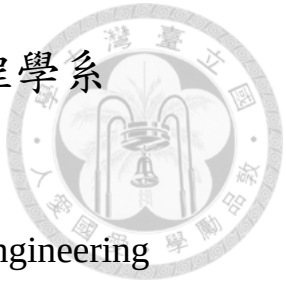


國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

基於安全性的排名優化

Ranking Optimization with Security Awareness

古佳怡

Chia-Yi Ku

指導教授：鄭卜壬 博士

Advisor: Pu-Jen Cheng, Ph.D.

中華民國 104 年 7 月

July, 2015

國立臺灣大學碩士學位論文
口試委員會審定書

基於安全性的排名優化

Ranking Optimization with Security Awareness

本論文係古佳怡君（學號 R02922008）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 104 年 7 月 31 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

（指導教授）

系 主 任

誌謝

首先，我要感謝我的指導教授—鄭卜壬老師。感謝在研究的過程中，老師耐心傾聽我的想法，點出我思考上的盲點，並且在我陷入困境時給予建議。每次和老師討論，總是能啟發我新的觀點，真的獲益良多。謝謝老師願意包容在研究的路上跌跌撞撞的我，引領我前進。也感謝擔任口試委員的蔡銘峰教授、張嘉惠教授、陳信希教授與曾新穆教授。感謝老師們提出諸多寶貴的意見，使本研究能夠更臻完善。

感謝實驗室的彥銘（麵包）學長，不管是目前還是之前的研究都給予我許多協助。特別是上一篇研究，除了一起構想方向、討論該如何解決問題外；也提供了我許多寫作上的建議，並在我完成初版後，逐字逐句地一起討論該如何潤飾與修改。實在麻煩學長太多，真的很感謝學長。如果我有找到好吃的義式披薩，會再告訴學長的！也感謝麵包學長與亭屹、浩丞與雅喬，我會很想念一起去後門 118 巷吃飯的日子。每次都好難決定要吃什麼，然後邊吃邊亂聊許多奇怪的東西。亭屹，你真的很認真與勤儉負責，也是我少數遇到和我一樣會把涼麵醬和水餃湯都吃光光的人。浩丞，和你聊天有種很溫暖與療癒的感覺。還有雅喬，竟然在這麼後來才發現我們有同樣的興趣，跟你聊天真的很開心。還有口試前我們彼此打氣、熬夜與各種極限的事蹟。不誇張的說，因為有妳的陪伴，口試前的日子我才能夠支撐下去。還有姜姜，每當看到鍵盤上的大寫鎖定（caps lock）鍵，我就會想起你每次經過時都會偷偷按下它的事情。當然還有同為 IRLab 的偉恩、丕勳、心萍、建德、彥傑、廷璋、家瑞，感謝有你們的陪伴與支持。

也要感謝 Shpping 與蔣易，每當研究上遇到挫折時我們互相訴苦，漫漫的研究生活因為有妳們而不孤單。也感謝室友云芳，我會想念好幾個晚上我們躺在各自的床上，天南地北的聊天，聊著彼此的夢想、人生、還有小時候看過的少女漫畫。當然還有阿戴，常常一起吃飯、更新彼此的近況與生活，還有各種不知道跟誰說才好，放在心底的事情。你真的是我的心靈導師，感謝有你。最後，我要感謝我的家人。雖然很少見面，但是我知道始終支持我的父親；支持我、常常擔心我在台北有沒有好好吃飯的媽媽與外公；分享各種有趣新知和互相說些解壓又無厘頭的話的妹妹，真的很感謝你們。因為有我的家人，我才能夠無後顧之憂的在台北求學。最後的最後，感謝所有相遇過、曾經幫助過我的人，這些都是很美好的回憶，我會永遠記得。因為有你們我才會在這裡，真的很謝謝你們。

中文摘要

隨著雲端儲存服務的普及，越來越多使用者選擇將資料儲存於雲端服務平台上（如：Dropbox、Google Drive、Apple iCloud Drive 等），以方便後續管理與使用。然而，對於一些較隱私的資料，使用者儘管仍欲使用服務供應商所提供的儲存與管理服務，但因擔心可能遭受他人窺視、攻擊以及資料外洩的風險，而往往會傾向不將該類資料放於雲端平台上。因此，為提供使用者一個更好且安心的使用環境，許多雲端服務所需具備的隱私要求（privacy requirements）與相關加密與搜尋技術相繼被提出。其中一部分為有關搜尋模式（search pattern）的隱藏，即當使用者下達一組關鍵字查詢（query）後，對於最終回傳的排名結果（ranked list）應做適度隱藏。以避免雖已對查詢所下達的關鍵字做隱藏的動作，但因回傳的排名結果間的相似，而致第三方推測出兩筆查詢間有所關聯。並減少因洩漏使用者的行為模式或較常存取哪些特定文件等資訊，以使第三方有機會對此作特定攻擊。

然而對於搜尋模式的隱藏，在現有辦法中皆存在著一些問題，如未考慮排名結果間次序性的重要度，搜尋模式隱藏時未對排名結果的效能做維持，或者耗費過多時間做隱藏等。為此，我們基於文件與查詢字眼間所計算出的相關分數（relevance score），提出同儕的概念。在維持排名結果品質的情況下，讓同一組的關鍵字查詢盡可能產生不同回傳結果，以隱藏使用者的搜尋行為。

關鍵字：安全性、隱藏搜尋模式、排名效能、排名關聯度、基於同儕

ABSTRACT

Since cloud service are wide-spread used nowadays, more and more users store their data on cloud storage (e.g., Dropbox, Google Drive, Apple iCloud Drive, etc.) for management. However, because of the risks of information leakage, user might tend not to store sensitive personal data in cloud. Thus some privacy requirements and corresponding technologies are being proposed. One of the requirements is search pattern privacy (i.e., hiding the query search's result). To avoid a third party infer query correlation from result correlation (i.e., similar query results might come from similar queries). And preventing focused attack because of private interest leakage.

Unfortunately, existing search pattern hiding techniques have some shortcomings. For example, the method does not consider rank order, do not maintain ranking performance, or is inefficient. Thus, based on the relevance score between document and query keyword, we propose the concept of peer. Not only to generate as different result as possible by the same query, but also maintain ranking performance. That means achieve search pattern privacy.

Keywords: security, search pattern privacy, ranking performance, rank correlation, peer-based

目錄

圖目錄

表目錄

第一章　　緒論

1.1　研究動機

隨著雲端儲存服務的普及，越來越多使用者選擇將資料儲存於雲端平台，以在任何時間與空間下，都能夠查詢、搜尋與管理所需資料。目前知名的雲端儲存平台有：Dropbox[1]、Google Drive[2]、iCloud Drive[3]等。然而將資料儲存在雲端上不免有許多安全上的風險，舉例來說，2014年即因 iCloud 遭受駭客入侵，導致上百位好萊塢女星的私密照遭到外流。因此，儘管雲端平台提供了許多便利的服務，但許多個人或商業用戶仍可能傾向於將較隱私或機密的資料放於私人儲存設備而非雲端平台上，以避免遭到入侵與後續可能承擔的風險。

為了減少此類擔憂，並提供使用者一個更加安全與放心的使用環境，許多相關雲端儲存與搜尋的隱私要求相繼被提出，整體來說可以分為兩大類 [1]：

一、資料儲存時的隱私（Storage Privacy）

(1) 文件內容本身需經加密（Data Privacy），以防他人對文件內容進行窺探。

(2) 文件內有的字詞索引資料需經加密(Index Privacy)，以防他人雖不知文件完整內容，但卻可由索引得知文件中各字詞所出現的頻率，並進而推測出該文件的部分或完整內容。

二、資料查詢時的隱私（Search Privacy）

(1) 查詢時所下達的關鍵字需經加密（Keyword Privacy），以防他人得知使用者目前欲查詢的內容，和使用者偏好查詢哪些關鍵字詞。此類將查詢關鍵字進行加密後所產生的物件，通常被稱為「暗門（Trapdoor）」。

(2) Trapdoor 需不和產生其的關鍵字有所連結（Trapdoor Unlinkability），即對於同一組關鍵字，應產生隨機且不可逆的 Trapdoor。以防雖已將查詢關鍵字進行加密，但因其所產生的的 Trapdoor 間彼此相同，而讓他人推測出兩 Trapdoor 為由同一組關鍵字所產生。如此他人將可得知使用者偏好使用哪些 Trapdoor，並嘗試對該類對使用者較重要、較常使用的 Trapdoor 進行竄改和集中攻擊與破解。

---

1 https://www.dropbox.com/

2 https://www.google.com/drive/

3 https://www.icloud.com/

(3) 搜尋時所回傳的文件頻率（Document Frequency）需經隱藏，即不能讓他人得知搜尋結果中各文件出現的確切頻率。以防當他人已知該儲存資料集的背景內容時，藉由所回傳的文件頻率等統計資料，以推測出使用者查詢和文件內所擁有的關鍵字。舉例來說，當已知儲存資料集為財務相關背景內容時，若有某些文件常被回傳，則可推測其內應含有此背景中較易被使用的關鍵字詞（如：報表）。並可回推出查詢其所使用的 Trapdoor 應由該類熱門關鍵字所產生。

(4) 搜尋模式（Search Pattern）需經隱藏，即不能讓他人得知確切的搜尋結果，即各文件的真實排名次序。以防雖已經由 Trapdoor Unlinkability，對同一組查詢關鍵字產生不同的 Trapdoor；但因所搜尋出的搜尋結果間彼此相似，甚至於完全相同，而讓他人推測出兩 Trapdoor 為由相似關鍵字詞所產生。除洩漏使用者搜尋模式外，他人也將可得知使用者偏好使用哪些文件，並可嘗試對該類對使用者較重要、較常使用的文件進行竄改和集中攻擊與破解。

為達成以上儲存和搜尋時的隱私要求，許多相關加密與隱藏技術相繼被提出。然而其中對於搜尋模式的隱藏，在現有辦法中皆存在著一些問題。如未考慮排名結果次序性的重要度，搜尋結果的隱藏並未盡可能對排名結果的品質進行維持，或者耗費過多的時間做隱藏等。

1.2 研究目的

為了達成更好的搜尋模式隱藏，避免耗費過多時間，或隱藏後反而損失過多的排名品質。我們希望藉由文件和查詢字眼間所計算出的相關分數（relevance score），得到和該文件相關分數差異較小的文件，並將其視為與該文件對等的同儕。基於此同儕概念進行交換，以在盡可能不損傷排名結果品質的情況下，讓同一組的關鍵字每次皆產生不同查詢結果，即減少結果間的關聯度，以隱藏使用者的搜尋行為。

1.3 內容架構

第一章先簡單的介紹研究背景、動機與研究目的。第二章則對於一些目前已被提出的相關研究進行討論。第三章則將介紹本研究相關系統架構，主要所基於的概念與研究方法，並嘗試對此方法進行使用者回饋。第四章講述實驗設計，如：實驗資料來源、與其比較的 Baseline 與評估的方法等，並對實驗結果進行討論。最後，在第五章闡述本研究結論與未來研究方向。

第二章　　相關研究與文獻

在過去的研究中有許多相關研究議題，如：對串流關聯度的隱藏、在條件限制下盡可能維持排名品質等；另則有增加無意義關鍵字和 Oblivious RAM 此兩類搜尋模式隱藏範疇。以下將對其做相關討論。

2.1　串流關聯度隱藏

雖相似於搜尋模式隱藏，但串流關聯度隱藏主要為隱藏使用者在網路中流入與流出量等純量間的關聯度。主要背景為，為隱藏使用者在網路中的移動軌跡，早期常見的作法為藉由在網路中安插中繼點，以混淆攻擊者所觀察到的使用者行經路線[2]。但之後 Gavin O'Gorman *et al.* (2009) 提出，因若直接將整個網路視為一個黑盒子，則可藉由觀察整個盒子流進與流出量間的關聯，以推測出使用網路的使用者是否為同一個，為此需對串流關聯度進行隱藏[3]。在此隱藏過程中對流量並不具有次序性的概念，即對各時間點的封包皆視為同等重要。但實際上，對於搜尋模式的隱藏來說，所隱藏的為經排名後具有次序性的搜尋結果，對於不同的排名間各有不同重要程度的差異。因此不考慮次序性的串流關聯度隱藏並不適用於搜尋模式隱藏中。

2.2　有條件限制的排名優化

在有些時候，所得出的排名結果需符合所訂立的限制條件，而不一定為原始的最佳排名。舉例來說，基於某些商業考量（如：廣告），可能會需將原始排名在前 K 名外的文件，強制塞入前 K 名當中；也可能為放大廣告效果，而指定兩文件排名需擺放在鄰近位置。為在以上幾點限制條件下計算新的排名結果，Fangzhao Wu *et al.* (2014)提出可將文件的相關分數 Model 為一個 Bradley-Terry 模型，並利用 Stochastic Gradient Decent 的方式，以找出最可能同時產生原始排名次序和符合條件限制的排名的模型 [4]。最後，據此模型即可得到最終優化過的排名。此方法雖可維持排名品質，但因只利用原始排名次序，未利用原始排名成績，而導致所參考的資訊量有所不足。並且其所訂立的條件限制和本研究目的亦有所不同，而不適用。

2.3 增加無意義關鍵字的排名隱藏

為隱藏搜尋模式，Ning Cao *et al.* (2014)提出，可在每個文件的索引檔中加入U個無意義、即不在任何文件內出現的字詞（如：ㄇ、ㄕ），各無意義字詞在文件內出現的頻率則不定 [1]。如此一來，當使用者欲做查詢動作時，即可隨機從該U個無意義字詞中取V個，並加入其真正想要查詢的關鍵字中。以讓同樣的查詢關鍵字，每次皆可隨機產生不同 Trapdoor；並因該 Trapdoor 內隨機隱含的無意義字詞作用，以使每次得到的排名結果皆不相同。但該方法中並未明確定義無意義字詞在文件中的出現頻率，即影響程度的多寡。並只考慮同一組查詢關鍵字每次需得到不同排名結果，而未考慮該排名結果的品質好壞。若因此而得到過於糟糕的排名結果，不免顯得本末倒置。相反地，若因無意義字詞對整體排名的影響度過小，導致同一組關鍵字各次所回傳的結果間差異不大，則不免又顯得白下功夫了。因此我們想要在盡可能維持排名結果的品質下，提出一種讓同一組關鍵字每次皆可得到一定差異程度的方法。

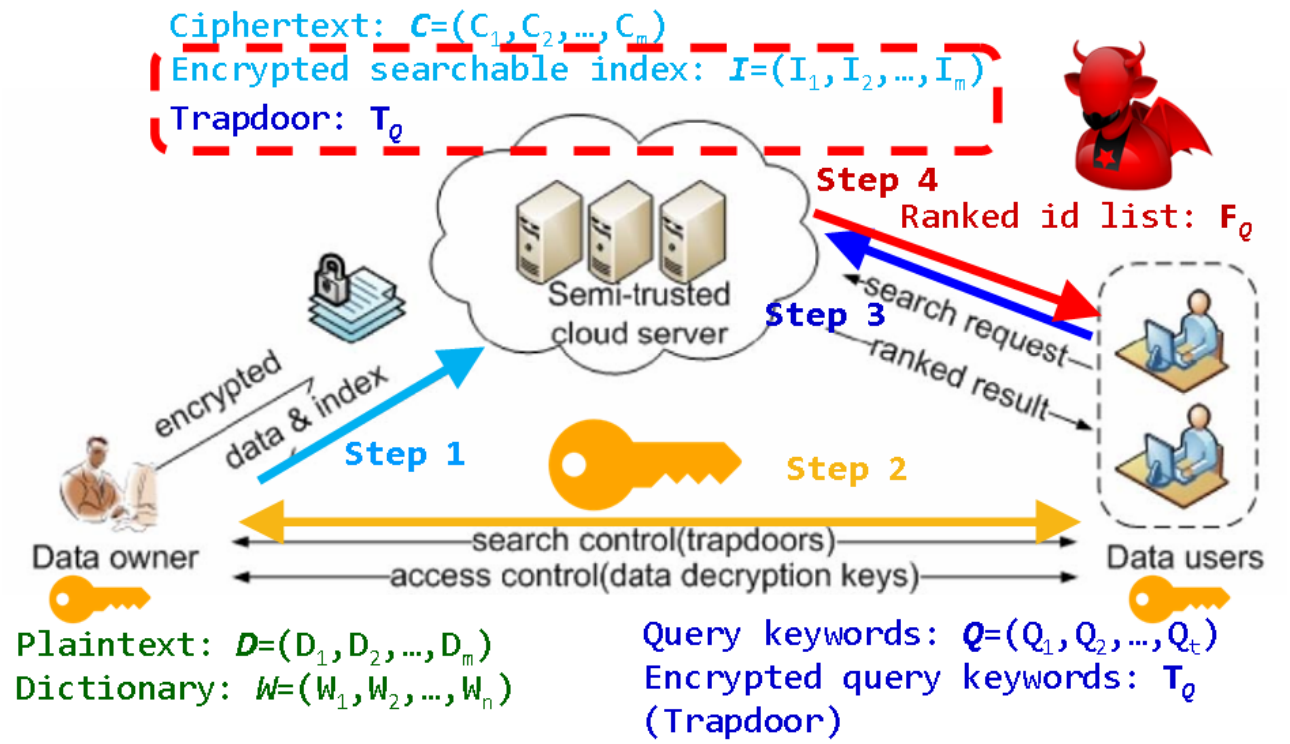
2.4 Oblivious RAM

另一常見且已被廣泛使用的搜尋模式隱藏方法則為 Oblivious RAM [5]，其概念最早由 Oded Goldreich *et al.* (1996)所提出，主要想法為在每次使用者進行搜尋後，皆由附加在伺服器外的 Oblivious RAM Node，對於所有儲存的文件和其索引位置進行洗牌（reshuffle）。對於同一文件來說，其舊有存放位置和現有存放位置已有所不同。因此，將可達成對於同一查詢動作，雖每次皆查詢出相同的對應文件，但卻可得到不同的文件 id 和存放位置，即隱藏了使用者的搜尋模式。但對於該類方法，每次搜尋需耗費 $O((\log N)^3)$的時間進行空間洗牌，其中 N 為整個資料儲存空間的大小，這是非常耗時且沒有效率的。為解決這一問題，許多改進辦法相繼被提出 [6-9]。其中 Emil Stefanov *et al.* (2013)提出，使用者每次搜尋後，Oblivious RAM Node 應只對有被回傳的索引和文件進行洗牌，以達到在最低耗費時間下，有效隱藏使用者的搜尋模式。但整體來說，每次搜尋後仍需耗費約 200 ms 的時間。並且因文件位置經過置換，則先前查詢時所得到的結果將無法再次使用（如：先前查詢結果中 id 為 1 的文件，已非現在 id 為 1 的文件了）。將導致使用者想索取已查過的文件時，需重新作完整的查詢流程，才能夠得到該些文件目前真正的 id 與位置。這除了增加伺服器計算量外，也將增加使用者的負擔。為此，我們希望提出的方法除了有先前提過的幾點要素外，也是一個相對有效率；雖每次皆回傳不同結果，但使用者仍可對舊有結果進行利用，以減輕伺服器與使用者雙方負擔的方法。

第三章　　研究方法

3.1　　系統架構

在整個加密搜尋的系統中，存在有資料擁有者（Data owner）、雲端伺服器（Cloud server）和資料使用者（Data user）此三種不同參與者身份。並且，為減輕三方負擔，除資料擁有者與使用者間的訊息交流通道，其他彼此間的通道是未經過加密的，需注意任何可能產生的資訊洩漏行為。此外，我們假設雲端伺服器是「誠實但具好奇心的（honest-but-curious）」。即對於所交付的工作，雲端伺服器會確實進行運算，並將計算結果誠實地傳予使用者。但另一方面，為滿足好奇心，雲端伺服器會嘗試對所運算出的結果進行統計資料的分析，以探究資料背景內容。因此，我們應儘可能避面透露過多資料內容予雲端伺服器知道。整體流程與系統架構如圖一 [1]：



圖一：加密搜尋的系統架構

首先對於資料擁有者，他的手中會擁有所欲儲存的資料明文（Plaintext）、藉此所產生的字詞字典（Dictionary）以及一組只有自己知道的加密鑰匙（Secret key）。之後，資料擁有者可藉由這組加密鑰匙對文件和文件索引進行加密，以產生密文（Ciphertext）和加密後的索引資料（Encrypted searchable index）。在下一步，資料擁有者會將密文和加密後的文件索引上傳至雲端伺服器中。為讓所信任的使用者可對這些加密資料進行查詢，資料擁有者會將手中

的加密鑰匙藉由經加密的安全通道，分享予信任的資料使用者使用。之後，當資料使用者欲做查詢時，會藉由該被授予的私人鑰匙，以對原始查詢關鍵字（Query keywords）進行加密。這組經加密的查詢關鍵字詞（Encrypted query keywords），即為上一章中所提到的 Trapdoor 概念。接著，資料使用者會將此 Trapdoor 上傳至雲端伺服器中。待收到 Trapdoor 後，雲端伺服器會將其和加密後的文件索引進行運算。並且由於前述的假設，伺服器應知道越少資料內容越好；因此，伺服器在運算後只會得到原始排名順序與各文件經查詢後的相關分數，此兩部分資料。之後，伺服器會將排名順序回傳予使用者知道。在最後回傳結果這一步驟中，若伺服器與使用者間的訊息通道遭受網路第三方（如：惡意攻擊者）窺探，則可能發生使用者搜尋模式洩露這一問題。

在整個系統流程中有關加密文件的產生，由於使用傳統對稱式或非對稱式加解密方法即可實行，因此在此不做詳細說明。唯對加密索引和 Trapdoor 的部份，因其經加密後雖無法得知原始內容，但仍須具備可搜尋的能力。為達成此目標，我們可以其中一種實作方法來對此進行說明。首先，資料擁有者會擁有一組加密鑰匙{ S，M_1，M_2 }，S 為一(n+U+1)的 bit vector，M_1、M_2 則各為(n+U+1)*(n+U+1)的可逆矩陣。接著為產生加密的文件索引，會先對於文件 j 產生原始文件索引 D_j，D_j 表示法如下：

$$D_j[i] = TF_{i,j} = \frac{f_{i,j}}{\sum_k f_{k,j}}$$

其中分子 $f_{i,j}$ 代表字詞 i 在文件 j 中所出現的次數，分母則為在文件 j 中所有字詞的出現次數之和，即為算出 TF-IDF 中 TF 的計算公式。之後資料擁有者會藉所擁有的加密鑰匙對此索引 D_j 進行加密，以得到加密的索引 I_j：

$$D_j \xrightarrow{extend} \left(D_j, s_{j1}, s_{j2}, \dots, s_{jU}, 1\right) \xrightarrow{split} \left\{D_j', D_j''\right\} \xrightarrow{encrypt} \left\{M_1^T D_j', M_2^T D_j''\right\} = I_j$$

其中 s_{ju} 代表的是資料擁有者對文件索引 j 所加入的無意義字詞 u。切割（split）時的所依照的規則則於其後再作說明。和加密索引產生方法相似地，當使用者想要對加密資料進行查詢時，需先產生所欲查詢的關鍵字詞 Q，Q 表示法如下：

$$\mathrm{Q}[i] = IDF_i = log\frac{m}{|\{d \in D: term_i \in d\}|}$$

其中分子 m 代表資料集中所有文件的數量，分母則為在該字詞 i 出現於多少文件，即為算出 TF-IDF 中 IDF 的計算公式。之後使用者會對藉經資料擁有者所授權給予的加密鑰

匙對此查詢關鍵字 Q 進行加密，以得到加密的查詢關鍵字 T_Q：

$$Q \xrightarrow{extend} (rQ, r\varepsilon_1, r\varepsilon_2, \dots, r\varepsilon_U, t) \xrightarrow{split} \{Q', Q''\} \xrightarrow{encrypt} \{M_1^{-1}Q', M_2^{-1}Q''\} = T_Q$$

其中ε_u代表的是使用者對查詢關鍵字所隨機加入的無意義字詞 u。r 和 t 則代表使用者所隨機產生的放大和位移係數。

在產生加密索引和加密查詢關鍵字時，為對產生過程中的切割有所依循規則，因此會藉由加密鑰匙 S 中所帶有的資訊以進行切割的動作。若 S[i]為 0，則：

$$\begin{cases} D_j'[i] = D_j''[i] = D_j[i] \\ Q'[i] + Q''[i] = Q[i] \end{cases}$$

反之則相反。之後由於分配律與矩陣反矩陣間可互消的概念，加密的索引 I 和加密的查詢關鍵字 T_Q 經計算後可得到一正比於原始 TF-IDF 的數值：

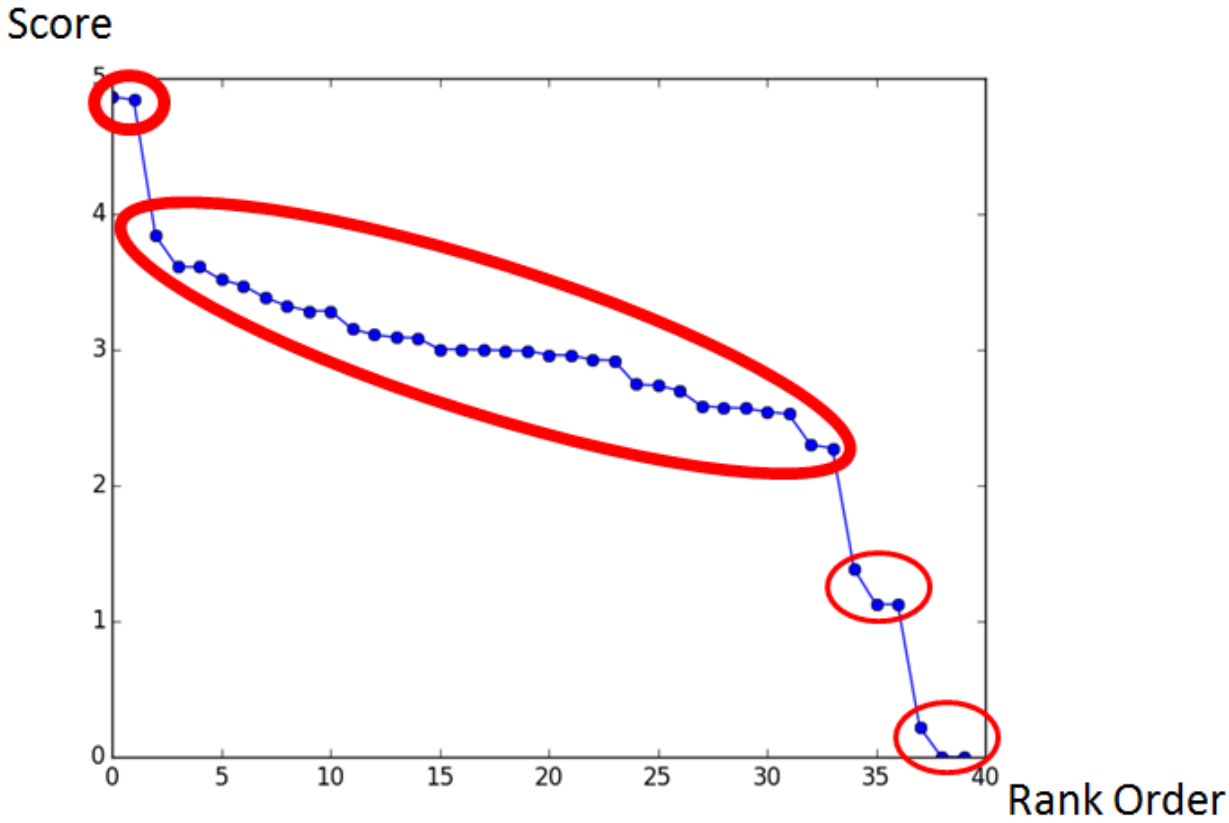
$$\begin{aligned} I_j \cdot T_Q &= \{M_1^T D_j', M_2^T D_j''\} \cdot \{M_1^{-1}Q', M_2^{-1}Q''\} = (M_1^T D_j')^T (M_1^{-1}Q') + (M_2^T D_j'')^T (M_2^{-1}Q'') \\ &= D_j' \cdot Q' + D_j'' \cdot Q'' = D_j \cdot Q \\ &= (D_j, s_{j1}, s_{j2}, \dots, s_{jU}, 1) \cdot (rQ, r\varepsilon_1, r\varepsilon_2, \dots, r\varepsilon_U, t) = r\left(D_j Q + \sum_u s_{ju}\varepsilon_u\right) + t \\ &\propto TFIDF(D_j, Q) \end{aligned}$$

此加密搜尋概念的安全性基於，即使第三方因窺視通訊管道而得到 $M_1^T D_j'$，但因並不知道加密鑰匙 M_1 和加密時經加密鑰匙 S 所切割出的 D_j' 的內容。因此藉由 $M_1^T D_j'$ 所含有的 2(n+U+1)個方程式的解，並不足以回推出 D_j' 和 M_1 中所含有的(n+U+1)和$(n+U+1)^2$個未知數。為方便用於我們之後所提出的搜尋模式隱藏的基本概念，我們將此方法做些微調整。首先，我們保留於搜尋關鍵字中加入無意義關鍵字的作法，以讓同一組關鍵字每次皆可隨機產生不同的 Trapdoor。但並不會在文件索引中加入對應的無意義字詞，即將所有無意義字詞的權重設為 0，以避免在無法調控排名品質的情況下，影響所計算出的排名結果。最後，因已藉由此方法讓每次所產生的 Trapdoor 都有所不同，是故我們將不使用 r 和 t，對排名成績進行相對放大與位移，以得到各文件的絕對相關成績值。

3.2 基本概念

我們所提出的搜尋模式隱藏方法主要概念的背景如下。對於雲端伺服器，其所能知道的資訊越少越好；因此，其只會知道計算後的原始排名次序與各文件所計算出的相關分數值，

這兩部分資料。藉由這些有限資訊，我們嘗試在維持排名品質之下，對該原始排名結果進行擾亂，以讓同一組查詢關鍵字每次皆可得到不同的排名結果。

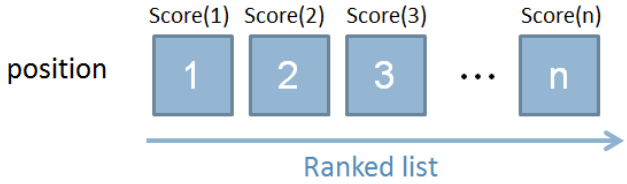


圖二：排名次序與其相關分數示意圖

由圖二我們可以看到，各文件所計算出的相關分數差距有所不同，有些文件間相關分數值彼此接近；但也有些文件相關分數差距顯著。因此我們提出假設，兩文件間分數差距越小，其和所查詢的關鍵字間的真實相關度差距應也越小，即可被視為地位相同的同儕（peer）。基於這個假設，對於圖二，我們會傾向對紅圈所圈選的文件進行彼此交換，以在盡可能維持排名品質的情況下，讓回傳結果間彼此有所差異。並且，我們另一個概念是，相較於位在後半部的文件，越在前側的文件對於排名結果間差異度的影響越高。主要原因為，由於排名結果間有重要程度的次序性；因此對於攻擊者來說，當其觀察到的排名結果差異主要集中於後側，而前幾名的文件幾乎相同時，則該攻擊者應會認為兩排名結果幾乎為相同的排名。基於這兩概念，我們開始進行以下研究。

3.3 同儕切割

首先，基於前述所提出的「當兩文件間分數差距越小，其真實相關度差距應也越小」這一假設，我們想要對同儕進行切割（Segmentation）。為此對於圖三，我們會先找出相對於整體，較為異常的分數差距（abnormal score gap）。



圖三：同儕切割符號示意圖

異常分數差距尋找方法如下：

$$g_i = |Score(i) - Score(i+1)|$$

$$\mu_{gap} = \frac{1}{n-1}\sum_{i=1}^{n-1} g_i$$

$$\sigma_{gap} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}\left(g_i - \mu_{gap}\right)^2}$$

其中 Score(i)代表的是對於位置 i 的文件所計算出的相關分數，n 則為整個排名結果中所擁有的文件數量。計算出分數差距的平均值μ_{gap}和標準差σ_{gap}後，對於大於$\mu_{gap} + k\sigma_{gap}$的$g_i$，我們將其視為異常的分數差距，並對該類差距進行同儕切割的動作。其中 k 為一實驗調控參數。

為判別切割後的品質好壞，即各 segment 內的真實相關程度是否一致，是否真的有將不同真實相關程度的文件切割至不同區塊。為此，我們對於同儕切割進行了一個實驗以作觀察，實驗所使用的資料集為 Microsoft 所提供的 LETOR 資料集[1]，該資料集內相關統計資料如下：

dataset	# queries	# documents	# relevance levels
MQ2008	784	15211	3

表一：同儕切割實驗數據

有關評估方法則使用在分群問題上，我們使用時常被使用的 Normalized Mutual

[1] http://research.microsoft.com/en-us/um/beijing/projects/letor/

Information (NMI)以進行分析，公式如下：

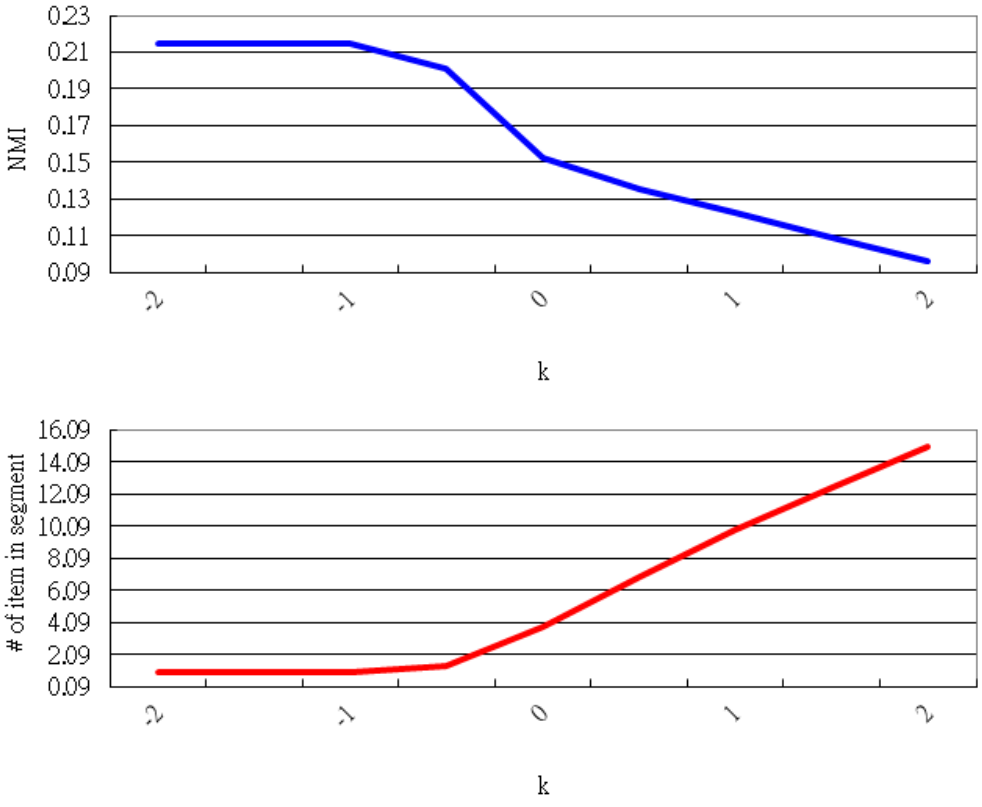
$$I(\Omega;C) = \sum_k \sum_j \frac{|w_k \cap c_j|}{N} \log \frac{N|w_k \cap c_j|}{|w_k||c_j|}$$

$$H(\Omega) = -\sum_k \frac{|w_k|}{N} \log \frac{|w_k|}{N}$$

$$H(C) = -\sum_j \frac{|c_j|}{N} \log \frac{|c_j|}{N}$$

$$NMI(\Omega, C) = \frac{I(\Omega;C)}{(H(\Omega)+H(C))/2}$$

其中 **Ω** = {w_1, w_2, …, w_K}為所切割出 segments，**C** = {c_1, c_2, …, c_J}則為文件的真實相關程度。NMI 主要概念為，除了各 segment 內的真實相關程度需一致外（即 $I(\Omega;C)$ 的部分），同時會對所切割出的 segments 總數進行處罰（即 $H(\Omega)$ 的部分）。以防止有 n 個文件就將其切割成 n 塊，即每塊中都只有一個文件。如此雖整體表現看似很好，但實為一不合理的作法的情況。經過計算後，NMI 的值會介於 0 至 1 之間，0 代表極差，1 代表極好。實驗結果如圖四所示：

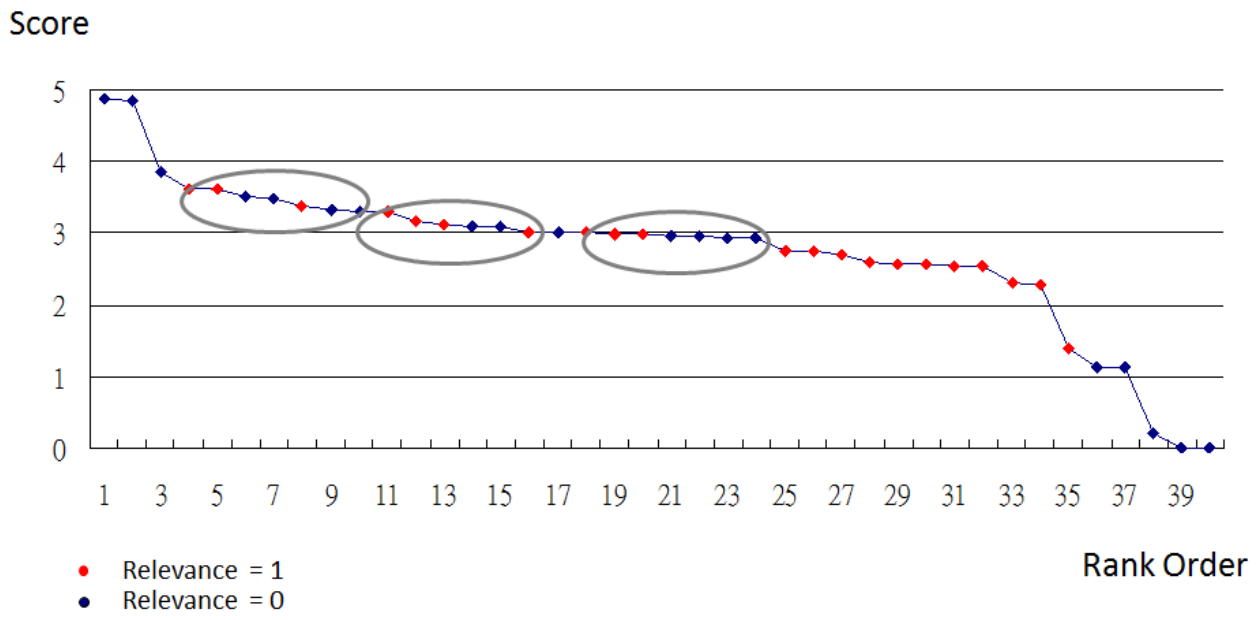


圖四：同儕切割實驗結果圖

橫軸為實驗參數值 k，縱軸分別為 NMI 和每個 segment 中所擁有的文件數量。從圖中可看出，

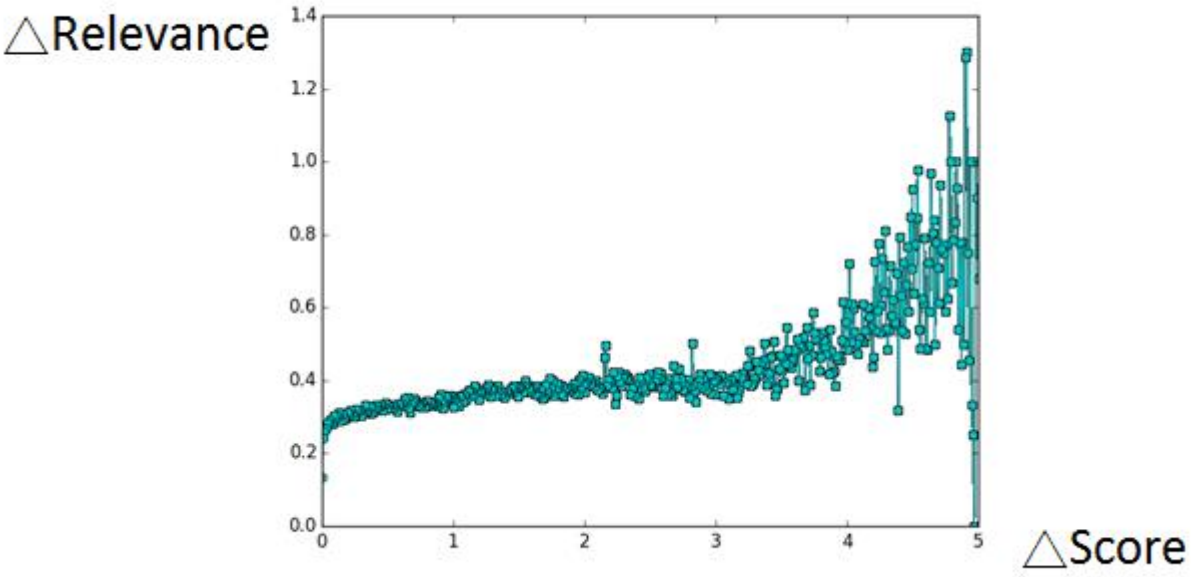
隨著 k 值的上升，即放鬆同儕切割條件，以致每區塊中有較多文件的情況下，NMI 值有下降的趨勢。反之，若將 k 值設為極小，以使 NMI 值最佳化時，也至多只能達到 0.21 左右，即很能進行很好的同儕切割動作。並且，對於每個 segment 來說，其內平均只會有 1.002 個文件。因此，若對各文件限制只能在其所屬的 segment 內進行交換的話，將造成交換自由度過低的問題。

因此，我們對 MQ2008 的排名結果進行細部觀察，以 query id 10 的回傳結果為例：



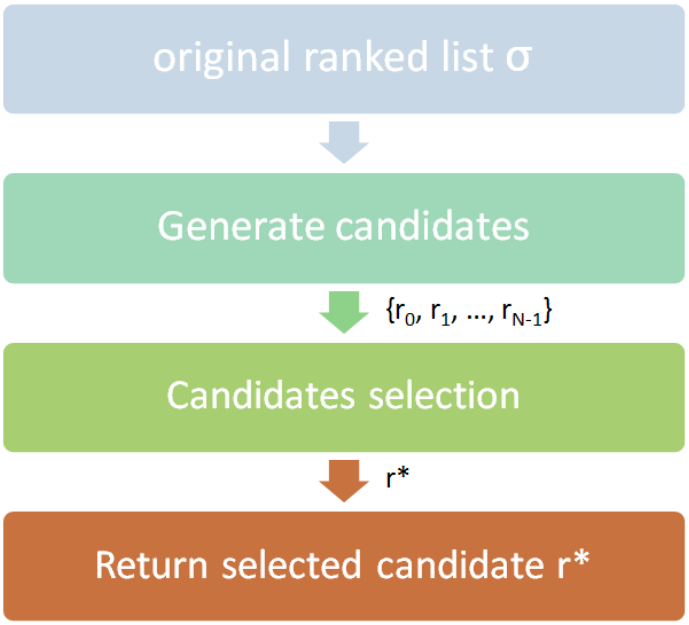
圖五：相關分數與排名結果關聯圖

由圖五中可以發現，即使兩文件的相關分數值相近，仍有可能其實為不同的真實相關程度（如圖五的灰圈所示），導致我們很難去很好的去作同儕切割的動作。但是另一方面，從圖五中也可以看出，雖然很難地去做很好的切割；但是對於相關的文件（紅點）來說，相較於其他較遠的文件，和其較近的文件和通常較有可能也是相關的文件。即由圖五中可看出相關文件（紅點）和不相關文件（藍點）間仍隱約有群聚的現象呈現。這也為我們前述的假設進行了佐證。為了更好地證實「當兩文件間分數差距越小，其真實相關度差距應也越小」這一假設，我們對資料集進行了統計分析，由圖六可看出先前的假設的確成立。因此在以下章節中我們雖不會真的對同儕進行切割，但仍保留該精神以設計後續研究方法。



圖六：相關分數差距與關聯程度差距關聯圖

3.4 基於同儕的交換方法



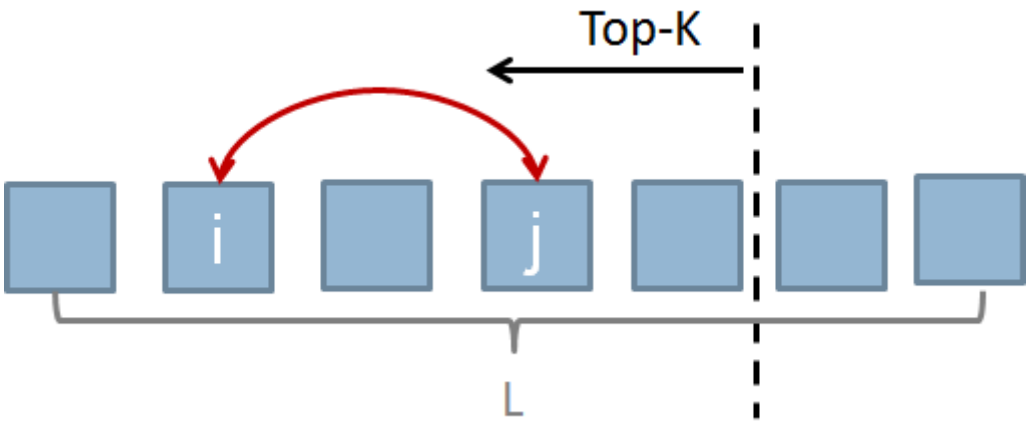
圖七：基於同儕交換方法的流程圖

我們的實驗流程如圖七所示。首先，我們會藉 3.1 章節中所提及的加密搜尋方法，以產生原始排名結果 σ。之後在盡可能保持排名品質的情況下，該原始排名結果 σ 做擾亂的動作，以產生和其有所差異的多個候選人$\{r_0, r_1, \ldots, r_{N-1}\}$。接著則在候選人$\{r_0, r_1, \ldots, r_{N-1}\}$中挑選較為

適當者 r^*，以作回傳的動作。因此，以下將對產生候選人選和挑選適當者此兩階段進行講述。

3.4.1 產生候選人

為產生和原始排名結果 σ 有所差異的候選人 r，我們會先藉由某些選擇策略以選定一個物件 i，再選定另一個物件 j，接著對彼此進行交換的動作，如圖八：



圖八：物件交換示意圖

重覆此動作 S 次後，將產生一個和原始排名結果 σ 有所差異的候選人 r_0。同樣地，重覆該動作 N 次以產生候選人$\{r_0\ r_1, \ldots, r_{N-1}\}$，其中 S 和 N 皆為使用者所設立的參數，K 代表只會回傳該排名結果中的前 K 個予使用者。比較特別的是，由於在進行交換動作時，可能產生物件交換後卻又被換回來的情形。為避免做白工的情況發生，我們會對此進行檢查，並設立一個容錯值 error_s 以盡可能滿足所需交換次數 S。相似地，由於所產生的候選人間也可能有相同的現象發生，因此也設立一個容錯值 error_n 以盡可能產生 N 為候選人選。表二為產生候選人的演算法流程。

Input σ: original ranked list S: # of swap for each candidate N: # of candidates need to be generated K: Top-K Output Candidates {r0, r1, …, rN-1}
s = 0 n = 0 error_s = 0

```
error_n = 0
candidates = {}
while n < N and error_n < N do
        r_n = copy(σ)
        while s < S and error_s < S do
                select 1st item to swap, r_n[i]
                select 2nd item to swap, r_n[j]
                if swap(r_n[i], r_n[j]) will let undo
                        error_s += 1
                else
                        swap(r_n[i], r_n[j])
                end if
          end while
          if r_n[0:K] in candidates
                  error_n += 1
          else
                  insert r_n[0:K] into candidates
          end if
end while
```

表二：產生候選人演算法

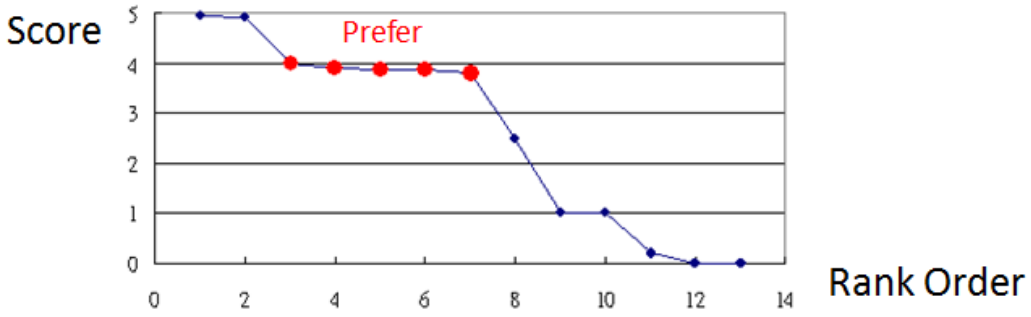
接著，我們將設計一些交換時的物件選擇策略，以對每個物件給予一對應選擇分數，當該分數越高時，越傾向於選擇該類物件。但因此過程僅為傾向而非絕對，帶有機率的概念存在，因此仍可使每次結果間有所差異。我們所提出的第一個交換物件 i 的選擇策略如下：

一、選擇交換後對整體相關程度影響較少的物件

因我們基本概念為「兩文件間分數差距越小，相關度差距也會越小」，因此對於兩物件被交換後對整個排名結果所造成的影響，令其為兩者分數差距的絕對值。並因選定第一個交換物件 i 後，其有一定機會在第二步驟中和任一物件進行交換，因此我們會將物件 i 和其他所有物件間的分數差進行加總的動作，即 Cost(i)。最後傾向於選出一個相對來說，即使和任一物件進行交換，對整體相關度影響較小的物件，即為圖九中紅點所示，以維持排名品質。此策略下的選擇分數經正規到 0 至 1 後的公式如下：

$$\text{Cost(i)} = \sum_{\forall j, j \neq i} |score(i) - score(j)|$$

$$\text{Select}_1^1(i) = \frac{Cost_{max} - Cost(i) + 1}{Cost_{max} - Cost_{min} + 1}$$

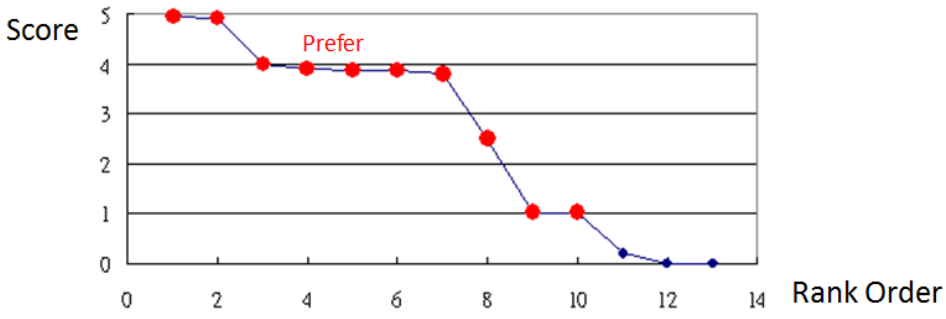


圖九：第一個交換物件的選定策略一示意圖

二、選擇位在前 K 名的物件

由於最後只會回傳前 K 名的排名結果予使用者知道，換言之，只有前 K 名的物件有機會被第三方窺探和攻擊。因此我們會傾向於選定前 K 名，以作為第一個被選定的物件，並對其進行交換的動作，當 K 值設為 10 時，即為圖十中紅點所示。此策略下的選擇分數經正規到 0 至 1 後的公式如下：

$$\text{Select}_2^1(i) = \begin{cases} 1 & i < K \\ 0 & \text{i} \geq \text{K} \end{cases}$$



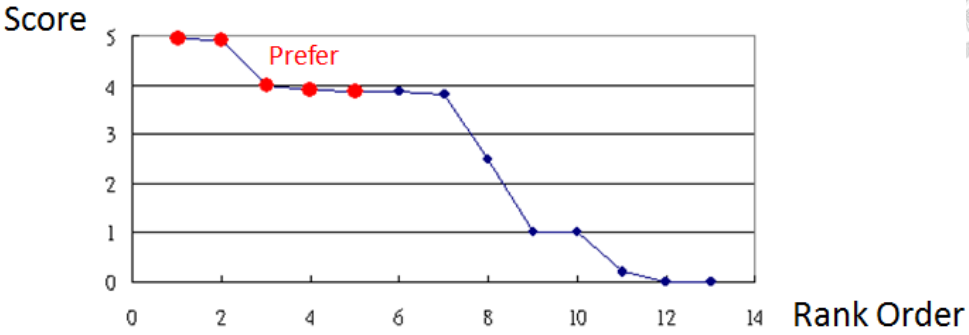
圖十：第一個交換物件的選定策略二示意圖

三、選擇位於前側的物件

回歸 3.2 章節中所提出的第二個概念，對於兩排名結果來說，由於其本身所帶有的次序性質，差異越發生於前側，對整體差異度影響也就越大。亦即排名結果中的前側變動較易混淆、影響第三方攻擊者的判斷。相反地，當變動較發生於後方時，因對於攻擊者，相對來說較不在意後方的變化，而仍會覺得兩排名結果間具有一定關聯程度。因此我們傾向於選擇位於前側的物件，以作為第一個交換物件，即為圖十一中紅點所示。此策略下的選擇分數經正

規到 0 至 1 後的公式如下，其中 L 為排列結果中的物件總數：

$$\text{Select}_3^1(i) = \frac{L-i}{L}$$

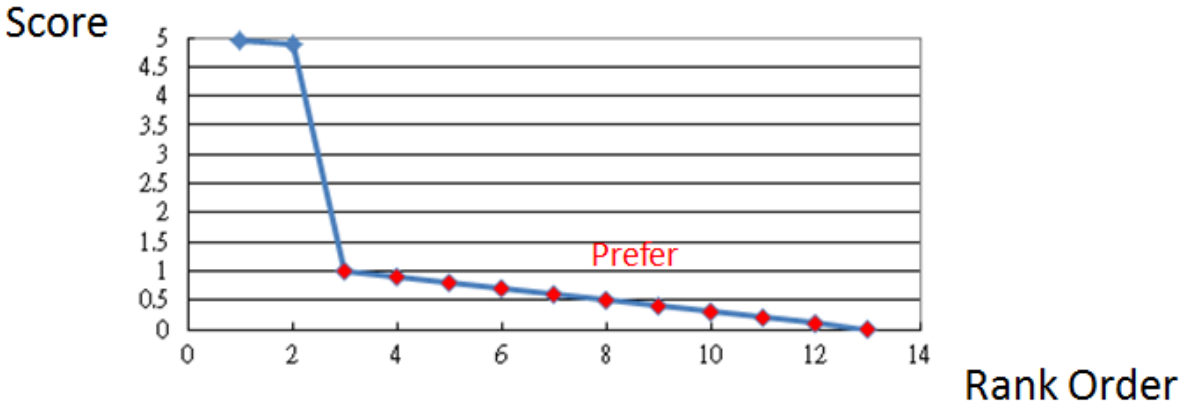


圖十一：第一個交換物件的選定策略三示意圖

四、選擇前側分數已累積足夠的物件

對於某些物件來說，在其位置之前可能已累積了足夠量的相關分數，換言之，大部分的排名品質都由位於其位置之前的物件所貢獻。倘若變動該類物件，應也不會損傷過多排名品質。因此，我們也會傾向於選擇前側分數已累積足夠的物件，即為圖十二中紅點所示。其中較特別的是，該類物件並不一定只出現於後側區域，前側也有可能出現，圖十二即為一例。此策略下的選擇分數經正規到 0 至 1 後的公式如下，其中 L 為排列結果中的物件總數：

$$\text{Select}_4^1(i) = \frac{\sum_{j<i} score(j)}{\sum_{k<L} score(k)}$$



圖十二：第一個交換物件的選定策略四示意圖

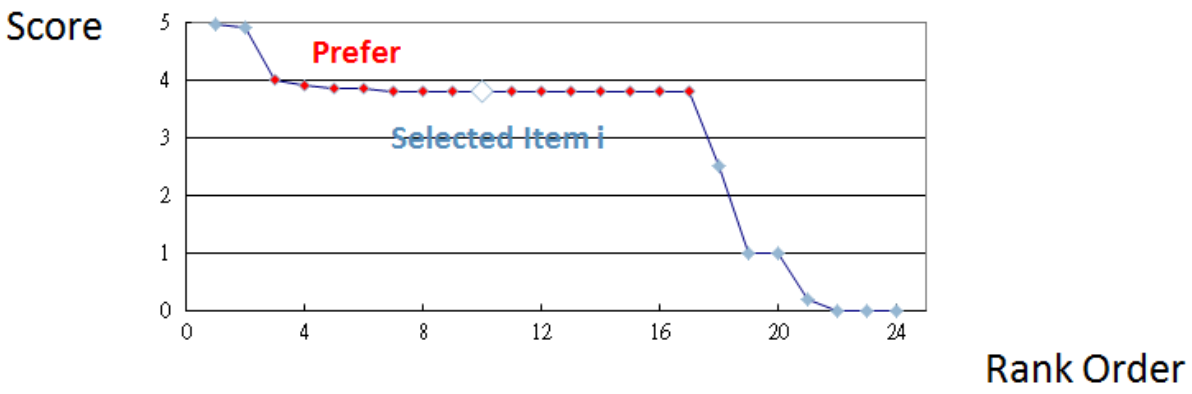
在選定了第一個交換物件 i 後，接著進行第二個交換物件 j 的選定策略討論：

一、選擇相較於第一個交換物件來說相關分數較近的物件

基於前述所提出的基本概念，對於分數差距較小的文件，其真實相關程度差異應也會較小。若將所選定的第一個交換物件對其進行交換，應不會損傷太多整體排名品質。因此，我們會傾向於選擇圖十三中紅點所示的物件。此策略下的選擇分數經正規到 0 至 1 後的公式如下：

$$\text{Select}_1^2(j) = \frac{\alpha}{|score(i) - score(j)| + \alpha}$$

其中α為一調控分數差距對選擇分數重要度的參數，以避免因過於選擇離自己分數接近的物件，而導致交換上的自由度過低，而無法作太多交換的動作。當α值越大時，分數差距對選擇分數的重要度越小。



圖十三：第二個交換物件的選定策略一示意圖

二、選擇相較於第一個交換物件來說位置距離較遠的物件

若選定了第一個交換物件後，只對其和鄰近位置的一、兩個物件做交換的動作，則因此交換距離不大，對於攻擊者來說幾乎可將此變動忽略不計。因此，我們會傾向於選定和第一個選定物件位置距離較遠的物件，即圖十四中紅點所示，以和第一個選定的物件做交換，以造成肉眼可辨別的變動程度。需特別提及的是，此策略和上一選擇策略間彼此並不互斥，即可能存在有相關分數接近，但位置距離遙遠的物件。亦即兩策略是可以同時並行的。此策略下的選擇分數經正規到 0 至 1 後的公式如下，其中 L 為排列結果中的物件總數：

$$\text{Select}_2^2(j) = \frac{|i - j|}{L - 1}$$

圖十四：第二個交換物件的選定策略二示意圖

3.4.2 挑選適當者

在產生了候選人後，我們需從中挑選較為適當者，以當作我們最後回傳予使用者的排名結果 r*。有關這部分的選擇方式如下：

$$r^* = \arg\max_{r} \left[\text{Performance}_k(r,\sigma) - \lambda \text{Correlation}_k(r,\sigma) \right]$$

trade-off coefficient
candidate
original one
Ranking performance
Top-K
Ranking correlation to original one

即找出一個同時最大化排名品質（Performance），且最小化和原始排名間關聯程度（Correlation）的候選人。其中 λ 為一可調控的平衡參數。接著將對排名品質和關聯程度計算這兩部分進行討論：

一、排名品質

目前對於排名品質較常見的評估方法是 Normalized Discounted Cumulative Gain（NDCG），公式如下：

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel(i)} - 1}{\log_2(i+1)}$$

$$NDCG_k(r) = \frac{DCG_k(r)}{Ideal\,DCG_k}$$

其中 rel(i)代表的是文件 i 的真實相關程度。但對於伺服器來說，其只擁有各文件的相關分數，而不知道真實相關程度為多少。因此我們假設相關分數越高，真實相關程度應也越高，以提出一趨近的 NDCG 的值，我們稱其為 Blind NDCG。修改後的公式如下：

$$DCG_k = \sum_{i=1}^{k} \frac{2^{score(i)} - 1}{\log_2(i+1)}$$

$$NDCG_k(r,\sigma) = \frac{DCG_k(r)}{DCG_k(\sigma)}$$

紅圈圈起來的地方，即為我們做修改的部分。並且因為以相關分數進行計算的 Blind NDCG 的最好成績，即為將原始相關分數由高至低排序的結果σ，因此我們使用其以對 Blind NDCG 正規化至 0 到 1。

二、關聯程度

目前較常見的排名結果關聯度比較，為使用 Kendall's τ，其公式如下：

$$\tau = \frac{C - D}{N(N-1)/2}$$

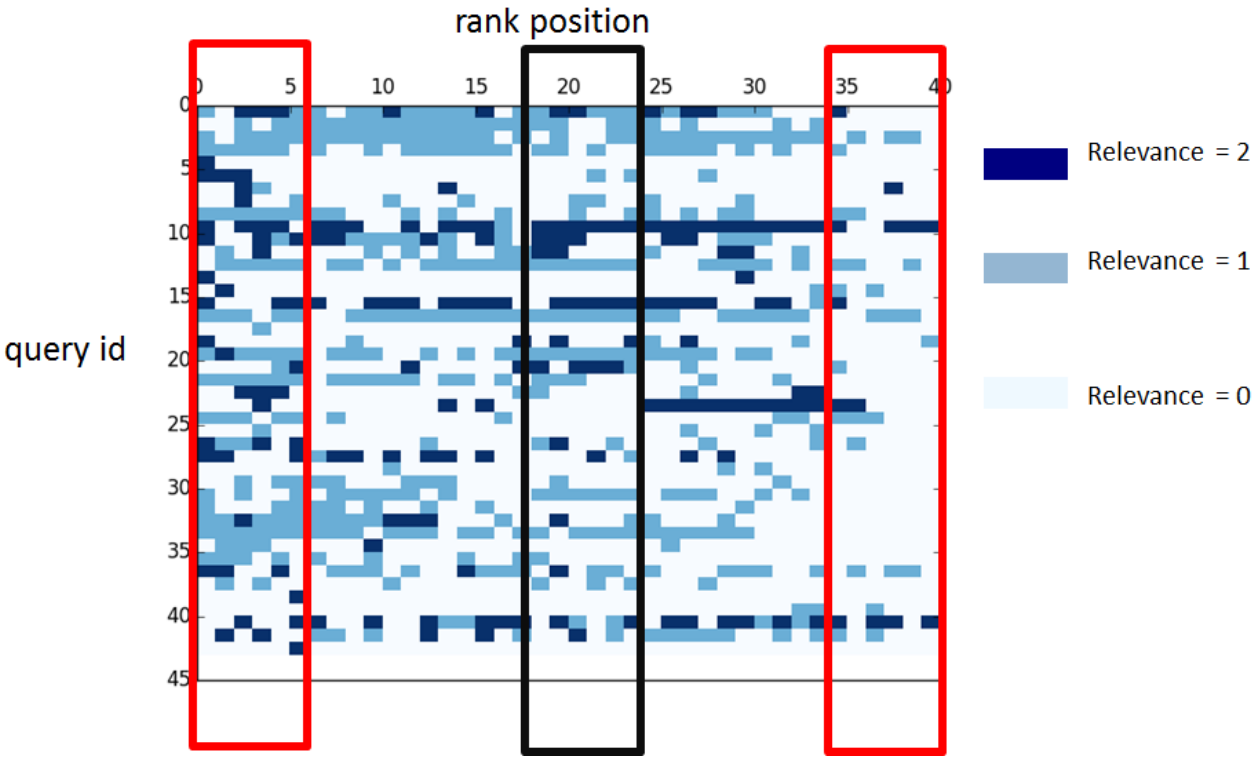
其中 N 為排名結果中的文件總數，C 為兩排名結果中先後順序一致的文件對（pair）的數量，D 則為先後順序不一致的文件對的數量。舉例來說，有兩個排名結果，分別為 12345 和 52134。對於文件對(1,5)，其在兩排名結果中的先後順序不一致。相反地，對於文件對(2,3)，其在兩排名結果中的先後順序則為一致。但此計算方法並未考慮越在前側對關聯度的重要性越高這一因素，因此 Emine Yilmaz *et al.* (2008) 對傳統 Kendall's τ 提出了修正 [10]，修正後的公式如下：

$$\tau_{AP} = \frac{2}{N(N-1)} \sum_{i=2}^{N} \left(\frac{C(i)}{i-1} \right) - 1$$

其中 C(i)為在位置 i 前的文件、和位置 i 的文件所產生的文件對先後順序為一致的數量。在此計算方式下，對於不一致情形隨機出現於各處的排名結果來說，所算出的$\tau_{AP} = \tau$。但對於不一致情形較易出現於前側的排名結果來說，所算出的關聯程度會較低，因此$\tau_{AP} < \tau$。反之則$\tau_{AP} > \tau$。由於此關聯程度τ_{AP}的計算方式較符合我們的需求，因此我們以此來當作我們挑選適當的候選者時的關聯程度計算標準。

3.5 使用者回饋機制

之後我們嘗試對整個同儕交換方法加入回饋機制。由於對於伺服器來說，其僅擁有排名結果和各文件的相關分數此兩部分資訊。除難以對系統進行回饋外，也不清楚自己所得出結果的好壞。因此，我們可請使用者對於所搜尋出的結果進行簡單回饋。首先，對於不同的搜尋模型（Retrieval Model）可能在某些位置算得比較準或不準。以圖十五為例，對於排列在前側的文件，其並不一定真的是和查詢關鍵字相關的結果；但對於排列在後側的文件，其通常都會是真的和查詢關鍵字不相關的結果（如圖十五中紅圈所示）。則若可以得到此一資訊，則可以幫助我們得知道該文件和附近文件的相關程度一致性，以減少交換時的品質損害。此外，對於某些位置來說，真正相關的文件可能易被排名於此處（如圖十五中黑圈所示）。若可以得到此一資訊，將可幫助我們在即使相關分數差距甚大的情況，仍可傾向於將一些原有預測中應為相關程度高的文件，和該區域的文件做互換。這除了提高交換時的自由度外，也將減少交換時的品質損傷。



圖十五：不同 query 在不同排名位置的真實相關程度示意圖

因此，我們將請使用者對其所得到的前 K 名排名結果標示其是否真的為相關的文件，有關這一部分，我們使用模擬使用者行為的方式以達成。在得到回饋資料後，若對各單一位置即計算出一個相關機率的話，會有 Overfit 的問題產生，因此需要做 Smoothing 的動作。對此，

我們使用了幾篇研究中所提及的不同方式進行計算，分別為 ProbFuse [11]、SegFuse [12]、SlideFuse [13]，其計算文件 d 的相關機率的公式皆如下：

$$P(d_k \mid m) = \frac{\sum_{q=1}^{Q} \frac{|R_{k,q}|}{|k|}}{Q}$$

其中 m 代表目前所使用的搜尋模型，Q 代表所下達的 query 量，k 代表文件 d 所屬於的區塊，$|R_{k,q}|$代表在區塊 k 中有多少文件和 query 真實相關，|k|則代表區塊中總共有多少文件。即為藉由文件 d 附近的文件以對文件 d 的相關機率進行 smoothing。以上三篇研究的差異主要在於區塊的定義與劃分：

一、ProbFuse：將排名結果均等切割為 25 個區塊

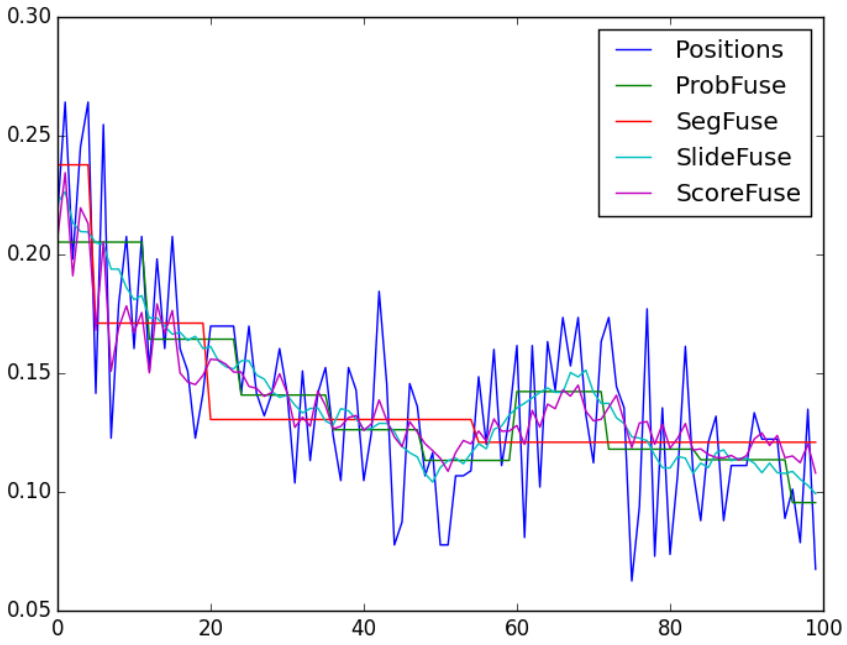
二、SegFuse：將排名結果切割為一大小呈指數上升的區塊，其中$|k| = (10 \times 2^{k-1}) - 5$

三、SlideFuse：對於文件 d 其所屬的區塊為其和其前後五筆資料，即為一活動式區塊

更進一步地，由於文件間具有相關分數的不同，因此我們提出了另一種劃分方法 ScoreFuse。其計算方式為將區塊的概念延伸至整個排名結果，即|k| = N，但在計算文件 d 的相關機率時，會對同區塊內的文件 j 授予權重值：

$$\frac{\alpha}{|score(d) - score(j)| + \alpha}$$

以讓和文件 d 分數差距小的文件，對文件 d 的相關機率所造成的影響程度較高。相對於單一位置（Positions）經過 smoothing 後各位置相關機率如圖十六所示：



圖十六：不同的 smoothing 結果

在計算出各位置的相關機率後，我們使用此機率以計算各不同位置的熵（Entropy），公式如下：

$$Entropy(d_k) = \sum_{i=0}^{2} P(d_k = i \mid m) \log_2 \frac{1}{P(d_k = i \mid m)}$$

對於熵較低的文件來說，其與附近位置的相關度應較明確，因此若選擇其進行交換應較不會傷害整體排名品質。舉例來說，對於真實相關程度為 0122110012 和 1122111121 兩區塊來說，後區塊的熵較低，即附近文件的相關度較為明確。因此，若選擇後一區塊中的文件進行交換，則對於整體排名品質的傷害度應會較低。因此，我們將原先所提出的第一個物件選擇分數修正為：

$$Select^1(d_k)' = Select^1(d_k) \cdot \left(\log_2 3 - Entropy(d_k) + 1\right)$$

即對位於較低熵的區塊的文件進行選擇分數的放大，以更趨向於選擇此類置換後較不損傷整體排名品質的文件。

第四章　　實驗

4.1　　實驗資料

我們實驗所使用為 Microsoft 所提供的 LETOR 資料集中的 MQ2007 和 MQ2008 此兩類資料集，該兩類資料集內的統計資料如下：

dataset	# queries	# documents	# relevance levels
MQ2008	1692	69623	3
MQ2008	784	15211	3

表三：實驗數據

4.2　　Baseline

由於就我們所知，目前未有研究進行維持排名品質下，對排名結果進行擾亂的實驗。因此，我們提出三個較為啟發性（Heuristic）的方法以作為和我們比較的 Baseline。其分別為：

1. Random：隨機對文件間進行兩兩互換
2. Top：為急速增加和原始排名間的差異程度，因此傾向於將排列於前側的文件換至別處
3. Gaussian：對每一文件皆設置一小的高斯函數 N(5, 0)，以讓其傾向於和附近的文件互換，以維持排名品質。

4.3　　評估方法

因本研究的研究目的為，在維持排名品質下，對同一組關鍵字查詢盡可能產生有所不同的排名結果。因此我們使用了 3.4.2 章節中所提及的 Normalized Discounted Cumulative Gain（NDCG）、傳統的 Kendall's τ、經修改後的τ_{AP}，此三者以進行結果好壞的比較。並且因為於 3.4.2 章節中已列出該三者的詳細公式內容，故不在此進行贅述。

唯由於 Shengli Wu *et al.* (2003)提出，無論各位置如何排列，只要以集合的觀點來看為相同的東西，則兩者應要被視為相同的排列結果 [14]。因此我們也會使用集合重疊係數（Overlap coefficient）以當作評斷標準，其公式為：

$$\mathrm{Overlap(X,Y)} = \frac{|X \cap Y|}{\min(|X|, |Y|)}$$

4.4 實驗結果

各項預設參數的設定如下：

i. 重複實驗五次，並取其平均值

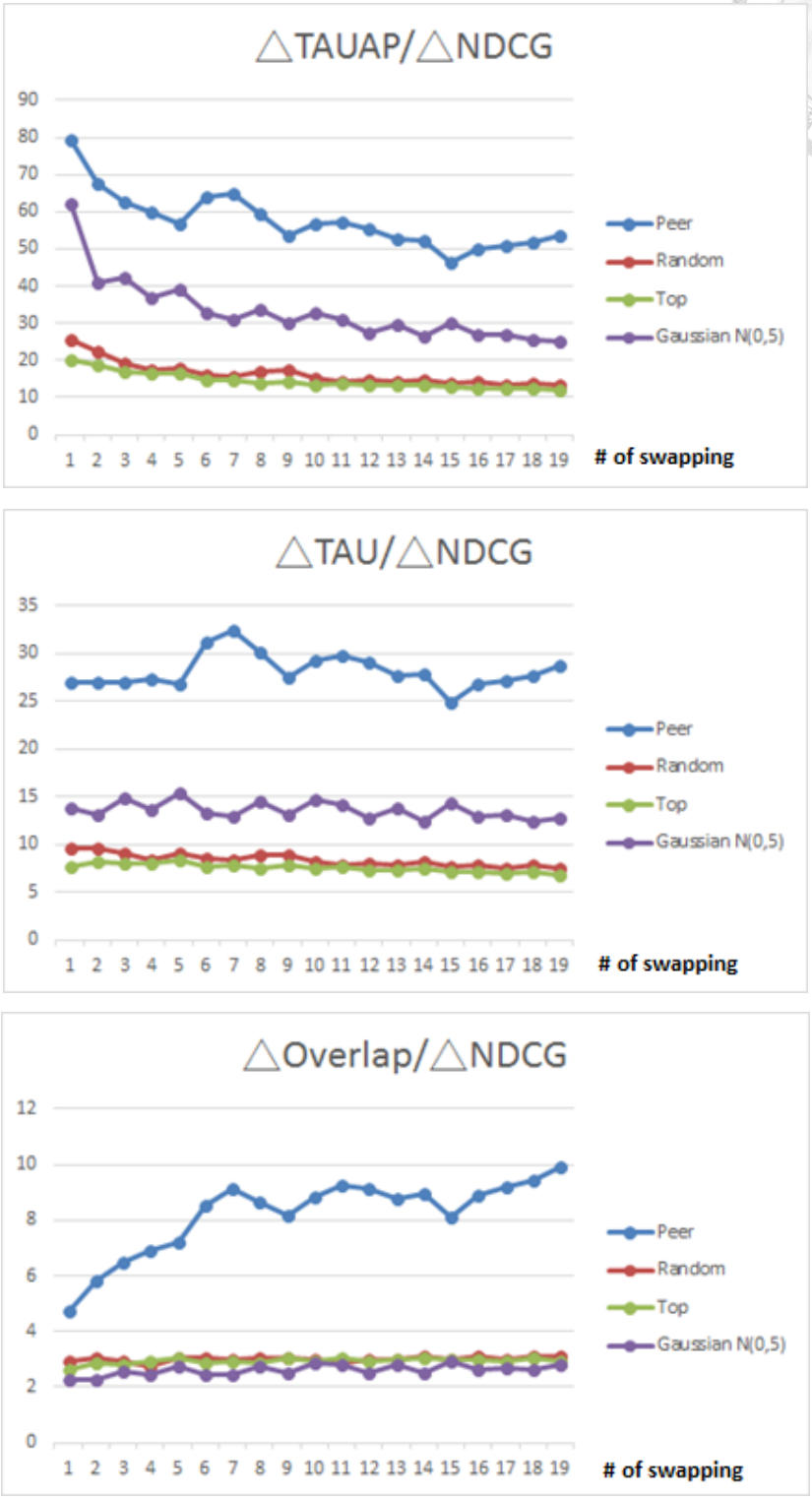
ii. 對於每個原始排名結果，會產生 20 個可能候選人，即 N = 20

iii. 對於所產生的最終排名結果，只會回傳前 20 名名，即 Top-K = 20

iv. 候選人的排名品質和其與原始排名間的差異程度重要性相同，即 $\lambda = 1$

v. 對於分數差距對選擇分數的影響係數 $\alpha = 0.005$

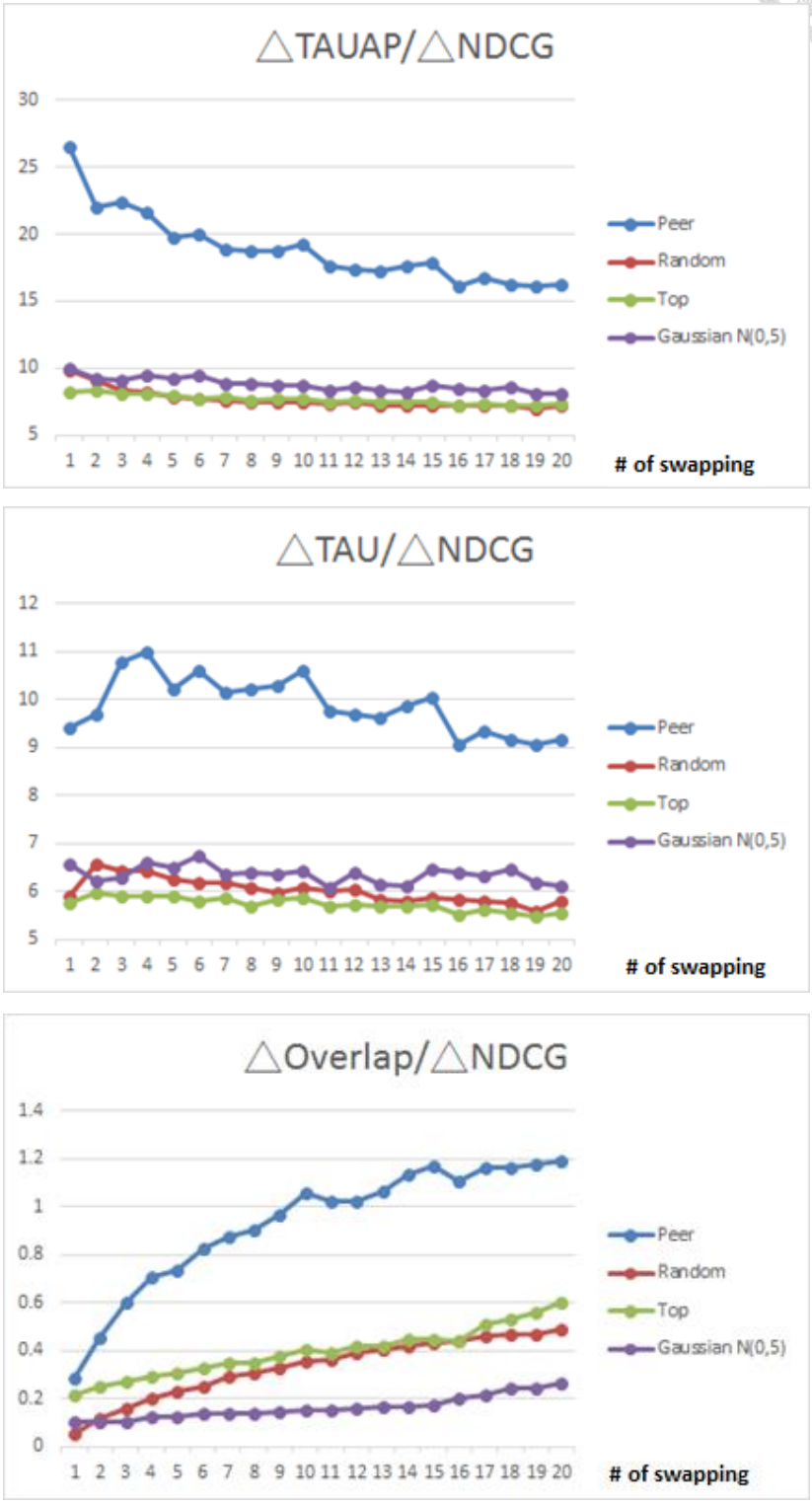
4.4.1 實驗一：與 Baseline 的比較



圖十七：實驗一結果（用於 MQ2007）

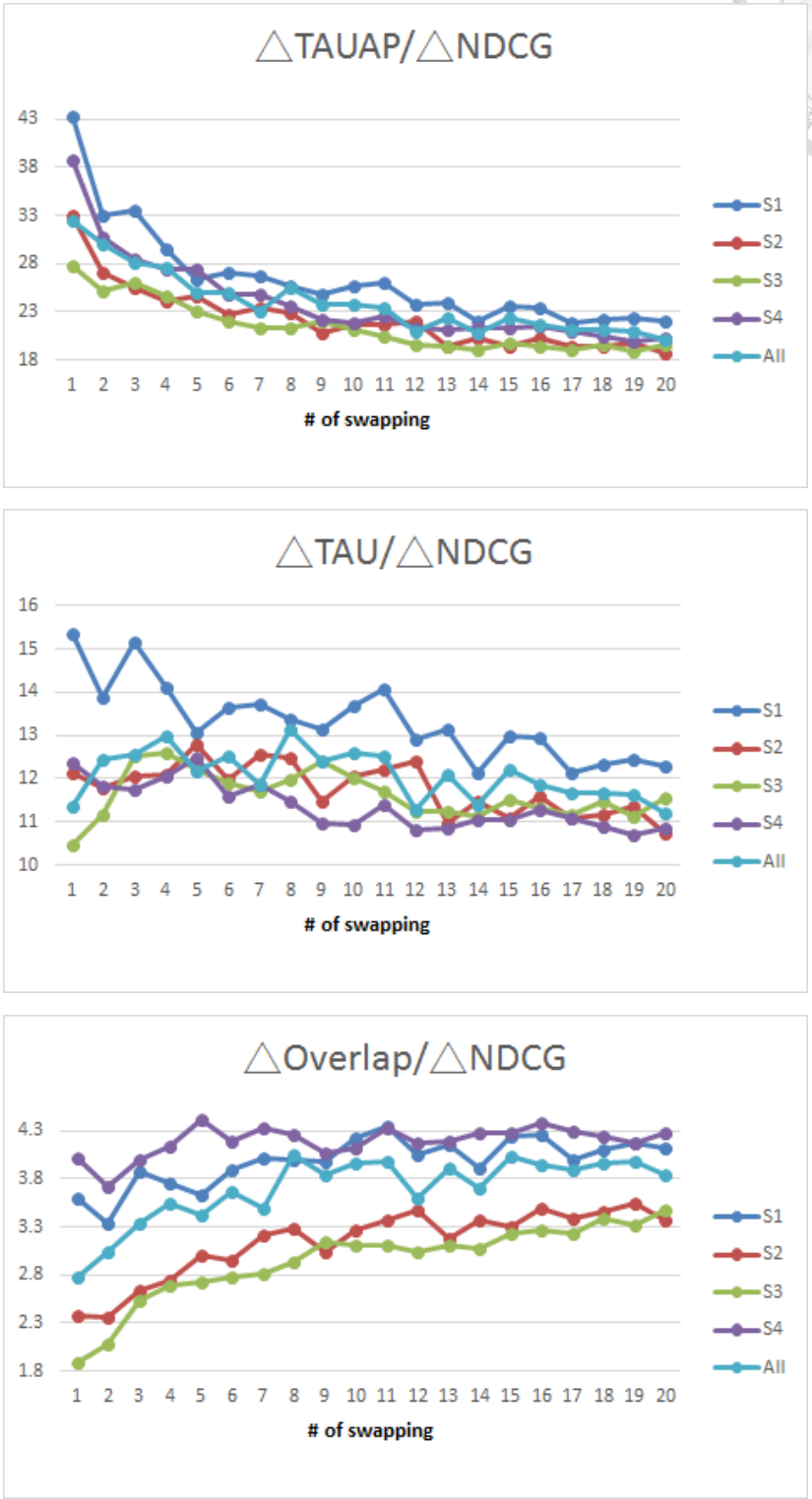
由圖十七可看出，相較於 Baseline，我們的方法在同樣的 NDCG 損失比例下，可大幅度地下降和原始排名間的關聯程度，即大幅提高了差異性。並且隨著交換次數的增加，關聯度

由一開始的大幅度下降，至後來下降比例逐漸趨於穩定。除對於 Overlap，因其將前 K 名視為一集合，當交換次數過低時，較不易有機會將前 K 名的文件換至 K 名之外。需隨著交換次數的增加，關聯度下降幅度才會開始上升。圖十八也可觀察出同樣現象。



圖十八：實驗一結果（用於 MQ2008）

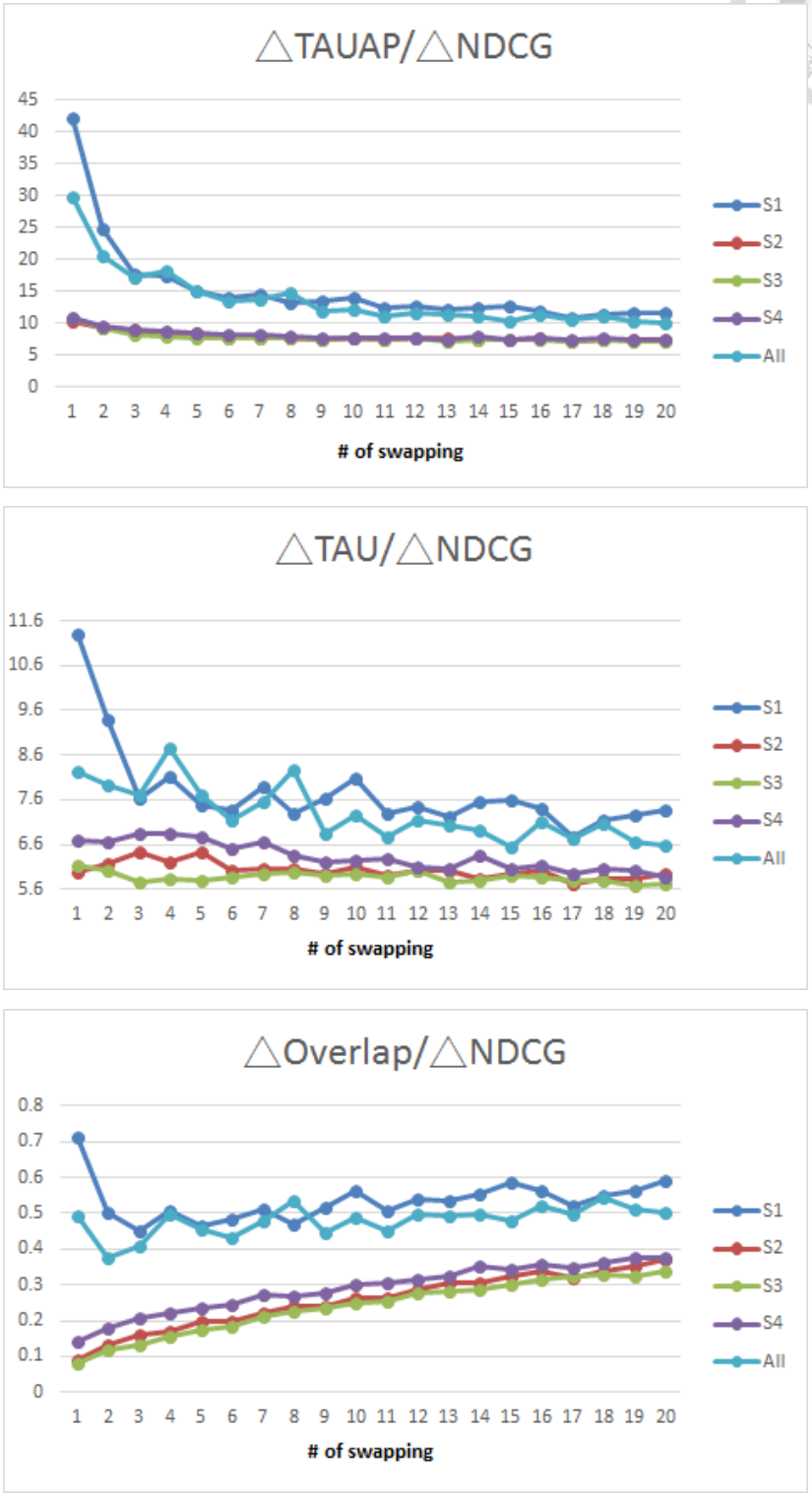
4.4.2 實驗二：第一個交換物件的策略比較



圖十九：實驗二結果（用於 MQ2007）

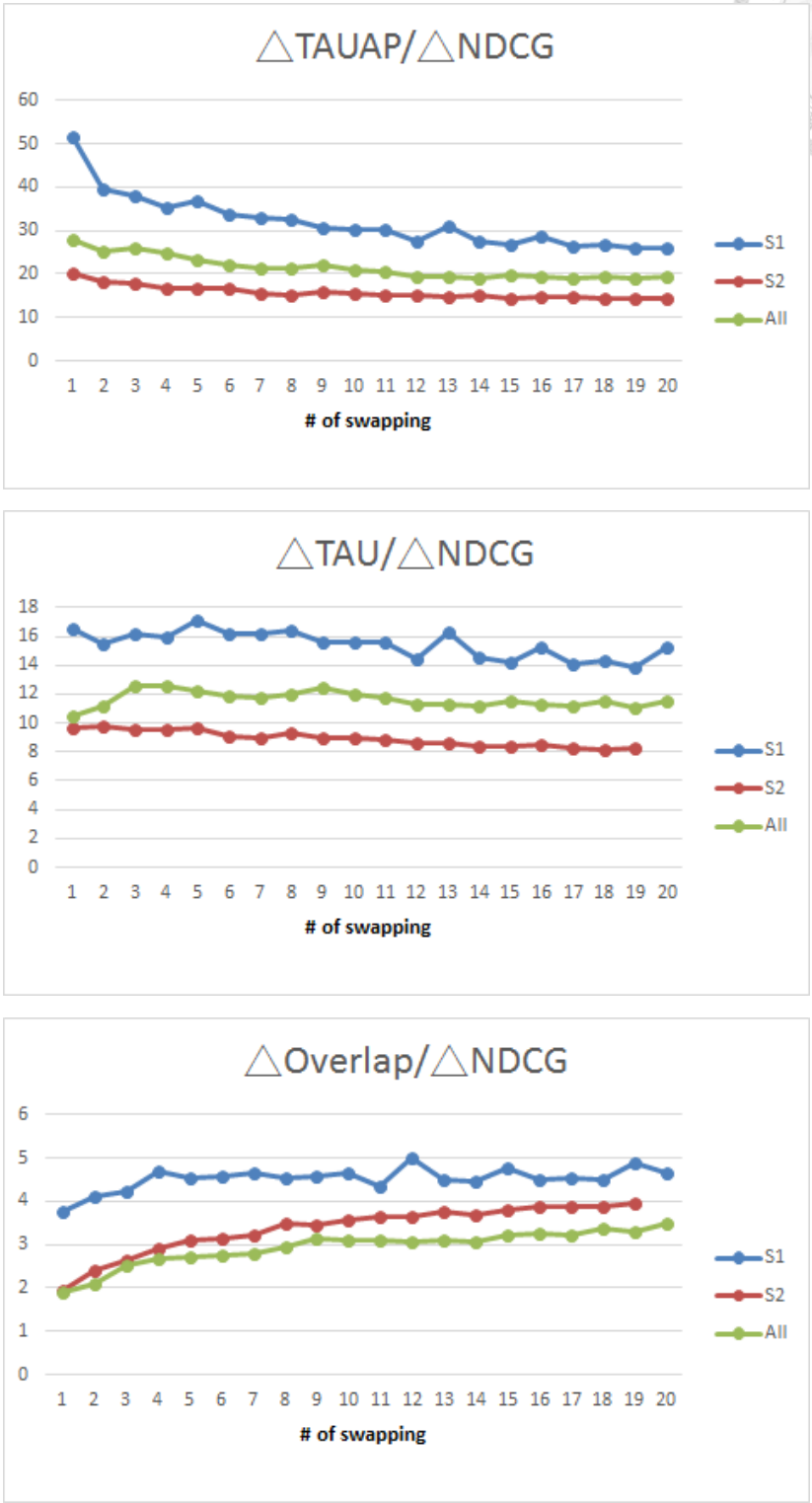
由圖十九可看出，相較於其他第一個物件的選擇策略，在三種不同關聯度比較中整體來看，第一個物件的選擇策略一（S1）和將四項策略皆正規至 0 至 1 後，進行合併（All）的選

擇策略較好。可能原因為 S1 在考慮相關程度的總變動量底下，將可選定一可和較多其他物件交換的物件，以增加自由度，但仍維持排名品質。圖二十也可觀察出同樣現象。



圖二十：實驗二結果（用於 MQ2008）

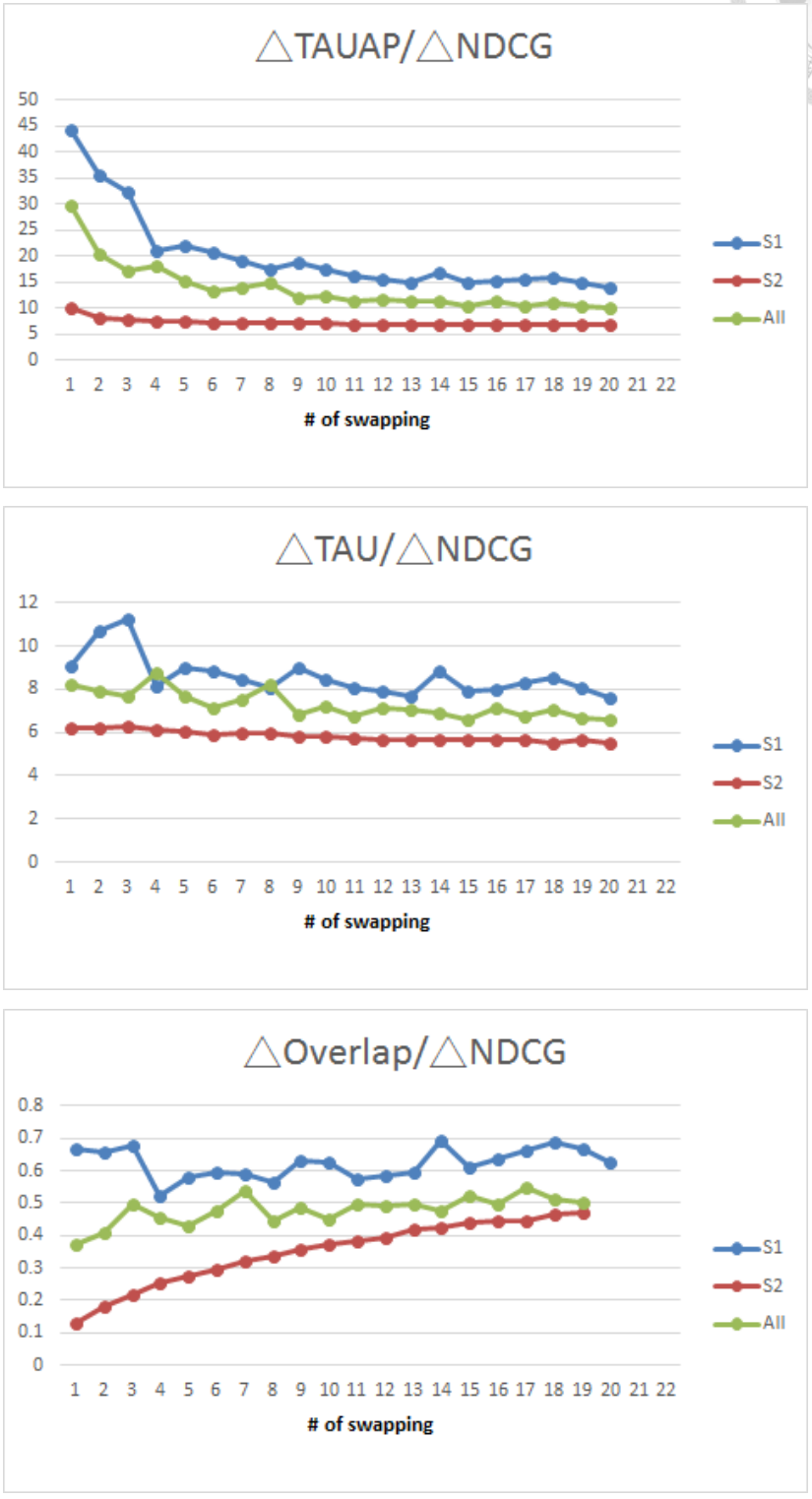
4.4.3 實驗三：第二個交換物件的策略比較



圖二十一：實驗三結果（用於 MQ2007）

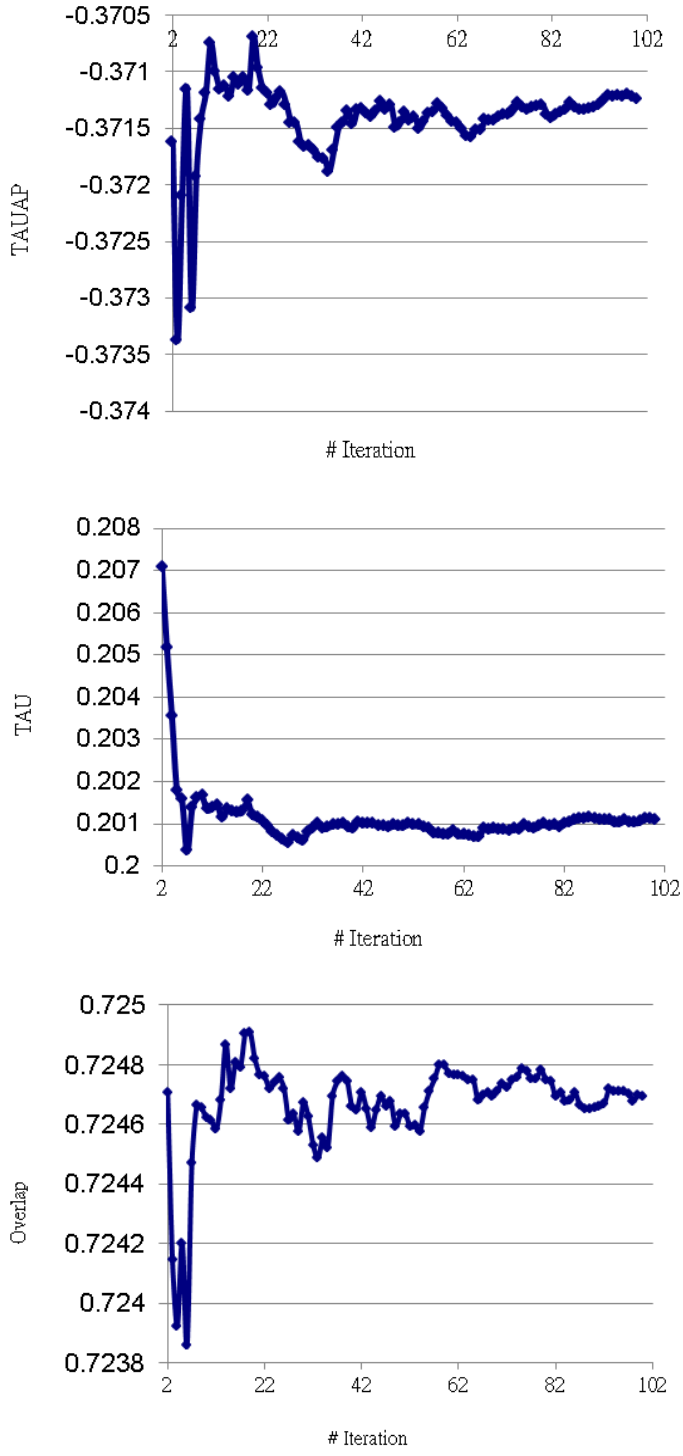
由圖二十一可看出，相較於其他第二個物件的選擇策略，第二個物件的選擇策略一（S1），即和關聯分數接近的交換較好。因如此可使 NDCG 改變量不至太大，即較可維持排名品質。

並且，此策略中雖為和分數接近者進行交換，但自由度並不低，並且三種關聯度皆可下降的至一定程度。圖二十二也可觀察出同樣現象。



圖二十二：實驗三結果（用於 MQ2008）

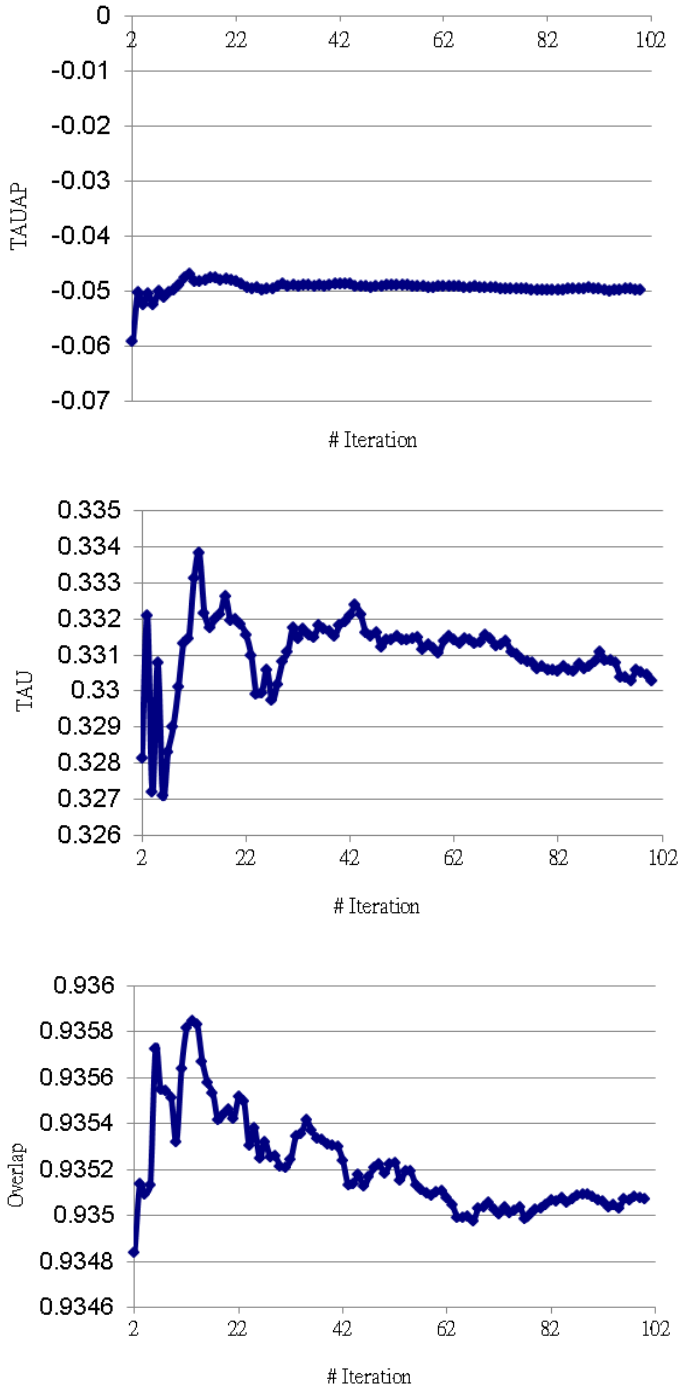
4.4.4 實驗四：搜尋結果間的關聯度變化



圖二十三：實驗四結果（用於 MQ2007）

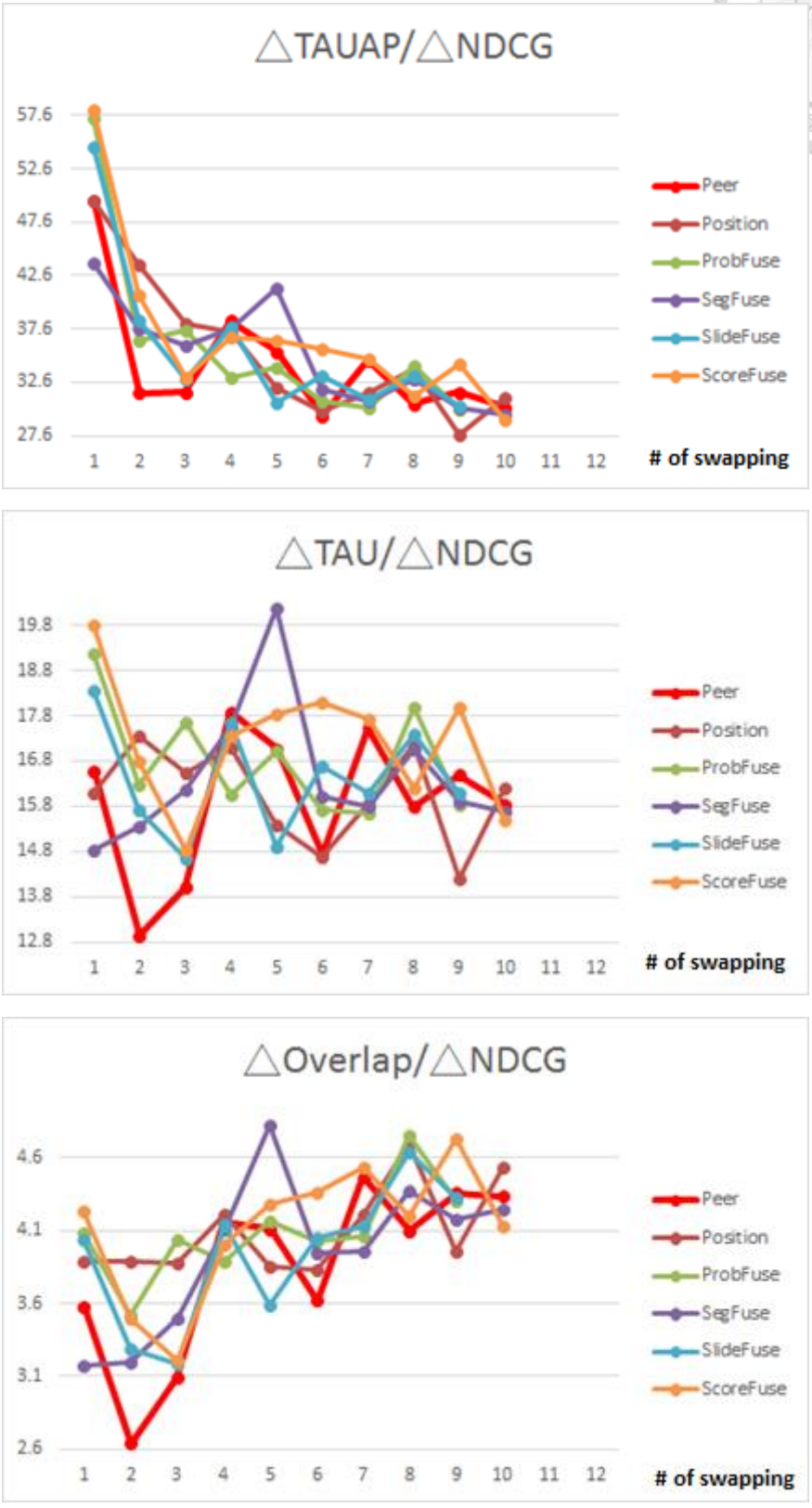
本實驗主要為研究，在短期內下達同一組關鍵字雖可得到有所差異的結果；但以長期來看，是否隨著下達次數的增加，致使所回傳的結果間開始顯現某些趨勢與關聯程度。即收集一定量的回傳結果後，彼此間將開始無甚差異。經實驗後，由圖二十三可看出，即使同一組

關鍵字詞被下達多次；但收集所有回傳結果後進行兩兩比較，關聯度仍保持至一定低值的穩定的狀態。而不會隨下達次數的上升，而有關聯度上升的現象。唯在一開始只收集少量回傳結果時，因本研究方法為使每次結果皆不同，因此仍有考慮機率的因素，致使起初為於不穩定狀態時，變動幅度較大。但隨著下達次數的增加，將能維持至一穩定低值。圖二十四也可觀察出同樣現象。



圖二十四：實驗四結果（用於 MQ2008）

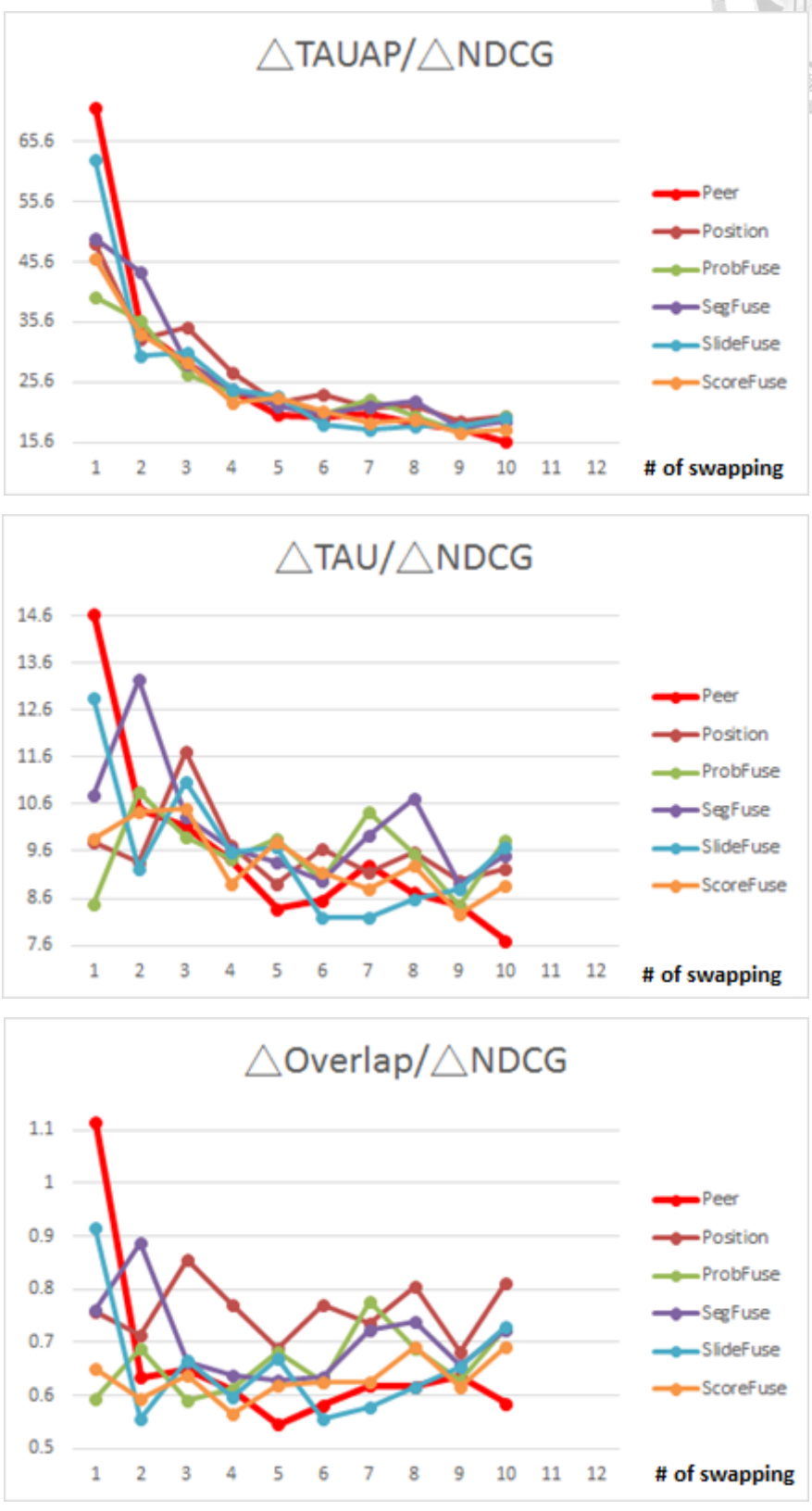
4.4.5 實驗五：加入使用者回饋機制



圖二十五：實驗五結果（用於 MQ2007）

此實驗進行在 30%訓練資料集（training datasets）中。由圖二十五可看出，相較於原先提

出的方法，加入使用回饋機制後的確有所改進。唯因整體方法中有考慮機率因素以使每次回傳結果皆不相同，故有所變動幅度。



圖二十六：實驗五結果（用於 MQ2008）

第五章　　結論與未來研究方向

本研究提出一個適用於加密搜尋環境下、有效隱藏使用者搜尋模式的方法。同時具我們所知，這也是第一個探討排名品質和排名差異度間平衡的研究。接著，在伺服器本身的條件限制下，我們提出一個基於使用者回饋的加強機制。藉由不同排名位置的文件的標記結果，加強與改善我們的系統品質。最後，我們將實驗測試在真實世界的資料中，並進行相關評估。

有關未來研究方向，由於對於不同搜尋關鍵字，亦會有不同的結果成績分布；因此我們可嘗試找出最適合該成績分布的物件交換選擇策略。此外，本研究為使用 TF-IDF 以作為檢索用途，因此我們也可嘗試找出其他亦被開發出適用於加密搜尋機制的檢索模型（例如：Okapi BM25）。如此將可在使用者下達同一組搜尋關鍵字時，系統採用不同的檢索模型進行檢索，以在不損失排名品質的前提下，得到不同排名結果。再者，除利用排名位置的標記以進行使用者回饋外，我們也可利用該標記以計算出各文件相關分數與其真實相關程度間的關係，並做更進一步的加強。最後，我們可藉由現有的一些排名彙整（rank aggregation）的方法，以對「是否當攻擊者蒐集一定量的排名結果後，即可彙整、回推出一個和原始排名結果高度相關的排序」這一問題，進行安全性的分析。

參考文獻

[1] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *IEEE Transactions. Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222-233, Jan. 2014.

[2] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," in *Communications of the ACM*, vol. 4, no. 2, pp. 84-90, Feb. 1981.

[3] G. O'Gorman, S. Blott, "Improving stream correlation attacks on anonymous networks," in *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 2024-2028, Mar. 2009.

[4] F. Wu, J. Xu, H. Li, and X. Jiang, "Ranking optimization with constraints," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 1049-1058, Nov. 2014.

[5] O. Goldreich, R. Ostrovsky, "Software protection and simulation on oblivious RAMs," in *Journal of the ACM*, vol. 43, no. 3, pp. 431-473, May 1996.

[6] X. Ding, Y. Yang, and R. Deng, "Database access pattern protection without full-shuffles," in *IEEE Transactions. Information Forensics and Security*, vol. 6, no. 1, pp. 189-201, Mar. 2011.

[7] K. Yang, J. Zhang, W. Zhang, and D. Qiao, "A light-weight solution to preservation of access pattern privacy in un-trusted clouds," in *Proceedings of the 16th European conference on Research in computer security*, pp. 528-547, 2011.

[8] P. Williams, R. Sion, and B. Carbunar, "Building castles out of mud: practical access pattern privacy and correctness on untrusted storage," in *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 139-148, Oct. 2008.

[9] E. Stefanov, E. Shi, "Oblivistore: High performance oblivious cloud storage," in *Security and Privacy, 2013 IEEE Symposium on*, pp. 253-267, May 2013.

[10] E. Yilmaz, J. A. Aslam, and S. Robertson, "A new rank correlation coefficient for information retrieval," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 587-594, July 2008.

[11] D. Lillis, F. Toolan, R. Collier, and J. Dunnion, "Probfuse: a probabilistic approach to data fusion," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 139-146, Aug. 2006.

[12] M. Shokouhi, "Segmentation of search engine results for effective data-fusion," in *Proceedings of the 29th European conference on IR research*, pp. 185-197, 2007.

[13] D. Lillis, F. Toolan, R. Collier, and J. Dunnion, "Extending probabilistic data fusion using

sliding windows," in *Proceedings of the 29th European conference on IR research*, pp. 358-369, 2007.

[14] S. Wu and F. Crestani. "Methods for ranking information retrieval systems without relevance judgments," in *Proceedings of the 2003 ACM symposium on Applied computing*, pp. 811-816, Mar. 2003.