UEH
UNIVERSITY

BÁO CÁO

Môn học: Khai Phá Dữ Liệu

Giảng viên hướng dẫn: Nguyễn An Tế

Nhóm sinh viên: Phan Trung Hiếu
Đỗ Thanh Hoa
Nguyễn Trương Hoàng
Nguyễn Huy Hoàng
Nguyễn Hoàng Huy

Đề tài: ***Nghiên cứu thuật toán ECLAT với tập dữ liệu Books.csv***

TP. Hồ Chí Minh, ngày 27 tháng 11 năm 2024

THÀNH VIÊN NHÓM

STT	Họ và tên	MSSV
1	Phan Trung Hiếu	88214020002
2	Đỗ Thanh Hoa	31211025193
3	Nguyễn Huy Hoàng	31221023992
4	Nguyễn Trương Hoàng	31221025159
5	Nguyễn Hoàng Huy	31221026058

LỜI NÓI ĐẦU

Kính gửi quý thầy cô và các quý độc giả, đây là bài báo cáo thuộc đồ án cuối kỳ môn Khai phá dữ liệu. Cấu trúc bài báo cáo chia làm 6 chương như sau:

Chương 1: Tổng quan đề tài của nhóm, mục tiêu mà nhóm muốn đạt được cũng như cách thức mà nhóm sẽ thực hiện để đạt được mục tiêu,

Chương 2: Cơ sở lý thuyết, thể hiện những lý thuyết, khái niệm được áp dụng trong đề tài,

Chương 3: Tổng quan bộ dữ liệu Books.csv,

Chương 4: Kết quả thực nghiệm mà nhóm thu hoạch được,

Chương 5: Đánh giá mức độ hiệu quả của thuật toán và so sánh với hai thuật toán Apriori và FP-Growth,

Chương 6: Thảo luận về kết quả đạt được, hạn chế cũng như đề xuất hướng phát triển.

Vì gặp giới hạn về thời gian và kiến thức nên bài báo cáo không thể tránh khỏi những sai sót. Nhóm tác giả rất mong nhận được sự đóng góp ý kiến từ quý thầy cô và các quý độc giả để đồ án được hoàn thiện hơn.

Nhóm tác giả cũng xin chân thành cảm ơn Ts. Nguyễn An Tế đã tận tình giảng dạy, hướng dẫn để nhóm hoàn thành đồ án này.

Trân trọng,

Nhóm 04

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

CHƯƠNG I: TỔNG QUAN ĐỀ TÀI

1.1. Tổng quan đề tài

Với sự bùng nổ của dữ liệu trong thời đại ngày nay, rất nhiều lĩnh vực đã từng bước gia nhập và ứng dụng những loại công nghệ và kỹ thuật để khai phá dữ liệu, để từ đó đưa ra những thông tin hữu ích cho việc quản lý, điều hành doanh nghiệp hay một dự án nào đó. Song, việc tìm hiểu sở thích của một người về một sự vật, sự việc nào đó cũng nằm trong phạm vi của công việc khai phá dữ liệu. Cụ thể hơn, trong bài báo cáo này, nhóm nghiên cứu sẽ tìm hiểu và ứng dụng thuật toán ECLAT trong việc khai phá thông tin về các thể loại sách thường mua kèm cùng nhau của tập dữ liệu Book.csv nhằm hỗ trợ những doanh nghiệp đang kinh doanh và hoạt động trong lĩnh vực này đưa ra các quyết định trong việc quản lý và vận hành các sản phẩm sách tại doanh nghiệp, tổ chức của mình.

1.2. Mục tiêu nghiên cứu

Cụ thể hơn về mục tiêu mà nhóm nghiên cứu mong muốn đạt được thông qua việc khai thác bộ dữ liệu Book.csv là phân tích mối quan hệ giữa các thể loại sách nhằm hiểu rõ xu hướng tiêu thụ và sở thích đọc sách của người dùng. Nhóm hướng đến việc xác định các thể loại sách thường xuyên được chọn cùng nhau để xây dựng hệ thống gợi ý sách, tối ưu hóa nội dung và chiến lược kinh doanh. Ngoài ra, nhóm mong muốn có thể đưa ra thông tin hỗ trợ phát hiện các thể loại sách ít phổ biến, từ đó điều cung cấp thông tin có giá trị giúp phục vụ nghiên cứu về thị hiếu đọc sách.

1.3. Phương pháp nghiên cứu

Về cách thức nghiên cứu, nhóm sẽ tìm hiểu thuật toán ECLAT để tìm ra các luật kết hợp mạnh giữa các đối tượng trong tập dữ liệu với nhau. Sau đó nhóm sẽ tiến hành so sánh và đánh giá kết quả đạt được để đưa ra kết luận thông tin có hữu ích cho người sử dụng.

Về nguồn dữ liệu sử dụng trong bài toán này, nhóm sử dụng bộ dữ liệu Book.csv do giảng viên cung cấp.

CHƯƠNG II: CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về khai phá luật kết hợp

Luật kết hợp (Association Rules) là các phát biểu "nếu/thì" được sử dụng để khám phá các mối quan hệ giữa các dữ liệu không liên quan trong cơ sở dữ liệu, cơ sở dữ liệu quan hệ hoặc các kho thông tin khác. Luật kết hợp được áp dụng để tìm ra mối liên kết giữa các đối tượng thường xuyên được sử dụng cùng nhau [3].

Khai phá luật kết hợp (Association Rule Mining - ARM) là một trong những phương pháp khai phá dữ liệu phổ biến và được biết đến rộng rãi nhất để khám phá các mối quan hệ giữa các biến trong cơ sở dữ liệu giao dịch hoặc các kho dữ liệu khác. Một luật kết hợp (association rule) được định nghĩa dưới dạng một quan hệ $X \Rightarrow Y$, trong đó X và Y là các tập mục (itemsets) không giao nhau (tức không có phần tử chung). Ý nghĩa trực quan của một luật kết hợp như vậy là khi X xuất hiện, Y cũng có xu hướng xuất hiện theo [4].

Hai tiêu chí cơ bản để đánh giá luật kết hợp là độ hỗ trợ (support) và độ tin cậy (confidence). Trong đó, Độ tin cậy (confidence) của một luật $X \Rightarrow Y$ là tỷ lệ giao dịch chứa X mà cũng chứa Y. Còn Độ hỗ trợ (support) của luật là tỷ lệ phần trăm của cơ sở dữ liệu chứa cả X và Y [4]. Hai tiêu chí này giúp xác định các mối quan hệ và tạo ra các quy tắc bằng cách phân tích dữ liệu dựa trên các mẫu "nếu/thì" được sử dụng thường xuyên. Các luật kết hợp thường cần đáp ứng cả độ hỗ trợ tối thiểu và độ tin cậy tối thiểu được chỉ định bởi người dùng [3].

Khi khai phá luật kết hợp thường bao gồm hai nhiệm vụ chính. Nhiệm vụ đầu tiên là Tìm các tập mục có hỗ trợ đủ lớn với việc phải xác định các tập mục có tần suất xuất hiện vượt qua ngưỡng hỗ trợ tối thiểu được xác định trước. Những tập mục này được gọi là tập mục phổ biến (frequent itemsets). Nhiệm vụ tiếp theo là Sinh ra các luật kết hợp, được tiến hành bằng cách kết hợp các tập mục phổ biến đó từ các tập mục lớn thu được với ràng buộc là phải thỏa mãn ngưỡng độ tin cậy tối thiểu [5].

Mục đích chính của việc khai phá luật kết hợp là tìm ra các mẫu phổ biến (Frequent Patterns - FP), mối liên kết và mối quan hệ giữa các cơ sở dữ liệu khác nhau thông qua các phương pháp khai thác khác nhau. Điều này khiến khai phá luật kết hợp được sử dụng để cải thiện quá trình ra quyết định trong các ứng dụng như phân tích giỏ hàng (Market

Basket Analysis), chẩn đoán bệnh, xây dựng hệ thống giao thông thông minh (ITS), dữ liệu nhật ký web, hệ thống chẩn đoán hỗ trợ máy tính đối với ung thư vú, phát hiện gian lận trên web, và quản lý quan hệ khách hàng (CRM) trong kinh doanh thẻ tín dụng [14].

2.2. Các khái niệm cơ bản

Cho cơ sở dữ liệu với tập mục I = {A, B, C, D, E}, và các giao dịch được mô tả như sau:

Tid	Items
1	A, B, D, E
2	B, C, E
3	A, B, D, E
4	A, B, C, E
5	A, B, C, D, E
6	B, C, D

Hình 2.1: Cơ sở dữ liệu giao dịch mẫu.

Một cơ sở dữ liệu (*Database*): Là một tập hợp các giao dịch $D = \{Ti\}$ gồm hai thành phần chính. Đó là tập con (T) và các hạng mục (i) trong đó[1]:

Giao dịch (*Transaction*): T là tập con chứa các hạng mục (*Items*), và được biểu diễn dưới dạng $T = \{I_1, I_2, ..., I□\}$ (Items trong T được quy ước theo thứ tự tăng dần.). Tập các hạng mục (*Items)*: $I = \{I_1, I_2, ..., I□\}$ *bao* gồm n mục xuất hiện trong cơ sở dữ liệu. Tập hợp con (*Itemset)*: Là một tập hợp con được ký hiệu là $X \subseteq I$. Nếu tập hợp con này có ***k*** phần tử, thì nó được gọi là (*k-itemset)*: $X□ = \{I_1, I_2, ..., I□\}$ với $k \leq n$.

Quy tắc kết hợp *(Association rule)*: Được biểu diễn dưới dạng

$X \Rightarrow Y, \varnothing \neq X \subset I, \varnothing \neq Y \subset I, X \cap Y = \varnothing$. Trong đó, X và Y là hai tập hợp con không giao nhau và đều khác rỗng.

Tần số (*frequency*): Được định nghĩa là số giao dịch trong D chứa X và được tính như sau,

[1] Định nghĩa về các khái niệm trong phần này được lấy từ file powerpoint trình chiếu *"Chương 4: Mẫu phổ biến, Luật kết hợp"* của thầy Nguyễn An Tế.

$$\text{freq}(X) = |\{T \in D | X \subseteq T\}|$$

Độ hỗ trợ (*support*): Thể hiện tần suất xuất hiện của X trong cơ sở dữ liệu, qua đó có thể đánh giá được mức độ phổ biến của mẫu trong tập dữ liệu đó và được tính như sau,

$$\sup(X \Rightarrow Y) = P(X \cup Y) = \text{freq}(X \cup Y) / |D|.$$

Độ tin cậy (*confidence*): Là tần suất xuất hiện của tập mục X và Y, được sử dụng để đánh giá mức độ chính xác của quy tắc phân lớp,

$$conf(X \Rightarrow Y) = P(Y \mid X) = freq(X \cup Y) / freq(X)$$

Tập phổ biến (*frequent pattern*) là tập hợp các mục có độ hỗ trợ (support) lớn hơn hay bằng độ hỗ trợ tối thiểu (để khai thác các tập mục, người ta thường đặt ra ngưỡng gọi là minSup, để dựa vào đó mà có thể loại hay nhận tập đó). Được định nghĩa như sau,

$$\text{freq}(X) \geq \text{minSup}$$

$$L_k = \{ \text{frequent k-itemsets} \}$$

2.3. Một số phương pháp khai phá luật kết hợp

Khai phá luật kết hợp là một lĩnh vực trong khai thác dữ liệu, chủ yếu nhằm tìm ra các mẫu và mối liên hệ giữa các tập hợp đối tượng trong dữ liệu. Dưới đây là một số thuật toán phổ biến trong khai phá luật kết hợp,

Một, thuật toán Apriori. Thuật toán này tìm kiếm các tập hợp item thường xuyên bằng cách sử dụng phương pháp loại bỏ. Nó bắt đầu từ các item đơn lẻ và dần dần xây dựng lên các tập hợp lớn hơn. Apriori sử dụng giá trị hỗ trợ tối thiểu để xác định các itemset thường xuyên và sau đó tạo ra các luật kết hợp.

Hai, thuật toán FP-Growth. FP-Growth sử dụng cấu trúc cây để lưu trữ dữ liệu, cho phép tìm kiếm itemset thường xuyên mà không cần quét dữ liệu nhiều lần. Nó sẽ tạo cây FP từ tập dữ liệu và sau đó duyệt cây để tìm các itemset thường xuyên.

Ba, thuật toán Relation sử dụng lý thuyết tập hợp để xác định các itemset và luật kết hợp. Thuật toán này hoạt động tương tự như Apriori nhưng tập trung vào mối quan hệ giữa các itemset.

Bốn, Thuật Toán CHARM. Thuật toán này tìm các itemset thường xuyên bằng cách sử dụng phương pháp tìm kiếm kết hợp và phát hiện các itemset tối thiểu thông qua việc sử dụng một cấu trúc dữ liệu đặc biệt để giảm thiểu số lượng phép toán cần thiết.

Năm, thuật toán SPADE (Sequential Pattern Discovery using Equivalence classes) là thuật toán tìm kiếm mẫu tuần tự bằng cách sử dụng phương pháp phân tách để tìm kiếm các mẫu tuần tự mà không cần quét lại toàn bộ dữ liệu.

Mỗi thuật toán trên có những ưu điểm và nhược điểm riêng ở một số khía cạnh,

Về hiệu suất, Apriori là thuật toán cơ bản và dễ hiểu, nhưng gặp khó khăn với dữ liệu lớn do yêu cầu số lần quét nhiều. FP-Growth có hiệu suất tốt hơn trong xử lý dữ liệu lớn, nhờ giảm thiểu số lần quét và tối ưu hóa tìm kiếm.

Về bộ nhớ, FP-Growth có thể yêu cầu thêm bộ nhớ để lưu trữ cấu trúc cây FP và Charm yêu cầu nhiều bộ nhớ cho các cấu trúc dữ liệu.

Về mức độ đơn giản hay phức tạp, Apriori có ưu điểm về tính đơn giản và dễ triển khai cho những người mới làm quen với khai phá dữ liệu. Ngược lại FP-Growth, Relation, Spade đòi hỏi kiến thức sâu hơn về cấu trúc dữ liệu và thuật toán vì độ phức tạp và khó triển khai.

2.4. Thuật toán ECLAT

Khai phá itemset là một trong những lĩnh vực quan trọng của khai phá dữ liệu. Nó giúp ta tìm ra được những mẫu mang lại thông tin hữu ích trong một cơ sở dữ liệu. Bài toán mà ta thường bắt gặp đối với khai phá itemset đó là tìm ra những itemset thường xuất hiện cùng nhau trong một giao dịch. Những itemset thường xuất hiện đó ta sẽ gọi là tập phổ biến.

Để tìm được những itemset phổ biến đó, một số thuật toán đã được đưa ra, có thể kể đến đó là Apriori, FP-Growth, ECLAT,... Các thuật toán này đều nhận cùng một đầu vào và có cùng một đầu ra. Cái làm nên sự khác biệt giữa các thuật toán này nằm ở chiến lược tìm kiếm tập phổ biến và cấu trúc dữ liệu thuật toán dùng để tìm ra các tập phổ biến. Nhìn chung, các thuật toán khai phá tập phổ biến khác nhau ở trên bốn phương diện: (i) thuật toán tìm kiếm tập phổ biến theo chiều sâu hay theo chiều rộng, (ii) cơ sở dữ liệu đối với

yêu cầu của thuật toán được biểu diễn như thế nào, (iii) cách xác định itemset tiếp theo sẽ được xem xét ở trong không gian tìm kiếm và (iv) cách mà thuật toán đo lường support của itemset [1].

ECLAT *(Equivalence Class Clustering and bottom-up Lattice Traversal)* là một thuật toán khai phá tập phổ biến bằng cách tìm kiếm các tập phổ biến theo chiều sâu *(depth-first search)*. Nếu như thuật toán Apriori yêu cầu cơ sở dữ liệu được biểu diễn ở chiều ngang *(horizontal representation)* thì ECLAT yêu cầu cơ sở dữ liệu được biểu diễn ở chiều dọc *(vertical representation)*.

TID	Itemsets
T1	OOP with C++, .Net Framework with C#, ASP.Net with C#
T2	LCCI Level1&2, LCCI Level 3
T3	OOP with Core JAVA, J2SE, J2EE
T4	.Net Framework with C#, ASP.Net with C#
T5	LCCI Level1&2
T6	OOP with Core JAVA, J2SE, J2EE
T7	OOP with Core JAVA
T8	OOP with C++, .Net Framework with C#, ASP.Net with C#
T9	J2SE, J2EE

Items	TID-set
OOP with C++	{T1,T8}
OOP with Core JAVA	{T3,T6,T7}
.Net framework with C#	{T1,T4,T8}
ASP.NET with C#	{T1,T4,T8}
Advanced JAVA(J2SE)	{T3,T6,T9}
Web JAVA (J2EE)	{T3,T6,T9}
LCCI Level 1&2	{T2,T5}
LCCI Level 3	{T2}

Hình 2.2: Cơ sở dữ liệu ngang (trái) và Cơ sở dữ liệu dọc (phải) - Nguồn: [2]

Với hình trên, ở bên trái ta có thể thấy đó là cơ sở dữ liệu ở chiều ngang với mỗi giao dịch sẽ tương ứng với danh sách các item. Trong khi đó, đối với cơ sở dữ liệu dọc, tương ứng với mỗi item sẽ là các giao dịch mà có item đó. Giả sử ta có một itemset X, thì các giao dịch ứng với itemset X sẽ được ký hiệu *tid(X)*.

Ý tưởng chính của ECLAT dựa vào cơ sở dữ liệu dọc để tìm các tập phổ biến, giả sử ta có hai itemset X và Y, ECLAT sẽ tìm tập ứng cử viên (k+1)-itemset bằng cách xem xét tập giao của tid(X) và tid(Y), khi đó $tid(X \cup Y) = tid(X) \cap tid(Y)$. Điều này giúp cho ta chỉ cần quét cơ sở dữ liệu một lần giảm thiểu chi phí tính toán rất nhiều so với việc phải quét cơ sở dữ liệu nhiều lần để tìm tập phổ biến như Apriori. Hơn nữa, cách tiếp cận như vậy giúp ta tính toán support của một itemset rất dễ dàng mà không phải quét lại cơ sở dữ liệu, bởi ta có thể tính support thông qua tid(X), nghĩa là sup(X) = |tid(X)| [1]. Quy trình ECLAT tìm các tập phổ biến có thể được mô tả qua các bước sau,

Một, chuyển cơ sở dữ liệu sang cơ sở dữ liệu dọc,

Hai, khai phá tập phổ biến. Bắt đầu với 1-itemset, nếu có N item ta sẽ có số các 1-itemset là N. Sau đó xét các 1-itemset này, nếu itemset nào không thỏa điều kiện sup() ≥ minSup (minimum support) thì ta sẽ loại itemset đó khỏi tập phổ biến. Sau đó tiếp tục quá trình này cho các 2-itemset, 3-itemset, 4-itemset,... Tổng quát hóa, ta sẽ tạo các tập ứng viên k-itemset bằng cách lấy phần giao của các (k-1)-itemset. Đồng thời, ở đây ta cũng xét đến tính chất Apriori, nếu ta sinh ra k-itemset chứa tập con là tập không phổ biến thì ta cũng loại itemset này khỏi kết quả cuối cùng.

Kết thúc quá trình trên, thuật toán sẽ cho ta các tập phổ biến có được từ cơ sở dữ liệu. Và với mỗi tập phổ biến, ta sẽ sinh các luật kết hợp, nếu confidence() ≥ minConf (minimum Confidence) của một luật kết hợp thỏa thì luật kết hợp đó được coi là luật kết hợp mạnh.

CHƯƠNG III: TỔNG QUAN BỘ DỮ LIỆU

3.1. Giới thiệu bộ dữ liệu

Bộ dữ liệu Book.csv là một bộ dữ liệu mô tả sự xuất hiện của các danh mục sách khác nhau trong các giao dịch hoặc mối quan tâm của người dùng cụ thể nào đó. Mỗi bản ghi trong bộ dữ liệu là một giao dịch hoặc thông tin về sở thích mua sách của một nhóm khách hàng nào đó.

Bộ dữ liệu gồm 2000 dòng với 11 cột, mỗi ô trong bộ dữ liệu chứa thông tin biểu thị sự xuất hiện có hay không của thể loại sách tương ứng với một giao dịch.

3.2. Các thuộc tính

STT	Tên thuộc tính	Mô tả	Chú thích
1	Child	Thể loại sách dành cho trẻ em	Dữ liệu dạng nhị phân: 0: không xuất hiện, 1: có xuất hiện
2	Youth	Thể loại sách dành cho thanh thiếu niên	Dữ liệu dạng nhị phân: 0: không xuất hiện, 1: có xuất hiện
3	Cook	Thể loại sách thuộc lĩnh vực nấu ăn	Dữ liệu dạng nhị phân: 0: không xuất hiện, 1: có xuất hiện
4	Science	Thể loại sách thuộc lĩnh vực khoa học	Dữ liệu dạng nhị phân: 0: không xuất hiện, 1: có xuất hiện
5	Music	Thể loại sách thuộc lĩnh vực âm nhạc	Dữ liệu dạng nhị phân: 0: không xuất hiện, 1: có xuất hiện
6	Art	Thể loại sách thuộc lĩnh vực nghệ thuật	Dữ liệu dạng nhị phân: 0: không xuất hiện, 1: có xuất hiện

7	Geog	Thuật loại sách thuộc lĩnh vực địa lý	Dữ liệu dạng nhị phân: 0: không xuất hiện, 1: có xuất hiện
8	Sport	Thể loại sách thuộc lĩnh vực thể thao	Dữ liệu dạng nhị phân: 0: không xuất hiện, 1: có xuất hiện
9	Tourism	Thể loại sách thuộc lĩnh vực du lịch	Dữ liệu dạng nhị phân: 0: không xuất hiện, 1: có xuất hiện
10	Business	Thể loại sách thuộc lĩnh vực kinh doanh	Dữ liệu dạng nhị phân: 0: không xuất hiện, 1: có xuất hiện
11	IT	Thể loại sách thuộc lĩnh vực công nghệ thông tin	Dữ liệu dạng nhị phân: 0: không xuất hiện, 1: có xuất hiện

Bảng 3.1: Sơ lược tập dữ liệu Books.csv

CHƯƠNG IV: KẾT QUẢ THỰC NGHIỆM

4.1. Kết quả đạt được

	Item	Support
4	Cook	0.4310
3	Child	0.4230
0	Science	0.2820
6	Geog	0.2760
7	Child & Cook	0.2560
5	Youth	0.2475
2	Art	0.2410
1	Music	0.2145

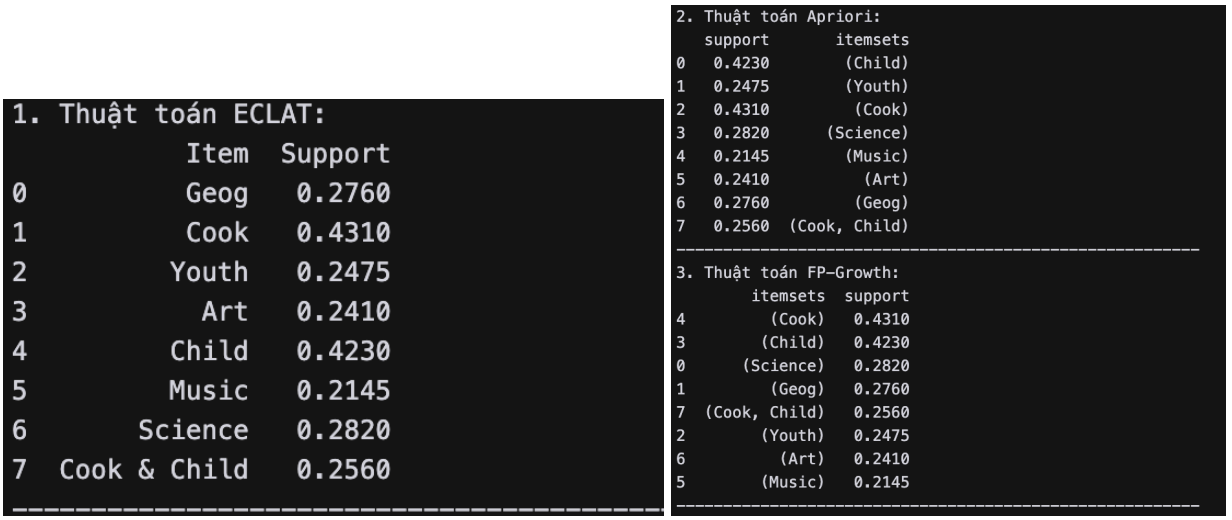
Hình 4.1: Kết quả về các tập phổ biến của thuật toán ECLAT

Dựa vào hình trên, thì có thể thấy tập phổ biến "Cook" có độ hỗ trợ lớn nhất (0.4310) tiếp theo là "Child" có độ hỗ trợ (0.4230) và "Music" có độ hỗ trợ (0.2145). Còn đối với tập hợp phổ biến nhất là giữa "Child & Cook" có độ hỗ trợ là (0.2560).

Cũng dựa vào kết quả trên, có thể thấy được ứng dụng của các tập phổ biến đó vào đời sống. Ví dụ như hình trên cho thấy {Child, Cook} có mối quan hệ với nhau cho thấy độc giả có xu hướng đọc cả hai thể loại này và có thể liên quan đến việc đối tượng của {Child, Cook} đều thuộc lĩnh vực văn học, nghiên cứu hay được mua cùng để làm quà tặng hoặc là người đọc có sở thích đa dạng và mua sách theo chủ đề kết hợp.

So sánh kết quả của 3 thuật toán: ECLAT, Apriori, FP-Growth[2],

[2] Thuật toán ECLAT nhóm sử dụng thư viện pyECLAT, thuật toán Apriori và FP-Growth nhóm sử dụng thư viện mlxtend.



Hình 4.2: Kết quả sau khi chạy ba thuật toán ECLAT, Apriori, FP-Growth.

Dựa vào hình trên, có thể thấy rằng kết quả tập phổ biến của ba thuật toán là như nhau, ECLAT tìm kiếm theo chiều sâu và tập trung vào việc lưu trữ danh sách các giao dịch và sẽ dựa vào việc tính toán lớp tương đương của các giao dịch đó. Còn Apriori sử dụng tìm kiếm theo chiều rộng và bắt đầu với các mục đơn lẻ và sau đó sinh tập con và thực hiện từng bước và cứ thế lặp đi lặp lại cho đến khi không tìm thấy được tập phổ biến nào mới. FP-Growth thì sử dụng FP-Tree để tối ưu hóa việc phát hiện các tập phổ biến. Nói tóm lại, ECLAT chỉ tìm tập phổ biến bằng chiến lược tìm kiếm theo chiều sâu và chỉ duyệt qua cơ sở dữ liệu một lần, Apriori phải duyệt qua nhiều tập con và tính toán số liệu, dẫn đến quá trình xử lý số liệu sẽ phức tạp hơn, còn FP-Growth sẽ nén dữ liệu lại để tiết kiệm bộ nhớ và giảm đi thời gian duyệt cơ sở dữ liệu.

Nói tóm lại, sự khác biệt của các thuật toán chủ yếu nằm ở cách xử lý dữ liệu và mục tiêu sử dụng, chi tiết của thông tin.

4.2. Ứng dụng

Xét tới ứng dụng thực tiễn thì ta hãy thử tìm hiểu xem các thể loại sách nào thường được mua cùng nhau, với độ hỗ trợ tối thiểu và độ tin cậy tối thiểu (minimum confidence) cùng có giá trị 0.2, ta sẽ chỉ có hai luật kết hợp đó là Cook ⇒ Child và Child ⇒ Cook. Điều này cho thấy rằng, khi sách thuộc thể loại Cook được mua thì thường Child cũng sẽ được mua, vì vậy mà ta có thể tận dụng điều này để xếp hai thể loại sách này cạnh nhau.

Nhưng với độ hỗ trợ tối thiểu 0.2 thì ta chỉ tìm được duy nhất một tập phổ biến với hai

phần tử, nếu xét tới việc áp dụng để đặt các thể loại sách cách hợp lý thì ta sẽ giảm độ hỗ trợ tối thiểu xuống 0.15 để có thể tìm ra nhiều luật kết hợp hơn. Không ngoài dự đoán, ta thu được số luật kết hợp nhiều hơn so với ngưỡng độ hỗ trợ tối thiểu cũ, các luật kết hợp có thể kể đến đó là Child ⇒ Youth, Cook ⇒ Art, Child ⇒ Geog,... Dựa trên các luật kết hợp này thì ta có thể sắp xếp sách thuộc thể loại Child, Youth, Geoge cùng với nhau; Cook và Art cùng với nhau. Nhìn chung, kết quả luật kết hợp tùy thuộc vào chúng ta hiệu chỉnh tham số độ hỗ trợ tối thiểu và độ tin cậy tối thiểu như nào, hai tham số càng có giá trị cao thì số lượng sẽ ít hơn nhưng luật kết hợp phần nào sẽ đáng tin cậy hơn do giá trị hai tham số càng cao thì luật kết hợp thỏa điều kiện dường như xuất hiện nhiều hơn trong các giao dịch.

CHƯƠNG V: ĐÁNH GIÁ

5.1. Độ phức tạp

Ở đây, ta sẽ tính toán độ phức tạp về không gian và thời gian. Trước nhất đó là độ phức tạp về không gian. Thuật toán ECLAT sẽ sử dụng cấu trúc dữ liệu không gian theo chiều dọc (vertical data structure) với mỗi item là danh sách ID giao dịch. Điều này khiến độ phức tạp không gian chủ yếu phụ thuộc vào số lượng mục (n) trong tập dữ liệu với số lượng giao dịch của mỗi mục (m). Qua đó độ phức tạp không gian sẽ là O(mn) [9].

Để đo lường độ phức tạp về thời gian, chúng ta sẽ đo lường thông qua các bước mà ECLAT thực hiện để tìm ra luật kết hợp. Đầu tiên, thuật toán ECLAT sẽ phải xây dựng một cấu trúc dữ liệu không gian theo chiều dọc cho mỗi item trong quá trình quét cơ sở dữ liệu ban đầu. Việc xây dựng cấu trúc dữ liệu này cho mỗi mục yêu cầu thực hiện một thao tác cho mỗi giao dịch chứa mục đó. Trong trường hợp xấu nhất, khi mỗi mục xuất hiện trong mọi giao dịch, có thể nói rằng việc xây dựng cấu trúc dữ liệu cho tất cả các mục có độ phức tạp thời gian là O(m * n). Điều này được giải thích bởi vì đối với mỗi mục n, ta cần thêm tối đa m mục vào cấu trúc dữ liệu của nó. Tiếp theo, để thực hiện việc khai phá luật kết hợp thì thuật toán sẽ bắt đầu đọc qua cấu trúc dữ liệu dọc này. Việc đọc cơ sở dữ liệu một lần có thể được coi là yêu cầu thời gian O(m * z), vì thuật toán quét qua m giao dịch và mỗi giao dịch có độ dài trung bình là z. Qua đó, độ phức tạp thời gian sẽ là O(mn) + O(mz) [9].

5.2. Thời gian

Thuật toán	Thời gian trung bình (s)
ECLAT	1.66
Apriori	0.09
FP-Growth	0.22

Bảng 5.1: Thời gian chạy trung bình của ECLAT, Apriori và FP-Growth

Nhìn chung, ECLAT có thời gian chạy trung bình là 1.66 giây, điều này cho thấy ECLAT chậm hơn đáng kể so với 2 thuật còn lại. Đối với Apriori thì có thời gian chạy trung bình 0.09 giây là thuật toán nhanh nhất trong ba thuật toán. Thời gian này rất ngắn cho thấy

hiệu suất tốt trong việc khai phá luật kết hợp. FP-Growth có thời gian chạy trung bình là 0.22 giây, nhanh hơn nhiều so với ECLAT nhưng chậm hơn một chút so với Apriori.

Để diễn giải cho kết quả trên, trước tiên ta xét hướng thuật toán sử dụng phương pháp "vertical data format" (định dạng dữ liệu theo chiều dọc), nghĩa là mỗi itemset được lưu trữ cùng với các TID-set (TIDs) của nó. Phương pháp này yêu cầu nhiều phép toán giao nhau giữa các tập hợp ID, dẫn đến thời gian xử lý lâu hơn, đặc biệt với dữ liệu lớn [7].

Về tối ưu hóa dữ liệu thiếu các cơ chế tối ưu hóa dữ liệu như FP-Growth, vì nó không sử dụng cấu trúc cây để nén thông tin. Việc xử lý nhiều phép toán giao nhau và quản lý danh sách ID có thể làm tăng độ phức tạp tính toán.

Về đặc điểm dữ liệu nếu dữ liệu có nhiều quan hệ phức tạp hoặc itemset lớn, ECLAT có thể gặp khó khăn hơn trong việc xử lý vì nó phải thực hiện nhiều phép toán giao nhau.

Về hiệu suất bộ nhớ ECLAT có thể cần nhiều bộ nhớ hơn để lưu trữ danh sách ID, dẫn đến việc truy cập bộ nhớ chậm hơn so với việc sử dụng cấu trúc cây trong FP-Growth.

5.3. So sánh với Apriori và FP-Growth

Ở phần này, ta sẽ so sánh với thuật toán ECLAT với thuật toán Apriori và FP-Growth để tìm hiểu xem có gì khác biệt giữa ba thuật toán này, ta sẽ so sánh trên ba khía cạnh, (i) Kết quả tập phổ biến mà ba thuật toán trả về, (ii) Thời gian thực chạy của ba thuật toán khi kích thước tập dữ liệu tăng lên, khi độ hỗ trợ tối thiểu (minimum support) tăng.

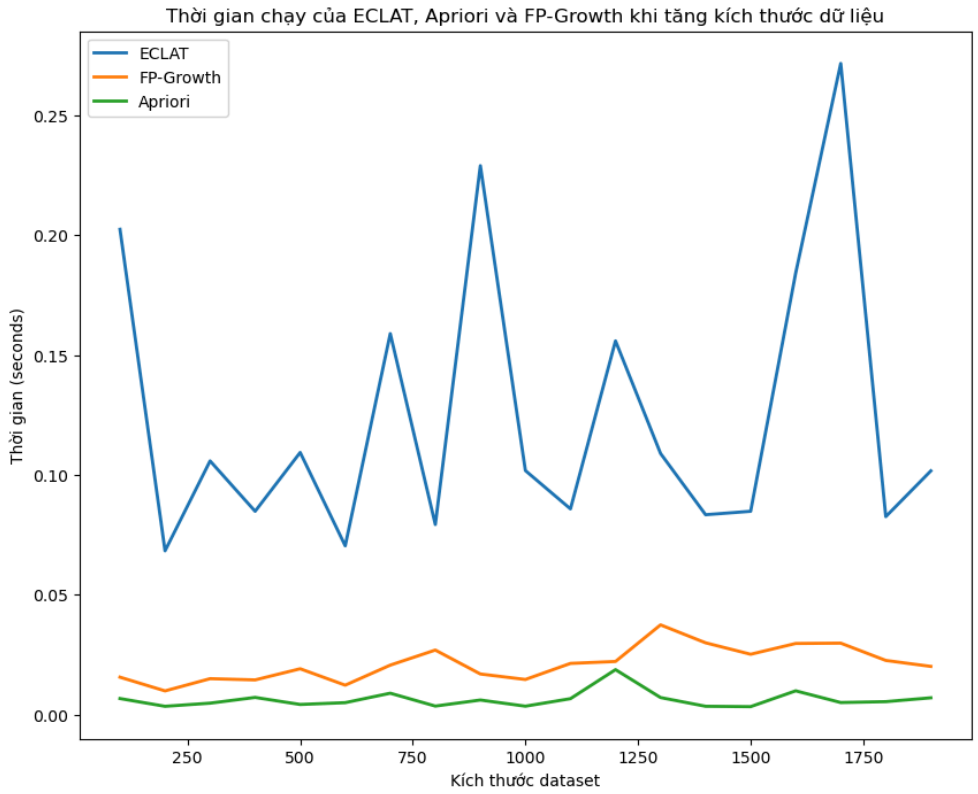
Đầu tiên, ta sẽ so sánh kết quả tập phổ biến mà thuật toán ECLAT trả về với hai thuật toán Apriori và FP-Growth. Tiến hành chạy thuật toán ECLAT, trên tập dữ liệu chỉ định, với độ hỗ trợ tối thiểu là 0.2. Khi so sánh tập phổ biến ECLAT trả về với Apriori và FP-Growth ở phần 4.1 trong chương IV, nhóm nhận thấy rằng ba thuật toán đều cho ra cùng một kết quả, như vậy, ở đây ta có thể kết luận ba thuật toán này có chung một kết quả về tập phổ biến.

Nhưng để xem xét sự khác biệt giữa ba thuật toán này rõ ràng hơn, nhóm sẽ đo lường thời gian chạy của ba thuật toán[3],

Thứ nhất, ta sẽ tìm hiểu xem khi kích thước tập dữ liệu tăng lên thì liệu có gì khác biệt

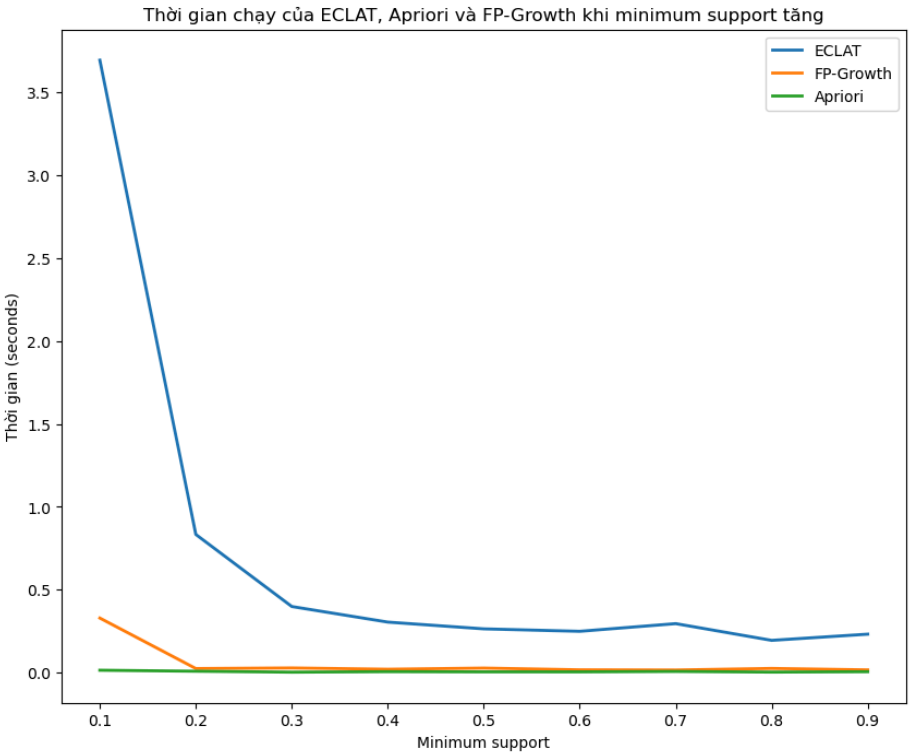
[3] Do nhóm hai thư viện khác nhau, pyECLAT cho ECLAT, mlxtend cho Apriori và FP-Growth nên có thể kết quả thực nghiệm mà nhóm đo lường sẽ khác so với lý thuyết.

giữa ba thuật toán hay không, bắt đầu từ (100, 11) cho tới (1900, 11) mỗi vòng lặp chiều thứ nhất sẽ được tăng lên 100,



Hình 5.1: Thời gian thực chạy của ECLAT, Apriori và FP-Growth khi tăng kích thước dữ liệu

Thứ hai, khi độ hỗ trợ tối thiểu tăng,



Hình 5.2: Thời gian thực chạy của ECLAT, Apriori và FP-Growth khi độ hỗ trợ tối thiểu tăng

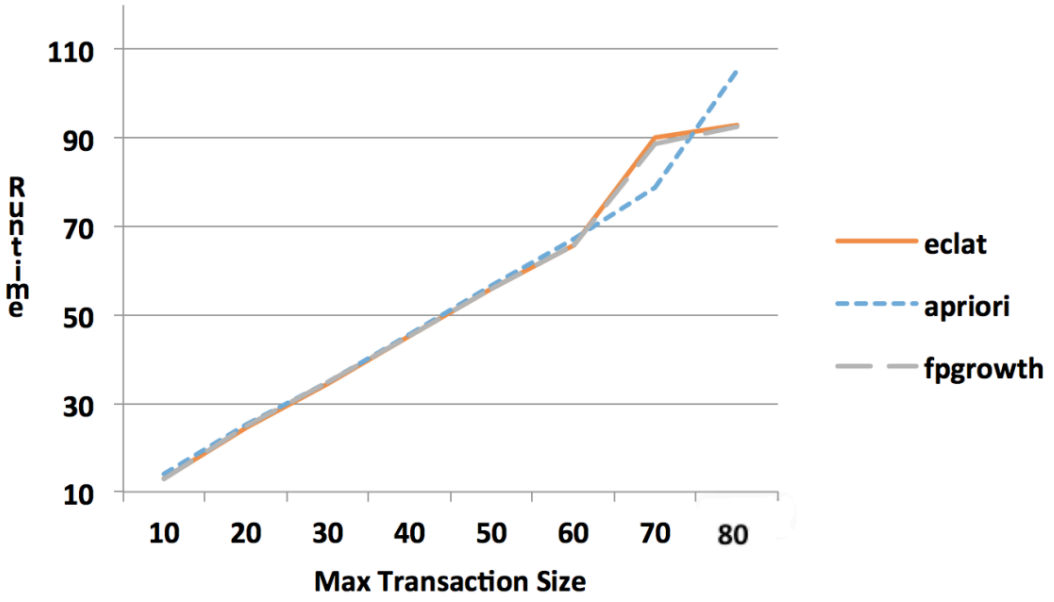
Thuật toán khai phá tập phổ biến sơ khai sẽ tìm hết các tập phổ biến có thể có trong một cơ sở dữ liệu từ đó dẫn đến chi phí cho bộ nhớ lớn để có thể lưu trữ các tập phổ biến này. Thuật toán Apriori ra đời nhằm khắc phục điều đó với tính chất Apriori: "Nếu tập X là tập phổ biến thì tập con của X cũng phổ biến". Tuy điều này có cải thiện về chi phí tính toán trong không gian tìm kiếm nhưng nhược điểm của Apriori đó là phải duyệt cơ sở dữ liệu nhiều lần để có thể tìm ra tập phổ biến cũng như từ đó mà tìm được các luật kết hợp. Vì thế mà một số thuật toán khác đã ra đời như ECLAT, FP-Growth nhằm khắc phục điều đó,... ECLAT chỉ duyệt cơ sở dữ liệu một lần và phần lớn chi phí nằm ở việc lấy phần giao giữa các itemsets trong khi FP-Growth dùng một cơ sở dữ liệu ảo (projected database) để giảm chi phí khi tìm kiếm [1]. Với những thông tin như vậy thì nhóm sẽ tiên liệu rằng ECLAT và FP-Growth sẽ có thời gian chạy nhanh hơn Apriori nhưng kết quả thu được ở hai trường hợp lại gây bất ngờ.

Ở trường hợp thứ nhất, khi mà kích thước tập dữ liệu tăng lên thì ECLAT là thuật toán có thời gian lâu nhất, tiếp đến là FP-Growth và cuối cùng nhanh nhất đó chính là Apriori. Tương tự với trường hợp thứ hai, khi độ hỗ trợ tối thiểu tăng thì ECLAT là lâu nhất, tiếp

đến là FP-Growth nhưng ở điểm gãy khúc (minimum support = 0.2) thì FP-Growth gần như không có chênh lệch đáng kể so với Apriori.

Chung quy lại, ở cả hai trường hợp Apriori đều cho thời gian chạy nhanh nhất. Nếu không xét tới ảnh hưởng của việc sử dụng hai thư viện khác nhau thì ta có thể lý giải điều này như sau. Các thuật toán khai phá tập phổ biến khác như ECLAT, FP-Growth,... vốn sinh ra để cải thiện về mặt chi phí tính toán bởi khi cơ sở dữ liệu lớn thì Apriori dường như có chi phí rất cao. Cũng lập luận đó, đặt trong trường hợp tập dữ liệu của nhóm với khoảng gần 2000 dòng và 11 cột thì một cách chủ quan, nhóm nhận thấy rằng kích thước của cơ sở dữ liệu chưa đạt tới ngưỡng để có thể cho thấy được sự khác biệt rõ ràng giữa ECLAT, FP-Growth và Apriori. Nguyên nhân cho việc ECLAT chạy lâu nhất có lẽ đến từ cách hoạt động của thuật toán này, phần lớn chi phí của ECLAT nằm phép toán giao giữa các tập hợp; FP-Growth chỉ phí chủ yếu nằm ở cấu trúc cây sử dụng để lưu trữ thông tin và sinh tập phổ biến; Apriori chi phí nằm ở việc lưu số lượng ứng viên và duyệt cơ sở dữ liệu nhiều lần, vấn đề này sẽ càng rõ hơn khi kích thước tập dữ liệu càng tăng, có thể sẽ xảy ra trường hợp bùng nổ tổ hợp. Đó cũng chính là lý do vì sao nhóm nhận định rằng kích thước của tập dữ liệu chưa đạt ngưỡng để có thể thấy được sự khác biệt về hiệu suất được cải thiện khi sử dụng ECLAT.

Để củng cố cho lập luận của nhóm, ta sẽ tham khảo kết quả nghiên cứu của Jeff Heaton từ [8], Jeff Heaton đã đo lường thời gian chạy của ECLAT, FP-Growth và Apriori khi transaction size tăng lên. Transaction size xác định số lượng tối đa các item có thể có của một giao dịch, transaction size lớn hơn nghĩa là số lượng phần tử tối đa của một tập phổ biến cũng lớn hơn và từ đó tăng kích thước của cấu trúc dữ liệu mà ta dùng để lưu trữ. Kết quả từ Jeff Heaton cho ta thấy rằng ở mức 60 thì ba thuật toán gần như tương đương nhau về thời gian nhưng cũng từ mức này trở đi Apriori bắt đầu tăng trưởng nhanh hơn, cho thấy rằng kích thước của tập dữ liệu có ảnh hưởng tới sự khác biệt về thời gian giữa ba thuật toán.



Hình 5.3: Thời gian chạy thực của ECLAT, FP-Growth và Apriori khi transaction size tăng (Jeff Heaton) - Nguồn [8]

CHƯƠNG VI: KẾT LUẬN

6.1. Thảo luận

Qua bài quá trình nghiên cứu về thuật toán ECLAT, nhóm nhận thấy một số ưu cũng như là nhược điểm của thuật toán này như sau,

Về nhược điểm, xét bản thân cơ chế của thuật toán, ECLAT tìm tập phổ biến bằng chiến lược tìm kiếm theo chiều sâu và chỉ duyệt qua cơ sở dữ liệu một lần, phần lớn chi phí nằm ở phép toán giao nhau nên có thể có itemset có số lượng danh sách giao dịch tương đương với kích thước của cơ sở dữ liệu dẫn đến chi phí tính toán tăng cao.

Về ưu điểm, có thể thấy rõ ưu điểm lớn nhất của ECLAT nằm ở phần tối ưu hoá chi phí tìm kiếm đối với các bộ dữ liệu lớn nhờ vào cách thức hoạt động khi chỉ duyệt cơ sở dữ liệu một lần. Nhưng kết quả thực nghiệm mà nhóm đạt được cho thấy ECLAT có thời gian chạy lâu hơn hai thuật toán Apriori và FP-Growth, nhóm đưa ra lập luận rằng kích thước của tập dữ liệu chưa đạt ngưỡng để có thể thấy được sự cải tiến về mặt chi phí của ECLAT so với Apriori. Và đây cũng là phần mà nhóm đã khám phá ra được qua quá trình thực hiện đánh giá và so sánh thuật toán với hai thuật toán FP-Growth và Apriori.

Song, bài nghiên cứu vẫn còn tồn tại một số hạn chế như nhóm đã sử dụng các chỉ số cơ bản (thời gian chạy, độ phức tạp) để đánh giá kết quả mà ECLAT mang lại. Tuy nhiên, việc áp dụng các thước đo chuyên sâu như Lift, Kulc chưa được thực hiện để đánh giá tập phổ biến. Điều này có thể dẫn đến việc bỏ lỡ một số thông tin hữu ích hoặc mối quan hệ ẩn tiềm năng trong bộ dữ liệu. Ngoài ra, trong quá trình thực hiện, nhóm đã tập trung vào việc sử dụng các thư viện mạnh mẽ như mlxtend, pyECLAT để khai thác luật kết hợp thay vì tự xây dựng thuật toán. Điều này dẫn tới một số hạn chế khác như nhóm không đào sâu vào cơ chế hoạt động nội tại của các thuật toán được sử dụng cũng như khó tùy chỉnh để đáp ứng các yêu cầu đặc thù trong bài toán (giả sử nếu như áp dụng vào những bài toán có độ phức tạp cao).

6.2. Hướng phát triển

Trước hết, nhóm mong muốn sẽ áp dụng được các thước đo chuyên sâu như Lift, Kulc cũng như tự xây dựng thuật toán bằng code để có thể đánh giá hiệu quả thuật toán ECLAT

một cách sát sao nhất. Việc này không chỉ giúp khai thác sâu hơn các mối quan hệ tiềm ẩn trong dữ liệu mà còn cung cấp cái nhìn toàn diện hơn về giá trị của các luật kết hợp.

Cuối cùng, nhóm cũng mong rằng có thể thử nghiệm các thuật toán trên tập dữ liệu lớn và đa dạng hơn sẽ được triển khai nhằm đánh giá hiệu suất thực tế, bao gồm tối ưu hóa thời gian chạy và giảm thiểu độ phức tạp trong môi trường dữ liệu lớn.

TÀI LIỆU THAM KHẢO

[1] Fournier‐Viger, P., Lin, J. C., Vo, B., Chi, T. T., Zhang, J., & Le, H. B. (2017). A survey of itemset mining. *WIREs Data Mining and Knowledge Discovery*, *7*(4). https://doi.org/10.1002/widm.1207

[2] Aung, H. H., & Myo, K. M. (2009, December 30). *ECLAT-Based association rules mining for education training centre*. http://onlineresource.ucsy.edu.mm/handle/123456789/1872

[3] Kumbhare, T. A., & Chobe, S. V. (2014). An overview of association rule mining algorithms. International Journal of Computer Science and Information Technologies, 5(1), 927-930.

[4] Romero, C., Romero, J. R., Luna, J. M., & Ventura, S. (2010, June). Mining rare association rules from e-learning data. In Educational Data Mining 2010.

[5] Dongre, J., Prajapati, G. L., & Tokekar, S. V. (2014, February). The role of Apriori algorithm for finding the association rules in Data mining. In 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT) (pp. 657-660). IEEE.

[6] Shaukat, K., Zaheer, S., & Nawaz, I. (2015). Association rule mining: An application perspective. International Journal of Computer Science and Innovation, 2015(1), 29-38.

[7] https://stats.stackexchange.com/questions/167439/what-is-the-difference-between-apriori-and-eclat-algorithms

[8] Heaton, J. (2017, January 30). *Comparing Dataset Characteristics that Favor the Apriori, Eclat or FP-Growth Frequent Itemset Mining Algorithms*. arXiv.Org. https://arxiv.org/abs/1701.09042

[9] Philippe Fournier-Viger (2022, June 01). How to Analyze the Complexity of Pattern Mining Algorithms. Truy cập tại: https://data-mining.philippe-fournier-viger.com/how-to-analyze-the-complexity-of-pattern-mining-algorithms%EF%BC%9F/

ĐÁNH GIÁ CÔNG VIỆC

1. Phan Trung Hiếu

Công việc	Đánh giá
Viết báo cáo: Chương II (2.2), Chương IV (4.1)	100%
Code thuật toán Apriori (sử dụng thư viện)	100%
Kiểm tra chính tả chương I và II	100%

2. Đỗ Thanh Hoa

Công việc	Đánh giá
Viết báo cáo: Chương I, III, VI	100%
Thiết kế powerpoint cho trình chiếu	100%

3. Nguyễn Trương Hoàng

Công việc	Đánh giá
Viết báo cáo: Chương II (2.1), Chương V (5.1)	100%
Code thuật toán ECLAT (sử dụng thư viện)`	100%
Kiểm tra chính tả chương III và IV	100%

4. Nguyễn Huy Hoàng

Công việc	Đánh giá
Viết báo cáo: Chương II (2.4, 2.5), Chương V (5.3), Chương IV (4.2)	100%
Code đo lường so sánh thời gian chạy giữa ECLAT, Apriori và FP-Growth	100%

5. Nguyễn Hoàng Huy

Công việc	Đánh giá
Viết báo cáo: Chương II (2.3), Chương V (5.2)	100%
Code thuật toán FP-Growth (sử dụng thư viện)	100%
Kiểm tra chính tả chương V và VI	100%