Data Mining Lab1(large itemset mining)

//作者: Bo-Wen Duan (Paul)

//聯絡方式: [bowenduan618@gmail.com](mailto:bowenduan618@gmail.com)

//修改日期: 2016 1/10

* 實際資料庫: retail.txt
* 輸出檔案: MiningResult.txt
* 執行檔: GenerateIargeItemSet.java
* 請將實際資料庫放置於C槽test資料夾中。
* 將資料夾”DataMininghw1”放置eclipse專案區中並匯入執行(或使用cmd編譯及執行GenerateIargeItemSet.java檔)。
* 基本概念

1. 關聯規則 A->B 的支持度 support=P(AB)，指的是事件 A和事件 B同時發生的機率。
2. 信賴度confidence=P(B|A)=P(AB)/P(A), 指的是發生事件 A的基礎上發生事件 B的機率。
3. 同時滿足最小支持度門檻值和最小信賴度門檻值的規則稱為強規則 。
4. 如果事件 A中包含 k個元素(項目)，那麼稱這個事件 A為 k-項目集， 並且事件 A滿足最小支持度門檻值的事件稱為高頻k-項目集 。

* 探勘過程：

1. 找出所有的高頻項目集；
2. 由高頻項目集產生強規則。

Apriori演算法採用"合併步驟"和"計算支持度之前判斷本身的子集合是否高頻"兩種方式來找出所有的高頻項目集。

* Apriori**介紹**

**Apriori** 演算法使用高頻項目集的先驗知識，使用逐層搜索的迭代方法， k-項目集用於探索 (k+1)-項目集。

1. 通過掃描交易記錄，找出所有的高頻 1-項集，該集合記做 L1 。
2. 利用 L1 找高頻2-項集的集合 L2， L2 找 L3 ，如此下去，直到不能再找到任何高頻 k-項集。 最後再在所有的高頻項目集中找出強規則，即產生原始資料庫擁有者所感興趣的關聯規則。

**Apriori** 演算法具有一個**性質** ：任一高頻項目集的所有非空子集合(子項目集)也必須是高頻的。 因為假如 P(I) < 最小支持度門檻值，當有元素(項目) A 添加到 I 中時，結果項目集(A∩I)不可能比 I 出現次數更多。 因此(A∩I)也不是高頻的。

* "合併步驟"和"計算支持度之前判斷本身的子集合是否高頻"

**Apriori** 算法採用"合併步驟"和"計算支持度之前判斷本身的子集合是否高頻"兩種方式來找出所有的頻繁項集。

1. "合併步驟"

為找出 Lk （所有的高頻 k-項目集的集合），通過將 Lk-1 （所有的高頻 k-1 項目集的集合）與自身合併產生候選k-項目集的集合。 候選集合記作 Ck 。 假設 l1 和 l2 是 Lk-1 中的成員。 記 li[ j ] 表示 li 中的第 j 項。 假設 Apriori 演算法對交易或項目集中的項目為自然排序，即對於（k-1）-項目集 li ， li[1] < li[2] <…< li[k-1 ] 。 將 L k-1 與自身連接，如果 (l1[1] = l2[1]) && ( l1[2] = l2[2])&&…&& (l1[k-2 ] = l2[k-2])&&(l1[k-1] < l2[k-1]) ，那認為 l1和 l2 是可連接。 連接 l1 和 l2 產生的結果是 {l1[1],l1[2],…,l1[k-1],l2[k-1]} 。

1. "計算支持度之前判斷本身的子集合是否高頻"

CK 是 LK 的超集合，也就是說， CK 的成員可能是也可能不是高頻的。 通過掃描所有的交易，確定CK 中每個候選的計數，判斷是否小於最小支持度，如果不是，則認為該候選是高頻的。 為了減少Ck數量，可以利用 Apriori 性質：任一高頻項目集的所有非空子集合也必須是高頻的，反之，如果某個候選的非空子集合不是高頻的，那麼該候選肯定不是高頻的，從而可以將其從 CK 中刪除。

* Apriori程式碼(GernateIargeItemSet.java)

程式主要結構:

* + Import檔: java.io.\*

java.util.\*

* + 類別: GernateIargeItemSet
  + 靜態變數: transactions //存放所有的交易資料

support //支持個數

confidence //信賴度

InputFileName //輸入檔名

OutputFileName //輸出檔名

* + 函式: main() //程式進入點

getfrequent1Item() //得到所有1-高頻項目集

getfrequentItemSet() //得到所有高頻項目集

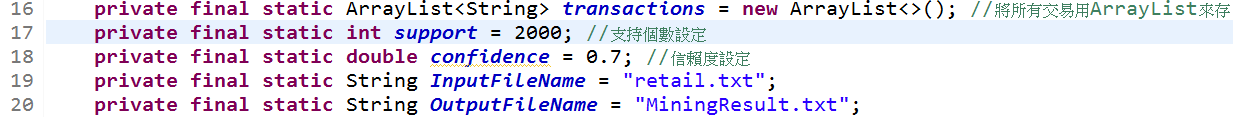
getCandidateItemSet() //得到所有候選項目集

sop() //列印方法

* Import檔
* java.io.\*: 檔案IO所需的套件包。
* java.util.\*: 集合所需的套件包。

(可視需求import會用到的子類別進來就好)

* 靜態變數



* 第16行: private final static ArrayList<String> **transactions** = new ArrayList<>();

宣告一個ArrayList泛型為String，裡面每一格儲存一行的原始交易資料。

* 第17行: private final static int **support** = 500;

設置Apriori所需整數型態的支持個數。(依使用者所需可做修改)

* 第18行: private final static double **confidence** = 0.7;

設置Apriori所需double型態的信賴度。(依使用者所需可做修改)

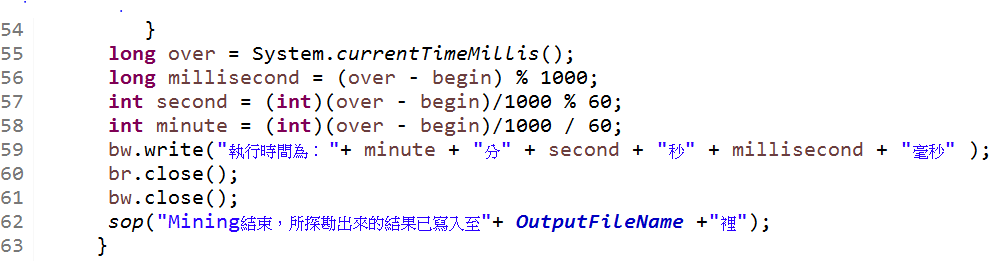
* 第19行: private final static String **InputFileName** = "retail.txt";

設置要輸入的檔案名稱。

* 第20行: private final static String **OutputFileName** = "MiningResult.txt";

設置Mining的結果要寫出的檔案名稱。

* 函式
* main()



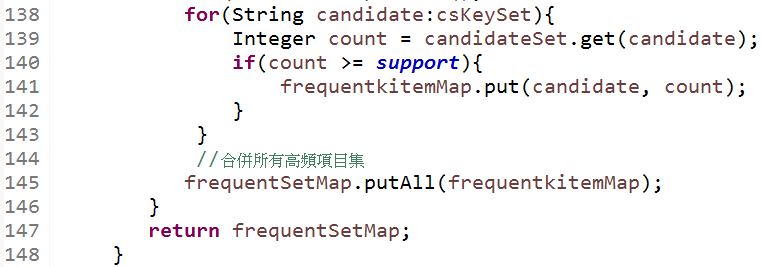
1. 第21行: 程式的進入點，throws IOException為輸入輸出丟出例外。
2. 第22行: 設置程式開始進行的時間。
3. 第23~26行: 宣告讀寫檔案以及緩衝讀寫的變數，處理檔案輸入輸出。
4. 第27行: 宣告String型態變數line接收原始資料(一行一行輸入)。
5. 第28行: 宣告int型態變數transactionscount計算原始資料的行數(交易筆數)。
6. 第29~32行: 使用while迴圈將每一行交易存入transactions(ArrayList)。
7. 第33~34行: 將設定的支持個數和交易筆總數寫出檔案。
8. 第35: 宣告一個HashMap(frequentMap)接收getfrequentItemSet()所回傳的HashMap。
9. 第36~37行: 用ArrayList(list\_Tran)儲存frequentMap.entrySet()的Map.Entry<String, Integer>。
10. 第38~48行: 覆寫Comparator(先比較長度再比較字串內容)，之後用Collections.sort()將list\_Tran作排序。
11. 第49~54行: 使用for迴圈宣告一個Map.Entry<String, Integer>(entry)接收list\_Tran，每一個進來再使用entry.getKey()(印出項目集)entry.getValue()(印出項目集支持個數)，一併寫出檔案。
12. 第55行: 設置程式結束時間。
13. 第56~58行: 處理程式進行時間換算。
14. 第59行: 程式Mining時間寫出檔案。
15. 第60~61行: 寫入和寫出的檔關閉。
16. 第62行: 印出Mining結束的訊息。

* getfrequent1Item()



1. 第79行: getfrequent1Item()為一個函式，回傳型態為HashMap(所有的高頻1-項目集)。
2. 第80~81行: 宣告兩個HashMap(sfrequent1ItemMap, rfrequent1ItemMap)。
3. 第82~94行: 將已經存入transactions裡面所有的交易，一個一個(也就是一行一行的交易)，宣告一個String型態的trans來接收。再來將trans(一行交易)進行切割(切成一個個的項目)。切割完之後對於每一個項目當成key，value當作這個項目的支持個數。宣告一個Integer的count接收每一個項目(key)所對應的value值，若為null，count設為1，若非null則所取出的count(value)再加1。形成最後的sfrequent1ItemMap(相異的單一項目,項目的支持個數)。
4. 第95~101行: 宣告Set<String>型態的keySet將sfrequent1ItemMap的所有key取出來。利用for-each迴圈宣告一個String型態key將keySet裡面的key一個一個取進去。利用整數型態count & sfrequent1ItemMap.get(key)以及support，if迴圈來判斷是否高頻(count大於等於support)，若大於等於將此(key,value)put進rfrequent1ItemMap。
5. 第102行: 回傳rfrequent1ItemMap。

* getfrequentItemSet()



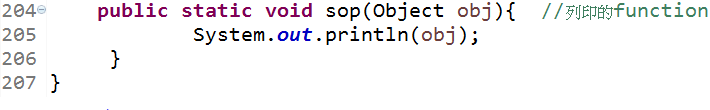
1. 第109行: getfrequentItemSet()為一個函式，回傳型態為HashMap(所有的高頻項目集)。
2. 第110~111行: 宣告兩個HashMap(frequentSetMap, frequentkitemMap)。
3. 第112~113行: 將getfrequent1Item()所回傳的HashMap分別放入frequentSetMap和frequentkitemMap中。
4. 第114~144行: 為一個while迴圈，判斷為true的條件是當frequentkitemMap不等於null以及size不為0(Apriori演算法中迭代從合併兩個高頻k-1項目集來產生一個候選k項目集，直到不能產生任何候選k-項目集為止)。
5. 第115行: 將frequentkitemMap帶入getCandidateItemSet()計算，所回傳的HashMap宣告一個HashMap(candidateSet)來接收。
6. 第116行: 宣告一個Set<String>(csKeySet)來接收candidateSet所取出的所有key(項目集)。
7. 第118~135行: 三個for迴圈第一個for迴圈宣告一個String變數(trans)將ArrayList(transactions)一個一個(一行一行交易)讀進來，讀進來的每一行交易進行切割存成陣列，再將陣列轉成List集合。第二個for迴圈宣告一個String變數(candidate)將csKeySet一個一個(一個一個項目集)讀進來。宣告一個boolean型態變數(flag)預設true。宣告一個String陣列candidateItems將candidate(每一個項目集)切割(形成一個一個項目)放進來。第三個for迴圈宣告一個String變數candidateItem將candidateItems一個一個放進來。利用if迴圈對項目集中每個項目檢查List集合的索引，只要出現-1，flag改為false然後break出第三個for迴圈，繼續下一個項目集比對，若此項目集的所有項目比對成功則跑下面if迴圈(此時flag為true)。宣告一個Integer的變數count接收candidateSet以此項目集為key回傳value，然後count+1再put回去。然後持續執行將所有交易比對完(相對於Apriori將候選項目集拿去跟原始交易資料掃描)。
8. 第137行: frequentkitemMap清空。
9. 第138~148行: 一個for迴圈中宣告一個String變數(candidate) 將csKeySet一個一個(一個一個項目集)讀進來。宣告一個Integer的變數count接收candidateSet以此項目集為key回傳value。If迴圈將每一個value與support做比較若value大於等於support將此項目集(key)與value放置frequentkitemMap。
10. 第145行: 將frequentkitemMap放置到frequentSetMap裡。
11. 第147行: 回傳frequentSetMap。

* getCandidateItemSet()



1. 第152行: getCandidateItemSet()為一個函式，回傳型態為HashMap(所有的候選項目集)。
2. 第153行: 宣告一個HashMap(candidateSet)。
3. 第154~155行: 宣告兩個Set<String>(kitemSet1,kitemSet2)來接收frequentkitemMap所取出的所有key(項目集)。
4. 第156~201行: 兩個for迴圈第一個for迴圈宣告一個String變數(kitem1)將kitemSet1一個一個(一個一個項目集)讀進來。第二個for迴圈宣告一個String變數(kitem2)將kitemSet2一個一個(一個一個項目集)讀進來。宣告兩個String陣列(tmp1, tmp2)將kitem1, kitem2切割後分別放入陣列中。宣告一個String變數c。if迴圈若temp1陣列長度為1比較temp1[0]和temp2[0]的元素，值若小於c為temp1[0]+ temp[0]。(到此c為所有的候選2-項目集)。else，宣告一個boolean變數flag預設為true。temp1和temp2陣列前面n-1個元素比對若有不相同的break迴圈kitemSet2下一個項目集進來切割比對。若temp1和temp2陣列前面n-1個元素比對都相同時比較兩個陣列最後一個元素，若值小於0同時flag為true，此時的c為kitem1+temp2陣列的最後一個元素。(候選k-項目集，k>2)。最後所產生的c要檢查c的子集合(子項目集)是否都為高頻。宣告一個boolean變數hasInfrequentSubSet預設為false。If c不為空，宣告一個String陣列tempC將C切割後放入，利用兩個for迴圈以及tempC的index產生所有c的子集合。將所有的子集合當作key放入frequentkitemMap回傳value，若value為null則hasInfrequentSubSet改為true然後break，若不為null則將c(key)以及value 0 放置candidateSet。
5. 第202行: 回傳candidateSet。

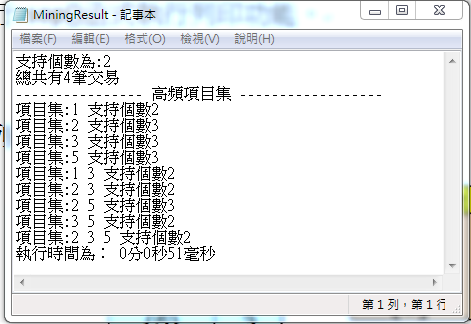
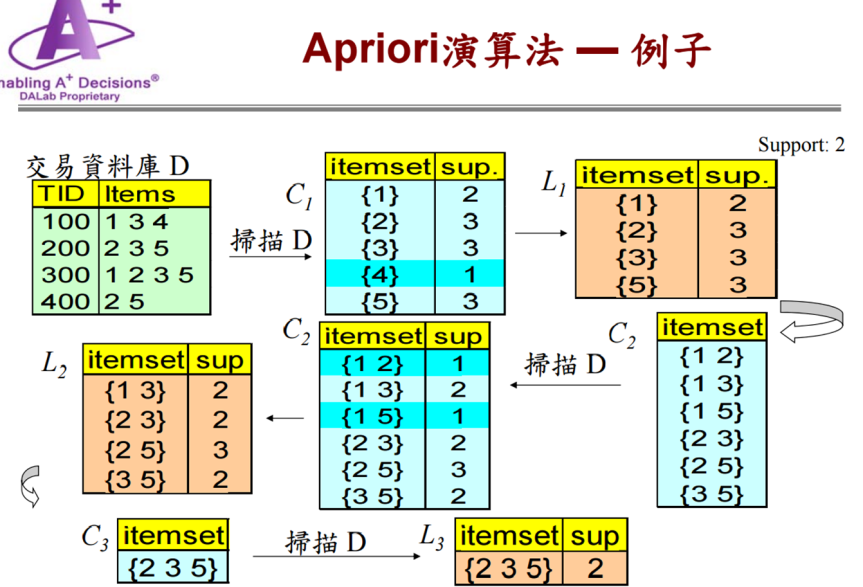
* sop()



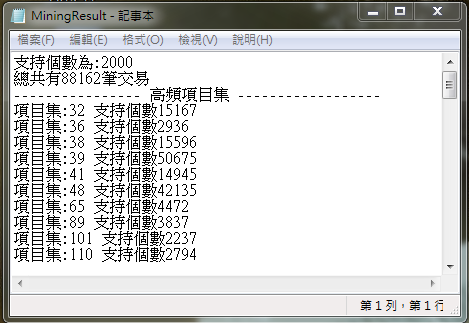
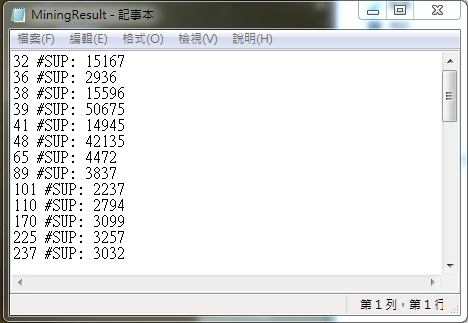
1. 第204~207行: sop()函式執行列印功能。

* 執行結果

1. 執行課本範例(支持個數設2)



1. 執行retail.txt(支持個數設500)左圖為spmf軟體執行結果，右圖為此程式執行結果。



* 執行效能(使用此程式Mining retail資料庫)
* 支持個數設2000執行時間：0分4秒302毫秒
* 支持個數設1000執行時間：0分49秒435毫秒
* 支持個數設500執行時間：11分5秒95毫秒