Adaptive Page-level Full Schema Induction
完整綱要推導之改進

丁中立
國立中央大學
pizza0504@gmail.com

張嘉惠
國立中央大學
chia@csie.ncu.edu.tw

摘要

在網頁資訊擷取(Web Data Extraction)的領域中，如何自動的從各種不同架構的網頁中擷取資料的相關議題至今已被探討研究十年，然而由於現今網頁的內容多樣與架構的複雜，現有的方法均有其限制之處，再加上大量網頁擷取的需求，使得網頁資訊擷取的研究仍面臨相當大的挑戰。

網頁資料擷取系統主要分成記錄層級(Record Level)和頁面層級(Page Level)兩大類別，雖然頁面層級相較於記錄層級能夠得到更完整的網頁資訊，但由於問題的複雜及實作的困難，使得現今提出的系統中，其擷取的效能與效率都有改進的空間。

在本篇論文當中，我們提出了一套頁面層級資訊擷取系統，以 M.-C. Chen 及 T.-S. Chen 所提出的頁面層級系統的架構為基底。而非監督式擷取不佳的情況往往需要使用者介入，因此我們也提供一個簡單友善的圖形介面，讓使用者可以用此系統，快速擷取出所需要的網頁資訊。此外我們再往上對其訓練的流程做改良，提升系統的擷取效能；在本論文的實驗中顯示，對於訓練的流程上的改良結果，準確率(Precision)提升了 30.1%、召回率(Recall)提升了 25.6%，在整體效能比較中，改善後的系統得到了最高的召回率 92%。在精確度(Accuracy)部份，實驗顯示改良後的系統以預設的模組參數值，在整體精確度就比 TEX 還要高出許多；若是再以人工調整模組參數，整體精確率可再向上提升至 98.8%，整體精確率比 TEX 還要高 27%。

關鍵詞：資料擷取，地標，使用者介面。

1. 前言

在現今的社會中，網際網路已成為人們最主要的資訊傳播媒介之一，各式各樣的資訊都散佈於全球資訊網(World Wide Web)內的各個角落，為了能夠有效率的吸收這龐大的資訊量，「搜尋」的功能變得異常重要，不論是使用搜尋引擎(Search Engine)尋找所需要的資訊，還是各大網站提供的站內搜尋，以尋找特定的實體資訊，這些動態產生的搜尋結果頁面已成為使用者生活中不可或缺的資訊來源。

這些動態產生結果的頁面通常是根據使用者所輸入的查詢詞(Query)而動態產生的網頁，這些網頁主要用來呈現搜尋後的結果；而結果網頁則通常是由網頁產生模組(Page Generation Model)來自動產生，透過 CGI(Common Gateway Interface)程式將欲呈現的資料嵌入固定的樣板(Template)架構來生成網頁。

使用者若想針對某種網頁進行資訊的擷取，非資訊背景的使用者的流程通常是在瀏覽器(Browser)中手動把資訊拷貝出來整理；而具資訊背景的工程師，則會使用網路爬蟲(Web Crawler)將網頁檔案抓下來，接著人工分析網頁的結構，再各別為各種不同結構的網頁撰寫多個不同的擷取程式(Extractor)來取出網頁中的資料。一但原始網頁改變架構，則擷取程式也需要重新撰寫，以對網頁結構的改變進行修正。不論是手工擷取網頁資訊或是針對不同結構的頁面撰寫擷取程式，皆相當浪費人力資源。網頁資訊擷取系統即是為此設計。

網頁資訊擷取系統的做法依照輸入的網頁數量可分為單一網頁(Single Page)分析或是多重網頁(Multiple Pages)分析，單一網頁分析是處理單一網頁中重複的區域進而對網頁中的內容進行區分，目標即為擷取出頁面中的記錄區域，因此也被稱為記錄層級(Record Level)；多重網頁分析則是分析多個網頁中相似的部分與相異的部分，進而產生出網頁所對應的綱要(Schema)，多重網頁分析除了擷取出網頁中的資料，更重要的目標是能夠推導出網頁背後的架構，故也可稱作頁面層級(Page Level)。

單一網頁分析的缺點有三點：1.可能誤認資料擷取樣版區塊為 False Positive 的資料區塊，但從多個網頁的角度分析如發現該區塊在所有網頁完全相同，即可過濾掉這類的錯誤；2.由於單一網頁資訊有限，較難有效透過推導出那些資料節點是屬於不一定會出現的選擇性節點(Optional Node)；3.承上所述，單一網頁分析產生的擷取程式及綱要通用性較差，對大量網頁的擷取較為不利。相對的，網頁層級的網頁可克服上述的困難，產生完整的綱要，提供後續網頁的擷取時省略分析的過程，更有效率的對相同樣版的網頁進行擷取，這些都是記錄層級系統所無法達成的優點。

儘管頁面層級多出上述的優點，但其在實作上卻要比記錄層級要困難許多，因為同時考慮多個網頁的資訊雖然有更豐富的有資訊可用，但相對的過多的資訊也同時帶來了很多雜訊，增加資料分析的

困難度。尤其是在詳細網頁中排比程序(Alignment Procedure)比表列網頁更為困難，繁雜的分析過程所耗費的時間也較記錄層級要來得多。截至目前為止，頁面層級相關的研究並不多，有推導出最後網頁的綱要的研究更是稀少。

另一方面資料擷取系統是否有搭配的驗證擷取程式也影響了擷取系統的效率。以記錄層級非監督式(Unsupervised)擷取系統來說，常見的做法一次只分析一個網頁，對於 N 個網頁的擷取工作就必須進行 N 次的分析與擷取。而頁面層級擷取系統的架構，多個網頁同時經過分析後，推導出網頁的綱要並擷取出資料，之後即可以此綱要對其它相似網頁透過簡易的驗證程序便可快速地擷取出資料。相較於過去針對每一網頁進行分析的非監督式擷取系統架構而言，越大量的網頁擷取工作，所帶來的效益更加龐大。

目前網頁層級有陳等人提出[7, 8]的做法，雖然在表列網頁上有很好的結果，但是在詳細網頁的結果並不是很好，其主要原因在於排比(Alignment)的效果不好，效果不好的理由有三點：1、對於沒有記錄集合的網頁，若誤判將不是記錄集合的部分加入記錄集合中，會讓整個記錄集合的整個排比亂掉。2、原架構的動態編碼只找最高頻率來做，對於網頁的整體支援較不足，易產生大量選擇性節點，影響排比的結果。3、對於某些網頁，雖然位於相同的區塊，但是會因為內容完全不同，導致這些同樣區塊的排比結果，因內容不同而不會排比在一起，使得產生的結果效果很差。因此本論文將以此架構為基底，改善原系統擷取準確度的不足之處，並且提供簡單友善的使用者介面及 API 供程式呼叫，讓所有背景的使用者，可以透過本系統快速的擷取網頁上的資訊。

本論文共分成五個章節，第一章為緒論，說明研究的動機與目的；第二章為相關研究，介紹現有的頁面層級擷取系統的相關研究；第三章為系統架構與研究方法，詳述如何透過修正訓練流程，改善整體擷取準確度；第四章為實驗，評估效能的改善；第五章為結論，總結本論文的貢獻。

2.相關研究

網頁資訊擷取的研究至今為止，已有約 10 年的歷史了，大致分成人工、監督式與非監督式。人工的方式是由程式設計者觀察網頁之架構，撰寫擷取程式來抓所需要的資料，但是這個方式需要有資訊背景的工程師，成本過高；監督式的做法，則需要人工去標記網頁，告訴系統擷取的目標，雖然省下了寫程式的成本，但是對於各式各樣不同的網頁，仍需要花費大量的時間去標記網頁；非監督式則是由系統自行分析網頁的架構，擷取存在網頁中的資料，過程中不需要使用者的介入，對使用者來說是負擔最小，因此也是近十年研究的主要方向。

本篇論文著重在非監督式網頁層級的擷取系統，其優點如緒論所述，但是在系統實作上卻是一大挑戰，不僅要能夠容忍網頁間相異且過於自由的架構，還要能夠正確的擷取出網頁中的資料。因此，非監督式頁面層級的研究並不多，現今主要有 2001 年的 RoadRunner[9]、2003 年的 EXALG[1]、2010 年的 FiVaTech[10]、2013 年的 TEX[13]以及 2014 年由 M.-C. Chen[7] 及 T.-S. Chen[8]所提出的架構。

RoadRunner[9]利用標籤的字串排比(String Alignment)來處理標籤(Tag)與文字符記(Token)，接著使回溯(Backtracking)演算法來找出網頁中重複的區塊，其最大的缺點就是回溯演算法的時間複雜度為指數時間(Exponential Time)。另外 RoadRunner 系統無法處理表格標籤(也就是<table>、<tr>、<td>)，會直接把原始的表格標籤格式輸出，對於以表格為主的網頁無法運作。

EXALG[1]則是使用等價類(Equivalence Class)的概念，將有相同出現頻率的文字符記切分成同一個等價類，並在多個等價類中，找出一個最大的等價類作為網頁的樣版(Template)，接著利用這個樣版，將所有文字切開分析，若切出的文字過於瑣碎，很容易導致網頁的樣版過於複雜。FiVaTech[10]則是利用網頁的樹狀結構，計算樹狀的編輯距離(Edit Distance)對節點進行了分群(Clustering)及排比(Alignment)，這些做法通常會具有較高的複雜度。TEX[13]則是尋找文字集合(TextSet)中的共同樣式(Shared Patterns)當為資料的地標，遞迴地切出最後的資料，節省了處理樹狀結構的時間，但缺點是沒有推導出網頁的綱要，而是輸出個別欄位，因此需要使用者將不同欄位之間的關係對應起來。

由於網頁層級的擷取系統之困難，除了 RoadRunner 之外大部份論文如 EXALG 和 FiVaTech 並沒有進一步的提出要如何透過訓練出來的綱要，擷取其它相同樣版的網頁；RoadRunner 則是透過主動學習(Active Learning)的方式，一次和一個新的網頁做匹配來訓練擷取規則(Wrapper)，然而其方法無法一次進行多個網頁的訓練；而 TEX 更是沒有產生網頁綱要。近年來僅有 T.-S. Chen 及 M.-C. Chen 所提出的架構，利用訓練出來的綱要來擷取其餘大量的測試網頁，是上述系統中流程架構最完整的系統。T.-S. Chen 及 M.-C. Chen[7,8]所提出的架構，則是以葉節點為處理的單元，節省了處理樹狀結構的時間，接著使用地標偵測(Landmark Detection)及重復樣板探勘(Repeat Pattern Mining)來決定網頁記錄集合所在的範圍，擷取資料並同時產生綱要，之後利用已經產生出來的綱要，去對其它的測試網頁，利用之前產生的綱要生成有限狀態機(Finite State Machine)，省去訓練分析的流程，直接擷取測試網頁的資料，效率上述系統中最快，其效率甚至比 TEX 還要快上 50 倍。但是系統的訓練流程中仍存在一些缺陷，使得擷取資料的精確度較為不佳。最後，在上述的系統中，沒有任何一個系統有提供圖

形化介面(Graphical User Interface)給非資訊背景的一般使用者。

3. 系統架構與研究方法

Chen 等人[7,8]所提出的架構主要是接受相同樣版的多筆網頁，經過前處理(Preprocessing)將彈性語法的網頁整理成符合標準語法且適合於程式分析的架構後，依續執行動態編碼(Dynamic Encoding)，對葉節點進行抽象化；接著進行地標偵測(Landmark Detection)，找出代表各筆記錄的節點；如果沒有偵測到合適的地標，則進行重複樣式探勘(Repeat Pattern Mining)，以葉節點的規律進行候選樣板的找尋分析後找出記錄的範圍(Record Boundary)。決定出記錄的範圍之後，則對所有的記錄及記錄外的網頁進行排比程序(Alignment Procedure)，擷取出訓練網頁的資料，並以此排比結果來產生綱要。在原來的訓練流程中，主要碰到的問題在於：

(a) 對於沒有存在記錄集合(Set)的訓練網頁時，仍可能會因為重複的樣式，找出錯誤的記錄集合，把不相關的東西全部放進記錄集合內，排比的結果不好，進而影響整體的效能。

(b) 動態編碼計算最高出現頻率去找出網頁中強調查詢詞(Query word)的部份去做動態編碼，但一個網頁中，往往不只有查詢詞的部份被強調而已，若僅對查詢詞做動態編碼，效果並不顯著；且在詳細網頁(Detail Page)中，往往不會有查詢詞的存在，讓動態編碼在詳細網頁中，無法產生任何有效的幫助。

(c) 對於過於豐富同樣區塊內容卻不盡相同的網頁，會產生大量的選擇性節點，這些分散又無法完全合併的節點，對於系統及使用者來說，是一大負擔。

為了改善上述的三種問題，本論文提出了新的訓練流程(**圖 1**)，新增了全網頁排比、動態編碼擴充、及低密度合併三個模組。各模組的詳細內容將於本章各小節中詳述。

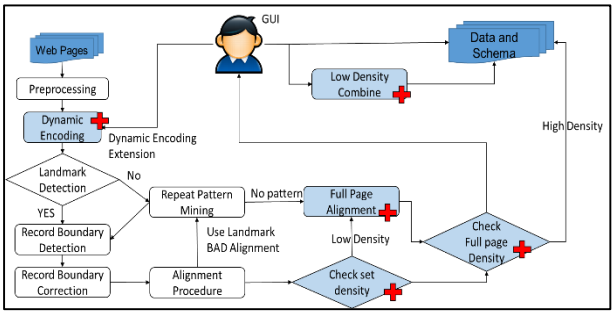

圖 2 系統訓練流程圖

3.1 全網頁排比(Full page alignment)

在 Chen 等人[7,8]的架構中，進行完前處理及動態編碼之後，接著就會利用地標偵測來找尋合適的地標，若沒有找到合適的地標則會進行重複樣式探勘(Repeat Pattern Mining)，但是這方法在遇到根本沒有存在記錄集合(Set)的訓練網頁時(**圖 2**)，方法會失效，因為網頁沒有合適的地標及重複的樣式，此時，系統會在沒有記錄集合的網頁中，硬找出錯誤的記錄集合，甚至可能會因為找不到記錄集合而導致綱要的產生失敗。

Association	Danish Football Association
Founded	1889
UEFA Affiliation	1954
FIFA Affiliation	1904
President	HYLDGAARD Poul
Gen. Secretary	HANSEN Jim Stjerne
Press Officer	BERENDT Lars
National Team Coach	JOHANSSON Bo

圖 1 沒有記錄集合的訓練網頁

首先，需要先定義何為不好、錯誤的記錄集合，之後才可以決定是否進行後續的調整。在本論文中採取資料密度(Density)的概念去檢視記錄集合的結果；若是一個經過資料排比(Alignment)後的結果，發現明明是在同樣樣板的網頁中重覆出現的記錄集合，但沒有辦法很好的排比在一起，使得最後整個產生的記錄集合結果密度過於低落時，就會認為或許是網頁中並沒有記錄集合的存在，導致系統找出錯誤的記錄集合，此時就需要進行後續的調整；反之若排比完的結果很好，表示最後產生的記錄集合結果的密度相當高，則會判斷此為良好、正確的記錄集合。定義記錄集合的密度計算公式如下：

$$\text{set density} = \frac{|b|}{|s|}$$

其中|s|為整個記錄集合中的基底節點(Basic Node)的所有個數，|b|則為記錄集合 s 中非空值的基底節點個數。當密度不足 0.6 時，我們會認為這是一個低密度的記錄集合，原始的網頁中可能沒有記錄集合，進而重新訓練網頁，將整個網頁做排比。

3.2 動態編碼擴充(Dynamic encoding extension)

根據 M.-C. Chen[7]的架構，動態編碼會計算最高出現頻率以找出網頁中強調查詢詞的部份去做動態編碼，但一個網頁中，往往不只有查詢詞的部份被強調而已，若僅對查詢詞做動態編碼，效果並不顯著；且在詳細網頁中，往往不會有查詢詞的存在，讓動態編碼在詳細網頁中，無法產生任何實質上的幫助。因此，我們需要將動態編碼調整的條件給放寬，讓他不僅是對查詢詞調整，而是希望在不影響原始網頁架構的前提下，進行更多的抽象化。首先，仍然需要定義何為抽象化程度較為不足、需要去做動態編碼的網頁，也就是說，在原始的網頁中，裡面的標籤有過於凌亂的文字格式化標籤，進而影響到整體的效能，這時就需要對這些凌亂的網

頁去做動態編碼，提升抽象化的程度；若原始的網頁，本來就已經是很漂亮的構架，我們會不希望對它有過多的干涉。

在本論文中，類似於 3.1 節的網頁重新訓練，仍然是採取資料密度的角度去思考，只是 3.1 節是以記錄集合為對象，而此處則是以整個網頁為對象，在做法上是先根據 M.-C. Chen 的架構，去對原始的網頁做動態編碼，接著檢視這個動態編碼跑出來的結果密度是否足夠，密度足夠的話，則表示此網頁原始的抽象化程度已經夠高，不應該再加以干涉；反之則回去擴充動態編碼。定義資料密度的計算公式如下：

$$density = \frac{\sum(density(c) < 0.6)}{|C|}$$

其中 C 代表網頁中的所有基底節點所各自對應的欄位，而 density(c)則表示為某一個基底節點對應的欄位之密度，我們將密度閾值設定為 0.6，density(c)小於 0.6 表示該基底節點對應的欄位密度過低，而密度過低的欄位一旦超過 3 成(也就是 density>0.3)時，就會啟動動態編碼擴充的模組，找出所有網頁裡面需要調整的路徑，對網頁的這些路徑進行調整，以達成整個網頁的抽象化。

圖 4 應調整的非查詢詞之強調節點

在使用者介面上，允許使用者可以多次遞迴地使用動態編碼擴充，直到使用者檢視結果認為符合預期，也可以在第一次的動態編碼後，直接使用低密度合併；在使用案例上，動態編碼修正至多一至二次即可，若遞迴次數過多，會嚴重破壞原始網頁的架構，此處系統預設為先進行第一次的動態編碼修正，嘗試解決問題，若問題無法得到解決，則改而使用低密度合併(3.3 節)。

3.3 低密度合併(Low density combine)

在某些類型的詳細網頁中(**圖 4**)，雖然是套用了同樣的樣板，但是卻會有一部份的區塊因為使用上的需求，而使得該區塊的內容不盡相同，雖然在人的眼中看起來可能還是有些相似，但實際上在網頁裡的標籤及路徑已經完全不相同了，因為這種情況的發生，會讓排比的過程中，讓這些明明是相同區塊的部份，卻又因為標籤及路徑的不同，讓產生大量的選擇性節點，這些大量的選擇性節點，會嚴重影響使用者介面上的體驗以及系統擷取的效果。

圖 3 相同區塊但內容相異

在 M.-C. Chen 及 T.-S. Chen 的架構中並沒有特別去處理，而這個問題在比較簡單例子上，或許還可以用動態編碼來解決，但是在實際的案例上往往並不是那麼簡單就可以解決。在本論文的系統中，如果遇到這樣的問題，會先嘗試使用前述(3.2 節)的動態編碼擴充來修正較簡單的案例，接著檢視是否仍然存在相同的問題，也就是說檢查動態編碼有沒有解決這個問題，如果還是沒有得到太大的改善，此時，我們就會認為是網頁本身的結構過於複雜，光使用動態編碼並無法解決，這時，就會傾向使用低密度合併來解決。

啟動低密度合併模組的條件與 3.2 節啟動動態編碼擴充的條件相同，也就是計算網頁中所有基底節點所各自對應的欄位之密度，如果欄位的密度過低(小於閾值 0.6)，且密度過低的欄位數超過全部欄位數的 3 成時，就會先行啟動動態編碼擴充的模組，之後再去用同樣的條件檢查密度是否仍然過低，如果動態編碼擴充有成功的解決問題的話(密度低於 0.6 的欄位數小於全部欄位數的 3 成)，表示動態編碼擴充模組，已經成功解決問題，則不需要啟動低密度合併；反之若仍無法解決問題，則認為是網頁本身的結構過於複雜，光使用動態編碼並無法解決，進一步的啟動低密度合併，將位於相同區塊但是內容卻不盡相同的部份做合併。

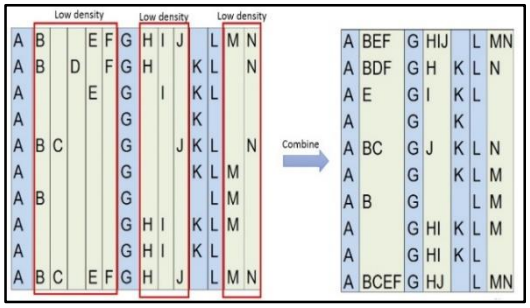

圖 5 低密度合併(低密度合併)意示圖

4.研究實驗

實驗主要分成兩個部份來討論，第一部份是評估三個模組對於訓練流程的改良的情況；第二部份則

是將改善後的系統與其它的系統做比較。實驗 4.1 節及 4.2 節所用到的資料集為 EXALG，包含了 9 個網站 242 個網頁；而 4.3 節則是使用 WEIR 的資料集，分成四大類型(書、股票、足球、遊戲)，包含了 40 個網站及 24028 個網頁。

4.1 模組間效能之改良

此處效能評估的定義如下：

$$\text{Precision} = \frac{B_{Sys} \cap B_{Ans}}{B_{Sys}}, \text{Recall} = \frac{B_{Sys} \cap B_{Ans}}{B_{Ans}}$$

$$\text{F1} - \text{Score} = \frac{2PR}{P+R}$$

其中 B_{Sys} 及 B_{Ans} 分別代表系統找到的基底節點數及答案中標記的基底節點數。

從**圖 6** 我們可得知在 M.-C. Chen 及 T.-S. Chen 的架構中準確率與召回率只有 39.9%及 66.4%有很大的改善空間，而最終改善後的效能可達準確率 70%與召回率 92%，分別提升了 30.1%與 25.6%。接下來將詳述單獨使用各個模組所帶來的效能之改善。

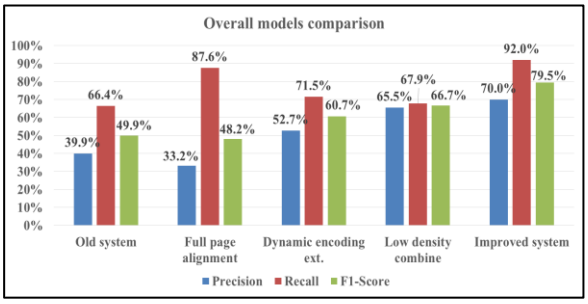

圖 7 模組整體效能比較圖

Full Page Alignment 代表原始的架構再加上全網頁排比模組，在表列網頁的部份，因為集合的密度夠高，所以不會受到模組影響而下降；而在詳細網頁中，可以把包含錯誤集合排除對於沒有任何記錄集合的網頁(**圖 2**)，本來有誤判為集合的情況，已經被完美的修正，加入此模組，雖然準確率稍稍下降，但是召回率卻有了大幅的改善提升。

Dynamic Encoding Extension 代表原始的架構再加上動態編碼擴充模組，對於內容較多格式化文字節點的詳細網頁會發揮功效(**圖 3**)，將一些被格式化文字節點所切開的文字合併起來，同步提升了準確率與召回率。值得一提，對於網頁有包含相同區塊但內容相異的問題(**圖 4**)只靠這個模組並沒有辦法得到很好的改善，因為其網站內容有在同一區塊卻不盡相同的情形發生，這些節點並沒有辦法經由動態編碼擴充模組來合併。

Low Density Combine 代表原始的架構再加上低密度合併模組，發揮於網頁存在同一區塊但是內容不盡相同的情形(**圖 4**)，把高密度的區塊當成地標，去合併地標間的低密度區塊，大幅提升了詳細網頁的準確率。

4.2 資料擷取系統間之效能比較

此處效能評估的定義同 4.1 節，以基底節點為角度，其實驗結果如**圖 7**，我們用改良過後的系統效能與 EXALG、FiVaTech、Road Runner、TEX 做比較，其中 EXALG 及 FiVaTech 的數據引用了 T.-S. Chen 等人的論文，而 Road Runner 與 TEX ，則是以他們公開出來的程式去做計算，而對於有網頁中有表格的情況 RoadRunner 不做處理只把整個原始的網頁標籤輸出，因此效能較低；加入了三個模組而得到效能改良的系統，整體的召回率達到 92%，為所有系統最高，但是對於整體精確度 70%卻是所有系統中的倒數第二低，其原因類似於 TEX 系統，我們的系統會把一些基底節點的屬性令為互斥(disjunction)，也就是說可能會有某些基底節點雖然是同一個，但是它們有互斥的特性，會把一個基底節點切成多個來表示，使得會產生較多的基底節點，造成準確率偏低；在圖形介面的使用上，使用者可以把這些節點合併。

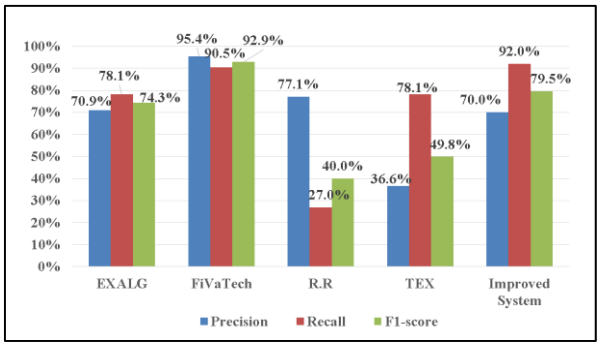

圖 6 效能比較圖

4.3 資料擷取系統間之精確率比較

在本節我們採用 WEIR 的資料集，比較對象為我們系統的預設參數與人工調整後的參數及 TEX 系統。WEIR 資料集有四種類型(書、股票、足球、遊戲)的詳細網頁，包含了 40 個網站及 24028 個網頁。而在精確率(Accuracy)計算的部份，公式定義如下：

$$\text{accuracy} = \sum sys_b \Big/ \sum ans_b$$

其中，ans_b 為 weir 所提供的人工擷取的基底節點 b 的答案，sys_b 則為系統所擷取出正確的基底節點 b 的個數。

從**圖 8** 可知，我們的系統使用預設參數值的整體精確率就比 TEX 還要高出 22.7%；若是再以人工調整參數，整體精確率可再向上提升至 98.8%，整體精確率比 TEX 還高出 27%。我們的系統在預設參數值下，造成某些網站低精確率的主要原因有兩種：一、會因找到錯誤的記錄集合，且該集合的密度高於預設值，使得全網頁排比模組不會啟動，錯誤無法得到修正；二、有些資料集，會因為該基底

節點的資料密度低於預設的閾值，雖然資料被正確的擷取出來，但是卻因為低密度合併模組，讓正確的資料不小心被合併的情況發生。而以上這兩種問題是源於不同領域、不同架構的網頁，本來就會有不同適合的參數閾值，而這些問題皆可以透過人工調整參數的情況得到相當大的改善。

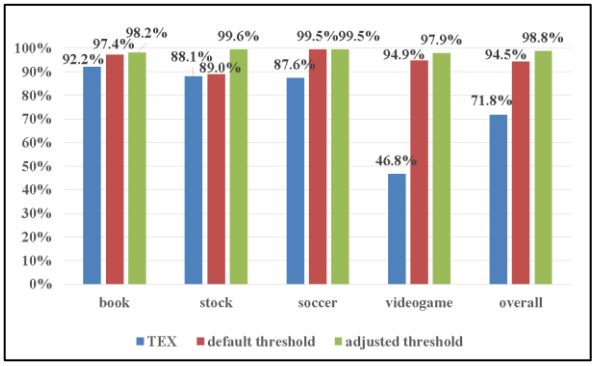

圖 8 整體精確率比較圖

5. 結論

在本篇論文當中，以 T.-S. Chen、M.-C. Chen 及 C.-H. Chang 所提出的頁面層級系統的架構為基底，再往上新增了三塊模組改良其訓練的流程，以提升系統的擷取效能；在本論文的實驗中，首先是檢視三塊模組各自對於系統效能的提升情況，實驗顯示三個模組各司其職，改善了不同的問題，無論單獨使用哪一個，皆能得到效能上的提升。接著則是檢視所有模組對效能的改善情形，實驗顯示不僅不會影響原本就很好的表列網頁，且在詳細網頁中，準確率(Precision)提升了 33.08%、召回率(Recall)提升 32.4%，在整體效能比較中，改善後的系統得到了最高的召回率。最後用 weir 的資料集，去做精確率的比較，實驗顯示改良後的系統效能比 TEX 還要高出許多，預設的模組參數值的整體的效能比 TEX 還要高 22.7%；若是再以人工調整模組參數，整體精確率可再向上提升至 98.8%，整體精確率比 TEX 還要高 27%。

參考網址

實驗用的資料集、實驗結果及 DEMO 皆提供在此 http://sites.google.com/site/nculab/plde

參考文獻

[1] A. Arasu, H. Garcia-Molina, "Extracting structured data from Web pages", presented at the Proceedings of the 2003 ACM SIGMOD international conference on Management of data, San Diego, California, 2003.

[2] R. Baumgartner, S. Flesca , G. Gottlob, "Visual Web Information Extraction with Lixto", 27th International Conference on Very Large Data Bases, 2001.

[3] M. Bronzi, V. Crescenzi, P. Merialdo, P. Papotti, "Extraction and Integration of Partially Overlapping Web Sources", 39th International Conference on Very Large Data Bases, 2013.

[4] C.-H. Chang, M. Kayed, M. Ramzy Girgis, K. Shaalan, "A Survey of Web Information Extraction Systems", Knowledge and Data Engineering, IEEE Transactions on, Vol 18(10), pp.1411-1428, 2006.

[5] C.-H. Chang, S.-C. Lui, "IEPAD: information extraction based on pattern discovery", presented at the Proceedings of the 10th international conference on World Wide Web, Hong Kong, Hong Kong, 2001.

[6] C.-H. Chang, Y.-L. Lin, K.-C. Lin, and M. Kayed, "Page-Level Wrapper Verification for Unsupervised Web Data Extraction", in Web Information Systems Engineering, 2013.

[7] M.-C. Chen, T.-S. Chen, C.-H, Chang, "應用路徑資訊輔助樣板探勘於網頁層級之資料擷取研究", Conference on Technologies and Applications of Artificial Intelligencester, 2013.

[8] T.-S. Chen, M.-C, Chen, C.-H, Chang, "基於頁面層級之快速網頁資料擷取與網要驗證", Conference on Technologies and Applications of Artificial Intelligencester, 2014.

[9] V. Crescenzi, G. Mecca, and P. Merialdo, "RoadRunner: Towards Automatic Data Extraction from Large Web Sites", presented at the Proceedings of the 27th International Conference on Very Large Data Bases, 2001.

[10] K. Kayed, C.-H. Chang, "FiVaTech: Page-Level Web Data Extraction from Template Pages" , IEEE Transactions on Knowledge and Data Engineering, vol. 22, pp. 249-263, 2010.

[11] A.H.F. Laender, B.A. Ribeiro-Neto, A.S. da Silva, J. S. Teixeira, "A Brief Survey of Web Data Extraction Tools", SIGMOD, Vol. 31, pp. 84-93, 2002.

[12] S. Lingam, S. Elbaum, "Supporting End-Users in the Creation of Dependable Web Clips", World Wide Web, 2007.

[13] H. A. Sleiman and R. Corchuelo, "TEX: An efficient and effective unsupervised Web information extractor", Know.-Based Syst., vol. 39, pp. 109-123, 2013.

[14] S. Soderland, "Learning Information Extraction Rules for Semi-Structured and Free Text", Mach. Learn., Vol 34(1-3), pp. 233-272, 1999

[15] J. Wong, J. I. Hong, "Making Mashups with Marmite: Towards End-User Programming for the Web", CHI, 2007.