國立交通大學

資訊管理研究所

碩 士 論 文

一個在不信任的環境下基於服務品質和協同過濾的

兩階段網路服務發現機制

A Two-Phase Web Service Discovery Mechanism based on QoS

and Collaborative Filtering in an Untrusted Environment

研 究 生：吳 致 衡

指導教授：羅 濟 群 教授

中華民國 九十九 年 六 月

一個在不信任的環境下基於服務品質和協同過濾的兩階段網路服務發現機制

A Two-Phase Web Service Discovery Mechanism based on QoS and Collaborative Filtering in an Untrusted Environment

研 究 生：吳 致 衡　　Student: Chih-Heng Wu
指導教授：羅 濟 群　　Advisor: Chi-Chun Lo

國立交通大學
資訊管理研究所
碩士論文

A Thesis
Submitted to Institute of Information Management
College of Management
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Information Management
June 2010
Hsinchu, Taiwan, the Republic of China

中華民國 九十九 年 六 月

一個在不信任的環境下基於服務品質和協同過濾的兩階段網路服務發現機制

研究生：吳致衡　　指導教授：羅濟群 教授

國立交通大學

資訊管理研究所

摘要

近年來，隨著網路服務(Web Service)的大量增加，提供相同功能的網路服務也越來越多，以目前 Universal Description, Discovery and Integration (UDDI) 所提供以關鍵字為基礎來查詢網路服務功能屬性的方式早已不敷使用，使用者很難從多個相同功能的網路服務中找出一個好的網路服務，甚至所找到的網路服務可能是無法使用的。因此，網路服務的非功能屬性，服務品質(Quality of Service, QoS)就成為一個網路服務查詢的重要指標。然而，以 QoS 為基礎的查詢方式又衍生出一個重要的問題，因為網路服務的 QoS 資訊有可能是不正確的，那麼在彼此不信任的環境下，服務提供者所保證的 QoS 可能會和實際的數值有很大的差距，這樣依據 QoS 所查詢到的網路服務就會不正確。因此，本論文提出一個以服務品質和協同過濾為基礎的兩階段網路服務發現機制，第一階段先以協同過濾來分析使用者的經驗，過濾 QoS 不正確的網路服務，第二階段再根據正確的 QoS 資訊來推薦網路服務。最後，本論文針對所提出的機制實作一雛形系統，並且設計模擬實驗以 Average Precision 指標來評估本機制的效能，而實驗的結果也指出，本機制的效能可以達 96%的精確率，是優於服務品質排序和協同過濾兩個方法。

關鍵字: 網路服務、服務品質、協同過濾、網路服務發現

A Two-Phase Web Service Discovery Mechanism based on QoS and Collaborative Filtering in an Untrusted Environment

Student：Chih-Heng Wu Advisor：Pro. Chi-Chun Lo

Nation Chiao Tung University

Institute of Information Management

Abstract

In recent years, there are an increasing number of Web Services providing similar functionalities. However, at the present time, Universal Description, Discovery and Integration (UDDI) only provide keyword-based service discovery to look up for the functional aspects of the desired Web services. Service consumers are hard to choose a better Web service from those providing similar functionalities. Moreover, the service link that service consumer found probably is unusable.

Therefore, Quality of Service (QoS) is becoming an important criterion for selection of the best available service. However, QoS-based service discovery approach has another important issue that the QoS value of Web service may not correct. Consequently, in this untrusted environment, the QoS value service provider guaranteed maybe different from the real QoS value and cause the result of service discovery based on QoS is wrong.

Hence, this paper proposes a two-phase Web service discovery mechanism based on QoS and collaborative filtering. In first phase, the mechanism filter out services with incorrect QoS value, and in second phase, the mechanism recommend services with correct QoS value to service consumer. In the end, this paper present simulation experiments to evaluate the effectiveness of our approach using a prototype implementation of the mechanism. The result of experiments also show that the mechanism can reach 96% precision is better than QoS ranking approach and collaborative filtering approach.

Keywords: Web Service, QoS, Collaborative Filtering, Web Service Discovery

誌謝

能夠完成碩士兩年的學業，首先要先感謝我父母對我的支持跟鼓勵，讓我可以專注在學業上順利完成碩士學位。另外，我要感謝我的指導教授羅濟群老師，在這兩年來對我孜孜不倦的教導，在學術上給我諾大的幫助，讓我受益良多。

接著，我要感謝資管所的老師們，給了我很好的學習環境和資源，教導我很多專業領域的知識，以及更重要的人生哲學和待人處事的道理，這些都將成為我往後人生最珍貴的資產。而在我求學的過程中，我也要感謝實驗室的同儕、學長姐和學弟妹的陪伴，不僅在課業上給我很大的幫助，也讓我在這繁重的求學生涯中增添了許多美好的回憶，大家一起熬夜奮戰的日子、出遊狂歡的時刻，這些回憶所編織出的友誼，都會讓我一輩子深深的珍惜。

最後，我要感謝我的女友雅婷，在這段期間不斷的給我關心，在我最低潮時給我鼓勵，陪我度過最難熬的時刻，讓我感受到溫暖和關懷，陪伴我完成碩士學業。

吳致衡

目錄

圖目錄

表目錄

第一章　緒論

1.1　研究背景與動機

在網際網路的蓬勃發展下，網路服務(Web Services)的技術，讓服務之間可以透過網路互相操作，使得網路服務可以重複的被使用[30]。而在這些技術中，統一描述查詢整合介面(Universal Description, Discovery and Integration, UDDI)在服務提供者和使用者之間扮演著中介者的角色，服務提供者可以將開發完成的服務發布到UDDI註冊中心，而使用者也可以透過UDDI註冊中心來查找符合自己需求的服務。

然而隨著網路服務的數量與日俱增，使用者要找到符合需求的服務也越來越難[15]，以目前UDDI所提供網路服務查詢的方式，只能滿足使用者對網路服務的功能需求[37,16]，而無法滿足使用者對網路服務的非功能屬性的需求，從多個提供相同功能的服務中，找出一個好的網路服務。因此，為了解決網路服務選擇(Service selection)的問題，網路服務的服務品質(Quality of Service, QoS)成為了服務查詢第二個重要的指標[37]。

QoS可以被理解為一組非功能性屬性，例如網路服務的價格(Price)、反應時間(Response Time)、可用性(Availability)、可靠度(Reliability)等等[16,3,37]，不同的使用者對網路服務的QoS會有不同的需求，這時就可以根據網路服務的QoS資訊來解決服務選擇的問題[30,37,11]。

然而根據QoS的服務查詢方法會衍生出兩個問題，其中一個是以目前UDDI的版本，服務提供者無法將服務的QoS資訊發布到UDDI註冊中心，這個問題在許多相關的解決辦法中大都以拓展UDDI中的*tModels*[2,5,30]來解決。然而一個最主要的問題在於如何驗證QoS的正確性，由於網路服務的QoS資訊是由服務提供

者所發布，這些資訊有可能沒有隨時更新而過時，甚至可能網路服務已經停止服務，但是網路服務和QoS資訊依然還存留於UDDI註冊中心中，使得根據QoS所找到的服務可能不符合需求或是無法使用。

有關這個問題的解決方法在以往的文獻中多有探討，除了IBM所提出的WSLA(Web Services Level Agreements)[14]的標準之外，其中大多以計算網路服務的名譽值(Reputation)[33,12,13,9,37]的方法為主，另外也有利用協同過濾機制的方法(Collaborative Filtering)[24,20,34,38]。WSLA主要是用於在服務提供者和使用者之間提供一個對於服務等級的協定契約，目的在於保證服務的品質，但是使用者在UDDI註冊中心查詢服務時，不可能和所有的服務提供者都簽立WSLA，那麼在彼此不信任的環境下，就無法保證服務的QoS資訊。

另外，在計算網路名譽值的方法中，其必須收集使用者針對網路服務QoS的評分，之後再依據評分來計算網路服務的排名。但是這必須考量到幾個問題，因為使用者的評分有可能是不正確的，甚至可能會有惡意的使用者和不公平的評分，而且使用者的評分標準都不同，這樣一來依據使用者評分所計算出來的名譽值可能就會不正確。

而協同過濾的機制則是以計算使用者或是網路服務的相似度，來預測目標使用者對網路服務的評分進而推薦網路服務，然而這會有兩個問題，一是其必須收集使用者對網路服務的評分，所以與計算名譽值的方法會有錯誤評分問題，另外一個是其無法針對使用者對QoS需求來推薦服務。因此，本論文希望能夠提出一個新的協同過濾機制，來解決惡意使用者和錯誤評分的問題，並且依據使用者對網路服務的QoS需求來推薦好的網路服務。

1.2 研究目的

本論文的研究目的在於提出一個基於服務品質和協同過濾機制的兩階段網路服務查詢的演算法，來解決網路服務 QoS 資訊不正確和網路服務選擇(Service Selection)的問題。第一階段先以協同過濾的機制分析舊有使用者過去的使用經驗，找出內含文化(Implicit Culture)[8]，來幫助新的使用者過濾掉 QoS 不正確或是無法使用的網路服務。其主要的想法在於一個人在陌生環境下的行為是無法達到最佳化的，這是因為新來的使用者還不具備在這環境下所需要的知識，但是環境中一些有經驗的使用者可能已經遭遇相同的問題並且具有相關的知識，因此若使用者之間能夠分享這些知識，例如，呼叫一個網路服務成功或失敗的經驗，那麼在這社群中的成員就能夠相互幫助，不僅可以找到品質好而且符合需求的網路服務，還減輕了資訊超載(Information Overload)的問題。

最後，為了解決服務選擇(Service Selection)的問題，第二階段再根據第一階段所過濾出來的網路服務，根據使用者對服務的 QoS 需求計算每個網路服務的 QoS 分數，並依據所得分數來排名網路服務推薦給使用者，藉此達到過濾 QoS 不正確的網路服務，同時滿足使用者查詢時對網路服務的功能和 QoS 需求的目的。

1.3 章節規劃

本論文的章節規劃如下：第一章緒論，針對研究背景、動機和目的，並就整體的研究方向做一個概略的描述。第二章文獻探討，針對現有的網路服務發現演算法的優劣進行比較。第三章為本論文所提出的兩階段網路查詢機制，可分成(1)協同過濾機制、和(2)服務品質排序等兩個階段。第四章則針對所提出的機制實作一雛形系統，並且設計實驗來分析實驗結果。最後，第五章則為本論文的總結，並且探討未來更進一步可以研究的方向。

第二章　文獻探討

本章節裡將探討網路服務技術的發展和基礎架構，以及目前現有網路服務查詢的主要方法，包括：衡量服務品質的 QoS-based 方法、收集使用者對網路服務的評分來計算名譽值的 Reputation-enhanced 方法、以及計算使用者或網路服務相似度的 Collaborative filtering 方法。

2.1　網路服務架構

在過往B2C(Business to Customer)的年代，Web服務的開發對象都是以個人為主，使用者所能使用的功能僅限於單一主機系統所提供的功能，然而隨著網際網路的發展，使得B2B(Business to Business)的商業模式逐漸興盛，為了讓企業內的服務能夠和外部夥伴間建立起流暢的作業流程，服務導向架構(Service-Oriented Architecture, SOA)被提了出來，使得現今的服務對象擴展到了應用程式，亦即應用程式之間必須要能夠相互溝通。

因此，為了因應需求，以服務為導向並且以XML為基礎的網路服務技術被提了出來，使得應用程式之間可以透過網際網路來交換訊息並整合資料。根據W3C(World Wide Web Consortium)對於網路服務的定義，一個網路服務是指一個應用程式可以藉由XML來描述查詢，並且利用URI(Uniform Resource Identifier)來辨識此應用程式的界面和連結方式，而應用程式之間亦是藉由XML形式的訊息，經由網際網路的標準技術規範來互相溝通。簡單而言，Web Service就是一種軟體元件，能夠透過HTTP以及資料格式的開放式標準，例如XML、WSDL、及SOAP等等，來為其他的應用程式提供服務。由於網路服務是以網際網路的開放標準為基礎，在已被廣泛使用的網路架構上來運作，使得網路服務具有良好互通性，讓在不同平台上用不同程式語言所建置的系統也可以輕易整合，克服目前分散式系統使用不同機制造成整合困難的情形，使其具有實現分散式架構的動態整

合、平衡負擔等優點。圖1為W3C所提出的一個網路服務的架構圖。

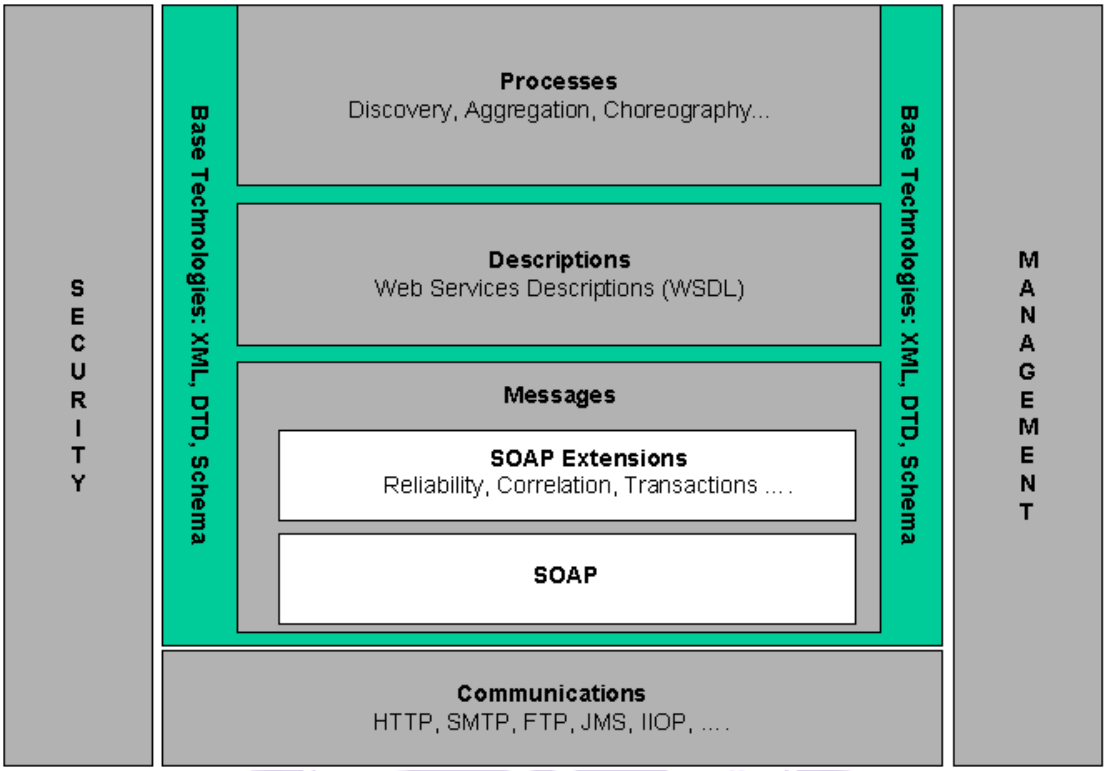


圖 1、W3C 網路服務架構圖

2.1.1 網路服務的角色

在網路服務的服務導向架構中主要會由三個角色所構成：包括服務提供者(Service Provider)、服務需求者(Service Requester)、和服務註冊中心(UDDI Registry)[30,23]。

其中服務提供者主要的目的在於開發網路服務以及建立服務的描述細節，並且將服務註冊於服務註冊中心。而服務註冊中心就扮演著仲介的角色，集成所有服務，接受來自服務提供者的服務註冊請求，也可以處理來自服務需求者的查詢要求。當服務需求者需要找尋服務時，就會透過服務註冊中心來搜尋，並且鏈結適當的網路服務。圖2簡單描述了這三個角色之間的關係。

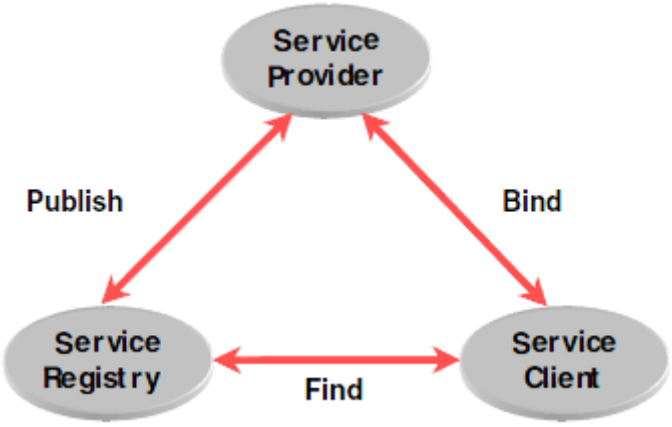


圖 2、服務導向架構圖[23]

因此，一個取得網路服務的流程可以拆成下面的步驟來完成：

(1) 註冊(Registry)：服務提供者將網路服務依UDDI 所需資料向服務註冊中心註冊。

(2) 發佈(Publish)：服務仲介者回應服務提供者註冊成功的訊息，並通報到其他仲介者，完成網路服務發布程序。

(3) 網路服務查詢(Find)：服務需求者向服務仲介者提出服務的查詢需求。

(4) 回應網路服務查詢：服務仲介者回應服務需求者查詢的結果，並且取得服務描述資訊(WSDL)，完成網路服務的發現(Find)程序，

(5) 呼叫網路服務：服務需求者依據WSDL的描述鏈結來對服務提供者發出服務要求(Request)的訊息

(6) 完成鏈結程序(Bind)：服務提供者回應服務需求者執行的結果，完成網路服務的鏈結程序。

從上面的流程可以看到，網路服務查詢(Find)扮演了一個重要的角色，其幫助服務需求者找到服務提供者所發佈的網路服務，使得整體的架構得以順利運行。因此，在接下來的章節中，本論文將會針對網路服務查詢所衍生的問題及方法作更深入的探討。

2.2 服務品質與網路服務

服務品質(Quality of Service, QoS)，根據Menascé, D. A.[1]的定義，其指的是一些服務的品質和屬性的組合，對於網路服務而言就是一組非功能的屬性(Non-Functional attributes)，這在目前的網路服務查詢中扮演著越來越重要的角色，因為以目前網路服務技術，例如WSDL、UDDI，只有考慮到使用者對網路服務功能的需求，使得使用者無法從大量提供相同功能的服務選出一個好的服務，因此OoS就成為了查詢網路服務的重要指標。

以下是一些文獻中常見的QoS屬性 [1,11,37,3]:

- 價格(Price)：指的是呼叫網路服務所需要的成本，通常是以計次的方式收費。
- 可用度(Availability)：指的是在一段時間區間裡網路服務可以回應使用者需求的機率是多少。
- 可取度(Accessibility)：指的是網路服務正常營運且可以不延遲(Non-Delay)的處理服務請求的機率。
- 反應時間(Response Time)：指的是當使用者像網路服務發出請求後得到回應所需的時間。
- 總處理能力(Throughput)：指的是在一段時間區間裡，網路服務可以處理請求的數量多寡。

2.2.1 服務品質資訊的儲存方法

以現有 UDDI V3 的規格[35]，其所提供的查詢方法只能夠滿足使用者對網路服務的功能需求[10,16]，服務提供者無法將網路服務的服務品質註冊到 UDDI 中。因此，為了解決這個問題，有些文獻加強了 UDDI 的查詢方法(Inquiry

Operation)，將服務品質的資訊遷入到傳輸的訊息中。

例如 Martin-Diaz, O.等人[2]所提出的 UDDIe，其提供了一個應用程式介面(Application Program Interface, API)讓使用者可以透過 *propertyBag*[35]來自定 QoS 資訊。然而這個方法只能用於 UDDIe 的 G-QoSM 架構中，而且只能提供很有限的 QoS 資訊。另外，Martin-Diaz, O.等人[27]也提出了 QRL(Quality Requirements Language)，其是一個以 XML 為基礎可以用於網路服務和其他分散式系統的語言，然而 QRL 並沒有明確的定義其與網路服務介面標準的關係，例如 WSDL。除此之外，QRL 也沒有提供足夠的資訊來說明如何管理網路服務的 QoS 資訊。

而在文獻中比較常見的方法是利用 UDDI 中的 *tModel*[35]來發布 QoS 資訊到 UDDI 中[30,37,33]。*tModel* 主要是 UDDI 用來描述網路服務的技術指紋，例如格式、傳輸協定、輸入和輸出參數為何、商業邏輯等等，這些細節都被定義在該服務實作的描述裡。一個網路服務可以參照到多個 *tModels*[35]，亦即一個網路服務可以擁有多個 QoS 資訊，而每一個 QoS 屬性可以用 *tModel* 裡的 *keyedReference*[35]來儲存，其在 *tModel* 中是一個名稱-數值(name-value)的結構[37]，亦即 QoS 屬性的名稱可用 *KeyName*[35]來儲存，而數值則用 *KeyValue*[35]來儲存。這個方法的優點在於，其是利用原本 UDDI 中已有的結構，不需要在制定額外的標準。圖 3 為一個利用 *tModel* 儲存 QoS 資訊的例子。

```
<tModel tModelKey="uuid:0e727db0-3e14-11d5-98bf-002035229c64">
    <name>uudi-org:qualityInformation</name>
    <description xml:lang="en">Quality of Service Information</description>
    <overviewDoc>
        <description xml:lang="en"></description>
        <overviewURL> http://www.uddi.org/specification.html
        </overviewURL>
    </overviewDoc>
    <categoryBag>
        <keyedReference
            keyName="uddi-org:types" keyValue="categorization"
            tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"/>
        <keyedReference
            keyName="uddi-org:types" keyValue="checked"
            tModelKey="uuid:c1acf26d-9672-4404-9d70-39b756e62ab4"/>
        <keyedReference
            keyName="uddi-org:types" keyValue="specification"
            tModelKey="uuid:c1acf26d-9672-4404-9d70 39b756e62ab4"/>
    </categoryBag>
</tModel>
```

圖 3、利用 *tModel* 儲存服務品質資訊[30]

2.3 網路服務查詢方法

在目前的文獻中，網路服務查詢主要有三個方法，包括：(1) QoS-based 方法、(2) Reputation-enhanced 方法、(3) Collaborative filtering 方法。

2.3.1 QoS-based 查詢方法

如何從一群提供相同功能的網路服務中找出一個好的網路服務，稱為服務選擇(Service Selection)[10,21]的問題，因此除了以往針對網路服務功能的查詢之外，網路服務的非功能屬性 QoS 也成為了查詢的重要指標，為了滿足使用者對網路服務功能性和非功能性的需求，許多文獻提出了以服務品質為基礎(QoS-based)的網路服務查詢方法。

Gouscos, D.等人[7]提出了一個簡單的方法，將重要 QoS 屬性分類成兩個群

組，分別是靜態(Static)和動態(Dynamic)屬性。例如，價格、所保證的回應時間(Service Response Time)、和所保證的失敗率(Probability of Failure)都被認為是靜態屬性，可以將其儲存到 UDDI 中。而像是服務實際的回應時間和失敗率的數值則可以放到 WSDL 文檔中或是透過資訊代理人(Information Broker)來提供，這個方法主要的優點在於相當直覺而且易於實作，但是依然無法解決 QoS 過時不正確的問題。

Ran, S. [30]提出了 Extending UDDI Model，除了傳統模型中的三個角色，服務提供者、服務需求者、和服務註冊中心之外，還多加了一個角色，稱為網路服務品質擔保者(Web Service QoS Certifier)，其主要是在服務提供者把服務註冊到 UDDI 之前，先驗證所提供的服務品質是否如所保證的相同，這樣服務需求者在鏈結網路服務之前，就可以先向 QoS Certifier 查詢所要鏈結網路服務的 QoS 是否正確。然而這個方法並沒有提出一個可靠的演算法來驗證網路服務的服務品質，而且只在服務註冊之前驗證服務品質的正確性，並沒有辦法提供保證當未來 QoS 更新時的正確性。圖 4 為 Ran, S.[30]所提出的模型架構圖。

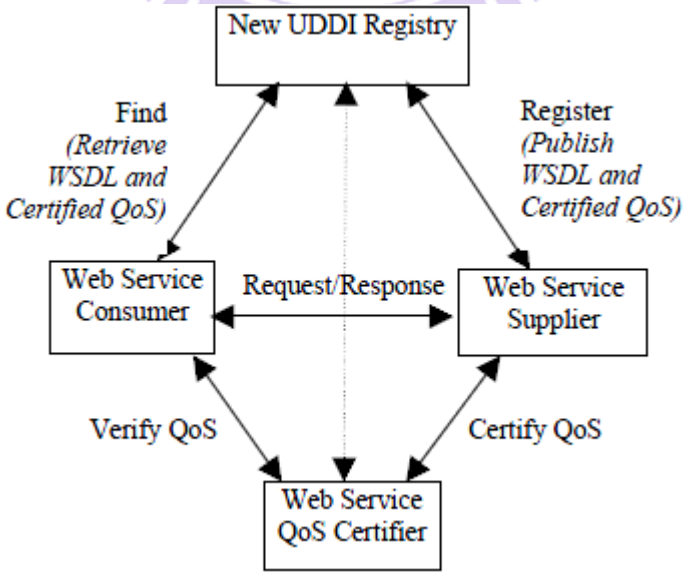


圖 4、一個新的網路服務註冊與發現模型[30]

Huang, A.F.M.等人[3]根據不同資料型態的 QoS 屬性，提出了一個三階段的服務選擇系統(Service Selection Scheme)來幫助服務請求者查詢服務。在 Text-based QoS matchmaking 的階段中，其是利用 UDDI 所提供的關鍵字查詢和網路服務目錄的查詢方式，而在 Numeric-based QoS matchmaking 階段中則區分了兩個情境，一是以 QoS 為基礎的單一服務發現(Single QoS-based Service Discovery)，其是指從多個服務當中衡量個別的 QoS 數值，並從其中選出 QoS 表選最好的服務給使用者。而另一個情境則是以 QoS 為基礎的最佳化選擇(QoS-based Optimization)，這是指從整體的工作流(Workflow)結構來看，選擇一個服務使得整體的 QoS 表現是最好的。圖 5 描述了這三個階段。

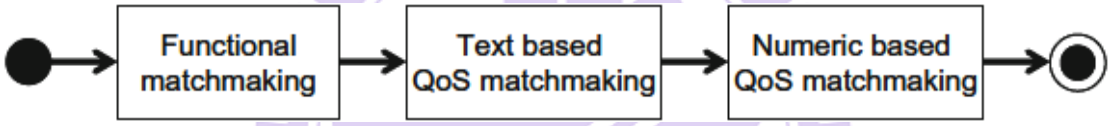


圖 5、三階段的網路服務選擇系統[3]

Al-Masri, E.等人[11]提出了 WSRB(Web Service Repository Builder)框架，利用 WSCE(Web service Crawler Engine) 根據使用者的功能需求向多個 UDDI 查詢網路服務並取得 QoS 資訊，之後再將所收集到提供相同功能網路服務的 QoS 資訊以矩陣的方式來儲存，一開始先找出每項 QoS 數值中的最大值來對 QoS 作正規化(Nomalize)，而後再依據使用者的 QoS 需求，給予不同的 QoS 數值相對應的權重，權重越高表示使用者的需求越高，並依此計算出每個服務的分數，最後依據分數的高低推薦服務給使用者。然而，此方法雖然解決了服務選擇的問題，但卻無法驗證 QoS 的正確性。

2.3.2 Reputation-enhanced 查詢方法

雖然加入 QoS 的考量來查詢網路服務可以解決服務選擇的問題，然而樣的查詢方法可能會面臨到一個問題，就是 QoS 資訊是由服務提供者來發佈在 UDDI 上，因此 QoS 資訊有可能是不正確或者是過時的，那麼依據 QoS 資訊來推薦給使用者的網路服務排名就可能會是錯誤的。因此有許多文獻提出以計算網路服務的名譽值(Reputation)來加強以 QoS 為基礎的查詢方法，驗證 QoS 不正確的問題。

Maximilien E.M.和 Singh, M.P.[9]提出了一個網路服務代理人協定，稱為 WSAP(Web Service Agent Proxy)，讓客戶端的代理人都能夠互相溝通，並且分享使用者的資訊，並藉此整合使用者在過去使用紀錄中的評分來計算網路服務的名譽值，因此代理人要負責監控使用者的使用行為，例如與 UDDI 的溝通訊息、和服務提供者的請求和回應等等。這篇文章同時也考慮到新的網路服務，因為還沒有使用者的使用紀錄而無法計算名譽值，這時就可以尋求可信賴的服務提供者或服務需求者來支持這個網路服務。

Majithia, S.等人[31]提出了一個以名譽值為基礎並結合語意(Semantic)的網路服務查詢框架，這個框架實作了一個名譽值管理系統，負責收集服務需求者針對不同領域背景網路服務的評分，並且對於不同領域背景的網路服務會給予不同的權重來計算名譽值，而這些權重就代表了對某特定使用者群組的重要性。另外，在名譽值計算中還加入了時間的考量，使得名譽值的分數會隨著時間而逐漸變小，亦即離現在時間點越近的名譽值，其重要性會比離現在時間點遠的名譽值更重要。

Wishart, R.等人[28]則提出了一個服務查詢的協定，稱為 *SuperstringRep*，其中的名譽值系統用來收集和管理使用者對服務的評分。當使用者在查詢服務時，

此系統會計算每個服務的名譽值用來反應出每個服務整體 QoS 的好壞，並且以此作為服務排名的依據。另外，針對每個使用者的評分，其還加上了年齡係數，使得較新的評分對名譽值的影響程度會大於較舊的評分。雖然這個方法能夠利用名譽值來排除 QoS 不正確的服務，但是其卻沒辦法針對個別使用者的 QoS 需求來推薦服務，而只能推薦名譽值高的服務。

Ziqiang, X.等人[37]則改進了 Wishart, R.的方法[28]。其在服務需求者和 UDDI 之間實作了一個查詢服務的代理人(discovery agent)，並利用這個代理人收集使用者針對網路服務 QoS 的評分。在計算名譽值時，其也加入了時間的考量，讓較新的評分的重要性大於舊的評分，不同的是，其除了計算服務的名譽值外，還加入了使用者對於 QoS 需求的考量，計算出服務的 QoS 分數。最後，再根據 QoS 分數和名譽值做一個整體的運算，做為服務排名的依據以排除 QoS 不準確的服務。

其他還有許多文獻，例如 Vu, L.H.和 Aberer, K.[13]、Li, J[15]、Sreenath, R.和 Singh, M.P.[26]等等，都是利用網路服務的名譽值來解決 QoS 不正確的問題。然而，這些方法都是依靠收集使用者對網路服務的評分來計算名譽值。因此，這會衍生兩個問題，一是使用者的評分可能是錯誤的，甚至可能會有惡意的使用者，擾亂了整體計算名譽值的基礎，使得名譽值不被信賴。另一是使用者的評分是主觀的，每個人的評分標準都會有所不同，這可能會造成在計算名譽值上的誤差。

2.3.3 協同過濾與網路服務

另外，有一些文獻是分析使用者過去的行為經驗來查詢服務。因此，底下先就協同過濾演算法做一個簡單的介紹，之後再針對利用協同過濾演算法的網路服務查詢方法來介紹。

2.3.3.1 協同過濾演算法

協同過濾主要是根據所收集的使用者檔案(User Profile)來預測使用者對一個物品(Item)喜好程度[17,19]，主要的演算法有三種，包括以使用者為基礎(User-based)的協同過濾演算法、以物品為基礎(Item-based)的協同過濾演算法、和混和式(Hybrid)協同過濾演算法。

以使用者為基礎(User-based)的協同過濾演算法[19]，其主要的作法是收集使用者過去對物品的評分來量化喜好程度，之後再針對目標使用者計算與其他使用者之間的相似度，並依相似度高低來預測目標使用者對特定物品的喜好程度，最後再依預測分數的高低來推薦目標使用者可能會感興趣的物品。圖 6 表達了此方法的概念。其將使用者 u_a 對物品 i_b 的評分 $x_{a,b}$ 排成一矩陣(User-Item Matrix)，並且計算使用者相似度，並將相似度由高到低排序，而其他使用者對物品的評分，就拿來預測目標使用者對特定物品的評分。

而相對於以使用者為基礎的方法，另外一種做法則是屬於以物品為基礎的協同(Item-based)過濾演算法[6]，與前述方法不同的地方在於，其是以計算物品之間的相似度，來達到預測目標使用者對特定物品的評分，再依預測分數來推薦物品。圖 7 表達了這種方法的概念。與前述方法相同，其亦將使用者 u_a 對物品 i_b 的評分 $x_{a,b}$ 排成一矩陣，但其是計算物品相似度，再將相似度由高到低排序，而目標使用者對其他物品的評分，就拿來預測目標使用者對特定物品的評分。

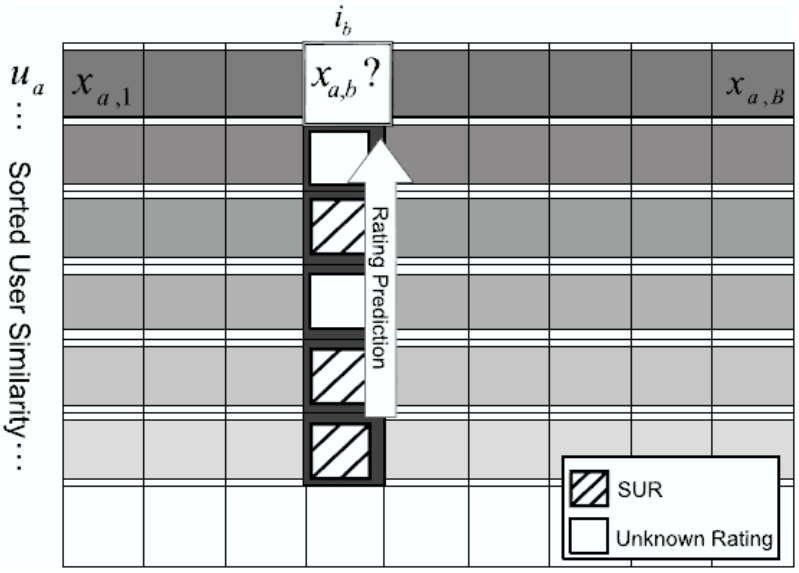


圖 6、User-based approaches[19]

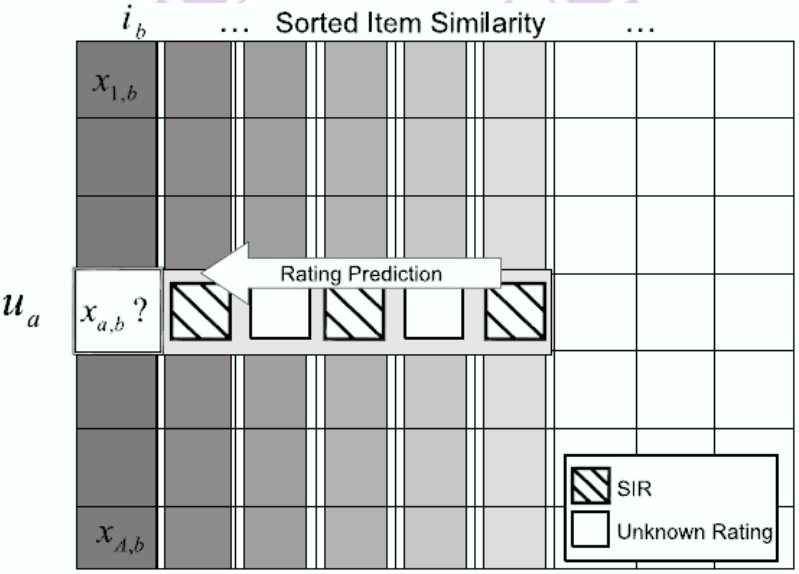


圖 7、Item-based approaches[19]

然而這兩個方法有共同缺點，就是只考慮單一的面向。以使用者為基礎的方法而言，若收集到的使用者檔案(User Profile)中，只有很少的使用者與目標使用者是相似的，那麼所預測出來的分數就會不正確。相同的，以物品為基礎的方法會擁有同樣的問題，當很少物品與目標物品相似時，那所預測的分數也會有誤差。因此，為了解決上述的問題，有文獻提出混和式(Hybrid)的協同過濾演算法[18]，

將使用者和物品的面向都一併考慮進來。其主要的作法是，同時計算目標使用者與其他使用者的相似度，以及目標物品於其他物品之間的相似度，並將這兩個相似度的數值整合計算由高到低排序來預測分數。圖 8 描述了此方法的概念。

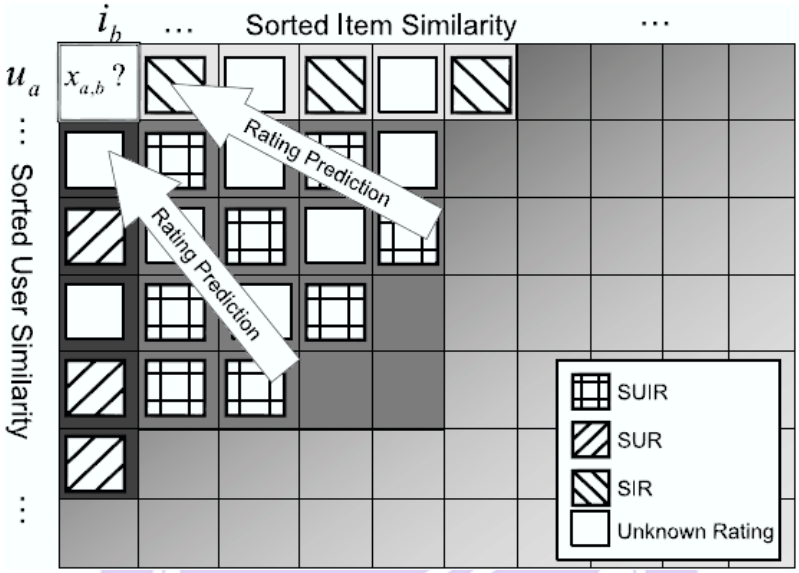


圖 8、Combining user-based and item-based approaches[19]

2.3.3.2 Collaborative filtering 查詢方法

在目前的文獻中，還只有少數的文獻利用協同過濾的機制來加強網路服務的查詢。例如，Manikra, S.U.和 Prabhakar, T.V.[34]提出了一個結合協同過濾和語意(Semantic)的網路服務選擇框架，來幫助使用者查詢網路服務，這個框架將服務需求的語意配對結合到推薦系統中，其中的推薦方法則是以物品為基礎(item-based)的協同過濾演算法，用來滿足使用者對服務的 QoS 需求，以及解決網路服務選擇的問題。然而，其缺點在於當其他物品很少與目標物品相似時，其所預測出來的分數就會有誤差。

因此，Zheng, Z.等人[38]則是利用混和式(Hybrid)的協同過濾演算法，同時考慮使用者之間的相似度和網路服務之間的相似度來查詢網路服務，其主要的作法

是在預測目標使用者對特定網路服務 QoS 的評分時，分別用 User-based 和 Item-based 的協同過濾方法計算出兩個分數後，乘以各自的權重相加成一個預測分數，再依據分數的高低來推薦服務。此方法的優點在於其是將使用者和網路服務做一個整體的考量，所預測出來的分數會較準確。然而，這兩個方法都有共同的問題在於，其必須考量到惡意使用者和錯誤評分的問題，以及使用者的評分是主觀的，每個人的評分標準都會有所不同，這可能會造成在預測分數上的誤差。

另外，也有不收集使用者對網路服務的評分，而是紀錄使用者過去的行為，並且利用協同過濾加以分析後來推薦網路服務的演算法。例如，Kokash, N.等人[25]將過去使用者查詢和呼叫服務的行為都記錄下來，並且透過歷史紀錄分析出使用者查詢字串和網路服務之間的關連性，找出請求和方法的配對(Request-Solution Pairs)，亦即每個網路服務都會配對到一個使用者需求，之後再利用協同過濾的機制，計算使用者需求之間的相似度，並依相似度的高低來推薦服務。

Kerrigan, M[24]則是參考了 Smyth, B[29]等人運用在網頁搜尋的方法，將原本協同過濾中的 Item-User matrix 轉換成了 Service-Goal matrix，其中的 Service 指的是網路服務，而 Goal 指的是使用者需求的語意。在矩陣中記錄了每個網路服務被每個 Goal 選到的次數，因此就可以根據這些次數，來分析 Goal 和網路服務之間的關連性，次數越高代表關聯性越高，之後再依關聯性的高低來推薦網路服務。

上述這些方法的優點在於，其不用收集使用者對網路服務的評分，因此不用解決惡意使用者和錯誤評分的問題。然而其缺點在於，這些方法並沒有辦法依據使用者對服務的 QoS 需求來推薦網路服務。

第三章　一個在不信任的環境下基於服務品質和協同過濾的兩階段網路服務發現機制

在本章中，一開始會先描述本論文所要解決的問題定義以及服務品質資訊的管理方法。接下來，再針對本論文所提出的兩階段網路服務發現機制來做詳細的介紹。

3.1　問題定義

利用網路服務的 QoS 資訊可以幫助使用者從多個提供相同功能的服務中選出一個好的服務，但是如何驗證服務提供者所發布 QoS 資訊的正確性，成為了最重要的研究議題。在第二章文獻探討的部分中，整理了三種主要的網路服務查詢方法，以及這些方法各自的缺點：

- QoS-based 查詢方法無法驗證 QoS 資訊的正確性。
- 收集使用者的評分，必須處理惡意使用者和錯誤評分的問題。
- 評分是主觀的，容易造成名譽值計算上的誤差。
- 協同過濾演算法，只能過濾 QoS 不正確的服務，而無法針對使用者的 QoS 需求來推薦服務。

因此，為了改善上述的缺點，本論文提出一個基於服務品質與協同過濾的兩階段網路服務發現機制，利用收集使用者查詢和呼叫網路服務的行為紀錄，以協同過濾找出符合使用者功能需求的網路服務，並且過濾掉 QoS 不正確的網路服務。之後再根據正確的 QoS 資訊，針對使用者對網路服務的非功能需求，計算 QoS 分數來推薦網路服務。這樣一來就可以不用收集任何使用者對網路服務的評分，來過濾掉 QoS 不正確的網路服務，亦可以達到滿足使用者 QoS 需求的目的，改進上述方法的缺點。

以下就先簡單描述一下本機制的兩個階段：

(1) 階段一：先利用協同過濾分析過往使用者的歷史紀錄，過濾掉 QoS 不正確的網路服務，只留下 QoS 正確而且符合使用者功能需求的網路服務，並作為下一個階段的數入值。

(2) 階段二：此階段的目的在於滿足使用者對網路服務的非功能性需求，因此在這裡會根據使用者的 QoS 需求，針對第一階段的網路服務來計算 QoS 分數，分數越高代表越能夠符合使用者的需求，最後再依照 QoS 分數的高低來推薦服務。

3.2 階段一：協同過濾

演算法第一階段的目的在於找出使用者間的隱含文化(Implicit Culture)[8]，來幫助目標使用者找到適當的服務，而不需要收集任何的使用者評分，避免惡意使用者以及錯誤評分的問題。其主要的方法是利用使用者過去查詢和請求網路服務的行為紀錄，透過協同過濾過濾掉 QoS 不正確的網路服務，並且幫助使用者找到符合功能需求的網路服務。主要的步驟如下：

- 步驟 1.1：建立查詢語句與網路服務矩陣
- 步驟 1.2：計算查詢語句相似度
- 步驟 1.3：計算網路服務與查詢語句的相關程度

3.2.1 步驟 1.1：建立查詢語句與網路服務矩陣

在協同推薦的演算法中，使用者對物品的評分或使用者所選過物品的行為都會被記錄到資料庫中，而形成一使用者和物品的矩陣。相同的，在網路服務查詢的流程中，本論文亦將使用者以查詢語句(query)在 UDDI 中查詢網路服務的行為，以及之後呼叫網路服務的行為都記錄在歷史資料庫中。因此，每個網路服務被每

個查詢語句所選到的次數，我們就可以將其表示為一個查詢語句與網路服務的矩陣。而在圖 9 的矩陣中，各符號所代表的意思如下：

- q_j，代表的是使用者所下過的查詢語句，而在列上非零數值的組合，表示是因為這個查詢語句而被呼叫過的網路服務組合。
- w_i，代表的是使用者所呼叫過的網路服務，而在行上非零數值的組合，表示是這個網路服務曾經因為這查詢語句的組合而被使用者呼叫過。
- I_{ij}，代表的則是特定的網路服務因為特定的查詢語句而被成功呼叫過的次數。

另外，為了要能夠過濾 QoS 不正確或是無法使用的網路服務，其中 I_{ij} 所記錄的次數，必須具備兩個重要的條件。第一是，使用者必須是成功的呼叫網路服務，亦即所呼叫的網路服務，其回應訊息不能是錯誤訊息或是無回應等等而必須是正常的回應訊息。第二是，在使用者呼叫網路服務的當下，系統所偵測到實際 QoS 的數值必須要符合服務提供者所提供的 QoS 數值。亦即，只有在上述兩個條件都符合的情況下，使用者呼叫網路服務的次數才會被計算到I_{ij}中。

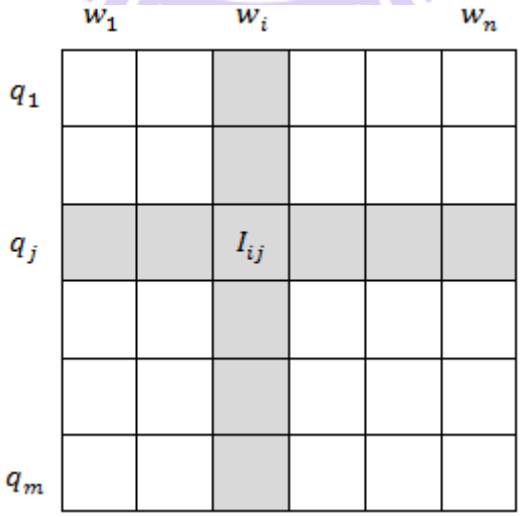


圖 9、查詢語句與網路服務矩陣

3.2.2 步驟 1.2：計算查詢語句相似度

當使用者需要查詢網路服務時，為了找到想要的服務功能，其會發出一個包含功能條件(functional requirements)的請求訊息，這個功能需求可能是一些關於網路服務名稱或描述的關鍵字，亦稱為查詢語句。因此，在接下來的步驟中，我們需要計算目標使用者所下查詢語句與其他在矩陣中查詢語句的相似度，目的在於利用相似度高的查詢語句來幫助目標使用者找到適當的網路服務。

在這裡本論文是利用向量空間模型(Vector Space Model)將查詢語句經由 TF-IDF(Term Frequency – Inverse Document Frequency)轉換成向量後，計算相似度。更詳細一點來說，一句查詢語句q_i在本模型中會被拆解成一串有順序的字詞(term)t_j，使得 $q_i = (t_1, t_2, \dots, t_n)$，而$|n_i|$為查詢語句的長度且$t_j \in T, j = 1 \dots n$，其中 T 為所有查詢語句Q所包含所有字詞的集合，而$Q = \{q_1, q_2, \dots, q_m\}$，$m$為系統所收集到所有查詢語句的數量。至於每個字詞$t_j$，我們以$n_{ij}$來表示$t_j$在$q_i$中所出現的次數，而$n_j$則代表在查詢語句中至少包含字詞$t_j$一次的查詢語句數量。以下為計算查詢語句$q_i$所包含的字詞$t_j$的 TF-IDF 權重的公式：

$$w_{ij} = \frac{n_{ij}}{|n_i|} * log\left(\frac{m}{n_j}\right)$$

因此，我們就可以將查詢語句q_i依所包含的字詞t_j轉換成 TF-TDF 權重的向量$\overrightarrow{q_i} = (w_{i1}, w_{i2}, \dots, w_{in})$，而在計算兩個向量$\overrightarrow{q_i}$, $\overrightarrow{q_k}$之間的相似度時，則是計算兩者之間的$cos\,\theta$，$cos\,\theta$越大表示兩個向量的角度越小，也就越相似，計算的公式如下：

$$sim(\overrightarrow{q_i}, \overrightarrow{q_k}) = \frac{\overrightarrow{q_i} \cdot \overrightarrow{q_k}}{|\overrightarrow{q_i}| * |\overrightarrow{q_k}|}$$

接下來我們將步驟 1.1 所建立的矩陣，依照查詢語句q_k與其他查詢語句的相似度高低來排序，會得到圖 10 的矩陣。

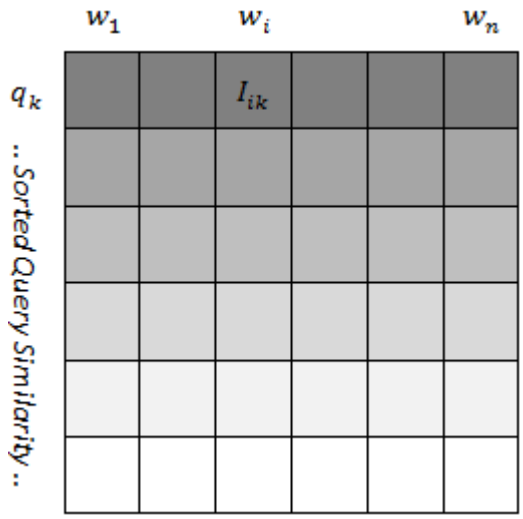


圖 10、依查詢語句相似度排序的矩陣

3.2.3 步驟 1.3：計算網路服務與查詢語句的相關程度

在協同過濾的第三個步驟，是計算網路服務與查詢語句的相關程度。當使用者下特定查詢語句後所選擇的網路服務，通常都是符合使用者功能需求的服務，例如當使用者下了"天氣預報"的查詢語句，其所選的網路服務會有很大的機率都是提供查詢天氣的服務。因此，我們可以利用步驟 1.2 所建立的查詢語句與網路服務的矩陣，來計算兩者的相關程度。

以圖 11 的矩陣為例子，其表示了每個網路服務w_i被每個查詢語句q_k呼叫過的成功次數I_{ik}，依此我們就可以計算網路服務和查詢語句之間的相關程度，而被呼叫過的成功次數越多的網路服務，代表了被某特定查詢語句選到的機率越高，相關程度也就越高。因此，每當使用者下了查詢語句後所呼叫的網路服務，都會幫助協同過濾的方法能夠更準確的找到適當的網路服務，並且過濾掉 QoS 不好的網路服務。相關程度計算的公式如下：

$$Relevance(w_i, q_k) = \frac{I_{ik}}{\sum_{\forall i} I_{ik}}$$

接下來，我們結合步驟 1.2 所計算查詢語句q_k與其他查詢語句q_j的相似度，

以及其他查詢語句q_j與其網路服務w_i的相關程度做一個綜合的考量，將相似度與相關程度相乘計算出最後的相關程度R_{ik}，來代表查詢語句q_k與所有網路服務的相關程度。計算公式如下：

$$R_{ik} = Relevance(w_i, q_j) * sim(q_k, q_j), \qquad \forall i, \forall j$$

這樣做的優點在於不僅可以找到和q_k相關程度高的網路服務，亦可以找到和q_k相似度高的查詢語句中與其相關程度高的網路服務來推薦給使用者，避免只考量到單一面向的缺點，例如目標查詢語句q_k在資料庫中很少與其相似的查詢語句，那麼網路服務就會很少與q_k有相關聯，使得所找到網路服務的精確率很低。圖 11 為依相似度和相關程度高低所排序的矩陣。

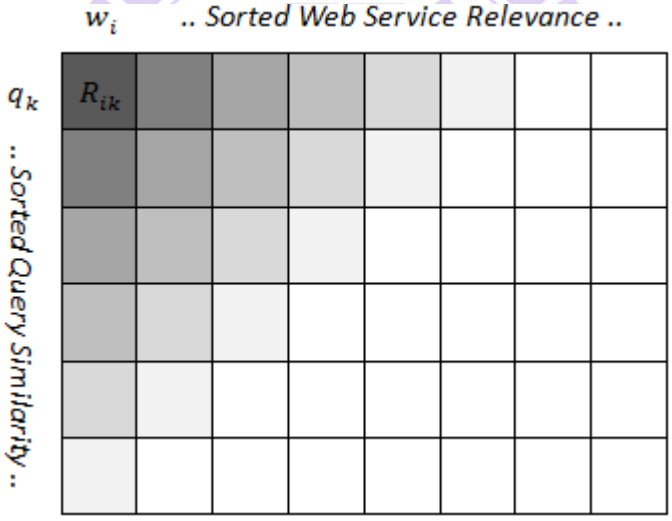


圖 11、依相似度和相關程度高低所排序的矩陣

最後再依據一個門檻值(threshold)來剔除掉相關程度太低的網路服務，只保留相關程度高的網路服務。因此，當第一階段結束時，我們所找到的網路服務有很大的機率都是符合使用者的功能需求並且 QoS 是正確的網路服務。

第一階段整體的流程如下 ：

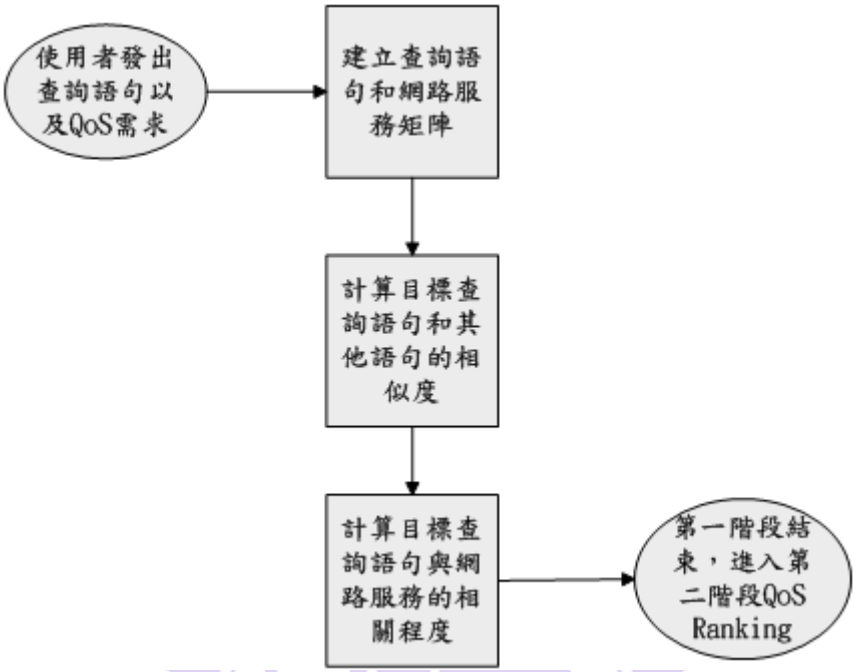


圖 12、第一階段流程圖

3.3 階段二 ： 服務品質排序

經由第一階段所找到的網路服務不僅可以滿足使用者的功能需求，亦可以排除 QoS 不正確的服務，然而，這些網路服務所提供的功能可能都是相同的，使用者仍然很難從中選出一個好的服務。因此，為了解決服務選擇(Service Selection)的問題，幫助使用者從這些網路服務中找到一個好的服務，第二階段會將第一階段所篩選過後的網路服務，再依據使用者對網路服務的 QoS 需求，來排名網路服務，做為推薦服務的依據。主要的步驟如下 ：

- 步驟 2.1 : 建立服務品質與網路服務矩陣
- 步驟 2.2 : 正規劃服務品質數值
- 步驟 2.3 : 計算服務品質分數

3.3.1 步驟 2.1：建立服務品質與網路服務矩陣

在本論文中，我們參考了過去的文獻[22,37,11]選了四個較常拿來衡量網路服務服務品質的屬性，詳細的描述如下：

- 價格(Price)：指的是請求網路服務的成本，單位是以次計費。
- 可用度(Availability)：指的是一段時間區段內，網路服務可用的機率為多少，單位是一天的機率(%/1-day)。
- 反應時間(Response Time)：指的是當使用者發出請求訊息給網路服務時，多久後可以得到正確的回應，單位為毫秒。
- 總處理量(Throughput)：指的是網路服務在一段時間區段內，網路服務可以處理最大的請求次數，單位為一秒鐘請求的次數(requests/second)。

因此，一開始我們先針對所有提供相同功能的網路服務，以及服務提供者所發佈相對應的服務品質資訊，建立一服務品質與網路服務的矩陣。如圖 13 所示，w_i代表網路服務，n為網路服務的個數。p_j為服務品質屬性，m為服務品質屬性的個數。q_{ij}代表網路服務w_i中服務品質p_j的數值。

另外，因為每個服務品質屬性所代表的意義都不同，因此在矩陣中q_{ij}所存的數值，亦要因為服務品質屬性的特性而有所不同。在這裡，本論文將服務品質屬性分為兩類。第一類是服務品質屬性的數值越小所代表服務品質越好，例如，網路服務的價格和反應時間。而第二類則是服務品質屬性的數值越大所代表服務品質越好，例如，可用度和總處理量。因此，在矩陣中會將第一類的服務品質屬性的原本數值以倒數的數值放到矩陣中，而第二類的服務品質屬性則是以原本的數值放到矩陣當中。

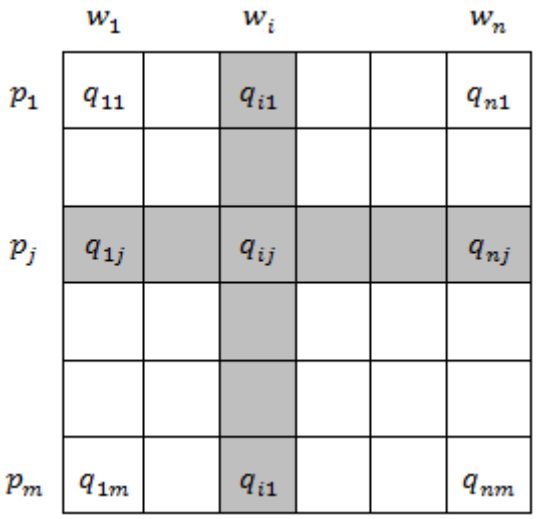


圖 13、服務品質與網路服務矩陣

3.3.2 步驟 2.2：正規化服務品質數值

由於每項服務品質的計算單位和大小差距都不相同，因此需要將步驟 2.1 的矩陣作正規劃的處理，使得所有的數值都在同一個比較基準之下，才能夠將網路服務依服務品質來排序。

以圖 13 為例，為了正規劃服務品質數值q_{ij}，一開始我們必須先找到每項服務品質屬性p_j中的最大值Mp_j：

$$Mp_j = \max(q_{ij}), \qquad \forall j$$

之後再將q_{ij}和Mp_j相比得到正規畫的數值N_{ij}，使得所有的數值都會介於 0 到 1 之間，計算的公式如下：

$$N_{ij} = \frac{q_{ij}}{Mp_j}, \qquad \forall i, \forall j$$

3.3.3 步驟 2.3：計算服務品質分數

接下來，為了能夠滿足不同使用者對網路服務的服務品質需求，在計算服務品質分數時，針對不同的服務品質p_j會給予不同的權重WT_j，例如當使用者想要

找到最便宜的網路服務，那就代表使用者對價格的服務品質屬性有比較高的需求，而應該在計算價格的服務品質上給予較高的權重。

因此，當使用者送出查詢網路服務的請求訊息時，當中的資訊會包含使用者針對每個服務品質的重要性所做的排序，排序越高代表越重要，權重也會越高。例如，當使用者的排序依序為價格、反應時間、總處理量、和可用度時，則在計算分數時價格的權重會最高，而可用度的權重會最低。除此之外，權重的數值都是介於 0 到 1 之間，而所有權重的加總則會等於 1。因此，針對步驟 2.2 的正規劃數值N_{ij}乘上相對的權重WT_j後就會得到個別服務品質的分數QS_{ij}。計算的公式如下：

$$QS_{ij} = WT_j * N_{ij}, \qquad \forall i, \forall j$$

最後，將各個網路服務中所有的服務品質分數加總，就可以得到針對使用者服務品質需求所計算而得的分數，並依照分數高低來推薦給使用者。其中，分數越高的網路服務代表越能夠符合使用者的服務品質需求。計算的公式如下：

$$QoSRank(w_i) = \sum_{j=1}^{n} QS_{ij}, \qquad \forall i$$

而整體第二階段的流程如下：

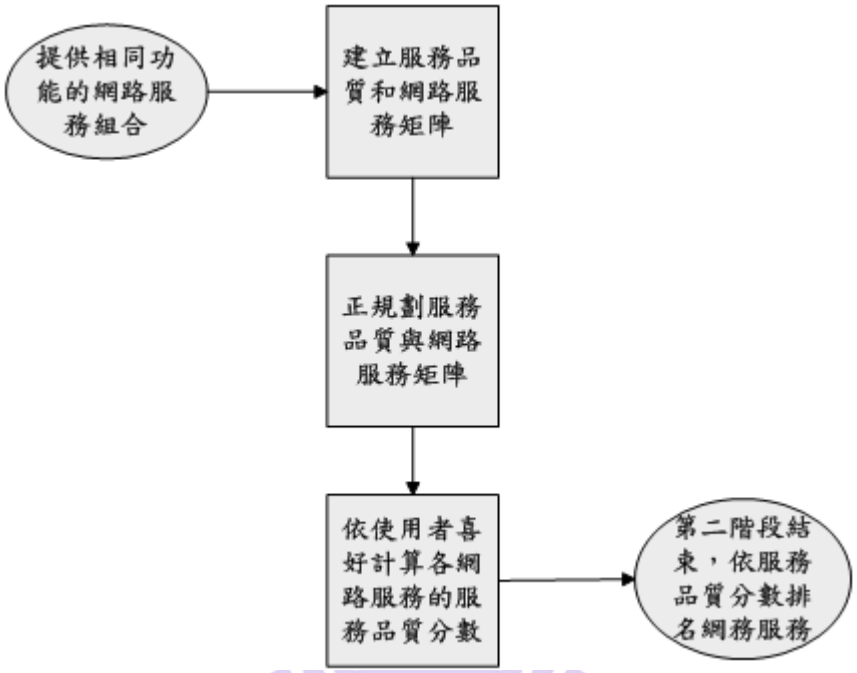


圖 14、第二階段流程圖

3.4 服務品質資訊的管理

在UDDI V3[35]的架構中，*tModel*是用來描述網路服務的技術指紋，例如格式、傳輸協定、輸入和輸出參數為何、商業邏輯等等，目的是希望讓服務請求者可以使用更有意義的搜尋方式，而不單侷限於商業名稱、分類或是服務的名稱等的查詢方式。而在UDDI中，*tModel*的實體(Instance)是以*keyedReference*呈現，並且透過鍵值*tModelKey*當作識別鑰匙，和*categoryBag*用於儲存分類資訊，使得*tModel*成為一個用於描述技術規格的參考系統。

3.4.1 服務品質資訊的發佈

因此，在本論文所提出的架構中，服務品質的資訊是擴充 UDDI 註冊中心的 *tModel* 來儲存。當服務提供者在發佈一個網路服務到 UDDI 註冊中心前，一個相對應的 *tModel* 會被創造出來用於存放網路服務的 QoS 資訊，使得服務請求者可以透過 *tModelKey* 來找到 QoS 資訊。而每個 QoS 屬性，則是由 *tModel* 裡的

keyedReference 來表示，其中 QoS 屬性的名稱是以 *keyName* 來表示，而數值則是以 *keyValue* 來表示。至於每個 QoS 屬性的單位量詞並沒有在 tModel 中表示出來，在這邊本論文假設這些單位量詞為預設值，亦即服務提供者和使用者對 QoS 的衡量單位是相同的，例如價格(Price)，其單位是每次請求為 0.01 美元、反應時間(Response Time)是以毫秒為單位等等。

圖 15 為服務提供者發佈天氣查詢服務的 QoS 資訊的範例：

- 價格(Price)：每次請求為 0.01 美元
- 可用度(Availability)：99%
- 反應時間(Response Time)：70 毫秒
- 總處理量(Throughput)：每秒 800 個請求

```
<tModel tModelKey="uuid:CD978180-3B13-11DF-A8EB-D2A3C1ECE15B">
  <name>QoS information for Weather service</name>
  <description xml:lang="en">
    Quality of Service Information for Weather service
  </description>
  <overviewDoc>
    <description xml:lang="en"></description>
    <overviewURL>
      http://<URL describing schema of QoS attributes>
    </overviewURL>
  </overviewDoc>
  <categoryBag>
    <keyedReference
      tModelKey=" uuid:CD978180-3B13-11DF-A8EB-D2A3C1ECE15B "
      keyName="Price"
      keyValue="0.01" />
    <keyedReference
      tModelKey=" uuid:CD978180-3B13-11DF-A8EB-D2A3C1ECE15B "
      keyName="Availability"
      keyValue="99.0" />
```

```
    <keyedReference
      tModelKey=" uuid:CD978180-3B13-11DF-A8EB-D2A3C1ECE15B "
      keyName="ResponseTime"
      keyValue="30" />
    <keyedReference
      tModelKey=" uuid:CD978180-3B13-11DF-A8EB-D2A3C1ECE15B "
      keyName="Throughput"
      keyValue="800" />
  </categoryBag>
</tModel>
```

圖 15、發佈天氣查詢服務 QoS 資訊的範例

因此，服務提供者發佈網路服務到 UDDI 前，會先創造一個擁有 QoS 資訊的 *tModel*，之後再透過 *bindingTemplate* 將 tModel 參照到網路服務。而發佈的方法則可以利用 UDDI 的 API(Application Programming Interface)，例如 UDDI4J[32]來達成。

3.4.2 服務品質資訊的更新

當服務提供者需要更新 QoS 資訊時，亦是利用 *tModelKey* 來更新 *tModel* 的內容，修改完畢後再用相同的 *tModelKey* 來儲存。而修改的權限就只有發佈服務的服務提供者才有權限可以修改網路服務和 QoS 資訊的內容，因此一個服務提供者應該要隨時的更新網路服務的 QoS 資訊，來確保資訊內容的正確性。而 UDDI 的 API，UDDI4J[32]亦可以幫助服務提供者來更新 QoS 資訊，主要的步驟如下：

(1) 以 *tModelKey* 找到 UDDI 中對應的 *tModel*

(2) 修改 *tModel* 中 *keyedReference* 的參數值

(3) 修改完後，以原本的 *tModelKey* 儲存 *tModel*，更新 QoS 資訊。

3.5 Cold Start 的問題

由於在一開始的環境中，資料庫並沒有任何使用者的行為紀錄，可提供本機制的第一階段來分析查詢語句和網路服務的相關程度。因此，為了解決 Cold Start 所產生的問題，本論文在一開始時會先利用原本 UDDI 所提供以關鍵字來查詢的 API(Inquiry API)，來幫助使用者查詢網路服務，之後再結合協同過濾的機制來提高網路服務查詢的精確率。

第四章　系統實作與模擬實驗分析

在本章中，為了衡量本論文所提出的兩階段網路服務發現機制的正確性，本論文實作了一個雛型系統稱為 WSDigger，並且設計實驗來分析評估此系統效能。因此，底下會先針對系統架構以及資料處理流程作一詳細的介紹，最後再就實驗結果來分析討論。

4.1　系統架構

本系統是以 Eclipse Platform(version 3.4.2)為開發平台，並使用 JAVA 程式語言來開發系統。其中，UDDI 註冊中心伺服器是以 JUDDI(version 2.0rc6)[4]來架設，並且利用 UDDI4J[32]來幫助服務提供者發佈網路服務，並且註冊對應的服務品質資訊到 UDDI 註冊中心的 *tModel* 中。除此之外，UDDI4J 也可以幫助服務需求者來查詢網路服務和服務品質資訊。圖 16 為 WSDigger 的架構圖。

在 WSDigger 中，主要的功能模組可以分為三個：協同過濾模組、服務品質排序模組、和使用者行為觀察模組，前兩個模組分別代表了本論文所提出兩階段網路服務發現演算法的第一階段和第二階段。當使用者發出查詢語句和服務品質的要求時，WSDigger 會透過協同過濾模組來滿足使用者對網路服務的功能需求，並且過濾掉服務品質不正確的網路服務，之後再透過服務品質排序模組來滿足使用者的非功能性需求，推薦網路服務給使用者。最後，當使用者選擇了一個網路服務，並且向服務提供者提出服務請求時，WSDigger 中的使用者行為觀察模組會將兩者之間的請求(resquest)和回應(response)訊息記錄，以及實際所偵測的服務品質資訊紀錄在歷史資料庫中，之後這些資訊將會用來改善協同過濾模組的精確率。

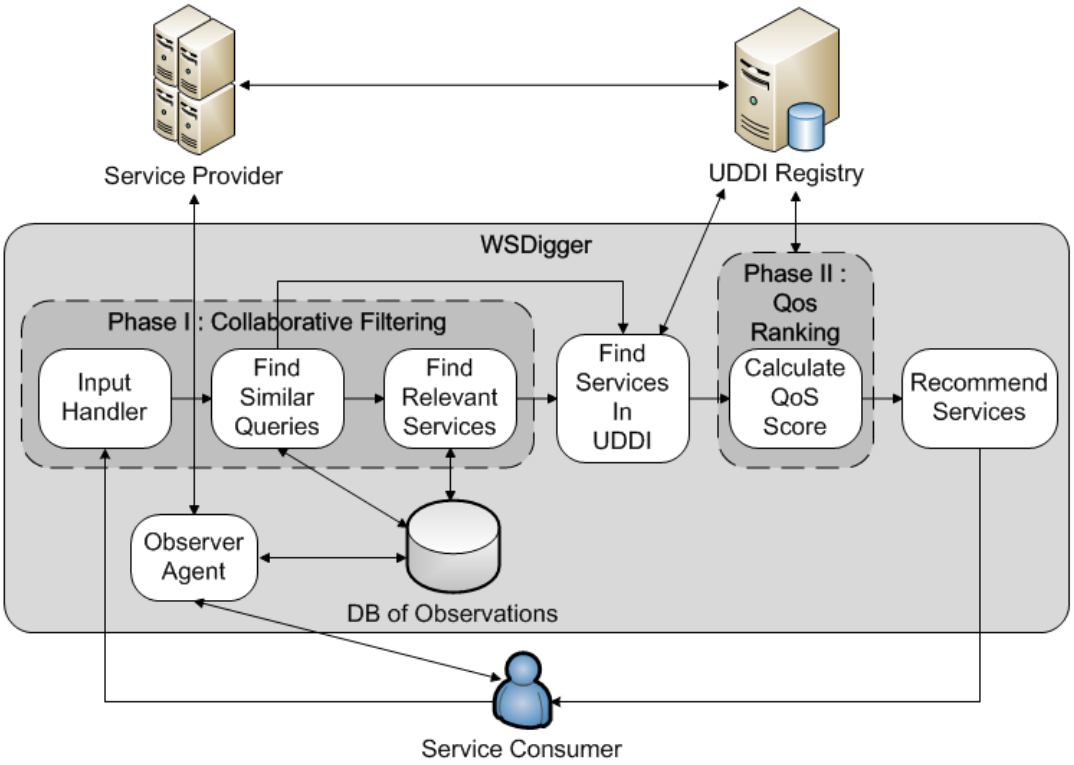


圖 16、系統架構圖

4.1.1 協同過濾模組

協同過濾模組主要的目的在於分析使用者送給 WSDigger 的查詢語句後，找出與此查詢語句相關程度較高的網路服務，此為本論文所提出兩階段網路服務發現演算法的第一階段。

在一開始時會將使用者的查詢語句依照 TF-IDF(Term Frequency-Inverse Documented Frequency)算出查詢語句中每個字詞的權重並轉換成向量。接下來，再計算此查詢語句與資料庫中其他查詢語句的相似度，然而在這裡會發生兩個需要額外處理的狀況，一是若歷史資料庫為 Cold Start 的狀態而沒有任何查詢語句歷史資料的狀況，另一是歷史資料庫中的查詢語句與使用者所下的查詢語句相似度都為 0 的狀況。在這兩個狀況下，以模組會將目標查詢語句以關鍵字的方式來查詢 UDDI 註冊中心的網路服務，並且取得網路服務集合作為 Output。

若非為上述兩種狀況，接下來協同過濾模組就會計算出各個查詢語句中與每個網路服務的相關程度，再乘上各個查詢語句與目標查詢語句的相似度，而得到所有網路服務與目標查詢語句的相關程度數值，之後依門檻值將相關程度低的網路服務過濾掉，只保留高於門檻值的網路服務集合。然而，由於歷史資料庫只存放過去使用者曾經呼叫過的網路服務，而不會有UDDI中所有的網路服務。因此，為了找出其他存在 UDDI 而且使用者也可能會感興趣的網路服務，仍然需要再將目標查詢語句以關鍵字的方式來查詢 UDDI 註冊中心的網路服務，並將所查詢到的網路服務集合與協同過濾的網路服務集合整合為一個集合作為 Output。圖 17 為協同過濾模組流程圖。

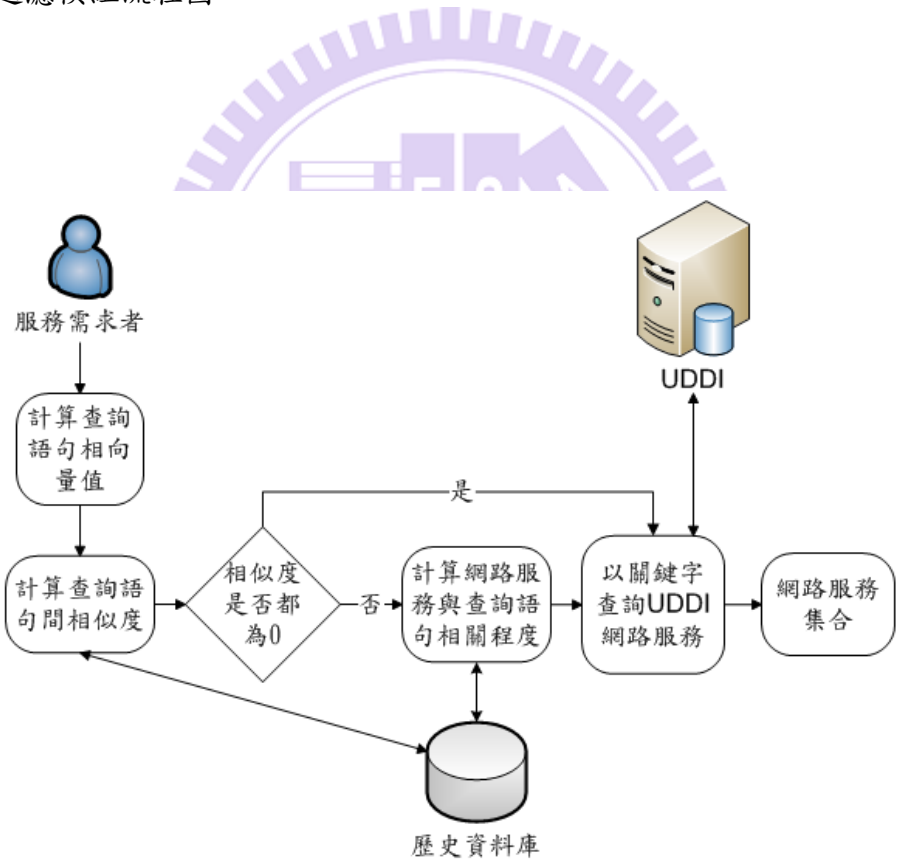


圖 17、協同過濾模組流程圖

4.1.2 服務品質排序模組

服務品質排序模組的目的在於將協同過濾模組的網路服務集合，再依照使用者的服務品質需求來排序網路服務，此為本論文所提出兩階段網路服務發現演算法的第二階段。

一開始時，由於服務品質的單位與意義都不同，為了能夠讓所有數值都在同一衡量標準下，服務品質模組會將網路服務集合的服務品質資訊作一正規劃。接下來，為了能夠滿足不同使用者對網路服務的服務品質需求，在計算服務品質分數時，會針對不同的服務品質而乘上不同的權重，最後再將個別的服務品質分數相加，就會得到個別網路服務總體的服務品質分數，而排序就根據這個分數由高到低來推薦網路服務給使用者。

4.1.3 使用者行為觀察模組

使用者行為觀察模組的目的在於，希望可以藉由記錄使用者的行為，來幫助協同過濾機制改善其所推薦網路服務的精確率。

因此，當服務需求者在 WSDigger 所推薦的網路服務集合中，選擇了一個網路服務向服務提供者請求服務時，使用者行為觀察模組會將服務需求者和提供者之間的行為紀錄到歷史資料庫中，例如：服務需求者呼叫了哪一個網路服務、網路服務是否有回應服務的請求、回應是成功訊息或失敗訊息等等。除此之外，此模組還會在服務需求者呼叫服務時，偵測網路服務實際的服務品質數值，例如：回應時間、可用度等等，來判斷此次的呼叫是否符合服務提供者所保證的服務品質資訊。

4.2 模擬實驗結果與分析

在本節中，將針對本論文所提出兩階段網路服務發現機制設計實驗，並且透過分析實驗結果來驗證本機制的正確性。

4.2.1 實驗環境與限制

由於現存而且公開的 UDDI 註冊中心，並沒有提供服務提供者發佈服務品質資訊的功能。因此，為了能夠正確驗證本論文所提出的網路服務發現機制，在實驗的環境中我們模擬了服務提供者的角色，並且將個別的網路服務給予不同的服務品質資訊發佈到實驗所架設的 UDDI 註冊中心。

為了更進一步驗證 WSDigger 可以滿足使用者對網路服務的功能和服務品質的需求，並且亦可以過濾掉服務品質不正確的網路服務。本論文參考了以往的文獻[25]在實驗中所收集的網路服務功能類別[36]，從中選了 4 個較熱門的功能類別，分別是天氣查詢、股價查詢、貨幣幣值轉換、以及語言翻譯等。每個類別中的網路服務又依照不同的服務品質數值可以分為 24 個網路服務，所以總共有 96 個網路服務，並且加以分析整理成為表 1~表 4，作為衡量實驗結果是否正確的基準表(benchmark)。

而在表 1~表 4 中，”better” 文字指的是此網路服務在服務提供者所提供的服務品質中，與其他的網路服務相比，擁有比較好的表現和競爭優勢。例如，價格較低或是回應時間較快等等。而 ”×” 符號則表示，此網路服務在服務提供者所提供的服務品質數值，和實際上的服務品質數值相比之下是比較差的。例如，服務提供者所保證網路服務的回應時間為 2 毫秒，但實際上的回應時間卻是 5 毫秒等等。

表 1、Benchmark Weather Service

Category	Web Service	QoS Info			
		Price	Response Time	Availability	Throughput
Weather	WS_a01	better			
	WS_a02	better			
	WS_a03	better			
	WS_a04	better			
	WS_a05	better	×		
	WS_a06	better		×	
	WS_a07		better		
	WS_a08		better		
	WS_a09		better		
	WS_a10		better		
	WS_a11		better ×		
	WS_a12		better	×	
	WS_a13			Better	
	WS_a14			Better	
	WS_a15			Better	
	WS_a16			Better	
	WS_a17		×	Better	
	WS_a18			better ×	
	WS_a19				better
	WS_a20				better
	WS_a21				better
	WS_a22				better
	WS_a23		×		better
	WS_a24			×	better

表 2、Benchmark Stock Service

Category	Web Service	QoS Info			
		Price	Response Time	Availability	Throughput
Stock	WS_b01	better			
	WS_b02	better			
	WS_b03	better			
	WS_b04	better			
	WS_b05	better	×		
	WS_b06	better		×	
	WS_b07		better		
	WS_b08		better		
	WS_b09		better		
	WS_b10		better		
	WS_b11		better ×		
	WS_b12		better	×	
	WS_b13			Better	
	WS_b14			Better	
	WS_b15			Better	
	WS_b16			Better	
	WS_b17		×	Better	
	WS_b18			better ×	
	WS_b19				better
	WS_b20				better
	WS_b21				better
	WS_b22				better
	WS_b23		×		better
	WS_b24			×	better

表 3、Benchmark Currency Service

Category	Web Service	QoS Info			
		Price	Response Time	Availability	Throughput
Currency	WS_c01	better			
	WS_c02	better			
	WS_c03	better			
	WS_c04	better			
	WS_c05	better	×		
	WS_c06	better		×	
	WS_c07		better		
	WS_c08		better		
	WS_c09		better		
	WS_c10		better		
	WS_c11		better ×		
	WS_c12		better	×	
	WS_c13			Better	
	WS_c14			Better	
	WS_c15			Better	
	WS_c16			Better	
	WS_c17		×	Better	
	WS_c18			better ×	
	WS_c19				better
	WS_c20				better
	WS_c21				better
	WS_c22				better
	WS_c23		×		better
	WS_c24			×	better

表 4、Benchmark Translation Service

Category	Web Service	QoS Info			
		Price	Response Time	Availability	Throughput
Translation	WS_d01	better			
	WS_d02	better			
	WS_d03	better			
	WS_d04	better			
	WS_d05	better	×		
	WS_d06	better		×	
	WS_d07		better		
	WS_d08		better		
	WS_d09		better		
	WS_d10		better		
	WS_d11		better ×		
	WS_d12		better	×	
	WS_d13			Better	
	WS_d14			Better	
	WS_d15			Better	
	WS_d16			Better	
	WS_d17		×	Better	
	WS_d18			better ×	
	WS_d19				better
	WS_d20				better
	WS_d21				better
	WS_d22				better
	WS_d23		×		better
	WS_d24			×	better

另外，在實驗中，我們也事先以手動的方式，依據 UDDI 中服務提供者對網路服務的描述，來產生出一組和網路服務功能相關的查詢語句組合，如表 5 所示。並且透過使用者模擬程式來模擬服務需求者的兩個行為，其中包括使用者的查詢服務行為以及網路服務選擇行為。在模擬查詢服務行為時，模擬程式會從查詢語句組合中隨機挑選一個查詢語句並且配合不同的服務品質需求，來發送服務查詢的請求。當 WSDigger 接受到請求時會分析查詢請求，並依照演算法推薦多個網路服務給使用者，這時模擬程式就會模擬網路服務選擇的行為，隨機從所推薦的多個網路服務中，挑選一個網路服務來呼叫，完成整個網路服務查詢的流程。

表 5、Query Sets for WS、SS、CS、TS

Weather service(WS)	'current weather prediction'
	'weather condition'
	'current weather'
Stock service(SS)	'stock quote'
	'company stock quote'
	'company stock price'
Currency service(CS)	'currency rate'
	'currency conversion rate'
	'get currency conversion rate'
Translation service(TS)	'multi language translation'
	'text translation'
	'language translation'

4.2.2 實驗案例設計

在實驗中，為了過濾與查詢語句相關程度太低的網路服務，本實驗的門檻值設定為 0.022。另外，針對個別服務品質的權重，則依據使用者對服務品質重視程度的排名，依序設定為 0.6、0.2、0.15、0.05 等 4 個權重。

因此，為了驗證依據本機制所推薦網路服務的正確性，在接下來的實驗中，將會用到上述的設定值，針對本實驗所選的 4 個服務品質，分別給予第一的排序來進行 4 個實驗，如表 6 所示，並且在實驗中比較本機制與協同過濾方法和服務品質排序方法的效能。

表 6、服務品質排序設定

實驗	服務品質排序
實驗一	價格、反應時間、可用度、總處理量
實驗二	反應時間、價格、可用度、總處理量
實驗三	可用度、反應時間、價格、總處理量
實驗四	總處理量、反應時間、可用度、價格

4.2.3 實驗評估指標

為了評估本機制的效能，評估的指標必須要能夠驗證所推薦的網路服務當中，有多少比率是符合服務需求者功能性和服務品質的需求，並且是服務品質正確的網路服務。因此，本實驗會以表 1~表 4 所分析整理出來的基準表(benchmark)來和所推薦的網路服務進行比對，若推薦越多符合使用者需求的服務，且這些服務的排名都越在前面時，則演算法的正確性越高。例如，當使用者想要找的是價格便宜且是查詢天氣的網路服務，那麼對照表 1 來看，演算法應該推薦的網路服務為 WS_a01、WS_a02、WS_a03、和 WS_a04 等四個網路服務。

而驗證正確性的指標則為 Precision 和 Average Precision。如圖 18 所示，左邊 CorrectWS 的區域代表的是正確的網路服務集合，右邊 RetrievedWS 的區域代表的是演算法所推薦的網路服務集合，而中間兩者交集的部分就是演算法找出來而且是正確的網路服務集合。因此，計算 Precision 的公式如下：

$$Precision = \frac{CorrectWS \cap RetrievedWS}{RetrievedWS}$$

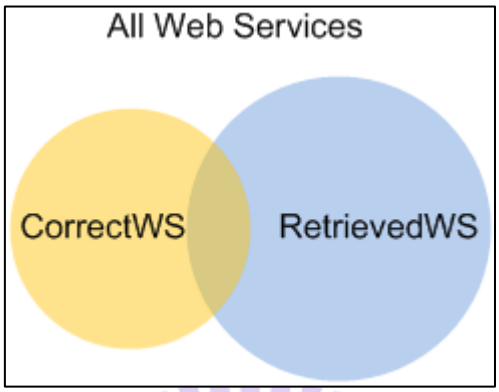


圖 18、CorrectWS 與 RetrievedWS 交集圖

此外，由於一個查詢語句經由演算法所推薦的網路服務是經過排序的，為了考量到正確的網路服務在所推薦的網路服務集合當中的排名，Average Precision 將依據正確網路服務的排名所計算出的各別 Precision 數值加總並且平均，這會使得當正確的網路服務排名越前面時，Average Precision 的數值也就越高。計算的 Average Precision 公式如下，其中r表示正確的網路服務在所推薦的網路服務集合中的排名，N代表正確的網路服務在所推薦的網路服務集合中的個數。

$$AvePrecision = \frac{\sum precision(r)}{N}$$

那以下就舉一個計算 Average Precision 的例子。在這裡使用者所下的查詢語句為*"current weather"*，並且想要價格比較低的網路服務，因此對照 benchmark 表 1 來看的話，正確的網路服務集合應為$\{WS_1 、WS_2 、WS_3 、WS_4\}$，而實際上系統所推薦的網路服務集合則為下表所示 ：

表 7、查詢結果範例

Service ranking for query *"current weather"* :		
1. WS_3 •	**9. WS_19**	**17. WS_8**
2. WS_2 •	**10. WS_6**	**18. WS_17**
3. WS_7	**11. WS_5**	**19. WS_14**
4. WS_10	**12. WS_4 •**	**20. WS_13**
5. WS_18	**13. WS_1 •**	**21. WS_24**
6. WS_16	**14. WS_12**	**22. WS_23**
7. WS_15	**15. WS_11**	**23. WS_21**
8. WS_22	**16. WS_9**	**24. WS_20**

因此，從表 7 我們可以看到正確的網路服務分別是排在第 1、第 2、第 12、和第 13 個位置，那麼這次查詢結果的 Average Precision 數值就如下所示：

$$AvePrecision = \frac{\left(1/1 + 2/2 + 3/12 + 4/13\right)}{4} = 0.64$$

4.2.4 實驗結果分析

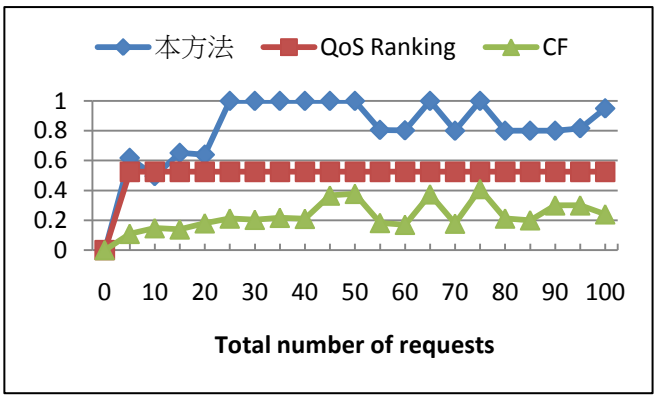
實驗一：以價格為最重要考量的服務查詢模擬實驗

在實驗一中，服務品質排序為價格、反應時間、可用度、和總處理量，亦即在 4 個服務品質中價格是最重要的考量。接下來，實驗一會針對 4 種不同功能的網路服務類別以及 3 個不同的演算法，分別發出 100 次的服務查詢請求，並且計算每次查詢的 Average Precision。

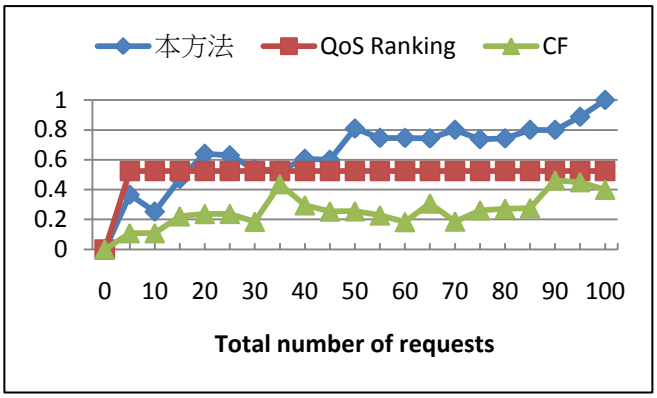
如圖 19 所示，本方法在 4 個不同功能的網路服務中，一開始的查詢精確率會是比較低的，這是因為一開始資料庫中的歷史資料太少，而無法提供本方法準確的分析推薦服務。但是隨著查詢次數的增加，歷史資料增多了之後，精確率也開始逐漸提高，到最後第 90 到第 100 次的查詢中，精確率都可以達到 80%以上的水準。

而 QoS Ranking 方法在 4 個不同功能的網路服務中，從一開始的到最後第 100 次查詢的精確率都是 52.5%，這是因為 QoS Ranking 只有依據使用者的服務品質需求來排序網路服務，因此在每一次的查詢中所有網路服務的排序都是相同的，亦即價格較低的網路服務都是在排名的最前面。此外，由於 QoS Ranking 只有考量到使用者的服務品質需求，而無法過濾服務品質不正確的網路服務，使得其精確率就只能維持在 52.5%的水準，而無法增加。

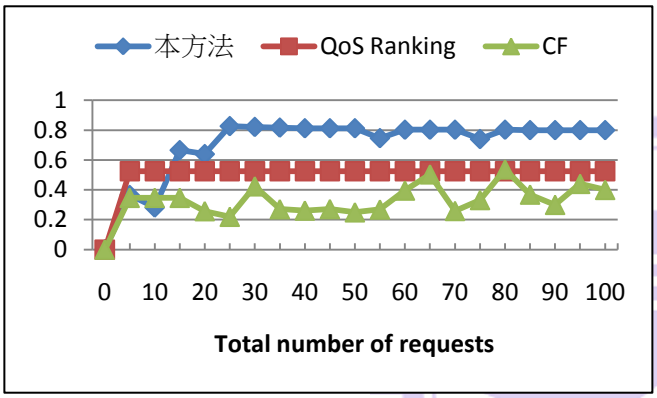
至於協同過濾方法(Collaborative Filtering, CF)則在 4 個不同功能的網路服務中，精確率都是最低的。雖然隨著查詢次數以及歷史資料量的增加，讓 CF 的精確率也有緩慢的爬升，但是因為 CF 在網路服務的排序上只有考量到查詢語句和網路服務之間的相關程度，而沒有加入使用者服務品質需求的考量，導致 CF 的精確率無法很快的達到較高的水準，使得在最後第 90 到第 100 次的查詢中，精確率的水準仍然只有在 20%到 40%之間。



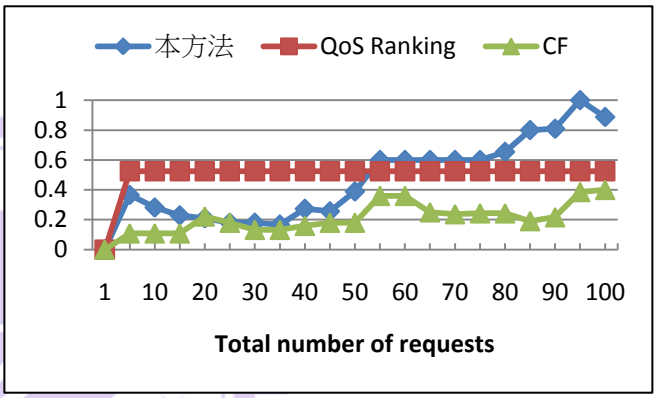
(a) Weather service with better Price



(b) Stock service with better Price



(c) Currency service with better Price



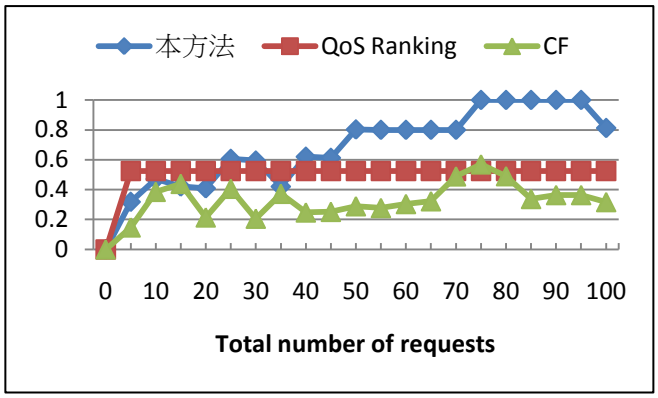
(d) Translation service with better Price

圖 19、與現有二類方法的 Average Precision 比較圖(實驗一)

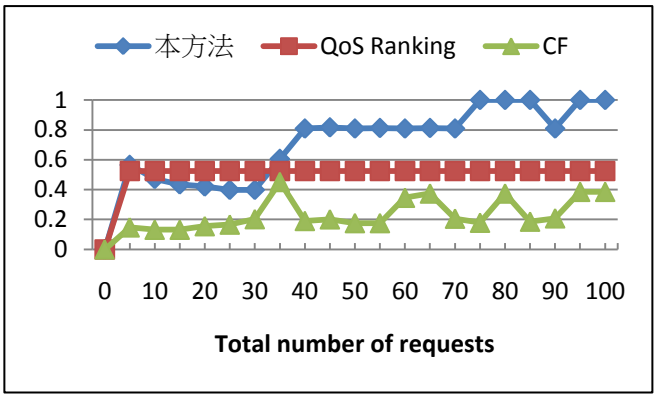
實驗二：以反應時間為最重要考量的服務查詢模擬實驗

實驗二中，實驗的環境大致與實驗一相同，唯一不同的地方在於，實驗二的服務品質排序為反應時間、價格、可用度、和總處理量。

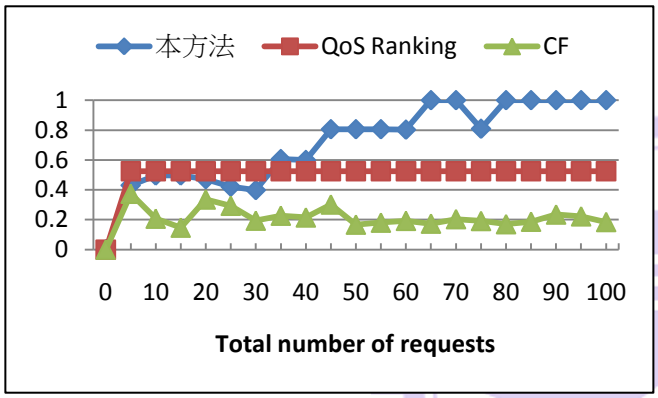
從圖 20 可以看到，實驗二的結果中，本方法在最後第 90 次到第 100 次時的精確率，在 4 個網路服務的類別下仍然都可以有 80%以上的精確率。而 QoS Ranking 方法仍然只有 52.5%的精確率，另外 CF 的方法在第 90 次到第 100 次時的精確率一樣只有 20%到 40%的精確率。



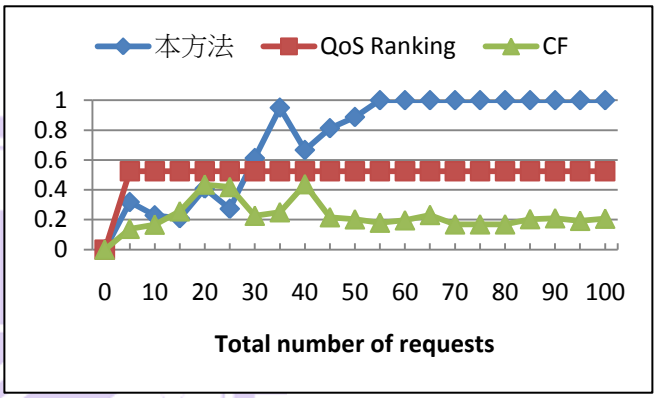
(a) Weather service with better Response Time



(b) Stock service with better Response Time



(c) Currency service with better Response Time



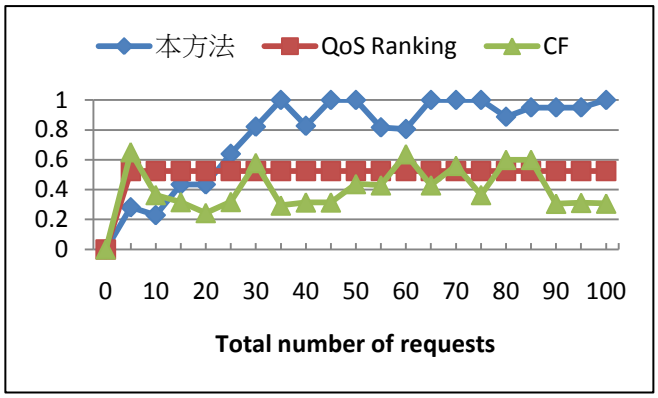
(d) Translation service with better Response Time

圖 20、與現有二類方法的 Average Precision 比較圖(實驗二)

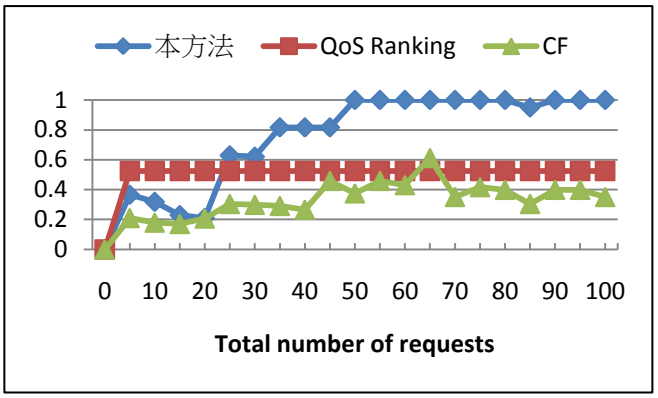
實驗三 ： 以可用度為最重要考量的服務查詢模擬實驗

在實驗三中，實驗的環境大致與實驗一相同，唯一不同的地方在於，實驗三的服務品質排序為可用度、反應時間、價格、和總處理量。

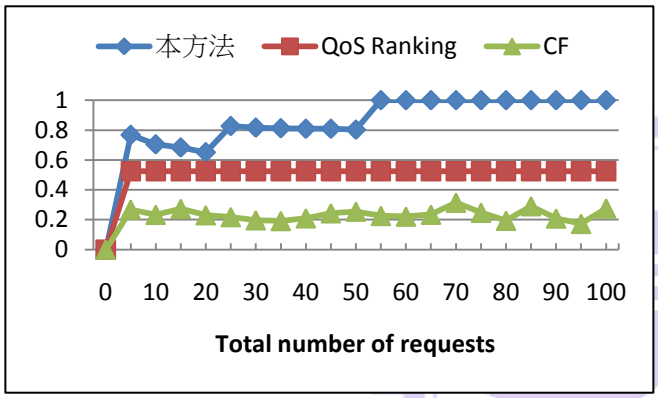
從圖 21 可以看到，本方法在最後第 90 次到第 100 次時的精確率，和實驗一和二相比有著較高的精確率，在 4 個網路服務的類別下都可以有 90%甚至將近 100%的精確率。而 QoS Ranking 方法仍然只有 52.5%的精確率，另外 CF 的方法在第 90 次到第 100 次時的精確率卻只有 20%到 40%的精確率。



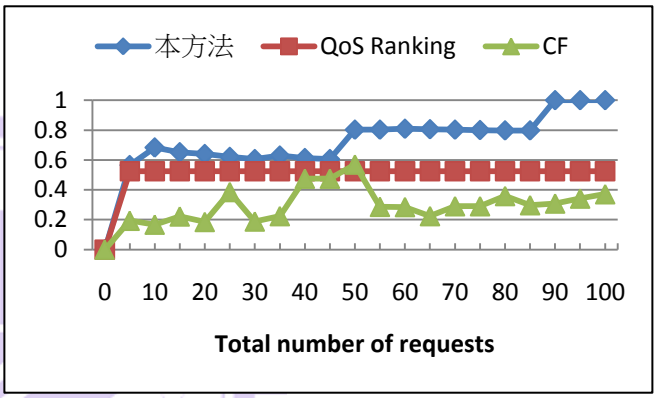
(a) Weather service with better Availability



(b) Stock service with better Availability



(c) Currency service with better Availability



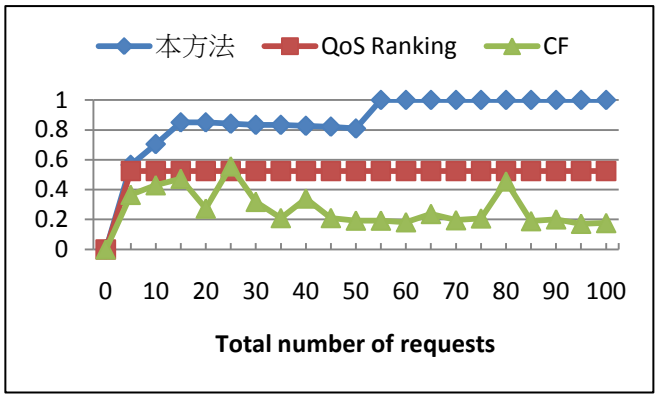
(d) Translation service with better Availability

圖 21、與現有二類方法的 Average Precision 比較圖(實驗三)

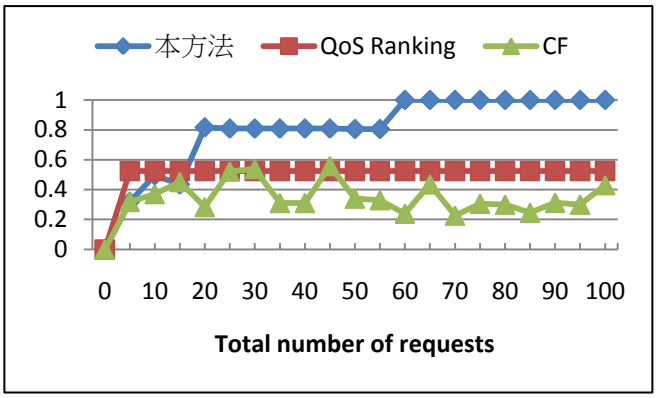
實驗四 ： 以總處理量為最重要考量的服務查詢模擬實驗

在實驗四中，實驗的環境大致與實驗一相同，唯一不同的地方在於，實驗四的服務品質排序為總處理量、反應時間、可用度、價格。

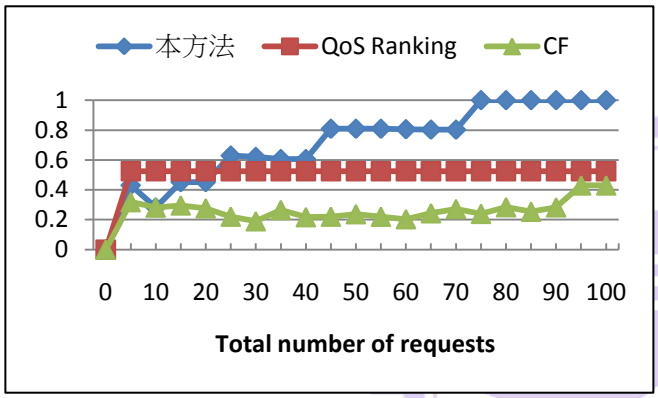
從圖 22 可以看到，本方法在最後第 90 到第 100 次時的精確率，在 4 個網路服務的類別下有將近 100%的精確率，是 4 個實驗中精確率最高的。而 QoS Ranking 則仍然只有 52.5%的精確率，另外 CF 的方法在第 90 次到第 100 次時的精確率則仍然只有 20%到 40%的精確率。



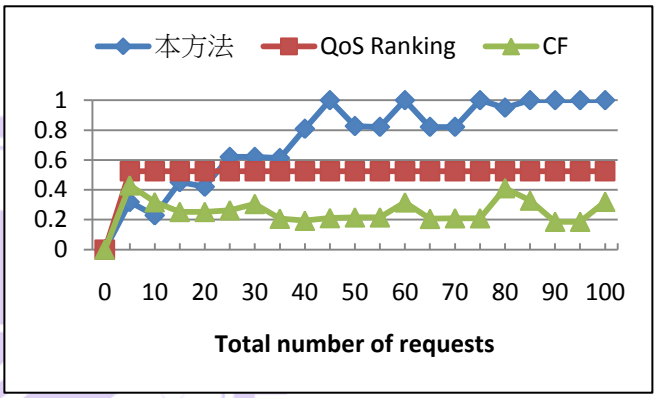
(a) Weather service with better Throughput



(b) Stock service with better Throughput



(c) Currency service with better Throughput



(d) Translation service with better Throughput

圖 22、與現有二類方法的 Average Precision 比較圖(實驗四)

4.2.5 總結

在這個章節中，本論文利用 Average Precision 來衡量每次查詢所找到正確網路服務的機率，並且針對 4 個服務品質排序設計了 4 個實驗，而在每個實驗中則以 4 個網路服務類別來呈現實驗的結果。總的來說，綜合上述 4 個實驗的結果，我們可以看到本方法在不同的服務品質排序以及不同的網路服務類別中，Average Precision 的表現都是優於 QoS Ranking 方法和協同過濾方法。

又如圖 23 所示，其是統合 4 個實驗中，各方法在不同網路服務類別的 Average Precision 數值，並且再加以平均計算後而得的總覽圖。從圖中我們可以看到 QoS Ranking 的精確率都只能維持在一開始的水準，無法有進一步的改善，這是因為 QoS Ranking 只能依據使用者的需求來排序網路服務，而無法排除服務品質不正確的網路服務。另外，雖然協同過濾方法和本方法的精確率，都會隨著查詢次數的增加而上升，但是由於協同過濾方法沒有將使用者的服務品質需求，考量到網路服務的排名中，因此精確率增加的速度較慢，到最後第 100 次查詢時的精確率最高亦只有 30%左右的水準。相對而言，本方法則隨著查詢次數的增加下，精確率的爬升速度相當快，大約在第 50 次的查詢次數左右就能夠達到 80%左右的精確率水準，最後到第 100 次的查詢時更達到 96%的精確率。這樣的實驗結果，同時這也意味著，本方法能夠利用使用者過去的服務查詢行為經驗，來幫助其他的使用者更快的找到適合且服務品質好的網路服務，亦即當使用者行為的歷史紀錄越多時，所推薦的網路服務也會更準確。

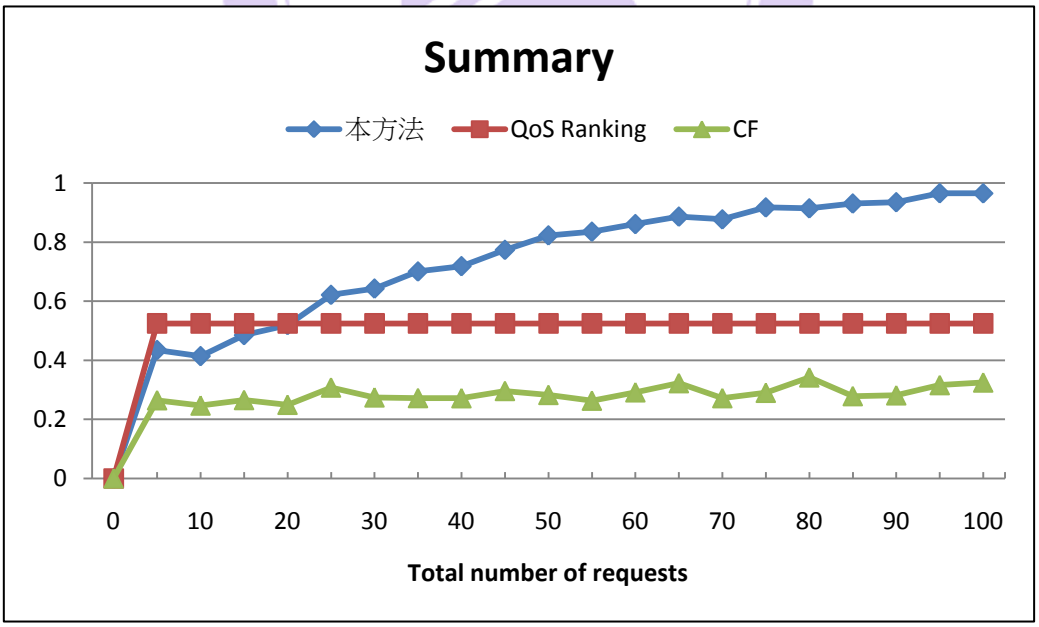


圖 23、與現有二類方法的 Average Precision 比較圖(Summary)

第五章　結論與未來方向

5.1　結論

隨著網路服務技術的快速發展，越來越多的服務提供者都提供了相同功能的服務，為了幫助使用者能夠從這群相同功能的服務當中選出一個好的服務，解決服務選擇的問題，並且驗證網路服務服務品質的正確性，來過濾掉服務品質不好的網路服務，本論文提出了一個基於服務品質和協同過濾機制的兩階段網路服務發現演算法。

在第一階段是利用過往使用者的查詢行為，並且透過協同過濾來滿足使用者的功能性需求，以及驗證服務品質的正確性，過濾掉不好的網路服務。因此，在這階段中，本方法不用收集任何使用者對網路服務的評分，避免了惡意使用者或錯誤評分的問題。第二階段再依據使用者的服務品質需求，來排序第一階段所推薦的網路服務集合，使得越符合使用者需求的網路服務，排名會再越前面，如此不僅可以滿足使用者對網路服務的非功能性需求，亦能提升網路服務查詢的精確率。

最後，在實驗中我們以 Average Precision 為評估指標，並且實作了一雛型系統稱為 WSDigger，來比較本方法與 QoS Ranking 方法和協同過濾方法的效能。實驗的結果也指出，本方法的效能可以達到 96%的精確率，其效能是明顯優於其他兩個方法。

5.2　未來方向

在未來研究方面，由於本論文在查詢網路服務時只有以文字來考量，一開始是以關鍵字為基礎的查詢方式，之後再利用協同過濾機制來提升精確率。因此，

在這裡可能可以結合語意(Semantic)的概念，除了能夠更準確的找出使用者有興趣的網路服務之外，更希望可以建構網路服務的本體論(Ontology)，來描述網路服務之間的關連性，提供更有效率的網路服務查詢的方法。此外，不管是在 UDDI 中或是使用者所發出的查詢請求中，本論文假設服務品質資訊的單位都是統一的預設值，但是服務品質的單位在實際上會有多個定義，因此，在未來也希望可以建構服務品質的本體論，讓不同的服務品質單位之間能夠自動的轉換數值。

參考文獻

[1] A. M. Daniel. “QoS Issues in Web Services”. IEEE Internet Computing, 6(6), 2002.

[2] A. Ali, O. Rana, R. Al-Ali, and D. Walker, “UDDIe: An Extended Registry for Web Services”, Proc. of 2003 Symposium on Applications and the Internet Workshops, 2003, pp.85-89.

[3] A. F. M. Huang, C.-W. Lan, and S. J. H. Yang, “An optimal QoS-based web service selection scheme”. Information Sciences 179 , 2009, pp. 3309-3322.

[4] Apache Software Foundation. "Welcome to jUDDI". Retrieved May 2010 from http://ws.apache.org/juddi/

[5] A. Blum, “UDDI as an Extended Web Services Registry: Versioning, quality of service, and more”. White paper, SOA World magazine, Vol. 4(6), 2004.

[6] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-Based Collaborative Filtering Recommendation Algorithms,” Proc. 10th Int’l WWW Conf., 2001.

[7] D. Gouscos, M. Kalikakis, and P. Georgiadis, “An Approach to Modeling Web Service QoS and Provision Price”. In Proc. of the 1st Int. Web Services Quality Workshop - WQW 2003, Rome, Italy, 2003, pp.1-10.

[8] E. Blanzieri, P. Giorgini, P. Massa, S. Recla. “Implicit culture for multi-agent interaction support”. In: Proceedings of the International Conference on Cooperative Information Systems (CooplS). Volume 2172 of LNCS., Springer, 2001, pp. 27–39

[9] E. M. Maximilien and M. P. Singh. “Reputation and Endorsement for Web Services”. ACM SIGecom Exchanges, 3(1):24–31, 2002.

[10] E. M. Maximilien and M. P. Singh. “A Framework and Ontology for Dynamic Web Services Selection”. IEEE Internet Computing, 8(5):84–93, Sept. 2004.

[11] E. Al-Masri and Q. H. Mahmoud, “QoS-based Discovery and Ranking of Web Services”, in IEEE 16th International Conference on Computer Communications and Networks, 2007, pp. 529-534.

[12] E. M. Maximilien and M. P. Singh, “Self-adjusting trust and selection for web services,” June 2005, pp. 385–386.

[13] H. L. Vu, M. Hauswirth, and K. Aberer. “QoS-based service selection and ranking with trust and reputation management”. Technical Report IC2005029, Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland, June 2005.

[14] IBM Corporation (2003). “Web Service Level Agreement (WSLA) Language Specification” Ver. 1.0. Retrieved April 30, 2006 from http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[15] J. Li, D. Ma, J. Han, X. Long, “Toward Trustworthy Semantic Web Service Discovery and Selection, Autonomic and Trusted Computing”, LNCS, 5586, Springer, 2009, pp. 209-220.

[16] J. Yan, J. Piao, “Towards QoS-Based Web Services Discovery Service-Oriented Computing”, ICSOC 2008 Workshops, LNCS, 5472, Springer, 2009, pp. 200-210.

[17] J. L. Herlocker, J. A. Konstan, and J. Riedl, “Explaining collaborative filtering recommendations”. In Proceedings of the 2000 Conference on Computer Supported Cooperative Work, 2000, pp. 241–250.

[18] J. Wang, A. P. de Vries, and M. J. Reinders. “Unifying user-based and item-based collaborative filtering approaches by similarity fusion”. In Proc. of SIGIR, 2006.

[19] J. Wang, A. P. de Vries, and M. J. Reinders. “Unified relevance models for rating prediction in collaborative filtering”. ACM Trans. on Information System (TOIS), 2008.

[20] K. Karta. “An investigation on personalized collaborative filtering for web service selection”. Honours Programme thesis, University of Western Australia, Brisbane, 2005.

[21] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. “QoS-Aware Middleware for Web Services Composition”. IEEE Transactions on Software Engineering, 30(5):311–327, May 2004.

[22] M. Tian, A. Gramm, T. Naumowicz, H. Ritter, J. Schi, “A Concept for QoS Inte-gration in Web Services”, Proceeding of WISEW, 2003.

[23] M. P. Papazoglou. “Service-oriented computing: concepts, characteristics and directions”. In Proceedings of the Fourth International Conference on Web Information Systems Engineering, pages 3–12, Dezember 2003.

[24] M. Kerrigan, “Web Service Selection Mechanisms in the Web Service Execution Environment (WSMX)”, In Proceedings of the 21st Annual ACM Symposium on Applied Computing (SAC), Apr 2006, Dijon, France.

[25] N. Kokash, A. Birukou, and V. D’Andrea, “Web Service Discovery Based on Past User Experience,” Proc. Int’l Conf. Business Information Systems (BIS 07), LNCS 4439, Springer, 2007, pp. 95–107.

[26] R. Sreenath, R., and M. P. Singh, “Agent-based Service Selection”. Journal of Web Semantics, Volume 1, Issue 3, 2004.

[27] O. Martin-Diaz, A. Ruiz-Cortes, R. Corchuelo, and M. Toro, “A Framework for Classifying and Comparing Web Services Procurement Platforms”, Proc. of 1st Int’l Web Services Quality Workshop, Italy, 2003, pp. 37-46.

[28] R. Wishart, R. Robinson, J. Indulska, and A. Josang, “SuperstringRep:

Reputation-enhanced Service Discovery". In Proc. of the 28th Australasian conf. on Computer Science, Vol. 38, 2005, pp.49-57.

[29] S. Barry, B. Evelyn, B, Peter, C. Maurice, and F. Jill, "Collaborative Web Search". In Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI-03, pp. 1417–1419. Morgan Kaufmann, 2003. Acapulco, Mexico.

[30] S. Ran, "A Model for Web Services Discovery with QoS", ACM SIGecom Exchanges 4(1), 2003, pp. 1-10.

[31] S. Majithia, A. Shaikhali, O. Rana, and D. Walker, "Reputation-based Semantic Service Discovery". In Proc. of the 13th IEEE Intl. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. (WETICE), 2004, pp.297-302, Modena, Italy.

[32] SourceForge.net. "Introduction". Retrieved April 2010, http://uddi4j.sourceforge.net/

[33] T. Xu, "Reputation-Enhanced Web service discovery with QoS", Ph.D. Dissertation, School of Computing, Queen's University, Canada, 2006.

[34] U. S. Manikrao, T.V. Prabhakar, "Dynamic Selection of Web Services with Recommendation System", International Conference on Next Generation Web Services Practices, August 2005, Seoul, Korea.

[35] UDDI Version 3.0.2 Specifications, October 2004, http://uddi.org/pubs/uddi_v3.htm.

[36] WebserviceX.NET, http://www.webservicex.net/WCF/default.aspx.

[37] Z. Xu, P. Martin, W. Powley, and F. Zulkernine. "Reputation-Enhanced QoS-based Web Services Discovery". In Proceedings of ICWS'2007, Salt Lake City, Utah, USA, 2007.

[38] Z. Zheng, H. Ma, M. R. Lyu, and I. King. "WSRec: A collaborative filtering based web service recommender system". In ICWS, 2009.