

國立中正大學

資訊工程研究所碩士論文

協同過濾深度學習之推薦系統

Deep Learning Recommendation  
System With Collaborative Filtering

研究生：林哲維

指導教授：張榮貴博士

中華民國一零六年六月

# 誌謝

感謝就讀碩士班期間，老師用心指導與包容，這兩年老師的諄諄教誨學生銘記在心，從一開始對於碩士論文研究方向迷惘，到後來一次次地與老師討論，老師有如大海明燈指引我方向訂定題目，從老師身上除了讓我學習到學業上的知識更讓我學習到做事該有的基本態度。另外要感謝家人與女友在我就讀期間，全力支持著我讓我無後顧之憂，擁有持續研究下去的動力，最後還要感謝實驗室的學長、同學、學弟們，在研究期間遇到瓶頸協助我一起討論解決問題，讓我在研究之路更為順利。

林哲維 謹誌 2017 年 6 月 中正大學



# 協同過濾深度學習之推薦系統

國立中正大學資訊工程研究所

## 摘要

近年來隨著網路的蓬勃發展，網路上的遠距離互動已經不僅僅是使用者與使用者的對話，已經發展成由機器自動回應使用者的地步，也就是所謂的人工智慧，而近年來很火紅的深度學習演算法，在人工智慧的領域，不論是語音處理、電腦視覺與自然語言的處理等，領域都取得非常大的成就。相對來說，深度學習在推薦系統的領域上還是處於一個早期的探索階段。因此本論文提出利用深度學習的推薦系統，此種方式可以有效解決以往基於內容推薦的系統中常遇到的問題，像是冷啟動的問題等，而且有效的利用 Alternating Least Squares(ALS)將用戶(User)對商品(Item)的評分兩個矩陣，充分的將數據中大量的缺失項目補足且減少矩陣維度，再利用 Collaborative Filtering 協同過濾技術找出用戶有相似行為的群體訊息，並根據這些訊息給用戶推薦。本論文中推薦系統能有效的解決傳統推薦系統相對不足的部分，並利用深度學習的方式，更準確的推薦用戶可能喜歡的商品。

關鍵詞：推薦系統、深度學習、機器學習、協同過濾

# 目錄

誌謝.....	I
摘要.....	II
目錄.....	III
圖片目錄.....	IV
表格目錄.....	V
一、緒論.....	1
二、相關研究.....	4
2.1、矩陣分解 (MATRIX FACTORIZATION) .....	5
2.2、協同過濾 (COLLABORATIVE FILTER) .....	6
2.3、交替最小二乘法 (ALS) .....	7
三、相關開發工具.....	8
3.1、APACHE SPARK .....	9
3.2、APACHE SPARK WITH MLLIB .....	10
3.3、PYTHON .....	11
3.4、MONGODB .....	12
四、方法流程.....	14
4.1、HADOOP 環境架設.....	16
4.2、SPARK 環境架設.....	21
4.3、MONGODB COMPASS .....	23
4.4、訓練流程.....	24
4.4.1、MODEL 訓練流程 .....	26
4.4.2、商品預測流程.....	29
五、實驗結果與分析.....	31
5.1、評估實驗.....	35
5.2、叢集實驗.....	38
六、結論.....	40
七、參考文獻.....	41

# 圖片目錄

圖 1、矩陣分解示意圖.....	5
圖 2、ALTERNATING LEAST SQUARES 示意圖[24] .....	7
圖 3、APACHE SPARK ECOSYSTEM[24] .....	9
圖 4、方法流程圖.....	15
圖 5、實驗環境相關硬體規格.....	19
圖 6、利用 PUTTY 下達指令安裝 .....	19
圖 7、建立與 SLAVE 連線 .....	20
圖 8、確認 HADOOP 啟動 .....	20
圖 9、確認 APACHE SPARK 啟動 .....	22
圖 10、成功執行 APACHE SPARK .....	22
圖 11、MONGODB COMPASS 畫面 .....	23
圖 12、情境流程圖.....	25
圖 13、訓練流程.....	27
圖 14、訓練流程部分程式碼.....	28
圖 15、預測流程部分程式碼.....	29
圖 16、預測流程.....	30
圖 17、GOOGLE 提供的相關訓練數據.....	32
圖 18、GOOGLE 提供的建築物照片 .....	32
圖 19、從用戶收集來的評價.....	33
圖 20、實驗數據圖 1.....	33
圖 21、實驗數據圖 2.....	34
圖 22、實驗數據圖 3.....	34
圖 23、實驗數據圖 4.....	36
圖 24、CPU 數量影響比較 .....	39
圖 25、WORKER 數量影響比較.....	39

# 表格目錄

表一、推薦方法優缺點.....	3
表二、軟體套件用途說明表.....	12
表三、MONGODB 特色缺點介紹.....	13
表四、HADOOP 相關參數設定.....	16
表五、SPARK 相關參數設定.....	21
表六、ALS 相關參數設定介紹.....	26
表七、實驗硬體配備.....	33
表八、用戶評分介紹.....	36
表九、實驗數據介紹.....	37





# 一、緒論

近年來隨著網路的蓬勃發展，網路上的遠距離互動已經不僅僅是使用者與使用者的對話，已經發展成由機器自動回應使用者的地步，也就是所謂的人工智慧，而近年來很火紅的深度學習演算法，在人工智慧的領域，不論是語音處理、電腦視覺與自然語言的處理，一再突破了以往的瓶頸，也因此深度學習成為了許多應用上常使用的一種方式，所以我們利用深度學習方式與推薦系統做結合。在以往的推薦系統框架中，推薦方法是整個推薦系統中最核心、最關鍵的部分，很大程度上決定了推薦系統性能的優劣。目前，主要的推薦方法包括：基於內容推薦、協同過濾推薦等幾種方法，而推薦系統的運作方式則是先評估使用者與商品之間的相關性，藉由 Recommendation Component 產生用戶與商品一個相關性的 Score，再根據這個 Score 我們就可以把分數最高的推薦給用戶，其實這個本質上就是一個個性化的排序問題。

目前推薦系統已經應用在很多地方了，根據科技新報中的一篇文章是關於亞馬遜 Amazon 銷售平台的推薦系統[31]，文章的內容主要在討論 Amazon 的銷售平台使用了推薦系統而帶來的銷售額，雖然有人提出了質疑，他覺得推薦系統所影響銷售額沒有網路上提出的 35% 這麼多，不過經過微軟的研究員 Amit Sharma[32]所提出的論文中，分析了 Amazon 實際上因為推薦系統所引起的點擊次數，實際上只有 25%，雖然與 35% 有些微差距，不過對於一個這麼有名的電子商務平台來說，有了推薦系統，就實質的影響了整體銷售額的百分之 25，這個數字的确是相當的驚人，如果能夠針對目前的系統去做出優化，又或者更符合人性化的推薦，想必 35% 應該是蠻容易達成的目標。另外不只在電子商務平台上，就連現在很多人常使用的社交軟體，影在不知不覺中利用相關推薦系統，在你的動態時報中出現許多你可能會有興趣的廣告，因此推薦系統所附加的價值也已經越來越高，所以推薦系統實質上是有它研究的價值。

由於各種推薦方法都有優缺點，可以透過表一得知相關優點及缺點，所以在實際中，組合推薦（Hybrid Recommendation）經常被採用。研究和應用最多的是內容推薦和協同過濾推薦的組合，因此在我們的研究中我們透過深度學習的方式去學習使用者的多層次抽象特徵。程式主要架構是建立在 Apache Spark 之上，利用 Spark 的叢集運算框架來進行整體的運算。透過 Matrix Factorization 針對未來需要推薦的商品類型資料去建模矩陣，再將 MF 矩陣、rank、Iteration、Lambda 等資訊放入 Alternating least squares 去做訓練，找出實際值與預測值差最小的 Model 作為最後的 Model，以供後續作為查詢的資料，再藉由這個 Model 即可得知每個用戶與每個商品之間預測出來最佳的前 5 名，做為推薦給用戶商品。





表一、推薦方法優缺點

推薦方法	優點	缺點
基於內容推薦	推薦結果直觀，容易解釋；不需要領域知識	稀疏問題；新用戶問題；複雜屬性不好處理；要有足夠的數據建構分類器
協同過濾推薦	未知的興趣容易發現、不需要領域知識、隨著時間推移性提高；推薦個性化、自動化程度高；能處理複雜的非結構化對象	稀疏問題；可擴展性問題；新用戶問題；取決於歷史相關數據；系統剛開始推薦品質差
基於規則推薦	不需要領域知識	規則提取較難、耗費時間；產品的同義性問題；個性化程度較低；
基於效用推薦	無冷啟動以及稀疏問題；對於用戶偏好變化敏感；能考慮非產品特性	推薦式靜態的，靈活度低；屬行重疊問題；
基於知識推薦	能把用戶需求應設到產品上；能考慮到非產品屬性	資訊難獲得；推薦是靜態的；

## 二、相關研究

本論文使用矩陣分解 (Matrix Factorization) [1]、協同過濾 (Collaborative Filtering)、交替最小二乘法 (Alternating Least Squares) 用於用戶與商品之間的關聯性訓練及計算研究，矩陣分解方法在 Spark 的 MLlib 中能夠快速地用來解決 user-item 稀疏矩陣中空值的部分，協同過濾演算法中是將每個 user 對每個 item 的評鑑作為基礎運算，交替最小二乘法是基於每個用戶如何喜歡這個電影，以及用戶對電影的評價來做為推薦的基礎。

本節將分成三階段說明，一、探討矩陣分解演算法 (MF) 如何有效解決稀疏矩陣等問題；二、探討協同過濾演算法 (CF) 的預測方法；三、探討交替最小二乘法 (ALS) 如何將用戶對於未知的商品有效的填充評分作為給用戶商品推薦。



## 2.1、矩陣分解 (Matrix Factorization)

本論文中使用矩陣分解 (Matrix Factorization) [1] 的方式，先將收集來的用戶針對各項商品之間的評價，轉換成稀疏矩陣 (Sparse Matrix) [2] 的格式方便我們運算。可是當資料量非常大的時候這些數據可能就會有幾個挑戰，一、Defining similarity 當有些商品可能非常相似又該如何做評斷；二、Dimensionality 當商品數量以及用戶數量非常大時，該如何有效地計算；三、Millions of Users/Items Sparse 並不是每個用戶都看過所有的影片，因此會有一個巨大的 Sparse Matrix。

因此本篇論文中應用了矩陣分解的方式將稀疏矩陣降低維度，減少許多缺少值的部分項目，在 MH1200[3] 有詳細的介紹到矩陣的推倒過程，透過矩陣分解後即可用兩個維度較小的矩陣相乘獲得原本的矩陣，其矩陣分解如下圖 1 所示。MF 模型一旦建立完成後的好處是，可以容易的 Predict，而且也有不錯的性能，但是當有巨大的用戶 (users) 以及商品 (items) 時，MF 模型中的兩個矩陣將會變得非常有挑戰性。

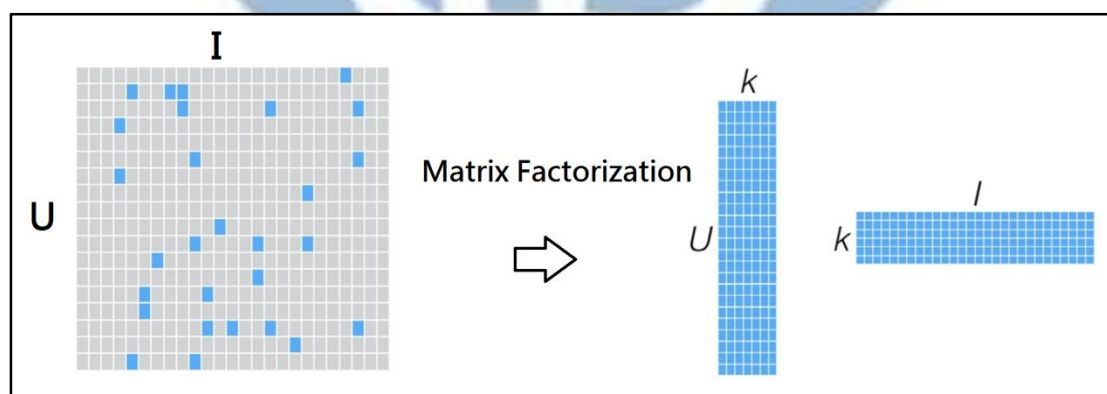


圖 1、矩陣分解示意圖

## 2.2、協同過濾( Collaborative Filter )

本論文在 Recommendation Model[4]中所使用的是協同過濾( Collaborative Filtering ) [5]的方法，在推薦系統產生推薦列表的方式通常有兩種：協同過濾以及基於內容推薦[6]的方式兩種，協同過濾方法根據用戶歷史行為（例如其購買的、選擇的、評價過的物品等）結合其他用戶的相似決策建立模型，這種模型可用於預測用戶對哪些物品可能感興趣，或用戶對物品的感興趣程度。在英國網路電台音樂社群 Last.fm[7]他們建立通過觀察用戶日常收聽的樂隊或歌手，並與其他用戶的行為進行比對，建立一個「電台」，以此推薦歌曲。Last.fm 會播放不在用戶曲庫中，但其他相似用戶經常會播放的其它音樂。鑑於這種方式利用用戶行為，因此可以認為它是協同過濾技術的一種應用範例。

論文中利用協同過濾方法，將每個 Users 對於每個 Items 的評鑑作為基礎運算，將你可能會喜歡的商品推薦給你。並且根據其他用戶的偏好來猜測你可以會喜歡怎樣的產品。每一種系統都有其優點與弱點。在上面的例子中，為了提供精準推薦，Last.fm 需要大量用戶信息。這是一個冷啟動問題，在協同過濾系統中是常見的問題[8][9][10]。

## 2.3、交替最小二乘法（ALS）

ALS 是 alternating least squares[11]的縮寫，意為交替最小二乘法，在前面有提到 Matrix Factorization 方法是跟 ALS 類似的，但是當有巨大的用戶（users）以及商品（items）時，Matrix Factorization 模型中的兩個矩陣將會變得非常有挑戰性。則 ALS 演算法是用來解決 Matrix Factorization 模型問題的一個優化技術，被證明高效、高性能並且能有效地並行化，目前在 MLlib 中 ALS 是唯一一個推薦模組，Apache Spark[12]官網上面有詳細的描述。

ALS 演算法中是透過一個 Factors 的方式來將 Matrix Factorization 所做的事情加以簡化，而這個 Factors 是指將用戶（users）所對應的商品（items）做縮減的動作，可能是依照類型的偏好，用戶（users）的選擇，把幾個商品（items）合成一個 Factors，把同類型的商品整合起來屬於一個 Factors，然後用這樣的方式來簡化整個計算的過程。圖 2 為 ALS 切割示意圖。

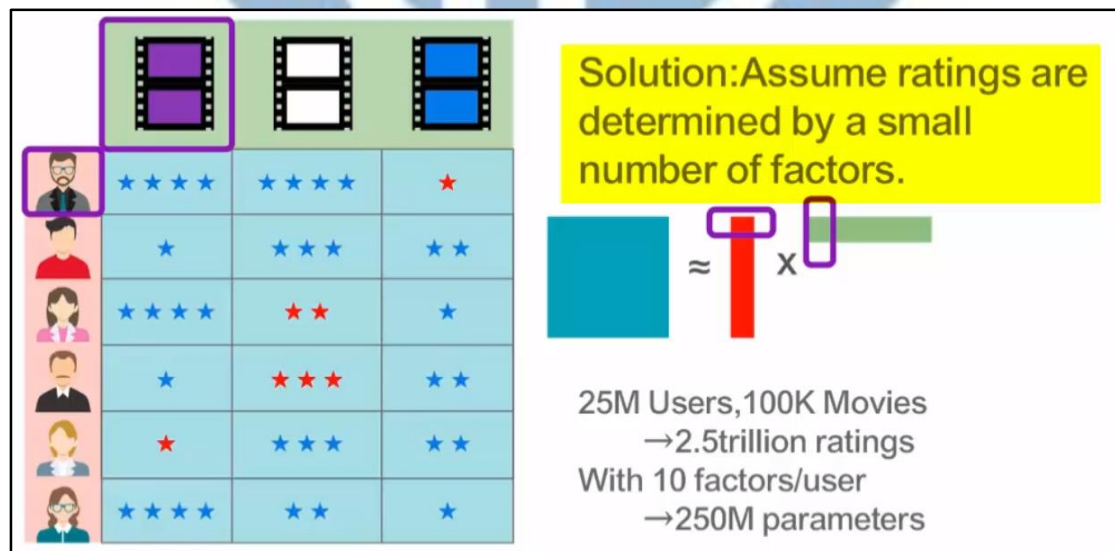


圖 2、Alternating Least Squares 示意圖[24]

### 三、相關開發工具

本論文提出之協同過濾深度學習推薦系統使用 Python 程式語言開發，相關深度學習演算法亦使用該語法撰寫，主要的運行框架是建構在 Apache Spark 上，透過 MLlib 中撰寫矩陣分解、交替最小二乘法等演算法，並匯入 MongoDB 建立相關推薦 Model 作為推薦的依據。因此本節第一節將說明 Apache Spark 相關資料，第二、第三、第四節則分別說明 Spark 中的 MLlib、Python、MongoDB 相關資料。相關開發套件軟體用途說明請參考表一。





## 3.1、Apache Spark

Apache Spark[13]是一個開源叢集運算框架，最初是由加州大學柏克萊分校 AMPLab 所開發。相對於 Hadoop 的 MapReduce 會在執行完工作後將中介資料存放到磁碟中[14]，Spark 使用了記憶體內運算技術，能在資料尚未寫入硬碟時即在記憶體內分析運算。Spark 在記憶體內執行程式的運算速度能做到比 Hadoop MapReduce 的運算速度快上 100 倍，即便是執行程式於硬碟時，Spark 也能快上 10 倍速度。Spark 允許用戶將資料加載至叢集記憶體，並多次對其進行查詢，非常適合用於機器學習演算法[15]。

另外 Spark 提供許多工具供我們使用，像是彈性分散式資料集（RDDs）、Spark SQL、Spark Streaming、MLlib、Graph 這幾種工具，在我們的論文中提到的則是 MLlib 這個工具，下一節將詳細介紹。此外下圖 3 說明整個 Apache Spark Ecosystem 包含哪些工具，以及支援哪些語法。

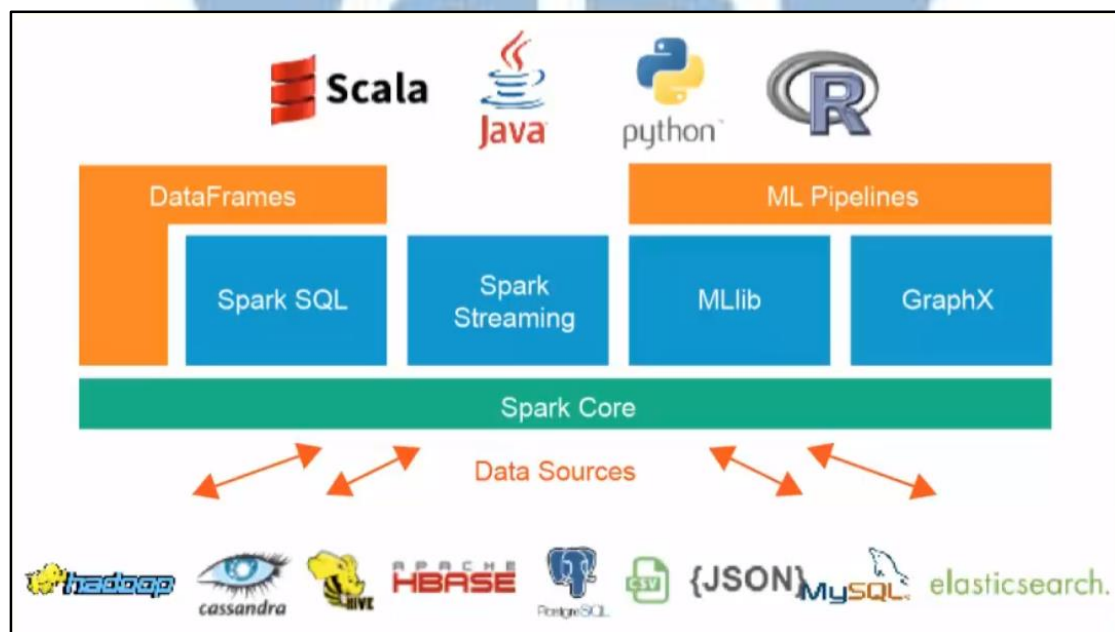


圖 3、Apache Spark Ecosystem[24]

## 3.2、Apache Spark With MLlib

MLlib 是 Spark 上分散式機器學習框架。Spark 分散式記憶體式的架構比 Hadoop 磁碟式的 Apache Mahout 快上 10 倍，擴充性甚至比 Vowpal Wabbit 要好 [16]。MLlib 可使用許多常見的機器學習和統計演算法，簡化大規模機器學習時間，其中包括：

- 一、匯總統計、相關性、分層抽樣、假設檢定、隨機資料生成
- 二、分類與回歸：支援向量機、回歸、線性回歸、決策樹、樸素貝葉斯
- 三、協同過濾：ALS
- 四、分群：k-平均演算法
- 五、維度縮減：奇異值分解（SVD），主成分分析（PCA）
- 六、特徵提取和轉換：TF-IDF、Word2Vec、StandardScaler
- 七、最佳化：隨機梯度下降法（SGD）、L-BFGS

論文中 Apache Spark MLlib 主要使用的工具是協同過濾的 Alternating Least Squares(ALS)。

## 3.3、Python

Python[17] 是一種物件導向、直譯式的電腦程式語言，支援命令式程式設計、物件導向程式設計、函數式編程、面向側面的程式設計、泛型編程多種編程範式，其創始人為吉多·范羅蘇姆（Guido van Rossum），Python 2.0 於 2000 年 10 月 16 日發布，增加了實現完整的垃圾回收，並且支援 Unicode。同時，整個開發過程更加透明，社群對開發進度的影響逐漸擴大。Python 3.0 於 2008 年 12 月 3 日發布，此版不完全相容之前的 Python 原始碼。不過，很多新特性後來也被移植到舊的 Python 2.6/2.7 版本。

雖然 Python[17] 可能被粗略地分類為「腳本語言」（script language），但實際上一些大規模軟體開發計畫例如 Zope、Mnet 及 BitTorrent，Google 也廣泛地使用它。Python 的支持者較喜歡稱它為一種高階動態程式語言，原因是「腳本語言」泛指僅作簡單程式設計任務的語言，如 shell script、VBScript 等只能處理簡單任務的程式語言，並不能與之相提並論。

Python 本身被設計為可擴充的。並非所有的特性和功能都整合到語言核心。Python 提供了豐富的 API 和工具，以便程式設計師能夠輕鬆地使用 C、C++、Cython 來編寫擴充模組。Python 編譯器本身也可以被整合到其它需要腳本語言的程式內。因此，有很多人把 Python 作為一種「膠水語言」（glue language）使用。使用 Python 將其他語言編寫的程式進行整合和封裝。在 Google 內部的很多專案，例如 Google App Engine，2004 年，Python 已在 Google 內部使用。

## 3.4、MongoDB

MongoDB[18]是一種文件導向資料庫管理系統，由 C++ 撰寫而成，以此來解決應用程式開發社群中的大量現實問題。2007 年 10 月，MongoDB 由 10gen 開發出來的 NoSQL 資料庫，而 NoSQL 是 Not Only SQL 的縮寫。MongoDB 是用來處理大數據級的資料庫，而他是以文本的方式儲存的資料庫。

MongoDB 的資料體結構是以 Key,Value 組合的，儲存的方式與 Json 格式完全相同，另外多了很多的靈活性，也就每一筆文件的是欄位的型態是不一定的，欄位的存在性也是不一定的，表二介紹 MongoDB 的特色及缺點。

**表二、軟體套件用途說明表**

開發所需軟體套件名稱	軟體套件用途說明
Python-2.7.10( Linux 版)	用於撰寫深度學習程式、資料的前處理、資料的轉換。
MongoDB-3.4	用於資料存取紀錄、包含預測推薦的資料。
MongoDB Compass	用於新增刪除訓練相關資料，UI 開發介面。
Jupyter (Ipython Notebook)	用於撰寫 Python 語言，UI 開發介面。
Postman chrome 擴充元件	用於檢視是否能成功從網站取得 get 或 post
Hadoop-2.7.1	用於實現 HDFS 的 MapReduce in Spark
Apache Spark-2.0.2	用於處理整個系統的運算架構
Spark MLlib	用於撰寫深度學習資料處理相關函式
Oracle Java	在安裝 Hadoop 需安裝 Java 套件
Putty	用於對主機下達執行命令

**表三、MongoDB 特色缺點介紹**

特色	缺點
MongoDB 可以處理資料庫為 T 級量的資料庫，也就是處理大數據的資料庫。	Console 畫面操作對有些人來說是不太方便的，雖然最近也已經有視窗化的工具。
分散式的資料庫模式，可以把眾多資料庫串連後處理大數據的資料。	對不熟悉 JavaScript 開發的人員來說，學習上的瓶頸較大，因為 Monogo 是基本 Javascript & JSON 的資料庫。



## 四、方法流程

本論文的方法流程主要分成幾個大點：首先建置 Hadoop 的 HDFS 環境以及 Apache Spark 叢集運算框架，實驗中我們利用分散式的叢集運算框架來進行複雜的推薦系統 Deep Learning 運算。

利用網路上公開的使用者資料來作為本實驗的訓練資料[25]，將資料利用 MongoDB Compass 套件將訓練資料丟入資料庫。用 Python 以及 Alternating Least Squares 撰寫出 Collaborative Filter 方法架構，將資料建構成 Sparse Matrix 再將資料做矩陣分解，分解後的矩陣進行 ALS.train 訓練，將原本的 Sparse Matrix 空缺資料做出相對應的預測，不過因為 ALS.train 需要針對模型的參數模型去做設定，不一樣的參數模型將導致不一樣的預測結果。

因此為了找出最好的一組結果，我們設計出 find\_model.py 去針對不一樣的參數模型設定，找尋出最好的預測結果模型。至於如何判斷最合適的參數模型，我們這邊針對參數去設定區間，每一組參數訓練完畢會產生一組預測結果，針對預測結果與原本既有的資料去做比對找出差異性最小的一組參數模型，作為此資料最合適的參數模型，並將其預測的結果回傳至 MongoDB 中以作為未來預測的結果。

有了最佳參數模型預測出來的結果，將沒有評分的使用者，列出 MongoDB 中預測推薦的前五名 item 作為推薦。圖 4 介紹了整個系統的流程，流程前面是透過穿戴型裝置來進行資料的收集，針對制定好的 Schema 將資料丟入資料庫，透過 find\_model.py 進入資料庫抓取資料利用 Apache Spark 做 Deep Learning，最後將訓練完成的資料丟回給資料庫，最後 predict.py 將沒有評分的用戶列出 predict 的前 5 個最高分數給用戶作為參考。



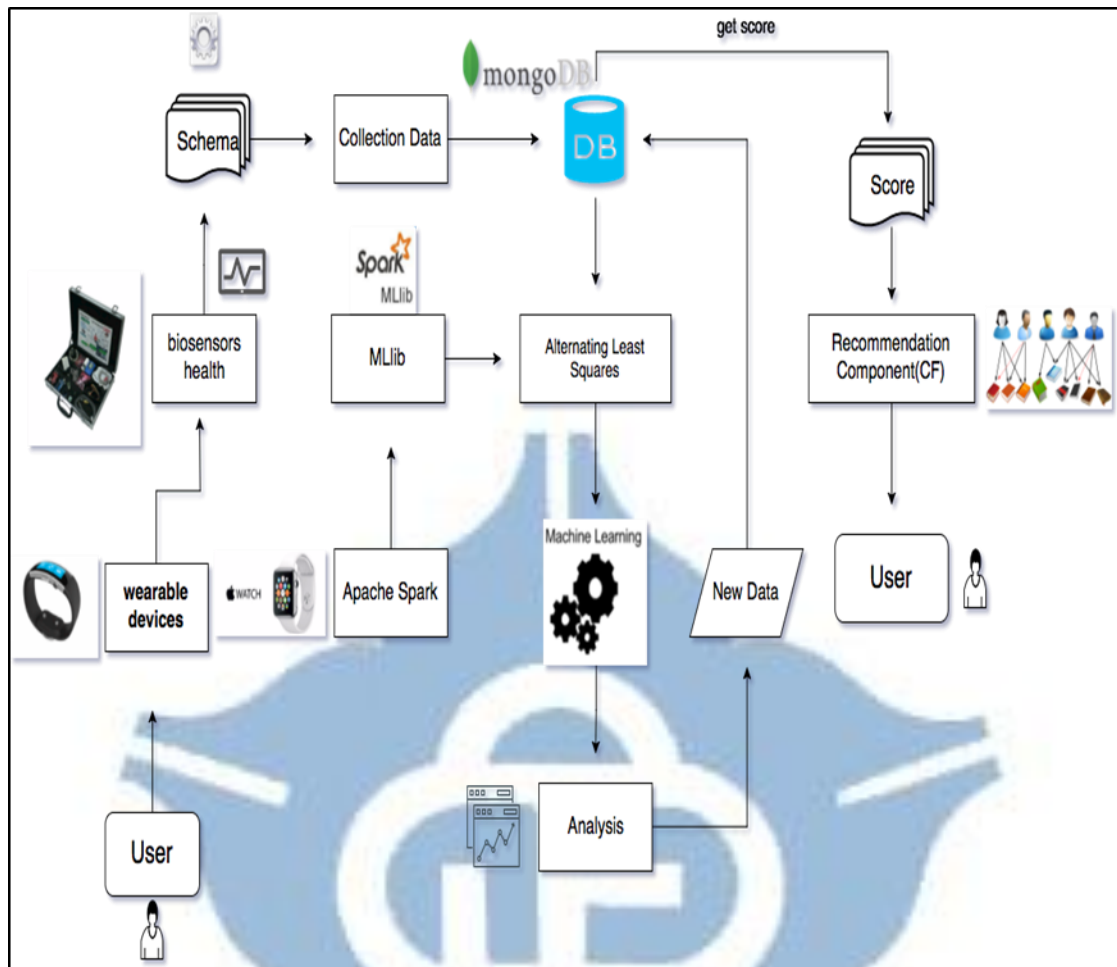


圖 4、方法流程圖

## 4.1、Hadoop 環境架設

首先需建構 Hadoop Server 環境[19]，實驗中 Master 所使用的環境是 Ubuntu Server 16.04 LTS 64 位元作為主要的 Server，另外配置兩台 Slave 作為分散式運算的 Worker，所使用的環境是 Ubuntu 16.04 LTS 64 位元，名稱分別是 Slave01、Slave02，詳細硬體規格請參考圖 5。透過 Putty 介面[20]下載 Hadoop 相關套件如圖 6，並修改每一台主機的 Hostname 以及設定 SSH 連線[21]如圖 6，確認彼此之間是否可以互相連通，並設定 SSH 無密碼登入[22] 如圖 7。相關 Hadoop 細節參數設定請參考[19]以及表三，並將設定完的資料打包傳送給所有 Slave 端，啟動 Hadoop 並且輸入 jps 確認是否有正確啟動即完成安裝步驟如圖 8，正常啟動資訊會包含 SecondaryNameNode、ResourceManager、NameNode 如圖 8。

表四、Hadoop 相關參數設定

Hadoop 路徑	新增相關參數	用途說明
~/.bashrc	export JAVA_HOME= /usr/lib/jvm/java-8-oracle	用於環境變數
/etc/network/interfaces	address 140.123.xxx.xx netmask 255.255.255.0 network 140.123.xxx.0	設定主機網路連線
/etc/hostname	Master Slave01 Slave02	設定主機名稱
/etc/hosts	140.123.xxx.xx master 140.123.xxx.xx slave01 140.123.xxx.xx slave02	ssh 連線設定
\$HADOOP_HOME/ slaves	slave01 slave02	Hadoop 連線設定
\$HADOOP_HOME/ core-site.xml	<configuration> <property>	Hadoop 連線設定

	<pre> &lt;name&gt;fs.defaultFS&lt;/name&gt; &lt;value&gt;hdfs://master:9000&lt;/value&gt; &lt;/property&gt; &lt;property&gt; &lt;name&gt;hadoop.tmp.dir&lt;/name&gt; &lt;value&gt;file:/usr/local/hadoop/tmp &lt;/value&gt; &lt;description&gt; Abase for other temporary directories.&lt;/description&gt; &lt;/property&gt; &lt;/configuration&gt; </pre>	
\$HADOOP_HOME/ hdfs-site.xml	<pre> &lt;configuration&gt; &lt;property&gt; &lt;name&gt; dfs.namenode.secondary.http-address &lt;/name&gt; &lt;value&gt;master:50090&lt;/value&gt; &lt;/property&gt; &lt;property&gt; &lt;name&gt;dfs.namenode.name.dir &lt;/name&gt;&lt;value&gt; file:/usr/local/hadoop/tmp/dfs/name &lt;/value&gt; &lt;/property&gt; &lt;property&gt; &lt;name&gt;dfs.datanode.data.dir &lt;/name&gt; &lt;value&gt; file:/usr/local/hadoop/tmp/dfs/data &lt;/value&gt; &lt;/property&gt; &lt;property&gt; &lt;name&gt;dfs.replication&lt;/name&gt; &lt;value&gt;2&lt;/value&gt; &lt;/property&gt; &lt;/configuration&gt; </pre>	Hadoop 連線設定 注意 value 值代表 slave 數量
\$HADOOP_HOME/	<configuration>	Hadoop 連線設定

mapred-site.xml	<pre> &lt;property&gt; &lt;name&gt;mapreduce.framework.name &lt;/name&gt; &lt;value&gt;yarn&lt;/value&gt; &lt;/property&gt; &lt;/configuration&gt; </pre>	
\$HADOOP_HOME/ yarn-site.xml	<pre> &lt;configuration&gt; &lt;property&gt; &lt;name&gt; yarn.resourcemanager.hostname &lt;/name&gt; &lt;value&gt;master&lt;/value&gt; &lt;/property&gt; &lt;property&gt; &lt;name&gt; yarn.nodemanager.aux-services &lt;/name&gt; &lt;value&gt;mapreduce_shuffle&lt;/value&gt; &lt;/property&gt; &lt;/configuration&gt; </pre>	Hadoop 連線設定


<div> <div>  <div> <div>Spark</div> <div>2.0.2</div> </div> </div> <div> <div>Spark Master at spark://master:7077</div> </div> </div>				
<div> <div>URL: spark://master:7077</div> <div>REST URL: spark://master:6066 (cluster mode)</div> <div>Alive Workers: 2</div> <div>Cores in use: 12 Total, 0 Used</div> <div>Memory in use: 9.5 GB Total, 0.0 B Used</div> <div>Applications: 0 Running, 68 Completed</div> <div>Drivers: 0 Running, 0 Completed</div> <div>Status: ALIVE</div> </div>				
Workers				
Worker Id	Address	State	Cores	Memory
worker-20170506153823-140.123.97.223-37621	140.123.97.223:37621	ALIVE	8 (0 Used)	6.7 GB (0.0 B Used)
worker-20170506153825-140.123.97.222-39931	140.123.97.222:39931	ALIVE	4 (0 Used)	2.8 GB (0.0 B Used)

圖 5、實驗環境相關硬體規格

```
jhewei@master: ~ [237x64]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

127.0.0.1      localhost
140.123.97.34  master
140.123.97.222 slave01
140.123.97.223 slave02
# The following lines are desirable for IPv6 capable hosts
::1          localhost ip6-localhost ip6-loopback
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
```

圖 6、利用 Putty 下達指令安裝

```
jhewei@master: ~ [237x64]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

jhewei@master:~$ ssh slave01
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-49-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

65 packages can be updated.
0 updates are security updates.

Last login: Tue May 23 13:23:14 2017 from 140.123.97.34
jhewei@slave01:~$ exit
logout
Connection to slave01 closed.
jhewei@master:~$ ssh slave02
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-49-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

65 packages can be updated.
0 updates are security updates.

Last login: Sun May  7 00:04:59 2017 from 140.123.97.200
jhewei@slave02:~$ exit
logout
Connection to slave02 closed.
jhewei@master:~$
```

slave01

slave02

圖 7、建立與 slave 連線

```
jhewei@master: ~ [237x64]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

jhewei@master:~$ ssh slave01 jps
23281 NodeManager
7793 Jps
23186 DataNode
2440 Worker

jhewei@master:~$ ssh slave02 jps
13176 NodeManager
10521 Jps
13037 DataNode
4238 Worker

jhewei@master:~$ jps
64006 Jps
48458 Master
3181 SecondaryNameNode
3358 ResourceManager
jhewei@master:~$
```

slave01 DataNode

slave02 DataNode

master Master

圖 8、確認 Hadoop 啟動



## 4.2、Spark 環境架設

確認 Hadoop 安裝完成後，安裝 Apache Spark[13]，透過 wget 方式下載相對應的 Hadoop 的 Spark 版本並解壓縮，本實驗所使用的 Apache Spark 版本是 2.0.2 for Hadoop2.6 另外環境參數請參考表三的相關設定，安裝過程請參考[23]。安裝完成利用 Start-All.sh 將 Apache Spark 啟動，透過 jps 指令確定狀態是否有執行成功如圖 9。確認每個 Slave 是否都有 Worker 節點以及 master 底下有 Mater 節點。完成以上設定即可進入 Spark Shell 進行操作如圖 10，以上完成本實驗相關設定。

表五、Spark 相關參數設定

Spark 路徑	新增相關參數	用途說明
~/profile	export SPARK_VERSION=2.0.2 export SPARK_HOME=/usr/local/spark/\$SPARK_VERSION export PATH=\$PATH:\$SPARK_HOME/bin	用於環境變數
/etc/hosts	140.123.97.xx master 140.123.97.xxx slave01 140.123.97.xxx slave02	可快速進行 ssh 動作
\$SPARK_HOME/conf	slave01 slave02	設定 slave 有哪幾台
\$SPARK_HOME/conf/ log4j.properties	log4j.rootCategory=INFO, console => log4j.rootCategory=WARN, console	減少啟動出現的 log 訊息
\$SPARK_HOME/conf/ spark-deafults.conf	Spark.master spark://master:7077 Spark.cores.max 5 Spark.executor.memory 2g	設定 excutor 硬體分配
\$SPARK_HOME/conf/ spark.env.sh	export HADOOP_CONF_DIR=\$HADOOP/tetc/hadoop	設定 hdfs 檔案位置



## 4.3、MongoDB Compass

本實驗是利用 MongoDB Compass 套件軟體將從客戶獲取的資料傳送至 MongoDB 中，目前使用的是 accommodations.csv[26]以及 ratings.csv[27]兩個資料群作為實驗的資料集，資料來源為網路上所提供，這兩份資料主要是客戶對於建築物評分表以及相關意見。下圖 11 中為建立資料執行過程，詳細方法請參考 [28]。

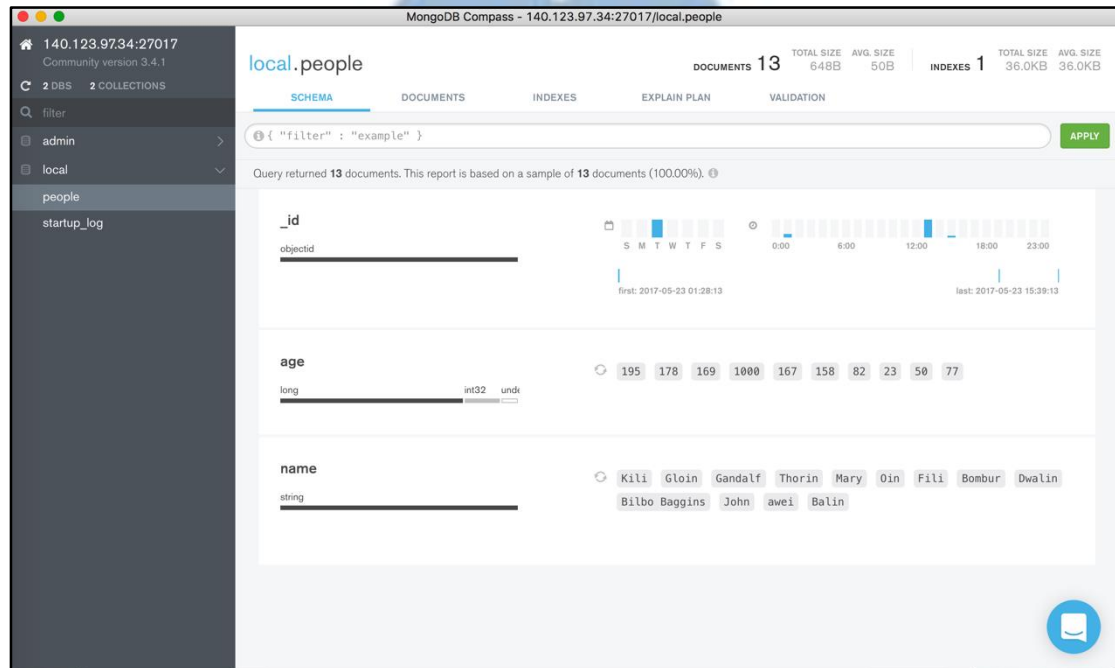


圖 11、MongoDB Compass 畫面

## 4.4、訓練流程

本論文提出的實驗架構如圖 12，實驗中利 Google 所提供的相關資訊作為本實驗的訓練資料及，資料包含 Accommodation、Rating 兩筆資料，資料內容分別是建築物 ID、簡述、地點以及建築物類型等資訊，另一筆資料內容是包含用戶對於建築物的評分表，包含用戶 ID、建築物 ID 以及評分分數作為我們訓練的資料集。實驗中分成兩個大項第一步驟是訓練流程，透過 Collaborative Filter 的方式將資料轉換，利用 Alternating Least Squares 找出最小交替二乘法來表示預測的矩陣，並且迭代的方式找出實際預測分數與預測矩陣的最小均方根誤差，並且記錄相關訓練參數傳回資料庫做為第二步驟相關使用的參數。

在步驟二中為預測流程，透過步驟一可以得知最適合此筆資料集的推薦模型參數，流程一開始載入由步驟一訓練得知的相關訓練參數，篩選出尚未評分的 ID 項目，此預測流程就是要針對這些尚未預測的項目做一個分數的預測，將篩選資料存入一個空間中，利用參數去訓練特徵向量並且做預測，最後將所有的尚未評分項目作評分，並且列出該 ID 的前五名評分項目作為往後推薦商品的依據，此兩步驟為本實驗整體步驟。

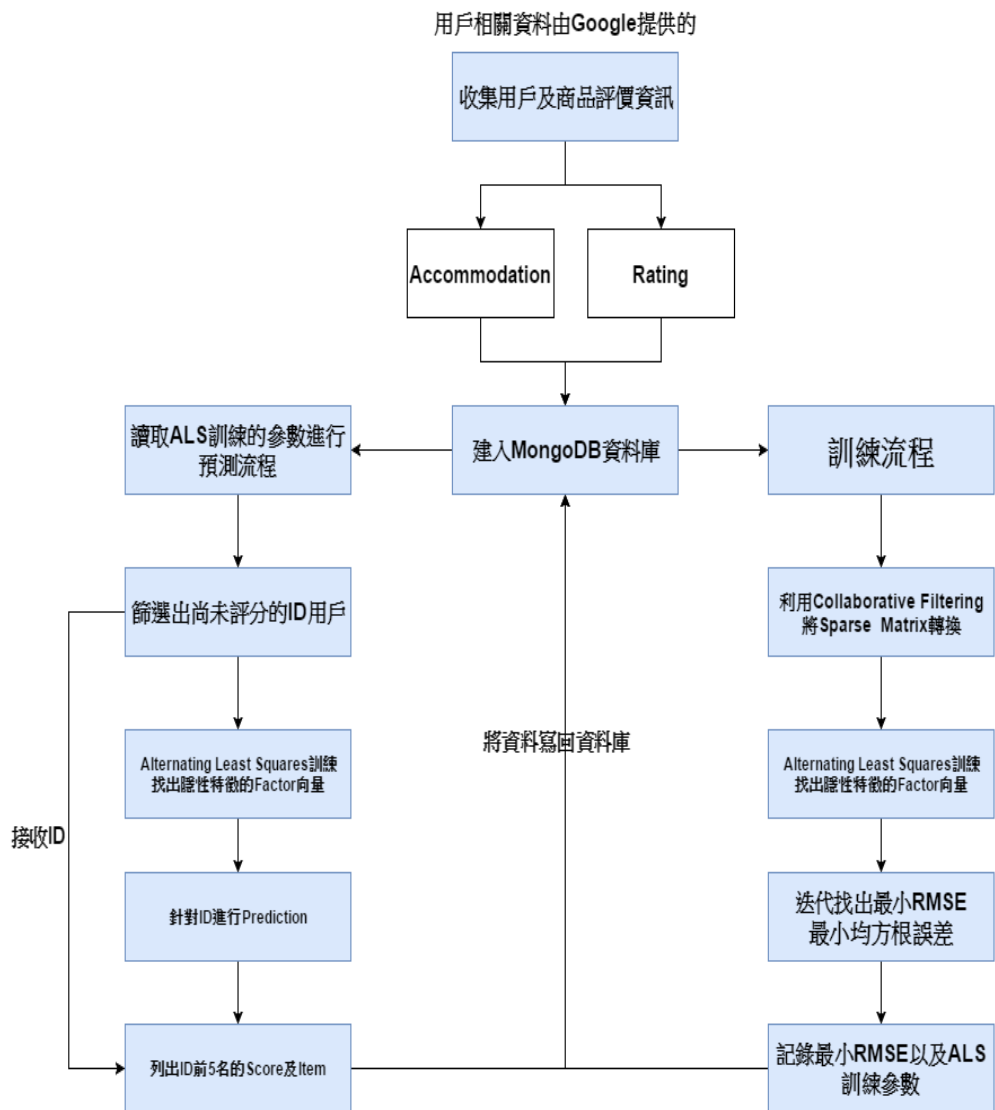


圖 12、情境流程圖

## 4.4.1、Model 訓練流程

本論文中撰寫訓練流程用來找出訓練資料集中最適的 Model，常見的深度學習中，訓練出來的 Model 好壞往往是訓練資料的多寡以及訓練資料的分配，本論文中針對訓練資料的分配，是將資料用 Random 分配成 60%Train data，20% Validation data 以及 20%Test data 的資料如圖 14。另外論文中 ALS 相關參數訓練的設計，是給定一個區間去做 Training 如圖 14。這些參數的用途介紹請參考表四。程式後半會使用 RMSE 計算出兩實際值與預測值之間的差異如公式 1 介紹，當這個差異越小所代表將會是此訓練集的 Best\_model，並且將 Best\_model 寫回 Mongodb 作為下一階段推薦流程的參數，詳細的流程請參考圖 13。

$$\min_{x,y} \sum_{u,i} c_{ui} (p_{ui} - x_u^T y_i - b_u - b_i)^2 - \ell(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2) \quad (\text{公式 1}) [29]$$

表六、ALS 相關參數設定介紹

<b>Rank</b>	主要影響需要萃取出 feature 的數量，數量的多寡會影響計算上的速度，以及準確度。
<b>Iteration</b>	迭代的次數越多表示所需的流程越久，如何地收斂取決於資料的大小以及資料的複雜程度。
<b>Lambda</b>	一般化係數，如果此參數為零，會讓結果貼近於我們的資料，但是對於未來預測的部分準確率會下降。



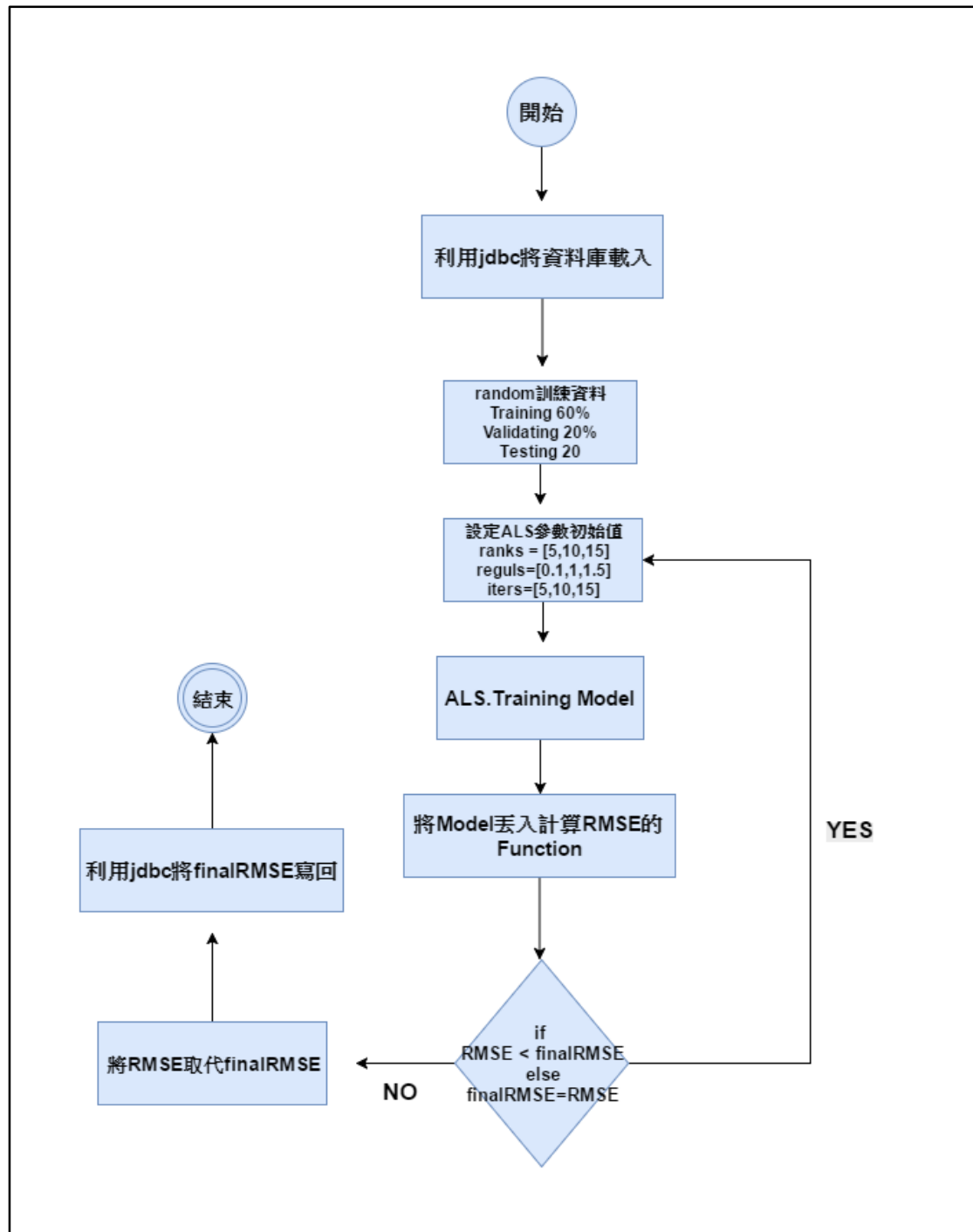


圖 13、訓練流程

```

conf = SparkConf().setAppName("app_collaborative")
sc = SparkContext(conf=conf)
sqlContext = SQLContext(sc)

jdbcUrl = 'jdbc:mysql://%s:3306/%s?user=%s&password=%s' % (CLOUDSQL_INSTANCE_IP, CLOUDSQL_DB_NAME, CLOUDSQL_USER, CLOUDSQL_PWD)

#計算distance function
#載入model,training的那筆資料,training個數
def howFarAreWe(model, against, sizeAgainst):
    # 不包含評分
    againstNoRatings = against.map(lambda x: (int(x[0]), int(x[1])))
    # 包含評分
    againstWiRatings = against.map(lambda x: ((int(x[0]),int(x[1])), int(x[2])))
    # 進行預測
    predictions = model.predictAll(againstNoRatings).map(lambda p: (p[0],p[1], p[2]))
    # 將資料join成(prediction, rating)
    predictionsAndRatings = predictions.join(againstWiRatings).values()

    # 計算差異並回傳
    return sqrt(predictionsAndRatings.map(lambda s: (s[0] - s[1]) ** 2).reduce(add) / float(sizeAgainst))
#結束

```

圖 14、訓練流程部分程式碼



## 4.4.2、商品預測流程

本論文中撰寫推薦流程用來訓練及推薦商品給用戶，程式一開始會載入資料庫相關檔案，並且篩選出尚未評分用戶 ID，主要就是針對這些用戶 ID 去做推薦。利用訓練流程所找到最佳 Model 的參數，作為在此程式中 ALS 的參數並且訓練出新的 Model。將篩選出來的尚未評分的所有用戶 ID 去做 Predict，計算出用戶尚未評分的項目並且列出推薦分數中前 5 高的商品 ID 以及分數，最後將這些數據儲存至資料庫以供未來網頁讀取，相關程式碼如圖 14，程式的主要流程請參考圖 15。

```
# 至db中讀檔案
jdbcUrl = 'jdbc:mysql://%s:3306/%s?user=%s&password=%s' % (CLOUDSQL_INSTANCE_IP, CLOUDSQL_DB_NAME, CLOUDSQL_USER, CLOUDSQL_PWD)
dfAccos = sqlContext.read.jdbc(url=jdbcUrl, table=TABLE_ITEMS)
dfRates = sqlContext.read.jdbc(url=jdbcUrl, table=TABLE_RATINGS)

# 找出有評分的项目及顧客
dfUserRatings = dfRates.filter(dfRates.userId == USER_ID).rdd.map(Lambda r: r.accoId).collect()
print(dfUserRatings)

# 找出沒有評分的使用者
rddPotential = dfAccos.rdd.filter(Lambda x: x[0] not in dfUserRatings)
pairsPotential = rddPotential.map(Lambda x: (USER_ID, x[0]))

#隨機資料分群
rddTraining, rddValidating, rddTesting = dfRates.rdd.randomSplit([6,2,2])
# 利用find_best_model.py所找到的最佳參數模型
# Rating, Rank, Iteration, Regulation
model = ALS.train(rddTraining, BEST_RANK, BEST_ITERATION, BEST_REGULATION)

# 計算所有的預測
predictions = model.predictAll(pairsPotential).map(Lambda p: (str(p[0]), str(p[1]), float(p[2])))
# 列出前5筆資料
topPredictions = predictions.takeOrdered(5, key=Lambda x: -x[2])
print(topPredictions)
```

圖 15、預測流程部分程式碼

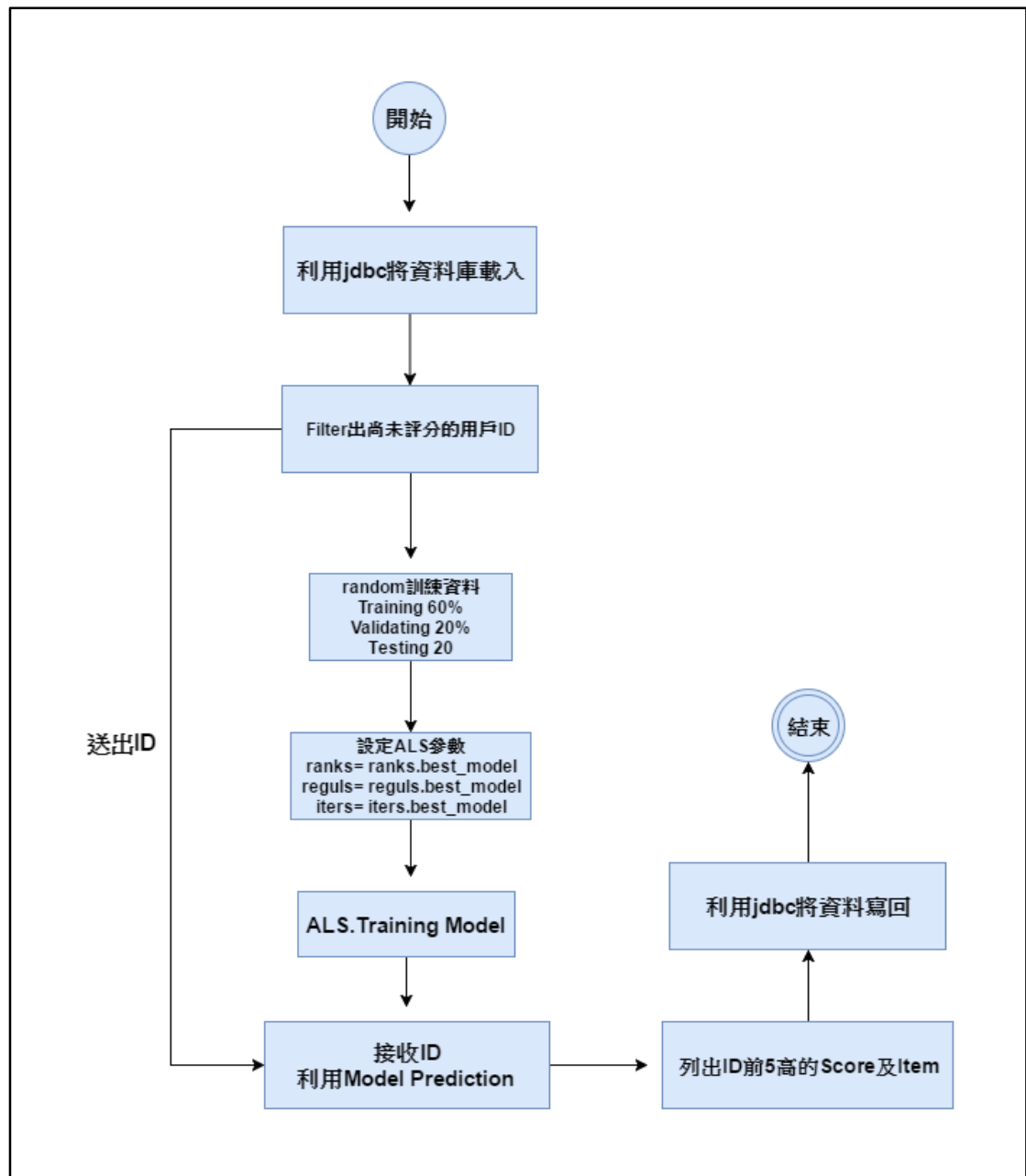


圖 16、預測流程

## 五、實驗結果與分析

本論文中所使用的 DataSet 是利用 Google 所提供的資料如圖 17、18，此筆資料的用處是，每個使用者對於部分建築物的評分分數，以及每筆建築物的相關資訊介紹。資料中的數據包含地點、建築物 ID、評分、地點以及簡短評語。這組數據主要是針對各地建築物所做的一個評分以及簡短的評語，而這些建築物分散在很多國家包含 Paris、London、Seattle 以及許多地方，其中建築物的數量總共有 100 棟建築物，圖 18 列出幾張建築物的外觀。另外從用戶收集到的評價總共有 1185 筆資料，資料內容包含用戶 ID、建築物 ID、評價分數三種資料如圖 19，目前因為本論文所收集的資料還不夠用來做為訓練的 DataSet 因此使用網路上所提供的，不過本論文題出的方法所訓練出來的推薦數據，是根據所輸入的 DataSet 的不同則輸出的推薦參數也會不同。本論文中的實驗硬體配備如表六，利用 Collaborative Filtering[5]的方式將收集來的資料建構成一個 Sparse Matrix[3]，以本實驗數據為例 X 軸與 Y 軸分別是用戶與建築物。而之所以會成為 Sparse Matrix 是因為每個使用者並沒有對所有的建築物去做評分，會導致矩陣中很多欄位是空的狀態。我們的目的就是利用 Deep Learning 的方式去訓練出用戶與用戶彼此之間的隱藏性特徵。

舉例來說某一用戶 1 對於 London 之中的 Castle 都相當的有興趣，用戶 1 給的評分都相當的高，對於另一個用戶 2 來說，該用戶 2 則是對於 Paris 之中的 Castle 相當的有興趣，可是該用戶 2 卻沒有看過 London 的 Castle，透過訓練的方式將會 Predict 用戶 2 在 London 之中的 Castle 給予一個相對高的分數作為推薦。因為在 ALS 當中每個用戶對於每個建築物的評價，將會被轉換成每個用戶對於每個 Factor 的評價，而這個 Factor 則是像上面所舉例的一樣，可能是特徵都是城堡，或者是這類的建築物都是屬於某一個都市，又或者這類型的建築物屬於較小的建築物，這些特徵就是所謂的隱藏性特徵，從實驗中圖 21 及 22 中可以看出，同一個用戶預測出來的前 5 比資料，所屬得特徵都是屬於 house。

利用這樣的方式將原本可能要計算的矩陣式非常大的簡化成兩個較小的矩陣相乘，分別是用戶的矩陣以及由 Factor 所組成的矩陣，問題則從原本的每個用戶對於建築物的評價，轉換成每個用戶對於每個 Factor 的評價，則我們的目標就是將這兩個 Vector 的乘積與原始的矩陣做比較，這兩者的差異越小越好，計算的方式是利用公式 1 去計算兩者的最小平方差，也就是計算出 RMSE 值。

圖 20 中為實驗中利用撰寫訓練流程所得到的 best\_model 參數，Rank、Regul、Iter 分別是之前所提到的 ALS 相關參數，Rank 主要影響需要萃取出 feature 的數





	A	B	C	D	E	F	G	H	I	J	K	L
1	10	1	1									
2	18	1	2									
3	13	1	1									
4	7	2	2									
5	4	2	2									
6	13	2	3									
7	19	2	2									
8	12	2	1									
9	11	2	1									
10	1	2	2									
11	20	2	2									
12	2	2	4									
13	3	2	1									
14	0	3	4									
15	4	3	5									
16	8	3	4									
17	7	3	4									
18	10	3	5									
19	16	3	5									
20	21	3	5									
21	23	3	5									
22	21	4	2									

圖 19、從用戶收集來的評價

表七、實驗硬體配備

電腦	CPU	記憶體	硬碟
Master	Intel(R) Xeon E5-2620	16G	3000G
Slave01	Intel(R) Core Q9500	4G	500G
Slave02	Intel(R) i7-2600	8G	1000G

[Stage 9151:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 9315:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 9364:.....]	(8 + 8) / 2]	[Stage 9639:.....]	(2 + 1) / 3]
[Stage 9763:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 9937:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 10191:.....]	(2 + 1) / 3]	[Stage 10515:.....]	(2 + 1) / 3]
[Stage 10241:.....]	(8 + 8) / 2]	1.10376079476	
[Stage 10629:.....]	(2 + 1) / 3]	1.86628141264	
[Stage 10813:.....]	(2 + 1) / 3]	1.86214771385	
[Stage 11063:.....]	(2 + 1) / 3]	[Stage 11391:.....]	(2 + 1) / 3]
[Stage 11098:.....]	(8 + 8) / 2]	1.43751324684	
[Stage 11585:.....]	(2 + 1) / 3]	[Stage 11586:.....]	(2 + 1) / 3]
[Stage 11685:.....]	(2 + 1) / 3]	1.44620044837	
[Stage 11943:.....]	(2 + 1) / 3]	1.44813932299	
[Stage 12267:.....]	(2 + 1) / 3]	1.44895503581	
[Stage 12381:.....]	(2 + 1) / 3]	2.77011662127	
[Stage 12565:.....]	(2 + 1) / 3]	2.77011799728	
[Stage 12819:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 13143:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 13257:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 13447:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 13695:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 14019:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 14133:.....]	(2 + 1) / 3]	1.86213387357	
[Stage 14317:.....]	(2 + 1) / 3]	1.86670786347	
[Stage 14571:.....]	(2 + 1) / 3]	1.86174075399	
[Stage 14602:.....]	(8 + 8) / 2]	[Stage 14695:.....]	(2 + 1) / 3]
[Stage 15009:.....]	(2 + 1) / 3]	1.45784373249	
[Stage 15193:.....]	(2 + 1) / 3]	1.44590412374	
[Stage 15443:.....]	(2 + 1) / 3]	1.44832864803	
[Stage 15771:.....]	(2 + 1) / 3]	1.44895557849	
[Stage 15885:.....]	(2 + 1) / 3]	2.77011687616	
[Stage 16069:.....]	(2 + 1) / 3]	2.77011799728	
[Stage 16323:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 16647:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 16761:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 16945:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 17199:.....]	(2 + 1) / 3]	2.77011799731	
[Stage 17217:.....]	(8 + 8) / 2]	[Stage 17523:.....]	(2 + 1) / 3]
Rank 10			
Regul 0.100000			
Iter 10			
Dist 1.060120			

圖 20、實驗數據圖 1

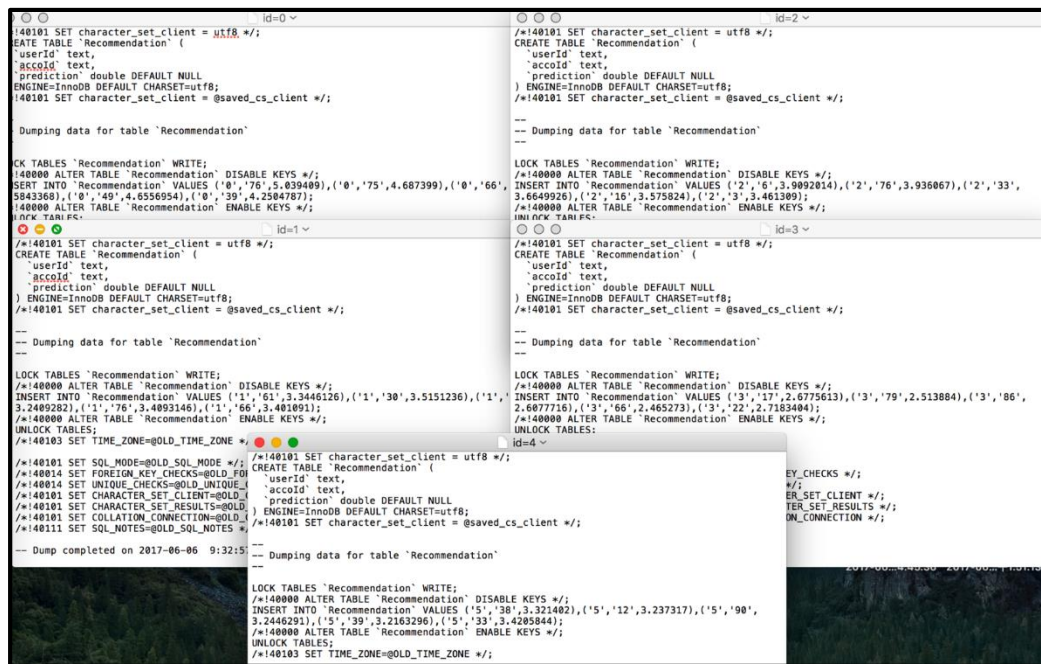


圖 21、實驗數據圖 2

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
65	Cosy Priv NYC		45	2	1	cottage																			
66	Beautiful P London		80	2	2.4	house																			
67	Great Calm Vancouver		2300	13	3.2	castle																			
68	Great Peace Paris		1800	21	1.1	castle																			
69	Hony Quiet NYC		65	1	3.1	cottage																			
70	Great Calm Paris		1050	10	2.2	manison																			
71	Cosy Calm San Francis		55	2	3.8	cottage																			
72	Beautiful C Paris		80	4	2.1	house																			
73	Nice Peace London		60	1	3.4	cottage																			
74	Great Calm Melbourne		2400	12	2.3	castle																			
75	Large Privi Berlin		50	4	3.6	house																			
76	Pleasant C Berlin		30	2	2.4	house																			
77	Great Privi Dublin		1150	10	2.4	manison																			
78	Great Privi Tokyo		2100	17	2.5	castle																			
79	Large Privi Vancouver		1050	10	4.8	manison																			
80	Big Quiet (San Francis		40	3	4.3	cottage																			
81	Hony Quiet Seattle		70	3	2.2	cottage																			
82	Cosy Peace San Francis		75	1	1.6	cottage																			
83	Cosy Calm San Francis		40	3	3.4	cottage																			
84	Great Peace Melbourne		700	8	3.2	manison																			
85	Nice Privi Auckland		55	1	4.9	cottage																			
86	Large Quiet London		100	4	4	house																			
87	Immens P San Francis		850	12	4.4	manison																			
88	Colonial Q Seattle		4100	16	3.6	castle																			
89	Nice Privi Seattle		45	2	3.2	cottage																			
90	Big Quiet I Seattle		35	5	3.2	house																			
91	Large Peace Melbourne		650	10	1.9	manison																			
92	Cosy Quiet San Francis		85	3	3.5	cottage																			
93	Great Quiet Vancouver		1800	16	3.9	castle																			
94	Great Peace Auckland		2900	25	3.3	castle																			
95	Great Calm San Francis		800	11	3.8	manison																			
96	Immens P Tokyo		800	9	3.8	manison																			
97	Cosy Quiet Auckland		75	1	2.3	cottage																			
98	Big Private Paris		2000	23	4.6	castle																			
99	Pleasant Q NYC		80	4	3.2	house																			

圖 22、實驗數據圖 3

## 5.1、評估實驗

本實驗中利用預測流程對用戶 0 所預測的分數如圖 23 所示，圖中內容主要包含訓練過程中的最佳參數 ALS[11]，以及該預測用戶在 DataSet 中所評分的建築物 ID，列出這個項目主要是因為，我們是要對每個用戶沒有評分的项目去做一個分數預測，所以先找出該用戶有評分過的项目，在對其他沒有評分過的项目去做評分。最後面內容包含的還有該用戶預測出來的前五名項目 ID 以及預測的分數，此預測內容為用戶 ID0 預測出項目 ID66、ID49、ID76、ID75、ID61 這五個項目，由預測的分數高至低排名分別是，4.56、4.45、4.39、4.03、4.02。

有了這些預測分數，利用用戶本身的評價資訊來證明本論文的推薦是有意義且有效的推薦，根據表七中顯示的包含用戶 ID0 在 DataSet 中評價的分數，圖中列出的是分數高於 3 的幾筆數據，也就是該用戶比較感興趣的幾筆評分，最後一個欄位屬於該项目的類型都是屬於 house，因此從該用戶的評分來說，用戶對於類型屬於 house 的项目評分出來的分數相對較高，可以解釋為該用戶對於 house 的喜愛程度較高，因此我們探討由本論文預測出來的項目是否會符合該用戶的喜愛。

由圖 23 的推薦分數我們列出表八，表中列出的 accoID 是根據本論文中推薦出的前五比分數較高的商品，分別是項目 ID66、ID49、ID76、ID75、ID61，並且列出每筆 ID 藉由 Accommodations DataSet 給的项目介紹，包含 Describe、Location、Type 等資訊。由表八很明顯的可以看出，藉由本論文所推薦的商品，不管是 Type 以及 Location 都與該用戶在評分表中的特徵相符，利用深度學習以及最小交替二乘法找出該用戶的特徵 vector，將屬於該特徵的用戶歸屬於一個類別，利用訓練找出所謂的 Boundary 分類，並且有效的作出預測。

```

17/06/08 13:12:16 INFO org.spark_project.jetty.util.log: Logging initialized @2390ms
17/06/08 13:12:16 INFO org.spark_project.jetty.server.Server: Jetty-9.2.z-SNAPSHOT
17/06/08 13:12:16 INFO org.spark_project.jetty.server.ServerConnector: Started ServerConnector@4b7c81d7{HTTP/1.1}{0.0.0.0:4040}
17/06/08 13:12:16 INFO org.spark_project.jetty.server.Server: Started @2564ms
17/06/08 13:12:17 INFO com.google.cloud.hadoop.fs.gcs.GoogleHadoopFileSystemBase: GHFS version: 1.6.0-hadoop2
17/06/08 13:12:18 INFO org.apache.hadoop.yarn.client.RMProxy: Connecting to ResourceManager at recommendation-m/10.148.0.3:8032
17/06/08 13:12:20 INFO org.apache.hadoop.yarn.client.api.impl.YarnClientImpl: Submitted application application_1496905550985_0002
16 0.1 10
[["12", "u'16'", "u'22'", "u'28'", "u'3'", "u'30'", "u'33'", "u'38'", "u'6'"]]
PythonRDD[13] at RDD at PythonRDD.scala:48
[Stage 23:> (0 + 0) / 3] [('0', '66', 4.56445
60389017155), ('0', '49', 4.450931411075099), ('0', '76', 4.39212002597854), ('0', '75', 4.038687972452062), ('0', '61', 4.0216248723581565)]
[Stage 184:> (0 + 3) / 3][Stage 184:=====> (1 + 2) / 3]
17/06/08 13:12:59 INFO org.spark_project.jetty.server.ServerConnector: Stopped ServerConnector@4b7c81d7{HTTP/1.1}{0.0.0.0:4040}

```

圖 23、實驗數據圖 4

表八、用戶評分介紹

userID	accoID	rating	Type	Location
0	3	4	House	London
0	6	5	House	Dublin
0	12	5	House	Seattle
0	16	4	House	Melbourne
0	22	4	House	Auckland
0	28	4	House	Tokyo
0	30	5	House	Berlin
0	33	4	House	Tokyo
0	38	4	House	San Francisco

表九、實驗數據介紹

accoID	Describe	Location	Type
66	Beautiful Private Villa	London	House
49	Big Private Villa	NYC	House
76	Pleasant Calm Villa	Berlin	House
75	Large Private Place	Berlin	House
61	Large Calm Place	NYC	house

## 5.2、叢集實驗

論文中針對 Spark 相關叢集設定做一個效能上的測試，此實驗是利用本論文訓練方式以及相同資料集，針對不同的 Spark 配置去做一個時間上的比較。實驗中主要分成四種配置，分別是 Cluster-1-2-2、Cluster-1-2-4、Cluster-1-4-2、Cluster-1-6-2，以 Cluster-1-2-2 舉例來說代表的意思是，此配置為一台主要的 Master Server 且包含兩台 Worker，而每一台機器的 CPU 數量為 2。

實驗分別針對 CPU 數量以及 Worker 數量去做一個探討與比較，對於 Cluster-1-2-2 以及 Cluster-1-2-4 是相同的叢集分配而改變不同的 CPU 數量，測試此分配對於執行相同訓練所需要的時間，藉由圖 X 可以明顯的看出 Cluster-1-2-2 以及 Cluster-1-2-4 並沒有很明顯的時間上差距，原因可能是因為此訓練方法可能尚未複雜到需要使用到 4 個 CPU 來做為訓練的設備，利用 2 個 CPU 就能夠快速解決此訓練方法。

另外一個比較是對於叢集的分散程度的 Worker 數量做一個討論與比較，對於 Cluster-1-4-2 以及 Cluster-1-6-2 來做比較，分別的 Worker 數量是 4 比 6，也就是說 Cluster-1-4-2 叢集包含 Master 以及 Worker 總共有 5 台主機，每台主機的 CPU 數量分別為 2CPU，則 Cluster-1-6-2 叢集包含 Master 以及 Worker 總共有 7 台主機，每台主機的 CPU 數量也是 2CPU。根據我們的實驗結果圖 X 中可以看出在 Cluster-1-6-2 所跑出來的時間高於 Cluster-1-4-2 有 124 秒之多，根據叢集是架構來看，使用越多的 Worker 來處理相對會縮短運算時間，可是本實驗中所測試出來的結果卻導致更長的執行時間。歸納出來的原因可能是因為 Master 與 Worker 之間傳輸的速度問題，本實驗採用的網路線沒有達到相關規格，可能是導致此實驗時間拉長的主因，因為越多的 Worker 代表需要有更多的傳遞次數，導致整個運算時間因此被拉長。



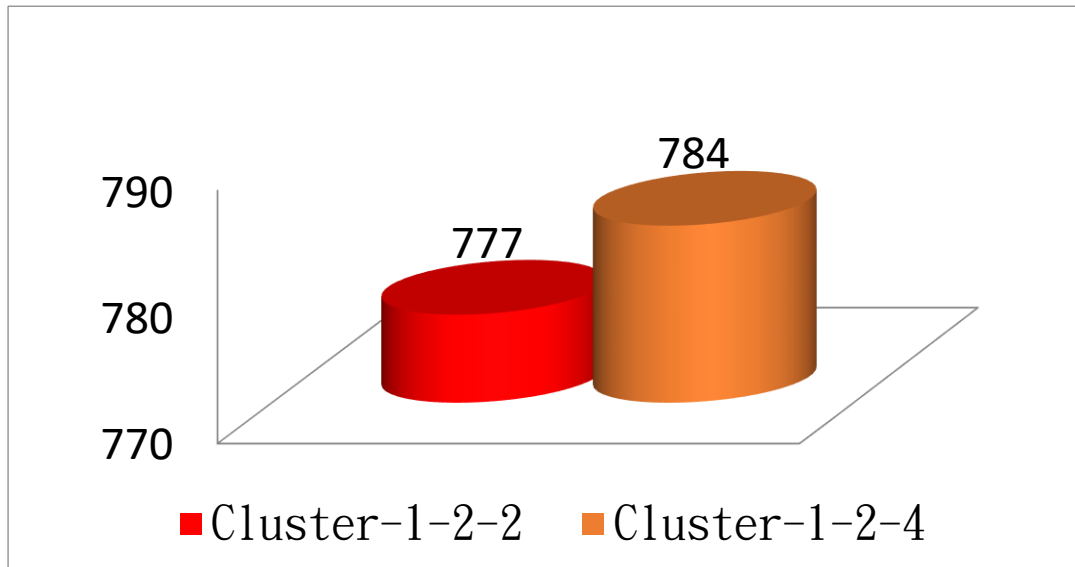


圖 24、CPU 數量影響比較

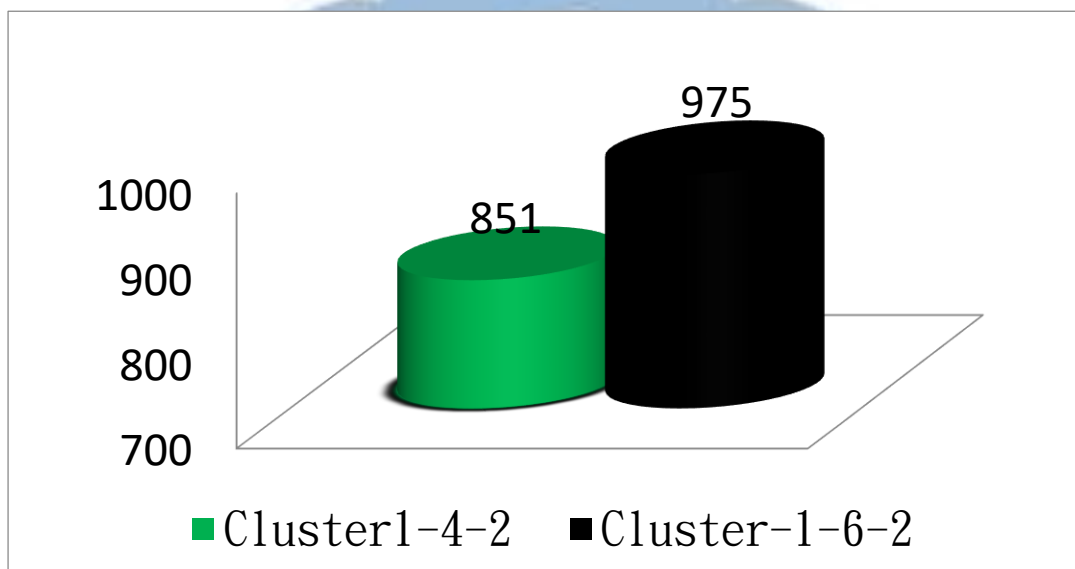


圖 25、Worker 數量影響比較

## 六、結論

本論文利用 Deep Learning 結合 Collaborative Filtering 的方式實作 Recommendation System，在傳統的 Recommendation System 中以本實驗數據為例，如果使用傳統基於內容的方式去做推薦，那麼我們就必須要得到許多關於產品的顯性信息，所謂的顯性信息就像是建築物的年齡、材質、藝術性等相關的建築物訊息，且為了達到準確的推薦訊息，顯性訊息的多寡嚴重影響推薦的準確性。

在目前科技的使用情況下顯示，太多的顯性訊息對使用者來說是一種負擔，有些資訊更會包含個人資料，會影響到個資問題，要收集到這麼多的訊息，以推薦系統的角度來看是不太容易的，更何況當顯性訊息數量太多，在設計整個推薦系統上的 Decision Tree[30]就會長得非常的大也非常複雜，比起 Deep Learning 的方式去訓練資料來說沒有比較輕鬆，而且傳統的基於內容的推薦方式會有所謂的冷啟動問題，這個問題因為第一次用戶沒有資料而無法做到一個推薦的方式。

相對於 Collaborative Filtering 的方式，只會需要收集到用戶對於商品的評價，並利用這些對商品評價，去做 ALS 深度學習的方式找出上面所提到的 Factor，再根據訓練的 Model 去做一個正確的推薦，基於 Collaborative Filtering 的方式有一個非常不錯的優點，因為他是根據行為較相似的群體用戶的信息來做推薦，所以用戶可能會被推薦出一些新奇的產品，推薦的商品可能是用戶之前都沒有買過的商品。在傳統的系統中，這個地方相對來說較為薄弱，通常是用顯性信息以及歷史信息來做推薦，這也是本論文為何要使用 Collaborative Filtering 作為主要的方式。

Hao Wang[31]等人所提出的論文中，利用 Collaborative Filtering 以及 Deep Learning 的方法能夠有效的推薦。在本論文中利用 Deep Learning 的 ALS 方式訓練的推薦 Model，確實能夠有效的推薦用戶商品。本論文未來所使用的資料集將利用一些使用者的生理資訊，並串接用戶的點餐資訊作為本實驗的一個應用。需要將生理資訊先做處理，再分層去訓練其隱性特徵，未來可作為探討之議題。

## 七、參考文獻

- [1] 矩陣分解，[https://en.wikipedia.org/wiki/Matrix\\_decomposition](https://en.wikipedia.org/wiki/Matrix_decomposition)
- [2]<https://zh.wikipedia.org/wiki/%E7%A8%80%E7%96%8F%E7%9F%A9%E9%98%B5>
- [3] Youtube 推倒過程，<https://www.youtube.com/watch?v=o8PiWO8C3zs>
- [4] 推薦系統  
<https://zh.wikipedia.org/wiki/%E6%8E%A8%E8%96%A6%E7%B3%BB%E7%B5%B1>
- [5] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Bosagh Zadeh WTF: The who-to-follow system at Twitter, Proceedings of the 22nd international conference on World Wide Web
- [6] Prem Melville and Vikas Sindhwani, Recommender Systems, Encyclopedia of Machine Learning, 2010.
- [7] Last.fm，<https://zh.wikipedia.org/wiki/Last.fm>
- [8] R. J. Mooney & L. Roy. Content-based book recommendation using learning for text categorization. In Workshop Recom. Sys.: Algo. and Evaluation. 1999.
- [9] Rubens, Neil; Elahi, Mehdi; Sugiyama, Masashi; Kaplan, Dain. Active Learning in Recommender Systems. (編) Ricci, Francesco; Rokach, Lior; Shapira, Bracha. Recommender Systems Handbook 2. Springer US. 2016. ISBN 978-1-4899-7637-6.
- [10] Elahi, Mehdi; Ricci, Francesco; Rubens, Neil. A survey of active learning in collaborative filtering recommender systems. Computer Science Review, 2016, Elsevier.
- [11] ALS 介紹，<https://read01.com/A4BjJm.html>
- [12] Apache Spark，<https://spark.apache.org/>
- [13] 維基百科 Apache Spark，[https://zh.wikipedia.org/wiki/Apache\\_Spark](https://zh.wikipedia.org/wiki/Apache_Spark)
- [14] Xin, Reynold; Rosen, Josh; Zaharia, Matei; Franklin, Michael; Shenker, Scott; Stoica, Ion. Shark: SQL and Rich Analytics at Scale(PDF). June 2013.
- [15] Matei Zaharia. Spark: In-Memory Cluster Computing for Iterative and Interactive Applications. Invited Talk at NIPS 2011 Big Learning Workshop: Algorithms, Systems, and Tools for Learning at Scale.

- [16] Sparks, Evan; Talwalkar, Ameet. Spark Meetup: MLbase, Distributed Machine Learning with Spark. slideshare.net. Spark User Meetup, San Francisco, California. 2013-08-06 [10 February 2014].
- [17] 維基百科 Python , <https://zh.wikipedia.org/wiki/Python>
- [18] 維基百科 MongoDB , <https://zh.wikipedia.org/wiki/MongoDB>
- [19] Hadoop 相關架設方式  
[http://www.cc.ntu.edu.tw/chinese/epaper/0036/20160321\\_3609.html](http://www.cc.ntu.edu.tw/chinese/epaper/0036/20160321_3609.html)
- [20] Putty , <https://zh.wikipedia.org/wiki/PuTTY>
- [21] SSH , [https://zh.wikipedia.org/wiki/Secure\\_Shell](https://zh.wikipedia.org/wiki/Secure_Shell)
- [22] SSH 無密碼登入 ,  
<https://blog.gtwang.org/linux/linux-ssh-public-key-authentication/>
- [23] Spark 安裝流程 , <https://jerryneet.io/install-hadoop-2-6-0-on-ubuntu14-04/>
- [24] <https://www.tibame.com/>
- [25] 訓練資料來源 , <https://movielens.org>
- [26] accommodations.csv ,  
<https://storage.googleapis.com/solutions-public-assets/recommendation-spark/sql/accommodations.csv>
- [27] ratings.csv ,  
<https://storage.googleapis.com/solutions-public-assets/recommendation-spark/sql/ratings.csv>
- [28] MongoDB , <https://www.mongodb.com/products/compass>
- [29] RMSE , [https://en.wikipedia.org/wiki/Root-mean-square\\_deviation](https://en.wikipedia.org/wiki/Root-mean-square_deviation)
- [30] Decision Tree , <https://zh.wikipedia.org/wiki/决策树>
- [31] Hao Wang, Naiyan Wang, Dit-Yan Yeung : Collaborative Deep Learning for Recommender Systems, 2015
- [32] 科技新報:Amazon 推薦系統  
<http://technews.tw/2016/07/17/amazon-page-system/>
- [33] Amit Sharma: Estimating the causal impact of recommendation systems from observational data