第二章

相關研究

我們的研究目的，是找出資訊科學學術研究領域的本體論資訊，因此我們在 2.1 小節介紹 Ontology 的相關研究；學術研究的成果，都記錄在各式各樣的書目中，而資訊計量學(Bibliometrics)就是針對這些書目資訊進行研究，我們將在 2.2 小節介紹；在我們要找的 Ontology 裡面，concept 裡的 properties，代表具有影響力的論文、會議、期刊和作者，為了計算影響力指標，我們將在 2.3 小節介紹 Webpage Ranking Algorithm。

2.1 本體論

Ontology 一詞最早源自於哲學，是一種哲學上的原則，主要用來處理自然或真實世界的概念。後來被引用到資訊科學的各個領域，依據不同的需求，被賦予不同的定義，例如：

- An ontology is an explicit specification of a conceptualization.(Gruber,1993)
- A logical theory which gives an explicit, partial account of a conceptualization. (Guarino and Giaretta,1995)
- Ontologies are defined as a formal specification of a shared conceptualization. (Borst,1997)
- An ontology is hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base. (Swartout et al., 1997)

總結來說，我們可以說 Ontology 是一個：

1. 為了讓人與人，人與代理人，或代理人與代理人分享瞭解資訊的結構。

2. 為了讓專門領域的知識能被再使用(reuse)。

3. 讓專門領域裡的假定(assumption)更為明確，使溝通更容易。

Ontology Engineering 指的是一系列和 Ontology 相關的活動，包含 Ontology 的發展流程、Ontology 的生命週期和建置 Ontology 所用到的方法、工具和語言。Ontology Engineering 通常全部由專家來完成，工作量龐大且非常耗時，是一件很困難的工作。Ontology Learning 是利用電腦自動去建立 Ontology 的過程，目的是為了減輕專家的負擔。而我們的研究，找出資訊科學學術研究領域的 Ontology，可以說就是 Ontology Learning 的一種。

Ontology Learning 的方法通常是一種 bottom-up 資料導向的方式，也就是由下而上建構 Ontology，且和用來 Mining 的資料來源息息相關。Maedche [31]利用不同的資料來源把 Ontology Learning 的方式分類，主要的資料來源有 text、dictionary、knowledge base、semi-structured schemata 和 relational schemata，而其中和我們的研究最相關的是從 text 去找出 Ontology。[20]利用事先定義好的 Pattern 來找出概念之間上下義的關係。例如：The bow lute, such as the Bambara ndang, is plucked and has an individual curved neck…..。利用 such as 的 pattern，可以發現 bow lute 就是 Bambara ndang 的上義詞。同理，也可以利用相關的 patterns 來找出下義詞。而這些 patterns 的制訂，部分是透過事先定義，部分則是透過電腦自動找出來的。[15]利用 Clustering 的方法自動去建構 Ontology，將相近的概念聚集成一群，不相近的分開。而其中關於兩個概念間距離的計算，主要依照各種考量而有所不同。

另外一類的方法則是利用 data mining 裡的 association rules[30]，來找出某特定領域的 Ontology。因為 association rules 主要依據統計數據，因此可以用來找出不在分類目錄裡面的概念間的關係。主要方法為先有一個基本的，非特定領域的 Ontology，和特定領域的辭典，利用這個辭典去文件裡面找出概念，再利用 association rules 找出概念彼此間的關係，如果這個關係不在基本的 Ontology 裡面，則把它加進去，最後就可以找出某特

定領域裡面的 Ontology 了。

另一個和我們相關的研究是[39]，目的是針對特定主題，利用 CiteSeer 上的相關學術論文，找出該主題的本體論資訊。主要目的有二，一種是利用 Author Co-citation Analysis 來找出該特定主題下的子題，第二種是從論文裡面去找出特定的物件，例如方法、語言、應用程式和演算法等，透過事先定義的關鍵詞，從每個子題所包含的論文中，用自然語言的技術來找出特定的物件。例如查詢 Semantic Web，可以找出 XML、RDF 等 Language。 雖然[39]和我們都以學術研究領域為主，但仍有許多的不同，主要的不同點如下：

1. 我們要找的是具有影響力的論文、會議、期刊和作者，而[39]則是找子題和特定的物件。
2. 我們以 CiteSeer 的查詢結果為基本資料集合，再做擴充。而[39]僅以 CiteSeer 的查詢結果為資料集合。
3. 我們的系統是全自動，而[39]則是半自動。

2.2 資訊計量學

資訊計量學(Bibliometrics)利用數學及統計的計算法，對所有傳播的出版形式及其作者進行組織、分類及量化的評估[49]。最廣為人知的就是 SSCI(Social Science Citation Index)和 SCI(Science Citation Index)。以下列出資訊計量學中最常見的幾個研究主題：

- 文獻成長現象。
- 文獻老化現象。
- 布萊德福定律(Bradford's Law)：探討科學文獻的分佈情形。同一主題的文獻可能散佈在不同的期刊中，依據和主題的相關程度可以將期刊分成若干區，布萊德福定律就是研究區與區之間期刊數的比例。

- 洛卡定律(Lotka's Law)：研究作者的生產力。發表過不同文獻數目的作者群之間的關係，例如發表 x 篇文獻的作者數是只發表過一篇文獻作者數的 $1/x^2$。
- 齊夫定律(Zipf's Law)：主要在探討字彙分佈的現象，研究字彙及其出現次數之間的關係。
- 文獻引用關係。

其中，以文獻引用關係，和我們的研究較相關，以下針對文獻引用關係探討。

一篇完整的學術論文，必由正文和之後所附的參考文獻所組成。正文本身稱為引用文獻，參考文獻稱為被引用文獻。一篇文獻被另一篇文獻所引用，代表這篇文獻提供了某些有價值的相關資訊，因此可以說，被引用文獻比沒有被引用的文獻較具品質，則文獻引用分析可以視為是文獻品質分析。

把文獻看成是圖形中的點，如果兩篇文獻之間具有引用關係，則由引用文獻到被引用文獻，有邊相連，如圖 2.1，大寫字母代表文獻，小寫字母代表作者，被稱為文獻引用關係圖，例如圖 2.1 中，有邊從 A 連到 C，代表 A 文獻引用了 C 文獻。由文獻引用關係圖，可以發現文獻彼此之間具有間接關係如下：

- 書目對(coupling)：兩篇文獻具有相同的參考文獻，如圖 2.1 中的 A 和 B。

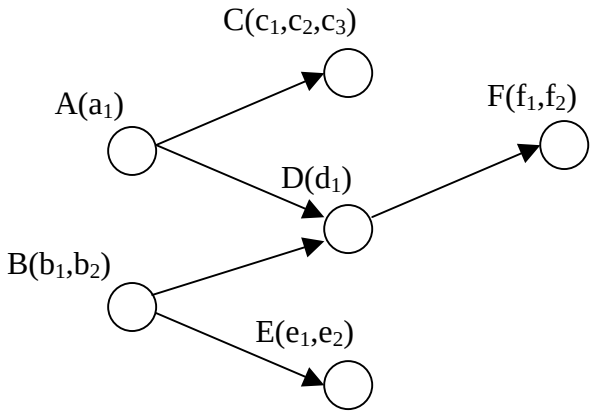

圖 2.1：Citation graph

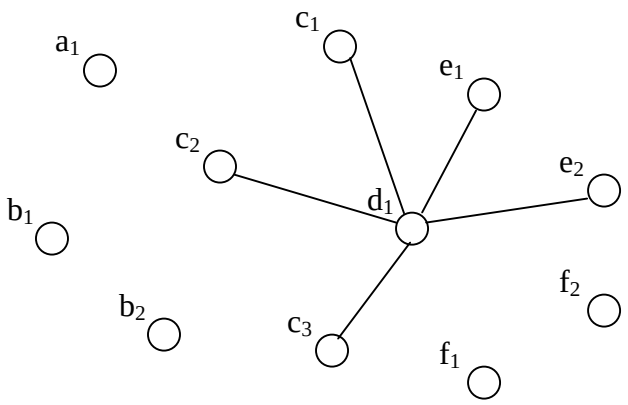

圖 2.2：Author co-citation graph

- 共被引(co-citation)：兩篇被引用文獻具有相同的引用文獻，如圖 2.1 中的 C 和 D、D 和 E。

[46]研究文獻引用關係圖，把整個引用關係圖塑造成網路系統，並特別針對書目對和共被引這兩種關係去分析。

由文獻引用關係圖，可以推出作者共被引關係圖。作者共被引關係(Author Co-citation Analysis)指的是，當兩篇文獻具有共被引的關係時，則他們的作者也具有共被引的關係，如圖 2.2。假設 a_1 是圖 2.1 中文獻 C 的作者，a_2 是文獻 D 的作者，因為 C 和 D 是共被引的關係，因此 a_1 和 a_2 也具有共被引的關係，所以 a_1 和 a_2 中間有邊相連。當一群作者藉由共被引關係相連，則代表這群作者可能研究某一個共同的領域，如圖 2.2 中的 c_1、c_2、c_3、d_1、e_1 和 e_2。而位於中心的 d_1，可能具有每種特殊意義。[45]將這種因引用關係相連的作者群，每一群稱為一個 Invisible College。而在[44]裡，則對作者共被引關係作更深的研究和分析。

引用關係的另一個研究是時間趨勢，藉由找出趨勢來預測未來的發展。[13]研究不同國家在過去的時間裡面，不同領域的論文在不同時間的發表數量。[24]研究文獻探勘和引用關係的結合，[43]則把文獻引用關係套用到 WWW 上去分析研究。

2.3 網頁排序演算法

網頁分數的計算方式，主要可以分成兩大類：

1. Keywords-based：利用網頁中所包含的關鍵字去計算分數。
2. Link-based：利用網頁的引用關係去計算分數。

網頁間的連結關係，是由網頁創造者所制訂的，所以可以以人的角度提供資訊，當 A 網頁連結到 B 網頁時，代表 A 網頁覺得 B 網頁具有某一部份的重要性，所以才會連結到 B 網頁，換句話說，就是具有語意上的資訊，而不再是單純語法上的資訊。因此以下將以 Link-based 的演算法為討論。這一類的演算法，主要可以分成兩種，PageRank[9]和 HITS[23]，原理將分別簡述如下。

1. PageRank：PageRank 的原理，簡單來說，當網頁 A 連結至網頁 B 時，則視為網頁 A 投給網頁 B 一票。所以網頁 B 的重要性，來自於所有連結到它的網頁，定義如下：

   Definition: An important page is

   1. linked by another important pages or

   2. linked by many pages.

   也就是說，重要的網頁會被重要的網頁連結到，或者是被很多的網頁連結到。所以一個網頁的重要性，來自於所有連結到這個網頁的網頁的重要性，基本原理如圖 2.3[33]。公式如下：

$$PR(p) = c \sum_{q->p} \frac{PR(q)}{Out(q)} \tag{2.1}$$

   p 代表網頁，q 代表連結到 p 的網頁，$PR(p)$代表網頁 p 的重要性，$Out(q)$代表從網頁 q 中連結到別的網頁的連結數量，c 是正規化的參數，目的是讓所有網頁分數平方和等於 1。上述的算式在下面的情況會發生問題，當有幾個網頁形成一個集合，彼此之間存在著連結，集合外面存在著網頁連結到集合裡面的網頁，但是集合裡面

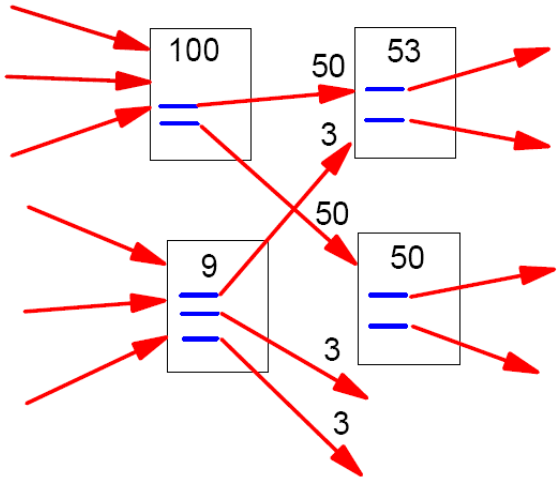

圖 2.3：PageRank 基本原理

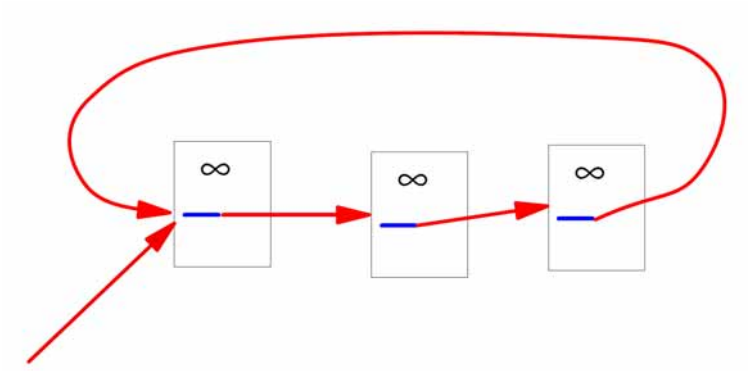

圖 2.4：連結迴圈

卻沒有連結到集合外面的其他網頁，如圖 2.4[33]，在這種情況之下，集合內的網頁分數會不斷的累加，但是卻沒有辦法分享給集合外面的網頁。針對這種情況，因此加入了另一個參數 $E(u)$，u 代表其他的網頁，$E(u)$代表到其他網頁的機率。如果落入迴圈的話，就可以挑其他的網頁，避免不斷累加造成的錯誤。公式如下：

$$PR(p) = c\sum_{q->p}\frac{PR(q)}{Out(q)} + cE(u) \tag{2.2}$$

如果以機率分佈的角度來詮釋，上面的公式可以改寫成：

$$PR(p) = (1-d) + d\sum_{q->p}\frac{PR(q)}{Out(q)} \tag{2.3}$$

d 代表沿著連結的機率，所以有 *1-d* 的機率會跳出迴圈。PageRank 針對 WWW 上所有的網頁作排序，把所有的網頁置於同一個排序標準之下，沒有針對不同主題另外去排序，當有一個詞彙同時代表兩個不同主題時，只有最常被連結到的那一個主題的相關網頁會排在最前面，而比較罕見的那一個主題的相關網頁就會變的越來越難找到。除非可以下更細部的查詢，才有查到的機會。

2. HITS：HITS 經由觀察之後，發現有的網頁本身很重要，有的網頁則都連結到重要的網頁，但本身可能並不重要，基於這兩點，以 Authority 來代表一個網頁的重要性，以 Hub 來代表一個網頁連結到重要網頁的情形。當一個網頁被越可靠的網頁連結到的話，代表越重要，Authority 的值也會越高；而一個網頁常常連結到重要網頁的話，代表它本身很可靠，Hub 的值也會越高。以下將簡述 HITS 的方法，首先定義基本物件如下：

   - Authority pages: 被很多 Hub pages 連結到的網頁。
   - Hub pages: 連結到很多 Authority pages 的網頁。

   每個網頁都包含兩個值，一個是 Authority 的值，也就是依據連結到這個網頁的 Hub pages 所計算出來的值，如圖 2.5(a)的 *Xp*。另一個是 Hub 的值，也就是依據這個網頁連結到的 Authority pages 所計算出來的值，如圖 2.5(b)的 *Yp*。當

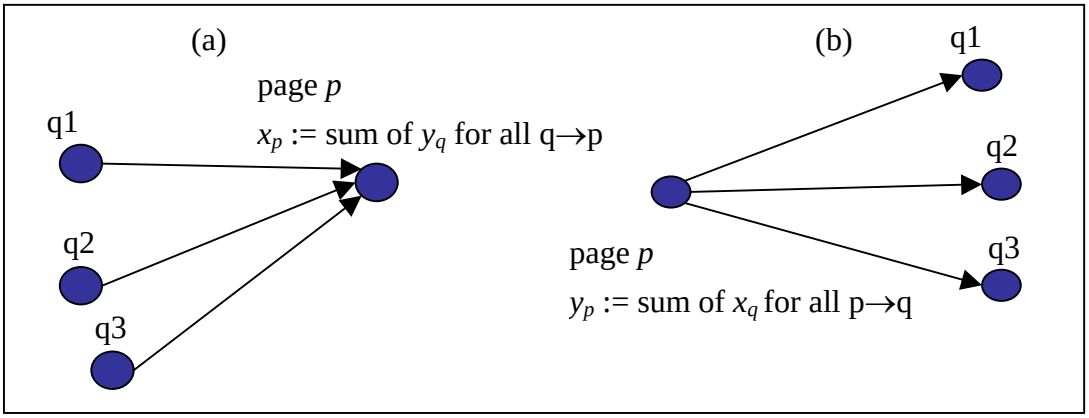

圖 2.5：Authority page(a)和 hub pages(b)

一個網頁被越多 Hub 值高的網頁所連結，則代表這個網頁的 Authority 的值越高，

也就是越重要，反之亦然。主要演算法如圖 2.6。

公式如下：

$$A(p) = \sum_{q \to p} H(q) \tag{2.4}$$

$$H(p) = \sum_{p \to q} A(q) \tag{2.5}$$

*A(p)*和 *H(p)*代表網頁 *p* 的 Authority 和 Hub 值，*O(q)*代表從網頁 *q* 連結出去的網頁數量，*I(q)*代表連結到網頁 *q* 的網頁數量。除此之外，HITS 並不是針對所有的網頁排序，而是針對特定主題的相關網頁排序。先利用一個 keywords-based 的搜尋引擎，把和這個主題相關的網頁都找出來，再利用這些網頁裡面的連結，把不在這個集合裡面的網頁找出來，加進這個集合裡面，最後再把這個集合裡面的所有網頁排序。

除了上面兩種方法之外， TimedPageRank[47]主要改進 PageRank，加入了時間的考量。公式如下：

$$PR(A) = (1-d) + d * \sum_{p_i -> A} \frac{w_i PR(p_i)}{C(p_i)} \tag{2.6}$$

$$TPR(A) = Aging(A) * PR(A) \tag{2.7}$$

A 代表網頁，*PR (A)*代表經由連結所得到的分數。*TPR(A)*代表經過時間正規化之後的分數，*d* 代表機率，p_i 代表連結到 *A* 的網頁，$C(p_i)$代表網頁 p_i 連結出去的網頁數量，w_i 是針對每一個連結的時間加權值，該值會隨著時間的增加而成指數性的減少，在[47]裡面並沒有詳細說明如何計算此數值，僅提到把指數的基底設成 0.5，如果設成 1，這部分就和 PageRank 一樣了。*Aging(A)*是經由網頁 *A* 的存在年齡計算之後所得的參數，在[47]裡面也沒有提到相關的算式，僅提到該結果介於 0.5 到 1 中間，越新的網頁越趨近於 1。因此針對[47]沒詳細說明的部分，在後面我們以符合[47]的精神的算式計算。

這些 Webpage Ranking Algorithms 雖然都是以計算網頁的分數為主，但如果把網頁改成論文，把網頁間的連結關係改成論文間的引用關係，再加上論文和其他 properties 的考量，就可以用這些 Webpage Ranking Algorithms 為基礎來解決我們的問題。

1. Obtain a set of Web pages using a keyword-based query and expand it to form a base set
2. Assign each page of the base set an initial authority and hub score of 1
3. According to its links, update the scores of each page
4. Normalize the scores so that $S(Xp)^2=1$ and $S(Yp)^2=1$ for all p in the base set
5. Do steps 3 and 4 iteratively until the scores converge

圖 2.6：HITS algorithm