TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM
KHOA CÔNG NGHỆ THÔNG TIN
Môn học: Khai thác dữ liệu và ứng dụng

Project Report

(Lab 02 - FIM)

Giảng viên hướng dẫn: Lê Hoài Bắc
Nguyễn Ngọc Đức
Nguyễn Khánh Toàn
Lê Minh Nhật
Sinh viên thực hiện: Nguyễn Văn Hậu (1712258)

TP. Hồ Chí Minh, ngày 06 tháng 12 năm 2021

Mục lục

1 Thông tin chung:

1.1 Thông tin sinh viên:

STT	Họ tên	MSSV	Nội dung công việc	Mức độ hoàn thành
1	Nguyễn Văn Hậu	1712258	Cài đặt 2 thuật toán, ứng dụng khai thác tập phổ biến và báo cáo.	

1.2 Check-list các yêu cầu của đồ án:

Yêu cầu	Tiến độ (Chưa hoàn thành/Đang tiến hành/Hoàn thành)	Note
1. Cài đặt thuật toán Apriori	Hoàn thành	
2. Cài đặt thuật toán Tree Projection	Chưa hoàn thành	
3. Mô tả thuật toán Apriori	Hoàn thành	
4. Mô tả thuật toán Tree Projection		
5. Ứng dụng khai thác dữ liệu (Apriori Algorithm)	Hoàn thành	
6. Ứng dụng khai thác dữ liệu (Tree Projection Algorithm)		
7. Báo cáo	Hoàn thành	

2 Association Rule Mining:

2.1 Giới thiệu:

- Khai thác luật kết hợp là một phần quan trọng trong Data mining. Khai thác luật kết hợp thường dùng để tìm tập phổ biến, luật kết hợp, mối tương quan cấu trúc giữa các tập items hoặc objects.
- Trong báo cáo này đề cập đến:
  - Frequent Itemset Generation:
    - Apriori Algorithm
    - Tree Projection
  - Rule Generation

2.2 Frequent Item Set:

- Là tập các item phổ biến xuất hiện cùng nhau và đáp ứng được ngưỡng minimum support và ngưỡng minimum confidence.

2.3 Support:

- Là số phần trăm trong một itemset của transaction, trong đó tất cả các item trong nhóm đó được sắp xếp cùng nhau.
- Công thức tính:
  Support(A, B) = Probability(A U B)

2.4 Confidence:

- Quy tắc X=> Y có độ tin cậy C với C% của transaction, tức là mỗi item có X thì cũng có Y.
- Công thức tính:
  Conficence(X => Y) = Probability(A U B) / Probability(A)

2.5 Rule Generation:

- Với ngưỡng độ tin cậy cho trước, luật kết hợp được tạo ra bằng cách duyệt qua các tập phổ biến có thể có, tính toán độ tin cậy và lược bỏ theo ngưỡng.

3 Apriori Algorithm:

3.1 Ý tưởng thuật toán:

- Apriori Algorithm là thuật toán cơ bản dùng để khai thác dữ liệu tìm ra các luật kết hợp, tập phổ biến.
- Nguyên tắc của thuật toán Aprori: Tất cả các tập con không rỗng của một tập phổ biến cũng phải phổ biến.

3.2 Các bước thực hiện của thuật toán:

- Bước 0:
  - Từ số phần trăm minSup suy ra con số support threshold.
  - Duyệt qua Data lần đầu tiên đếm số transaction lưu vào C1, từ đó tìm ra L1 frequent 1-itemsets đạt ngưỡng minSup đã cho.
- Bước 1:
  - Phát sinh C(k+1) từ L(k)
  - Cắt tỉa C(k+1), nếu như tất cả các subset trong C(k+1) không có trong tập phổ biến trước đó L(k) thì cắt ra khỏi C(k+1).
  - Kiểm tra nếu như C(k+1) rỗng, tức là không có ứng viên nào được sinh ra, kết thúc thuật toán.
- Bước 2:
  - Duyệt qua Data, đếm tần suất của các item có trong C(k+1)
- Bước 3:
  - Với những item thỏa mãn minSup, ta lưu vào L(k+1) tức là tập phổ biến tiếp theo.
- Bước 4:
  - Lặp lại bước 1
  - Kết quả trả về là tập phổ biến trong L(k+1) hoặc không có tập phổ biến.

3.3 Ưu - nhược điểm của thuật toán:

- Ưu điểm:
  - Thuật toán đơn giản, dễ hiểu và dễ cài đặt.
- Nhược điểm:
  - Phải duyệt qua Data nhiều lần, gặp vấn đề lớn nếu như xử lý trên tập dữ liệu lớn.
  - Số lượng tập ứng viên sinh ra rất lớn, yêu cầu về không gian lưu trữ lớn.
  - Thực hiện việc tính độ phổ biến nhiều lần, lặp lại.

3.4 Ứng dụng trong khai thác dữ liệu:

3.4.1 Lưu ý về chương trình cài đặt:

- Giá trị trả về của hàm Apriori Algorithm có gồm hai tham số: Số lần xuất hiện của từng item L, và tập phổ biến Lk. Hai tham số này đều có thể truyền trực tiếp vào hàm Rule Generation để tính toán, nhưng do hạn chế hiểu biết về ngôn ngữ Julia, em chưa xử lý được giữa keys của 2 với keys của [2], thế nên truyền vào Data để đếm lại số lần xuất hiện của mỗi item.
- Trong cài đặt thuật toán này, em xin phép được dùng hai biến Ck, Lk (tương đương với cấp k) và Ck1, Lk1 (tương đương với cấp k+1) thay vì dùng một Array lưu trữ tất cả các Ck, Lk sinh ra tương ứng k=2 → n.
- Biến L trong thuật toán nhằm lưu lại số lần suất hiện của các item có trong tập phổ biến.
- Một số hàm hỗ trợ là:
  - **read_file**(): Hàm hỗ trợ đọc file txt và tải dữ liệu đầu vào chương trình.
  - **subset**(): Hàm tìm các sub itemset dưới 1 cấp.
    - VD: subset([1, 2, 3]) = [1, 2], [1, 3], [2, 3]
  - **powerset**(): Hàm tìm tất cả các sub itemset.
    - VD: powerset([1, 2, 3]) = [], [1], [2], [3], [1, 2], [1, 3], [2, 3], [1, 2, 3]

3.4.2 Khai thác tập phổ biến *chess*:

- Thông tin về tập dữ liệu:
  - Transaction count: 3 196
  - Item count: 75
  - Average item count per transaction: 37
  - Has item name: No
- Khai thác dữ liệu: Với minSup = 95, minConf = 80
  - Tập phổ biến:

Items	Support
[29, 36, 40, 52, 58]	3046
[29, 40, 52, 58, 60]	3099

- Luật kết hợp:

Association Rule	Confidence
[29] ==> [36]	96.95%
[36] ==> [29]	99.52%
[36] ==> [29, 36]	99.52%
[29, 36] ==> [36]	100.0%
[29, 36] ==> [40]	99.16%
[40] ==> [29, 36]	96.47%
[40] ==> [29, 40]	99.53%
[29, 40] ==> [40]	100.0%
[29, 40] ==> [36, 40]	96.93%
[36, 40] ==> [29, 40]	99.51%
[36, 40] ==> [29, 36, 40]	99.51%
[29, 36, 40] ==> [36, 40]	100.0%
[29, 36, 40] ==> [52]	99.64%
[52] ==> [29, 36, 40]	95.67%
[52] ==> [29, 52]	99.53%
[29, 52] ==> [52]	100.0%
[29, 52] ==> [36, 52]	96.94%
[36, 52] ==> [29, 52]	99.51%
[36, 52] ==> [29, 36, 52]	99.51%
[29, 36, 52] ==> [36, 52]	100.0%
[29, 36, 52] ==> [40, 52]	99.15%

[40, 52] ==> [29, 36, 52]	96.45%
[40, 52] ==> [29, 40, 52]	99.53%
[29, 40, 52] ==> [40, 52]	100.0%
[29, 40, 52] ==> [36, 40, 52]	96.91%
[36, 40, 52] ==> [29, 40, 52]	99.51%
[36, 40, 52] ==> [29, 36, 40, 52]	99.51%
[29, 36, 40, 52] ==> [36, 40, 52]	100.0%
[29, 36, 40, 52] ==> [58]	99.97%
[58] ==> [29, 36, 40, 52]	95.34%
[58] ==> [29, 58]	99.53%
[29, 58] ==> [58]	100.0%
[29, 58] ==> [36, 58]	96.95%
[36, 58] ==> [29, 58]	99.52%
[36, 58] ==> [29, 36, 58]	99.52%
[29, 36, 58] ==> [36, 58]	100.0%
[29, 36, 58] ==> [40, 58]	99.16%
[40, 58] ==> [29, 36, 58]	96.47%
[40, 58] ==> [29, 40, 58]	99.53%
[29, 40, 58] ==> [40, 58]	100.0%
[29, 40, 58] ==> [36, 40, 58]	96.92%
[36, 40, 58] ==> [29, 40, 58]	99.51%

[36, 40, 58] ==> [29, 36, 40, 58]	99.51%
[29, 36, 40, 58] ==> [36, 40, 58]	100.0%
[29, 36, 40, 58] ==> [52, 58]	99.64%
[52, 58] ==> [29, 36, 40, 58]	95.67%
[52, 58] ==> [29, 52, 58]	99.53%
[29, 52, 58] ==> [52, 58]	100.0%
[29, 52, 58] ==> [36, 52, 58]	96.94%
[36, 52, 58] ==> [29, 52, 58]	99.51%
[36, 52, 58] ==> [29, 36, 52, 58]	99.51%
[29, 36, 52, 58] ==> [36, 52, 58]	100.0%
[29, 36, 52, 58] ==> [40, 52, 58]	99.15%
[40, 52, 58] ==> [29, 36, 52, 58]	96.45%
[40, 52, 58] ==> [29, 40, 52, 58]	99.53%
[29, 40, 52, 58] ==> [40, 52, 58]	100.0%
[29, 40, 52, 58] ==> [36, 40, 52, 58]	96.91%
[36, 40, 52, 58] ==> [29, 40, 52, 58]	99.51%

3.4.3 Khai thác tập phổ biến *mushrooms*:

- Thông tin về tập dữ liệu:
  - Transaction count: 8 416
  - Item count: 119
  - Average item count per transaction: 23
  - Has item name: No
- Khai thác dữ liệu: (Với minSup = 95 và minConf = 80)
  - Tập phổ biến:

Items	Support
[36, 90, 94]	8192

  - Luật kết hợp:

Association Rule	Confidence
[36] ==> [90]	100.0%
[90] ==> [36]	97.43%
[90] ==> [36, 90]	97.43%
[36, 90] ==> [90]	100.0%
[36, 90] ==> [94]	99.9%
[94] ==> [36, 90]	99.71%
[94] ==> [36, 94]	99.71%
[36, 94] ==> [94]	100.0%
[36, 94] ==> [90, 94]	100.0%
[90, 94] ==> [36, 94]	99.71%

3.4.4 Khai thác tập phổ biến *retail*:

- Thông tin về tập dữ liệu:
  - Transaction count: 88 162
  - Item count: 16 470
  - Average item count per transaction: 10.30
  - Has item name: No
- Khai thác dữ liệu: Với minSup = 8 và minConf = 70
  - Tập phổ biến:

Items	Support
[40, 42, 49]	7366

  - Luật kết hợp:

Association Rule	Confidence
[42] ==> [40]	76.37%
[42] ==> [40, 42]	76.37%
[40, 42] ==> [42]	100.0%
[40, 49] ==> [49]	100.0%
[42, 49] ==> [40, 49]	81.68%

3.4.5 Khai thác tập phổ biến *foodmart*:

- Thông tin về tập dữ liệu:
  - Transaction count: 4 141
  - Item count: 1 558
  - Average item count per transaction: 4.42
  - Has item name: No
- Khai thác dữ liệu: Với minSUp = 0.1 và minConf = 40
  - Tập phổ biến:

Items	Support
(rỗng)	

Không có tập phổ biến nào được tạo ra

  - Luật kết hợp:

Association Rule	Confidence
(rỗng)	

3.4.6 Thống kê:

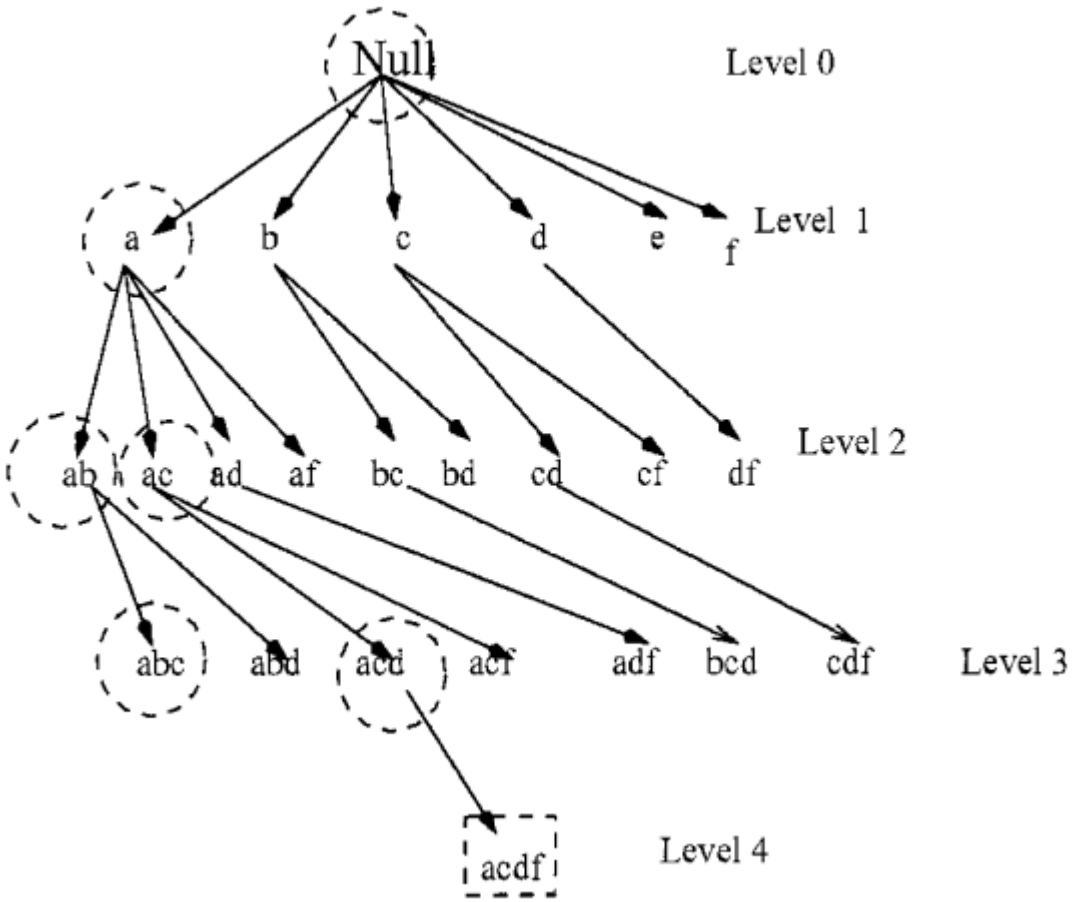
- Thu thập dựa vào báo cáo từ @time macro trong Julia.

Dataset	minSup	minConf	Time (seconds)	Memory
chess	95	80	0.949181	2.84 M
mushroom	95	80	0.809009	2.72 M
retail	8	70	1.603728	6.85 M
foodmart	0.1	40	198.646797	47.14 M

4 Tree Projection Algorithm:

4.1 Ý tưởng thuật toán:

- Theo em được tìm hiểu theo bài báo (xem trong tài liệu tham khảo), Tree Projection Algorithm là thuật toán để tìm ra tập các mục thường xuyên trong cơ sở dữ liệu dựa vào cấu trúc cây "Lexicographic".
- Một **cấu trúc** cây Lexicographic được định nghĩa như sau:
  - Một đỉnh tồn tại trong cây tương ứng với mỗi tập các mục thường xuyên. Gốc của cây tuong ứng với một tập rỗng.
  - Đặt I = $[i_1,...,i_k]$ là một tập các mục thường xuyên, trong đó các item được sắp xếp theo thứ tự trong tù điển. Cha của nút I là tập các mục $[i_1,...i_{k-1}]$.
- Tập frequent 1-extension của một tập các mục sao cho mục cuối cùng đóng góp vào cho phần mở rộng được gọi là một **frequent lexicographic tree extension**.
- Tập các nhánh ứng viên của một nút P được định nghĩa là các phần tử trong E(Q) xếp sau nút P theo thứ tự từ điển.
- Các cấp độ (**level**) trong cây lexicographic tương ứng với kích thước của các nhóm mục khác nhau.
- Một node được tạo ra (**generated**) nhờ sự mở rộng của nút cha.
- Một node được cho là đã kiểm tra (**examined**) khi một frequent lexicographic tree extension của nó được xác định.
- Một node trong cây được xác định là **unactive** nếu từ nó không thể mở rộng thêm cây con. Ngược lại, node được cho là **active**.
- Một node active được cho là node biên (**boundary node**) nếu nó đã được tạo nhưng chưa được kiểm tra.
- Có nhiều chiến lược thuật toán khác nhau để tạo cây lexicographic, đơn giản nhất là **Depth-First** và **Breadth-First Search.** Tất cả các nút ở mức k có thể được tạo trước khi các nút ở mức k+1 hoặc các mẫu dài hơn có thể được phát hiện sớm hơn để loại bỏ một số nhánh khác của cây.



Cấu trúc của một lexicographic tree (Source: *A Tree Projection Algorithm for Generation of Frequent Item Sets)*

4.2 Các bước thực hiện:

- Trong bài báo cáo này chỉ trình bày thuật toán Breadth First Creation cho cấu trúc lexicographic tree.
- Bước 1: Tạo node null cùng với tất cả các node ứng với tất cả các item ở level 1.
- Bước 2: Tạo ma trận tam giác ở các node cấp (k-1) và đếm support của các node ở cấp (k+1).
- Bước 3: Kiểm tra support cho tất cả các mục trong ma trận. Tạo các nút mới ở cấp (k+1) ứng với các mục đạt điều kiện support.
- Bước 4: Tỉa tất cả các nút unactive khỏi cây, tỉa danh sách các mục đang active ở tất cả các node.
- Bước 5: Tăng biến k lên 1 đơn vị. Kiểm tra xem ở level thứ k, cây tạo ra có băng null hay không? Nếu bằng null thì dừng, kết thúc thuật toán, nếu không quay lại bước 2.

4.3 Ưu - nhược điểm của thuật toán:

4.3.1 Ưu điểm:

- Cây lexicographic tiêu tốn ít bộ nhớ hơn nhiều so với việc triển khai hàm hash tree trong Apriori Algorithm.
- Cấu trúc cây Lexicographic giảm đáng kể thời gian duyệt qua cơ sở dữ liệu để đếm tập các mục thường xuyên.
- Thuật toán này đạt tốc độ nhanh khi khai thác cơ sở dữ liệu mà trong đó, các tập có số lượng item trong một mục ít.

4.3.2 Nhược điểm:

- Nếu một mục có chứa số lượng nhiều các item, quá trình tạo cây sẽ rộng, đặt yêu cầu về bộ lưu trữ phải lớn.

4.4 Ứng dụng khai thác dữ liệu:

(Chưa cài đặt thuật toán thành công, chưa khai thác được)

5 Chương trình cài đặt:

5.1 Chuẩn bị trước khi chạy chương trình:

- Mở **terminal** bên trong thư mục 'Source'.
- Trong thư mục 'Source' cần phải có thư mục '**Data**' chứa các file dữ liệu dạng .txt để lấy dữ liệu đầu vào cho chương trình.

5.2 Cú pháp tham số dòng lệnh:

- Cú pháp run thuật toán **Apriori**:
  - **minSup** và **minConf** là số phần trăm thuộc miền [0:100]

```
julia FIM-Apriori.jl [Đường dẫn file txt] minSup minConf
```

- Cú pháp run thuật toán **Tree Projection**: *(chưa cài đặt)*

5.3 Kết quả mong đợi:

6 Tài liệu tham khảo:

- Tài liệu môn học, silde lý thuyết trên Moodle.
- https://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html
- Association Rule Mining: http://athena.ecs.csus.edu/~mei/associationcw/Association.html
- Apriori: Association Rule Mining In-depth Explanation and Python Implementation: https://towardsdatascience.com/apriori-association-rule-mining-explanation-and-python-implementation-290b42afdfc6
- R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. A Tree Projection Algorithm for Generation of Frequent Itemsets, 2001.