XML-based Multimedia Data Embedding System for Content-aware Applications

江季翰 (Jiang Ji-Hang)　　洪煒哲(Hung Wei Che)　張見興(Chang Chien Hsing)

jhjiang@nfu.edu.tw　　darkdio2002@gmail.com

國立虎尾科技大學 資訊工程系

摘要

現今這樣資訊爆炸的時代，如何從網路上精確的搜索到所需的資料將成為一個重要的課題，但是現有的網路搜尋引擎在搜尋時，以現有的檔名或圖形特徵搜索方法搜尋圖片的精確度往往很低，如果能針對多媒體檔案作內容描述並且嵌入其中，在搜尋多媒體檔案時直接針對描述資訊作搜尋，將能大幅提高搜尋準確度，這就是一種內容感知的應用。本文把描述資訊以 W3C[12]（World Wide Web Consortium）所制定的 XML[11]文件格式嵌入 JPEG 圖片中，在個人電腦上開發出以此為應用的相簿系統，讓使用者能以描述資訊的內容作為搜尋條件找尋其所需之圖片。在系統中使用 in-memory 壓縮技術，以減少 XML 文件 in-memory 時記憶體之耗用，並且使用雜湊演算法增加搜尋的效率。

Abstract

In the Information age, how to precise get the desirous of data in Internet will be an important question, but when we search multimedia data use internet search engine in existence, the accuracy of result that is searched by file name or figure's feature algorithm in existence is usually low, if we can insert a content describe that is aimed at a multimedia file into the multimedia file, when we search a multimedia file, we can aim at the content describe that is insert into multimedia file, the accuracy of search result will better upgrade, the method is a type of content aware. This paper insert describe information that is XML document form formulated by W3C into JPEG figure, and develop an album system that can use the describe information at PC, let user can use describe information to find desirous of figure. The System use in-memory compact technology to reduce memory depletion when XML document in-memory, and use hash algorithm to increase search efficiency.

第壹章　前言

近年來，數位資料在網路上已隨手可得，如何快速的在眾多資料之中精確搜尋到想要的東西必定是未來的重點之一，然而現今對於多媒體資料的檢索還無法真正達到精確搜尋的要求。以現有的檔案名稱和圖形特徵搜尋方式搜索時，現在的圖形特徵搜索所用的的演算法和技術並沒有辦法精確判斷出圖片中的特徵，而如果使用檔案名稱作查詢索引時，檔案名稱往往無法完整描述圖片內容。上述的原因造成多媒體資料查詢的準確度不高，往往會搜索到許多不相干的資料。如果能做出多媒體檔案的內容描述並嵌入多媒體檔案之中，在搜尋多媒體檔案時直接針對描述資訊作搜尋，將能大幅提高搜尋準確度，這就是一種內容感知技術的應用。

一般人常會利用資料庫來建立圖片的索引以增加搜尋的精確度，但這種方式是使用外部文件來記錄多媒體的描述資訊，因此複製時需要將描述資訊連帶複製，否則將會失去其描述資訊，而無法使用描述資訊搜索。資料隱藏和浮水印的技術常會被用來在多媒體檔案中藏入資訊以作為數位簽章，這些技術雖然目前已經相當的發達，但卻相當不適合使用來嵌入描述資訊，因為資料嵌入需要經過編碼步驟，取出時也需要經過解碼的手續，這樣的方法除了耗時以外，被藏入資訊的多媒體資料也有可能因此而失真，因此較好的方式是在原有的多媒體檔案結構標準下，尋求擴充的可能。

目前已經有一些應用是直接在多媒體檔案原有的架構標準下藏入描述資訊，這樣一來，不但可以解決在外部儲存描述文件所產生的問題，而且在做查詢分類和整理時都可以直接從多媒體檔案中

取出這些描述資訊來使用，這樣的操作方式不僅是相當直覺化，而且也相當的方便，但如果將描述資訊的格式直接制定在多媒體檔案的結構裡，將會造成格式擴充不易的問題，而無法符合所有使用者的需求，但如果能先利用一種開放且容易擴充的格式來描述這些多媒體資料的內容，再將其嵌入多媒體檔案之中，不但解決了多媒體資料本身缺乏描述資訊的問題，也讓描述資訊具有擴充格式的能力。

XML[11]是由 W3C[12]所制定的資料交換格式，XML 有著能夠擴充並適應許多不同需求的優點，目前已經廣泛的被使用在各種領域，以作為資料交換的格式或用來制定文件描述格式，人們可以依自己所需選擇適合的描述資訊格式，然後自己擴充需要的欄位，再將這些訂好的描述資訊嵌入多媒體檔案中，這樣一來將可以達到客製化的需求。

XML 在處理時，常會使用 DOM[10]處理方法，但在使用 DOM 時必須先建立 DOMTree，所佔用的空間會比原本的 XML 文件多上許多，所以在系統中會先將 XML 文件先經過 in-memory 處理，再對 XML 文件作存取的動作。

本文將使用 XML 格式編寫之描述資訊藏入 JPEG 圖片，並且開發出一個能夠使用此描述資訊的相簿系統，此相簿系統在搜索時必須能使用 JPEG 圖片中內含的描述資訊做出精確的查詢。次要目標則是在 XML 文件處理時，使用 in-memory 壓縮技術處理 DOM 儲存結構，以節省記憶體空間上的耗用。

第貳章　文獻探討

第一節　圖片格式資料

1. JFIF（JPEG File Interchange Format）

JFIF[6]是由 JPEG 小組（Independent JPEG Group）所建立的額外標準 JFIF（JPEG File Interchange Format，JPEG 檔案交換格式），JFIF 會詳細的說明如何從一個 JPEG 串流產生出一個適合電腦儲存和傳輸的檔案，表 1 為在 EXIF 格式中使用的 JPEG Marker Segments。

表 1：EXIF 格式中使用的 JPEG Marker Segments(資料來源：EXIF 規格書[5])

	Marker Name	Marker Code
SOI	Start of Image	FFD8.H
APP1	Application Segment 1	FFE1.H
APP2	Application Segment 2	FFE2.H
DQT	Define Quantization Table	FFDB.H
DHT	Define Huffman Table	FFC4.H
DRO	Define Restart Interoperability	FFDD.H
SOR	Start of frame	FFC0.H
SOS	Start of Scan	FFDA.H
EOI	End of Image	FFD9.H

JPEG 是一種針對相片影像而廣泛使用的一種失真壓縮標準方法，JPEG 規格書只有描述如何將一個影像轉換為位元組的數據串流，而沒有說明這些位元組如何在任何特定的儲存媒體上被封存起來。所以當有人稱呼一個檔案為『JPEG 檔案』時，其實大都是在說一個以 JFIF 格式封裝的 JPEG 壓縮影像。

在 JPEG 規格書中已經有定義出 16 個擴充欄位 Application Marker 以提供各種擴充應用，分別為 0xFFE0～0xFFEF，每一個擴充欄位有 64KB 的空間可以使用。JPEG 檔案中有一系列“0xFF??”的字串，被稱為 Marker，用來標記 JPEG 檔的分段（Segment）。“0xFFD8”表示圖片的開始，“0xFFD9”表示圖片結束區段，這兩個 Marker 後面沒有資訊，而其他 Marker 後面都會跟著一些字串或資料。JFIF 所用的擴充欄位是 APP0 0xFFE0，而 EXIF 所用的擴充欄位則是 APP1 0xFFE1 及 APP2 0xFFE2。

第二節　XML 資料

XML 是由國際標準組織（International Standard Organization, ISO）在西元 1986 年所公佈的標準通用標示語言（Standard Generalized Markup Language)所衍生而來。XML 的相關技術如圖 1 所示，在本專題中主要使用的相關技術有 XML Schema 和 DOM

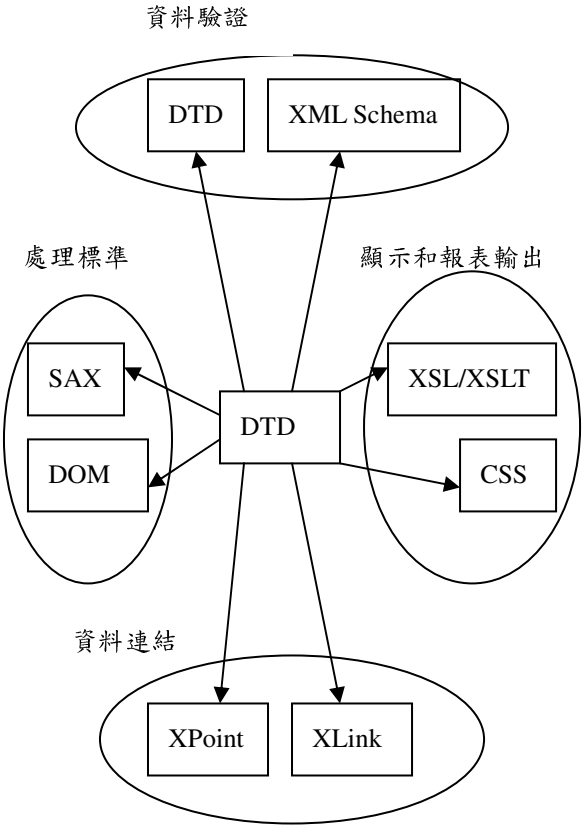


圖 1：XML 技術（參考資料：XML 網頁製作徹底研究[3])

1. XML (eXtensible Markup Language)

1998 年 2 月，美國 W3C[12]組織正式公佈了 XML 1.0 的語法。XML 具有 SGML 的延展性、文件自我描述特性，以及強大的文件結構化功能，但是卻沒有 SGML 的龐大複雜導致不易普及化的缺點，並且 XML 具有文件自我描述的能力，所以 XML 也是一種用來定義其他語言的語法系統，因此 XML 可以應用在各種特定應用的領域範圍，用來描述及規範交換文件之標準格式。

2. XML Schema

XML Schema[9]的檔案描述能將 XML 文件的格式定義成一定型式，其作用在於定義及規範特定 XML 文件內容的架構。由於在交換文件的過程中，資料內容需要相當高的準確性和執行效率，因此這些資料的內容和結構必須統一且嚴謹，而 XML Schema 在 XML 的作用就是制定這些特殊用途的 XML 文件格式。

3. DOM（Document Object Model）

DOM[10]是一種跨平台及跨語言的 XML 應用程式介面(API)，它使得 XML 文件的內容可以動態的應用程式和描述語言（Script Language）存取、更新和架構。

DOM 剖析器（parser）對 XML 文件分析之後，將整個 XML 文件以一棵 DOM Tree 的形式存放在記憶體中，且 DOM 剖析器的 Tree 結構概念與 XML 文件的結構相同。DOM 提供應用程式可以隨時對 DOM Tree 中的任何一個部分進行走訪與操作，應用程式也可以通過 DOM Tree 對 XML 文件進行隨機走訪，這樣可以就達到任意地走訪或操作整個 XML 文件中的內容的功能，但因為 DOM 剖析器把整個 XML 文件轉化成 DOM Tree 放在記憶體中，因此，當 XML 文件較大或結構較複雜時，對記憶體的需求就較高，而且對於結構較為複雜的樹進行搜尋也是一項耗時的操作。

第三節　In-memory 處理方法

字典式壓縮法是將文件或檔案中出現過的資料記錄在字典（Dictionary）中，原本放置資料的地方則變成放置取得資料的參考，之後的資料如果已經有出現在字典內，直接放參照即可，不必在字典中加入新的資料，用在有大量重複資料的檔案時，將可節省大量的空間。

Neumuller 和 N.Wilson 在 2002 年時，提出了一個使用字典式壓縮法來對 XML 文件作 in-memory 的方法[1]，使用的 parse 方式是以 DOM 的處理方法為基礎，這個方法稱為 DDOM[1][2](Dictionary-based DOM)。此種方式在與其他使用 DOM 處理方法為基礎的剖析方式比較之下，可以發現其使用的記憶體改善不少；以另外兩種比較有名的方法 Xerces 和 Crimson 作比較，其使用記憶體比較如表 2 所示。

表 2：DOM 處理方法效率比較（資料來源：Compact In-Memory Representation of XML Data）

XML file	*DDOM(rel.)*	*Xerces(rel.)*	*Crimson(rel.)*
21.4KB	92.0KB(1.0)	312.8KB(3.4)	195.0KB(2.1)
213.3KB	461.1KB(1.0)	1602.4KB(3.5)	19161.3KB(4.2)
2.1MB	3.2MB(1.0)	14.9MB(4.6)	19.3MB(6.0)
20.7MB	25.2MB(1.0)	141.7MB(5.6)	191.0MB(7.6)

在表 2 中所測出的各數據大小為這3種方法在實做出 DOM 的處理方法時所耗用的記憶體，其中以 DDOM 的記憶體使用量作比較，可以發現使用記憶體的確有明顯減少，而且 XML 檔案變的更大時，其使用的記憶體差距倍數也會增加。

第四節　最小完美雜湊（Minimal Perfect Hashing）

雜湊搜尋法的原理是儘量減少搜尋範圍到只有一個，而其資料搜尋是透過雜湊表來執行搜尋，所以雜湊搜尋法最主要的工作是使用雜湊函數建立「雜湊表」(Hashing Tables)雜湊函數是一種數學運算，其目的是減少資料範圍，將搜尋鍵值轉換成索引位置。不過任何雜湊函數都不能保證鍵值在執行運算後得到索引位置是唯一或還有索引位置可以放。如果遇到將數個鍵值轉換成同一個索引位置的情況就是發生碰撞（Collisions），如果一個識別字經過雜湊函數運算後，所對應的存放位置已經滿了則稱之為溢位。最小完美雜湊就是指沒有碰撞(collision)和溢位（overflow)發生，而且做出來的雜湊表大小和原本的資料個數一樣的雜湊法。

第參章　研究方法

第一節　DDOM

Neumuller 和 N.Wilson 所提出的方法是將一份完整的 XML 文件以字典式壓縮法分別用陣列和鍊結串列來儲存其原本文件結構和資料，而其存放文件結構的陣列會存放對應是哪種類別的資料，並且找出是屬於那個元素的 domain，以這兩個作為參照去存放對應種類資料的 Dictionary 中取得所需資料。

為了證實 DDOM 的使用記憶體效率，選擇了現實生活中的七個 XML 例子作為測試用途，七個 XML 的測試數據如表 3 和表 4 所示，且定義變數 P=原 XML 檔案大小/處理過後使用記憶體大小，於表 3 和表 4 一併列出使用 DOM 和 DDOM 所使用的記憶體和原本的檔案大小的比例：

表 3：七個 XML 使用例子

編號	檔案種類	檔案大小
1	對話記錄	27KB
2	資料夾階層	108KB
3	畫面配置檔	234KB
4	系統配置檔	293KB
5	系統配置檔	855KB
6	系統配置檔	1148KB
7	函式說明檔	1.9MB

表 4：DOM 和 DDOM 使用範例

編號	DOM (P)	DDOM (P)
1	195KB(7.2)	136.424KB(5.1)
2	877KB(8.1)	331.696KB(3.1)
3	1843KB(7.9)	354.752KB(1.5)
4	1823KB(6.2)	758.208KB(2.6)
5	6162KB(7.2)	2591.088KB(3.0)
6	7888KB(6.9)	2655.432KB(2.3)
7	8454KB(4.4)	4144.224KB(2.2)

以下即為 DDOM 演算法。

輸入：

一份格式良好(Well form)的 XML 文件。

步驟 1. 讀入 XML 文件，於 Type 陣列放入 START_DOCUMENT 的 Type 參照。

步驟 2. 以 SAX Parser 分析文件，讀出並且判斷出現在遇到的事件種類。

步驟 3. 檢查節點種類，如果為 Element 節點就到步驟 4 的『Element 節點和 Attribute 節點處理』，如果為 PCDATA 節點就到步驟 5 的

PCDATA 節點處理。

步驟 4. Element 節點和 Attribute 節點處理，之後到步驟 6。

步驟 5. PCDATA 節點處理。之後到步驟 6。

步驟 6. 判斷文件是否已經結束，是的話就到就到步驟 7，不是的話就回到步驟 2。

步驟 7. 於 Type 陣列中放入 END_DOCUMENT 的 Type 參照。

輸出：

記錄檔案結構的 Type 陣列，包含了節點種類和資料索引。

記錄 Element 資料的 Element 的 Dictionary。

記錄 Attribute 資料的各自對應包含的 Element 的 Attribute 的 Dictionary。

記錄 PCDATA 資料的各自對應包含的 Element 的 PCDATA 的 Dictionary。

圖 2 即為 DDOM 演算法流程圖。

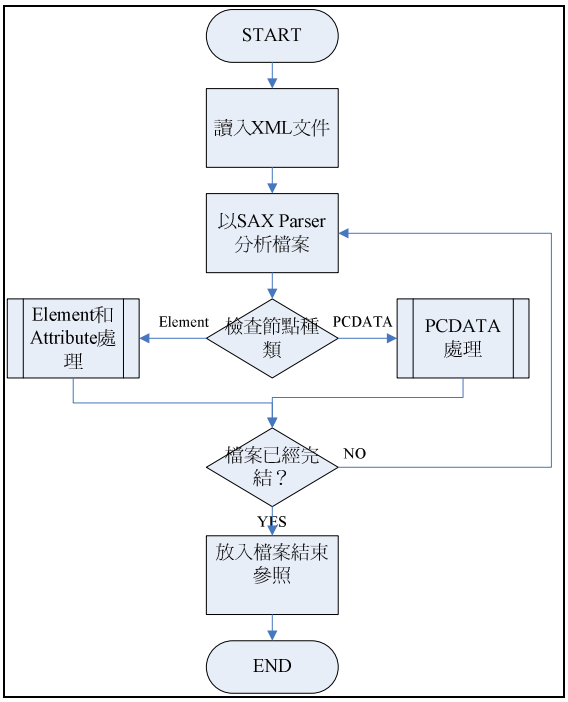


圖 2：DDOM 演算法流程圖

以下為 Element 節點的處理過程：

步驟 1. 讀取 element 名稱。

步驟 2. 檢查 Element Dictionary 中是否有和讀取的 element 名稱相同的資料？如果有的話到步驟 3，沒有的話到步驟 4。

步驟 3. Element 名稱加入目前 Element Dictionary 的末端，並在 TYPE 陣列中放入 Type 參照和 Data 索引（在 Element Dictionary 的位置），然後到步驟 5。

步驟 4. 檢查是之前為 element 種類的 Type 參照中是否有對應目前讀取到 element 名稱的 START_ELEMENT 參照?如果沒有的話到步驟 5，有的話到步驟 6。

步驟 5 定義為 START_ELEMENT，然後到步驟 7。

步驟 6. 定義為 END-ELEMENT，然後到步驟 7。

步驟 7. 於 TYPE 陣列中放入 type 參照和 Data 索引。

步驟 8. 取得此 Element Node 的 Attribute Node 個數。

步驟 9. Attribute 節點處理。

圖 3 即為 Element 節點處理流程圖：

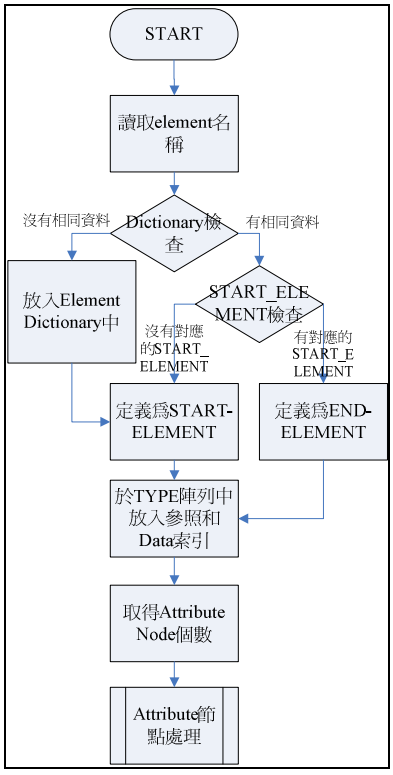


圖 3：Element 處理過程

以下為 Attribute 節點的處理過程：

步驟 1. 判斷是否有 Attribute 節點要處理？有的話到步驟 2，沒有的話結束處理。

步驟 2. 取得 attribute 的名稱。

步驟 3. 檢查是否有存在對應包含的 Element 名稱的 ATTR Dictionary ?沒有的話到步驟 4，有的話到步驟 5。

步驟 4. 建立一個對應此 element 名稱的 ATTR 陣列，接著到步驟 6。

步驟 5. 檢查是否有對應名稱的 START-ATTR ?沒有到步驟 7，有到步驟 8。

步驟 6. 將 attribute 名稱放到對應的 ATTR 陣列中。

步驟 7. 定義為 START-ATTR 並於 TYPE 陣列中放入參照，接著到步驟 9。

步驟 8. 定義為 END-ATTR 並於 TYPE 陣列中放入參照，接著到步驟 9。

步驟 9. 取得 attribute 的值。

步驟 10.attribute 值對應的是 PCDATA Dictionary，所以要檢查對應此 element 名稱的 PCDATA 陣列存在?沒有則到步驟 11，有的話到步驟 12。

步驟 11.建立對應此 element 名稱的 PCDATA 陣列。

步驟 12.檢查此 PCDATA 陣列中是否有相同的 PCDATA ?沒有的話到步驟 13，有的話到步驟 14。

步驟 13.將資料放入此 PCDATA 陣列中。

步驟 14.將此 PCDATA 參照放入 TYPE 陣列中，接著回到步驟 1。

圖 4 即為 Attribute 節點的處理流程圖：

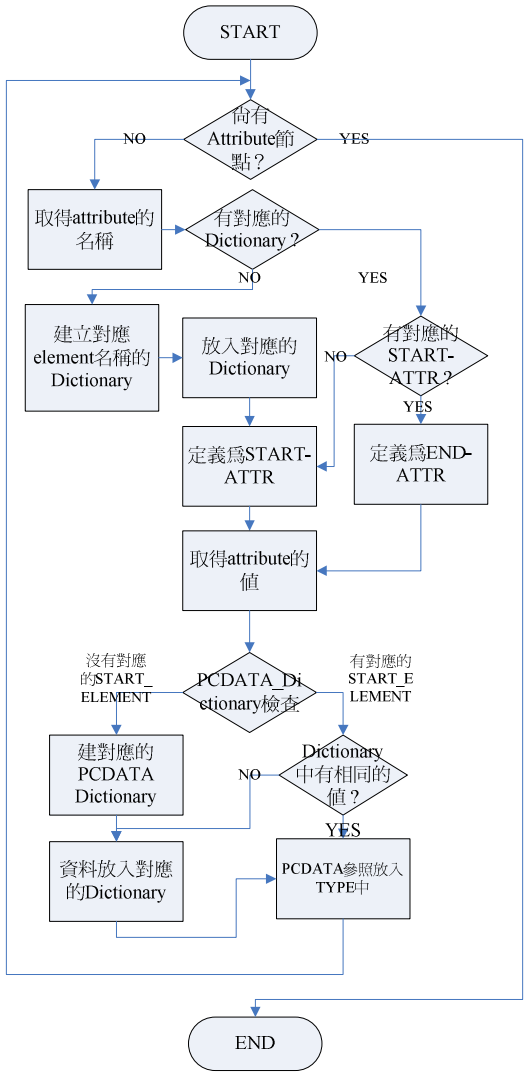


圖 4：Attribute 處理過程

以下為 PCDATA 節點的處理過程：

步驟 1. 讀取 PCDATA 值。

步驟 2. 檢查是否有對應到包含的 element 名稱的 PCDATA 陣列存在 ?沒有的話到步驟 3，有的話到步驟 4。

步驟 3. 建立對應此 element 名稱的 PCDATA 陣列，接著到步驟 5。

步驟 4. 檢查此對應的 PCDATA 陣列中是否有相同的 PCDATA ?沒有的話到步驟 5，有的話到步驟 6。

步驟 5. 將資料放入此 PCDATA 陣列中，之後結束 PCDATA 的處理。

步驟 6. 將此 PCDATA 參照放入 TYPE 陣列中，之後結束 PCDATA 的處理。

圖 5 即為 PCDATA 的處理流程圖。

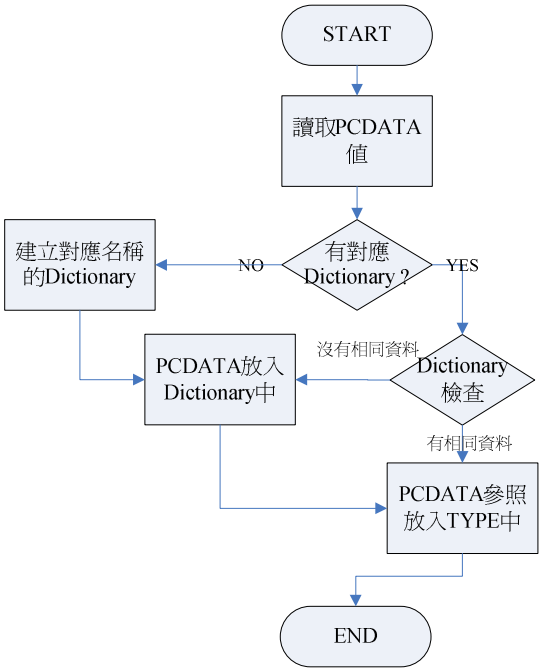


圖 5：PCDATA 處理流程

第二節　最小完美雜湊

Zbigniew J.Ceach、George Havas 和 Bohdan S.Majewski 於 1992 年提出了一個利用圖論(Graph Theory）的概念來實現最小完美雜湊的演算法。

I Perfect 的實現

假設 $W=\{w_1, w_2, \ldots, w_m\}$是 m 個關鍵字(Keyword)所組成的集合，將這 m 個關鍵字視作圖論中的邊，然後做出這 m 個邊的相關頂點，完成的 G = (V, E)如果是 acyclic（非週期循環），則不會有衝突的發生。

II Minimal 的實現

考慮一個 graph G=(V,E)　|E|=m　|V|=n，找出函式 g：V → [0,m-1]和 h：E → [0,m-1]，這也就是說要找出函式：h(e=(u,v)∈E) = (g(u)+g(v)) mod m，其中 u 和 v 是邊 e 的兩邊頂點。如果能找出這兩個函式，就代表 E（資料）和 V（雜湊使用資料）的長度是可以相等的。

III Ceach 等人的演算法

在使用此演算法時會經過以下三個步驟，在此假設 $W=\{w_1, w_2, \ldots, w_m\}$是 m 個關鍵字(Keyword)所組成的集合：

1. Mapping：

   將 m 筆資料轉換成圖 G = (V, E)

   V={0,.....,n-1}　E={$(f_1(w), f_2(w)):w \in W$}

   ，E 的兩端頂點則是使用下列的 f_1 和 f_2 函式轉換

   $$f_1(w)=\left(\sum_{i=1}^{|w|} T_1(i, w[i])\right) \bmod n$$

   $$f_2(w)=\left(\sum_{i=1}^{|w|} T_2(i, w[i])\right) \bmod n$$

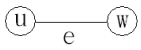
圖 6：頂點轉換的函式（參考來源：An optimal algorithm for generating minimal perfect hash functions[3]）

其中的 T_1 和 T_2 是隨機亂數創造的表格。

2. Assignment：

   將 mapping 步驟中創建的圖 G 使用下列函式轉換成用來計算 Hash 計算用的表格 g：

   g(w) = (h(e) - g(u)) mod m，e = (u, w)



3. Hash：

   使用 mapping 和 assignment 中已經做出的 T_1、T_2和 g 三個表格，以下列函數做出 hash 的計算：h(w) = (g(f_1 (w))+ g(f_2 (w))) mod m

IV 範例說明

以 12 個月份的英文縮寫作例子，因為有 12 筆資料，所以 Hash 的表格大小 m = 12，而作 Hash 計算用的 g 表格大小為 m 個邊有的最多頂點大小 n = 2½*m=2½*12=25。12 月份的英文縮寫中第二和三個字母的組合不會重複，所以作為使用資料。

1. Mapping：

   首先亂數創造出 T_1 和 T_2 表格。

T_1

	a	b	c	e	g	l	n	o	p	r	t	u	v	y
2	11		1	13				21	17			1		
3		9	21		13	5	19		20	1	0		3	12

T_2

	a	b	c	e	g	l	n	o	p	r	t	u	v	y
2	5		2	21				24	8			12		
3		9	23		5	2	7		12	17	2		11	8

圖 7：T_1 和 T_2 表格創造（資料來源：An optimal algorithm for generating minimal perfect hash functions[3]）

計算是否有造成 cycle：

f_1 (jan)=(T_1 (2,a)+ T_1 (3,n))mod 25

=(11+19)mod 25=5

f_2 (jan)=(T_2(2,a)+ T_2 (3,n))mod 25

=(5+7)mod 25=12

f_1 (feb)=22，f_2 (feb)=5

f_1 (mar)=12 ，f_2 (mar)=22

此 T_1 和 T_2 表格會造成如圖 8 的 cycle 狀態，所以必須重新創建 T_1 和 T_2

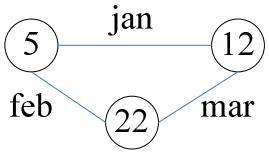


圖 8：cycle 狀態

重新創建的 T_1 和 T_2 如圖 9 所示：

T_1

	a	b	c	e	g	l	n	o	p	r	t	u	v	y
2	19		3	14				7	20			24		
3		11	21		15	14	10		3	2	17		1	15

T_2

	a	b	c	e	g	l	n	o	p	r	t	u	v	y
2	3		13	7				11	21			22		
3		10	12		19	3	10		2	8	1		24	15

圖 9::T_1 和 T_2 表格創造(2)(資料來源:An optimal algorithm for generating minimal perfect hash functions[3])

圖 9 所 T_1 和 T_2 表格經過 f_1 和 f_2 函式換算過後如所示不會造成 cycle，所以創建成功

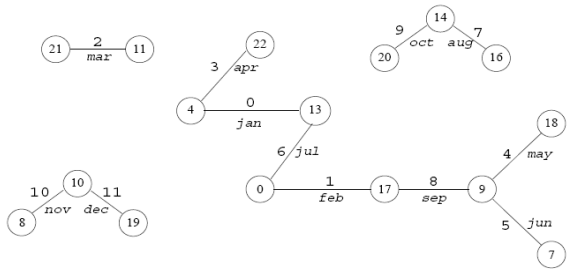


圖 10:創建成功的 acycle 圖(資料來源:An optimal algorithm for generating minimal perfect hash functions[3])

2. Assignment：

找出 graph 中頂點值為 0 的作為起點，然後開始使用 g(w) = (h(e) - g(u)) mod m 的函式計算。

g(17)=(1-g(0)) mod 12 = 1

g(9)=(8-g(17)) mod 12 = 7

g(18)=(4-g(9)) mod 12 = 9…..

最後建立出來的表格 g 如圖 11 所示。

g

0	4	7	8	9	10	11	13	14	16	17	18	19	20	21	22
0	6	10	0	7	10	0	6	0	7	1	9	1	9	2	9

圖 11：建立出來的表格 g (資料來源：An optimal algorithm for generating minimal perfect hash functions[3])

3. Hash 的使用

要作 Hash 計算出資料對應的位置時，要使用函式 h(w) = (g(f_1 (w))+ g(f_2(w))) mod m 作計算

ex:計算 nov 的的位置

f_1 (nov)=(7+1)mod 25 = 8

f_2 (nov)=(11+24)mod 25 = 10

(g(8)+g(10))mod 12 =(0+10) mod 12=10

第肆章 系統介紹

第一節 系統架構

本文的主要目的為內嵌資訊於多媒體資料中，為了提供內嵌描述資訊（metadata）的相關應用，本文會設計出一個相簿系統以展示成果，此系統名稱為資料內嵌應用系統（Data Embedding Application System，DEAS），系統架構如圖 12 所示。使用者可以使用 DEAS 來管理圖片的 metadata 描述資訊，並且以 metadata 作為搜尋的依據。

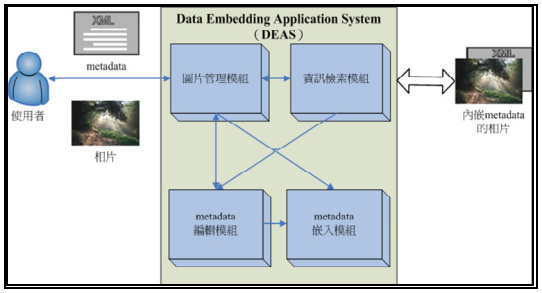


圖 12：系統架構圖

I 圖片管理模組

圖片管理模組為本系統的操作介面，目的為協助使用者分類、管理及瀏覽圖片和其描述資訊。圖 13 為圖片管理模組之使用案例圖，所有系統功能皆由此模組與其他模組互動達成。使用者能用類似檔案總管的方式來管理圖片，並且以幻燈片的方式顯示在此資料夾中所有圖片的預覽圖。

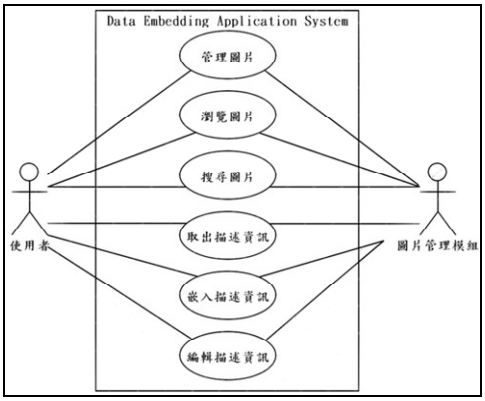


圖 13：圖片管理模組使用範例

II metadata 嵌入模組

metadata 嵌入模組主要的功能是將 metadata 和 metadata 的 schema 嵌入到圖片之中，為本系統的核心模組。其使用案例如圖 14 所示，使用者如果選擇自訂 Schema，則此模組會將使用者自訂的 Schema 與其描述資訊一同嵌入至 JPEG 圖片之中。

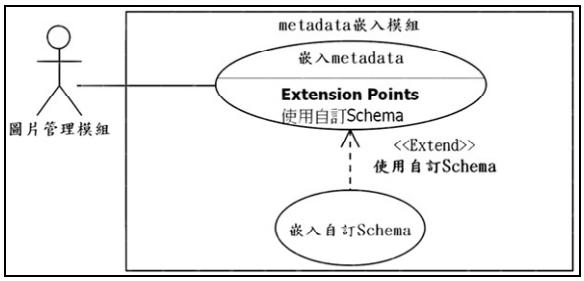


圖 14：metadata 嵌入模組

III資訊檢索模組

資訊檢索模組可以將圖片內嵌的 metadata 讀出交給其他模組使用，其使用案例如圖 15 所示。

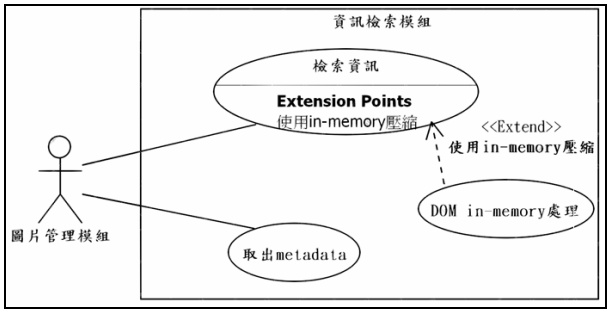


圖 15：圖片管理模組使用範例

在查詢時，資料檢索模組會透過檔案系統對有內嵌 metadata 的圖片作查詢，最後將查詢的結果回傳給使用者。此外，本模組也提供使用者使用經過 in-memory 壓縮的 DOM 處理方法，以減少記憶體的耗用。

IV metadata 編輯模組

metadata 編輯模組讓使用者可以編輯圖片的 metadata 描述資訊，如果圖片中已經含有 metadata，則此模組會先透過資訊檢索模組取出已嵌入的 metadata，再編輯該 metadata。描述資訊在經過本模組編輯後會透過 metadata 嵌入模組將其直接嵌入其所屬之圖片中。圖 16 為 metadata 編輯模組的使用案例。

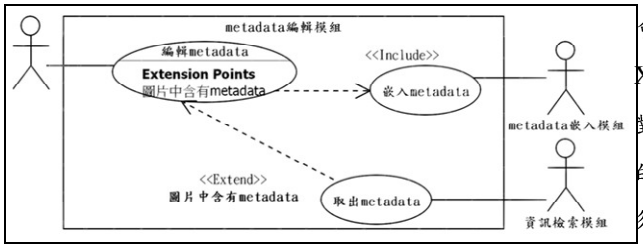


圖 16：metadata 編輯模組使用範例

第二節 系統功能

I 嵌入功能

DEAS 系統的嵌入功能可分以下兩種：兩種方式都是將 metadata 和 definition 直接嵌入 JPEG 格式檔案中，並沒有做 in-memory 的動作。

1. 嵌入已經存在的 XML 描述資料

要嵌入已經存在的 XML 描述資料時，系統會先判斷是否有 definition，如果沒有的話，會直接取得要嵌入的 metadata 所佔空間，放在 JPEG Application Marker 中的 0xFFE4 之後，接著放入代表 metadata 開頭的 MET Marker 和所佔空間大小的數字，後面再放入 metadata 本身；如果有 definition 的話，會分別計算 metadata 和 definition 所佔空間之後，先放入代表 definition 開頭的 DEF Marker 和所佔空間大小的數字，後面放入 definition，接著則是放入 metadata（開頭代表字、所佔空間大小的數字和 metadata 本身）。

2. 自行設計描述資訊並且嵌入

此方式是在嵌入時才自行設計要嵌入的資訊項目和內容，系統會提供使用者輸入所需資料。首先輸入所要的項目數量，DEAS 系統會根據前面輸入的資料產生特定 XML 格式的 metadata 和其對應的 definition，接著計算出 metadata 和 definition 大小，之後嵌入 metadata 和 definition 於檔案的 APR4 之處，嵌入的方式和前面的「嵌入已經存在的 XML 描述資料」一樣。

II 搜尋功能

在搜索資料時，系統會根據使用者輸入的描述項目名稱和描述項目內容來作搜尋，也可以搜尋分別符合兩者其中之一的檔案。在搜尋的時候，系統會到 JPEG 格式檔案的 APR4 的位置確認是否有 XML 格式的描述資訊存在，如果有的話，則開始對其中的 metadata 作搜索，分別以描述項目名稱的資料搜索 name 元素標籤中的內容，這個比對必須完全符合；以描述項目內容的資料搜索 strValue 元素標籤或者是 intValue 元素標籤中的內容，這個比對則是部份符合即可，檢查是否有符合條件。

結論

把描述資訊嵌入到多媒體檔案中，將可以提供各種許多內容感知技術的相關應用。如果能在網路搜尋引擎或檔案系統檢索上導入相關的技術，必定能大幅減少不精確的查詢結果。使用可擴充的 XML 文件作為描述資訊的格式則可讓使用者針對需求客製化，除了可以讓描述資訊的內容更加符合使用者需求外，也提供了更加精確的描述，使得搜尋的結果可以更加精確。這些相關的應用必定是未來資料查詢和分類的潮流，因此這種系統整合必定是未來資料查詢系統中不可缺的要素。

第伍章 參考文獻

論文與書籍

[1] Mathias Neumüller, John N. Wilson. Improving XML Processing Using Adapted Data Structures. Web, Web-Services, and Database Systems 2002: 206-220.

[2] Mathias Neumüller: Compact Data Structures for Querying XML. EDBT PhD Workshop 2002: 127-130.

[3] Zbigniew J. Czech, George Havas, Bohdan S. Majewski. An optimal algorithm for generating minimal perfect hash functions. Information Processing Letters, Pages: 257 - 264 , October 1992.

[4] 陳會安，2002，XML 網頁製作徹底研究，旗標出版股份有限公司，原著：

[5] Exif Specification. http://www.Exif.org/Exif2-2.PDF, 2007/1/16.

[6] JFIF Specification.
http://www.w3.org/Graphics/JPEG/itu-t81.pdf,
2007/1/16.

網站：

[7] XML 台灣資訊網.
http://www.xml.org.tw/, 2007/1/16.

[8] EXIF.org. http://exif.org/, 2007/1/16.

[9] W3C, XML Schema.
http://www.w3.org/XML/Schema, 2007/1/16.

[10] W3C, Document Object Model (DOM).
http://www.w3.org/DOM/, 2007/1/16.

[11] W3C, Extensible Markup Language (XML).
http://www.w3.org/XML/, 2007/1/16.

[12] World Wide Web Consortium (W3C).
http://www.w3.org/, 2007/1/16.

[13] SWT サンプル集.
http://amateras.sourceforge.jp/cgi-bin/fswiki/wiki.cgi/swt?page=FrontPage, 2007/1/16.

[14] SWT Tips and Samples.
http://cjasmin.fc2web.com/, 2007/1/16.