東海大學

資訊工程研究所

碩士論文

指導教授: 林祝興博士

基於協同過濾演算法的大數據分析及其效能評估

Big Data Analysis and Performance Evaluation Collaborative Filtering Algorithms

研究生: 蕭伊廷

中華民國 106 年 7 月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 蕭 伊 廷 所提之論文

基於協同過濾演算法的大數據分析及其效能評估

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 陳筱雯 簽章

委 員 賴威伸

劉榮秀

溫嘉霖

指 導 教 授 林熙禎 簽章

中華民國 106 年 7 月 14 日

摘　要

為了滿足不同巨量資料使用者的目的與需求，各種推薦演算法陸續被提出來與運用，其中最為著名的是協同過濾推薦演算法。根據過去的研究，在協同過濾 演算法中，若使用皮爾森相關係數會在特別的情況下出現錯誤。本篇論文使用正常恢復相似性度量 (Normal Recovery Similarity Measure) 修正其相似值，以修正協同過濾推薦演算法的誤差值，並以此為協同過濾演算法的基礎。我們將之實作在雲端 Hadoop 環境中，並且測量在具有 3、6 以及 9 個節點等情況下的執行時間，與單機執行的時間作比較，進而分析其加速比以及效能。

關鍵字**:** 協同過濾、Hadoop、Pearson Correlation Coefficient、MapReduce.

Abstract

In order to satisfy various demands of different Big Data applications, many recommendation algorithms have been proposed, in which the collaborative filtering approach has been widely adopted. In certain situations, the Pearson correlation coefficient used in the collaborative filtering algorithm will be incorrect. In this thesis, we propose to use the Normal Recovery Similarity Measure to modify the similarity value in order to reduce the error of collaborative filtering recommendation algorithm. We implemnet the proposed collaborative filtering system on a stand alone PC and a cloud computing environment with 3, 6 and 9 nodes. The execution time and performance of the proposed system are measured and analyzed. From the experimental results, we find that, by cloud computing, performance of the proposed collaborative filtering system can be effectively enhanced.

Keywords: Collaborative Filtering ,Hadoop ,Pearson Correlation Coefficient , MapReduce.

Table of Contents

List of Figures

List of Tables

第一章　簡介

現今社會的網路資訊科技迅速發展，每天每個人都離不開網路的使用，這成就了今日的巨量資料流動，所以我們對於巨量資料的運用就成為了非常重要的課題。許多的公司都致力於如何才能滿足消費者的需求，首先就必須要了解使用者需要甚麼，而如何推薦消費者需要或者喜歡的東西就是滿足消費者的第一個步驟，也是最重要的一個步驟，我們可以經由消費者的使用習慣以及喜好來做推薦，巨量的資料就可以作為我們分析的依據，而大數據分析更是成為了與生活息息相關的一環。

巨量資料有四個特性 (4V)，第一個是資料量龐大（Volume），每天都有 200 億的 E-mail 在發送，資料規模非常龐大，因此必須對產生的大量數據有能力保 存及處理。第二個是變化非常快速（Velocity），對於資料的處理與反應的時效 必須即時快速。第三個是資料型態的複雜多樣性（Variety），使用者使用資料型態非常多樣，資料的來源有文字、圖像、網頁、串流資料、剪輯視頻、音樂作品，各種不同的資料型態。第四個是價值（Value），大數據有很高的價值，雖然目前能提出的有效資訊大約 3%，如何將原始資訊轉換成為有用的訊息是一門重要的學問，例如資料探勘。[1]

巨量資料中為了滿足不同使用者的目的與需求，推薦演算法因此而產生，其中協同過濾推薦演算法就是其一。目前的協同推薦演算法著重在單機上的設計，為了應對巨量資料的趨勢，能夠讓系統能夠在當下知道使用者的興趣，及時滿足使用者需求，對於資料的處理速度已是決定性的關鍵。單機的執行速度已經無法滿足即時性的需求，因此，雲端與協同過濾演算法結合具有實現的價值。

根據過去的研究，在協同過濾演算法中，若使用皮爾森相關係數會在特別的情 況下出現錯誤。本篇論文使用 Normal Recovery Similarity Measure 修正其相似 值，以修正協同過濾推薦演算法的誤差值，並以此為協同過濾演算法的基礎。 我們將之實作在雲端 Hadoop 環境中，並且量測在具有 3、6 以及 9 個節點等情 況下的執行時間，與單機執行的時間作比較，進而分析其加速 (speedup) 以及 效能。在實驗中發現，在 9 個節點的情況下能夠獲得最多 2.58 倍的加速比，在第二個實驗中，我們實作了 Jaccard Similarity、皮爾森相似度、正常恢復相似性度量的預測結果分析，得知在資料集較稀疏以及推薦一個項目時，這三種演算法的推薦效果會相同。

本論文一共分為五章，本章說明簡介，在第二章說明本次論文所使用的演算法及相關知識，在第三章說明實驗環境與實驗步驟，第四章說明實驗的結果，第五章則說明結論與未來展望。

第二章　相關文獻探討

2.1　協同過濾演算法

協同過濾演算法 (Collaborative Filtering) 是利用擁有類似經驗之群體所顯示的偏好，來預測使用者未呈現的偏好資訊。協同過濾演算法最主要分為兩類：以記憶為基礎的協同過濾、以模型為基礎的協同過濾。其中以記憶為基礎的協同過濾演算法又分為以使用者為基礎的協同過濾演算法 (user-based) 與以項目為基礎的協同過濾演算法 (item-based)，如圖 2.1。

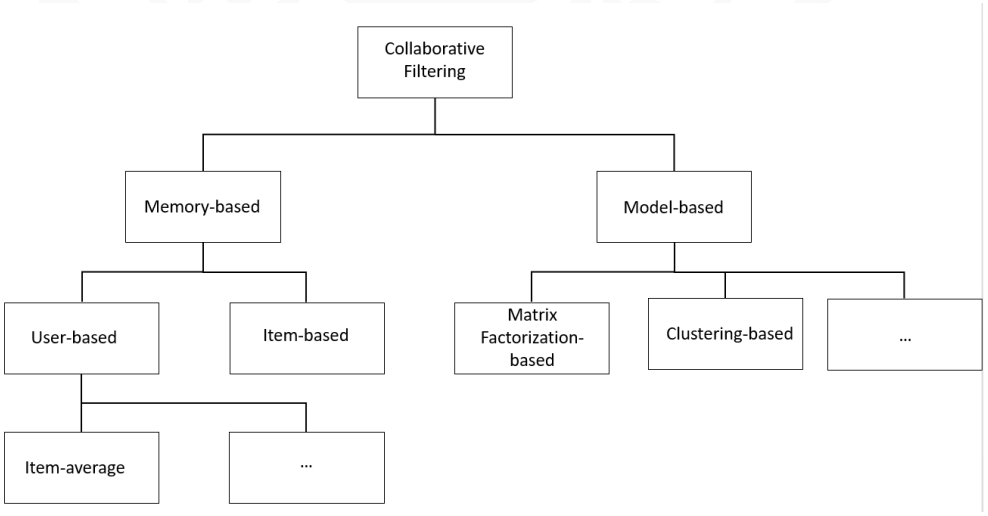

Figure 2.1: 協同過濾演算法的分類

協同過濾的優點為：(1). 因為不需要分析項目的特徵，能有效的將原本難以分析的項目做推薦，例如: 音樂。(2). 因為共用其他人的興趣與經驗，可以避免分析項目本身的特性或不精確的內容而導致的錯誤預測。(3). 使用者可能存在著隱藏偏好，經過預測之結果會是使用者跟原本的偏好不同的項目，發現使用

者的隱藏偏好。(4). 能夠有效的利用其他相似使用者的回饋資訊，來做為個性化的依據，提高個性化學習的速度。[2] 協同過濾的缺點為：在歷史資訊不足的時候，系統的推薦會非常的不準確，因為缺乏相似的項目或使用者來做判斷，且在新加入的項目或使用者也會有相同情形，因為歷史資料的不足而無法準確的預測該新加入使用者的興趣偏好。以使用者為基礎的協同過濾演算法適合使用在項目數遠大於使用者時，且使用者的變化較少時；以項目為基礎的協同過濾演算法適合使用在使用者個數遠大於項目個數，且項目變化較少時。由於本實驗中的項目數較多且固定，所以本文是採用以使用者為基礎的協同過濾演算法。

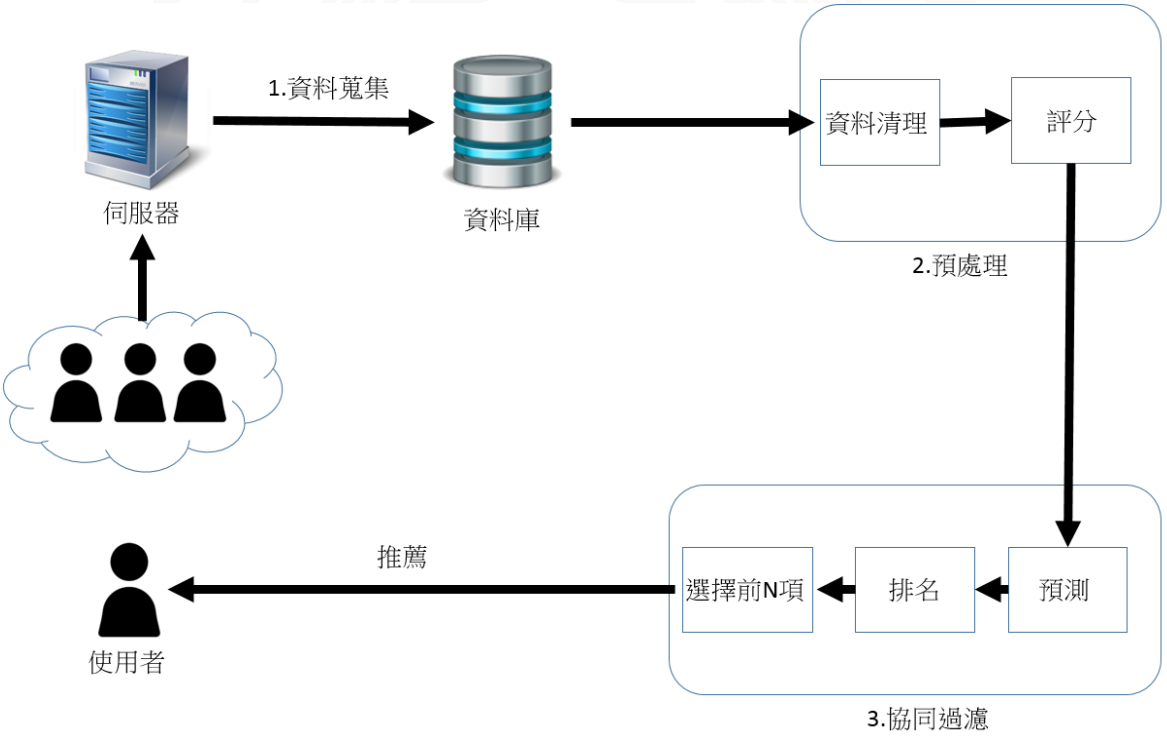

Figure 2.2: 協同過濾系統架構

如圖 2.2 所示，典型協同過濾推薦系統的框架包括：1）數據採集;2）預處理;3）協同過濾。首先，通過無線網絡收集用戶數據並存儲在資料庫中。接著進行預處理，正確的預處理操作對於確保數據的完整性和可靠性至關重要。基於這些數據，實施協同過濾演算法以預測用戶興趣並推薦相關項目，以節省時間和精力。

2.1.1 協同過濾演算法的資料收集

數據收集是整個推薦系統的基礎。收集的數據主要分為四類：用戶資料，產品資料，用戶行為和用戶評分。

A. 用戶資料: 許多企業要求用戶在伺服器上註冊，並在使用服務之前填寫個人資料。個人資料通常包括姓名，電話，性別，愛好等等。基於對上述用戶資料的分析，企業可以更具體地建立用戶檔案，並向移動客戶推送促銷信息。

B. 產品資料: 商家傾向於根據自己的功能、品牌、價格等對商品進行分類。 例如，視頻網站通常會在其視頻中添加標籤，以幫助消費者可以更方便地找到 自己喜歡的東西。

C. 用戶行為: 在瀏覽網站或聽音樂的同時，用戶可能被存儲大量行為數據的服務器進行監視，例如歌曲的播放持續時間、書的購買日期、甚至是數位網頁上的點擊次數。這些數據通常是大量的，需要通過特定的數據挖掘方法進行分析。

D. 用戶評分: 一些網站提供評分系統，並指導消費者評估他們所經歷的項目，如電影，歌曲和網絡服務。這些評分反映了消費者的喜好，並受到企業越來越多的關注。如果有效利用上述數據，可以在推薦系統中發揮重要作用。協同過濾不需要有關用戶（用戶資料）和項目（產品資料）的任何信息，它側重於用戶的反饋，包括顯性反饋（用戶評分）和隱性反饋（用戶行為）。這兩種數據的主要特點總結在表 2.1 中。[8]

Table 2.1: 用戶行為與用戶評分比較[8]

特性	用戶行為	用戶評分
資料大小	大	小
結構	較無結構化，通常記錄在日誌當中	結構化資料，容易將其使用
覆蓋範圍	非常廣，包含所有使用者紀錄	狹窄，部分使用者會在使用過後評分
主觀/客觀	客觀	主觀
應用難度	困難	容易
可靠性	不穩定	穩定

2.1.2 資料預處理

隨著行動網路的發展，由於用戶設備的多樣性和網路的異構性，收集的數據通常是各種格式的。因此，資料預處理已經成為推薦系統不可或缺的一部分，它負責確保協同過濾的輸入數據能夠完整和可靠。預處理通常分為以下三個步驟。

A. 資料清理: 原始數據可能由於設備故障或傳輸錯誤產生的雜訊數據的存在而不能直接利用，誤差率可能變得非常高。此外，有些消費者可以任意評估項 目，例如給予所有項目最高等級來節省時間，這可能降低整體評級信息的可靠 性。

B. 產生隱藏評分: 大多數協同過濾推薦系統只將明確的用戶評級視為有價值的信息。然而，大部分用戶並不總是對已經使用的項目進行評估，這導致數據稀疏的問題。由於廣泛應用的移動客戶端，特定的用戶行為被收集並儲存在雲中，具有巨大的潛在價值，這可能成為減輕此問題的關鍵。例如，推薦系統收集大量用戶評級和行為作為訓練集，然後對其進行特定的機器學習算法，如神經網絡或決策樹，構建可以將用戶行為轉化為隱含級別的預測模型，如圖 2.3 所示。通過這種方式可以大大提高評級分數據。

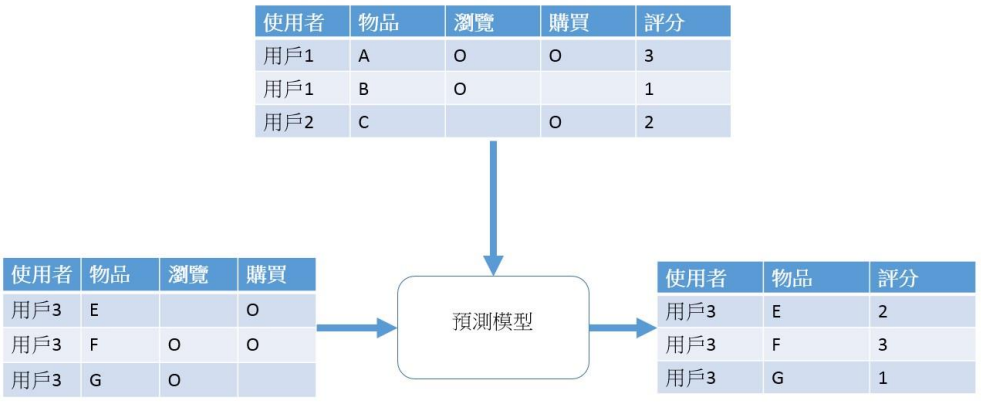

Figure 2.3: 根據用戶行為生成隱性評分

C. 資料整合: 顯性和隱性評分數據都被集成到一個矩陣中，即評分矩陣，如圖 2.4 所示。顯然，這個矩陣中仍然有很多缺失的元素，需要通過協同過濾來填充。

	A	B	C	D
U1	?	1	?	3
U2	3	2	2	?
U3	2	?	1	3
U4	4	2	2	?

Figure 2.4: 具有缺少的評分矩陣

2.1.3 使用者為基礎的協同過濾

使用者為基礎的協同過濾是以與使用者興趣相似的相鄰使用者來計算出使用者也可能喜歡的項目，當相似鄰居個數提高時，預測準確度越高。以下圖 2.5 為例，根據所有使用者的紀錄，當使用者 a 想找推薦的電影，而目前他看過電影 1 及電影 3，可以根據紀錄中的使用者 b 所看過的電影 1、3、4 的關係，我們可以表示使用者 a 及使用者 b 相似度較高，所以推薦給使用者 a 電影 4。

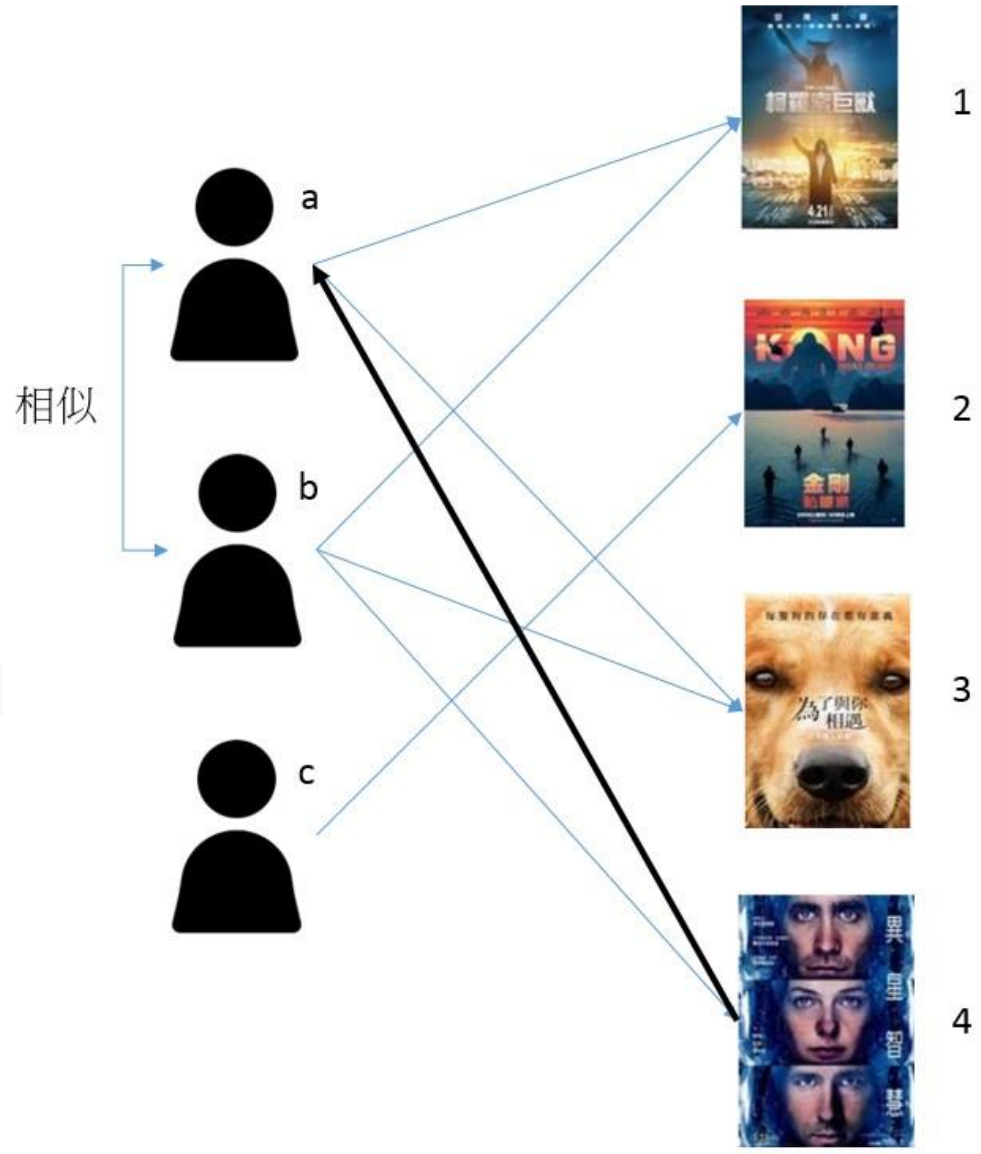

Figure 2.5: 使用者為基礎的推薦關係圖

2.1.4 以項目為基礎的協同過濾

以項目為基礎的協同過濾是有一個基本假設「能夠引起使用者興趣的項目，必定會與其之前評分高的項目相似」，再透過計算項目之間的相似程度來做為推薦的依據。以下圖 2.6 為例，根據所有使用者的紀錄知道喜歡電影 1 的使用者都

喜歡電影 3，所以推測出電影 1 和電影 3 的相似度較高，而使用者 c 喜歡電影 1，所以我們推薦電影 3 給使用者 c。

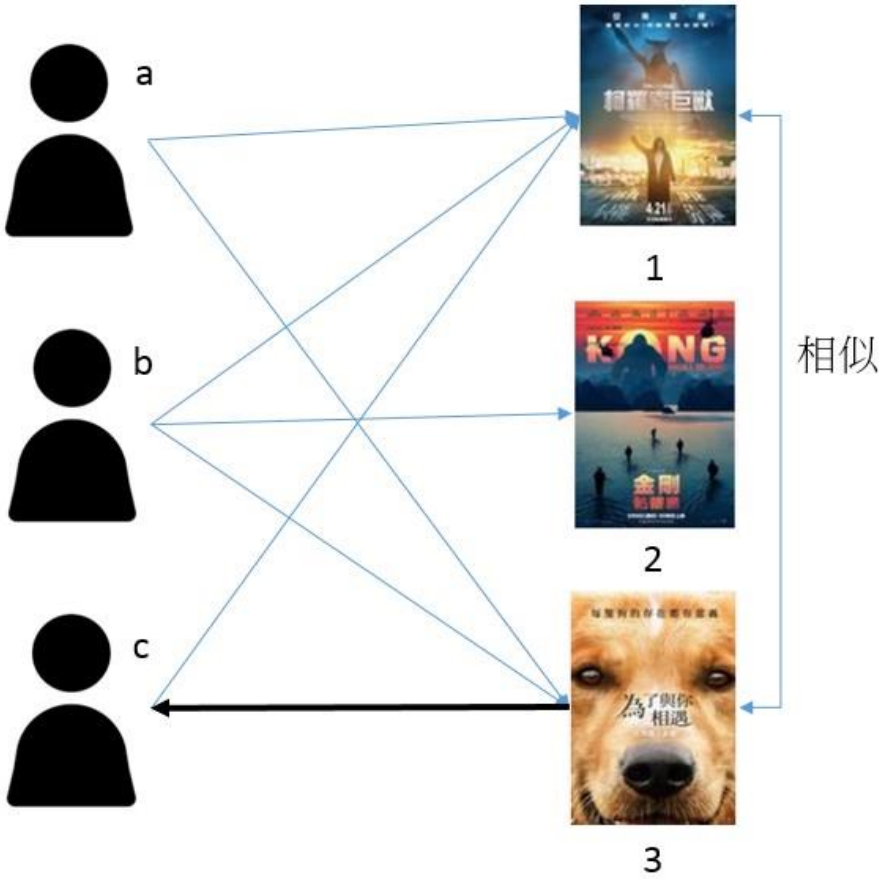

Figure 2.6: 項目為基礎的推薦關係圖

2.1.5 協同過濾程序

下圖 2.7 給出了協同過濾的五個一般程序，需要如下詳細分析

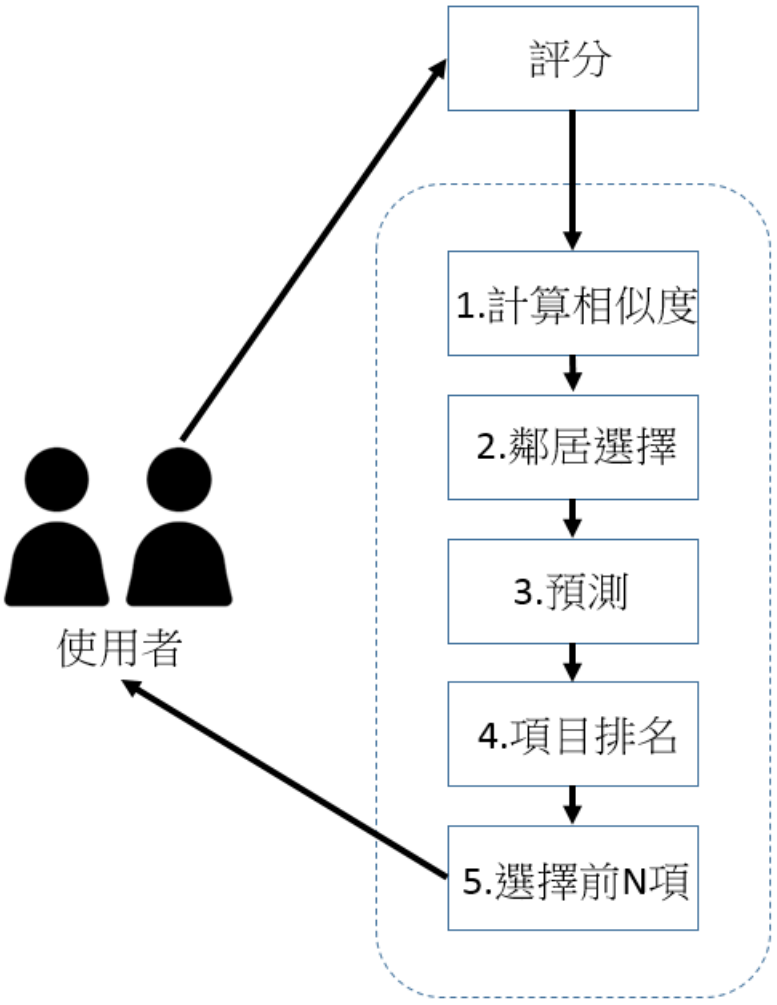

Figure 2.7: 協同過濾演算法程序

Step 1: 相似度計算: 用戶或項目之間的相似度計算是協同過濾中的關鍵步驟。協同過濾中廣泛使用了許多相似度，例如:Cosine 相似度、進階 Cosine 相似度、皮爾森相關係數。在下節會有詳細說明

Step 2: 鄰居選擇: 只要有不同的使用者加入鄰居，預測的精確度就會發生改 變。因此，應該通過某些方法仔細選擇活躍用戶的鄰居，傳統的 Top-N 算法選 擇 N 個最相似的鄰居進行預測。此外，不同國家或地區的人們更有可能有不同 的偏好。因此，為活躍用戶選擇鄰居時需要考慮用戶位置。由於行動網路的發 展，可以通過移動客戶端或 IP 位址獲取位置資訊，並將其發送到伺服器進行進 一步分析。通常，用戶可以根據其位置被分成幾個分區，並且同一分區內的用戶在鄰居選擇中具有優先級。

Step 3: 預測: 通過鄰居的相似度以及評分作為計算依據，使用方法會在下節詳細說明。

Step 4: 項目排名: 一旦獲得預測，推薦系統就需要根據其預測的評分對所有項目進行排名。為了提高建議的多樣性，具有較大預測值和較低受歡迎度的項目應該排名較高。

Step 5: 選擇前 N 個項目: 在對所有選項進行排名之後，它們的 Top-N 項被 提供給用戶，其中 N 是在推薦任務之前需要預設的參數。

2.1.6 皮爾森相關係數

皮爾森相關係數有兩種概念，一種是大小或強弱，是以該值的絕對值而言，其絕對值越大，代表兩者關聯性越高；其值越小，代表兩者關聯性越低。一種是方向符號，是指該系數是正值或負值，其係數為正值時，兩者關係為順向變化，一者變大，另一者也會跟著變大，一者變小，另一者也跟著變小，稱為「正相關」; 負值為逆向變化，一者變大，另一者隨著變小，一者變小，另一者隨著變大，稱為「負相關」; 而係數為零的話，一者變小，另一者可能變大或變小或沒有變化，稱之為「零相關」。[3]

2.1.7 正常恢復相似性度量

協同過濾演算法中的相似度有許多不同的演算法，例如：Jaccard similarites、Cosine similarities、Adjusted cosine similarities、Pearson correlation-based similarity 等。由以下公式可以知道 Jaccard Similarity 的核心概念為:

$$\mathrm{J(A,B)} = \frac{|A \cap B|}{|A \cup B|}$$，公式(2.1)

使用者 A 與使用者 B 共同評分過的項目個數除以使用者 A 或使用者 B 評分過的項目個數，其值落在 0 到 1 之間。

上述的相似度演算法中以皮爾森相關係數最為出名，其值會落在 1 到 -1 之間，若是以使用者為基礎的協同過濾，其公式為：

$$\mathrm{Sim(u,v)} = \frac{\sum_{i\in I}(r_{u,i}-\bar{r}_u)(r_{v,i}-\bar{r}_u)}{\sqrt{\sum_{i\in I}(r_{u,i}-\overline{r_u})^2}\sqrt{\sum_{i\in I}(r_{v,i}-\overline{r_v})^2}}$$，公式(2.2)

I 為使用者 u 和 v 之間有評分過的項目，$r_{u,i}$ 表示使用者 u 對於物品 i 的評分，$r_{v,i}$ 表示使用者 v 對於物品 i 的評分，$\bar{r}_u$ 和 $\bar{r}_v$ 代表使用者 u 所有評分的平均值及使用者 v 所有評分的平均值。若是以物品為基礎的協同過濾，其公式為：

$$\mathrm{Sim(i,j)} = \frac{\sum_{u\in U}(r_{u,i}-\bar{r}_i)(r_{u,j}-\bar{r}_j)}{\sqrt{\sum_{u\in U}(r_{u,i}-\bar{r}_i)^2}\sqrt{\sum_{u\in U}(r_{u,j}-\bar{r}_j)^2}}$$，公式(2.3)

U 為物品 i 和 j 之間有相同使用者評分的項目，$r_{u,i}$ 表示使用者 u 對於物品 i 的評分，$r_{u,j}$ 表示使用者 u 對於物品 j 的評分，$\bar{r}_i$ 和 $\bar{r}_j$ 代表物品 i 所有評分的平均值及物品 j 所有評分的平均值。

以上這兩個協同過濾演算法，使用相同的預測公式，使用者為基礎的協同過濾公式為：

$$\mathrm{Score}_{u,i} = \frac{\sum Rating_{v,i} * sim(u,v)}{\sum sim(u,v)}$$，公式(2.4)

公式的意義代表: 使用者 v 有的評分，而使用者 u 沒有評分過的所有項目乘上使用者 u、v 的相似度，除以使用者 u、v 相似度的總和。而以物品為基礎的協同過濾公式則為：

$$\mathrm{Score}_{u,i} = \frac{\sum Rating_{u,j} * sim(i,j)}{\sum sim(i,j)}$$，公式(2.5)

公式的意義代表: 物品 j 有被使用者 u 評分，而物品 i 沒有評分過的所有項目乘上物品 i、j 的相似度，除以物品 i、j 相似度的總和。

	i_1	i_2	i_3	i_4	i_5
u_1	1	2	3	4	5
u_2	2	2	3	4	
u_3	3	2		4	
u_4	1	1	1	1	2
u_5	5	5	5	5	6

Figure 2.13: 皮爾森相關係數的計算範例 (資料來源 [4])

但是，在特定的情況下，皮爾森相關係數的計算下會發生一些錯誤，以下舉 兩個例子，在圖 2.13 中計算使用者 u_1 和使用者 u_2 的相似度以及使用者 u_2 和 使用者 u_3 的相似度，會得到下面的結果：

$$\mathrm{Sim}(u_1, u_2) > \mathrm{Sim}(u_3, u_2)$$

但是實際上使用者 u_2 和 u_3 的相似度應該要比較高，因為使用者 u_1 的評分範

圍落在 1~5 分，而使用者 u_2 和 u_3 的評分範圍都是 2~4 分。第二個例子，計算使

用者 4 和使用者5之間的相似度為：

$$\mathrm{Sim}(u_4, u_5) = 0.9701$$

是相似度非常高的使用者，但是實際上這兩位使用者的相似度應該是非常低 的。在文章 [4] 中他們提出一種改善的方式: 使用正常恢復相似性度量 (Normal Recovery Similarity Measure)：

$$\mathrm{Sim(u,v)} = 1 - \frac{dist(\vec{u},\vec{v})}{dist_{max}}$$

$$= 1 - \frac{\sqrt{\sum_{i\in I}\left(nr_{u,i} - nr_{v,i}\right)^2}}{\sqrt{\sum_{k=1}^{|I|}(1-0)^2}}$$

$$= 1 - \frac{\sqrt{\sum_{i\in I}(\frac{r_{u,i}-r_{umin}}{r_{umax}-r_{umin}} - \frac{r_{v,i}-r_{vmin}}{r_{vmax}-r_{vmin}})^2}}{\sqrt{\sum_{k=1}^{|I|}1}}$$，公式(2.6)

再將此公式(2.6)簡化為:

$$\mathrm{Sim(u,v)} = 1 - \frac{\sqrt{\sum_{i\in I}(\frac{r_{u,i}-r_{umin}}{r_{umax}-r_{umin}} - \frac{r_{v,i}-r_{vmin}}{r_{vmax}-r_{vmin}})^2}}{\sqrt{|I|}}$$, 公式(2.7)

經由公式(2.7)的計算，其值會落在 0 到 1 之間，在這公式的計算下修復了上 面兩個錯誤的情況，

$$\mathrm{Sim}(u_1,u_2) < \mathrm{Sim}(u_3,u_2), Sim(u_4,u_5) = 0$$

使用者 u_1 和 u_2 的相似度小於使用者 u_2 和 u_3 的相似度，使用者 u_5 和 u_6 之間的相似度為0。其預測分數之算式為公式(2.8)正常恢復相似度預測公式：

$$\widehat{r_{u,\imath}} = r_{umin} + (r_{umax} - r_{umin})\frac{\sum_{u'\in U} Sim(u,u')\times nr_{u',i}}{\sum_{u'\in U} Sim(u,u')}$$，公式(2.8)

r_{umin} 為使用者 u 評價過的最低分數，r_{umax} 為使用者 u 評價的最高分數，Sim(u,$\bar{u}$) 為使用者 u 及使用者 $\bar{u}$ 的相似度。本文根據 Normal Recovery Similarity Measure 為公式來做為協同過濾演算法的基礎。

2.2 Apache Hadoop

Hadoop 是在 2004 年由 Google 發布的 MapReduce 架構，之後 Hadoop 被開發用於 Nutch 的開放式搜尋引擎，最後與 Nutch 分離，成為 Apache Foundation 自己的項目。到今天為止，Hadoop 是市面上最著名的 MapReduce 框架，更有許多公司圍繞 Hadoop 發展。[5]

2.2.1 MapReduce

Hadoop 是基於 Java 的 MapReduce 框架 (圖 2.20) [6]。

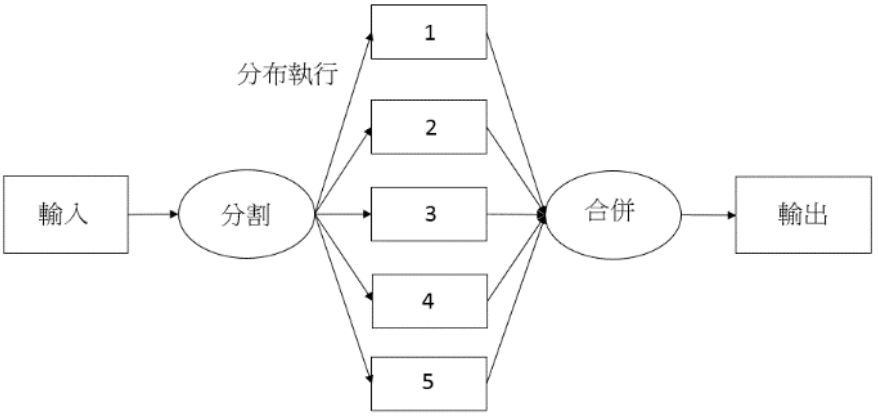

Figure 2.20: MapReduce 概念

在 Hadoop 的環境下，我們可以將要執行的任務做分割，將任務拆解成許多小任務，再分工給各節點處理，處理過後再將之輸出彙整，合併成為需要之結 果。

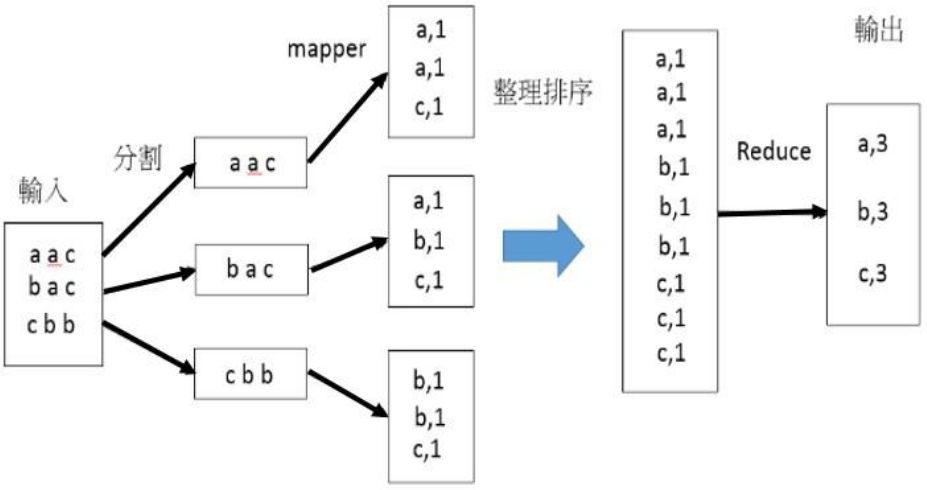

Figure 2.21: MapReduce 範例

圖 2.21 MapReduce 範例中，一開始輸入檔案為 aacbaccbb，經過分割後為 aac、bac、cbb，經由 Mapper 計算個數，整理排序後再經由 Reduce 後計算出個數，輸出其結果為 (a,3)、(b,3)、(c,3)。

2.2.2 Hadoop Distributed File System

Hadoop 的核心 Hadoop Distributed File System (HDFS) 是被設計用來支援大型檔案的應用程式，具有 Master/Slave 的結構，NameNode 是主節點，DataNode 是從節點。[5] NameNode 維護儲存在 HDFS 的一個文件數據，包含文件的 block 及它們在 DataNode 的位置訊息。NameNode 會將要處理的資料分割成為每個大小為 64MB 的 Block。NameNode 通常會維持 3 個檔案區塊的副本，分別儲存至 3 個不同的 DataNode，只要有某個 DataNode 的檔案有毀損，可以再由另外兩個 DataNode 上的副本做修復 [6]。Secondary NameNode 是一個在 Hadoop 中沒甚麼用處的部件，只為 NameNode 做一些內務管理的功能。DataNode 將 HDFS 中的文件實際儲存至本地的磁碟中。

圖 2.22 數字 1 表示客戶端的請求，2 表示獲得 Block 的清單以及其位置，3 表示直接讀取 DataNode 中的 block，4 表示關閉。HDFS 讀檔步驟：

A. 客戶端聯繫 NameNode，回傳 Block 列表及其位置 (包括副本位置)。

B. 客戶端直接連繫 DataNode 開始讀取 Block。如果 DataNode 失敗，客戶端將接收託管副本的 DataNode。

C. 當 Block 被讀取時會進行計算校正值，如果校正值與寫入時不同，則該 Block 會被回收。JobTracker 為 Hadoop 中主要的部件之一，負責管理作業的整體執行，他執行將單獨的 Mapper 和 Reduce 分配到各節點的功能，確認每個節點執行的狀況，甚至重新安排失敗的任務。TaskTracker 在單個節點上運行，負責啟動和管理單個 Map/Reduce 任務，與 JobTracker 做溝通。

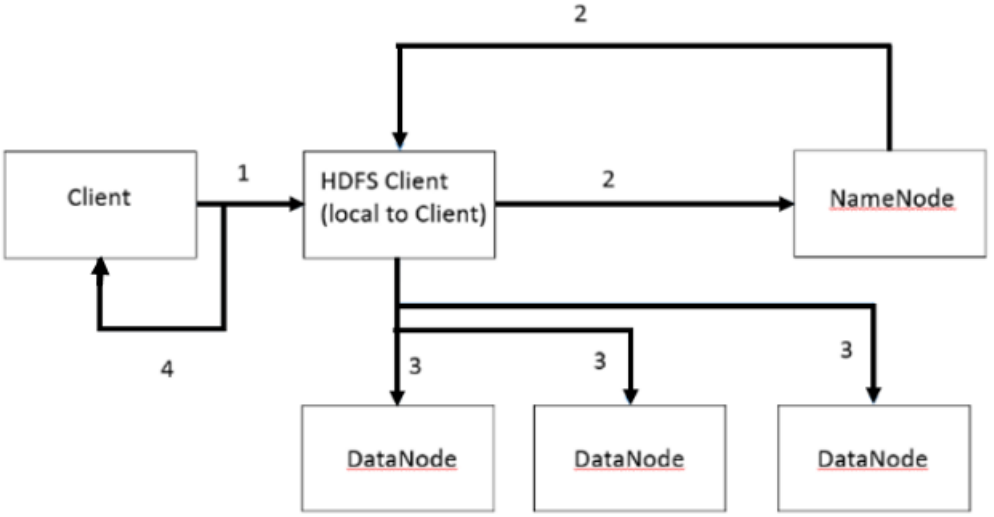

Figure 2.22: HDFS 讀檔過程

2.2.3 Apache Mahout

Apache Mahout 是 Apache Software Foundation 旗下的一個開放原始碼的專案，其專案的目的在於提供快速建構與高可擴展性的機器學習應用環境。機器學習是一個很廣大的領域，其內容包羅萬象，而 Mahout 的目標便是提供各種類型的經典演算法，以程式庫的方式呈現，並且具備高效及規模可擴充性的特質。Mahout 為了提供規模可擴充性，而基於 Apache Software Foundation 另一個名為 Hadoop 之專案的平台，以 MapReduce 的演算法時做了像群集 (clustering)、分類 (classification)，以及協同過濾 (collaborative filtering) 的核心。

除了運行在分散式多節點的 Hadoop 平台上也提供了單節點與其他版本上的應用，這使得開發者可以依據其對計算力及規模可擴充性的需要，來決定系統運行的方式，甚至依需要隨時改變配置方式。

第三章　研究方法與實驗環境

3.1 實驗環境

我們使用的語言為 Java，單機環境：作業系統-ubuntu 14.04 LTS 64bit、記憶體-11.4GB、CPU-Intel Pentium CPU G860。實驗中以四台主機作為硬體的雲端環境：Master：作業系統-ubuntu 14.04 LTS 64bit、記憶體-11.4GB、CPU-Intel Pentium CPU G860。Slave：作業系統: Windows 10 64bit、記憶體-8GB、CPU-Intel core i7-4790。虛擬機環境：作業系統-ubuntu 14.04 LTS 64bit、記憶體-1.9GB、CPU-Intel core i7-4790，並分為 3、6、9 個節點做測試。本次使用之資料從 IMDB 電影評分網站 (http://www.imdb.com/) 中取得，總共使用 115668 筆評分紀錄，在本實驗的資料中已去除評分項目不到 10 個的使用者，以及評分全部為相同的使用者，(詳見表 3.1)：

Table 3.1: 資料說明

使用者數量	電影數量	評分紀錄數量
100	47084	115668

因為協同過濾演算法在相似的相鄰個數 k 值提升時精確度會提高，所以在實作過程中，分別會以相似鄰居個數 k 為 1 到 10，來做測試。

3.2 基於使用者的協同過濾演算法

在上面提到的方程式作為以使用者為基礎的協同過濾演算法，本次實驗架構主要分為 3 個部分：(1) 計算所有使用者評分的最大值減最小值；(2) 計算所有使用者的相似度; (3)計算出預測的評分。[7]

由圖 3.1 中來表示實驗方法，最初輸入的資料為<i,u,$R_{i,u}$>，經 MapReduce 後計算出相似度，在計算出預測分數，主要結構由 Map-I 和 Readuce-I 組成 MapReduce-I、Map-II 和 Readuce-II 組成 MapReduce-II、Map-III 和 Readuce-III 組成 MapReduce-III、Map-IV 和 Readuce-IV 組成 MapReduce-IV，Map-I 和 Readuce-I 計算出每個使用者的最大值減去最小值，Map-II 和 Readuce-II 計算出相似度，Map-III 和 Readuce-III 紀錄了要在 Map-IV 和 Readuce-IV 中使用的相似度矩陣。

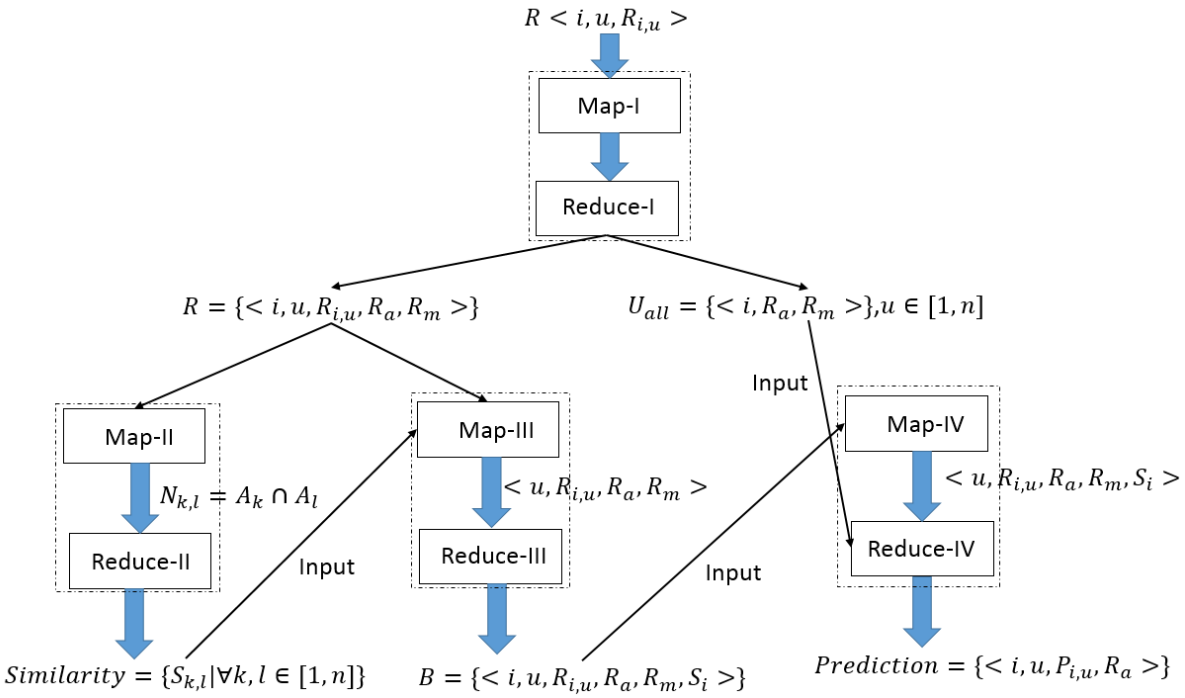

Figure 3.1:基於使用者的協同過濾演算法 MapReduce 流程圖

MapReduce-I：R<i,u,$R_{i,u}$> 中 i 代表項目，u 代表使用者，$R_{i,u}$ 代表使用者 u 對物品 i 的評價，經過計算後取得最大值減最小值 R_a，並且記錄最小值 R_m，此步驟有 2 個輸出，其一是 R=<i,u,$R_{i,u}$,R_a,R_m>，此輸出用於 MapReduce-II 和 MapReduce-III，再者 U_{all}=<i,R_a,R_m>,$u \in [1, n]$，將在 MapReduce-IV 中使用。

MapReduce-II：使用 R=<i,u,$R_{i,u}$,R_a,R_m> 做輸入，計算使用者 k 和 l 同

時有評價的項目 A_k 和 A_l 的總和$N_{k,l}=A_k \cap A_l$，計算出相似度 $S_{k,l} | \forall\ k, l \in [1, n]$，在做為 MapReduce-III 的輸入。

MapReduce-III：與MapReduce-II 使用相同輸入，在Reduce 中與相似度做整合。

MapReduce-IV：使用 MapReduceIII 的輸出作為輸入內容，將其與 MapReduce-I 的第二個輸出做預測值的計算。

在第二個實驗項目中，我們使用相同的資料，推薦一個預測分數最高項目，使用鄰居個數為 5，以三個不同的演算法做預測，Jaccard Similarity、皮爾森相似度、正常恢復相似性度量。

第四章　研究結果與分析

在單機實驗中原本要使用矩陣來儲存使用者資訊，但因為記憶體有限，所以改為當需要使用到該使用者資訊時，再從硬碟中讀取。

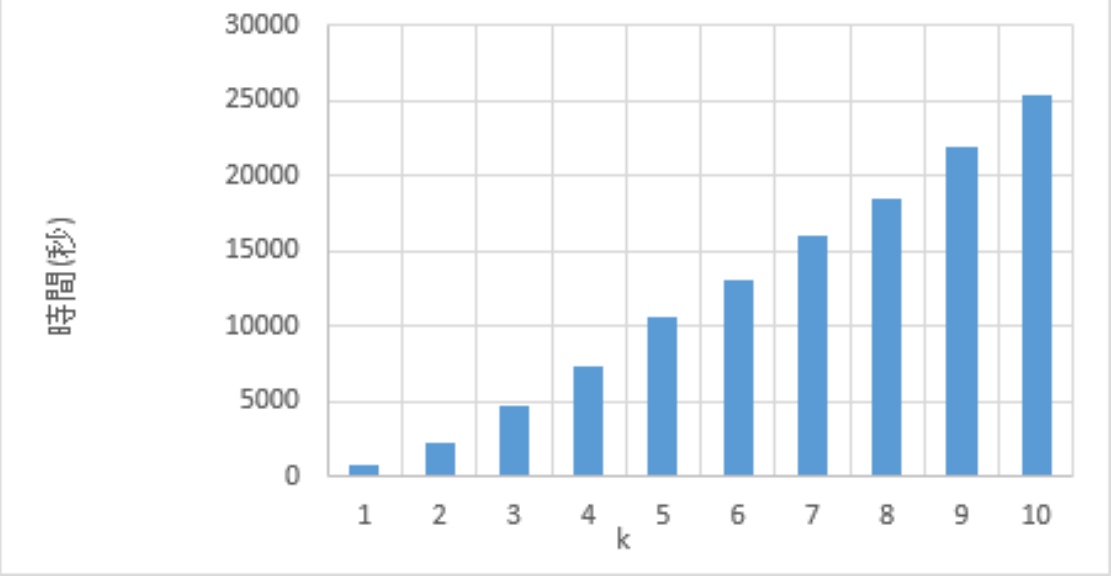

Figure 4.1: 單機執行時間

由圖 4.1 可以看出，其中 k 為鄰居個數，當 k 值提高時，執行時間會大幅提高，因為根據算式的 設定，當 k 值提高時，其時間會是指數成長。

由於 Hadoop 的硬體環境以 3 台主機組成，剛好對應 3、6、9 個節點。表 4.1 為單機與 3 個節點在調整 k 值的情形下 (k=1-10) 的效能比較。在 3 個節點時剛好是以 1 台主機執行，比起單機的執行還要多了傳輸的時間及配置時間，所以執行效能不佳。表 4.2 與表 4.3 分別為單機與 6 個節點與 9 個節點，在調整 k 值的情形下的效能比較。根據表 4.2 與表 4.3，在 6 個節點與 9 個節點時，比起單機的執行效能，有明顯改善。圖 4.2 為單機與 Hadoop 具 3、6、9 個計

算節點執行時間曲線比較圖。由圖 4.2 可觀察當雲端環境的硬體資源與節點數過低時 (橘色曲線)，不適合使用雲端執行，然而，當雲端硬體資源與節點數提高時(灰色與黃色曲線)，協同過濾演算法可有效地加速計算。

使用之加速比計算公式為:

$$\text{加速比(speedup)} = \frac{T_S}{T_P}$$

T_S代表單機執行時間，T_P代表 Hadoop 執行時間。

Table 4.1: 單機與 3 個節點的效能比較

k	單機	3 個節點	加速比
1	799	1183	0.675402
2	2247	3134	0.675402
3	4665	6436	0.724829
4	7397	11862	0.623588
5	10695	15672	0.682427
6	13042	21979	0.593385
7	16019	25278	0.633713
8	18527	32801	0.56483
9	21947	35128	0.624772
10	25450	36912	0.689478

Table 4.2: 單機與 6 個節點的效能比較

k	單機	6 個節點	加速比
1	799	534	1.496855
2	2247	1468	1.530654
3	4665	3154	1.479074
4	7397	5331	1.387545
5	10695	7079	1.510807
6	13042	8812	1.480027
7	16019	10970	1.460255
8	18527	12339	1.501499
9	21947	14853	1.477614
10	25450	17696	1.438178

在 6 個節點的情況下，能夠看出加速比都大於 1，說明在 6 個節點的時候已經比單機執行的速度要快上 0.5 倍左右。

Table 4.3: 單機與 9 個節點的效能比較

k	單機	9 個節點	加速比
1	799	327	2.443425
2	2247	899	2.499444
3	4665	1925	2.423377
4	7397	2856	2.589986
5	10695	4366	2.449611
6	13042	5357	2.434572
7	16019	6358	2.519503
8	18527	7193	2.575699
9	21947	9273	2.366764
10	25450	10796	2.357355

在 9 個節點的情況下，能夠看出加速比已經提高到了 2 倍以上，大約在 2.4 倍左右，最高加速比出現在鄰居個數 k 等於 4 的出現最高加速比 2.58 倍。

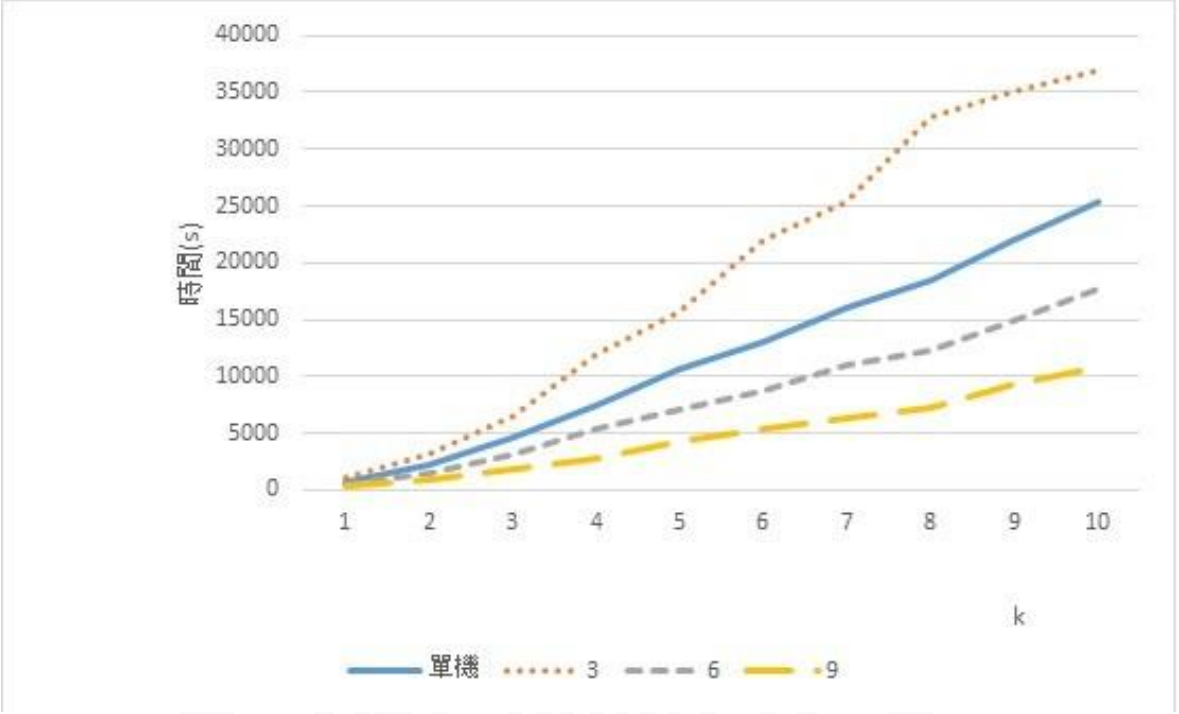

Figure 4.2: 單機與 Hadoop 具 3、6、9 節點執行時間比較

在第二個實驗中，我們使用了三個不同的演算法來計算結果預測，實驗出來的結果完全一致，我們推測造成次結果的原因有兩個，第一個是資料集，因為本次使用的資料來源為網站的評分，不具有強制性，所以矩陣較為稀疏(資料集的矩陣密度不達 50%)，使用者可能只會想評分自己喜歡的項目，造成相似度都是正相關，所以相似度的計算就會有相似的結果；第二個原因是我們只推薦了一個最高的項目，所以可能會有忽略其他可能的項目。

表 4.4 為三個不同演算法的部分推薦結果，從 100 個使用者中以30 個使用者來表示結果，表格內容為電影編號，每列代表不同的使用者，從表中可以看到每個使用者在三個不同演算法的推薦結果都是相同的。

Table 4.4: 三個演算法的部分推薦結果

使用者	Jaccard	Normal	Pearson
1	914863	914863	914863
2	39192	39192	39192
3	2802850	2802850	2802850
4	5229638	5229638	5229638
5	63929	63929	63929
6	39192	39192	39192
7	3675568	3675568	3675568
8	3845888	3845888	3845888
9	63929	63929	63929
10	914863	914863	914863
11	2802850	2802850	2802850
12	63929	63929	63929
13	63929	63929	63929
14	39192	39192	39192
15	5229638	5229638	5229638
16	39192	39192	39192
17	80678	80678	80678
18	1289401	1289401	1289401
19	4561950	4561950	4561950
20	816692	816692	816692
21	63929	63929	63929
22	734585	734585	734585
23	734585	734585	734585
24	3675568	3675568	3675568
25	120152	120152	120152
26	203119	203119	203119
27	2486678	2486678	2486678
28	63929	63929	63929
29	4574334	4574334	4574334
30	5834256	5834256	5834256
…	…	…	…

第五章　結論

雲端時代的來臨，資料量增加的速度非常快，在巨量資料的環境下，當需要找出方法來解決問題時，執行速度將是關鍵。我們在本文中使用了協同過濾演算法以及雲端，並將兩者做結合後，能有效提高速度比 2.58 倍。在實際資料量增加的情況中，單機的運算會需要較多的時間，且在儲存資料的容量有一定的限制，經過 Hadoop 平台上使用 MapReduce 將運算與資料分散給不同的主機做運算，可以節省許多時間以及資料過多的負擔，且 HDFS 確保資料的正確性，經過正常恢復相似度量的修正後，提高預測的準確度。在實驗的結果中可以發現，在鄰居個數 (K) 提高的情況下，執行的時間成正比的成長，在節點數 6 與 9 時大幅改善執行時間，使協同過濾演算法之效能提高，應付未來巨量資料，但還是存在一些不足之處，例如協同過濾演算法在矩陣分布較為稀疏的情況，或面臨新的物品或使用者的情況，都會使其難以預測。在未來會在找尋其他推薦演算法，以及改善方向，例如：機器學習。

References

[1] Vadim E. Kotov, “Big Data, Big Systems, Big Challenges: A Personal Experience,” Springer, Berlin, Heidelberg, 19 April, 2015.

[2] Tiejian Luo , Su Chen, Guandong Xu, Jia Zhou, ” Collaborative Filtering,” Springer New York, 28 June 2013.

[3] Jiguang Wang, ”Pearson Correlation Coefficient,” Springer New York, 2013.

[4] Huifeng Sun, Zibin Zheng, Junliang Chen, and Michael R. Lyu, “Personalized Web Service Recommendation via Normal Recovery Collaborative Filtering,” *IEEE Transactions on Services Computing,* VOL. 6, NO. 4, pp.10-12, 2013.

[5] Jason Venner, ” Pro Hadoop,” Apress, 2009.

[6] Sameer Wadkar, Madhu Siddalingaiah, “Pro Apache Hadoop,” Apress, 2014.

[7] Jing Jiang, Jie Lu, Guangquan Zhang, Guodong Long,” Scaling-up Item-based Collaborative Filtering Recommendation Algorithm based on Hadoop,” 2011 IEEE World Congress on Services

[8] Zhe Yang, Bing Wu, Kan Zheng, Xianbin Wang, and Lei Lei, ” A Survey of Collaborative Filtering-Based Recommender Systems for Mobile Internet Applications,” May 26, 2016

[9] askubuntu. https://askubuntu.com/

[10] Hadoop 群集安裝配置. http://www.powerxing.com/install-hadoop-cluster/

[11] M. Deshpande and G. Karypis, “Item-Based Top-N Recommendation,” ACM Trans. Information System,vol. 22, no. 1, pp. 143-177, 2004.

[12] CL Zheng , KR Hao , YS Ding , "A Collaborative Filtering Recommendation Algorithm Incorporated with Life Cycle ," Advanced Materials Research, 2013, 765-767(765-767):630-633.

[13] Q. Liu , et al. , "Enhancing Collaborative Filtering by User Interest Expansion via Personalized Ranking," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2012. 42(1): p. 218-233.

[14] Jonathan L. Herlocker , et al. ," Evaluating Collaborative Filtering Recommender Systems," ACM Transactions on Information Systems (TOIS), 2004. 22(1): p. 5-53.

[15] M.C Pham, et al. ,” A Clustering Approach for Collaborative Filtering Recommendation Using Social Network Analysis, "1. UCS, 2011. 17(4): p. 583-604

[16] H.-N. Kim, et al. , "Collaborative Filtering Based on Collaborative Tagging for Enhancing the Quality of Recommendation," Electronic Commerce Research and Applications, 2010. 9(1): p. 73-83.

[17] https://www.slideshare.net/waue/hadoop-map-reduce-3019713 (MapReduce 程式設計)

[18] Qing Wang,”” Design and Implementation of Recommender System Based on Hadoop,” Software Engineering and Service Science (ICSESS), 2016 7th IEEE International Conference

[19] Zhi-Dan Zhao, Ming-Sheng Shang, ”User-based Collaborative-Filtering Recommendation Alogorithms on Hadoop, ”Third International Conference on Knowledge Discovery and Data Mining, pp 478-481,2010.