Data Mining

project1 Association Analysis

孫啟慧

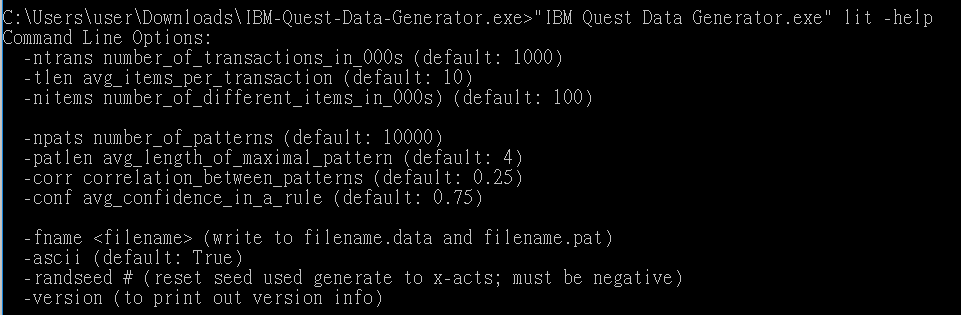
P78063033

1. **實驗設計**

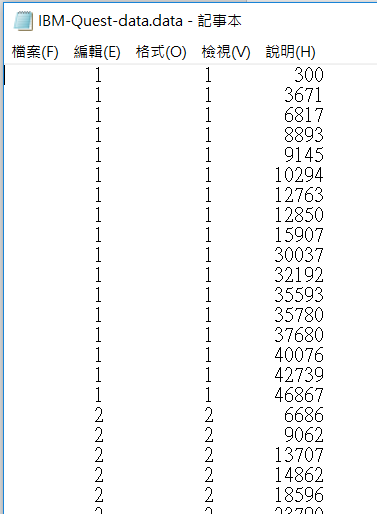
我用JAVA實作了Apriori演算法，其中分為使用Set和List實現的暴力法，以及使用 FP-growth來進行優化的Apriori。利用IBM Quest Synthetic Data Generator生成的使用者交易資料來模擬實際交易，並作為input來分別測試兩種不同的Apriori演算法的性能差異。我還選取了Kaggle中NHL Game Data的game.csv，通過運行自己寫的Apriori在其上，以生成相應的association rule，來分析各個賽季的成績受哪些因素的影響。最後我們使用WEKA對以上的實驗資料集進行分析，在相同的support和confidence設定之下，得到相應的association rule，從而對自己撰寫的Apriori程式的結果進行驗證。

* 1. **Input的預處理**：

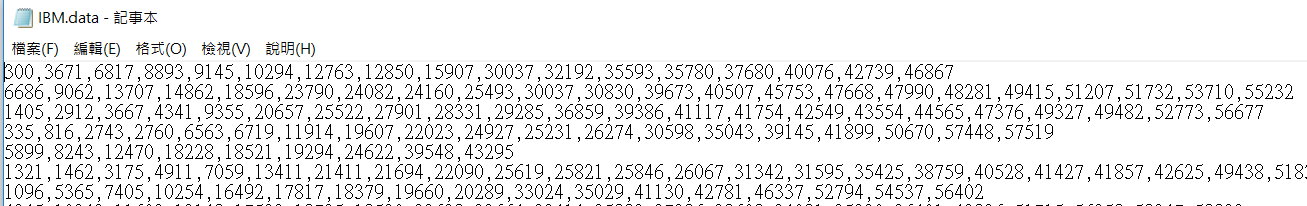
1. IBM Quest Synthetic Data Generator生成的資料。

該資料生成器包含多個指令，具體的預設參數顯示如下：

生成的資料格式如下，包含多列的空格。



其中第一欄是transaction id，第二欄是customer id，第三欄則是商品id。因為相同的transaction id和customer id代表同一個一個人的同一筆transaction，從而我有寫一個處理資料的InputProcess.java文件 （\Apriori Brute\src\InputProcess.java）對資料進行處理，處理後的資料如IBM.data所示，即每一列代表一次transaction，用逗號分割的各個數字分別代表不同種類的商品id，不同列是不同的transaction。



1. Kaggle的NHL Game Data的game.csv

Kaggle作為一個資料建模和資料分析的競賽平臺，上面經常會有研究人員和企業發起不同的比賽，通過他們提供的資料，參賽者可以組隊參加比賽對其進行建模，提交的各個模型會得到其準確率的評判，從而決定出相應的名次。

我選取了NHL Game Data比賽中提供的game.csv，來作為相應的input data。 該csv包含game\_id，season，season type，away\_team\_id，home\_team\_id，outcome，home\_rink\_side\_start等項目，由於資料類型較為複雜，故我只選取了season，outcome，home\_rink\_side\_start三個欄位，對其進行預處理之後如game1.csv所示。其中season代表相應的賽季，outcome則是表示隊伍贏得比賽的情況，其中home代表主場隊伍獲勝，而away則是客場隊伍獲勝。 Home\_rink\_side\_start則是代表主場隊伍相較於Time/Score keepers的相對位置。

通過對預處理過後的game1.csv進行相應的關聯規則的分析，我期望可以得到每個賽季中相對位置對勝率的影響。

\*因為第一種input的原始資料分隔是空格，另外一個則是逗號，故寫的Apriori中資料的讀取部分有所調整，對於兩種分隔形式均可識別。即整個程式的input既可以是.csv也可以是用空格分割的資料形式。

1. Weka

Weka是由紐西蘭懷卡托大學用JAVA開發的資料採擷軟體，我有參考網上的視頻嘗試著玩了一下，且UI介面可以很容易上手。雖然其他功能可以吃csv檔案，但是association rule則只能吃ARFF格式的檔案，好在weka本身有提供兩種方法來進行格式的轉換，一種是在weka的主功能表中可以找到“Simple CLI”模組，根據相應的命令列即可完成檔案類型的轉換。其指令為：

java weka.core.converters.CSVLoader filename.csv > filename.arff

而另外一種方法則是weka3.5之後的版本有提供“Arff Viewer”的模組，可以對相應的csv檔案進行流覽，然後可以選擇另存為ARFF檔。通過設定相同的參數可以對我自己實現的Apriori演算法和FP-growth演算法生成的association rule進行驗證。

* 1. **Brute 的apriori：**

**演算法設計：**

Step 1. 讀入input，得到1 item的頻繁子集，並計算相應的count將其保存到Arraylist當中。

Step2. 迴圈遍歷Step 1中的得到的Arraylist當中，通過判斷該子集的任兩個itemset是否相等來決定是否要merge相應的兩個itemset，經過迴圈連結k-1次得到最終的k items頻繁子集。

Step3. 對於Step2中最終生成的k items set是否滿足最小minSup (support count) 進行判斷，小於minSup的頻繁子集將會被刪掉。接著通過計算各子集之間的confidence，並和給定的minConf（min confidence）進行比較，大於minConf的子集將會組成association rule進行輸出。

* 1. **應用Fp-growth 的Apriori：**

**演算法設計：**

Step 1. 讀入input，對其進行掃描，生成1 item的頻繁子集，按照出現的次數降冪排列。如果次數一樣則按照item的字典序排列。

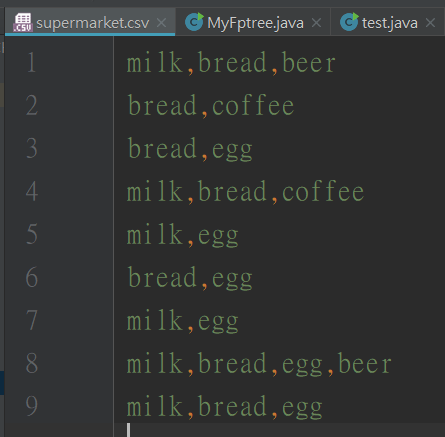
*\*為了減少演算法運行的overhead，根據Apriori定理，非頻繁項的父集一定不是頻繁集，故這裡小於minsSup（min support count）的選項，我直接將之從保存1 item頻繁子集的linked list中刪除*。

Step 2. 再次掃描input，對於出現在step 1中生成的1 item 頻繁子集的項按其順序重新排列，並建立FpTree。首先設置root節點為null，檢查該項中的元素是否有在FpTree中，若沒有則將該項加入到樹中，若已經有分支存在則判斷新的項是否跟分支中的各個點重合，若重合則只更新相應點的support，否則將創建新的點。同時這些點將會加入到Step 1生成的linked list當中去。

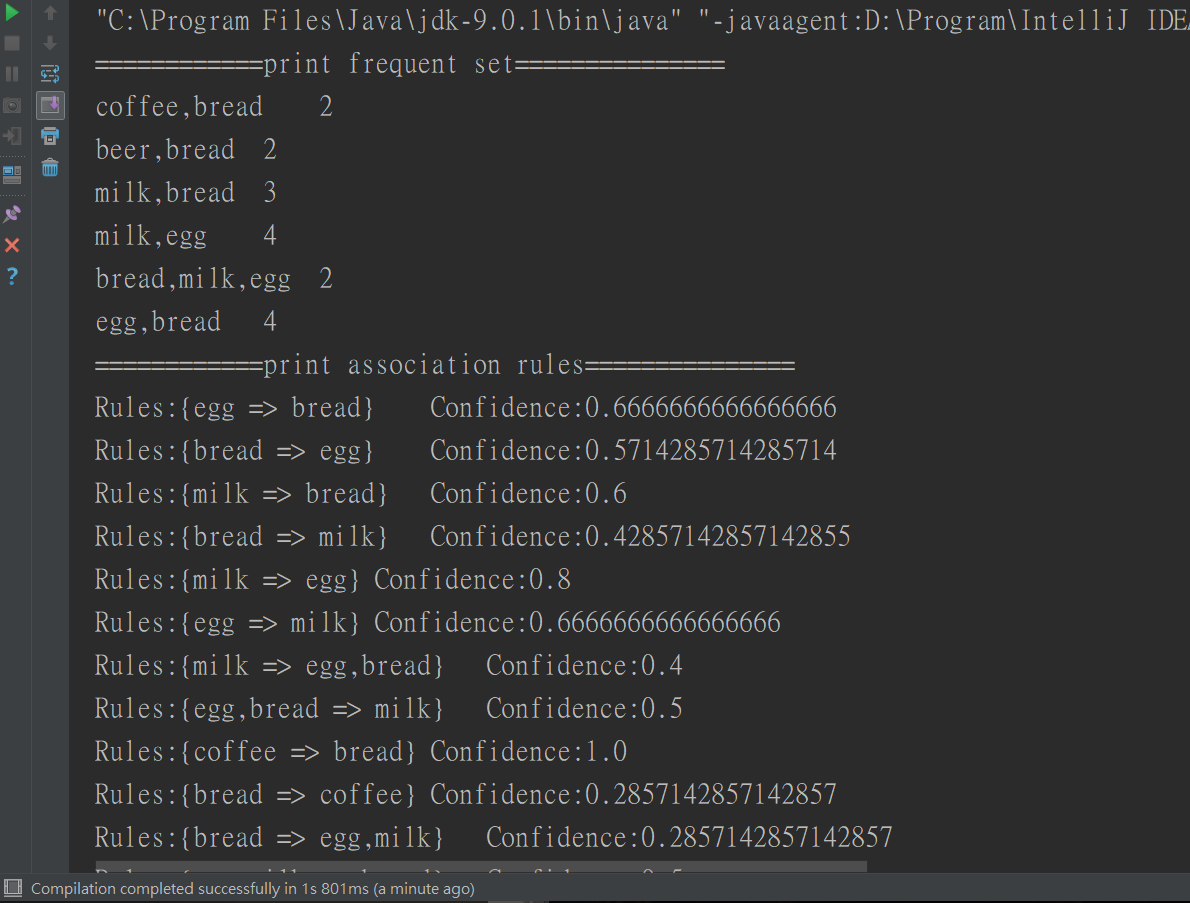
Step 3. FpTree建立之後，從尾部掃描Step1中建立的linked list，通過找尋其homonym節點和父節點，則可以生成所有的頻繁項集。

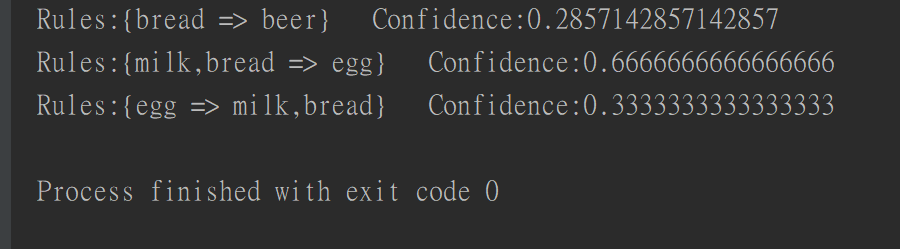
Step4. 對於每個頻繁項集分別求其proper subset（真子集，即比其小的子集合），通過計算相應的confidence，根據預先定義的minConf可以生成最後的關聯規則。

為了驗證所寫程式的正確性，我在調試的時候，以老師moodle中給的fpgrowth的投影片做input （\Apriori fp-growth\src\supermarket.csv）



將minsupport設定為3，minconfidence設定為0.25可以得到如下頻繁子集和關聯規則。





1. **實驗結果**
2. **暴力法的Apriori**

使用暴力法的Apriori實驗可以發現其耗時較長，主要的原因在於生成子集的時候各個子集之間的連接，需要反復掃描整個input會產生大量的overhead，且連接之後會產生大量的資料，根據提前給定的support和confidence進行pruning也需要對生成的結果進行掃描和操作，故time consuming和overhead都比較大。

使用IBM.data (input部分的第一種)作為input，在support count為4且minConf為0.25的時候會產生很多規則，如（\Apriori Brute\src\Apriori-output.txt）所示。通過改變不同的support count和minConf可以發現，隨著support count的升高或minConf的升高，得到的association rule會變少，當兩者一同升高的時候到一定程度可能出現沒有association rule的情況。

因為IBM.data中均為數位，故所有的input在以字元流的形式讀取之後，我有用Integer.valueOf()將所有的內容一Integer的形式進行存儲和處理。故對於kaggle中的資料進行處理之前必須將所有的內容處理成Integer的形式。

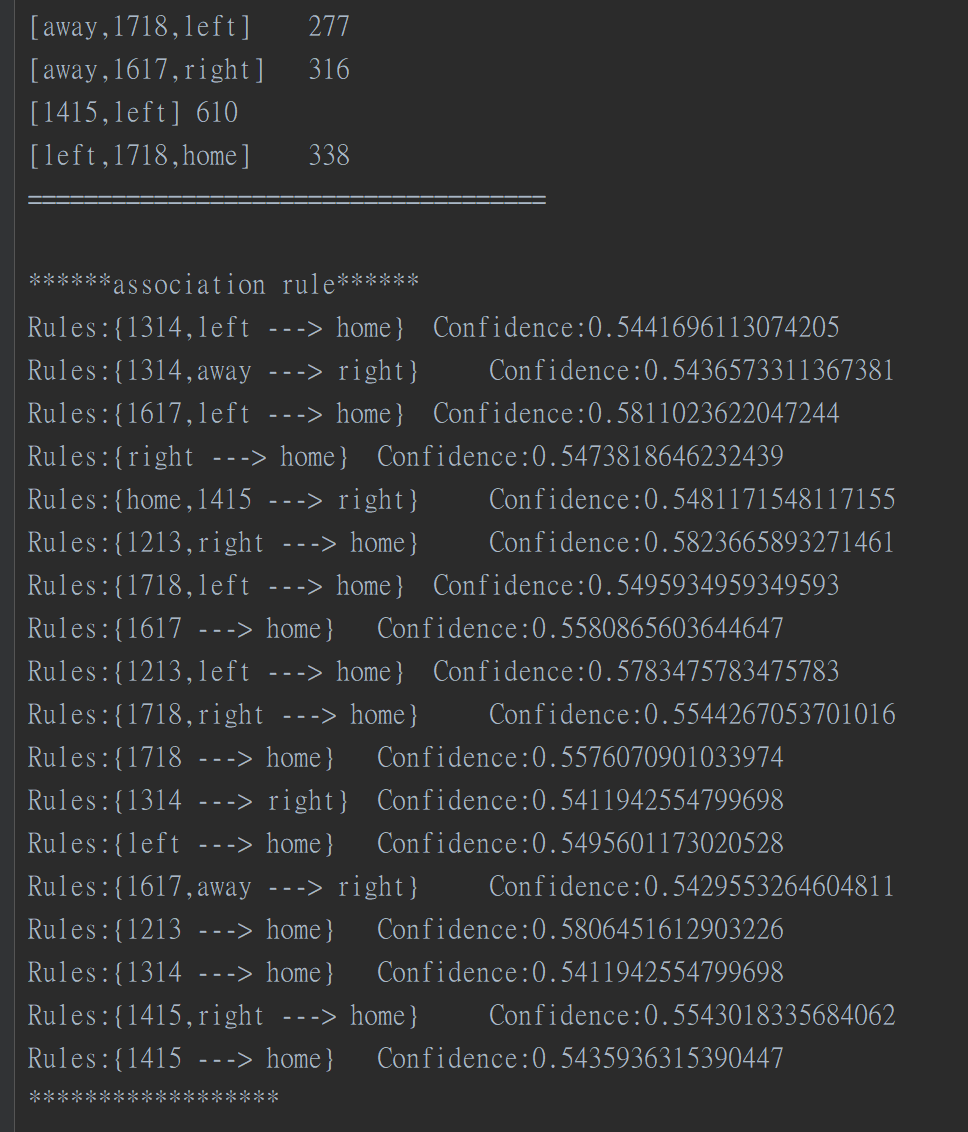
1. **使用FP-growth的Apriori**

使用FP-growth來優化演算法可以減少相應的執行時間。因為掃描整個資料集的時候就會得到各個頻繁子集，按照其出現頻率降冪排列，小於給定support的子集直接不予以考慮同時。第二次掃描會建立frequent pattern tree，通過Fp-growth進行mining考慮confidence得到相應的rule。

使用Calendar中的getTimeInMillis（）可以得到整個實驗完成的時間，單位為ms。輸出的時間表明使用FP-growth可以優化整個演算法，在相同的設定之下，其耗費的時間較少。

使用IBM.data (input部分的第一種)作為input，在support count為4且minConf為0.25的時候會產生很多規則，如（\ Apriori fp-growth \src\fptree-output.txt）所示。

以kaggle得到的資料作為input，在support count為100而minconfidence等於0.54的時候我有得到結果（\Apriori fp-growth\src\fptree-game-output.txt）。通過分析相應的association rule可知1213、1415和1718賽季home隊在Time/Score keepers 的右側其獲勝的幾率較大。而1314和1617賽季home隊在左側其獲勝的幾率較大。1516年home隊和away隊基本贏率對半分。



1. **總結**

FP-growth的奧妙在於使用簡單的tree結構來表示原有資料的資訊，其size遠小於原來的結構，通過對資料結構進行遞迴可以完整frequent pattern的mining，從而得到association rule。另外由於tree結構的size有變小，故可以放入記憶體中極大地加快了運算。如果tree過大，則可以通過切分原始資料分塊進行FP-growth在進行整合。

1. **寫作業中遇到的困難**

因為兩種演算法資料結構和思路不一樣，所以其相應的計算頻繁子集的方法也不一樣。在撰寫FP-growth時關於分析關聯性的getRelationRules函數的時候因為考慮不周到，以至於獲得相應子集之後計算其confidence時，並沒有分情況考慮單子集和多子集的confidence，故生成規則的時候有出現null pointer，耗費了一段時間debug。另外在最後結果輸出的時候因為其存儲為map<map,string>的形式，故要特別改變格式進行輸出。

這次作業的完成不僅讓我更好的理解了Apriori演算法和FP-growth演算法，更是更進一步的熟悉了對於linked list和map的使用，對於coding的能力大有幫助。同時通過找尋相應的kaggle input讓我對於association rule的使用情況有了進一步的掌握。