Data Mining Project 1 Report

資工碩一 P76074240 蔡文傑

一、摘要

本篇報告分別實作 Apriori 和 FP-growth 兩種分析關聯法則的演算法，將這兩種方法分別應用於 IBM Quest Synthetic Data Generator[1]產生出的交易資料與 Kaggle 網站提供的商店交易資料[2]，目的是從大量交易資料(Transaction)找出 Frequent Item Set 與 Rule。接著本篇報告會分析兩著演算法效能差異，找出演算法中最小支持度門檻(Minima Support Threshold)參數對結果與執行時間的影響。此外，本篇報告更進一步地透過調整 IBM Quest Synthetic Data Generator 中產生交易資料(Transaction)的參數，如：Transaction 個數、平均每個 Transaction 之中的 Item 個數與總共的 Item 個數來觀察演算法表現結果。最後本篇報告會將結果與現成的 Python FP-growth 套件[3]進行比較與分析。

[1] IBM Quest Synthetic Data Generator, https://sourceforge.net/projects/ibmquestdatagen/

[2] Retailrocket recommender system dataset, https://www.kaggle.com/retailrocket/ecommerce-dataset#events.csv

[3] Pyfpgrowth, https://pypi.org/project/pyfpgrowth/

二、資料集說明

1. IBM Quest Synthetic Data Generator[1]

此為 IBM 設計的資料產生器，執行結果如下：

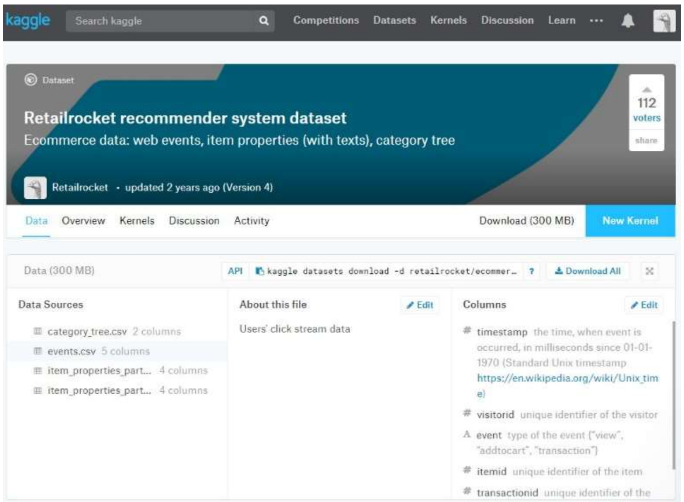
```
C:\ >"IBM Quest Data Generator.exe" lit -help
Command Line Options:
   -ntrans number_of_transactions_in_000s (default: 1000)
   -tlen avg_items_per_transaction (default: 10)
   -nitems number_of_different_items_in_000s) (default: 100)

   -npats number_of_patterns (default: 10000)
```

```
   -patlen avg_length_of_maximal_pattern (default: 4)

   -corr correlation_between_patterns (default: 0.25)

   -conf avg_confidence_in_a_rule (default: 0.75)


   -fname <filename> (write to filename.data and filename.pat)

   -ascii (default: True)

   -randseed # (reset seed used generate to x-acts; must be negative)

   -version (to print out version info)


C:\ >
```

在本篇報告中會試著調整其-ntrans、-tlen 與-nitems 以產生不同特性的資料集做後續分析。

2. Retailrocket recommender system dataset[2]

此為 Retail Rocket (retailrocket.io)網站上顧客行為紀錄的資料集，在本篇報告中只使用了其中的(events.csv)檔案。



此檔案紀錄了 2015-05-03 至 2015-09-18 期間 1407580 位顧客(viewid)瀏覽(view)與購買(addcard)某個物品的行為，本篇報告中透過顧客(viewid)與日期對此資料集及做分群，整理出每一天顧客瀏覽與購買的 Transaction 紀錄，共 235061 筆 Transaction 資料。

我們的轉換過程可見 parse_kaggle_dataset.html

三、實驗設計

1. 實作 Apriori 演算法。
2. 實作 FP-growth 演算法。
3. 將 Apriori 演算法、FP-growth 演算法與並且與 Pyfpgrowth 套件應用於 IBM Quest Synthetic Data Generator 產生的資料集。
4. 將 Apriori 演算法、FP-growth 演算法與並且與 Pyfpgrowth 套件應用於 Retailrocket recommender system dataset。

四、實驗結果

1. 程式碼請見 apriori.py

```
(base) C:\Dropbox\NCKU\2 資料探勘\hw1\hw1_release>python apriori.py

------------------------ Transations:
[array([0, 4, 5, 6, 7, 8, 9], dtype=int64), array([0, 1, 3, 4, 5, 6, 7, 8, 9], dtype=int64),
array([2, 3, 5, 6, 7, 9], dtype=int64), array([0, 1, 3, 4, 5, 6, 7, 8, 9], dtype=int64),
array([3, 4, 5, 6, 7, 8, 9], dtype=int64), array([0, 4, 5, 6, 7, 8, 9], dtype=int64), array([0,
1, 3, 4, 5, 6, 8, 9], dtype=int64), array([3, 4, 5, 6, 7, 8, 9], dtype=int64), array([0, 3, 5,
6, 7, 8, 9], dtype=int64), array([0, 1, 3, 4, 5, 6, 7, 8, 9], dtype=int64)]



------------------------ Frequent Item Set:
item: (7,) , 9.000
item: (8,) , 9.000
item: (5, 7) , 9.000
item: (6, 7) , 9.000
item: (8, 9) , 9.000
item: (8, 5) , 9.000
item: (8, 6) , 9.000
item: (9, 7) , 9.000
item: (8, 5, 6) , 9.000
item: (9, 5, 7) , 9.000
item: (9, 6, 7) , 9.000
item: (5, 6, 7) , 9.000
```

```
item: (8, 9, 5) , 9.000
item: (8, 9, 6) , 9.000
item: (5, 6, 8, 9) , 9.000
item: (5, 6, 7, 9) , 9.000
item: (5,) , 10.000
item: (6,) , 10.000
item: (9,) , 10.000
item: (5, 6) , 10.000
item: (9, 6) , 10.000
item: (9, 5) , 10.000
item: (9, 5, 6) , 10.000

------------------------ Rules:
Rule: (5,) ==> (7,) , 0.900
Rule: (6,) ==> (7,) , 0.900
Rule: (9,) ==> (8,) , 0.900
Rule: (5,) ==> (8,) , 0.900
Rule: (6,) ==> (8,) , 0.900
Rule: (9,) ==> (7,) , 0.900
Rule: (5,) ==> (8, 6) , 0.900
Rule: (6,) ==> (8, 5) , 0.900
Rule: (5, 6) ==> (8,) , 0.900
Rule: (9,) ==> (5, 7) , 0.900
Rule: (5,) ==> (9, 7) , 0.900
Rule: (9, 5) ==> (7,) , 0.900
Rule: (9,) ==> (6, 7) , 0.900
Rule: (6,) ==> (9, 7) , 0.900
Rule: (9, 6) ==> (7,) , 0.900
Rule: (5,) ==> (6, 7) , 0.900
Rule: (6,) ==> (5, 7) , 0.900
Rule: (5, 6) ==> (7,) , 0.900
Rule: (9,) ==> (8, 5) , 0.900
Rule: (5,) ==> (8, 9) , 0.900
Rule: (9, 5) ==> (8,) , 0.900
Rule: (9,) ==> (8, 6) , 0.900
Rule: (6,) ==> (8, 9) , 0.900
Rule: (9, 6) ==> (8,) , 0.900
Rule: (5,) ==> (8, 9, 6) , 0.900
```

```
Rule: (6,) ==> (8, 9, 5) , 0.900
Rule: (9,) ==> (8, 5, 6) , 0.900
Rule: (5, 6) ==> (8, 9) , 0.900
Rule: (9, 5) ==> (8, 6) , 0.900
Rule: (9, 6) ==> (8, 5) , 0.900
Rule: (9, 5, 6) ==> (8,) , 0.900
Rule: (5,) ==> (9, 6, 7) , 0.900
Rule: (6,) ==> (9, 5, 7) , 0.900
Rule: (9,) ==> (5, 6, 7) , 0.900
Rule: (5, 6) ==> (9, 7) , 0.900
Rule: (9, 5) ==> (6, 7) , 0.900
Rule: (9, 6) ==> (5, 7) , 0.900
Rule: (9, 5, 6) ==> (7,) , 0.900
Rule: (7,) ==> (5,) , 1.000
Rule: (5,) ==> (6,) , 1.000
Rule: (6,) ==> (5,) , 1.000
Rule: (7,) ==> (6,) , 1.000
Rule: (9,) ==> (6,) , 1.000
Rule: (6,) ==> (9,) , 1.000
Rule: (8,) ==> (9,) , 1.000
Rule: (9,) ==> (5,) , 1.000
Rule: (5,) ==> (9,) , 1.000
Rule: (8,) ==> (5,) , 1.000
Rule: (8,) ==> (6,) , 1.000
Rule: (7,) ==> (9,) , 1.000
Rule: (9,) ==> (5, 6) , 1.000
Rule: (5,) ==> (9, 6) , 1.000
Rule: (6,) ==> (9, 5) , 1.000
Rule: (9, 5) ==> (6,) , 1.000
Rule: (9, 6) ==> (5,) , 1.000
Rule: (5, 6) ==> (9,) , 1.000
Rule: (8,) ==> (5, 6) , 1.000
Rule: (8, 5) ==> (6,) , 1.000
Rule: (8, 6) ==> (5,) , 1.000
Rule: (7,) ==> (9, 5) , 1.000
Rule: (9, 7) ==> (5,) , 1.000
Rule: (5, 7) ==> (9,) , 1.000
Rule: (7,) ==> (9, 6) , 1.000
```

```
Rule: (9, 7) ==> (6,) , 1.000
Rule: (6, 7) ==> (9,) , 1.000
Rule: (7,) ==> (5, 6) , 1.000
Rule: (5, 7) ==> (6,) , 1.000
Rule: (6, 7) ==> (5,) , 1.000
Rule: (8,) ==> (9, 5) , 1.000
Rule: (8, 9) ==> (5,) , 1.000
Rule: (8, 5) ==> (9,) , 1.000
Rule: (8,) ==> (9, 6) , 1.000
Rule: (8, 9) ==> (6,) , 1.000
Rule: (8, 6) ==> (9,) , 1.000
Rule: (8,) ==> (9, 5, 6) , 1.000
Rule: (8, 5) ==> (9, 6) , 1.000
Rule: (8, 6) ==> (9, 5) , 1.000
Rule: (8, 9) ==> (5, 6) , 1.000
Rule: (8, 5, 6) ==> (9,) , 1.000
Rule: (8, 9, 5) ==> (6,) , 1.000
Rule: (8, 9, 6) ==> (5,) , 1.000
Rule: (7,) ==> (9, 5, 6) , 1.000
Rule: (5, 7) ==> (9, 6) , 1.000
Rule: (6, 7) ==> (9, 5) , 1.000
Rule: (9, 7) ==> (5, 6) , 1.000
Rule: (5, 6, 7) ==> (9,) , 1.000
Rule: (9, 5, 7) ==> (6,) , 1.000
Rule: (9, 6, 7) ==> (5,) , 1.000

(base) C:\Dropbox\NCKU\2 資料探勘\hw1\hw1_release>
```

2. 程式碼請見 fpgrowph.py

```
(base) C:\Dropbox\NCKU\2 資料探勘\hw1\hw1_release>python fpgrowph.py


------------------------ Transations:
[array([0, 4, 5, 6, 7, 8, 9], dtype=int64), array([0, 1, 3, 4, 5, 6, 7, 8, 9], dtype=int64), array([2, 3, 5, 6, 7, 9], dtype=int64), array([0, 1, 3, 4, 5, 6, 7, 8, 9], dtype=int64), array([3, 4, 5, 6, 7, 8, 9], dtype=int64), array([0, 4, 5, 6, 7, 8, 9], dtype=int64), array([0, 1, 3, 4, 5, 6, 8, 9], dtype=int64), array([3, 4, 5, 6, 7, 8, 9], dtype=int64), array([0, 3, 5, 6, 7, 8, 9], dtype=int64), array([0, 1, 3, 4, 5, 6, 7, 8, 9], dtype=int64)]
```

```
------------------------ Frequent Item Set:
item: (7,) , 9.000
item: (7, 9) , 9.000
item: (6, 7) , 9.000
item: (5, 7) , 9.000
item: (6, 7, 9) , 9.000
item: (5, 7, 9) , 9.000
item: (5, 6, 7) , 9.000
item: (5, 6, 7, 9) , 9.000
item: (8,) , 9.000
item: (8, 9) , 9.000
item: (6, 8) , 9.000
item: (5, 8) , 9.000
item: (6, 8, 9) , 9.000
item: (5, 8, 9) , 9.000
item: (5, 6, 8) , 9.000
item: (5, 6, 8, 9) , 9.000
item: (5,) , 10.000
item: (6,) , 10.000
item: (5, 6) , 10.000
item: (9,) , 10.000
item: (6, 9) , 10.000
item: (5, 9) , 10.000
item: (5, 6, 9) , 10.000

(base) C:\Dropbox\NCKU\2  資料探勘\hw1\hw1_release>
```

3. 將 Apriori 演算法、FP-growth 演算法與 Pyfpgrowth 套件應用於 IBM Quest Synthetic Data Generator 產生的資料集。

3.1 固定 trasation 數目(trasations = 10000)，觀察 minima support 大小與實行時間關係，執行時間比較(單位：秒)：

	Apriori	FP-growth	Pyfpgrowth
Mim.sup=900	0.8226443487219512	0.05219484306871891	0.0365846180357039
Mim.sup=300	1.7990779876708984	0.13667751010507345	0.1289132465608418
Mim.sup=100	58.27382839983329	2.850673302076757	1.0344123682007194

3.2　固定 minima support 大小(minima support=300)，觀察 trasation 數目與實行時間關係，執行時間比較(單位：秒)：

	Apriori	FP-growth	Pyfpgrowth
Transaction=10000	1.8170774690806866	0.12296994123607874	0.15499818697571754
Transaction=5000	0.6400609510019422	0.01945443032309413	0.019474192056804895
Transaction=1000	0.15733187785372138	0.0036582970060408115	0.003650657832622528

詳細程式執行過程請見 experiment_3.html。

4. 將 Apriori 演算法、FP-growth 演算法與 Pyfpgrowth 套件應用於 Retailrocket recommender system dataset。

執行時間比較(單位：秒)：

	Apriori	FP-growth	Pyfpgrowth
Mim.sup=900	5.1547880987636745	0.018718854058533907	0.017675184179097414
Mim.sup=300	5.480314610991627	0.0170785491354 7635	0.0252280761487782
Mim.sup=100	5.443474090192467	0.02188724000006914	0.02464825380593 5383

詳細程式執行過程請見 experiment_4.html。

五、討論與結論

1. 在本篇報告中，我們實作分別實作出 Apriori 演算法與 FP-growth 演算法，從程式執行時間比較可以看出，無論是調整 minima support 還是 Transaction 個數，FP-growth 演算法總是會有比較好的效率，可以在比較短的時間內執行完成。
2. 當使用本篇報告實作的 FP-growth 演算法與 Pyfpgrowth 套件進行比較可以發現執行時間相當接近，尤其是在測量改變 Transaction 個數運行時間的時候。但是在測量改變 minima support 時會發現比 Pyfpgrowth 套件稍微慢一點。
3. 不論從 IBM Quest Synthetic Data Generator 產生的資料集還是 Retailrocket recommender system dataset 資料集中，都可以觀察到當 minima support 設定的越高時，程式執行時間也就會越快。
4. 執行時間會隨著 Transaction 數量增加而需要更多的運算時間。