網路學習環境下所需的電子文件註記系統[1]

楊亨利
國立政治大學資訊管理學系

江柏寬
國立政治大學資訊管理學系

摘要

傳統畫線或在文件旁寫筆記可幫助個人記錄所學習到的知識與心得，並且使個人得以瞭解自己的學習狀況以改善學習策略。在資訊化的時代，則有賴電子註記。然而有些文件並不允許使用者修改其內容，因此無法直接在上面添加註記，在這種情境之下，常見的作法便是選擇將文件與註記分開儲存，在不修改原始文件的情況下進行註記，並且在瀏覽的時候將兩者暫時結合呈現。這解決了使用者無權編輯文件的問題，但是在文件遭到修改時容易產生註記定位上的混淆，註記很可能會呈現在不正確的位置。本研究提出一個可以支援個人與團體使用的註記系統架構，系統採用註記與文件分開儲存的模式，並且具備了同步與非同步的溝通機制，使用者可與他人即時交換文字資訊並且分享註記畫面。除此之外，本研究也設計了一種註記定位演算法，可以在文章經過修改之後將註記重新正確顯示，並且實作一個雛型系統以驗證其正確性。

關鍵字：註記、註記系統、註記定位、註記定位演算法

[1] 本論文接受國科會計畫NSC96-2416-H-004-015-MY2補助，特此誌謝。

Annotation System under e-learning Environment

Heng-Li Yang
Department of Management Information Systems, National Cheng-Chi University

Po-Kuan Chiang
Department of Management Information Systems, National Cheng-Chi University

Abstract

Traditionally, people take notes to record what they gain from intense study. When people read, they often underline important parts of a document or write notes in the margin. These notes could help they improve learning strategy. In the information age, the handwritten notes would be replaced by digital annotations. However, users might be not allowed to change some documents. In this situation, the common solution would store the original document and annotations separately and then temporarily merge them while users wish to browse the annotated document. This approach might raise the annotation anchoring problem while the original document has been changed later. This study has proposed an annotation system framework to support personal learning and group communication (either synchronous or asynchronous). An anchoring annotation algorithm has been proposed to display annotations correctly while the original document was changed. A prototype has been implemented and its correctness has been verified.

Key words : Annotation, Annotation System, Annotation Position, Anchoring Annotation Algorithm

壹、導論

一般人在閱讀與學習時常常有用螢光筆劃線、在頁邊做摘要等「做筆記」的行為，這樣的做法可以幫助我們記憶、思考以及釐清文章內容。隨著時代的進步，近年來網路的發展越來越迅速，資訊取得與傳遞的模式也變得與以往大不相同，許多學習的資源漸漸從實體書籍紙張轉變為以電子檔案為媒介，甚至有很多有價值的資訊是存放在網路空間裡以網頁的方式呈現。因此在電腦螢幕上閱讀文章瀏覽資訊的機會大增，使用數位科技來輔助學習的趨勢也越來越明顯。此時需要仰賴「電子註記系統」來記錄當時產生的感想及疑惑，並且協助自己思考並集中注意力（Brown & Brown 2004；Cunningham & Knowles 2005）。進一步，除了個人註記外，還有分享註記的需求（Brush 2002；Marshall & Brush 2004）。提供註記輔助、並協助搜尋註記，已成為資訊環境之必要（Agosti & Ferro 2005；Agosti et al. 2007；Qayyum 2008）。甚至，可將註記系統擴大運用，如網站也可藉由分析讀者的註記資料來產生個人化的網頁內容（Fiala & Meisner 2003），教師在學生考卷上批改也可視同註記（Plimmer & Mason 2006）。

學習者在閱讀文獻時會面臨各種不同檔案格式的文件，有的是屬於一般的電子化檔案文件，而有些是屬於網頁格式的資料，這兩大類文件的註記都會遇到一個問題，那就是有些文件並不允許使用者修改其內容，因此無法直接在上面添加註記，或是有些資訊媒介本身並不支援註記呈現的功能。在這種情境下，常見的作法便是選擇將文件與註記分開儲存，在不修改原始文件的情況下進行註記，並且在瀏覽的時候將兩者暫時結合呈現。這解決了使用者無權編輯文件的問題，卻也產生許多的問題尚待解決，採用註記與文件分開儲存的策略在文件遭到修改時容易產生註記定位上的混淆，註記很可能會呈現在不正確的位置或是直接造成註記系統的執行錯誤，這是我們急需解決的問題。

本研究在此背景下思考如何解決註記定位混淆的問題，從註記系統的建置方式開始，提出一個新的註記重定位的演算法，並且實作註記系統雛型，進行驗證。

貳、文獻探討

一、文件註記系統

受限於技術和媒體特性的不同，在電子文件上進行註記與傳統紙張文件有很大差異，一般紙上註記有很高的彈性和豐富性，讀者可以自由的在紙張上任何位置插入任何形式的標記符號，也可以在文件上隨意書寫文字，但是在電子文件上的註記活動則受到了很多限制，通常只能夠使用系統所提供的功能來處理文件。

目前許多的商用軟體提供了附加註記的功能，例如Microsoft Word支援簡單的註記能力，使用者可以自由改變段落的文字格式，如粗體、斜體、底線等，系統也提供了效果類似於使用螢光筆的「醒目提示」功能。另外，Word可以在文章中插入個人的心得評

論，使用者只要自行圈選任何文字範圍，按下「註解」按鈕，就可以在頁邊的註解方塊中編輯文字，註解方塊和使用者所圈選的文字之間會有虛線連接，並且文字上會出現醒目的底色與標記。

另外一個常見的文件瀏覽工具Adobe Acrobat也提供了基本的註記能力，使用者可以在文章中的文字段落上添加反白（底色方塊）、底線、刪除線等醒目標記，此外，使用者也可以將文字心得插入文章中。系統提供了兩種註解模式，在第一種模式中，使用者可以利用滑鼠雙擊已經建立的反白、底線或刪除線等醒目標記，在鄰近的位置會出現可以輸入註解的文字方塊，當滑鼠滑過醒目標記的文字時，便會彈出該註解的作者和內容。另一種模式則是在文件中任何位置按下註解按鈕，便會出現一個小圖示，同時使用者在註解方塊裡輸入文字，當滑鼠滑過小圖示時，該註解便會彈出。

以上兩種商業軟體是非常典型的文書處理軟體，兩者都是將產生的註記儲存於文件檔案內部。Word讓文件擁有者可以自由的對文件內容進行編輯，即使文章已經被添加了註記，系統也能夠正確顯示修改過的內容以及依附其上的註記。而Adobe Acrobat則可限制使用者的編輯權限，讀者不能修改任何的文件內容，只能單純的在文件中添加系統所提供的註記。

二、網頁註記系統

隨著網路學習日漸普遍，因此越來越多的研究著重在網頁上的註記。許多系統讓人們可以很容易的在網頁上添加、閱讀註記，藉此達到非同步的合作（Asynchronous Collaboration）。典型的網頁註記系統設計是將註記資料與網頁分開儲存，在使用者想要瀏覽的時候才將兩者結合，呈現當初註記時的樣貌。也因為註記是單獨儲存的關係，所以即便使用者並沒有權限修改網頁的內容，很多系統仍然可以提供使用者對任何網頁進行註記的功能（Brush 2002）。

根據系統運作的不同，Brush（2002）將網頁註記系統分為下列三種：

1. 伺服器基礎（Server-based）註記系統：此類註記系統是由伺服器來提供註記的能力，經過處理的網頁安置在註記伺服器後，就可以讓瀏覽者進行註記，缺點是只能對放在此類特定伺服器的網頁進行註記，使用上比較不方便。
2. 代理伺服器基礎（Proxy-based）註記系統：此類註記系統則是透過代理伺服器讓使用者能夠自由的對任何網頁進行註記，通常使用者只要使用一般的網頁瀏覽器，透過特定的URL當做媒介讀取網頁，網頁資料會先經過註記代理伺服器，在這個過程中將註記所需要的使用者介面嵌入網頁，如果有之前所創造的註記也會一併加入網頁之中，使用者就可以開始進行註記了，不用額外安裝軟體的特性使得很多系統都採取此種架構。
3. 瀏覽器基礎（Browser-based）註記系統：許多註記系統採用擴充特定瀏覽器的方式來提供對網頁的註記能力，為了在瀏覽器上建立註記的使用介面，使用者必須先安裝一些軟體，然後才能在網頁上添加並且瀏覽註記。

三、註記的儲存方式

註記系統在處理使用者所產生的註記檔案時，通常有兩種不同的儲存策略：與文件共同儲存，或是將註記與文件分開儲存。這兩種策略各有不同的特性，對前者來說，由於註記和文章內容緊密的結合，在文章被修改的時候註記也會跟著改變，使用者能夠同時瀏覽所有的文章內容與附加註記，不至於會發生註記與文章分崩離析的清況。使用這種策略非常方便，但是有個先決條件就是使用者必須具備修改、編輯該文件的權限。在某些情境下使用者並未具備這些權限，例如我們可以瀏覽很多網頁，但是卻不具有這些網頁的編輯權，導致我們無法直接在網頁上加註任何資訊。所以如果將文件與註記分開儲存，便可以避開這個權限的問題，藉由註記系統的幫助，使用者能夠將閱讀時產生的註記儲存在另一個檔案裡，並且在需要的時候和文件一起呈現。

分開儲存註記所產生的註記檔案和文件本身比較起來其檔案大小較為輕巧，所以也很適合分享給別人，但是使用這種策略會面臨一些困難，由於註記與文件分別儲存，文件內容有可能在多種不同的情況下遭到修改，導致原本的註記無法在修改過後的文件中正確呈現，這些註記可能呈現在錯誤的地方，或是變成無法呈現的「孤兒註記」（Orphaning of Annotations）。

目前註記系統有三種常見的策略來避免產生「孤兒註記」（Brush 2002）：

1. 限制文件的修改：部分系統限制使用者不能修改瀏覽的文章，或是假設使用者不會修改這篇文章，如此一來註記可以使用最簡單的方式記錄位置資訊，並產生較為輕巧的註記儲存檔，而且不必面對重定位的問題。
2. 限制註記位置：部分網頁註記系統為了應付網頁的修改行為，只允許在特定的位置加入註記，通常使用者只能註記於文章的指定位置或是文件的某些特徵上，例如段落的結尾或是特定的HTML標籤，如此一來簡化了註記位置與格式的變化，系統或是註記的管理者就較易於處理定位的問題。
3. 嘗試重新定位：有些系統並不限制註記的樣式，在操作上給予使用者極大的自由，並且企圖在文章經過任何修改之後嘗試找出註記的正確位置。

四、註記的重定位

註記重定位系統必須在註記產生的時候就儲存足夠的位置資訊供定位之用，儲存的資訊可分為兩類：儲存被註記的文章內容或儲存註記位置的結構資訊。

（一）使用文件內容定位

Brush（2002）認為只使用註記區域的文字資訊作為定位的依據可以不受限於文章的資料格式。只要文件的檔案類型主要是以文字來表示其內容，這種策略就可以在這份文件上使用。這策略好處是在建置系統的過程中不必在意文件是否具有完善的結構，只要能夠正確讀取其文字內容即可。這可用以建置一個能夠處理多種不同文件格式的註記系統，也不必因文件格式的結構不同而使用不同的定位策略。如此一來，不但可以節省系統的開發和維護成本，也可以讓使用者在操作不同格式文件時有更一致的感覺。CritLink

（Yee 2002）和WebAnn（Brush & Bargeron 2001；Brush et al. 2001；Brush 2002）都是使用這種定位模式。

CritLink系統的做法是儲存整段註記範圍裡的文字，並且記錄劃記範圍的前後文，系統在註記產生的時候就會先提醒使用者必須選擇足夠的範圍，使得被劃記的文字在整篇文章中獨一無二。遇到註記需要重新定位的時候，系統將會利用這段獨特的文字與註記的前後文來辨識新的位置。這種作法執行起來十分簡單，但是卻很容易出錯，如果這些獨特的句子遭到修改，所屬的註記很可能就會找不到適當的位置。即使這些子句並未被修改，如果在文章中其他的位置找到完全相同的句子，系統也有可能將註記放置於錯誤的位置。一般來說系統儲存註記範圍文字的策略又分為兩種子類型，分別為儲存註記範圍中在整篇文章裡只出現一次的獨特子句以及儲存整段文字，這兩種不同的策略各自有其缺點，如果系統定位策略使用的是前者，也就是紀錄註記範圍裡的獨特子句，並未儲存整段文字的話，很有可能會發生獨特子句並未被修改，但是註記範圍裡的其他內容卻被改變的情況發生，如此一來輕則註記的範圍長度被誤判，嚴重的話可能會將應該被刪除的註記誤植在文章中。後者則是容易因為一點點微幅的更動就造成系統無法找到完全相同的文字段落，因而導致註記定位的失敗。

Brush（2002）的系統試圖經由實驗確認使用者對於註記重定位的期待，他們認為利用關鍵字來連結修改過的文章與原文之間的關係較為符合使用者的期望，文中某些特別的字詞具有較為顯著的價值，即使劃記範圍的文字經過修改，系統仍應辨認出這些重要的關鍵字。因此，Brush系統有別於其他系統，採用的是紀錄「關鍵字」的策略，Brush系統決定關鍵字的策略是選擇在文章中「出現頻率」最低的字作為關鍵字，除此之外，系統會另外儲存足夠且適當的註記位置資訊，並且利用這些資訊作為定位的依據。

在文章經過修改之後，Brush系統會依照以下的步驟來判斷註記的新位置。

1. 在文件中尋找關鍵字，找到包含關鍵字的區段之後將其列為「候選錨區」，並且由原候選區盡可能向外尋找其他關鍵字，在合理的長度內將候選區範圍延伸以納入附近的關鍵字（系統設定候選區長度不得超過原註記範圍的兩倍）。由於候選錨區可能不止一個，必須進一步做篩選。藉由比較各關鍵字之間的相對字元距離，和修改之前的原文差距越小，該候選錨區的「信賴分數」就越高。
2. 從候選錨區開始，向前向後尋找和原文相符的註記開頭與結尾，在合理的長度內將候選區範圍延伸以納入找到的開頭與結尾。同樣的，也會計算新找到的開頭與結尾與關鍵字之間的字元距離，越符合原本距離者，分數越高。
3. 比較候選錨區的「位置」和「長度」，和原文越符合者信賴分數越高。
4. 由候選錨區向外尋找附近的前後文資訊，如果找到與原文相符合的前後文，則增加其信賴分數。
5. 統計各候選錨區的信賴分數以決定新的註記位置。

候選區的信賴分數代表的是候選區與原始註記區域的相似程度，越高分的候選區就越可能是正確的註記位置，但是最高分的候選區並不一定是絕對正確的結果，尤其是在所有的候選區信賴分數都不夠高的時候，產生的結果很可能不符合使用者的期待，使用

者可能寧可讓註記消失，也不願意註記呈現在一個錯誤的位置。因此系統需要設置一個門檻值，候選區必須要超過此門檻才能被呈現。

和關鍵句子相比，使用關鍵字作為辨識的策略更有彈性，可以容許文章經過更多的修改，仍然能夠找到註記真正的位置與長度，即使註記範圍本身的文字內容經過修改，只要還有足夠比例的關鍵字留下，還是能夠產生正確的結果。

（二）使用文件結構定位

使用者在安放註記時往往不會很小心，他們在文章中所添加的底線及底色方塊時，不一定將其依附在文章的內容上，反而常是將其位置依附在文章的結構特徵上。因此，有些系統採取和前一節所述的定位策略不同的方式，他們選擇使用註記位置的結構資訊來作為重定位的依據。但是單純使用文章的位置結構可能產生不符合期望的結果，所以有一些系統同時使用辨識文章內容的方式來輔助。

Annotea（Kahan et al. 2001）使用XPointer（XML Pointer Language）來為註記定位，其功用是紀錄註記的結構資訊。XPointer本身屬於W3C的XPath技術的一種延伸性規範。XPointer所提供的文件指向機制可以讓使用者指向XML文件中的某一個區段，例如樹狀結構中的某一段、屬性、或者是文字內容中的某一字元。利用XLink中的延伸性連結，使用者可以對XML文件的內容做更進一步的定址。譬如，使用者可以利用Xpointer尋找某一XML文件第五個元素的第三個子元素的第七個子元素。這種定位策略對未經修改的文章來說十分有用，但是對於不斷修改版本的文件來說，註記很容易失去定位的依據而淪為孤兒註記，甚至是被呈現在錯誤的地方，尤其是在被註記的節點本身被修改的時候。

由於單純依靠某些結構特徵來辨認註記位置如易產生錯誤，Robust Locations（Phelps & Wilensky 2000）則使用多項不同性質的位置資訊幫助註記定位，提供相當大的彈性：

1. 獨一識別碼（Unique Identifier, UID）：在文件中特定代表該物件的獨特名稱。即使文章經過大幅度的修改，UID通常都還是能夠正確指向物件，除非物件本身遭到刪除。要使用UID作為定位的辨識依據需要依靠文件作者對物件UID做適當的設定，如果使用者將註記放置於不具有UID的位置或是僅僅只劃記部分的物件內容的話，系統就無法利用UID作為定位依據。
2. 文件樹遊走（Tree Walk）：由文件結構根節點行至目標內容位置所在葉節點的路徑。由於在由根節點行進至葉節點的遊走過程中會不斷縮小目標範圍，讓註記的結構位置越來越精確，因此在處理部份文章內容被刪除的情境下表現較優秀，即使某些內容經常被修改，經由文件結構仍可指出其位置
3. 脈絡（Context）：指的是註記範圍前方及後方一小段的連續文字內容。對於長度較短的文章來說，使用脈絡定位的效果十分顯著，但如果文章中出現與脈絡重複的文字段落則會造成混淆，這種情況在文章越長的情況下越容易發生，然而即使定位失敗，這些前後文紀錄仍可提供原始註記位置的參考。

五、文獻小結

伺服器基礎的系統只能開啟放置在伺服器端的文件，管理者可以對文件先進行處理以達到功能的最佳化，但由於此系統無法對伺服器以外的網頁進行註記，所以在使用上十分沒有彈性，不適合本研究的需求。代理伺服器基礎系統能夠瀏覽任何網際網路上公開的網頁並且提供註記功能，由於所有的功能以及操作介面都是由代理伺服器所提供，使用者端無須安裝任何程式，只需使用一般的瀏覽器即可開始進行註記活動。但是本研究希望能夠同時具備處理檔案文件以及網頁的能力，而代理伺服器基礎系統並不適合處理儲存於使用者端的檔案文件。相反地，安裝於使用者端的瀏覽器基礎註記系統除了擴充瀏覽器對網頁的註記功能外，也可處理一般檔案文件檔案的註記，因此被採用為本研究的運作模式。

表1列出兩類常見註記重定位的方式策略的優缺點。表2則整理文獻中註記系統重定位相關作法以及優缺點。由於本研究假設使用者沒有權限編輯目標文件，也不會主動向文件擁有者徵取文件的修改權限，因此文件的擁有者自然也無從得知自己的文章正在被瀏覽並加以註記，使用結構作為定位依據並不適當。其次，由於本研究必須處理多種文件檔案格式，而每種文件的結構並不相同，如採用文件結構作為定位策略則必須要針對每種文件各別設計定位策略，不但大大的增加設計的複雜度，在使用上也會造成不一致的感覺。因此本研究主要採用文章內容作為定位依據的策略，不但可以使用單一定位模式處理多種文件格式，產生的結果也可能較為符合使用者的期待。

表1：使用文件結構及文件內容作為定位依據優缺點比較

	使用文件結構	使用文章內容
定位依據	●物件的獨特ID ●結構樹中的路徑 ●圖形的座標軸資訊 ●影像（聲音）的時間或框頁數	●獨特子句 ●儲存整段註記範圍的文字 ●劃記範圍的前後文 ●紀錄「關鍵字」
優點	●可用於非文字內容的文件 ●處理速度快	●不受限於文章的資料格式 ●較易符合使用者期待
缺點	●只能處理特定結構的文件格式 ●文件的擁有者可能在不知情的情況下修改文件結構	●相似與重複的內容容易造成混淆 ●只能處理文字文件

資料來源：本研究

文獻中如WebAnn單純利用文章內容作為辨識依據的註記定位演算法，可在任何以文字為主的檔案類型上運作。但其演算法在註記的定位上十分依賴關鍵字，對註記範圍內的其他文字並不重視，對前後文等註記周圍的內容雖然納入評分，仍然認為其重要性不高。本研究對這些缺失進行修正，提出了一套新的演算法，對註記範圍內的所有內容都相當重視，使用註記全文來進行序列比對求得適當的候選區，並更進一步計算候選區與原文的相似度以進行較精確的評分。至於前後文的部份，本演算法視其為重要的定位依據，由於前後文緊鄰註記，與註記的開頭結尾有密切的關係，因此可以作為調整註記開始、結束的位置的重要依據。另外，由於文章的段落是一段具有整體性的結構，註記

與其所處段落必定有相關性，段落的正確性為評估註記位置可信度的重要依據，因此本研究利用註記與其所處段落之間的關係來幫助定位，我們認為正確的註記必定會位於正確的段落中。而文章段落雖然是結構化資訊的一種，但是卻不屬於特定的檔案格式所專有，而是一種依附文章內容的結構，因此本研究仍保留「可以在任何以文字為主的檔類型上運作」的優點。

表2：文獻中註記系統相關作法以及優缺點

註記系統	註記重定位模式	定位策略類型	優點	缺點
CritLink（Yee 2002）	要求使用者所劃記的文字段落必須獨一無二	文字內容	執行方式較為簡單，容易實行	劃記文字不可遭到修改，否則會造成錯誤的定位結果
WebAnn（Brush 2002）	辨認註記範圍中的關鍵字	文字內容	能容忍較大程度的文章內容遭到修改，且重定位結果較符合使用者期待	關鍵字遭到修改會嚴重影響註記的重定位結果
Annotea（Kahan et al. 2001）	利用XPointer指向XML文件中的某一個區段	文件結構	能夠非常快速的找到註記的正確位置	註記節點或結構遭到修改容易定位錯誤，且只能使用於特定結構之文件
Robust Locations（Phelps & Wilensky 2000）	使用UID、Tree Walk及Context等多項不同性質的位置資訊幫助註記定位	文字內容與文件結構	使用多項不同的位置資訊幫助註記定位，彈性大	必須紀錄大量的定位資訊，且在特定結構之文件中才能發揮最佳效能

資料來源：本研究

參、建議之學習註記系統架構設計

一、 學習註記系統整體介紹

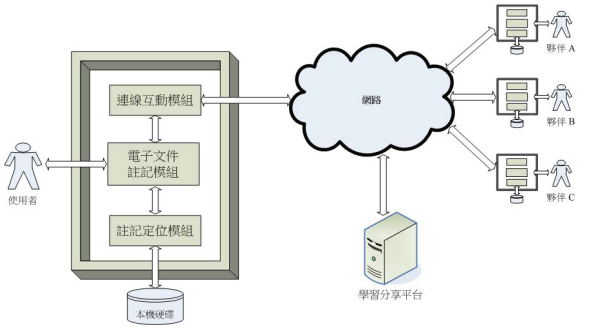

圖1：學習註記系統環境示意圖

本研究所開發之註記系統主要功能是幫助使用者在文件及網頁上建立個人註記，並且提供使用者一個相互溝通的機制。整體之環境示意圖如圖1所示。圖1左邊為註記系統

之核心，說明如下：

1. 「電子文件註記模組」主要的功用是呈現文件的內容，並且依「註記定位模組」所提供註記的定位資訊，將註記呈現於指定位置，讓使用者能夠在畫面中同時瀏覽文章與註記。使用者可依對應文件格式選擇為「檔案文件註記功能」或「網頁註記功能」。兩者均提供使用者各種劃記模式的工具列介面及插入文字註解等功能，使用者也可以利用下方的註解區輸入自己的心得，如圖2。另外，也容許使用者作學習資訊管理，整理多種類型的註記，如可對「疑問」類註記，設計「解答」欄。也提供搜尋功能，如圖3，容許使用者日後將註記的內容與類型列為搜尋條件。

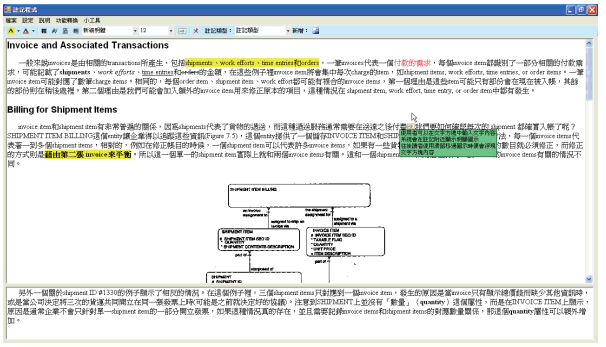

圖2：註記範例圖

圖3：設定搜尋條件

2. 「註記定位模組」乃是依循定位演算法，在產生註記時分析並記錄定位所需的相關資訊，將其與註記內容共同儲存，並在下次呼叫的時候，提供註記重定位，以找到正確的註記位置，其演算法容後說明。
3. 「連線互動模組」主要是提供使用者互動連線的功能，使用者可以藉此和夥伴建立即時通訊的機制以進行討論。使用者在通訊的過程中可以對文件進行註記，無論是底色或是文字方塊，甚至是頁底的註解文字，在添加註記之後按下對話框旁的傳送鈕，對方的畫面上就會出現相同的註記符號，如此一來便可以迅速清晰的與對方進行文章的討論，如圖4。另外連線互動模組也能夠幫助使用者連接伺服器上的學習分享平台，方便註記與文件的上傳、下載以及資訊的流覽。

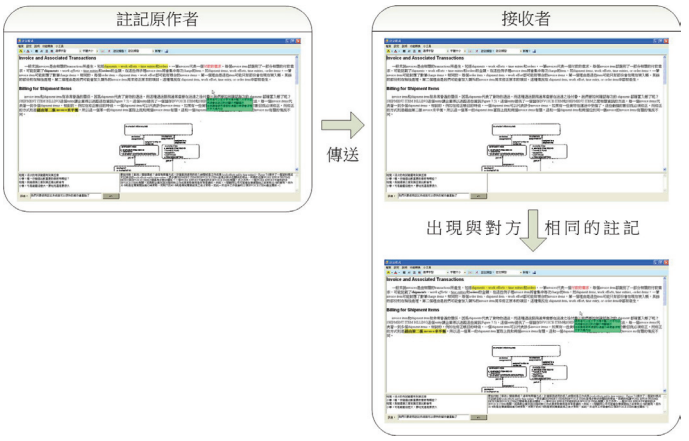

圖4：傳送註記範例圖

二、註記定位與重定位演算法

本研究之註記定位與重定位演算法分為兩個階段，首先在使用者進行註記活動時，我們必須要先紀錄其產生的註記的位置資訊，然後在使用者下次開啟檔案並且讀取之前的註記時，系統會執行演算法的第二階段，計算出最佳的結合模式。第一階段所需資訊如下所述：

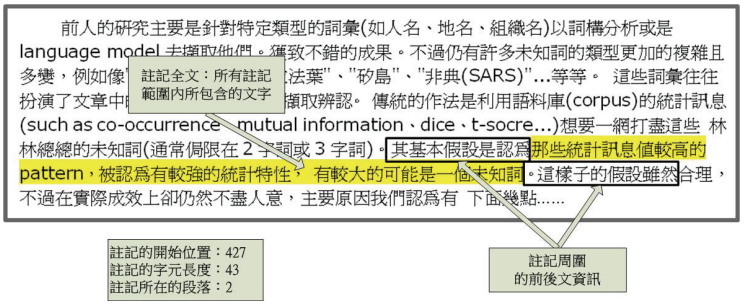

圖5：註記定位資訊範例圖

1. 註記產生時，如圖5所示，同時紀錄以下資訊：

（1）註記的開始位置（和檔案開頭的字元距離）。

（2）註記的字元長度。

（3）註記的前文：依照參數設定，從註記範圍開頭往前擷取指定長度的文字作為前文。

（4）註記的後文：依照參數設定，從註記範圍結尾往後擷取指定長度的文字作為後文。

（5）註記全文：完整擷取所有註記範圍內所包含的文字。

（6）註記所座落的段落，以數字表示。

2. 註記儲存之前系統將自動紀錄以下文章資訊：

（1）所屬文章全文所產生的MD5 fingerprint。

（2）文章各段落包含的關鍵字、開頭與結尾：演算法會尋找此段落範圍中在文章裡出現頻率最少的數個單詞作為關鍵字，並且從段落的開頭結尾擷取固定長度的文字。

在儲存註記檔案之前，我們會先使用雜湊演算法對目標文件進行運算，取得文件的內容特徵，我們所採用的演算法是MD5演算法，利用MD5可以對任何長度的檔案資料進行運算，得到一組固定長度為128位元的「指紋」（Fingerprint）（Check Sum）特徵值。將來我們可以利用這組資料來辨認檔案是否有被修改過，如果對檔案進行雜湊的結果改變的話，即代表該檔案已經被修改過。使用MD5所產生的指紋重複率低於百萬分之一，所以可信度相當高（Rivest 1992）。

第二階段，日後欲將檔與註記結合時：

1. 檢驗檔案是否經過修改：使用MD5演算法來確認檔案是否有經過修改，如果檔案沒有改變的話，直接使用原註記的位置與長度來呈現註記後結束。如果發現檔案被修改過，才進入第二步。
2. 尋找唯一且完全符合的區段：搜尋註記的全文以及前後文資訊，如果發現找到唯一且完全符合的區段，則決定其為註記的新位置。
3. 辨識文章段落：藉由段落的開頭結尾與關鍵詞搜尋，找到原本各段落所在的位置。如果發現文章中出現兩個段落以上具有相關的關鍵詞，兩者都將被列為「候選段落」，再比較關鍵詞的數量、段落的開頭結尾以及段落的長度，和原文差距最小者，信賴分數越高。將信賴分數最高的候選段落設定為正確的段落。
4. 設定候選區並評估其信賴分數：如果找不到唯一且完全符合的區段，則使用區域排比演算法（如下節所述）尋找一到數個文章中與原註記全文最相似的局部區域作為候選區，每個候選區都會向周圍搜尋前後文資訊，並且辨識其是否位於與原註記相同的段落。進一步，我們評估各候選區全文及前後文與原註記的相似度，再額外參考座落的段落是否正確，以此決定各候選區的信賴分數。
5. 決定註記的開始與結束位置：我們會依據註記候選區全文與前後文來決定註記的開始與結束位置；判斷前文與候選區何者的相似度較高，並且使用較相似者來決定註記的開始位置。而註記的結束位置也是使用同樣的方式在後文與候選區全文兩者中決定。
6. 註記重新定位：如果最高分的候選區其信賴分數超過系統設定的門檻值的話，就會被接受為註記的新位置。系統會依照註記的類型作適當的處理並且將其呈現於該處，否則系統將放棄這個註記，不會將其顯示。

三、候選區的選取：序列比對

在生物學上常需要比對物種DNA序列的相似度，因此時常會使用到序列比對的方式，而所謂的序列比對指的是將欲比對的序列並列，盡量將相同的字元放至於相對的位

置，並且允許利用間隔（通常使用「－」符號）表示缺漏的字元，以利計算序列間的相似度。目前較有名的序列比對演算法主要有整體排比（Global Alignment）（Needleman & Wunsch 1970）、區域排比（Local Alignment）（Smith & Waterman 1981）兩種。前者可以十分精確的求出兩序列間最佳的序列，而後者則是能夠找出兩序列間最相似的局部區域。兩者的時間複雜度都是O（mn），正比於兩序列的長度m、n。

本研究首創將區域排比演算法運用於註記的重定位問題，利用該演算法可尋找文章中與原註記範圍文字最為相似的幾個局部區域，作為註記的候選區，再進一步以字串相似性的指標來判定何者為最相似。

兩序列間比對有三種情況會出現，如果兩邊的字元相符，這種情境是「Match」，如果兩邊的字元不相符則為「Mismatch」，如果只有一邊有字元，另一邊使用間隔表示無相對字元的話則為「Gap」。有兩序列長度分別為m、n，區域排比主要是利用一個（$m+1$）×（$n+1$）的矩陣來求取兩序列間最相似的局部區域，我們必須先為Match、Mismatch與Gap三種情況設定對應的分數，並且依以下規則在矩陣中填入對應的分數$s_{i,j}$（Smith & Waterman 1981）：

$$
\begin{aligned}
&\text{if } x_i = \mathrm{y_j} \text{ then}\\
&\quad s_{i,j} = s_{i\text{-}1,j\text{-}1} + w(x_i, y_j)\\
&\text{else}\\
&\quad s_{i,j} = \max\begin{cases} 0 \\ s_{i-1,j} + w(x_i, -) \\ s_{i,j-1} + w(-, y_j) \\ s_{i-1,j-1} + w(x_i, y_j) \end{cases}\\
&\text{endif}
\end{aligned}
$$

$$
w(x_i, y_j) = \begin{cases} \text{mismatch score} & (x_i \neq \mathrm{y_j}) \\ \text{match score} & (x_i = \mathrm{y_j}) \end{cases}
$$

$$
w(x_i, -) \text{ and } w(-, y_j) = \text{gap score}
$$

以兩序列x＝｛GAATCGTC｝、y＝｛GCCTTGC｝為例，如果我們分別設定情境分數為Match＝＋8、Mismatch＝－5與Gap＝－3，則其建立的矩陣如圖6所示，從矩陣中可以很清楚的觀察到18是其中的最高分，由此位置開始向前回溯即可求出兩序列最相似的局部區域。

		G	A	A	Ⓣ	C	Ⓖ	T	Ⓒ
	0	0	0	0	0	0	0	0	0
G	0	8	5	2	0	0	8	5	2
C	0	5	3	0	0	8	5	3	13
C	0	2	0	0	0	8	5	2	11
T	0	0	0	0	8	5	3	13	10
Ⓣ	0	0	0	0	8	5	2	11	8
Ⓖ	0	8	5	2	5	3	13	10	7
Ⓒ	0	5	3	0	2	13	10	8	**18**

圖6：一個區域排比的計算範例

因此兩序列間最相似的局部區域為以下灰底標示的部份：

	G	A	A	T	C	G	T	C
G	C	C	T	T	-	G	-	C

在某些例子裡我們可能會發現到最高分並不唯一，有時也可能需要擷取數個高於某門檻值的結果，此時可以由矩陣中擷取所有分數最高的相似局部區域，以及其他符合標準的相似局部區域做進一步比較。本研究即運用此方式擷取文章中多個與原註記頗為相似的區域作為候選區，並以下列相似度評分做進一步分析。

四、字串相似度

針對上述所選擇的候選區，我們需要評估其與原註記的相似度。本研究使用三種序列相似度的比對方法：「最長共同子序列（Longest Common Ssubsequence，簡稱LCS）」、「最長共同連續子序列（Longest Common Consecutive Subsequence，簡稱LCCS）」、「連續子序列的長度和」（Sum of Common Consecutive Subsequence，簡稱為SCCS），前兩者是現有的比對方法，第三種是本研究改良自LCCS的演算法。

（一）最長共同子序列（LCS）

LCS的演算法是由Needleman與Wunsch（1970）所提出，假設兩序列X、Y其長度分別為length（X）＝m、length（Y）＝n，此演算法是利用一個（m＋1）×（n＋1）的矩陣來計算兩者的LCS。將目標的兩序列分別排列於矩陣兩軸之後，由左上角開始在矩陣中填入數字，首先將0填入所有x＝0或y＝0的欄位，之後依照由左至右、由上而下的順序來決定各欄位中的數字，如果遇到$x_i = y_j$，也就是發現兩者是相同的序列元素時，該欄位就填入「左上欄位的數字＋1」，而如果遇到xi與yj不相等的情況時，則是填入「上方或左方欄位中的最大數值」，不斷重覆此方式即可決定兩序列的LCS矩陣內容。

然後，依根據矩陣內容便可決定兩序列最長共同子序列的路徑：其路徑由最右下角開始往前回溯，依照填入矩陣時的順序往回走，持續到路徑指向內容為0的欄位為止，過程中所有劃「↖」的欄位即代表LCS的成員，路徑起始的最右下角欄位中的數字則是代表LCS的長度。演算法如下：

if $i = 0$ or $j = 0$ then

$$w(x_i, y_j) = 0$$

elseif $x_i = y_j$

$$w(x_i, y_j) = w(x_{i-1}, y_{j-1}) + 1$$

else

$$w(x_i, y_j) = \max \begin{cases} w(x_{i-1}, y) \\ w(x, y_{j-1}) \end{cases}$$

endif

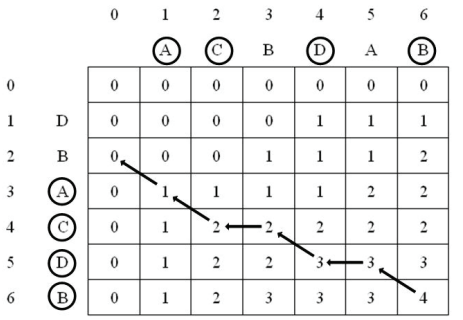

圖7：一個LCS的計算範例

以兩序列 X＝｛A,C,B,D,A,B｝，Y＝｛D,B,A,C,D,B｝做為例子，利用上述的方式來求其LCS，我們可以得到LCS（X,Y）＝｛A,C,D,B｝、LCS-Length（X,Y）＝4的結果，其計算矩陣如圖7所示。

（二）最長共同連續子序列（LCCS）

所謂的最長共同連續子序列（LCCS）就是尋找兩序列間相同且連續的子序列，藉此判斷序列間的相似程度，此方法和LCS的最大不同點即在於其要求的共同子序列必須要是「連續」的，子序列成員在原序列中的索引值必須要是連續的數值。如下述演算法來填入矩陣內容：

if $i=0$ or $j=0$ then
    $w(x_i, y_j)=0$
elseif $x_i = y_j$
    $w(x_i, y_j)=w(x_{i-1}, y_{j-1})+1$
else
    $w(x_i, y_j)=0$
endif

在矩陣中找到的最大值即為LCCS_Length（x,y），從該座標開始向斜上走↖，在到達0之前所經過的元素即是LCCS的成員，以兩序列X＝｛A,B,C,B,D,A｝，Y＝｛D,A,B,C,D,B｝為例，實作之後可以計算出LCCS的長度與路徑，LCCS_Length（X,Y）＝3、LCCS（X,Y）＝｛A,B,C｝，其過程如圖8所示。

（三）共同連續子序列的長度和

有時候使用LCS 以及LCCS 無法準確的評估序列的相似度，這是因為上述的兩種方法只使用最長的子序列作為指標，忽略了比較短的子序列的重要性。舉例來說，如果有X＝｛做筆記可以提升學習的效率｝，Y＝｛適時的獎勵可以提振員工的士氣｝, Z＝｛學習有效率的做筆記｝。我們知道X 句與Z 句明顯的較為相似，但是依據最長共同連續子序列來判斷YZ 何者與X 最為相似時，我們發現 LCS（X,Y）＝｛可以提的｝、LCS（X,Z）＝｛學習效率｝，兩者的長度相同。 LCCS（X,Y）＝｛可以提｝、LCCS

（X,Z）＝｛做筆記｝，兩者長度仍然相同。由此可知位於共同連續子序列裡的元素數量也是一個重要的相似度指標，我們認為其數量越高，代表兩者的相似程度越高，因此我們稍微修改LCCS的演算法，自訂了一個字串相似度的比對方式稱為「連續子序列的長度和」SCCS。

		0	1	2	3	4	5	6
			A	B	C	B	D	A
0		0	0	0	0	0	0	0
1	D	0	0	0	0	0	1	0
2	A	0	1	0	0	0	0	2
3	B	0	0	2	0	1	0	0
4	C	0	0	0	3	0	0	0
5	D	0	0	0	0	0	1	0
6	B	0	0	1	0	1	0	0

圖8：一個LCCS的計算範例

SCCS的表格和產生的演算法與LCCS完全相同，圖9是兩序列X＝｛A,B,C,B,D,A｝，Y＝｛D,A,B,C,D,B｝的LCCS（x,y）運算矩陣，矩陣中每個c（i,j）值都代表了共同連續子序列的數量，將矩陣中所有的數字加總便可以得到所有共同連續子序列的數量，子序列的長度不限，任何長度大於0 的序列都會被算進去，因此這兩個序列有13 個共同連續子序列。我們在這裡可以發現一個很明顯的問題，雖然這兩個序列有13 個連續共同子序列，但是其中卻有很多成員重複的情況發生，例如Y 序列中的第二個元素A 就同時出現在連續共同子序列｛A,B,C｝與｛D,A｝中。我們應較為重視比較長的連續共同子序列，為了避免重複出現的情況發生，序列成員應歸屬於較長的共同連續子序列，也就是說我們應該將Y 序列中的第二個元素A 視為子序列｛A,B,C｝的子元素。

		0	1	2	3	4	5	6
			A	B	C	B	D	A
0		0	0	0	0	0	0	0
1	D	0	0	0	0	0	1	0
2	A	0	1	0	0	0	0	2
3	B	0	0	2	0	1	0	0
4	C	0	0	0	3	0	0	0
5	D	0	0	0	0	0	1	0
6	B	0	0	1	0	1	0	0

圖9：一個利用LCCS計算共同連續子序列數量範例

在序列成員應該歸屬於較長的共同連續子序列的原則之下，我們來說明如何求得兩序列的SCCS。運算的要點是：找到最長的共同連續子序列，再找次長的共同連續子序列，每次都擷取目前最長的共同連續子序列，已經被擷取過的序列元素不可被其他序列

使用，最後將所有取得的共同連續子序列做長度的加總。以下我們以另兩序列Y、Z 為例，Y＝｛D,A,B,C,D,B｝, Z＝｛A,B,C,A,D,B｝,詳細說明其SCCS（Y,Z）的計算過程。

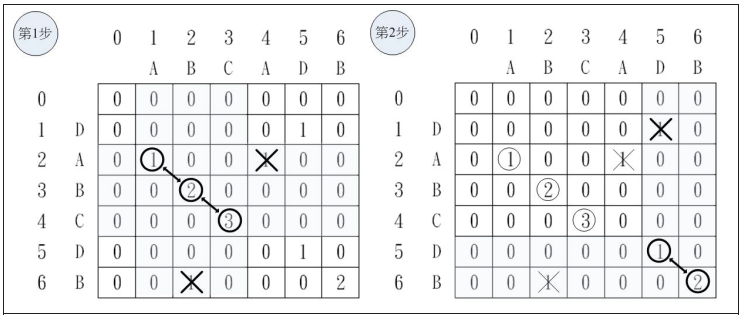

圖10：一個SCCS的計算範例

利用LCCS 的演算法產生矩陣如圖10，我們首先尋找矩陣中的最大值＝3，找到目前最長的共同連續子序列為｛A,B,C｝，依據序列成員應該歸屬於較長的共同連續子序列的原則，A、B、C 這三個元素不可再被重複納入其他較短的共同連續子序列中，因此我們在表格中將其垂直與水平欄位中的值都刪除（圖中以打叉表示）。第二步，搜尋目前次長的共同連續子序列，找到長度為2 的共同連續子序列｛D,B｝，將其垂直與水平欄位中的值都刪除。至此已沒有任何共同連續子序列可供選擇，最後可以得到依長度排序的2 個共同連續子序列：｛A,B,C｝、｛D,B｝，其長度和為3＋2＝5，因此其共同連續子序列的長度和SCCS=5。

五、註記候選區的信賴分數

註記候選區與原註記的相似度越高，其信賴分數也會越高，而系統也因此認定信賴分數最高的候選區為註記正確的定位點。上述之三種相似度的評估方法各有優缺點，單獨使用無法絕對準確的指出兩序列間的相似程度，因此我們使用三者的平均值作為本研究辨識字串相似度的方式。

本系統信賴分數的計算方式是分別計算全文、前後文以及座落段落各自的信賴分數，再作加權總合以求得最後的總分。

註記「全文」的部份，其信賴分數的計算如下所示：

$$S_a = \frac{LCS_a + LCCS_a + SCCS_a}{3 \times L_a}$$

S_a：候選區在註記開頭項目所得到的信賴分數。
LCS_a：「候選區全文」與「原註記全文」最長共同子序列的長度。
$LCCS_a$：「候選區全文」與「原註記全文」最長共同連續子序列的長度。
$SCCS_a$：「候選區全文」與「原註記全文」共同連續子序列的長度和。
L_a：「原註記全文」的長度。

註記「前文」的部份，其信賴分數的計算如下所示：

$$S_f = \frac{LCS_f + LCCS_f + SCCS_f}{3 \times L_f}$$

S_f：候選區在註記前文項目所得到的信賴分數。
LCS_f：「候選區前文」與「原註記前文」最長共同子序列的長度。
$LCCS_f$：「候選區前文」與「原註記前文」最長共同連續子序列的長度。
$SCCS_f$：「候選區前文」與「原註記前文」共同連續子序列的長度和。
L_f：「原註記前文」的長度。

註記「後文」的部份，其信賴分數的計算如下所示：

$$S_b = \frac{LCS_b + LCCS_b + SCCS_b}{3 \times L_b}$$

S_b：候選區在註記後文項目所得到的信賴分數。
LCS_b：「候選區後文」與「原註記後文」最長共同子序列的長度。
$LCCS_b$：「候選區後文」與「原註記後文」最長共同連續子序列的長度。
$SCCS_b$：「候選區後文」與「原註記後文」共同連續子序列的長度和。
L_b：「原註記後文」的長度。

註記的總信賴分數計算方式如下所述：

$$S = J_1 \times S_a + J_2 \times S_f + J_3 \times S_b + J_4 \times S_c$$

S：候選區的「總信賴分數」。
S_a 、S_f 、S_b：分別是候選區在「註記全文」、「註記前文」、「註記後文」項目所得到的信賴分數。
S_c：候選區「是否座落在正確段落」，正確為 1，反之為 0。
J_1 、J_2 、J_3、J_4：分別是「註記全文」、「註記前文」、「註記後文」、「座落段落」的加權值。此四者均容許使用者自行設定。

肆、系統測試

本研究所設計出系統雛型經過測試，以評估「註記重定位」演算法，我們採用Recall與Precision兩種指標做為評估效能的依據，其定義如下所示：

$$\text{Recall} = \frac{\text{系統接受且結果正確的次數}}{\text{實驗進行的次數}}，\quad \text{Precision} = \frac{\text{系統接受且結果正確的次數}}{\text{系統接受的次數}}$$

我們在註記重定位的過程中設有「門檻值」，如果最高分的後選區候選分數低於門檻值，則判定其相似度過低，無法代表正確的結果，因此系統不做任何的處理，選擇放棄此項註記。Precision中的分母「系統接受的次數」即代表了通過門檻值限制的次數，而Precision與Recall的分子「系統接受且結果正確的次數」則是表示候選分數超過門檻值且定位完全正確的次數。

一、實驗設計

對文字內容的修改情境大概可以分為刪除、改寫、增加以及移動這幾種類型，因此本實驗針對以上幾種類型的修改進行測試，如表3所示。

表3：文字內容修改類型

改變類型	描述
刪除關鍵字	一到少於一半的關鍵字被刪除
	一半以上的關鍵字被刪除
	全部的關鍵字被刪除
刪除開頭結尾	一到少於一半的開頭結尾被刪除
	一半以上的開頭結尾被刪除
	所有開頭結尾都被刪除
刪除前後文	一到少於一半的前後文被刪除
	一半以上的前後文被刪除
	所有前後文都被刪除
刪除註記範圍文字	註記範圍文字刪減，幅度小於原長度的50%
	註記範圍文字刪減，幅度為50%~90%
改寫	註記範圍文字改寫，幅度小於原長度的50%
	註記範圍文字改寫，幅度為50%~90%
增加	註記範圍文字增加，幅度小於原長度的100%
	註記範圍文字增加，幅度大於原長度的100%
交換	註記範圍文字順序交換，幅度小於50%
	註記範圍文字順序交換，幅度為50%~100%
移動	移動註記範圍文字至本段落其他位置
	移動註記範圍文字至其他段落
複製	複製註記範圍文字至本段落其他位置
	複製註記範圍文字至其他段落
改變段落	文章中增加其他段落
	註記段落在文章中的位置移動
	文章中其他段落被刪除
	註記段落某段落合併

我們設計了一個測試註記重定位正確性的介面，可以針對不同的情境與評分參數進行實驗。測試的流程首先是從參數設定的介面，設定本次實驗的評分參數與實驗情境，隨後讀取實驗用的文件即可開始測試。除了供使用者手動測試之外，系統可以自動進行重定位的測試，實驗的過程中系統會模擬使用者的註記行為，藉由亂數的方式選定劃記的範圍以及長度，再依照參數設定的情境對文章進行修改並評估其重定位的正確性，系統會自動重複進行100次的實驗並統計其實驗結果。此外，為了與Brush（2002）所提出的演算法做比較，本系統實際模擬並建置其演算法，於相同的環境中測試，因此，雖然本實驗之演算法並未使用所謂「關鍵字」的概念，但是為了和Brush所提出的演算法做比較，因此實驗中包含了關鍵字被修改的情境，其關鍵字的決定方式也與Brush完全相同。例如當我們決定要測試「一半以上少於全部的關鍵字被刪除」的情境並且目前有5個關鍵字時，系統會在符合情境的情況之下亂數決定要刪除3個還是4個關鍵字，並隨機從這5個關鍵字中刪除指定的數量。

由於註記重定位的結果有時不一定會完全正確，因此我們在實驗中也記錄了「部分正確」的次數作為參考。所謂的部分正確指的是註記的確重定位在正確的位置，包含了部分的原註記內容，但是其註記範圍的「開始」或「結束」位置有所誤差，使得有些文字內容沒被包含在新的註記之內，或是多圈選了一些額外的文字內容，對使用者來說，這種結果仍然是有價值的，和完全的錯誤不同。

本次實驗所使用的文章是朱自清的「背影」，該文共有1328個全形字。我們利用本系統的測試功能對每一種不同情境（註記長度由最小20字元，至最長459字）與參數設定進行實驗，每種組合個別進行100次的測試，每次測試於一般PC環境下，即使於最複雜的情境時，系統執行重定位在均可以在一秒鐘內顯示結果。系統會評估每種組合的正確性並加以紀錄，在測試完成後統計並顯示其Recall與Precision值。

二、實驗結果討論

本實驗發現當信賴分數低於40分時，相似度其實很低，幾乎不可能符合使用者的要求，應該主動將其棄卻，因此建議門檻值應設為40。表4為本實驗部分結果列表。由於篇幅限制，我們僅列出「完全正確」者。本系統在無門檻設定時「Recall完全正確」的平均值為96.7%，若進一步包含「Recall部分正確」的平均值則高達97.9%。而在設定40分之門檻值後，實驗結果「Precision完全正確」的平均值提昇到98.7%，若進一步包含「Precision部分正確」的平均值甚至接近100%（99.9%），相較Brush（2002）系統的演算法精確很多（其「Recall完全正確」的平均值僅67.52%，若進一步包含「Recall部分正確」的平均值也僅達86.4%。達40分之門檻值「Precision完全正確」的平均值僅69.68%）。而且在各種情境中都可以正確的重定位，即使在註記文字遭到非常嚴苛的刪改（50%到90%的內容）時，仍然有不錯的正確率，其穩定度令人十分滿意。另外，我們對每項修改情境皆進行10次實驗，統計25項情境所產生之250次實驗結果，顯示J1、J2、J3與J4的最佳設定值分別為60、15、15、10。此顯示四者加權值應同時被考量，註記文字（J1）很重要、應優先考慮，但其前後脈絡（J2、J3）、所在段落（J4）也應適度

考量，以避免文中相同文字被誤判。唯J_2、J_3、J_4權重不宜過高，否則會造成本末倒置的情況。

表4：實驗結果列表

改變類型	描述	本研究		Brush	
		Recall 完全正確	Precision 完全正確（門檻值=40）	Recall 完全正確	Precision 完全正確（門檻值=40）
刪除關鍵字	一到少於一半的關鍵字被刪除	99/100	99/100	95/100	95/100
	一半以上的關鍵字被刪除	99/100	99/100	89/100	89/100
	全部的關鍵字被刪除	100/100	100/100	1/100	1/4
刪除開頭結尾	一到少於一半的開頭結尾被刪除	100/100	100/100	58/100	58/99
	一半以上的開頭結尾被刪除	100/100	100/100	35/100	35/98
	所有開頭結尾都被刪除	100/100	100/100	0/100	0/95
刪除前後文	一到少於一半的前後文被刪除	99/100	99/100	99/100	99/100
	一半以上的前後文被刪除	100/100	100/100	98/100	98/100
	所有前後文都被刪除	100/100	100/100	98/100	98/100
刪除註記範圍文字	註記範圍文字刪減，幅度小於原長度的50%	97/100	97/100	38/100	38/93
	註記範圍文字刪減，幅度為50%~90%	85/100	85/88	3/100	3/33
改寫	註記範圍文字改寫，幅度小於原長度的50%	100/100	100/100	38/100	38/92
	註記範圍文字改寫，幅度為50%~90%	77/100	77/80	1/100	1/31
增加	註記範圍文字增加，幅度小於原長度的100%	100/100	100/100	99/100	99/100
	註記範圍文字增加，幅度大於原長度的100%	90/100	90/99	98/100	98/100
交換	註記範圍文字順序交換，幅度小於50%	100/100	100/100	47/100	47/92
	註記範圍文字順序交換，幅度為50%~100%	76/100	76/81	5/100	5/33
移動	移動註記範圍文字至本段落其他位置	99/100	99/100	96/100	96/100
	移動註記範圍文字至其他段落	99/100	99/100	98/100	98/100
複製	複製註記範圍文字至本段落其他位置	98/100	98/100	98/100	98/100
	複製註記範圍文字至其他段落	100/100	100/100	98/100	98/100
改變段落	文章中增加其他段落	100/100	100/100	99/100	99/100
	註記段落在文章中的位置移動	100/100	100/100	98/100	98/100
	文章中其他段落被刪除	100/100	100/100	99/100	99/100
	註記段落某段落合併	100/100	100/100	100/100	100/100
平均值		0.9672	0.987	0.6752	0.6968
標準差		0.06867	0.02213	0.39162	0.36249

本研究的註記定位演算法與Brush的演算法主要的差異在於：

1. 本演算法使用區域排比擷取候選區，Brush的註記演算法則是利用關鍵字。
2. 本演算法使用整個註記範圍文字的相似度來為候選區評分，Brush的演算法則主要是利用關鍵字作為評分依據。

3. 本演算法較重視註記周圍的前後文資訊，用其修正註記的開始與結束位置。

4. 本演算法額外使用了段落是否正確之指標。

Brush的註記演算法由於只紀錄部分的定位資訊，當這些關鍵的定位資訊被更動時會產生較大的影響。其中以關鍵字最為重要，關係到其候選區之建立，如果文章註記範圍內之關鍵字全數遭到修改很可能導致無法辨認候選區，當關鍵字在文章中的出現頻率較高時則是容易產生很多錯誤的候選區。此外，關鍵字同時是候選區評分的重要因素，但是關鍵字的保留程度與註記的相似度並不是完全相當，這可能會導致定位的誤差，使系統錯認某段具有相似關鍵字的文字區段。

在Brush的演算法中雖然有評估前後文以調整信賴分數，但是並未做更進一步的利用。其主要依賴開頭結尾資訊來決定候選區的開始與結束位置，因此當開頭與結尾遭到修改的時候，註記的範圍往往會產生偏差，使得其結果只有部分正確，無法達到完全正確定位的目標。而本演算法利用前後文作為修正註記範圍的依據，如果前後文與原註記前後文的相似度較高，而候選區全文與原註記全文的相似度較低時，顯示候選區內容變動較大，不易產生正確的開始結束位置，本演算法就會參考前後文的位置來調整註記的範圍，以達到更精確的重定位結果。

一般造成重定位錯誤的因素主要有兩大類，首先是找到不符的註記段落，這種情境會使得註記定位於完全錯誤的位置；其次是找不到註記正確的開始與結尾位置，這類情境會使重定位無法「完全正確」，但是能得到「部分正確」的結果。

Brush與本實驗所提出之重訂位策略差異較大的修改情境為「刪除開頭結尾」、「交換」、「改寫」與「刪除註記範圍文字」四項，Brush的重定位策略由於只使用註記開頭結尾的文字做為決定註記範圍的依據，所以在改變情境「刪除開頭結尾」中容易遇到第二類的錯誤，使得註記無法得到完全正確的重定位結果，在「交換」情境也會遇到類似的狀況，交換的過程容易使開頭或結尾的文字離開原本的位置，移動到其他地方。本實驗所提出之重訂位策略由於使用前後文作為重定位的輔助，因此可以減少此類錯誤。

在「改寫」與「刪除註記範圍文字」兩類的情境中狀況就比較複雜，文章在這些情境中遭到刪改的部份有可能是開始與結尾，也有可能是註記範圍中的任何文字，因此兩類的錯誤都可能出現，使得重定位很難得到正確的結果，尤其是Brush的重定位策略特別重視關鍵字的使用，容易因為關鍵字消失而造成定位錯誤。如前所述，本實驗所提出之重訂位策略可以減少第二類的錯誤，另外，由於本實驗利用所有註記文字的相似度來尋找註記位置，能夠容忍任意內容且較大幅度的文字修改，避免因為關鍵字遭到刪改所產生的第一類錯誤，因此本實驗之演算法在重定位的正確性上較優於Brush的演算法。

伍、結論與建議

目前已經有許多商業或學術軟體提供和本系統類似的註記功能，但是這些軟體都有一些使用上的不方便，首先，它們大部分非常缺少瀏覽並且處理網頁資料的能力。第二，許多軟體將註記視為文件內容的一部分，使用者必須要具備修改文件的權限才能使

用系統所提供的註記功能，因此可以閱讀文件時不一定可以對其進行註記。第三，這類型的註記系統多半將其功能侷限在單人單機使用，不支援任何透過網路與他人交換資訊的機制，作為使用者互相交流學習的媒介十分不方便。第四，現存將註記與文件分開儲存的註記系統多半不具備對註記重定位的能力，一旦文件經過修改往往會使得註記的呈現產生錯誤。

本研究的目的是以自我學習與合作學習的概念為基礎，建置一個可以支援個人學習與群體工作的註記系統，克服上述註記系統普遍具備的缺點，提供使用者瀏覽檔案文件與網頁內容的功能以及同步與非同步的溝通機制，利用定位演算法作為註記與文章之間定位的依據，以因應註記與文件分離儲存所產生的重定位問題，並且對本系統的重定位功能進行效能方面的測試與評估，希望本研究的成果能對學習科技方面的發展有所幫助。

一、系統特色

本研究之系統具備了一般註記軟體常見的註記功能如畫線、螢光底色、改變字體大小、文字方塊便利貼、頁邊註記等，同時還具有下列的優點及特色：

1. 本系統採用瀏覽器基礎的建置模式，提供使用者閱讀檔案文件與瀏覽網頁的能力，並且可以自由的對文章進行註記，無論是否處於連線的狀態下，本系統都能運作，使用者可以選擇離線觀看檔案文件，在需要瀏覽網路資訊或是想要與他人進行討論時才連上網際網路，在使用上非常便利。
2. 我們採用註記與文件分開儲存的模式使註記得以獨立於文件檔案之外，使用者可以對一些原本不具編輯權限的目標進行註記，並且產生比文件小很多的註記資訊檔案，輕便的檔案大小有利於藉由網路來進行傳遞，獨立的註記資訊檔則較具有彈性，除了可以在日後顯示於原始文件上之外，也可以加以分析以產生其他有價值的資訊。
3. 本系統具備了同步與非同步的溝通機制，使用者可與他人即時交換文字資訊並且分享註記畫面，讓身處兩地的雙方就如同面對面一般，一起瀏覽同一份文件、看到同樣的註記並且彼此進行交談。此外透過註記的分享平台，使用者也可以選擇將自己的註記分享於網路上，或是下載他人的註記作為參考。這些功能對於團體的合作與學習都是十分有用的工具。
4. 由於本系統將註記與文件分開儲存，必須處理文件在無預期的情況下被修改時產生註記位置錯誤的狀況，大部分的註記系統並不能處理此問題，部分具備重定位能力的系統其處理方式也不甚理想，因此本研究為這種情況設計了一套有效的重定位模式，所提出的演算法相較文獻上的有重大突破，實驗顯示本系統在各種文件修改的情況下仍能將98.7%的註記完全正確的重新定位；如果我們接受註記重定位其開始或結束位置略有誤差，但絕大部分的註記內容都被正確涵蓋的結果，其正確率更高達99.9%以上。

二、未來研究方向與建議

由於時間上的限制，本系統雛型尚有許多方面的功能有待加強：

1. 本系統主要是採取字串的相似度比對來進行註記的重定位，未來可以研究使用其他的方式來幫助重定位的實現，並且嘗試讓其結果更符合使用者的期待，例如使用語意辨識技術幫助系統找到與原註記意思最接近的段落等。
2. 本系統能夠處理的檔案格式仍然太少，由於網頁格式的種類較少、其資料結構也會符合固定的規範，目前本系統對htm、html、xml、mht等格式均可支援。但是檔案文件的種類眾多且各有獨特的資料格式，均須特殊處理，目前可處理的文件格式包含純文字的txt、Word的rtf。未來可朝著這方面研究。
3. 我們所提供的同步交流機制較為簡單，無法傳送聲音與彼此的影像，這是在技術上尚待完成的功能。另外，本研究將重點放在單純的文字資訊上，對於圖形、動畫等多媒體註記並未探討，這是另一個值得發展的方向。

參考文獻

1. Agosti, M., Bonfiglio-Dosio, G. and Ferro, N.“A Historical and Contemporary Study on Annotations to Derive Key Features for Systems Design,”*International Journal on Digital Libraries* (8:1), Nov. 2007, pp.1-19.
2. Agosti, M. and Ferro, N.“Annotations as Context for Searching Documents,”*Lecture Notes in Computer Science* (3507), 2005, pp.155-170.
3. Brown, P.J. and Brown, H.“Integrating Reading and Writing of Documents,”*Journal of Digital Information* (5:1), 2004, pp.1-18.
4. Brush, A.J.B.“Annotating Digital Documents for Asynchronous Collaboration,”Ph.D. dissertation, University of Washington, 2002.
5. Brush, A.J.B. and Bargeron, D.“Robustly Anchoring Annotations Using Keywords,”*Technical Report*, MSR-TR-2001-107, 2001.
6. Brush, A.J.B., Bargeron, D., Gupta, A., and Cadiz, J.J.“Robust Annotation Positioning in Digital Documents,”*Proc. Conference on Human factors in Computing Systems (CHI)*, 2001,Seattle, WA, pp.285-292.
7. Cunningham, S.J. and Knowles, C.“Annotations in an Academic Digital Library: The Case of Conference Note-Taking and Annotation,”*Lecture Notes in Computer Science* (3815), 2005, pp.62-71.
8. Fiala, Z. and Meisner, K.“Annotating Virtual Web Documents with DynamicMarks,”*Workshop XML Technologien fur das Semantic Web* (XSW), 2003, Berliner XML Tage, Berlin.
9. Kahan, J., Koivunen, M., Prud'Hommeaux, E. and Swick, R.R.“Annotea: An Open RDF

Infrastructure for Shared Web Annotations," *Tenth World Wide Web Conference,* Hong Kong, 2001, pp.623-632.

10. Marshall, C.C. and Brush A.J. "Exploring the Relationship between Personal and Public Annotations," *Joint Conference on Digital Libraries*, 2004,New York, NY.
11. Needleman, S.B. and Wunsch, C.D. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," *Journal of Molecular Biology* (48:3), 1970, pp.443-453.
12. Phelps, T.A. and Wilensky, R. "Robust Intra-document Locations," *Ninth World Wide Web Conference*, 2000, Amsterdam.
13. Plimmer, B. and Mason, P. "A Pen-based Paperless Environment for Annotating and Marking Student Assignments," *Proceedings of the 7th Australasian User Interface Conference*, 2006, Hobart, Australia.
14. Qayyum, M.A. "Capturing the Online Academic Reading Process," *Information Processing & Management* (44:2), March 2008, pp. 581-595.
15. Rivest, R. "The MD5 Message-Digest Algorithm," RFC 1321, The Internet Engineering Task Force, April 1992.
16. Smith, F.F. and Waterman, M.S. "Identification of Common Molecular Subsequences," *Journal of Molecular Biology* (147), 1981,
17. Yee, K.P., "CritLink: Advanced Hyperlinks Enable Public Annotation on the Web," *Proceedings of Conference on Computer Supported Cooperative Work (CSCW)*, Nov. 2002, New Orleans, Louisiana.