\frac{(i+k)w}{4}$ 之右，$Y=\frac{(j+1)h}{4}$ 到 $\frac{(j+t)h}{4}$ 之間，共(t-1)塊面積

(7) $X=\frac{(i+1)w}{4}$ 之左，$Y=\frac{(j+t)h}{4}$ 之下，共 1 塊面積

(8) $X=\frac{(i+1)w}{4}$ 到 $\frac{(i+k)w}{4}$ 之間，$Y=\frac{(j+t)h}{4}$ 之下，共(k-1)塊面積

(9) $X=\frac{(i+k)w}{4}$ 之右，$Y=\frac{(j+t)h}{4}$ 之下，共 1 塊面積

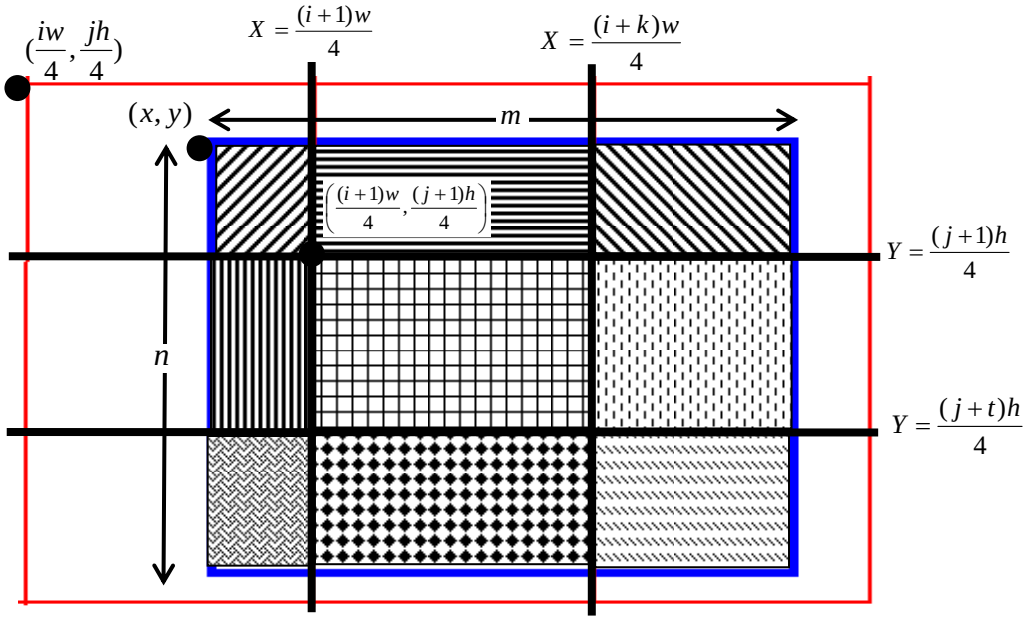


圖 23、分別跨過多條直軸線及橫軸線的元件

➢ **公式推導：**

如圖 23，元件上邊線與直軸線相交於$\left(\frac{(i+g)w}{4},y\right)$點，元件下邊線與直軸線相交於$\left(\frac{(i+g)w}{4},(y+n)\right)$，$g\in[1,2,..,k]$，元件左邊線與橫軸線相交於$\left(x,\frac{(j+q)h}{4}\right)$點，元件右邊線與橫軸線相交於$\left((x+m),\frac{(j+q)h}{4}\right)$，$q\in[1,2,..,t]$，則：

元件落在位置區塊的面積權重(PW)

=(元件左上面積×所在權重)

+(元件中上面積×所在權重)

+(元件右上面積×所在權重)

+(元件左中面積×所在權重)

+(元件中間面積×所在權重)

+(元件右中面積×所在權重)

+(元件左下面積×所在權重)

+(元件中下面積×所在權重)

+(元件右下面積×所在權重)

$$
\begin{aligned}
=&\left(\left(\frac{(i+1)w}{4}-x\right)\times\left(\frac{(j+1)h}{4}-y\right)\times\rho(B(i,j))\right)\\
&+\left(\sum_{g=1}^{k-1}\left(\frac{w}{4}\times\left(\frac{(j+1)h}{4}-y\right)\times\rho(B(i+g,j))\right)\right)\\
&+\left(\left((x+m)-\frac{(i+k)w}{4}\right)\times\left(\frac{(j+1)h}{4}-y\right)\times\rho(B(i+k,j))\right)\\
&+\left(\sum_{q=1}^{t-1}\left(\frac{h}{4}\times\left(\frac{(i+1)w}{4}-x\right)\times\rho(B(i,j+q))\right)\right)\\
&+\left(\sum_{q=1}^{t-1}\left(\sum_{g=1}^{k-1}\left(\frac{w}{4}\times\frac{h}{4}\times\rho(B(i+g,j+q))\right)\right)\right)\\
&+\left(\sum_{q=1}^{t-1}\left(\frac{h}{4}\times\left((x+m)-\frac{(i+k)w}{4}\right)\times\rho(B(i+k,j+q))\right)\right)\\
&+\left(\left(\frac{(i+1)w}{4}-x\right)\times\left((y+n)-\frac{(j+t)h}{4}\right)\times\rho(B(i,j+t))\right)\\
&+\left(\sum_{g=1}^{k-1}\left(\frac{w}{4}\times\left((y+n)-\frac{(j+t)h}{4}\right)\times\rho(B(i+g,j+t))\right)\right)\\
&+\left(\left((x+m)-\frac{(i+k)w}{4}\right)\times\left((y+n)-\frac{(j+t)h}{4}\right)\times\rho(B(i+k,j+t))\right)
\end{aligned}
$$

- **區塊元件權重(Component Weight, CW)**

學者 George [5]研究指出，讀者在閱讀網頁時，若遇到有加底線、粗體的文字，因為具有明顯區隔作用，因此讀者的注意力會被吸引過去。網頁中的水平分隔線，也同樣具有區隔效果，能暫時阻斷讀著繼續閱讀後面的相關內容，因而把注意力集中過去。此外，粗體框線、表格、不同底色等效果，也具有區隔作用。

這些引人注目的區域，又稱為「熱點區域」，例如聳動的標題、大張的圖片、特殊文字等，能讓讀者一進入網頁，目光就馬上被吸引過去。一張 210×230 像素大小的圖片，就能獲得超過一半受試者的注意，且圖片愈大愈能引起注意[5]。

廣告能否吸引到目光，取決於它的位置及大小。研究發現置於首頁上方和左邊的廣告，最能引人注目，而右邊及下方的廣告則較不引人注意[5]。

依據[5]，本論文在 W3C 的 HTML 規格書中定義的三種 HTML DOM 節點，以及本論文擴充兩類 DOM 節點中，把具有熱點區域效果的區塊元件，給予不同 DOM 節點值 (node value)。

其中在計算面積或元件數目的比例時，會先取自然對數 ln，理由是因為雖然算出來的比例越大其權重就越高，但不應取線性正比，故取自然對數可以得到較為合理的結果。

(1) Insignificant Node

屬於 Insignificant Node 的 HTML 標籤，如表 4 所列，是瀏覽器上無法顯示出來的節點。其主要功能是用在網頁的標頭宣告、空白換行、引述文字、註解文字、或做為其他特定功能的隱藏控制項。這些標籤原則上都不會影響到網頁的呈現，也與網頁上的內容無關。因此 node value 值皆設為 1。

表 4、Insignificant Node 節點值

作用	HTML 標籤	給定依據	node value
宣告文件	<!DOCTYPE>、<HTML>、<HEAD>、<BODY>、<BGSOUND>、<META>、<STYLE>、<SCRIPT>	與內容無關	1
空白或換段換行符號	、空白內容、 、<P>、<WBR>、<NOBR>		1
引述文字、段落標籤	<BLOCKQUOTE>		1
註解文字	<!-- -->		1
隱藏的控制項	<INPUT type=hidden>		1

(2) Inline Node

屬於 Inline Node 的 HTML 標籤，如表 5 所示，是會影響網頁上文字顯示外觀的節點，換句話說，若在一段相同的文字，套上不同的 Inline Node，就會產生不一樣的效果。

如圖 24 所示，在文字上套用<H1>的效果，該段文字就會放大並以粗體顯示；而套用<H6>的效果，則是會縮小顯示。

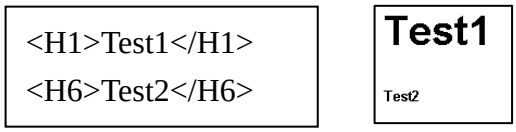


圖 24、套用不同 Inline Node 的效果

因此 Inline Node 權重比例的算法，是根據顯示出來的字型樣式、以及字型大小，用 $B(text)\times size(text)$ 表示之。若實際顯示出來是粗體、斜體、刪除線體、上標字、下標字等有變化的字型樣式，則 B 設為 2，若顯示出來沒有特殊變化，則 B 設為 1。size 則是實際在網頁上看到的字型大小，若無設定的話預設值是 3。除了依照<FONT>的 size 屬性來判斷外，其他標籤也換算成相對應的 font size 大小。例如<H2>TEXT</H2>的效果，就等於<FONT size=5> <B> TEXT </B> </FONT>。

表 5、Inline Node 節點值

作用	**HTML 標籤**	**給定依據**	**Node value=** $B(text)\times size(text)$
設定字體字型、顏色、大小之標籤	<FONT>	字體愈大，權重愈高	B=1 size 依照其<FONT>標籤內 size 屬性而定，預設是 3
設定標題文字之標籤	<H1> 、 <H2> 、 <H3> 、 <H4> 、 <H5>、<H6>	<H1>至<H6>權重由大至小	B=2 <Hn>的 size 為：7-n
置中對齊之標籤	<CENTER>	設定的內容愈多，權重愈大	B=2，size=3
粗體文字標籤	<B>、<STRONG>		B=2，size=3
斜體文字標籤	<ADDRESS> 、 <CITE>、<DFN> 、<EM>、<I>		B=2，size=3

加線文字標籤	<S>、<STRIKE>、<U>		B=2，size=3
固定字寬字體標籤	<CODE>、<KBD>、<LISTING>、<SAMP>、<TT>、<VAR>、<XMP>		B=1，size=3
放大與縮小字型	<SMALL>、<BIG>		B=1，<SMALL>的 size=2 <BIG>的 size=4
上標字與下標字	<SUP>、<SUB>、<ACRONYM>、<ABBR>、<DEL>、<INS>、<Q>		B=2，size=2

(3) Visible Node

屬於 Visible Node 的 HTML 標籤，是可以實際在瀏覽器上看到，且其下沒有包含子節點。如表 6 所示，Visible Node 包括文字、超連結、圖片、分隔線等。Node value 的算法，就依照該 Visible Node 與整個網頁所佔的比重。例如 Text node 就是計算該段文字與整個網頁的文字之比例。

值得注意的是，超連結的計算方式有考慮內部連結所佔的比例。若網頁中的超連結是連到同一部 Web Server 或同一個 domain name，就屬於內部連結，可能是選單目錄之類；反之若是連到外部的 Web Server 或 domain name，就屬於外部連結，可能是廣告或相關網站。因此內部連結愈多，則權重值就愈高。

水平分隔線(Separate node)本身就具有分隔效果，但它的資訊量並不多，因此 Node value 設為 0.5。

表 6、Visible Node 節點值

作用	HTML 標籤	給定依據	Node value
Text node	#TEXT	字數愈多，權重愈大	$\frac{\ln(TextCount)}{\ln(WebPageTextCount)}$ 其中 TextCount 為節點字數， WebPageTextCount 為網頁所有字數
Url node	<A>	字數愈多，權重愈大；內部連結權重大於外部連結	$URLTextCount \times urlTarget$ URLTextCount 為超連結字數 若為內部連結，則 urlTarget=2 若為外部連結，則 urlTarget=1
Image node	<IMG>、<MAP>、<AREA>、<MARQUEE>、<EMBED>	長、寬愈大，權重愈大	$\frac{\ln(width \times height)}{\ln(PageWidth \times PageHeight)}$ width 與 height 為圖片的寬與高 PageWidth 與 PageHeight 為網頁的寬與高
Separate node	<HR>	比相鄰節點大	0.5

(4) Format Node

屬於 Format Node 的 HTML 標籤，其下包含多個子節點，並用特定格式在網頁上呈現，如表格、表單等，如表 7 所示。因此 Node value 計算方式，是考慮該節點包含了哪些元件，再計算該節點在網頁上所佔的面積或數目比例。

節點所包含的元件，可分成文字、圖片及互動元件(如文字方框、按鈕...等)三種類別，[31]定義了元件組成內容的權重判斷表，其權重可參照表 8 進行設定。

表 7、Format Node 節點值

作用	HTML 標籤	給定權重依據	Node value
Area format node	<TABLE>、<TBODY>、<TR>、<TD>、<TH>、<CAPTION>、<DIV>、<SPAN>、<IFRAME>	長寬愈大，包含子節點愈多，權重愈大	$\theta \times \frac{\ln(nodeArea)}{\ln(WebPageArea)}$ nodeArea 為元件所佔面積 WebPageArea 為網頁面積
Combined format node	<FORM>、<FIELDSET>、<EGEND>、<INPUT>、<TEXTAREA>、<SELECT>、<OPTION>、<LABEL>	設定的內容愈多，權重愈大	
List format node	<OL>、<LI>、<UL>、<DL>、<DT>、<DD>	設定的內容愈多，權重愈大	$\theta \times \frac{\ln(listCount)}{\ln(AllComponentCount)}$ listCount 為 List format node 數目 AllComponentCount 為網頁所有元件數目

表 8、區塊組成內容權重的判斷表

區塊組成內容	權重 θ	說明
文字	0.75	應是內文
圖片	0.5	只有圖片
文字+圖片	1	內文
只有互動元件	0	沒有內容
文字+互動元件	0.1	可能為登入或搜尋區塊
圖+互動元件	0.1	可能為登入或搜尋區塊
文字+圖+互動元件	0.25	可能為登入或搜尋區塊，但考慮其多樣性，權重稍高

算出 Node Value 之後，就可進一步計算區塊元件權重(Component Weight, CW)，其算法是本身節點的 Node value 與所有兄弟節點的 Node value 總和之比率：

$$CW(n_i) = \frac{NodeValue(n_i)}{\sum_{j=1}^{ns_i} NodeValue(n_j)}$$

■ **主題區塊資訊含量(Information Content, IC)**

若只單純計算資訊含量，則節點 i 的資訊含量就等於其下所有子節點的資訊含量總和，乘上獨立節點比率之後，再加上節點 i 的區塊位置權重、節點 i 的區塊元件權重，最後再乘上區塊階層權重，公式如下所示：

$$IC(N) = \left((\sum_{k=1}^{m} IC(child(k) \times \frac{n}{m+1}) + PW(N) + CW(N) \right) \times \varepsilon$$

$$\varepsilon = level \times \frac{1}{NumberOfLevel}$$

其中 IC(N)為節點 N 的資訊含量；IC(child(k))為節點 N 底下第 k 個子節點的資訊含量；m 為節點 N 底下所有子節點的個數；n 為節點 N 底下選擇要獨立的子節點個數，其中 n≦m。

要考慮獨立節點權重($\frac{n}{m+1}$)，是因為如果節點 N 底下要獨立的子節點愈多，就代表 N 節點底下擁有愈多重要的子節點，因此節點 N 的資訊含量就相對變大。分母 m+1 的用意是因為樹葉節點的子節點數目為 0，要避免分母為 0 的情況。

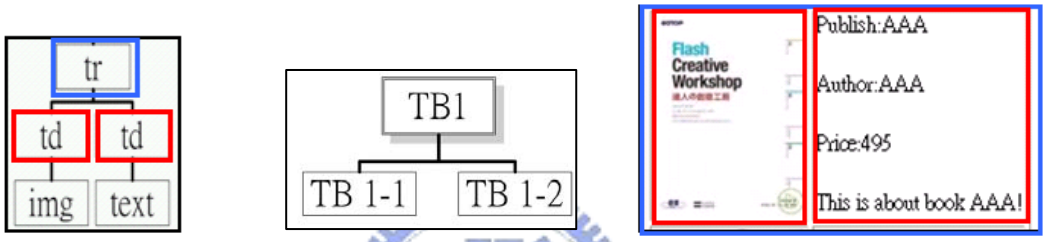
資訊含量的計算方法，是考慮其下所有子節點資訊含量的和以及要獨立子節點比率，再乘上自己的區塊位置權重、區塊元件權重。為了不讓根節點的資訊含量愈算愈大，所以要計算區塊所在的階層權重 ε。level 為該節點所在的階層，NumberOfLevel 是整棵樹的層級數。例如一棵樹有五層，則第二層的階層權重為：$2 \times \frac{1}{5} = 0.4$

■ **用賽局進行決策**

當任一節點算出資訊含量後，就要進行決策：

- 若資訊含量大於或等於門檻值，就選擇獨立成為一個主題區塊
- 若資訊含量小於門檻值，就選擇不獨立。

以圖 25 為例，圖 25(a)的 TR 經過計算資訊含量後，決定要獨立成為主題區塊(TB1)；而 TR 下面的兩個 TD 子節點(以 TDs 表示)經過計算資訊含量後，也決定要獨立成為主題區塊(TB1-1 與 TB1-2)，如圖 25(b)。那麼就會形成圖 25(c)的情況，一個區塊內包含了兩個子區塊，而這兩個子區塊的內容各是一張圖片及一段文字。



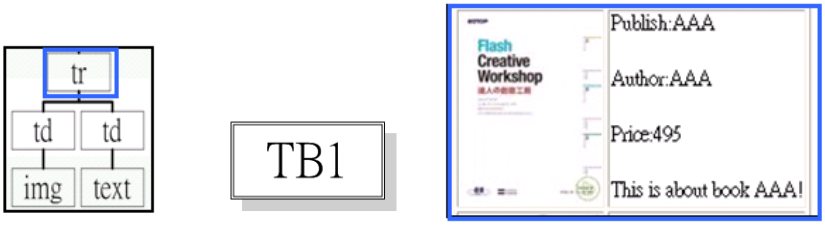
(a) (b) (c)

圖 25、判斷要獨立成為主題區塊

但這種結果並非最好，因為它使用不必要的區塊來包含不夠豐富的子區塊。由表 9 可以看出，主題區塊 TB1 裡面就包含了較完整的一段 HTML 碼，而 TB1-1 與 TB1-2 包含著不完整的 HTML 碼。較好的情況應該是如圖 26 所示，TR 獨立成為主題區塊 TB1，而 TDs 不獨立，那麼 TR 獨立之後的區塊，就包含了底下 TDs 的內容，代表 TR 合併了 TDs。如此一來，只用了一個區塊，且這個區塊的內容就夠豐富，相當於一筆資料列。

表 9、主題區塊的豐富程度

Themeal block(TB)	HTML code
TB1	<tr> <td><img …></td> <td>#TEXT</td> </tr>
TB1-1	<td><img …></td>
TB1-2	<td>#TEXT</td>



(a) (b) (c)

圖 26、較理想的判斷結果

由此可知，若每個節點計算資訊含量後都決定要獨立成主題區塊，最後產生的主題區塊樹，就和原來的 DOM Tree 差不多，也就失去切割主題區塊的意義了。

因此，所有非樹葉節點在決策時，不能只考慮到自己，應該也要一併考慮到其下所有子節點所做之決策對它本身的影響。以圖 12 為例，當節點<td>要判斷時，並不能直接決定自己能不能獨立成為主題區塊，而是必須先看看其下子節點所做的決策。圖 27 中，節點<td>下面的四個子節點，有三個決定要獨立成為主題區塊(table、text、table)，一個決定不獨立(hr)。

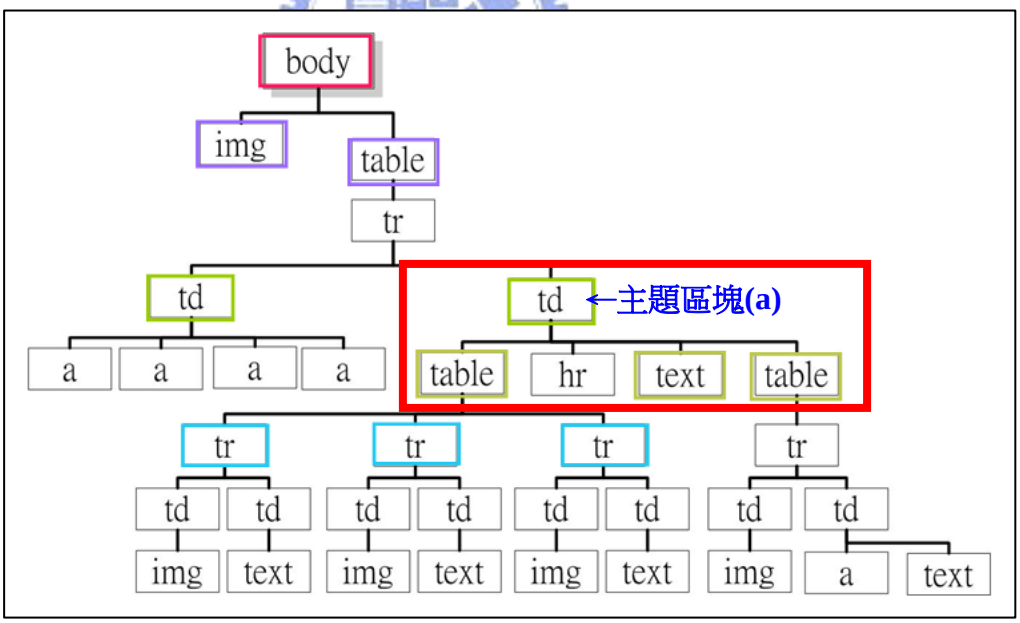


圖 27、主題區塊之判斷考量

賽局理論便可以用來處理這個問題。賽局是一種研究方法，意指兩個或兩個以上理性的玩家(player)，為了追求自己目標最大值，而與對方形成互相衝突的一種對抗狀態，對抗狀態最後到達穩定的結局，就稱為「均衡」，此時對雙方皆有利，達到整體共同報酬最大化[11]。

本論文就是引入這樣的概念，將讓每個區塊在決策時，與它下面所有的子節點進行賽局(Game)，在計算出各別的報酬之後，找出均衡點，該點就是對雙方皆有利的選擇。

賽局(Game)表示式

賽局最基本的三個元素，就是玩家(Player)、決策(Action)、以及因決策所得到的報酬(Utility)。在每個 Game G 中，有兩個 Player，一個是玩家 i，另一個則是其下的所有子節點的集合，我們用玩家-i 來表示。每個 Player 皆有兩個 Action，選擇獨立，或者選擇不獨立。選擇 Action 的 Utility，以 U(i, -i)來表示。表 10 列出了賽局理論與主題區塊擷取之間的關係，整個 Game 可以用下列式來表示：

$$G = \langle N, A, U \rangle$$

其中 G 代表整個 Game，N 代表 Player 的集合，A 代表 Action 的集合，U 代表 Utility 的集合。

表 10、Game Theory 與主題區塊擷取的關係

Game Theory	主題區塊擷取
Player	DOM Tree 中的節點
Action	獨立　或　不獨立
Utility	選擇 Action 所得到的報酬

賽局玩家(Player)定義

在本論文中，一個賽局有兩個 Player，一個是玩家 i，另一個則是玩家 i 之下的所有子節點的集合，以-i 來表示。

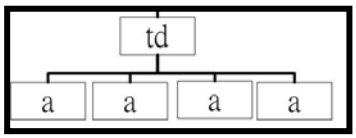


圖 28、Player i 與-i

以圖 28 為例，兩個 Player 分別為：

Player i = {<td>}
Player -i = {<a>, <a>, <a>, <a>}

在真正的賽局理論中，每場賽局是兩個對等的玩家，彼此要去猜測對方可能的想法，再依此做出對自己最有利的選擇。而在本論文的定義中，所謂的玩家是指節點與其下所有子節點集合，兩者為了找出一個共同報酬最大化的均衡點，所進行的賽局。

因此，本論文的玩家帶有一點合作的意味，且能知道對方得到的報酬。這一點是跟傳統賽局不同的地方。

賽局策略(Action)定義

每個 Player 皆有兩種 Action，選擇要獨立還是不獨立：

$$a = \{獨立，不獨立\}$$

值得注意的是，-i 集合裡的所有節點皆選擇不獨立，則-i 的決策才為『不獨立』；若-i 集合裡有一個以上節點選擇獨立，則-i 的決策就為『獨立』。兩個 Player，每個 Player 有兩種 Action，因此會有以下四種可能：

- **i 選擇獨立，−i 選擇獨立**

代表玩家 i 的資訊含量夠高，但其下所有子節點的資訊含量也夠高，玩家 i 與玩家-i 將各自獨立成為主題區塊，也就是一個區塊包含了多個不夠豐富的區塊。

- **i 選擇獨立，−i 選擇不獨立**

代表玩家 i 的資訊含量夠高，而且下所有子節點的資訊含量過低，這種情況就由 i 獨立，合併掉下面子節點，形成一個夠豐富的區塊。

- **i 選擇不獨立，−i 選擇獨立**

這種情況代表 i 覺得它的資訊含量不夠，不足以獨立成為主題區塊，因此選擇不獨立。

- **i 選擇不獨立，−i 選擇不獨立**

這種情況代表節點 i 及其所有的子節點，資訊含量皆過低，沒有什麼有意義的內容，因此皆不需要獨立成為主題區塊。

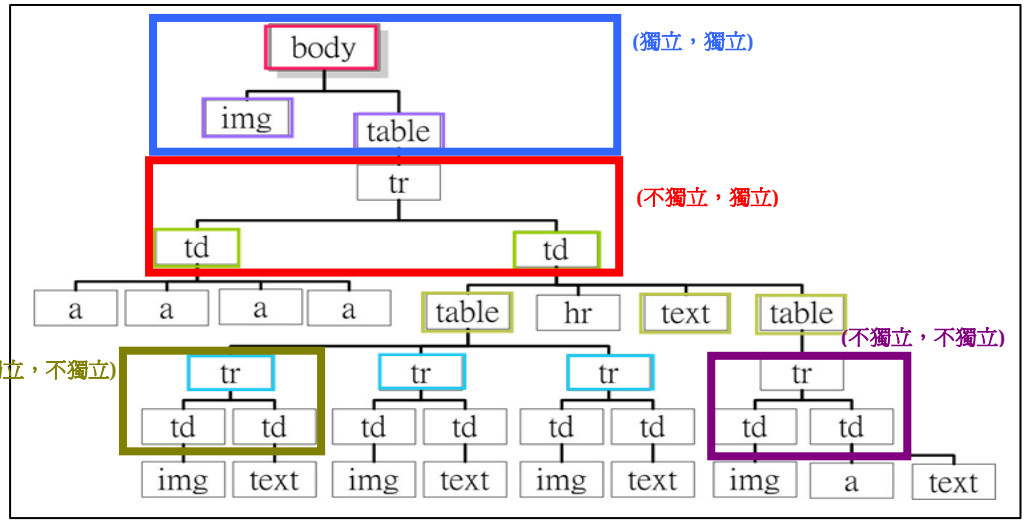


圖 29、Player i 與-i 選擇策略的四種情況

圖 29 說明了節點 i 與-i 所做的策略，可以組合出四種情況。由此我們可以發現節點 i 在計算資訊含量時，會受到其下想要獨立的子節點個數所影響，若底下完全沒有子節點要獨立，則節點 i 的權利就愈大，可以併下所有的子節點；反之若有愈多子節點想要獨立，節點 i 就不見得能夠併下所有子節點了，除非它自己本身的資訊含量夠高。

賽局報酬(Utility)函式

依照玩家選擇的策略，會對應到不同的報酬函式算法。以下將用一個例子，來說明玩家選擇不同策略的報酬計算方式。以圖 30 為例，假設門檻值 t=0.5，玩家-i 集合裡面有三個節點(table、text、table)計算出資訊含量後，大於門檻值 t，因此決定獨立，如圖 30 中有粗框實線的節點；而玩家 i(即節點 td)的資訊含量節就等於其下所有子節點的資訊含量總和，加上獨立節點比例、區塊位置權重、區塊元件權重，再乘上階層權重。

假設 PW(i)=0.24，CW(i)=0.25，ε=0.41，則：

$$IC(i)=\left(\left((0.6+0.3+0.52+0.6)\times\frac{3}{4+1}\right)+0.24+0.25\right)\times 0.41=0.7$$

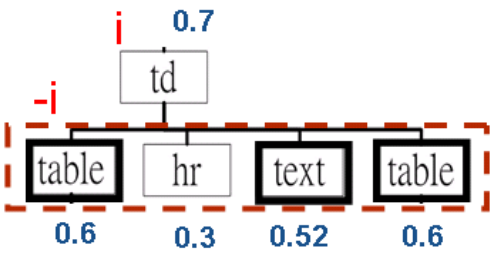


圖 30、玩家 i 與玩家-i 的資訊含量

雖然玩家 i 算出來的資訊含量高於門檻值，但尚不能直接就決定要獨立，而是必須先與玩家-i 進行一場賽局。假設 $F_Y()$ 為選擇獨立的報酬函式，$F_N()$ 為選擇不獨立的報酬函式，則：

- **情況一，i 選擇獨立，-i 選擇獨立**

玩家 i 選擇獨立的報酬，就是計算它自己本身的資訊含量，加上底下所有選擇不獨立的子節點資訊含量平均，即 $F_Y(i) = IC(i) + \frac{\sum_{k=1}^{n} IC(-i[k])}{n}$，其中 n 為-i 集合裡面選擇不獨立的子節點個數；而玩家-i 選擇獨立的報酬，就是計算自己本身資訊含量的平均值，即 $F_Y(-i) = \frac{\sum_{k=1}^{m} IC(-i[k])}{m}$，其中 m 為-i 集合裡面選擇獨立的子節點個數。

- **情況二，i 選擇獨立，-i 選擇不獨立**

玩家 i 選擇獨立的報酬，就是計算玩家 i 合併玩家-i 後的資訊含量，即 $F_Y(i) = IC(i) + \frac{\sum_{k=1}^{n} IC(-i[k])}{n}$；而玩家-i 已經被並入玩家 i，因此選擇不獨立的報酬就等於玩家 i 的資訊含量，即 $F_N(-i) = IC(i)$。

- **情況三，i 選擇不獨立，-i 選擇獨立**

玩家 i 選擇不獨立的報酬為它自己本身的資訊含量，即 $F_N(i) = IC(i)$；玩家-i 選擇獨立的報酬，就是計算自己本身資訊含量的平均值，即 $F_Y(-i) = \frac{\sum_{k=1}^{m} IC(-i[k])}{m}$。

- **情況四，i 選擇不獨立，-i 選擇不獨立**

玩家 i 選擇不獨立的報酬為 $F_N(i) = IC(i)$；而玩家-i 選擇不獨立的酬報為自己本身資訊含量的平均值 $F_N(-i) = \frac{\sum_{k=1}^{n} IC(-i[k])}{n}$。

在圖 30 的例子中，四個情況各自的報酬計算如表 11 所示：

表 11、依圖 30 為範例的報酬計算結果

情況一	i 獨立	$F_Y(i) = 0.7 + \frac{0.3}{1} = 1$
	-i 獨立	$F_Y(-i) = \frac{0.6 + 0.52 + 0.6}{3} = 0.57$
情況二	i 獨立	$F_Y(i) = 0.7 + \frac{0.6 + 0.3 + 0.52 + 0.6}{4} = 1.2$
	-i 不獨立	$F_N(-i) = 0.7$
情況三	i 不獨立	$F_N(i) = 0.7$
	-i 獨立	$F_Y(-i) = \frac{0.6 + 0.52 + 0.6}{3} 0.57$
情況四	i 不獨立	$F_N(i) = 0.7$
	-i 不獨立	$F_N(-i) = \frac{0.6 + 0.3 + 0.52 + 0.6}{4} = 0.5$

■ **挑出均衡點**

逐一計算出這四種情況的報酬之後，就可以利用報酬矩陣來找出納許均衡(Nash Equilibrium)，也就是對雙方而言皆有利的選擇情況。

(1)對玩家 i 而言

- 在玩家-i 選擇獨立的情況下，玩家 i 獨立的報酬是 1，不獨立的報酬是 0.7，因此玩家 i 會選擇獨立。
- 在 TDs 選擇不獨立的情況下，玩家 i 獨立的報酬是 1.2，不獨立的報酬是 0.7，因此玩家 i 會選擇獨立。

(2)對玩家-i 而言

- 在玩家 i 選擇獨立的情況下，玩家-i 獨立的報酬是 0.57，不獨立的報酬是 0.7，因此玩家-i 會選擇不獨立。
- 在玩家 i 選擇不獨立的情況下，玩家-i 獨立的報酬是 0.57，不獨立的報酬是 0.5，因此玩家-i 會選擇獨立。

		玩家-i	
		獨立	不獨立
玩家 i	獨立	(1 , 0.57)	(1.2 , 0.7)
	不獨立	(0.7 , 0.57)	(0.7 , 0.5)

圖 31、玩家 i 與玩家-i 的納許均衡

如圖 31 所示，最後雙方決策的交集點，就是納許均衡(玩家 i 獨立，玩家-i 不獨立)。因此，這個賽局結束之後，節點 td 決定要獨立，並合併其下所有子節點，成為一個主題區塊，這種情況下節點 td 的報酬是最大的；而原本打算獨立的子節點集合(table、text、table)，經過賽局的協調之後，發現選擇不獨立，被 td 合併之後的報酬反而較大，因此改變為不獨立，成為 td 的一份子，如圖 32 所示。

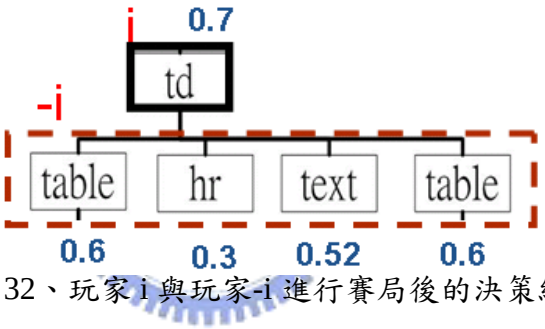


圖 32、玩家 i 與玩家-i 進行賽局後的決策結果

正規賽局的報酬矩陣，可能會出現找不到均衡點的情況。如圖 33 的賽局中，並無法找到交集的均衡點。在這種情況下，就會進一步計算雙方報酬的平均值，挑選出具有最大值的選擇。

		玩家-i	
		獨立	不獨立
玩家 i	獨立	(1 , 0.5)	(0.7 , 0.57)
	不獨立	(0.7 , 0.57)	(1.2 , 0.5)

圖 33、玩家 i 與玩家-i 無均衡點

如圖 34，四種情況的報酬平均值之中，最大值是(不獨立,不獨立)的 0.85，因此，這回合的賽局結果，就是玩家 i 不獨立，玩家-i 也不獨立。

		玩家-i	
		獨立	不獨立
玩家 i	獨立	0.75	0.635
	不獨立	0.635	0.85

圖 34、計算報酬平均最大值

若雙方報酬平均值出現兩個以上的最大值，代表玩家選擇獨立或不獨立的效果都差不多。如圖 35，這個賽局報酬矩陣出現三個最大值，在所有最大值情況中，由玩家 i 先選擇，並以『獨立』為第一優先，若無『獨立』的選擇，再考慮『不獨立』。接著由玩家-i 選擇，以『不獨立』為第一優先，若無『不獨立』選擇，再考慮『獨立』。

		玩家-i	
		獨立	不獨立
玩家 i	獨立	0.85	0.85
	不獨立	0.635	0.85

玩家 i 的選擇　　玩家-i 的選擇

圖 35、依優先選擇來決策

(2) 建立主題區塊樹(GRAB Step 2.2)

DOM Tree 中每個節點經由演算法步驟 2.1，計算資訊含量，以及進行賽局決策之後，會決定該節點是否可以獨立成為主題區塊。決定要獨立的節點，會先暫存起來，等到整棵 DOM Tree 跑完之後，再反向從根節點逐一加入子節點。

遞迴搜尋的方法，是採用以深度為優先的搜尋法（Depth-First-Search）。DFS 是搜索演算法的一種，是沿著樹的深度遍歷樹的節點，直到最後的根節點。圖 36 是以 DFS 跑過整棵 DOM Tree 的順序，節點上的數字即代表順序。決定要獨立的節點會先記錄其 ccmID 並暫存起來，等到整棵 DOM Tree 跑完，所有欲獨立的節點都挑選出來之後，將最後要獨立的節點做為樹根節點，反向建置出整棵主題區塊樹。

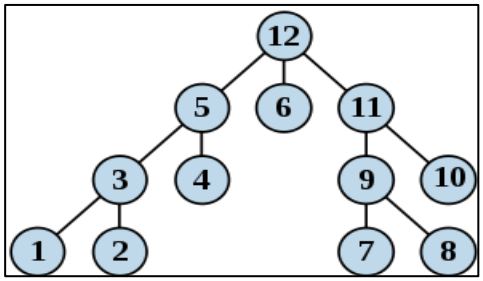
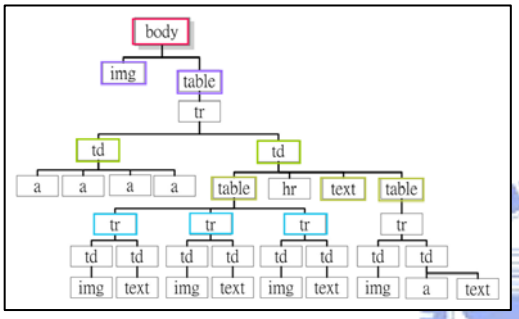
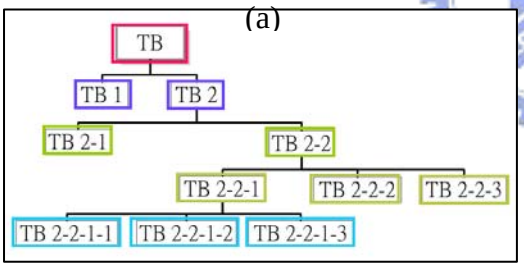


圖 36、以 DFS 進行搜尋的順序

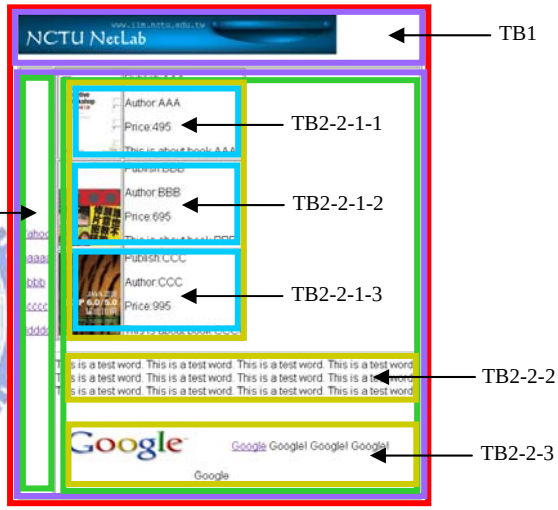
圖 37 說明 DOM Tree、Theme-based Tree、及原始網頁的對應關係。在 DOM Tree 裡面的<td>節點，就對應到 Theme-based Tree 的 TB 2-1 區塊，而每個主題區塊就等同於 DOM Tree 的一棵子樹，其內容就是一個網頁區塊，即一段完整的 HTML 語法。



(a)



(b)



(c)

圖 37、DOM tree 與 Theme-based Tree 及原始網頁之對應

3.2.3 GRAB 演算法步驟 3：精煉(refinement)

演算法的第三步驟，是調整上個步驟所產生的主題區塊樹。一個網頁之中，會包含著許多個主題區塊，因此把一個網頁切割成多個區塊後，會發現可能有某幾個區塊是相同主題，或內容是相似的，但是由於位置不相鄰，而被歸類為不同的主題區塊。

再者，由於每個使用者的觀點不同，可能會主觀的認定某幾個區塊本來就應該放在一起的，因此有必要讓使用者自行決定要取得的區塊範圍及內容。

針對這兩個問題，本論文利用 B.Liu[2]所提出的「一般化節點」(Generalized Node)的概念，把同主題或相似的區塊組成一個一般化節點，使得這些區塊雖然實際上是不相鄰，但邏輯上是在一起的。

合併主題區塊樹，除了能讓相似的主題區塊一起被呈現，同時也能簡化樹狀結構，當樹的深度愈淺，階層數愈少，則從樹根節點到葉根節點的處理效率就會提升。再者，在樹中的一些多餘區塊，也會一併在這個步驟解決掉，使得主題區塊樹更加乾淨俐落。

這個步驟會以 DFS 方式逐一判斷主題區塊，基於本論文定義的門檻值 H、L 來進行調整：

(1) 若資訊含量(IC) ≧ H

則代表該主題區塊已具有足夠的主題性，因此可以獨立成為主題區塊，不須合併或分割。

(2) 若資訊含量(IC) ≦ L

則代表該主題區塊的主題性仍不足夠，因此必須進行分割。此主題區塊會先標記起來，降低門檻值 T，經由第二個步驟再處理一次，產生出更細緻的主題區塊樹，再加入原本的主題區塊樹。

(3) 若主題區塊的資訊含量介於 L 跟 H 之間，L ＜ IC ＜ H

則須利用 TSTM(Theme-based STM)演算法，進一步做相似度比較。

TSTM 演算法

本論文提出一個 TSTM 的演算法，用來計算兩兩子樹的相似度，並考慮標籤相似度、內容相似度、以及資訊含量的多寡程度。若計算結果是相似的，兩棵子樹就形成一個 Generalized Block。TSTM 的演算法如下：

```
Input： two subtrees
Output： the node similarity of these two trees
Alogrithm： TSTM(T1, T2)
  If r1 and r2 (the roots of the two trees T1 and T2) contain distinct symbols OR have visual conflict then
    Return 0;
  Else
    m ：= the number of first-level sub-trees of T1;
    n ：= the number of first-level sub-trees of T2;
    Initialization： M[i,0] ：= 0 for i=0,…,m;
                    M[0.j] ：= 0 for j=0,…,n;
    For i=1 to m do
      For j=1 to n do
        M[i,j] ：= max(M[i,j-1], M[i-1,j], M[i-1,j-1]+W[i,j]),
          Where W[i,j] ：= TSTM(T1[i], T2[j])
      }
    }
Return (M[m,n] + AVG(IC(m), IC(n)) + content_similarity(r1, r2))

Procedure： content_similarity(r1, r2)
If r1 and r2 are not leaf nodes then
     Return 0;
Else
     cs ：= LCS(r1.data, r2.data);
     w ：= the number of words contained in cs;
     m ：= the maximal number of words contained in r1.data and r2.data;
return w/m;
```

TSTM 演算法是以 Z.Yanhong 和 L.Bing 提出的 STM 演算法以及 ESTM 演算法為基礎。STM[3]演算法是計算兩棵子樹的節點標籤(tag)相似度，若標籤相似則傳回 1。如圖 38，若以 P 為根節點，(A)跟(B)兩棵子樹的節點相似度為：

(根節點的相似度)+(底下所有子樹最大相似度的和) = 1+6 = 7

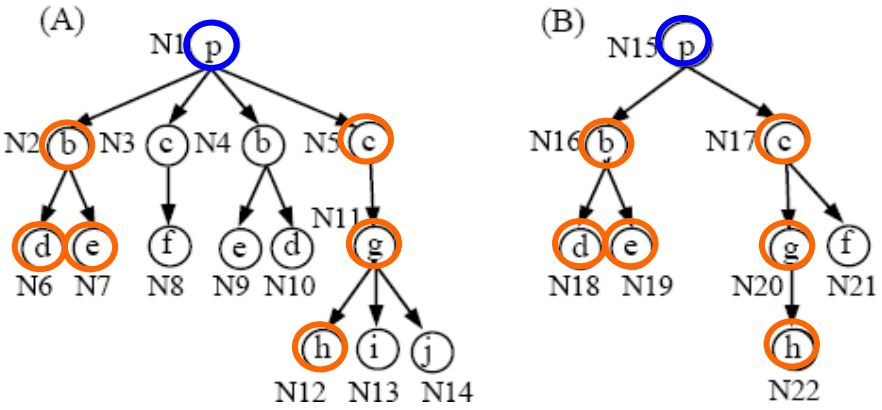


圖 38、STM 演算法

來源：Web Data Extraction Based on Partial Tree Alignment [3]

ESTM[4]是 Z.Yanhong 和 L.Bing 為了改良 STM 所提出的演算法。STM 只考慮標籤相似度，例如 A 樹的『TABLE』標籤跟 B 樹的『TABLE』標籤就視為相似，若標籤相似則傳回 1，並未考慮考其內容。ESTM 則是修改傳回值，加入 LCS(Longest Common Subsequence)演算法計算內容相似度。

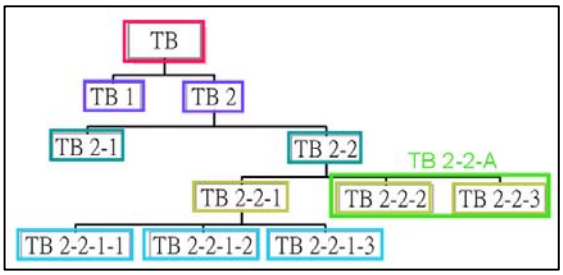
本論文提出的 TSTM 演算法，以 STM/ESTM 為基礎，並將之擴充，考慮節點資訊含量的多寡程度。即使兩棵子樹的標籤、內容都相同，但資訊含量的多寡卻可能不同，也會影響到它的相似程度。如圖 39，一個新聞網頁中，可能同時會有摘要新聞(a)跟新聞全文(b)，雖然它們的內容相似，但資訊含量卻不同，因此若是將(a)、(b)兩個主題區塊並在一起，對使用者而言反而是一種累贅。在這種情況下，本論文會將資訊含量相對較低的區塊併入其他主題區塊，例如將許多摘要新聞整合成一塊『摘要新聞』主題區塊。

(a)

(b)

圖 39、不同資訊含量的相似主題區塊

在圖 40 的例子中，假設主題區塊 TB2-2-2 與 TB2-2-3 是相似的，那麼就會進行主題區塊合併，如圖 40(a)所示，將 TB2-2-2 與 TB2-2-3 合併為 TB2-2-A 後，裡面的內容就包含了 TB2-2-2 與 TB2-2-3 的內容。



(a)

```
This is a test word. This is a test word. This is a test word. This is a test word. <br>
This is a test word. This is a test word. This is a test word. This is a test word. <br>
This is a test word. This is a test word. This is a test word. This is a test word.
<br><p>

 <table border=0>
   <tr>
     <td><img src="images/google.jpg" height=100 width=200>Google</a></td>

     <td><a href="http://www.google.com">Google</a></td>
     <td>Google! Google! Google!</td>
   </tr>
 </table>
```

(b)

(c)

圖 40、整併過的主題區塊樹對應到網頁

第四章　系統實作與分析

本論文針對 GRAB 演算法設計一個雛型系統，並進行一個系統實作與分析，實際呈現主題區塊資料擷取的實作範例。在這一個章節，首先分別介紹所有模組的架構、資料流程及功能，接著以 10 種主題共 30 個網頁做為資料集，分析並討論實作結果。

4.1 系統實作

本系統是採用 Microsoft Visual Studio 2005，使用 C#.NET 語言開發。透過 C#.NET 的 WebBrowser 物件，可以藉由 URL 或讀入 HTML 檔案，取得完整 HTML 碼，且可利用 WebBrowser 物件的方法或屬性，來對網頁加以處理。在 Tidy 方面，是透過使用 ATL 來建立 Tidy 的 COM 元件。系統主畫面如圖 41 所示

系統上方可以輸入網址，或從功能表列選擇讀取 HTML 檔案。功能操作包括：(1)轉成 DOM Tree，之後再轉成主題區塊樹，計算資訊含量、(2)開始進行擷取，將賽局決策後選擇要獨立的區塊，用紅色方框顯示在網頁上、(3)輸入 ccmID 取得主題區塊內容。

系統左上方是轉換出來的 DOM Tree，左下方則是轉換出來的主題區塊樹，並且將資訊含量標記上去。在主題區塊樹上按滑鼠右鍵，即可顯示出該主題區塊的 HTML 內容，如圖 42 所示。

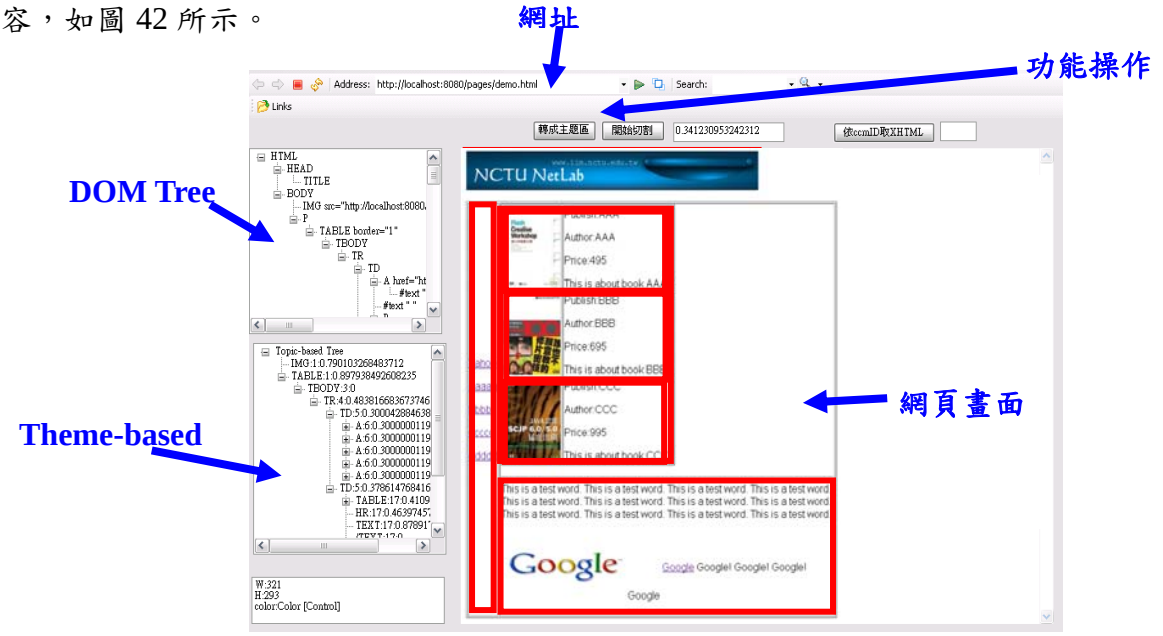


圖 41、系統主畫面功能

圖 42、擷取出來的 HTML 內容

本系統的架構圖如圖 43 所示，共有 6 個模組：DOM Tree 轉換模組、主題區塊樹轉換模組、資訊含量計算模組、賽局決策模組、主題區塊樹調整模組、主題區塊樹擷取模組。

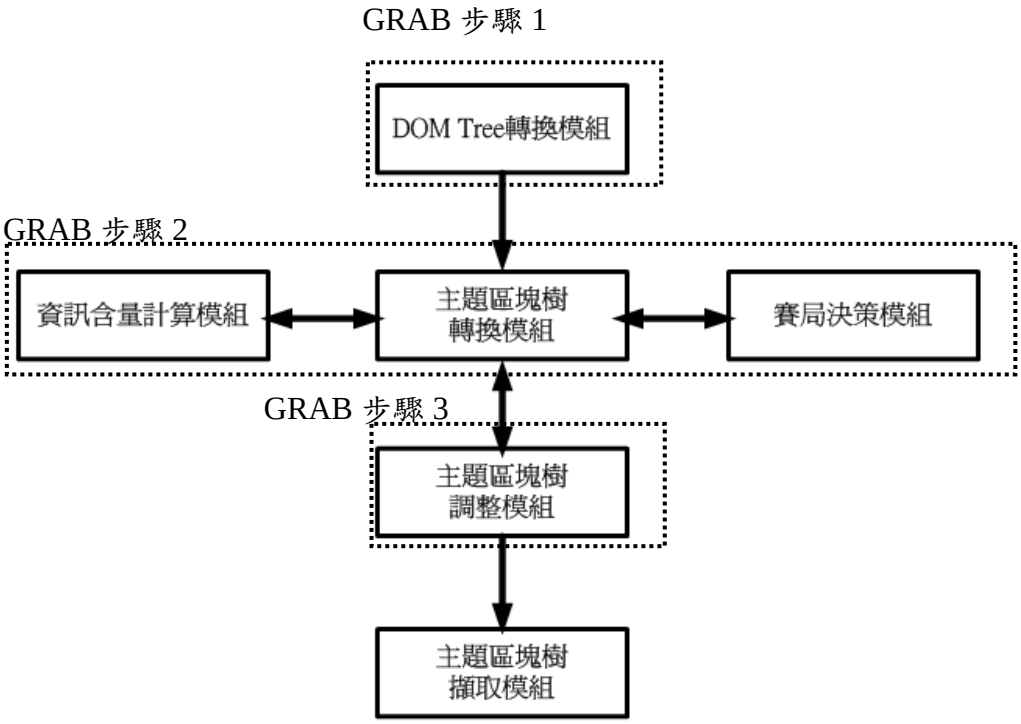


圖 43、系統架構圖

4.1.1 DOM Tree 轉換模組

此模組 Input 是一個 URL 或下載到主機的 HTML 檔案；Output 則是一棵 DOM Tree，實際上就是一份 XML 文件。

取得 URL 或 HTML 檔案之後，開始進行 GRAB 演算法步驟 1，首先會利用 Tidy 套件，將原始 HTML 轉換成符合 XML 格式的 XHTML，並修復遺漏的標籤，解決標籤冗贅或錯置的情形。接著，再從 BODY 標籤開始，替每一個標籤加入 ccmID 及 nodeType 屬性，重新輸出成一個 XML/XHTML 檔案。模組運作程式如圖 44 所示。

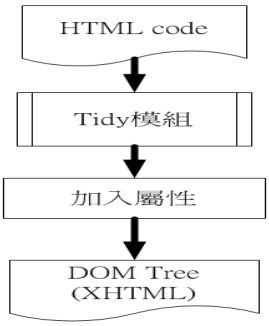


圖 44、DOM Tree 轉換模組流程

4.1.2 主題區塊樹轉換模組

此模組的 Input 是一份 XML/XHTML 文件，也就是一棵 DOM Tree；Output 則是一個主題區塊樹(Theme-based Tree)，每個節點的內容記載著要獨立成主題區塊的節點之 ccmID 值。

圖 45 描述了此模組的流程。取得 DOM Tree 的 XML/XHTML 內容之後，開始執行 GRAB 演算法步驟 2，以 DFS 巡覽方式，以由下而上的方式，逐一計算每個節點的資訊含量，接著進行賽局決策。計算資訊含量則是呼叫另一個『資訊含量計算模組』所傳回的結果，賽局決策則是呼叫『賽局決策模組』所傳回的結果。

整棵 DOM Tree 跑完一遍之後，所有決定要獨立的節點就反向建置成一棵主題區塊樹，樹中節點就記錄著 ccmID。最後輸出的主題區塊樹也是一份 XML 文件。

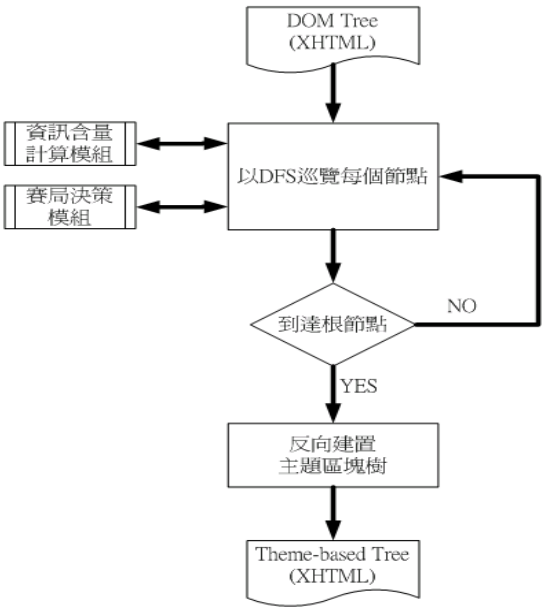


圖 45、主題區塊樹轉換模組流程

4.1.3 資訊含量計算模組

此模組的 Input 是一棵以節點 A 為樹根的一棵子樹，Output 則是節點 A 的資訊含量。

取得一棵子樹之後，會開始判斷樹根節點、以及判斷下面包含著哪些子節點，來計算資訊含量，最後傳回介於 0 到 1 之間的資訊含量結果。模組流程如圖 46 所示。

值得注意的是，在計算區塊位置權重(PW)值的時候，可能會遇到網頁不只一頁，需要用滑鼠下拉的情形。在此本論文就假設一進入網頁所看到的第一頁，是最重要的部份，其餘要往下拉才能瀏覽的部份，就較不重要。因此針對這種多頁網頁，本論文只處理進入網頁的第一頁，也就是不須用滑鼠滾動，就可以瀏覽的部份。

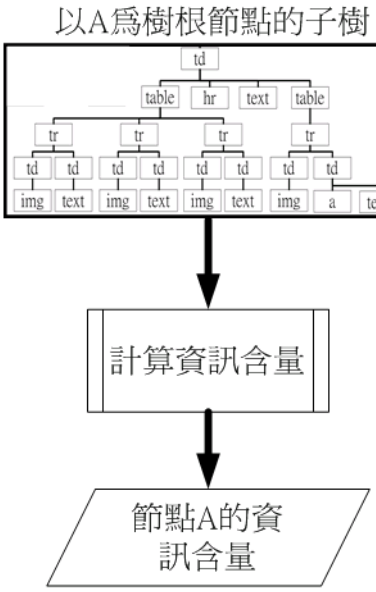


圖 46、資訊含量計算模組流程

4.1.4 賽局決策模組

此模組的 Input 是一棵標有資訊含量的子樹，即玩家 i 與玩家-i 兩層子樹。Output 則是一個均衡的結果，傳回自訂的物件，包含兩個值，例如(0.75 , 0.635)。

此模組流程如圖 47 所示，取得標有資訊含量的子樹後，開始分別計算四種情況下，每個玩家的報酬為何，並將報酬矩陣用二維陣列來表示，陣列中放的是自訂的物件。接著試著找尋均衡點。若找不到唯一均衡點，就找報酬平均最大值。若出現多個最大值，則進行最後的決策，玩家 i 以獨立為第一優先選擇，玩家-i 以不獨立為第一優先選擇。

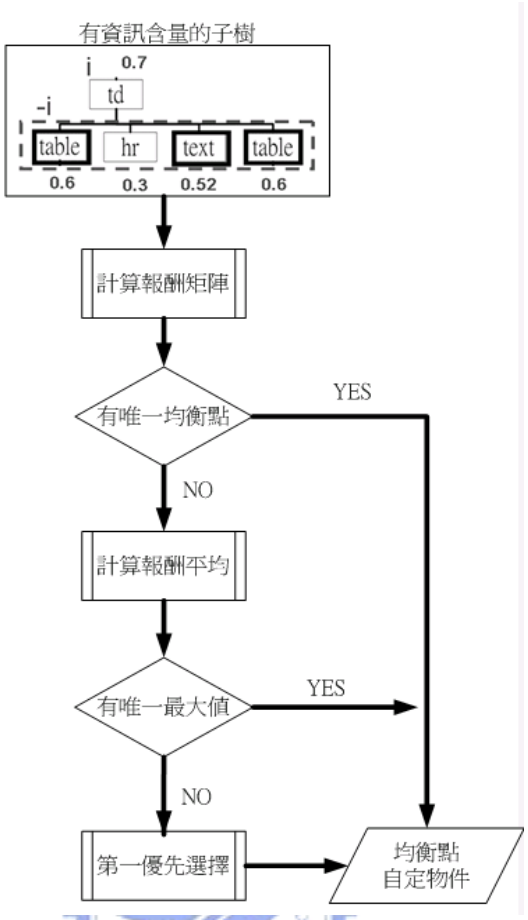


圖 47、賽局決策模組流程

4.1.5 主題區塊樹調整模組

此模組的 Input 是一棵主題區塊樹，實作上以 XML 文件表示；Output 則是一棵調整過的主題區塊樹，同樣是以 XML 文件表示。

這個模組會執行 GRAB 演算法步驟 3，逐一判斷每個主題區塊是否符合門檻值，若未符合的，就把該主題區塊 b1(實際上也只是一段 XML 碼)再丟入『主題區塊轉換模組』，做更細部的切割，傳回一棵更細緻的主題區塊樹 b1 後，再加入原本的主題區塊樹。

若資訊含量介於 H 和 L 之間，就執行 TSTM 演算法，針對主題區塊樹做兩兩子樹比較，計算相似度之後，相似的主題區塊組合成一個一般化節點。

處理過的主題區塊樹，也是一份 XML 文件。系統接著依照上面的 ccmID，在原始網頁上畫方框，代表一個主題區塊。此模組流程圖 48 所示。

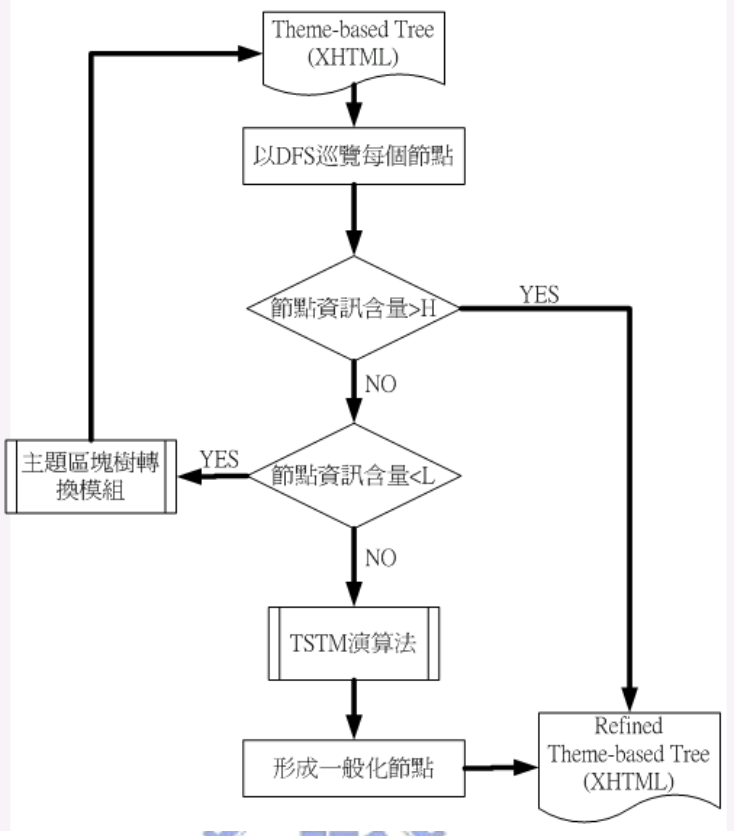


圖 48、主題區塊樹調整模組流程

4.1.6 主題區塊樹擷取模組

模組的 Input 是使用者欲擷取的區塊 ccmID，Output 則是該主題區塊的 HTML 碼。

在系統中，使用者可以看到網頁畫上了許多方框，當使用者決定要擷取某個方框時，就傳入該塊區塊的 ccmID。接著再用這個 ccmID 找到 DOM Tree 中相對應的 HTML 碼，傳回給使用者。模組流程如圖 49 所示。

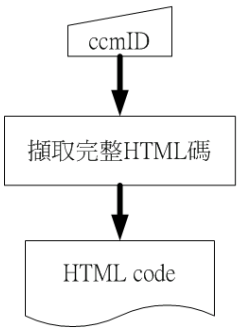


圖 49、主題區塊樹擷取模組流程

4.2 實驗結果與討論

4.2.1 實驗環境與限制

為了驗證 GRAB 演算法的正確性，本論文實際以 Complete-planet 網站(www.completeplanet.com)的資料來進行實驗。Complete-planet 是目前最大的深網庫(deep web)，收集了超過 7 萬個網頁資料及搜尋引擎，其中包括 43 個領域，涵蓋真實世界所有主題領域。本論文從中選擇了 10 個領域，每個領域挑選 3 個網頁，共 30 個網站來進行實驗，如表 12 所示。

GRAB 演算法是針對 W3C DOM Node，考量所有 HTML Node 的特性。由於目前大多數網頁都會使用 CSS 語法來設計頁面樣式，或是嵌入許多 JavaScript 或 Flash 元件，所以這類的網頁無法完整地轉換成 DOM Tree，難以算出區塊權重，資訊含量也就難以計算出來。因此，本實驗的限制是只採用 HTML 為主的網頁，先經由人工方式，判斷網頁內容是否大部份都是 Flash 物件(超過頁面元件的 70%)，並判斷是否包含過多的 JavaScript 或 CSS(超過網頁原始碼 60%)。

為了更進一步驗證 GRAB 演算法對於結構化與非結構化網頁的擷取效果，每個領域所挑選的網站，皆包含了：(1)結構化區塊佔多數的網頁、(2)非結構化區塊佔多數的網頁。以下各別用符號◯表示結構化的網頁；符號△來表示非結構化的網頁。

表 12、實驗網頁資料集

領域	網頁URL	
(1) Business	http：//www.commerce.gov/ http：//www.thestreet.com/ http：//www.cognos.com/	◯ △ △
(2) Education	http：//www.ed.gov/index.jhtml http：//cimc.education.wisc.edu/ http：//www.thegateway.org/	◯ △ △
(3) Finance & Economics	http：//www.investorlinks.com/ http：//www.nationalbank.co.nz/ http：//www.quickmba.com/	△ △ ◯
(4) Health	http：//www.pubmedcentral.nih.gov/ http：//www.medem.com/ http：//www.familydoctor.org/	◯ △ △
(5) Jobs & Careers	http：//www.museumjobs.com/ http：//www.jobspan.com/ http：//www.technojobs.co.uk/	◯ △ △

(6) News	http：//www.usatoday.com/ http：//www.foxnews.com/ http：//www.cnn.com/	○ △ △
(7) Politics	http：//www.ruthsforum.com/ http：//www.nrc.gov/ http：//www.bbk.ac.uk/polsoc/	○ △ △
(8) Science	http：//www.ciesin.org/ http：//www.labinternet.com/ http：//www.forfas.ie/	△ ○ △
(9) Sports	http：//www.makeithappen.com/wis/ http：//www.iis-sports.com/trivia/ http：//www.mext.go.jp/english/	○ △ △
(10) Travel	http：//www.abercrombiekent.com/ http：//www.scubatimes.com/ http：//www.hotelstravel.com/	△ △ ○

4.2.2 實驗設定

在實驗中，本論文所設定的門檻值 t 為 0.6、H 為 0.8 以及 L 為 0.3。這些門檻值的設定是經過測試，選擇對於各個領域、各種屬性中最適當的值，作為本實驗的設定。接下來將要做兩個實驗：

第一個實驗，我們利用 10 個領域的資料(Business、Education、Finance&Economics、Health、Jobs&Careers、News、Politics、Science、Sports、Travel)，來觀察本論文的演算法對於處理這些不同領域的效果。

第二個實驗，將本論文的演算法與現有三大類方法做比較，驗證 GRAB 演算法擷取主題區塊的效能(正確性)，並針對處理結構化與非結構化網頁的效能來做驗證。

為了達到較精準的結果，實驗二的資料集，是從實驗一效果最好的 3 個領域而來。

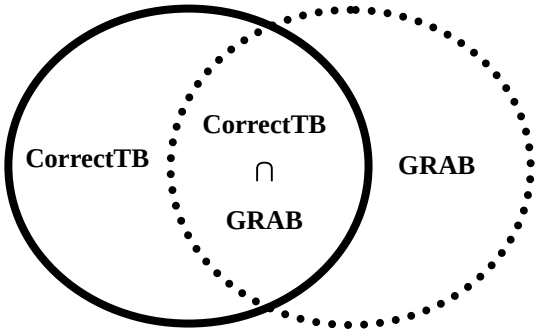
4.2.3 實驗評估方式

本研究的兩個實驗，都是要驗證所有擷取出來的主題區塊，是否具有足夠的主題性、符合使用者想看到的區塊。為了驗證這點，本論文以手動的方式，先標註出每個網頁的主題區塊，製作成一棵主題區塊樹，接著將網頁經由 GRAB 演算法產出一棵主題區塊樹，再將這兩樹主題區塊樹做比對。兩者的相似程度，就是 GRAB 演算法的正確性。

驗證正確性的公式，是採用一般常見的 Precision 與 Recall。如圖 50 所示，CorrectTB

代表所有正確的主題區塊數目，即實線圓形的區域；GRAB 代表 GRAB 演算法找出來的主題區塊數目，即虛線圓形的區塊；而中間的交集部份，就是 GRAB 找出來且正確的主題區塊數目。

此外，本論文也使用 F-Measure 來做為實驗結果的評估方式。F-Measure 是一種平均準確率和召回率的評估方法，當 Precision 及 Recall 愈高時，F-Measure 的品質也就愈好。以下就用兩個小節來說明這兩個實驗的結果。



$$precision = \frac{\text{找到且正確的主題區塊個數}}{\text{找到的主題區塊個數}} = \frac{CorrectTB \cap GRAB}{GRAB}$$

$$recall = \frac{\text{找到且正確的主題區塊個數}}{\text{所有主題區塊個數}} = \frac{CorrectTB \cap GRAB}{CorrectTB}$$

$$F \cdot Measure = \frac{2 \times precision \times recall}{precision + recall}$$

圖 50、precision、recall 及 F-Measure 計算公式

4.2.4 針對各領域的效能(實驗一)

實驗定義：

第一項實驗要驗證 GRAB 演算法對 10 種不同領域的擷取效能。每一種領域皆挑選出 1 個結構化 HTML 網頁、2 個非結構化 HTML 網頁。計算出 precision、recall、以及 F-Measure 之後，再從中挑選出三個領域，做為實驗二的資料。

實驗結果分析：

演算法效能的實驗結果如表 13 所示。將這 10 個領域的 Precision、Recall、及 F-Measure 平均之後，可以得到如圖 51 的實驗結果。從圖表中可以發現，GRAB 演算法在這 10 種領域有很好的效能，平均值皆在 70%到 90%之間。其中又以 Jobs&Careers、

News、與 Politics 這 3 個領域的效能最好，Precision 都將近 90%，Recall 也都接近 80%。因此，接下來將用這 3 個領域的網頁做為實驗二的資料集，以達到更精準的結果。

表 13、主題區塊效能實驗數據(實驗一)

Theme	Web Site	CorrectTB	GRAB	CorrectTB ∩ GRAB	Precision	Recall	F-Measure
Business	WebSite 1	9	7	6	0.857	0.667	0.750
	WebSite 2	14	13	12	0.923	0.857	0.889
	WebSite 3	10	7	6	0.857	0.600	0.706
Education	WebSite 1	10	11	9	0.818	0.900	0.857
	WebSite 2	10	15	9	0.600	0.900	0.720
	WebSite 3	11	14	8	0.571	0.727	0.640
Finance & Economics	WebSite 1	13	18	10	0.556	0.769	0.645
	WebSite 2	14	9	8	0.889	0.571	0.696
	WebSite 3	12	10	8	0.800	0.667	0.727
Health	WebSite 1	10	11	8	0.727	0.800	0.762
	WebSite 2	20	19	17	0.895	0.850	0.872
	WebSite 3	17	22	15	0.682	0.882	0.769
Jobs & Careers	WebSite 1	17	14	12	**0.857**	**0.706**	**0.774**
	WebSite 2	5	4	4	**1.000**	**0.800**	**0.889**
	WebSite 3	7	6	5	**0.833**	**0.714**	**0.769**
News	WebSite 1	35	20	18	**0.900**	**0.514**	**0.655**
	WebSite 2	20	19	17	**0.895**	**0.850**	**0.872**
	WebSite 3	31	27	25	**0.926**	**0.806**	**0.862**
Politics	WebSite 1	8	7	6	**0.857**	**0.750**	**0.800**
	WebSite 2	11	10	10	**1.000**	**0.909**	**0.952**
	WebSite 3	5	5	4	**0.800**	**0.800**	**0.800**
Science	WebSite 1	11	8	7	0.875	0.636	0.737
	WebSite 2	7	7	6	0.857	0.857	0.857
	WebSite 3	9	10	8	0.800	0.889	0.842
Sports	WebSite 1	5	8	5	0.625	1.000	0.769
	WebSite 2	6	6	5	0.833	0.833	0.833
	WebSite 3	10	16	8	0.500	0.800	0.615
Travel	WebSite 1	11	16	10	0.625	0.909	0.741
	WebSite 2	9	7	7	1.000	0.778	0.875
	WebSite 3	8	13	7	0.538	0.875	0.667

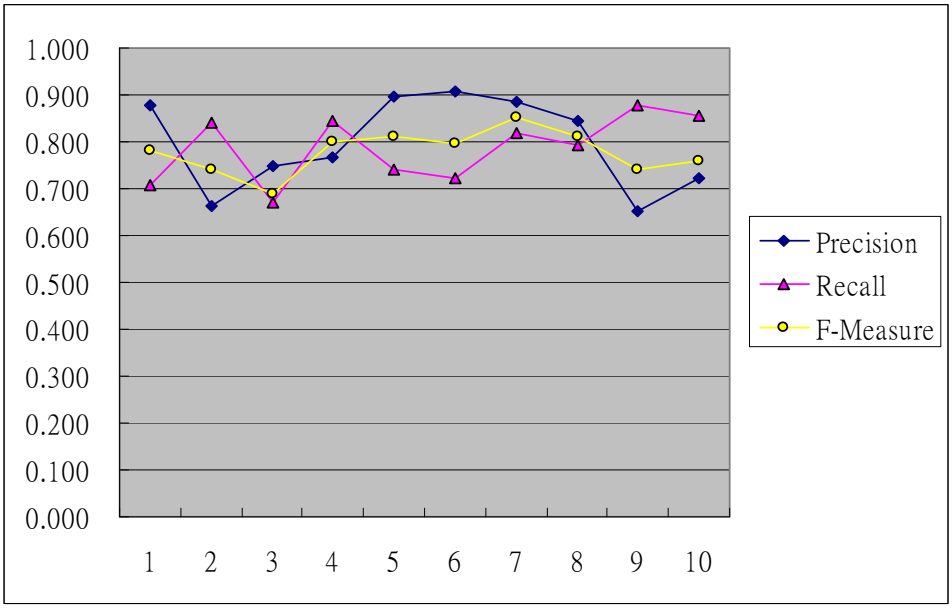


圖 51、主題區塊正確性實驗數據圖(實驗一)

4.2.5 與現有擷取方法比較(實驗二)

實驗定義：

第二個實驗要驗證 GRAB 演算法與現有三種網頁區塊擷取方法的效能比較。資料集是採用 Jobs&Careers、News、與 Politics 這 3 個領域的網頁，接著逐一實作這三類方法，計算每個方法在 3 個領域的 Precision、Recall、F-Measure 平均值。此外，並各別針對結構化網頁、非結構化網頁來做比較，驗證 GRAB 在處理結構化/非結構化網頁的效能。

實驗結果分析：

首先，在準確率(Precision)方面，如圖 52 與表 14 所示，GRAB 演算法在這三個領域的表現都遠比現有的三類擷取方法優秀，也就是說，GRAB 所有挑選出來的區塊裡面，有 90%以上都是正確的。而其中又以新聞網頁(News)與現有方法的效能差距最大，主要的原因是本演算法的第二步驟，有運用到「第三次眼球追蹤」的新聞網頁眼球移動模式，因此對於新聞網頁的效果特別好。

GRAB 有較好準確率的第二個原因，是因為有經由賽局做進一步分析，挑選出最適策略，其中大部份的均衡點是落在(獨立，不獨立)的位置。因此一些資訊含量較小的區塊就會被併掉，最後產生的主題區塊樹也更精簡準確，也能減少最後產生出來的主題區塊數量。

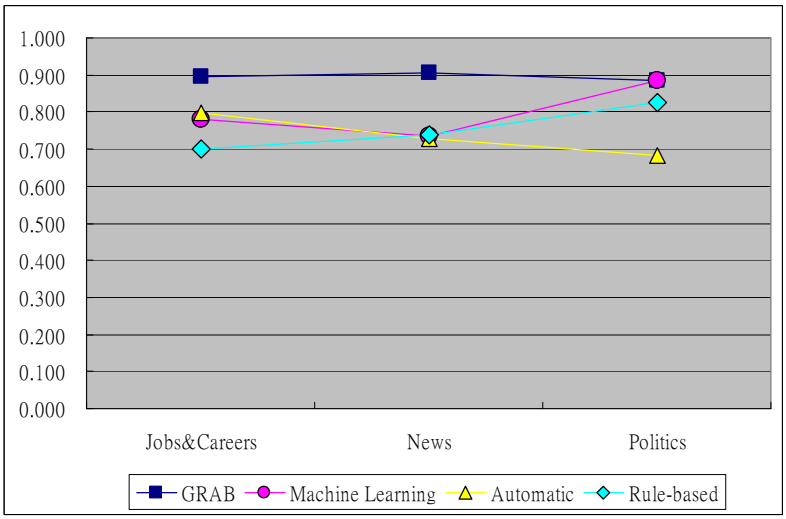


圖 52、與現有三類方法 Precision 比較圖(實驗二)

表 14、與現有三類方法 Precision 比較(實驗二)

方法	Jobs&Careers	News	Politics
GRAB	**0.897**	**0.907**	**0.886**
Machine Learning	0.780	0.734	0.883
Automatic	0.797	0.728	0.681
Rule-based	0.699	0.737	0.825

在召回率(Recall)方面，如圖 53 與表 15 所示，GRAB 演算法在這三個領域的表現，除了在 Jobs&Careers 領域效能較低之外，其他兩個領域的效能也都優於其他三類擷取方法。這樣的結果可以說明，在所有應該要被挑選出來的區塊中，GRAB 演算法能挑出 70%~80%的正確區塊。這樣的效果跟現有三類方法差不多，且在 News 及 Politics 領域的效果比其他三類都要好。

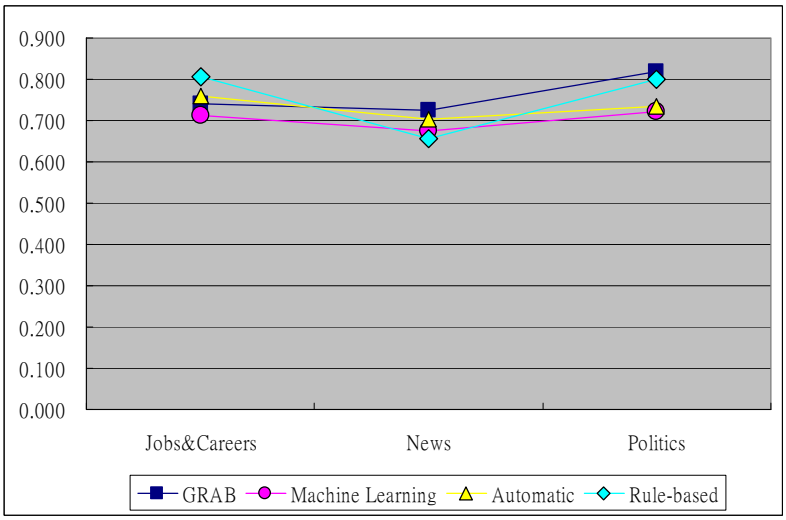


圖 53、與現有三類方法 Recall 比較圖(實驗二)

表 15、與現有三類方法 Recall 比較(實驗二)

方法	Jobs&Careers	News	Politics
GRAB	0.740	**0.724**	**0.820**
Machine Learning	0.713	0.676	0.723
Automatic	0.760	0.702	0.734
Rule-based	**0.807**	0.657	0.801

在效能品質(F-Measure)方面，如圖 54 與表 16 所示，GRAB 演算法在這三個領域皆在 80%以上，且都高於其他三類的擷取方法，證明 GRAB 演算法的效能方面的品質，是優於其他三者。

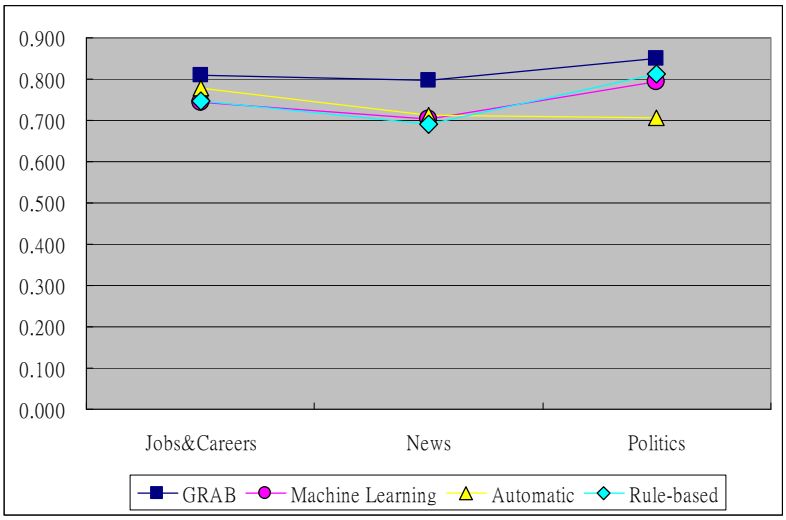


圖 54、與現有三類方法 F-Measure 比較圖(實驗二)

表 16、與現有三類方法 F-Measure 比較(實驗二)

方法	Jobs&Careers	News	Politics
GRAB	**0.811**	**0.796**	**0.851**
Machine Learning	0.743	0.703	0.794
Automatic	0.777	0.714	0.706
Rule-based	0.747	0.691	0.812

在處理結構化網頁方面，如圖 55 與表 17 所示，GRAB 演算法在處理這三個領域的結構化網頁的效果中， News 領域的效果勝過其他三類方法，而且在其他兩個領域上的效能也有不錯的表現(75%以上)

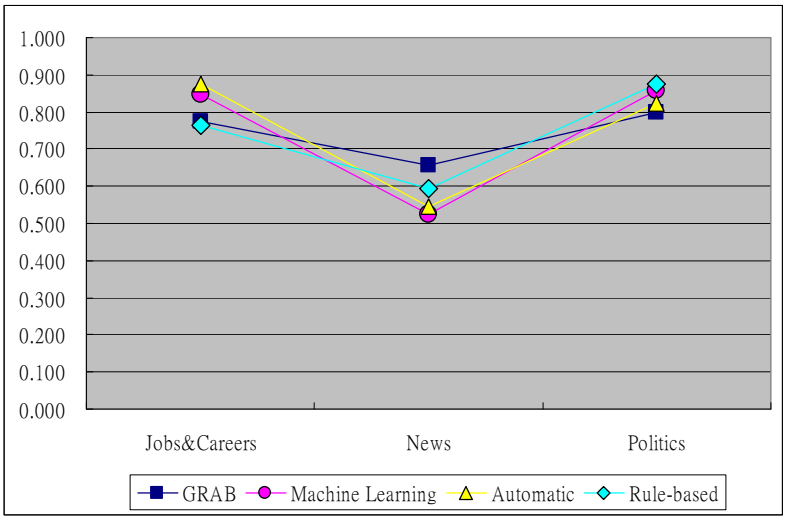


圖 55、處理結構化網頁的 F-Measure 比較圖(實驗二)

表 17、處理結構化網頁的 F-Measure 比較(實驗二)

方法	Jobs&Careers	News	Politics
GRAB	0.774	**0.655**	0.800
Machine Learning	0.848	0.523	0.857
Automatic	**0.875**	0.545	0.824
Rule-based	0.765	0.594	**0.875**

在處理非結構化網頁方面，如圖 56 與表 18 所示，GRAB 演算法在處理這三個領域的效果都遠勝過其他三類方法，擁有 90%到 95%的效能品質。

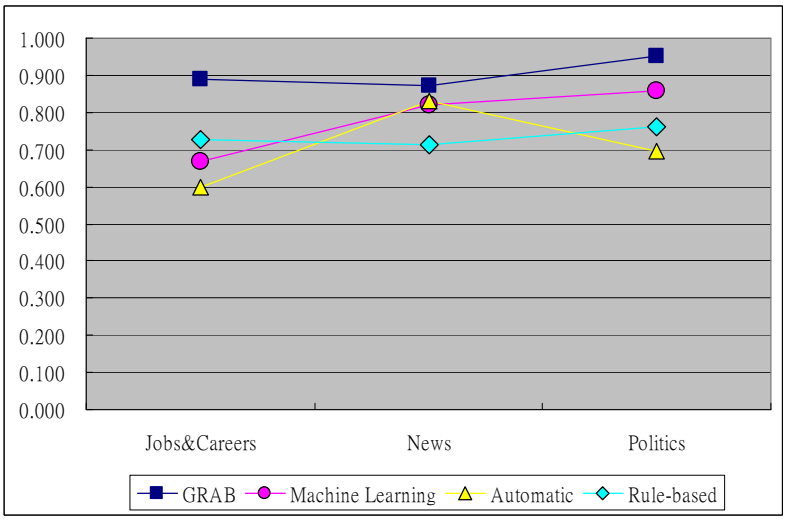


圖 56、處理非結構化網頁的 F-Measure 比較圖(實驗二)

表 18、處理非結構化網頁的 F-Measure 比較(實驗二)

方法	Jobs&Careers	News	Politics
GRAB	**0.889**	**0.872**	**0.952**
Machine Learning	0.667	0.821	0.857
Automatic	0.600	0.829	0.696
Rule-based	0.727	0.714	0.762

4.2.6 總結

本章節概述了實驗系統模組及實驗環境與假設，並設計了兩個實驗：

(1)在實驗一，實際擷取 10 個領域的網頁資料，實驗結果證明 GRAB 演算法具有 70%到 90%的效能(正確性)，其中 Jobs&Careers、News、以及 Politics 這 3 個領域的效能最好，準確率都在 90%。

在實驗一的數據中，可以看出某些領域的跳動幅度相當大，仔細研究擷取出來的區塊後，發現效果較低的網頁中，大部份都是文字，且文字的相似度很高，而 GRAB 演算法第三步驟，是透過 LCS 來計算文字相似度，因此會將這些相似的主題區塊合併起來。

(2)在實驗二，使用實驗一效果最好的 3 個領域做為資料集，與現有的三類方法做比較。實驗結果證明 GRAB 的效能優於現有的三類方法，其中 News 領域的效果差異最大，原因是本演算法採用「第三次眼球追蹤」的新聞網頁眼球移動模式，並利用賽局進行最適決策，因此最後產生的主題區塊樹也更精簡準確。

實驗二也證明 GRAB 在處理結構化網頁的效果，與現有三類方法的效能差不多，但在 News 領域的效能仍是優於現有三類方法。在處理非結構化網頁的效果則是優於現有三類方法，證明本論文所提之 GRAB 演算法，能處理非結構化網頁，並得到很好的效果。

第五章　結論與未來研究

針對現有三大類網頁區塊擷取方法的不足，以及能處理非結構化網頁，本論文提出以賽局為基礎的主題區塊擷取方法，將網頁轉成 HTML DOM Node，並針對每個元件所在位置、特性，計算出資訊含量，再用賽局方法來決定是否要形成主題區塊。具有相似內容的主題區塊會一起呈現，並轉換成易於儲存、檢索與分析的結構化資料，便於往後的應用。

實驗結果證明，GRAB 具有不錯的效能，尤其是在新聞領域更具效果。與現有方法比較之後，也證明 GRAB 具有較好的效能，而且處理非結構化網頁的效果很好。

本論文提出的 GRAB 演算法，是以 W3C 定義之 DOM Node Type 為基礎，並進一步擴充定義兩種 Node，做為計算資訊含量的基礎，結合眼球追蹤、網頁元件特性、以及賽局理論的方法，能讓資訊含量的計算更符合讀者在閱讀網頁的行為模式，賽局的方式能改變部份主題區塊原本的決策，找到一個讓兩位玩家皆滿意的決策，並能減少擷取出來的主題區塊數目，也就能減少產生主題性不足的主題區塊。

未來研究的方向整理如以下幾點：

(1)擴大處理能力

目前本論文的 GRAB 演算法，針對 HTML 網頁有很好的效果，但對於包含許多 CSS、JavaScript、<DIV>、或以 Flash 撰寫的網頁，效果就較不理想。現今也愈來愈多網頁利用 CSS、JavaScript、<DIV>標籤來建置網頁，因此未來可加強在這部份的處理。

(2)結合語意網

除了在能夠準確的擷取使用者有興趣的網頁主題資料之外，希望更可以建立主題之間的相關性，進而可以利用語意網路的建模方法建置使用者可能會有興趣的相關主題，提供更有效率的資料擷取與探勘的方法。

(3)語意相似計算

目前在計算主題區塊相似度的方法是採用 LCS，只考慮文字。未來可以設計成語意相似計算，讓主題區塊樹的整併可以更精確。

參考文獻

英文部份：

[1] B. Chidlovskii, “Wrapper Generation by k-Reversible Grammar Induction”, In ECAI2000 workshop on Machine Learning for Information Extraction, 2000. Access from http：//citeseer.nj.nec.com/469912.html on June 2002

[2] B. Liu, R. Grossman, and Y. Zhai. Mining Data Records in Web Pages. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD’03), Page 24-27, 2003.

[3] B. Liu, and Y. Zhai. Web Data Extraction Based on Partial Tree Alignment. In the Proceedings of the 14th international conference on World Wide Web, Page 76-85, 2005.

[4] B. Liu, and Y. Zhai. Structured Data Extraction from the Web Based on Partial Tree Alignment. IEEE Transactions on knowledge and data engineering, vol.18, no.12.

[5] C. A. George, "Usability testing and design of a library website：an iterative approach, "OCLC Systems & Services： 21：3 (2005)： 167-180

[6] C. H. Chang and S. C. Lui. IEPAD： Information Extraction Based on Pattern Discovery. In Proceedings of the 10th international conference on World Wide Web, Page：681-688, 2001.

[7] C. N. Hsu, and C. C. Chang. Finite-state Transducers for Semi-Structured Text Mining. In Proceedings of IJCAI-99 Workshop on Text Mining： Foundations, Techniques and Application, Page 38-49, 1999.

[8] C. N. Hsu, and M. T. Dung. Generating Finite-state Transducer for Semi-Structured Data Extraction from the Web. Information Systems, 23(8)：521-538, 1998

[9] D. Cai, S. Yu, J. Wen, and W. Ma, "Extracting Content Structure for Web Pages Based on Visual Representation", in Proc. APWeb, 2003, pp.406-417.

[10] D. Cai, S. Yu, J. R Wen, and W. Y Ma. VIPS： a Vision-based Page Segmentation Algorithm. Microsoft Research, Redmond, WA 98052.

[11] G.. Robert. Game Theory for Applied Economists. Princeton Univ Pr, Page 1-11, 1992.

[12] I. Muslea, S. Minton, and C. A. Knoblock. STALKER： Learning Extraction Rules for Semistructured Web-based Information Sources. In Proceedings of AAAI Workshop on AI and Information Integration, Pages 74081, 1998.

[13] J. Han, D. Han, C. Lin, H. Zeng, Z. Chen, and Y. Yu, "Homepage live： automatic block tracing for web personalization", in Proc. WWW, 2007, pp.1-10.

[14] J. Song, D. L. Wang, Y. B. Bao, and D. R. Shen, “collecting and storing web archive based on page block”, Journal of Software, 2008, 19(2), pp.275-290.

[15] J. Wang, and F. H. Lochovsky. Data Extraction and Label Assignment for Web Databases. In Proceedings of the twelfth international conference on World Wide Web, Page 187-196, 2003.

[16] L. Eikvil, “Information Extraction from world wide web -A Survey-”, Norwegian Computing Center, No. 945, July 1999. Access from http： //citeseer.nj.nec.com/eikvil99information.html onJune 2002.

[17] N. Kushmerick, D. S. Weld, and R. B. Doorenbos. Wrapper Induction for Information Extraction. In Intl.Joint Conference on Articial Intelligence (IJCAI),pages 729-737, 1997.

[18] N. Kushmerick, “Wrapper Induction： Efficiency and Expressiveness”, Artificial Intelligence, Vol. 118, Iss. 1-2, pp. 15-68, April 2000.

[19] N. Kushmerick, D. S. Weld, R. Doorenbos, “Wrapper Induction for Information Extraction”, Intl. Joint Conferenceon Artificial Intelligence (IJCAI), pp. 729-737, 1997.

[20] S. J. Lim, Y. K. Ng, “Change Discovery of Hierarchically Structured, Order-Sensitive Data in HTML/XML Documents”, Applications and the Internet Proceedings 2004 International Symposium, pp. 178-187, 2004.

[21] S. Lin and J. Ho, "Discovering informative content blocks from Web documents", in Proc. KDD, 2002, pp.588-593.

[22] S. Outing, L. Ruel, "The Best of Eyetrack III：What We Saw When We Looked Through Their Eyes, "<http：//www.poynterextra.org/eyetrack2004/main.html>(14 May 2007)

[23] S. Yu, D. Cai, J. R. Wen, and W. Y. Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, pp. 11-18, Budapest, Hungary, May 20-24, 2003.

[24] T. Peng, C. Zhang, and W. Zuo, "Tunneling enhanced by web page content block partition for focused crawling", presented at Concurrency and Computation： Practice and Experience, 2008, pp.61-74.

[25] W. Liu, X. Meng, and W. Meng. Vision-based Web Data Records Extraction. In Proceedings of the 9th SIGMOD International Workshop on Web and Databases (SIGMOD-WebDB2006), Chicago, Illinois, June 30, 2006.

[26] Y. F. Tzeng. The Mining and Extraction of Primary Informative Blocks and Data Objects from Systematic Web Pages.

[27] Y. Kim, J. Park, T. Kim, and J. Choi. Web Information Extraction by HTML Tree Edit Distance Matching. 2007 International Conference on Convergence Information Technology.

中文部份：

[28] 李季壕，動態網頁之樣版與資料分析研究

[29] 李泓儒，淨化網頁：網頁區塊化以及資料區域擷取

[30] 李逸群，網頁異動偵測技術在網際網路新聞資訊擷取上之應用

[31] 姚文鋒，網站內網頁之區塊等級分析

[32] 范綱岷，”使用超本文標記語言剖析樹建構多網頁資訊萃取及融合代理人”

網站部份：

[33] Document Object Model(DOM)-W3C Recommendation. http：//www.w3c.org/DOM/