Big Data Systems : Term Project

Student : 610721204　陳克威

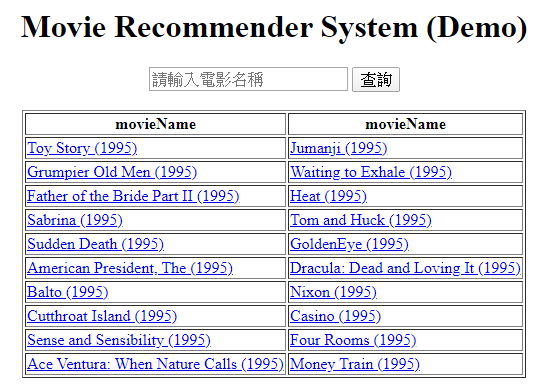


圖1 - 電影資訊網站首頁

1. Data Set

此專案使用著名的MovieLens Latest Datasets (2018/09)做為數據集[1]，總共包含280,000個用戶、58,000部電影以及27,000,000個評分。

此數據集為開放資料，可以在下方永久網址下載。<https://grouplens.org/datasets/movielens/latest/>。

1. Open-Source Big Data Systems / Tools

2-1資料處理工具

此專案需要對用戶-電影的評分矩陣(User-Movie Rating Matrix)進行矩陣分解，由於矩陣分解是迭代算法，因此我們選擇PySpark (RDD)做為平行化運算工具，並使用PySpark - MLlib進行ALS矩陣分解。

2-2 資料處理過程

將評分資料讀入PySpark RDD後處理成ALS的輸入格式，接著使用ALS訓練矩陣分解，此處需要實驗調整參數，不同的資料大小、類型及稀疏度都可能需要不同的參數設定，此處經過實驗後選擇特徵數為20個，正規化參數為0.1，迭代次數10次。

經過ALS矩陣分解後，可以使用新的矩陣來推薦電影，例如為個別用戶推薦預測評分最高的Top-N個電影(用戶尚未評分過的電影)，也可以反過來使用電影推薦用戶(如發送E-mail通知)。

2-3 冷起始(Cold Start)問題

以ALS矩陣分解做為推薦系統非常容易，對比傳統User-based或Item-based的推薦系統，ALS不需要計算相似度，只需要將用戶的向量對所有電影的向量做内積(Inner Product)取得預測評分後取Top-N做推薦即可，然而這也造成冷起始的問題，也就是當新用戶沒有評分資料時就無法做出推薦，因此我們希望以電影的相似度做為非個人化(non-personalized)的推薦來解決冷起始的問題。

2-4 電影相似度

此專案選擇使用評分矩陣做為輸入，此輸入與用戶及電影的資訊無關，因此我們需要利用電影評分的向量做為相似度計算，在原始評分矩陣中資料非常稀疏(Sparse)，而且維度也很大(等同於用戶數)，因此不太適合直接計算相似度。而在經過ALS矩陣分解後，我們可以利用分解後的低維電影向量來計算相似度，這裡維度為20個，我們選擇常見的Cosine Similarity來進行計算，當然這裡也可以使用PySpark進行平行處理，計算完成後我們可以取Top-N來做為推薦。

1. Movie Web Site with Recommender System

3-1網站架設

本專案使用資料集內提供的電影編號、電影名稱與IMDb連結做為資料來源，使用PHP、JavaScript與MySQL來實現電影查詢功能，並且有自動完成(Autocomplete)的功能，如圖2所示。

3-2電影推薦頁面

當使用者選擇電影後會進入電影推薦頁面，此時系統會找出此電影的ID，接著呼叫Python計算此電影與其他電影(約56,000部)的相似度並排序後返回前20部電影做為推薦，系統再顯示這20部電影的資訊(IMDb連結)，如圖3所示，整個處理過程經過測試一般在2秒以內可以完成。

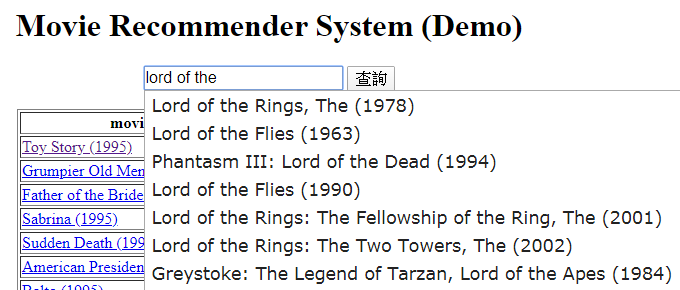


圖2 - 電影查詢功能

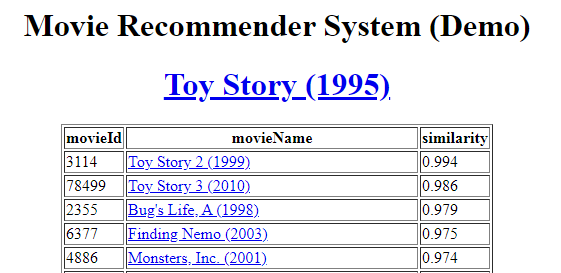


圖3 - 利用電影相似度做推薦

1. Future Work

此專案在線上服務過程中進行相似度計算，因此執行時間比較長(約2秒)，以基本Demo網站來說已經足夠，但是在一般場景下的網站卻會顯得太慢與沒效率。

在未來我們可以先進行離線預處理(Offline Preprocessing)，善用Spark的平行化計算可以為我們省下許多時間，計算完相似度與排序後可以為每部電影儲存前100部相似電影，在線上服務時就可以直接取Top-N做為電影推薦，延遲時間(Latency)就可以達到所謂的毫秒量級，也因為Spark的高效率使我們可以經常性的更新這個評分矩陣與相似度矩陣，如每周、每日，甚至是每小時。

另外，我們可以為網站做些樣式設計以及電影資料的抓取(爬蟲)，讓使用者可以不用透過連結，而是在網站內就能預覽電影資訊，不過這已經有些偏離專案主題了，在未來希望可以實現完整且速度快的電影資訊網站(以及推薦系統)。

1. Reference

[1] MovieLens Latest Datasets

(last access:2020/01/14)<https://grouplens.org/datasets/movielens/latest/>

[2] How does Netflix recommend movies? Matrix Factorization

(last access:2020/01/14)<https://www.youtube.com/watch?v=ZspR5PZemcs>

[3] Recommendation Engines Using ALS in PySpark (MovieLens Dataset) (last access:2020/01/14)<https://www.youtube.com/watch?v=FgGjc5oabrA>

[4] 深入理解Spark ML：基于ALS矩阵分解的协同过滤算法与源码分析 (last access:2020/01/14) <https://blog.csdn.net/u011239443/article/details/51752904>

[5] Apache Spark with a Recommender System (last access:2020/01/14) <http://www.3leafnodes.com/apache-spark-introduction-recommender-system>

[6] Building a Movie Recommendation Service with Apache Spark & Flask - Part 1 (last access:2020/01/14) <https://www.codementor.io/@jadianes/building-a-recommender-with-apache-spark-python-example-app-part1-du1083qbw>

[7] Movie Recommendation using Big Data Engine powered by Apache Spark (last access:2020/01/14) <http://datasqz.com/movie-names/movie-similar>