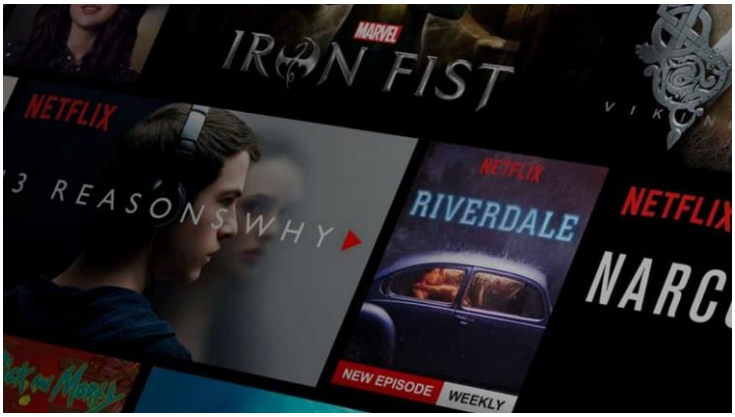



# 電影推薦系統

資料庫系統期末專題

組別：CCLab
組員：
資科碩一　盧佳妤 108753120
　　陳先灝 108753107
　　王均捷 108753113
　　段寶鈞 108753116
財政四　　方文忠 105205039

課程名稱：資料庫系統
日期：2020/06/29

## 壹、 背景與動機

在當今的社會之中，社群網路與影音串流平台已經成為了我們生活中重要的一部份。其中，個人化推薦是這些平台重要的功能之一，也就是從使用者相關的資料，以及其過去的使用紀錄之中，預測使用者未來的行為。以影劇平台 NetFlix 為例，它便是透過使用者在註冊時填寫的資料，以及其過往的觀看紀錄和評分之中，推薦使用者「他們可能會喜歡的影片」。

在同樣的例子裏，進行個人化推薦之前，基本需要保存的資料是：

1. 使用者註冊資料(如帳號、密碼、個人基本資料等等)
2. 影劇相關資料(如電影標題、導演、作品描述等等)
3. 使用者的過往觀看與評分紀錄

且這些資料都需要隨著使用者的操作與新作品的推出進行更新。

此外，在這樣一個影音串流平台之中，為了方便使用者的使用，搜尋功能也是必須的，除了能讓使用者以關鍵字搜尋電影之外，使用者的個人搜尋紀錄，也可以成為未來進階推薦方法的重要資料之一。

綜上所述，一個理想的個人化影音推薦系統需要能有效率地保存、更新資料，且需具備基本的搜尋功能。考慮到這些，資料庫便是一個最好而簡便的實現方式。

因此，我們以 IMDB 電影資料庫為基礎，建立了一個具備更新與刪除，以及搜尋等基本資料庫功能的系統。並搭配 Movielens 資料集，取得初始的使用者行為紀錄，再以 Graph Embedding 與 context-based 等推薦演算法，針對每個使用者的觀看行為，進行了電影的個人化推薦。

## 貳、 需求分析

一、資料需求分析

1. 電影資料集
   - 電影：每部電影有一個名稱、一個發行年、1~n 個導演、1~n 個演員、1~n 個製作公司、1~n 個電影類別。
   - 導演：導過 1~n 部電影。
   - 演員：參與過 1~n 部電影。
   - 製作公司:製作過 1~n 部電影。
2. 會員資料(使用者資料)
   - 每個人有一帳號、一密碼、一姓名、一性別。
3. 使用者點擊資料
   - 使用者點擊 0~n 部電影。
   - 電影被 0~n 個人點擊過。

二、系統功能分析

1. 登入及註冊功能，成為會員後可更新個人資訊和刪除帳戶。
2. 搜尋功能：可根據關鍵字(如：演員、導演…等)回傳相關搜尋結果。
3. 推薦功能，分為三種：

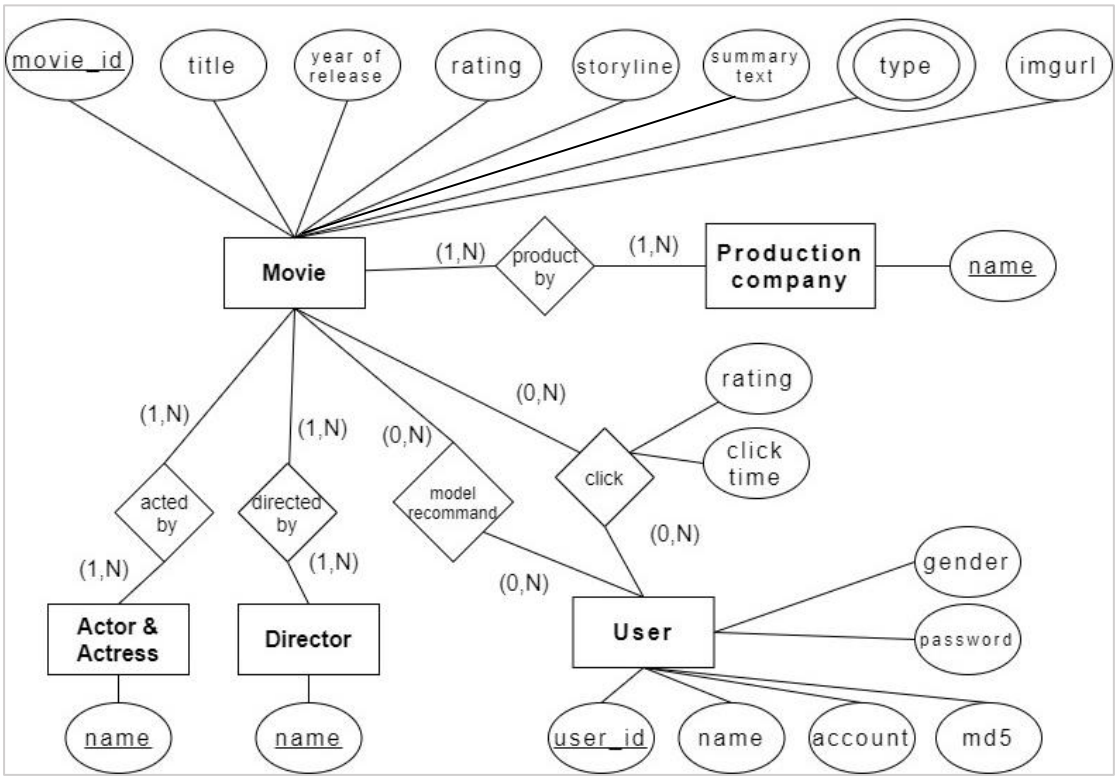
   a.根據會員點擊電影次數排序後，取前幾部電影的導演、演員直接推薦有相同導演或演員之電影。

   b.將會員及電影點擊數據丟入模型訓練，透過推薦模型找出會員可能想看的電影並推薦。

   c.新會員沒有任何點擊電影資料，故推薦資料庫中點擊次數前十高的電影。

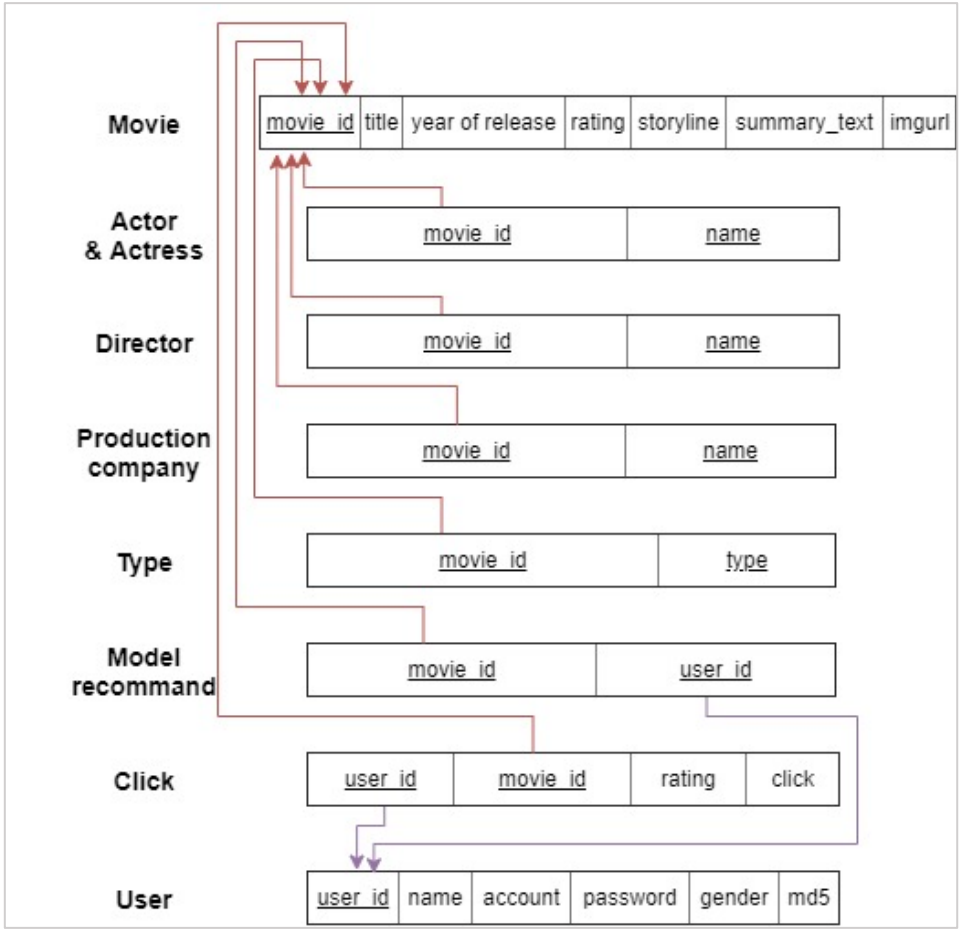
## 參、 ER Model

電影與製作公司、導演、演員為多對多關係；使用者會點擊電影，記錄點擊次數及評分，使用者會點擊 0(新使用者)~N 部電影，電影會被 0~N 個人點擊過；另外，會透過模型推薦電影給使用者。



## 肆、 Relational Schema

根據 ER Model 轉 Relational Schema 的規則轉換如下。




## 伍、 推薦演算法概述

一般在進行推薦的時候，以影音串流平台為例，是將「使用者」、「電影」視為 Node，「喜愛與否」視為 Edge，建立成一張 Graph 進行。其中，將每一個 Node 以向量表示的方法，則稱為 Graph Embedding。我們使用的便是其中一種 Graph Embedding 的做法。

我們重要的假設在於，考慮一組都喜歡 A 的使用者 B 和 C，希望能夠讓 A、B、C 的向量表示互相接近(比如 Cosine Similarity 較大)，而與其他無關的 node 向量表示距離較遠。

此外，引進 Message Passing 的概念，迭代地進行每個 Node 及其 Neighbor 的 Embedding 過程，讓整張 Graph 的 Node Embedding 彼此互相影響，從而取得更好的結果。

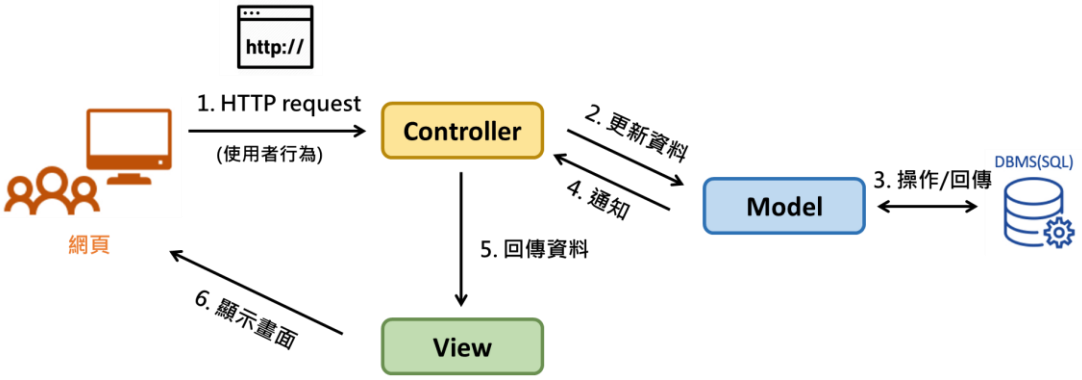
而考量到單純的使用者行為可能有所不足，我們將文字間的關聯性（例如電影的劇情描述）也視為一種特徵加入 Graph Embedding 的考量之中，讓文字特

徵類似（ 如皆與情愛、科幻有關的電影 ），具備相似的 Embedding。

最後，為了能有效利用資料庫的性質並取得多樣化的推薦結果，我們也混合了傳統的推薦方法，即推薦與使用者過往觀看紀錄中，具有相同導演或是男女主角的電影給使用者，形成一混合性的推薦系統。

## 陸、 系統架構

推薦系統開發之程式語言及 DBMS 為前端使用 javascript、html、css 實作，後端資料庫管理系統使用 SQLite 並使用 Python 操作。




功能分為三部分：會員之登入、註冊、更新資訊、刪除資料；搜尋，分為兩部分，呈現順序為先查找電影 title 再搜尋電影的介紹內文；推薦功能，分為三部分，如需求分析章節所述，在會員有點擊資料的情況下，網頁呈現優先順序為模型推薦結果、依導演推薦、依演員推薦，若為新會員，則推薦點擊次數前十高的電影。

## 柒、 心得

在實作資料庫專案的過程中，不同階段遇到的困難及學習到的東西不盡相同，前處理時最主要遇到的困難是，從 IMDB 爬資料下來的時候是以電影名稱作為索引，但跟 MovieLens 1M 的資料作合併時很容易出錯。因為兩邊文字使用的編碼不同，所以部分符號在轉換為 utf-8 編碼時會不一樣。這樣一來在使用電影名稱將兩邊資料進行合併時就會有無法順利對在一起的情形產生。我們嘗試了很多方法過後，已經可以把大部份資料成功合併，但還是有十餘筆資料對不起來，此時只好選擇用人工改動的方式來處理例外情形。另外，在建立 CSV 檔後傳上資料庫時，發現將使用者評分過的電影資料表上傳會出錯，原因是 MovieLens 1M 上有部分電影的名稱兩兩相同，但我們從 IMDB 上只有爬到其中一部的資料。所以需要特別做一次過濾 ID 的步驟，以保持 IMDB 和 MovieLens 1M 兩邊的電影數量相等。

而爬蟲能幫我們自動化、大量的爬取資料，結合資料庫格式化的儲存資料，

能幫助我們後續的應用（如：搜尋、集合運算）更有效率。這次爬取 IMDB 網頁的資訊，雖然 IMDB 是靜態的呈現資訊，相較於動態比較好取得，但是由於依電影紀錄當下的年代不同以及紀錄資料不一樣，導致至少有三種不一樣的格式，需要依格式爬取。

另一個遇到的問題是，即使頁面格式相同，因為資料不齊全的因素，造成呈現位置不一樣（比如 movie A 的公司在第六行，movie B 的公司卻在第三行），這種情況下只能線性搜索整張表，以取得這種資訊

最後一個問題是資料不齊全，因為搜索錯誤也是得到空的資訊，所以得到空的回傳值時需額外確認到底是不是真的沒有資料，所幸這種情況並不多資料庫的儲存能與爬蟲的資訊獲取相得益彰，這次實作是很好的學習經驗。

在前後端的部分，我們討論並摸索很多種串接的方式，並且每一種可能的狀況都需要考慮到，才能避免例外情況發生，導致系統出錯。

總之，在此次專案上可以將老師上課的內容實作出來，除了加深印象外，也學習到許多知識，收穫量多，很有成就感。

## 捌、 分工

- 前端、API：王均捷 28%
- 後端：盧佳妤 18%
- 推薦模型：陳先灝 18%
- 資料庫 data 爬蟲並整理：段寶鈞、方文忠 各 18%