LỜI CẢM ƠN

Em xin chân thành cảm ơn các thầy giáo, cô giáo trong ngành Công nghệ thông tin – Đại Học Dân Lập Hải Phòng, đã tận tâm giảng dạy các kiến thức trong 4 năm học qua cũng với sự động viên từ gia đình và bạn bè và sự chố gắng hết sức của bản thân.

Đặc biệt em xin bày tỏ sự biết ơn sâu sắc đến thầy giáo Tiến sĩ Phùng Văn Ổn, người đã tận tình hướng dẫn, động viên em thực hiện đồ án này.

Rất mong sự đóng góp ý kiến từ tất cả thầy cô, bạn bè đồng nghiệp để đồ án có thể phát triển và hoàn thiện hơn đồ án này.

Hải phòng, tháng 7 năm 2010

Người thực hiện

Nguyễn Ngọc Châu

MỤC LỤC

MỞ ĐẦU

Trong những năm gần đây, việc nắm bắt được thông tin được coi là cơ sở của mọi hoạt động sản xuất, kinh doanh. Các nhân hoặc tổ chức nào thu thập và hiểu được thông tin, và hành động dựa trên các thông tin được kết xuất từ các thông tin đã có sẽ đạt được thành công trong mọi hoạt động.

Sự tăng trưởng vượt bậc của các cơ sở dữ liệu (CSDL) trong cuộc sống như: thương mại, quản lý đã làm nảy sinh và thúc đẩy sự phát triển của kỹ thuật thu thập, lưu trữ, phân tích và khai phá dữ liệu… không chỉ bằng các phép toán đơn giản thông thường như: phép đếm, thống kê… mà đòi hỏi một cách xử lý thông minh hơn, hiệu quả hơn. Các kỹ thuật cho phép ta khai thác được tri thức hữu dụng từ CSDL (lớn) được gọi là các kỹ thuật **Khai phá dữ liệu** (datamining). Đồ án nghiên cứu về những khái niệm cơ bản về khai phá dữ liệu, luật kết hợp và ứng dụng thuật toán khai phá luật kết hợp trong CSDL lớn.

Cấu trúc của đồ án được trình bày như sau:

CHƯƠNG 1: TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU VÀ PHÁT HIỆN TRI THỨC

Trình bày kiến thức tổng quan về khai thác và xử lý thông tin.

Khái niệm về luật kết hợp và các phương pháp khai phá luật kết hợp

Trình bày về thuật toán Apriori và một số thuật toán khai phá luật kết hợp

CHƯƠNG 2: MÔ HÌNH TÌM KIẾM THÔNG TIN

Trình bày các thành phần cơ bản của một search engine

Trình bày nguyên lý hoạt động của search engine và một số giải thuật tìm kiếm của search engine

CHƯƠNG 3: ỨNG DỤNG, THỬ NGHIỆM KHAI PHÁ DỮ LIỆU VIỆC LÀM TÍCH HỢP TỪ CÁC WEBSITE TUYỂN DỤNG

Nội dung của chương là áp dụng kỹ thuật khai phá dữ liệu vào bài toán tìm xu hướng chọn ngành nghề của các ứng viên và tuyển dụng của của các doanh nghiệp.

Cuối cùng là kết luận lại những kết quả đạt được của đề tài và hướng phát triển tương lai.

Chương 1: TỔNG QUAN VỀ KHAI PHÁ DỮ LIỆU VÀ PHÁT HIỆN TRI THỨC

I. Tổng quan về khai phá dữ liệu

1. Tổ chức và khai thác cơ sở dữ liệu truyền thống

Việc dùng các phương tiện tin học để tổ chức và khai thác cơ sở dữ liệu (CSDL) đã được phát hiện từ những năm 60 của thế kỷ trước. Từ đó cho đến nay, rất nhiều CSDL đã được tổ chức, phát triển và khai thác ở mọi quy mô và các lĩnh vực hoạt động của con người và xã hội. Theo như đánh giá cho thấy, lượng thông tin trên thế giới cứ sau 20 tháng lại tăng lên gấp đôi. Kích thước và số lượng CSDL thậm chí còn tăng nhanh hơn. Với sự phát triển của công nghệ điện tử, sự phát triển mạnh mẽ của công nghệ phần cứng tạo ra các bộ nhớ có dung lượng lớn, bộ xử lý có tốc độ cao cùng với sự phát triển của các hệ thống viễn thông, người ta đã và đang xây dựng các hệ thống thông tin nhằm tự động hoá mọi hoạt động của con người. Điều này đã tạo ra một dòng dữ liệu tăng lên không ngừng vì ngay cả những hoạt động đơn giản như gọi điện thoại, tra cứu sách trong thư viện, ... đều được thực hiện thông qua máy tính. Cho đến nay, số lượng CSDL đã trở nên khổng lồ bao gồm các CSDL cực lớn cỡ gigabytes và thậm chí terabytes lưu trữ các dữ liệu kinh doanh ví dụ như dữ liệu thông tin khác hàng , dữ liệu bán hàng, dữ liệu các tài khoản, ... Nhiều hệ quản trị CSDL mạnh với các công cụ phong phú và thuận tiện đã giúp con người khai thác có hiệu quả nguồn tài nguyên dữ liệu. Mô hình CSDL quan hệ và ngôn ngữ vấn đáp chuẩn (SQL) đã có vai trò hết sức quan trọng trong việc tổ chức và khai thác CSDL. Cho đến nay, không một tổ chức nào sử dụng tin học trong công việc mà không sử dụng các hệ quản trị CSDL và các hệ công cụ báo cáo, ngôn ngữ hỏi đáp nhằm khai thác CSDL phục vụ cho các hoạt động tác nghiệp của mình. Cùng với việc tăng không ngừng khối lượng dữ liệu, các hệ thống thông tin cũng được chuyên môn hoá, phân chia theo lĩnh vực ứng dụng như sản xuất, tài chính, hoạt động kinh doanh, ... Như vậy bên cạnh chức năng khai thác dữ liệu có tính chất tác nghiệp, sự thành công trong công việc không còn là năng suất của các hệ thống thông tin nữa mà là tính linh hoạt và sẵn sàng đáp lại những yêu cầu trong thực tế, CSDL cần đem lại những "tri thức" hơn là chính những dữ liệu trong đó. Các quyết định cần phải có càng nhanh càng tốt và phải chính xác dựa trên những dữ liệu sẵn có trong khi khối lượng dữ liệu cứ sau 20 tháng lại tăng gấp đôi làm ảnh hưởng đến thời gian ra quyết định cũng như khả năng hiểu hết được nội dung dữ liệu. Lúc này, các mô hình CSDL truyền thống và ngôn ngữ SQL đã cho thấy không có khả năng thực hiện công việc này. Để lấy thông tin có tính "tri thức" trong khối dữ liệu khổng lồ này, người ta đã tìm ra

những kỹ thuật có khả năng hợp nhất các dữ liệu từ các hệ thống giao dịch khác nhau, chuyển đổi thành một tập hợp các CSDL ổn định, có chất lượng được sử dụng chỉ cho riêng một vài mục đích nào đó. Các kỹ thuật đó gọi chung là kỹ thuật tạo kho dữ liệu (data warehousing) và môi trường các dữ liệu có được gọi là các kho dữ liệu (data warehouse).

Nhưng chỉ có kho dữ liệu thôi chưa đủ để có tri thức. Các kho dữ liệu được sử dụng theo một số cách như:

Theo cách khai thác truyền thống: tức là kho dữ liệu được sử dụng để khai thác các thông tin bằng các công cụ truy vấn và báo cáo.

Các kho dữ liệu được sử dụng để hỗ trợ cho phân tích trực tuyến (OLAP-OnLine Analytical Processing): Việc phân tích trực tuyến có khả năng phân tích dữ liệu, xác định xem giả thuyết đúng hay sai. Tuy nhiên, phân tích trực tuyến lại không có khả năng đưa ra các giả thuyết.

Công nghệ khai phá dữ liệu (data mining) ra đời đáp ứng những đòi hỏi trong khoa học cũng như trong hoạt động thực tiễn. Đây chính là một ứng dụng chính của kho dữ liệu.

2. Tổng quan về kỹ thuật phát hiện tri thức và khai phá dữ liệu (KDD – Knowledge Discovery and Data Mining)

2.1 Phát hiện tri thức và khai phá dữ liệu là gì?

Nếu cho rằng các điện tử và các sóng điện tử chính là bản chất của công nghệ điện tử truyền thống thì dữ liệu, thông tin và tri thức hiện đang là tiêu điểm của một lĩnh vực mới trong nghiên cứu và ứng dụng về phát hiện tri thức (Knowledge Discovery) và khai phá dữ liệu (Data Mining).

Thông thường chúng ta coi dữ liệu như một dãy các bit, hoặc các số và các ký hiệu, hoặc các "đối tượng" với một ý nghĩa nào đó khi được gửi cho một chương trình dưới một dạng nhất định. Chúng ta sử dụng các bit để đo lường các thông tin và xem nó như là các dữ liệu đã được lọc bỏ các dư thừa, được rút gọn tới mức tối thiểu để đặc trưng một cách cơ bản cho dữ liệu. Chúng ta có thể xem tri thức như là các thông tin tích hợp, bao gồm các sự kiện và các mối quan hệ giữa chúng. Các mối quan hệ này có thể được hiểu ra, có thể được phát hiện, hoặc có thể được học. Nói cách khác, tri thức có thể được coi là dữ liệu có độ trừu tượng và tổ chức cao.

Phát hiện tri thức trong các cơ sở dữ liệu là một qui trình nhận biết các mẫu hoặc các mô hình trong dữ liệu với các tính năng: hợp thức, mới, khả ích, và có thể hiểu được. Còn khai thác dữ liệu là một bước trong qui trình phát hiện tri thức gồm có các thuật toán khai thác dữ liệu chuyên dùng dưới một số qui định

về hiệu quả tính toán chấp nhận được để tìm ra các mẫu hoặc các mô hình trong dữ liệu. Nói một cách khác, mục đích của phát hiện tri thức và khai phá dữ liệu chính là tìm ra các mẫu và/hoặc các mô hình đang tồn tại trong các cơ sở dữ liệu nhưng vẫn còn bị che khuất bởi hàng núi dữ liệu.

Định nghĩa: "KDD là quá trình không tầm thường nhận ra những mẫu có giá trị, mới, hữu ích tiềm năng và hiểu được trong dữ liệu".

Còn các nhà thống kê thì xem Khai phá dữ liệu như là một qui trình phân tích được thiết kế để thăm dò một lượng cực lớn các dữ liệu nhằm phát hiện ra các mẫu thích hợp và/hoặc các mối quan hệ mang tính hệ thống giữa các biến và sau đó sẽ hợp thức hoá các kết quả tìm được bằng cách áp dụng các mẫu đã phát hiện được cho các tập con mới của dữ liệu. Qui trình này bao gồm ba giai đoạn cơ bản: thăm dò, xây dựng mô hình hoặc định nghĩa mẫu, hợp thức/kiểm chứng.

2.2 Quy trình phát hiện tri thức

Qui trình phát hiện tri thức được mô tả tóm tắt trên Hình 1:

Hình 1: quá trình phát hiện tri thức

Bước thứ nhất: Hình thành, xác định và định nghĩa bài toán. Là tìm hiểu lĩnh vực ứng dụng từ đó hình thành bài toán, xác định các nhiệm vụ cần phải hoàn thành. Bước này sẽ quyết định cho việc rút ra được các tri thức hữu ích và cho phép chọn các phương pháp khai phá dữ liệu thích hợp với mục đích ứng dụng và bản chất của dữ liệu.

Bước thứ hai: Thu thập và tiền xử lý dữ liệu. Là thu thập và xử lý thô, còn được gọi là tiền xử lý dữ liệu nhằm loại bỏ nhiễu, xử lý việc thiếu dữ liệu, biến đổi dữ liệu và rút gọn dữ liệu nếu cần thiết, bước này thường chiếm nhiều thời gian nhất trong toàn bộ qui trình phát hiện tri thức.

Bước thứ ba: Khai phá dữ liệu, rút ra các tri thức. Là khai phá dữ liệu, hay nói cách khác là trích ra các mẫu và/hoặc các mô hình ẩn dưới các dữ liệu. Giai đoạn này rất quan trọng, bao gồm các công đoạn như: chức năng, nhiệm vụ và mục đích của khai phá dữ liệu, dùng phương pháp khai phá nào?

Bước thứ tư: Sử dụng các tri thức phát hiện được. Là hiểu tri thức đã tìm được, đặc biệt là làm sáng tỏ các mô tả và dự đoán. Các bước trên có thể lặp đi lặp lại một số lần, kết quả thu được có thể được lấy trung bình trên tất cả các lần thực hiện.

Tóm lại: KDD là một quá trình chiết xuất ra tri thức từ kho dữ liệu mà trong đó khai phá dữ liệu là công đoạn quan trọng nhất.

2.3 Các phương pháp khai phá dữ liệu

KDD bao gồm hai yếu tố quan trọng không thể thiếu được là Dự đoán (Prediction) và Mô tả (Description)

Dự đoán: Đòi hỏi sử dụng một vài biến hoặc trường để dự đoán thông tin tiềm ẩn hoặc một giá trị tương lai của một biến thuộc tính mà ta quan tâm đến.

Mô tả: Tập trung là nổi bật lên mô hình kết quả mà con người có thể hiểu sâu về thông tin dữ liệu.

Với hai đích chính đã nêu ở trên, người ta thường sử dụng các phương pháp sau cho khai phá dữ liệu:

- Phân lớp, phân loại (Classification): Là việc học một hàm ánh xạ từ một mẫu dữ liệu vào một trong số các lớp đã được xác định trước đó.
- Hồi qui (Regression): Là việc học một hàm ánh xạ từ một mẫu dữ liệu thành một biến dự đoán có giá trị thực.
- Phân nhóm (Clustering): Là việc mô tả chung để tìm ra các tập hay các nhóm, loại mô tả dữ liệu. Các nhóm có thể tách nhau hoặc phân cấp.
- Tổng hợp (Summarization): Là công việc lên quan đến các phương pháp tìm kiếm một mô tả tập con dữ liệu, thường áp dụng trong việc phân tích dữ liệu có tính thăm dò và báo cáo tự động.
- Mô hình ràng buộc (Dependency modeling): Là việc tìm kiếm một mô hình mô tả sự phụ thuộc giữa các biến, thuộc tính theo hai mức: phụ thuộc cục bộ vào cấu trúc của mô hình, phụ thuộc vào thước đo, ước lượng của một định lượng nào đó.

- Dò tìm biến đổi và độ lệch (Change and Deviation Dectection): Chú ý vào những thay đổi quan trọng trong dữ liệu từ các giá trị chuẩn hoặc đã được xác định trước đó.
- Biểu diễn mô hình (Model Representation): Là việc dùng một ngôn ngữ L_ Language nào đó để mô tả các mẫu mô hình có thể khai phá được. Mô tả mô hình rõ ràng thì học máy sẽ tạo ra mẫu có mô hình chính xác cho dữ liệu. Tuy nhiên, nếu mô hình quá lớn thì khả năng dự đoán của học máy sẽ bị hạn chế. Như thế sẽ làm cho việc tìm kiếm phức tạp hơn cũng như hiểu được mô hình là không đơn giản.
- Kiểm định mô hình (Model Evaluation): Là việc đánh giá, ước lượng các mô hình chi tiết, chuẩn trong quá trình xử lý và phát hiện tri thức với sự ước lượng có dự báo chính xác hay không và có thoả mãn cơ sở logic hay không? Ước lượng phải được đánh giá chéo (cross validation) với việc mô tả đặc điểm bao gồm dự báo chính xác, tính mới lạ, tính hữu ích, tính hiểu được phù hợp với các mô hình. Hai phương pháp logic và thống kê chuẩn có thể sử dụng trong mô hình kiểm định.
- Phương pháp tìm kiếm (Search Method):Gồm có hai thành phần: (1) – Trong bảng tham biến (phạm vi tìm kiếm tham số) thuật toán phải tìm kiếm các tham số tronng phạm vi các chuẩn của mô hình kiểm định rồi tối ưu hoá và đưa ra tiêu chí (quan sát) dữ liệu và biểu diễn mô hình đã định. (2) – Mô hình tìm kiếm, xuất hiện như một đường vòng trên toàn bộ phương pháp tìm kiếm, biểu diễn mô hình phải thay đổi sao cho các hệ mô hình phải thay đổi sao cho các hệ gia phả mô hình phải được thông qua.

2.4 Các lĩnh vực liên quan đến phát hiện tri thức và khai phá dữ liệu

Phát hiện tri thức và khai phá dữ liệu liên quan đến nhiều ngành, nhiều lĩnh vực: thống kê, trí tuệ nhân tạo, cơ sở dữ liệu, thuật toán học, tính toán song song và tốc độ cao, thu thập tri thức cho các hệ chuyên gia, quan sát dữ liệu... Đặc biệt phát hiện tri thức và khai phá dữ liệu rất gần gũi với lĩnh vực thống kê, sử dụng các phương pháp thống kê để mô hình dữ liệu và phát hiện các mẫu, luật... Ngân hàng dữ liệu (Data Warehousing) và các công cụ phân tích trực tuyến (OLAP) cũng liên quan rất chặt chẽ với phát hiện tri thức và khai phá dữ liệu.

Khai phá dữ liệu có nhiều ứng dụng trong thực tế. Một số ứng dụng điển hình như:

- Bảo hiểm, tài chính và thị trường chứng khoán: Phân tích tình hình tài chính và dự báo giá của các loại cổ phiếu trong thị trường chứng khoán. Danh mục vốn và giá, lãi suất, dữ liệu thẻ tín dụng, phát hiện gian lận, ...

- Phân tích dữ liệu và hỗ trợ ra quyết định.
- Điều trị y học và chăm sóc y tế: Một số thông tin về chuẩn đoán bệnh lưu trong các hệ thống quản lý bệnh viện. Phân tích mối liên hệ giữa các triệu chứng bệnh, chuẩn đoán và phương pháp điều trị (chế độ dinh dưỡng, thuốc, ...)
- Sản xuất và chế biến: Quy trình, phương pháp chế biến và xử lý sự cố.
- Text mining và Web mining: Phân lớp văn bản và các trang Web, tóm tắt văn bản,...
- Lĩnh vực khoa học: Quan sát thiên văn, dữ liệu gene, dữ liệu sinh vật học, tìm kiếm, so sánh các hệ gene và thông tin di truyền, mối liên hệ gene và một số bệnh di truyền, ...
- Mạng viễn thông: Phân tích các cuộc gọi điện thoại và hệ thống giám sát lỗi, sự cố, chất lượng dịch vụ, ...

II. Ứng dụng luật kết hợp vào khai phá dữ liệu

Việc dự đoán các thông tin có giá trị cao dựa trên số lượng dữ liệu lớn về nghiệp vụ càng ngày càng trở lên quan trọng đối với nhiều tổ chức, doanh nghiệp. Chẳng hạn, những vấn đề các nhà quản lý và kinh doanh cần biết là các kiểu mẫu hành vi mua hàng của các khách hàng, xu hướng kinh doanh, vv… Những thông tin này có thể học được từ những dữ liệu có sẵn.

Một trong những vấn đề khó khăn nhất trong việc khai phá dữ liệu trong CSDL là có một số vô cùng lớn dữ liệu cần được xử lý. Các tổ chức doanh nghiệp quy mô vừa có thể có từ hàng hàng trăm Megabyte đến vài Gigabyte dữ liệu thu thập được. Các ứng dụng khai phá dữ liệu thường thực hiện phân tích dữ liệu khá phức tạp, mất nhiều thời gian trong toàn bộ CSDL. Vì vậy, tìm một thuật toán nhanh và hiệu quả để xử lý khối lượng dữ liệu lớn là một thách thức lớn.

Phần này trình bày cơ sở lý thuyết của luật và luật kết hợp, khai phá dữ liệu dựa vào luật kết hợp, đồng thời trình bày một số thuật toán liên quan đến luật kết hợp.

1. Lý thuyết luật kết hợp

Từ khi nó được giới thiệu từ năm 1993, bài toán khai thác luật kết hợp nhận được rất nhiều sự quan tâm của nhiều nhà khoa học. Ngày nay việc khai thác các luật như thế vẫn là một trong những phương pháp khai thác mẫu phổ biến nhất trong việc khám phá tri thức và khai thác dữ liệu (KDD: Knowledge Discovery and Data Mining).

Một cách ngắn gọn, một luật kết hợp là một biểu thức có dạng: $X \Rightarrow Y$, trong đó X và Y là tập các trường gọi là item. Ý nghĩa của các luật kết hợp khá dễ nhận thấy: Cho trước một cơ sở dữ liệu D là tập các giao tác - trong đó mỗi giao tác T$\in$ D là tập các item - khi đó $X \Rightarrow Y$ diễn đạt ý nghĩa rằng bất cứ khi nào giao tác T có chứa X thì chắc chắn T có chứa Y. Độ tin cậy của luật (rule confidence) có thể được hiểu như xác suất điều kiện p(Y $\subseteq$ T | X $\subseteq$ T). Ý tưởng của việc khai thác các luật kết hợp có nguồn gốc từ việc phân tích dữ liệu mua hàng của khách và nhận ra rằng "Một khách hàng mua mặt hàng x1 và x2 thì sẽ mua mặt hàng y với xác suất là c%". Ứng dụng trực tiếp của các luật này trong các bài toán kinh doanh cùng với tính dễ hiểu vốn có của chúng – ngay cả đối với những người không phải là chuyên gia khai thác dữ liệu – làm cho luật kết hợp trở thành một một phương pháp khai thác phổ biến. Hơn nữa, luật kết hợp không chỉ bị giới hạn trong phân tích sự phụ thuộc lẫn nhau trong phạm vi các ứng dụng bán lẻ mà chúng còn được áp dụng thành công trong rất nhiều bài toán kinh doanh.

Việc phát hiện luật kết hợp giữa các mục (item) trên dữ liệu "giỏ" là bài toán rất đặc trưng của khai phá dữ liệu. Dữ liệu giỏ là dữ liệu bao gồm các mục được mua bởi khách hàng với các thông tin như ngày mua hàng, số lượng, giá cả, … Luật kết hợp chỉ ra tập các mục mà thường được mua nhất với cùng các tập mục khác.

Hiện nay, có nhiều thuật toán dùng cho việc phát hiện luật kết hợp. Tuy nhiên, vấn đề nảy sinh là số lần quét (duyệt) CSDL quá nhiều sẽ ảnh hưởng rất lớn đến hiệu quả và tính khả thi của thuật toán trên các CSDL lớn. Đối với các CSDL được lưu trên đĩa, phép duyệt CSDL sẽ gây ra số lần đọc đĩa rất lớn. Chẳng hạn một CSDL kích thước 1GB sẽ đòi hỏi khoảng 125000 lần đọc khối cho mỗi lần duyệt (với kích thước khối là 8KB). Nếu thuật toán có 10 lần duyệt thì sẽ gây ra1250000 lần đọc khối. Giả thiết thời gian đọc trung bình là 12ms một trang, thời gian cần thiết để thực hiện một thao tác I/O này là1250000*12ms hay sấp sỉ 4 tiếng đồng hồ !!!

Trong phần này, chúng ta xem xét một số định nghĩa, tính chất có liên quan đến luật và luật kết hợp. Đồng thời chúng ta tìm hiểu ý nghĩa của luật kết hợp.

1.1 Luật kết hợp

a) **Ý nghĩa luật kết hợp:** Luật kết hợp là một lãnh vực quan trọng trong khai thác dữ liệu. Luật kết hợp giúp chúng ta tìm được các mối liên hệ giữa các mục dữ liệu (items) của cơ sở dữ liệu. Trong môi trường mạng nhu cầu tìm việc trực tuyến đã trở thành xu hướng phát triển các website tuyển dụng ngày càng nhiều thông tin về người tìm việc và doanh nghiệp tuyển người ngày

càng nhiều do nhu cầu của xã hội, do đó chúng ta có thể tìm xu hướng tuyển dụng và nhu cầu việc làm để các nhà quản lý đưa ra nhu cầu việc làm của xã hội. Hay như trong ngành viễn thông, các loại dịch vụ cung cấp cho khách hàng ngày càng nhiều, do đó chúng ta có thể tìm mối liên kết giữa việc sử dụng các loại dịch vụ để phục vụ cho việc quảng cáo, tiếp thị. Ví dụ như để tìm hiểu thói quen sử dụng các dịch vụ viễn thông của khách hàng, người ta thường đặt câu hỏi "Những dịch vụ nào khách hàng thường hay sử dụng cùng lúc với nhau khi đăng ký sử dụng tại trung tâm chăm sóc khách hàng ?". Các kết quả nhận được có thể dùng cho việc tiếp thị dịch vụ như liệt kê các dịch vụ khách hàng hay sử dụng cùng lúc nằm gần nhau, hoặc khuyến mãi dịch vụ kèm theo….

b) **Định nghĩa luật kết hợp:** Cho một tập I = {I1, I2, ...,Im} là tập gồm m khoản mục (item), còn được gọi là các thuộc tính (attribute). Các phần tử trong I là phân biệt nhau. $X \subseteq I$ được gọi là tập mục (itemset). Nếu lực lượng của X bằng k (tức là |X| = k) thì X được gọi là k-itemset.

Một giao dịch (transaction) T được định nghĩa như một tập con (subset) của các khoản mục trong I ($T \subseteq I$). Tương tự như khái niệm tập hợp, các giao dịch không được trùng lặp, nhưng có thể nới rộng tính chất này của tập hợp và trong các thuật toán sau này, người ta đều giả thiết rằng các khoản mục trong một giao dịch và trong tất cả các tập mục (item set) khác, có thể coi chúng đã được sắp xếp theo thứ tự từ điển của các item.

Gọi D là CSDL của n giao dịch và mỗi giao dịch được đánh nhãn với một định danh duy nhất (Unique Transasction IDentifier-TID). Nói rằng, một giao dịch $T \in D$ hỗ trợ (support) cho một tập $X \subseteq I$ nếu nó chứa tất cả các item của X, nghĩa là $X \subseteq T$, trong một số trường hợp người ta dùng ký hiệu T(X) để chỉ tập các giao dịch hỗ trợ cho X. Kí hiệu support(X) (hoặc supp(X), s(X)) là tỷ lệ phần trăm của các giao dịch hỗ trợ X trên tổng các giao dịch trong D, nghĩa là:

$$\text{supp(X)} = \frac{\left| T \in D \middle| X \subseteq T \right|}{|D|} \%$$

Ví dụ về cơ sở dữ liệu D (dạng giao dịch) : **I** = {A, B, C, D, E}, **T** = {1, 2, 3, 4, 5, 6}. Thông tin về các giao dịch cho ở bảng sau :

Định danh giao dịch (TID)	Tập mục (itemset)
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

Bảng 1: Ví dụ về một cơ sở dữ liệu dạng giao dịch – D

Ta có: supp({A }) = 4/6 (%)= 66.67 %;

supp({ABDE}) = 3/6 =50%;

supp({ABCDE}) = 1/6 = 16.67%; ...

Tập phổ biến (frequent itemset):

Support tối thiểu minsup$\in$(0, 1] (Minimum Support) là một giá trị cho trước bởi người sử dụng. Nếu tập mục X $\subseteq$ I có supp(X) $\subseteq$ minsup thì ta nói X là một tập phổ biến-frequent itemset (hoặc large itemset). Một frequent itemset được sử dụng như một tập đáng quan tâm trong các thuật toán, ngược lại, những tập không phải frequent itemset là những tập không đáng quan tâm. Trong các trình bày sau này, ta sẽ sử dụng những cụm từ khác như "X có support tối thiểu", hay "X không có support tối thiểu" cũng để nói lên rằng X thỏa mãn hay không thỏa mãn support(X) $\subseteq$ minsupp.

Ví dụ: Với cơ sở dữ liệu D cho ở bảng 3, và giá trị ngưỡng minsupp = 50% sẽ liệt kê tất cả các tập phổ biến (frequent-itemset) như sau :

Các tập mục phổ biến	Độ hỗ trợ (supp) tương ứng
B	100% (6/6)
E, BE	83% (5/6)
A, C, D, AB, AE, BC, BD, ABE	67% (4/6)
AD, CE, DE, ABD, ADE, BCE, BDE	50% (3/6)

Bảng 2 : Các tập phổ biến trong cơ sở dữ liệu ở bảng 1

với độ hỗ trợ tối thiểu 50%

<u>Một số tính chất (TC) liên quan đến các frequent itemset</u>:

TC 1. support cho tất cả các subset: nếu A $\subseteq$B, A, B là các itemset thì supp(A) $\geq$ supp(B) vì tất cả các giao dịch của D support B thì cũng support A.

TC 2. Nếu một item A không có support tối thiểu trên D nghĩa là support(A) < minsupp thì một superset B của A sẽ không phải là một frequent vì support(B) $\leq$ support(A) < minsup.

TC 3. Nếu item B là frequent trên D, nghĩa là support(B) $\geq$ minsup thì mọi subset A của B là frequent trên D vì support(A) $\geq$ support(B) > minsup.

<u>Định nghĩa luật kết hợp</u>:

Một luật kết hợp có dạng R: X $\Rightarrow$ Y, trong đó X, Y là các itemset, X, Y $\subseteq$ I và X $\cap$Y = $\varnothing$. X được gọi là tiên đề và Y được gọi là hệ quả của luật.

Luật X $\Rightarrow$ Y tồn tại một độ hỗ trợ support - supp. Supp(X $\Rightarrow$ Y) được định nghĩa là khả năng mà tập giao dịch hỗ trợ cho các thuộc tính có trong cả X lẫn Y, nghĩa là:

Support(X$\Rightarrow$Y) = support(X$\cup$Y).

Luật X $\Rightarrow$ Y tồn tại một độ tin cậy c (confidence - conf). Conf c được định nghĩa là khả năng giao dịch T hỗ trợ X thì cũng hỗ trợ Y. Nói cách khác c biểu thị số phần trăm giao dịch có chứa luôn A trong số những giao dịch có chứa X.

Ta có công thức tính conf c như sau:

$$\text{conf}(X \Rightarrow Y) = p(Y \subseteq T \mid X \subseteq T) = \frac{p(Y \subseteq T \wedge X \subseteq T)}{p(X \subseteq T)} = \frac{\sup p(X \cup Y)}{\sup p(X)} \%$$

Ta nói rằng, luật $X \Rightarrow$Y là **thoả trên D** nếu với một support tối thiểu minsup và một ngưỡng cofidence tối thiểu minconf cho trước nào đó mà:

Support(X $\Rightarrow$ Y) $\geq$ minsup và confidence(X$\Rightarrow$ Y) $\geq$ minconf

Chú ý rằng, nếu luật $X \Rightarrow Y$ mà thoả trên D thì cả X và Y đều phải là các Frequent Itemset trên D và khi xét một luật có thoả hay không, thì cả support và confidence của nó đều phải quan tâm, vì một luật có thể có *confidence = 100% > minconf* nhưng có thể là nó không đạt support tối thiểu minsup.

1.2 Một số tính chất của luật kết hợp

Trước hết ta phải giả sử rằng với luật $X \Rightarrow Y$, X có thể là rỗng, còn Y phải luôn khác rỗng và $X \neq Y$ vì nếu không thì:

$$\text{confidence}(X \Rightarrow Y) = \frac{\text{support}(X \cup Y)}{\text{support}(X)} = 1$$

<u>Ta có các tính chất sau</u> :

1) Nếu $X \Rightarrow Z$ và $Y \Rightarrow Z$ là thoả trên D, thì không nhất thiết là $X \cup Y \Rightarrow Z$.

Để ý đến trường hợp $X \cap Y = \varnothing$ và các giao dịch trên D hỗ trợ Z nếu và chỉ nếu chúng hỗ trợ X hoặc hỗ trợ Y. Khi đó, $\text{support}(X \cup Y) = 0$ và $\text{cofidence}(X \cup Y) = 0$.

Tương tự ta cũng có : Nếu $X \Rightarrow Y$ và $X \Rightarrow Z$ không thể suy ra $X \Rightarrow Y \cup Z$.

2) Nếu luật $X \cup Y \Rightarrow Z$ là thoả trên D thì $X \Rightarrow Z$ và $Y \Rightarrow Z$ có thể không thoả trên D.

Chẳng hạn, khi Z là có mặt trong một giao dịch chỉ nếu cả X và Y đều có mặt trong giao dịch đó, nghĩa là $\text{support}(X \cup Y) = \text{support}(Z)$. Nếu support cho X và Y lớn hơn $\text{support}(X \cup Y)$, thì 2 luật trên sẽ không có confidence yêu cầu. Tuy nhiên, nếu $X \Rightarrow Y \cup Z$ là thoả trên D thì có thể suy ra $X \Rightarrow Y$ và $X \Rightarrow Z$ cũng thoả trên D Vì $\text{support}(XY) \geq \text{support}(XYZ)$ và $\text{support}(XZ) \geq \text{support}(XYZ)$.

3) Nếu $X \Rightarrow Y$ và $Y \Rightarrow Z$ là thoả trên D thì không thể khẳng định rằng $X \Rightarrow Z$ cũng giữ được trên D.

Giả sử $T(X) \subset T(Y) \subset T(Z)$ và $\text{confidence}(X \Rightarrow Y) = \text{confidence}(Y \Rightarrow Z) = \text{minconf}$. Khi đó ta có $\text{confidence}(X \Rightarrow Z) = \text{minconf}^2 < \text{minconf}$ vì $\text{minconf} < 1$, nghĩa là luật $X \Rightarrow Z$ không có cofidence tối thiểu.

4) Nếu luật $A \Rightarrow$ (L-A) không có confidence tối thiểu thì cũng không có luật nào trong các luật $B \Rightarrow$ (L-B) có confidence tối thiểu trong đó L-A, B là các intemset và $B \subseteq A$.

Thật vậy, theo tính chất TC1, vì $B \subseteq A$. Nên $\text{support}(B) \geq \text{support}(A)$ và theo định nghĩa của confidence, ta có :

$$\text{confidence}(B \Rightarrow (L\text{-}B)) = \frac{\sup port(L)}{\sup port(B)} \leq \frac{\sup port(L)}{\sup port(A)} < minconf.$$

Cũng vậy, nếu luật (L-C) $\Rightarrow$C là thoả trên D, thì các luật (L-K) $\Rightarrow$K với K$\subseteq$C và K$\neq\varnothing$ cũng thoả trên D.

Bài toán khai phá luật kết hợp:

Có thể diễn đạt một bài toán khai phá luật kết hợp như sau[2][3][8]:

Cho một tập các item I, một cơ sở dữ liệu giao dịch D, ngưỡng support tối thiểu minsup, ngưỡng confidence tối thiểu minconf, tìm tất cả các luật kết hợp X$\Rightarrow$Y trên D sao cho: support(X$\Rightarrow$Y) $\geq$minsup và confidence(X$\Rightarrow$Y) $\geq$minconf.

1.3 Phân loại luật kết hợp

Tuỳ theo ngữ cảnh các thuộc tính dữ liệu cũng như phương pháp trong các thuật toán mà người ta có thể phân bài toán khai phá luật kết hợp ra nhiều nhóm khác nhau. Chẳng hạn, nếu giá trị của các item chỉ là các giá trị theo kiểu boolean thì người ta gọi là khai phá luật kết hợp boolean (Mining Boolean Association Rules), còn nếu các thuộc tính có tính đến khoảng giá trị của nó (như thuộc tính phân loại hay thuộc tính số lượng chẳng hạn) thì người ta gọi nó là khai phá luật kết hợp định lượng (Mining Quantitative Association Rules)… Ta sẽ xem xét cụ thể các nhóm đó.

Lĩnh vực khai thác luật kết hợp cho đến nay đã được nghiên cứu và phát triển theo nhiều hướng khác nhau. Có những đề xuất nhằm cải tiến tốc độ thuật toán, có những đề xuất nhằm tìm kiếm luật có ý nghĩa hơn, v. v. và có một số hướng chính sau đây.

Luật kết hợp nhị phân (binary association rule hoặc boolean association rule): là hướng nghiên cứu đầu tiên của luật kết hợp. Hầu hết các nghiên cứu ở thời kỳ đầu về luật kết hợp đều liên quan đến luật kết hợp nhị phân. Trong dạng luật kết hợp này, các mục (thuộc tính) chỉ được quan tâm là có hay không xuất hiện trong giao tác của cơ sở dữ liệu chứ không quan tâm về "mức độ" xuất hiện. Có nghĩa là việc gọi 10 cuộc điện thoại và 1 cuộc được xem là giống nhau. Thuật toán tiêu biểu nhất khai phá dạng luật này là thuật toán **Apriori** và các biến thể của nó. Đây là dạng luật đơn giản và các luật khác cũng có thể chuyển về dạng luật này nhờ một số phương pháp như rời rạc hoá, mờ hoá, v. v. . . Một ví dụ về dạng luật này : "gọi liên tỉnh='yes' AND gọi di động="yes" $\Rightarrow$ gọi quốc tế='yes' AND gọi dịch vụ 108 = 'yes', với độ hỗ trợ 20% và độ tin cậy 80%"

Luật kết hợp có thuộc tính số và thuộc tính hạng mục (quantitative and categorial association rule): Các thuộc tính của các cơ sở dữ liệu thực tế có kiểu rất đa dạng (nhị phân – binary, số – quantitative, hạng mục – categorial,. v. v).

Để phát hiện luật kết hợp với các thuộc tính này, các nhà nghiên cứu đã đề xuất một số phương pháp rời rạc hoá nhằm chuyển dạng luật này về dạng nhị phân để có thể áp dụng các thuật toán đã có. Một ví dụ về dạng luật này "phương thức gọi = 'Tự động' AND giờ gọi ? '23:00:39..23:00:59' AND Thời gian đàm thoại? '200.. 300' ⇒ gọi liên tỉnh ='có' , với độ hỗ trợ là 23. 53% , và độ tin cậy là 80%".

Luật kết hợp tiếp cận theo hướng tập thô (mining association rules base on rough set): Tìm kiếm luật kết hợp dựa trên lý thuyết tập thô.

Luật kết nhiều mức (multi-level association rule): Với cách tiếp cận theo luật này sẽ tìm kiếm thêm những luật có dạng " mua máy tính PC ⇒ mua hệ điều hành AND mua phần mềm tiện ích văn phòng, …" thay vì chỉ những luật quá cụ thể như " mua máy tính IBM PC ⇒ mua hệ điều hành Microsoft Windows AND mua phần mềm tiện ích văn phòng Microsoft Office, …". Như vậy dạng luật đầu là dạng luật tổng quát hoá của dạng luật sau và tổng quát theo nhiều mức khác nhau.

Luật kết hợp mờ (fuzzy association rule): Với những hạn chế còn gặp phải trong quá trình rời rạc hoá các thuộc tính số (quantitave attributes), các nhà nghiên cứu đã đề xuất luật kết hợp mờ nhằm khắc phục các hạn chế trên và chuyển luật kết hợp về một dạng tự nhiên hơn, gần gũi hơn với người sử dụng một ví dụ của dạng này là : "thuê bao tư nhân = 'yes' AND thời gian đàm thoại lớn AND cước nội tỉnh = 'yes' ⇒ cước không hợp lệ = 'yes', với độ hỗ trợ 4% và độ tin cậy 85%". Trong luật trên, điều kiện thời gian đàm thoại lớn ở vế trái của luật là một thuộc tính đã được mờ hoá.

Luật kết với thuộc tính được đánh trọng số (association rule with weighted items): Trong thực tế, các thuộc tính trong cơ sở dữ liệu không phải lúc nào cũng có vai trò như nhau. Có một số thuộc tính được chú trọng hơn và có mức độ quan trọng cao hơn các thuộc tính khác. Ví dụ khi khảo sát về doanh thu hàng tháng, thông tin về thời gian đàm thoại, vùng cước là quan trọng hơn nhiều so với thông tin về phương thức gọi.. . Trong quá trình tìm kiếm luật, chúng ta sẽ gán thời gian gọi, vùng cước các trọng số lớn hơn thuộc tính phương thức gọi. Đây là hướng nghiên cứu rất thú vị và đã được một số nhà nghiên cứu đề xuất cách giải quyết bài toán này. Với luật kết hợp có thuộc tính được đánh trọng số, chúng ta sẽ khai thác được những luật "hiếm" (tức là có độ hỗ trợ thấp, nhưng có ý nghĩa đặc biệt hoặc mang rất nhiều ý nghĩa).

Khai thác luật kết hợp song song (parallel mining of association rules): Bên cạnh khai thác luật kết hợp tuần tự, các nhà làm tin học cũng tập trung vào nghiên cứu các thuật giải song song cho quá trình phát hiện luật kết hợp. Nhu

cầu song song hoá và xử lý phân tán là cần thiết bởi kích thước dữ liệu ngày càng lớn hơn nên đòi hỏi tốc độ xử lý cũng như dung lượng bộ nhớ của hệ thống phải được đảm bảo. Có rất nhiều thuật toán song song khác nhau đã đề xuất để có thể không phụ thuộc vào phần cứng. Bên cạnh những nghiên cứu về những biến thể của luật kết hợp, các nhà nghiên cứu còn chú trọng đề xuất những thuật toán nhằm tăng tốc quá trình tìm kiếm tập phổ biến từ cơ sở dữ liệu.

Ngoài ra, còn có một số hướng nghiên cứu khác về khai thác luật kết hợp như: Khai thác luật kết hợp trực tuyến, khai thác luật kết hợp được kết nối trực tuyến đến các kho dữ liệu đa chiều (Multidimensional data, data warehouse) thông qua công nghệ OLAP (Online Analysis Processing), MOLAP (multidimensional OLAP), ROLAP (Relational OLAP), ADO (Active X Data Object) for OLAP..v.v.

1.4 Đặc tả bài toán khai phá dữ liệu

Với các định nghĩa trên, ta có thể mô tả cấu trúc cơ bản của một thuật toán khai phá luật kết hợp. Mặc dù, trong thực tế, các thuật toán có thể có sự khác nhau về một số vấn đề, nhưng về cơ bản thì chúng tuân theo một lược đồ chung. Có thể tóm tắt lược đồ qua 2 giai đoạn chính sau:

Khai phá tất cả các tập phổ biến-Frequent itemset (Large itemset)

Như đã lưu ý trước đây, số lượng các tập frequent có khả năng tương đương với kích thước mũ của tập các item, trong đó hàm mũ tăng theo số các item. Phương pháp cơ bản trong mỗi thuật toán là tạo một tập các itemset gọi là candidate với hi vọng rằng nó là frequent.

Điều mà bất kì thuật toán nào cũng phải quan tâm là làm sao để tập các candidate này càng nhỏ càng tốt vì nó liên quan chi phí bộ nhớ để lưu trữ các tập candidate này chi phí thời gian cho việc kiểm tra nó là một Frequent itemset hay không.

Để tìm ra những candidate itemset là frequent với các support cụ thể của nó là bao nhiêu thì support của mỗi tập candidate phải được đếm bởi mỗi giai đoạn trên CSDL (tức là thực hiện một phép duyệt trên từng giao dịch của cơ sở dữ liệu để tính giao dịch support cho mỗi candidate itemset).

Công việc khai phá các Frequent Itemset được thực hiện lặp đi lặp lại qua một giai đoạn (pass) nhằm mục đích nhận được kết quả cuối cùng là mỗi Frequen Itemset biểu thị tốt nhất sự tương quan giữa các item trong cơ sở dữ liệu giao dịch D.

Khai phá luật kết hợp (sinh ra các luật kết hợp mạnh từ các tập mục phổ biến)

Sau khi xác định được tập Frequent Itemset cuối cùng, người ta thực hiện tiếp thuật toán sinh ra các luật dựa trên mỗi frequent itemset này đồng thời xác định luôn confidence của chúng trên cơ sở các số đếm support của mỗi frequent itemset và subset của mỗi frequent itemset. Với mỗi frequent itemset X, mỗi subset riêng biệt của nó là được chọn như là tiền đề của luật và các item còn lại thì được đưa vào hệ quả của luật, do X chính nó là một frequent, và tất cả các subset của nó cũng là Frequent (theo tính chất **TC3** mục 1.1). Mỗi luật được sinh ra như trên có được chấp nhận hay không chấp nhận còn phụ thuộc vào mức confidence tối thiểu (minconf) mà người sử dụng chỉ ra. Một luật sẽ được coi là chấp nhận nếu confidence của nó lớn hơn hoặc bằng cofidence tối thiểu này. Theo tính chất TC4, mục 1.2, nếu một luật là không được chấp nhận thì không có một subset nào của tiền tố của nó là có thể cân nhắc để sinh thêm các luật khác.

Nói chung thì tư tưởng sinh ra luật kết hợp có thể mô tả như sau:

Nếu ABCD và AB là các frequent itemset thì ta có thể xác định xem luật $AB \Rightarrow CD$ có được xem là chấp nhận hay không bằng cách tính confidence của nó theo định nghĩa $conf = \frac{\sup port(ABCD)}{\sup port(AB)}$. Nếu conf $\geq$ minconf thì luật được coi là chấp nhận được (để ý rằng luật là thoả mãn yếu tố support vì support ($AB \Rightarrow CD$) = support(ABCD) $\geq$ minsup).

2. Các đặc trưng của luật kết hợp

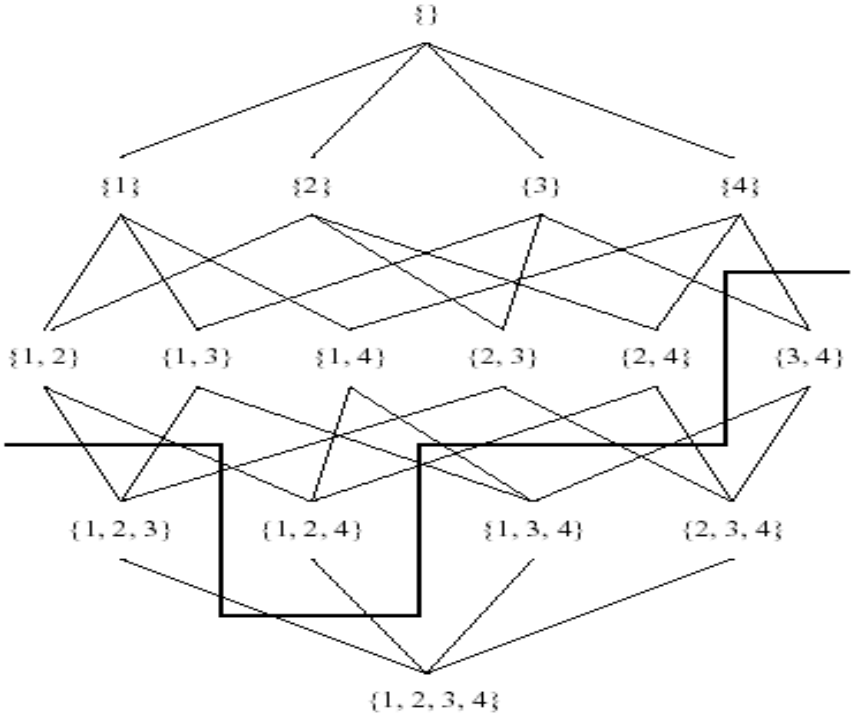
2.1 Không gian tìm kiếm luật:

Như đã giải thích trên đây, ta phải tìm tất cả các *itemset* thỏa ngưỡng minsupp. Với các ứng dụng thực tiễn, việc duyệt tất cả các tập con của I sẽ hoàn toàn thất bại vì không gian tìm kiếm quá lớn. Trên thực tế, sự tăng tuyến tính số lượng các *item* vẫn kéo theo sự tăng theo cấp lũy thừa các *itemset* cần xem xét. Với trường hợp đặc biệt I ={1,2,3,4}, ta có thể biểu diễn không gian tìm kiếm thành một lưới như trong hình 2.

Các tập phổ biến nằm trong phần trên của hình trong khi những tập không phổ biến lại nằm trong phần dưới. Mặc dù không chỉ ra một cách tường minh các giá trị hỗ trợ cho mỗi *itemset* nhưng ta giả sử rằng đường biên đậm trong hình phân chia các tập phổ biến và tập không phổ biến. Sự tồn tại của đường biên như vậy không phụ thuộc vào bất kỳ cơ sở dữ liệu D và minsupp nào. Sự tồn tại của nó chỉ đơn thuần được đảm bảo bởi tính chặn dưới của *itemset* thỏa ngưỡng minsupp.

Nguyên lý cơ bản của các giải thuật thông thường là sử dụng đường biên này để thu hẹp không gian tìm kiếm một cách có hiệu quả. Khi đường biên được

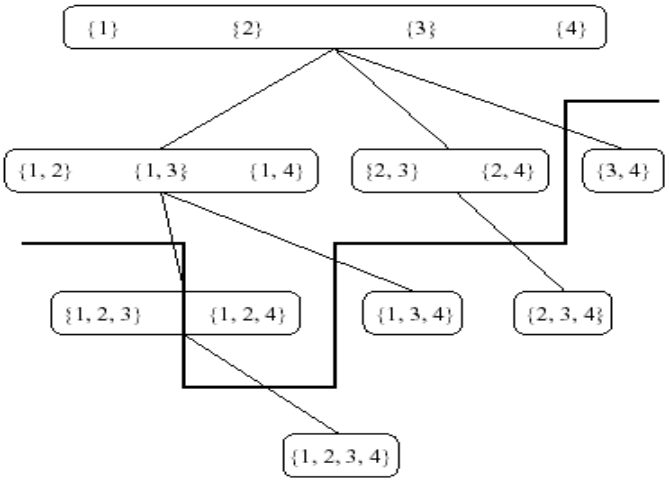
tìm thấy, chúng ta có thể giới hạn trong việc xác định các giá trị hỗ trợ của các *itemset* phía trên đường biên và bỏ qua các *itemset* phía dưới đường biên.



Hình 2: Dàn cho tập I = {1,2,3,4}

Cho ánh xạ: $I \rightarrow \{1,\ldots, |I|\}$ là một phép ánh xạ từ các phần tử $x \in I$ ánh xạ 1-1 vào các số tự nhiên. Bây giờ, các phần tử có thể được xem là có thứ tự hoàn toàn trên quan hệ "<" giữa các số tự nhiên. Hơn nữa, với $X \subseteq I$, cho X.item: $\{1,\ldots,|X|\} \rightarrow I$: $n \mapsto X.item_n$ là một ánh xạ, trong đó $X.item_n$ là phần tử thứ n của các phần tử $x \in X$ sắp xếp tăng dần trên quan hệ "<". n-tiền tố của một *itemset* X với $n \leq |X|$ được định nghĩa bởi $P=\{X.itemm \mid 1 \leq m \leq n\}$.

Cho các lớp E(P), $P \subseteq I$ với $E(P) = \{X \subseteq I \mid |X| = |P|+1$ và P là một tiền tố của X} là các nút của một cây. Hai nút sẽ được nối với nhau bằng 1 cạnh nếu tất cả các *itemset* của lớp E có thể được phát sinh bằng cách kết 2 *itemset* của lớp cha E', ví dụ như trong hình 3.



Hình 3: Cây cho tập I = {1, 2, 3, 4}

Cùng với tính chặn dưới của *itemset* thỏa ngưỡng minsupp, điều này suy ra: Nếu lớp cha E' của lớp E không có tối thiểu hai tập phổ biến thì E cũng phải không chứa bất kỳ một tập phổ biến nào. Nếu gặp một lớp E' như vậy trong quá trình duyệt cây từ trên xuống thì ta đã tiến đến đường biên phân chia giữa tập phổ biến và không phổ biến. Ta không cần phải tìm tiếp phần sau đường biên này, tức là ta đã loại bỏ E và các lớp con của E trong không gian tìm kiếm. Thủ tục tiếp theo cho phép ta giới hạn một cách có hiệu quả số lượng các itemset cần phải duyệt. Ta chỉ cần xác định các support values của các *itemset* mà ta đã duyệt qua trong quá trình tìm kiếm đường biên giữa tập phổ biến và tập không phổ biến. Cuối cùng, chiến lược thực sự để tìm đường biên là do lựa chọn của chúng ta. Các hướng tiếp cận phổ biến hiện nay sử dụng cả tìm kiếm ưu tiên bề rộng (BFS) lẫn tìm kiếm ưu tiên chiều sâu (DFS). Với BFS, giá trị hỗ trợ của tất cả (k-1)-itemset được xác định trước khi tính giá trị hỗ trợ của *k-itemset*. Ngược lại, DFS duyệt đệ quy theo cấu trúc cây mô tả ở trên.

2.2 Độ hỗ trợ luật

Trong phần này, một *itemset* có khả năng là phổ biến và ta cần phải xác định độ hỗ trợ của nó trong quá trình duyệt dàn, được gọi là một *itemset* ứng viên. Một hướng tiếp cận phổ biến để xác định giá trị hỗ trợ của một *itemset* là đếm các thể hiện của nó trong cơ sở dữ liệu. Với mục đích đó, một biến đếm (counter) được tạo ra và khởi tạo bằng 0 cho mỗi *itemset* đang duyệt. Sau đó, quét qua tất cả các giao tác và khi tìm được một ứng viên là tập con của một giao tác thì tăng biến đếm của nó lên. Thông thường, tập con tạo ra và bảng tìm kiếm ứng cử viên được tích hợp và cài đặt bằng một *hashtree* hay một cấu trúc dữ liệu tương tự. Tóm lại, không phải tất cả các tập con của mỗi giao tác đều

được tạo ra mà chỉ những giao tác có chứa trong các ứng viên hoặc có một tiền tố chung với ít nhất một ứng cử viên mới được tạo ra.

Một cách tiếp cận khác để xác định giá trị hỗ trợ của các ứng viên là sử dụng giao tập hợp (set intersection). Một TID (Transaction IDentifier) là một khóa-biến nhận dạng giao tác duy nhất. Với một phần tử đơn, *tidlist* là tập hợp của các biến nhận dạng tương ứng với các giao tác có chứa phần tử này. Do đó, các *tidlist* cũng tồn tại cho mỗi *itemset* X và được biểu diễn bởi *X.tidlist. Tidlist* của một ứng viên $C = X \cup Y$ xác định bởi: $C.tidlist=X.tidlist \cap Y.tidlist$. Các *tidlist* được sắp xếp theo thứ tự tăng dần để các phép giao được hiệu quả.

Lưu ý rằng bằng cách dùng vùng đệm cho *tidlist* của các ứng viên phổ biến như là các kết quả trung gian, ta có thể tăng đáng kể tốc độ phát sinh *tidlist* cho các ứng viên tiếp theo. Cuối cùng, các độ hỗ trợ thực sự của ứng cử viên chính là |C.tlist|.

3. Một số giải thuật cơ bản khai phá các tập phổ biến

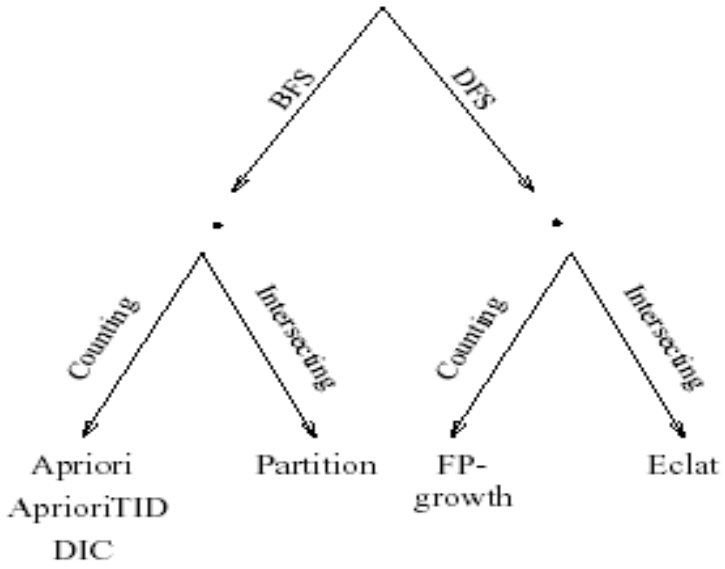
Phần này sẽ trình bày và hệ thống hóa một cách ngắn gọn các giải thuật đang được dùng phổ biến hiện nay để khai phá các tập phổ biến. Chúng sẽ được thực hiện dựa vào những nguyên tắc cơ bản của phần trước. Mục tiêu của chúng ta là thể hiện được những sự khác biệt giữa các cách tiếp cận khác nhau.

Các giải thuật mà ta xem xét trong bài này được hệ thống hóa như hình vẽ 4. Các giải thuật được phân loại dựa vào việc:

a) Duyệt theo không gian tìm kiếm (BFS, DFS)

b) Các định giá trị hỗ trợ của tập *item (itemset)*

c) Ngoài ra, một giải thuật có thể dùng một số các tối ưu khác để tăng tốc thêm.



Hình 4: Hệ thống hóa các giải thuật

3.1 Giải thuật BFS(BFS – Breadth first search)

Giải thuật phổ biến nhất của loại này là giải thuật Apriori, trong đó có trình bày tính chặn dưới của itemset thỏa ngưỡng minsupp. Giải thuật Apriori tạo ra việc sử dụng các tính chất này bằng việc tỉa bớt những ứng viên thuộc tập không phổ biến trước khi tính độ phổ biến của chúng. Cách tối ưu có thể thực hiện được vì các giải thuật tìm kiếm ưu tiên theo chiều rộng (BFS) bảo đảm rằng các giá trị hỗ trợ của các tập của một ứng viên đều được biết trước. Giải thuật Apriori đếm tất cả các ứng viên có k phần tử trong một lần đọc cơ sở dữ liệu. Phần cốt lõi của bài toán là xác định các ứng viên trong mỗi giao tác. Để thực hiện được mục đích này phải dựa vào một cấu trúc gọi là hashtree. Các item trong mỗi giao dịch được dùng để đi lần xuống trong cấu trúc hashtree. Bất cứ khi nào tới được nút lá của nó, nghĩa là ta đã tìm được một tập các ứng viên có cùng tiền tố được chứa trong giao dịch đó. Sau đó các ứng viên này sẽ được thực hiện tìm kiếm trong giao dịch mà nó đã được mã hóa trước thành ma trận bit. Trong trường hợp thành công biến đếm các ứng viên trong cây được tăng lên.

Giới thiệu bài toán:

Apriori là thuật toán được Rakesh Agrawal, Tomasz Imielinski, Arun Swami đề xuất lần đầu vào năm 1993. Bài toán được phát biểu: Tìm t có độ hỗ trợ s thỏa mãn $s \geq s_0$ và độ tin cậy $c \geq c_0$ (s_0, c_0 là hai ngưỡng do người dùng xác định và s_0=minsupp, c_0 =minconf) . Ký hiệu L_k tập các tập k - mục phổ biến, C_k tập các tập k-mục ứng cử (cả hai tập có: tập mục và độ hỗ trợ).

Bài toán đặt ra là:

Tìm tất cả các tập mục phổ biến với minsupp nào đó.

Sử dụng các tập mục phổ biến để sinh ra các luật kết hợp với độ tin cậy minconf nào đó.

Quá trình thực hiện (duyệt):

Thực hiện nhiều lần duyệt lặp đi lặp lại, trong đó tập (k-1) - mục được sử dụng cho việc tìm tập k-mục. Lần thứ nhất tìm tất cả các độ hỗ trợ của các mục, xác định mục phổ biến (mục thoả mãn độ hỗ trợ cực tiểu-minsupp). Giả sử tìm được L_1-mục phổ biến.

Các lần duyệt còn lại: Bắt đầu kết quả tìm được bước trước nó, sử dụng các tập mục mẫu (L_1) sinh ra các tập mục phổ biến tiềm năng (ứng cử) (giả sử L_2), tìm độ hỗ trợ thực sự. Mỗi lần duyệt ta phải xác định tập mục mẫu cho lần duyệt tiếp theo.

Thực hiện lặp để tìm L_3, ..., L_k cho đến khi không tìm thấy tập mục phổ biến nào nữa.

Chú ý:

Ứng dụng L_{k-1} để tìm L_k bao gồm hai bước chính:

Bước kết nối: tìm L_k là tập k-mục ứng được sinh ra bởi việc kết nối L_{k-1} với chính nó cho kết quả là C_k. Giả sử L_1, L_2 thuộc L_{k-1}. Ký hiệu L_i^j là mục thứ j trong L_i. Điều kiện là các tập mục hay các mục trong giao dịch có thứ tự. Bước kết nối như sau: Các thành phần L_{k-1} kết nối (nếu có chung k-2-mục đầu tiên) tức là:$(L_1[1]=L_2[1]) \cap (L_1[2]=L_2[2]) \cap ... \cap (L_1[k-2]=L_2[k-2]) \cap (L_1[k-1]=L_2[k-1])$.

Bước tỉa: C_k là tập chứa L_k (có thể là tập phổ biến hoặc không) nhưng tất cả tập k-mục phổ biến được chứa trong C_k. Bước này, duyệt lần hai CSDL để tính độ hỗ trợ cho mỗi ứng cử trong C_k sẽ nhận được L_k. Tuy nhiên để khác phục khó khăn, giải thuật Apriori sử dụng các tính chất: 1- Tất cả các tập con khác rỗng của một tập mục phổ biến là phổ biến; 2 - Nếu L là tập mục không phổ biến thì mọi tập chứa nó không phổ biến.

3.1.1 Mô phỏng thuật toán Apriori:

Như trên đã nói, các thuật toán khai phá Frequent Itemset phải thiết lập một số giai đoạn (pass) trên CSDL. Trong giai đoạn đầu tiên, người ta đếm support cho mỗi tập riêng lẻ và xác định xem tập nào là phổ biến (nghĩa là có support $\geq$ minsup). Trong mỗi giai đoạn tiếp theo, người ta bắt đầu với tập các tập phổ biến đã tìm được trong giai đoạn trước để lại sinh ra tập các tập mục có khả năng là phổ biến mới (gọi là tập các ứng cử viên - candidate itemset) và thực

hiện đếm support cho mỗi tập các ứng cử viên trong tập này bằng một phép duyệt trên CSDL. Tại điểm kết của mỗi giai đoạn, người ta xác định xem trong các tập ứng viên này, tập nào là phổ biến và lập thành tập các tập phổ biến cho giai đoạn tiếp theo. Tiến trình này sẽ được tiếp tục cho đến khi không tìm được một tập phổ biến nào mới hơn nữa.

Để tìm hiểu các thuật toán, ta giả sử rằng, các item trong mỗi giao dịch đã được sắp xếp theo thứ tự từ điển (người ta sử dụng khái niệm từ điển ở đây để diễn đạt một thứ tự quy ước nào đó trên các item của cơ sở dữ liệu). Mỗi bản ghi - record của cơ sở dữ liệu D có thể coi như là một cặp <TID, itemset> trong đó TID là định danh cho giao dịch. Các item trong một itemset cũng được lưu theo thứ tự từ điển, nghĩa là nếu kí hiệu k item cử một k-itemset c là c[1],c[2],…,c[k], thì c[1]<c[2]<…<c[k]. Nếu c=X.Y và Y là một m-itemset thì Y cũng được gọi là m-extension (mở rộng) của X. Trong lưu trữ, mỗi itemset có một trường support-count tương ứng, đây là trường chứa số đếm support cho itemset này.

Thuật toán Apriori

Các kí hiệu:

L_k: Tập các k-mục phổ biến (large k-itemset) (tức tập các itemset có support tối thiểu và có lực lượng bằng k).

Mỗi phần tử của tập này có 2 trường: itemset và suport-count.

C_k: Tập các candidate k-itemset (tập các tập k-mục ứng cử viên). Mỗi phần tử trong tập này cũng có 2 trường itemset và support-count.

Nội dung thuật toán Apriori được trình bày như sau:

Input: Tập các giao dịch D, ngưỡng support tối thiểu minsup

Output: L- tập mục phổ biến trong D

Method:

L_1={large 1-itemset} //tìm tất cả các tập mục phổ biến: nhận được L_1

for (k=2; $L_{k-1} \neq \varnothing$; k++) **do**

begin

C_k=apriori-gen(L_{k-1}); //sinh ra tập ứng cử viên từ L_{k-1}

for (mỗi một giao dịch $T \in D$) **do**

begin

C_T = subset(C_k, T); //lấy tập con của T là ứng cử viên trong C_k

for (mỗi một ứng cử viên c$\in$ C_T) **do**

c.count++; //tăng bộ đếm tần xuất 1 đơn vị

end;

$L_k = \{c \in C_k | \ c.count \geq minsup\}$

end;

return $\cup_k L_k$

Trong thuật toán này, giai đoạn đầu đơn giản chỉ là việc đếm support cho các item. Để xác định tập 1-mục phổ biến (L_1), người ta chỉ giữ lại các item mà support của nó lớn hơn hoặc bằng *minsup.*

Trong các giai đoạn thứ k sau đó (k>1), mỗi giai đoạn gồm có 2 pha. Trước hết các large(k-1)-itemset trong tập L_{k-1}được sử dụng để sinh ra các candidate itemset C_k, bằng cách thực hiện hàm Apriori_gen. Tiếp theo CSDL D sẽ được quét để tính support cho mỗi ứng viên trong C_k. Để việc đếm được nhanh, cần phải có một giải pháp hiệu quả để xác định các ứng viên trong C_k là có mặt trong một giao dịch T cho trước.

Vấn đề sinh tập candidate của Apriori – Hàm Apriori_gen:

Hàm Apriori_gen với đối số là L_{k-1}(tập các large(k-1)-itemset) sẽ cho lại kết quả là một superset, tập của tất cả các large k – itemset. Sơ đồ sau là thuật toán cho hàm này.

Input: tập mục phổ biến L_{k-1} có kích thước k-1

Output: tập ứng cử viên C_k

Method:

function apriori-gen(L_{k-1}: tập mục phổ biến có kích thước k-1)

Begin

For (mỗi $L_1 \in L_{k-1}$) **do**

For (mỗi $L_2 \in L_{k-1}$) **do**

begin

If (($L_1[1]=L_2[1]$) $\cap$ ($L_1[2]=L_2[2]$) $\cap$ ... $\cap$ ($L_1[k-2]=L_2[k-2]$) $\cap$ ($L_1[k-1]=L_2[k-1]$)) **then**

c = $L_1 \oplus L_2$; // kết nối L_1 với L_2 sinh ra ứng cử viên c

If has_infrequent_subset(c, L_{k-1}) **then**

remove (c) // bước tỉa (xoá ứng cử viên c)

else $C_k = C_k \cup \{c\}$; kết tập c vào C_k

end;

Return C_k;

End;

Hàm kiểm tra tập con k-1 mục của ứng cử viên k-mục không là tập phổ biến:

function has_infrequent_subset(c: ứng cử viên k-mục; L_{k-1} tập phổ biến k-1 mục)

Begin

//sử dụng tập mục phổ biến trước

For (mỗi tập con k-1 mục s của c) **do**

If s $\in L_{k-1}$ **then** return TRUE;

End;

Có thể mô tả hàm Apriori_gen trên theo lược đồ sau:

Input: tập các large(k-1)- itemset L_{k-1}

Output: tập candidate k-itemset C_k

Method:

Hàm Apriori-gen() //bước nối

1. insert into C_k

2. **select** $p.item_1$, $p.item_{2,}$..., $p.item_{k-1}$, $q.item_{k-1}$

3. **from** $L_{k-1}p$, $L_{k-1}q$

4. **where** $p.item_1=q.item_1$, …, $p.item_{k-2}=q.item_{k-2}$, $p.item_{k-1}<q.item_{k-1}$

//bước cắt tỉa:

5. **for** (mọi tập mục $c \in C_k$) **do**

6. **for** (mọi (k-1) tập con s của c(**do**

7. **if** ($s \notin L_{k-1}$) **then**

8. **delete** c khỏi C_k;

Với nội dung trên, ta thấy hàm này có 2 bước:

- Bước nối (join step): Bước này nối L_{k-1} với L_{k-1}. Trong bước này, cho rằng các item của các itemset đã được sắp xếp theo thứ tự từ điển. Nếu có k-2 item đầu tiên (gọi là phân tiền tố) của hai(k-1)-itemset $i_1$và i_2($i_1 \neq i_2$) nào đó mà giống nhau thì ta khởi tạo một candidate k-itemset cho C_k bằng cách lấy phần tiền tố này hợp với 2 item thứ k-1 của i_1 và i_2 (có thể phải sắp lại thứ tự cho các item này). Điều kiện $p.item_{k-1} <q.item_{k-1}$ đơn giản chỉ là việc tránh k-itemset trùng lặp được đưa vào C_k.

- Bước cắt tỉa (prune step): Đây là bước tiếp theo sau bước join. Trong bước này, ta cần loại bỏ tất cả các k-itemset $c \in C_k$ mà chúng tồn tại một(k-1)-subset không có mặt trong L_{k-1}. Giải thích điều này như sau: giả sử s là một(k-1)-subset của c mà không có mặt trong L_{k-1}. Khi đó, *support (s)<minsup*. Mặt

khác, theo tính chất p1.1, vì $c \supset s$ nên *support(s)<minsup*. Vậy c không thể là một large-itemset, nó cần phải loại bỏ khỏi C_k.

Ví dụ : Giả sử tập các item I = {A ,B, C, D, E} và cơ sở dữ liệu giao dịch:

D = {<1, {A,C,D}>, <2,{B,C,E}>, <3,{A,B,C,E}>,<4,{B,E}>}.

Với minsup = 0.5 (tức tương đương 2 giao dịch). Khi thực hiện thuật toán Apriori trên ta có sơ đồ sau:

D (CSDL)

TID	Các mục
1	{A, C, D}
2	{B, C, E}
3	{A, B, C, E}
4	{B, E}

Quét toàn bộ D

C_1

1 - itemset	Count-support
{A}	2 - 50%
{B}	3 – 75%
{C}	3 – 75%
{D}	1 - 25%
{E}	3 - 75%

Xóa bỏ mục có support < minsup

L_1

1 - itemset	Count-support
{A}	2 - 50%
{B}	3 – 75%
{C}	3 – 75%
{E}	3 - 75%

Kết nối

L_1 & L_1

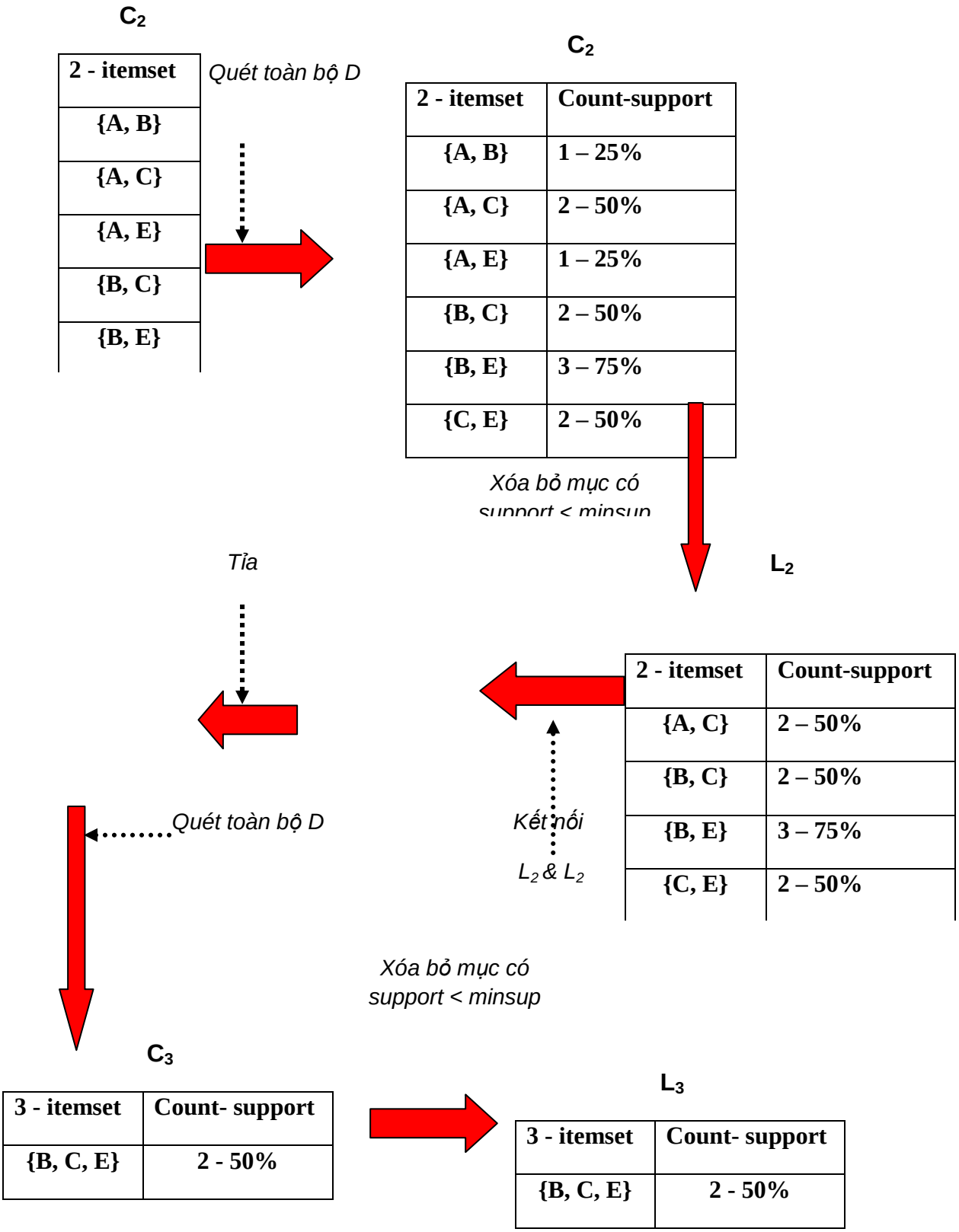
C_2

2 - itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

Tỉa

C_2

2 - itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}



Hình 5. Ví dụ thuật toán Apriori

3.1.2 Một số biến thể của giải thuật Apriori

Giải thuật **Apriori_TID** là phần mở rộng theo hướng tiếp cận cơ bản của giải thuật Apriori. Thay vì dựa vào cơ sở dữ liệu thô, giải thuật AprioriTID biểu diễn bên trong mỗi giao tác bởi các ứng viên hiện hành.

L_1= {Large 1-itemset};

C'_1 = Database D;

for (k=2; $L_{k-1} \neq \emptyset$; k++) **do**

Begin

C_k = apriori_gen(L_{k-1});

$C'_k = \emptyset$;

for tất cả $t \in C'_{k-1}$ **do**

begin

// xác định tập ứng viên trong C_k chứa trong giao dịch với định

//danh t. Tid (Transaction Code)

C_t = {$c \in C_k$ | (c-c[k]) $\in$ t.Set_of_ItemSets ^ (c-c[k-1]$\in$ t.Set_of_ItemSets}

for những ứng viên $c \in C_t$ **do** c.count ++;

if ($C_t \neq \emptyset$) **then** C'_k+= < t.Tid, C_t >

end

L_k = {$c \in C_k$ | c.count $\geq$ minsup};

End

return = $\cup_k L_k$;

Thuật toán này cũng sử dụng hàm apriori_gen để sinh ra các tập ứng cử viên cho mỗi giai đoạn. Nhưng thuật toán này không dùng CSDL D để đếm các support với các giai đoạn k > 1 mà sử dụng tập C'_k. Mỗi phần tử của C'_k có dạng <Tid, $\{X_k\}$>, trong đó mỗi X_k là một tập phổ biến k_itemset tiềm năng trong giao dịch Tid. Khi k = 1, C'_k tương ứng với D, trong đó mỗi item i được coi là một itemset {i}. Với k>1, C'_k được sinh ra bởi C'_k+= < t.Tid, C_t >. Phần tử của C'_k tương ứng với giao dịch t là <t.Tid, {c∈ | c chứa trong t}>. Nếu một giao dịch không chứa bất kỳ tập ứngviên k_itemset nào thì C'_k sẽ không có một điểm vào nào cho giao dịch này. Do đó, số lượng điểm vào trong C'_k có thể nhỏ hơn số giao dịch trong CSDL, đặc biệt với k lớn. Hơn nữa, với các giá trị k khá lớn, mỗi điểm vào có thể nhỏ hơn giao dịch tương ứng vì một số ứng viên đã được chứa trong giao dịch. Tuy nhiên, với các giá trị k nhỏ, mỗi điểm vào có thể lớn hơn giao dịch tương ứng vì một một điểm vào trong C'_k bao gồm tất cả các ứng viên k_itemset được chứa trong giao dịch.

Giải thuật **AprioriHybrid** kết hợp cả hai hướng tiếp cận trên. Ngoài ra còn có một số các giải thuật tựa Apriori(TID), chúng được định hướng để cài trực tiếp trong SQL.

Giải thuật **DIC** là một biến thể khác nữa của giải thuật Apriori. Giải thuật DIC làm giảm đi khoảng phân biệt nghiêm ngặt giữa việc đếm và việc phát sinh các ứng viên. Bất kỳ ứng viên nào tới được ngưỡng minsupp, thì giải thuật DIC bắt đầu phát sinh thêm các ứng viên dựa vào nó. Để thực hiện điều này giải thuật DIC dùng một prefix-tree (cây tiền tố). Ngược với hashtree, mỗi nút (nút lá hoặc nút trong) của prefix-tree được gán một ứng viên xác định trong tập phổ biến. Cách sử dụng cũng ngược với hashtree, bất cứ khi nào tới được một nút ta có thể khẳng định rằng tập *item* đã kết hợp với nút này trong giao tác đó. Hơn nữa, việc xác định độ hỗ trợ và phát sinh ứng viên khớp nhau sẽ làm giảm đi số lần duyệt cơ sở dữ liệu.

3.1.3 Cải tiến thuật toán Apriori:

Như đã trình bày ở trên, quá trình tìm luật kết hợp gồm hai giai đoạn:

1) Tìm các tập phổ biến với ngưỡng minsupp $\in (0, 1]$ cho trước;

2) Với các tập phổ biến tìm được trong bước 1 và với ngưỡng độ tin cậy minconf $\in (0, 1]$ cho trước, liệt kê tất cả các luật kết hợp thỏa mãn ngưỡng minconf.

Công việc chiếm hầu hết thời gian của bước 1 là xác định một tập dữ liệu có phải là tập phổ biến hay không. Trong thực tế, ta không cần thiết phải khai phá tất cả các tập mục phổ biến trong bước thứ nhất mà chỉ cần khai phá tập các

mục phổ biến đóng. Phần này trình bày về việc sử dụng ánh xạ đóng để tìm các tập phổ biến đóng. Do tập phổ biến đóng nhỏ hơn rất nhiều so với tập tất cả các tập phổ biến nên thời gian của thuật toán tìm tập phổ biến sẽ giảm đi đáng kể.

Định nghĩa 1: Kết nối Galois

Cho quan hệ nhị phân R$\subseteq$ I x X. Cho R$\subseteq$ I & R$\subseteq$ T thì các ánh xạ:

t: I $\rightarrow$ T, t(X) = {y$\in$T/$\forall$x$\in$X, x R s}, X $\subseteq$ I.

- t(X) là tập hợp tất cả các giao tác của T chứa tất cả các thuộc tính của X.

i:T$\rightarrow$I, i(S) = {x$\in$I/$\forall$s$\in$S, x R s}, S $\subseteq$ T.

- i(S) là tập hợp tất cả các thuộc tính của I xuất hiện ở tất cả các giao tác trong S

Ví dụ: Cho CSDL D

	A	B	C	D	E
1	1	1	0	1	1
2	0	1	1	0	1
3	1	1	0	1	1
4	1	1	1	0	1
5	1	1	1	1	1
6	0	1	1	1	0

Ta có: t(AB) = 1345; t(BCD) = 56; t(E) = 12345

i(123) = BE; i(345) = ABE; i(23) = BE

Cặp ánh xạ (t, i) được gọi là kết nối Galois trên T x I.

Định nghĩa 2: Ánh xạ hợp

Cho X $\subseteq$ I và S $\subseteq$ T, ta định nghĩa hai ánh xạ hợp:

Cit: I -> I $C_{it}(X) = i(t(X))$

Cti: T -> T $C_{ti}(S) = t(i(S))$

Ví dụ:

$C_{it}(AB) = i(t(AB)) = i(1345) = ABE$

C_{ti} (23) = t(i(23)) = t(BE) = 12345

Định nghĩa 3: Ánh xạ đóng

Cho tập U, subset(U) = {X | X ⊆ U}. Ánh xạ f: subset(U) -> subset(U) được gọi là đóng trên U nếu mọi tập con X, Y ⊆ U ta có các tính chất sau:

T1) Tính phản xạ: X ⊆ f(X)

T2) Tính đồng biến: Nếu X ⊆ Y thì f(X) ⊆ f(Y)

T3) Tính lũy đẳng: f(f(X)) = f(X)

Nhận thấy C_{it} và C_{ti} là hai ánh xạ đóng trên các tập mục và các tập giao dịch tương ứng.

Định nghĩa 4: Bao đóng của tập mục dữ liệu

Cho X ⊆ I, bao đóng của X là $X^+ = C_{it}(X)$

Ví dụ: Xét CSDL D ở trên

A^+ = ABE vì $C_{it}(A)$ = i(t(A)) = i(1345) = ABE

B^+ = B vì $C_{it}(B)$ = i(t(B)) = i(123456) = B

AC^+ = ACE vì $C_{it}(ACE)$ = i(t(AC)) = i(45) = ACE

Định nghĩa 5: Tập phổ biến đóng

X ⊆ I là tập phổ biến theo ngưỡng minsupp. Ta nói X là tập phổ biến đóng theo ngưỡng minsupp nếu $X = X^+ = C_{it}(X)$.

Ví dụ, xét CSDL trên, ta có: B, BC là tập phổ biến đóng theo ngưỡng minsupp = 0,4 vì $C_{it}(B)$ = B $C_{it}(BC)$ = BC và supp(B)=1, supp(BC)=0,66.

BCD không là tập phổ biến đóng theo ngưỡng minsupp = 0,4 vì $C_{it}(BCD)$=BCD nhưng supp(BCD)=0,33 < minsupp.

Định nghĩa 6: Bao đóng của một tập mục

Cho K ⊆ supset(I) thỏa minsupp, ta định nghĩa $K^+ = \{X^+ \mid X \in K\}$ là bao đóng của họ K.

Thuật toán 1: Tìm bao đóng của tập I

Format: Fred_1_Item(T, I, minsupp)

Input: CSDL D, minsupp, tập các mục I

Output: $K^+ = \{X^+ \mid X \in K, X^+ = C_{it}(X)$ và supp(X) ≥ minsupp}

Method:

K=∅;

For mỗi X ∈ I do

If (supp(X) ≥ minsupp) then

K:= K ∪ {C_{it}(X)}

Endif;

Endfor;

Return (K);

Thuật toán 2: Tìm tập đóng, tìm Fix(C_{it})

Format: Fix(T, I, minsupp)

Input: CSDL D, minsupp, tập các mục I

K = Fred_1_Item(T, I, minsupp)

Output: K^+ {X ⊆ K | X = X^+ và supp(X) ≥ minsupp}

Method:

K^+:=∅;

While (K≠K^+) do

K' = K^+;

K_1:={X ∪ Y | X, Y ∈K};

K_2 := ∅;

For mỗi X ∈ K_1 do

K_2:=K_2 ∪ {X^+}

Endfor

Frequent(K_1, K_2, minsupp, K^+);

K:=K';

Endwhile;

Return(K^+);

Thuật toán 3: Tìm các tập thường xuyên của K

Format: Frequent(K_1, K, minsupp, K^+)

Input: K ⊆ I, minsupp;

K_1 ={X ∈K đã tính ở bước trên và supp(X) ≥ minsupp}

Output: K^+ = {X ∈K | supp(X) ≥ minsupp}

Method:

K_2 := ∅;

For mỗi X ∈ K do

If (∃ Y ∈ K_1) and (X ⊆ Y) then

K_1 := K_1 ∪ {X}

Else

If not((∃ Y ∈ K_2) and (X ⊆ Y)) then

If supp(X) ≥ minsupp then

K_1 := K_1 ∪ {X}

Else

K_2 := K_2 ∪ {X};

Endif;

Endif;

Endfor;

Return(K_1);

Ví dụ: Xét CSDL D ở trên, với I = {A, B, C, D, E}=ABCDE; T={1, 2, 3,4, 5,6}=123456; minsupp = 0,4 (tương đương với ≥ 3 giao dịch)

Áp dụng thuật toán 1 ta được K = {ABE, B, BC, BD, BE}

Áp dụng thuật toán 2 với Input: K = {ABE, B, BC, BD, BE}

Ta được Output: K2 = {ABCE, ABDE, BCD, BCE, BDE}

Áp dụng thuật toán 3 với Input: K_1 = {ABE, B, BC, BD, BE}

Ta được Output: {ABE, B, BC, BD, BE,ABDE, BCE, BDE}

Nhận xét: Trên đây trình bày một cải tiến của việc tìm tập phổ biến bằng cách sử dụng các kết quả lý thuyết về ánh xạ đóng, bao đóng, …Thuật toán đưa ra tránh phải tìm toàn bộ các tập phổ biến, thay vào đó chi phải tìm một só lượng nhỏ hơn các tập phổ biến đóng, điều này cải tiến đáng kể tốc độ tính toán trong trường hợp dữ liệu có dung lượng lớn.

3.2 Giải thuật DFS (Depth First Search)

Giả sử việc đếm các thể hiện được thực hiện trên tập các ứng viên có kích thước hợp lý, với mỗi tập các ứng viên đó thì cần một thao tác duyệt cơ sở dữ liệu. Chẳng hạn như, giải thuật Apriori dựa vào BFS thực hiện duyệt cơ sở dữ liệu mỗi *k-kích thước* ứng viên một lần. Khi thực hiện tìm kiếm ưu tiên theo chiều sâu (DFS) tập ứng viên chỉ gồm chỉ gồm một nút của cây từ phần 2.2. Một điều hiển nhiên là nếu phải thực hiện duyệt cơ sở dữ liệu cho mỗi nút thì tổng chi phí kết quả thật khổng lồ. Vì thế việc kết hợp DFS với việc đếm các thể hiện là không thật sự thích hợp.

Gần đây có một cách tiếp cận mới được gọi là **FP-growth** đã được trình bày. Trong bước tiền xử lý giải thuật FP-growth dẫn xuất cách biểu diễn rất dày đặc của dữ liệu giao tác, do đó cần một FP-tree. Việc phát sinh ứng viên của FP-tree được thực hiện thông qua việc đếm các thể hiện và DFS. Ngược với hướng tiếp cận của DFS, FP-growth không theo nút của cây từ phần trên, mà đi trực tiếp xuống một số phần của tập item trong không gian tìm kiếm. Trong bước thứ hai, FP-growth dùng FP-tree để dẫn xuất tất cả các giá trị hỗ trợ của tất cả các tập phổ biến.

3.3 Giải thuật DHP (Direct Hashing and Pruning)

Thuật giải Direct Hashing and Pruning thực chất là một biến thể của thuật toán Apriori. Các hai thuật toán đều phát sinh ra các ứng viên k+1 phần tử từ một tập k-phần tử (với số lượng lớn). Và cũng với số lượng lớn các tập k+1 phần tử này đọc xác nhận bằng cách đếm sự xuất hiện của các ứng viên k+1 phần tử này trên database (thực chất là tính lại 2 độ support). Sự khác biệt của thuật toán DHP ở đây là chúng ta sẽ sử dụng kỹ thuật hashing (băm) để loại bỏ ngay các tập không cần thiết cho pha phát sinh các ứng viến kế tiếp.

Nhận xét rằng, tập các ứng viên được phát sinh ban đầu, đặc biệt là tập 2-phần tử là vấn đề mấu chốt để đánh giá mức độ hiệu quả của data mining vì trong mỗi bước, các tập k-phần tử (L_k) được dùng để tạo các ứng cử viên (k+1)-phần tử (C_{k+1}) bằng cách ghép L_k với chính một phần tử L_k khác trong bước kế. Nói chung, càng nhiều tập *c* trong C_k thì chi phí xử lý cho việc xác định L_{k+1} càng tăng. Trong giải thuật Apriori, $|C_2| = \binom{|L_1|}{2}$ vì vậy số bước để xác định L_2 từ C_2 bằng cách quét qua toàn bộ cơ sở dữ liệu và kiểm tra trên từng transaction lên tập C_2 là quá tốn kém. Bằng cách xây dựng một C_2 đã được giảm thiểu đáng kể, thuật giải DHP thực hiện việc đếm trên tập C_2 nhanh hơn nhiều so với Apriori.

Trong quá trinh đếm độ support của C_k trong thuật giải DHP bằng cách quét qua cơ sở dữ liệu, thuật giải cũng tích lũy những thông tin cần thiết để hỗ trợ việc tính toán trên các ứng viên (k+1)-phần tử theo ý tưởng là tất cả các tập con (k+1)-phần tử của mỗi transaction sau vài thao tác cắt xén được băm vào trong bảng băm. Mỗi mục trong bảng băm chứa một số các tập đã được băm vào theo hàm băm. Sau đó, bảng băm này đươc dùng để xác định C_{k+1}. Để tìm ra C_{k+1}, thuật giải phát sinh ra tất cả các tập (k+1)-phần tử từ L_k như trong trường hợp của Apriori. Ở đây, thuật giải chi đưa một tập (k+1)-phần tử vào C_{k+1} chỉ khi tập (k+1)-phần tử này qua được bước lọc dựa trên bảng băm. Như vậy thuật giải đã giảm được việc phát sinh các phần tử dư thừa trong C_k để giảm chi phí kiểm tra khi phát sinh tập L_k. Qua kiểm nghiệm cho thấy, thuật giải DHP đã giảm đáng kể kích thước của C_{k+1}.

Input: Database D

Output: Tập phổ biến k-item

```
/* Database = set of transaction;
   Items = set of items;
   transaction = <TID, {x∈ Items}>;
   F1 là tập phổ biến l-item */
   F1=∅;
/*
   H2 là bảng băm có 2-item
*/
for each transaction t∈Database do begin
  for each item x in t do
    x.count++;
  for each 2-itemset y in t do
    H2.add(y);
end
 for each item i∈Item do
  if i.count/|Database|≥minsupp
  then F1=F1∪i;
end
 H2.prune(minsupp)
/* Tìm Fk tập phổ biến k-item, k≥2 */
for each (k:=2; Fk-1≠∅; k++) do begin
```

```
 // Ck: tập các ứng viên k-item
 Ck=∅
for each x∈{F_{k-1}*F_{k-1}} do
 if H_k.hassupport(x)
 then C_k = C_k∪x;
end
 for each transaction t∈Database do begin
 for each k-itemser x in t do
   if x∈C_k
   then x.count++;
 for each (k+1)-itemset y in t do
    if ¬∃z | z = k –subset of y
          ∧¬H_k.hassupport(z)
    then H_{k+1}.add(y);
end
// F_k là tập phổ biến k-item
 F_k=∅;
for each x∈C_k do
 if x.count/|Database|≥minsupp
 then F_k=F_k∪x;
end
 Hk+1.prune(minsupp)
 end
 Answer=∪_k F_k
```

Trong bước khởi tạo, trong khi đếm số lần xuất hiện của các tập 1-phần tử, sự xuất hiện của các giá trị băm cho tập 2-phần tử cũng được đếm. Khi đó tập các ứng cử viên được loại khỏi bảng băm nếu giá trị băm tương ứng trong bảng băm nhỏ hơn minSupp. Một tập (k+1)-phần tử trong một transaction được thêm vào bảng băm H_{k+1} nếu giá trị băm của tất cả tập con k-phần tử của ứng viên (k+1)-phần tử thỏa minSupp trong H_k. Giải thuật DHP cũng xét đến việc loại bỏ các transaction không chứa bất kỳ một tập phổ biến nào khỏi cơ sở dữ liệu cũng như loại bỏ các item không tham gia tập phổ biến sau mỗi bước.

Trong trường hợp kích thước của cơ sở dữ liệu tăng thì thuật giải DHP cải thiện đáng kể tốc độ so với giải thuật Apriori. Tuy nhiên, mức độ này còn phụ thuộc nhiều vào kích thước bảng băm

3.4 Giải thuật PHP(Perfect Hashing and Pruning)

Trong thuật giải DHP, nếu chúng ta có thể định nghĩa một bảng băm lớn sao cho mỗi tập item có thể được ánh xạ vào các ô riêng biệt trong bảng băm thì giá trị băm của bảng băm sẽ cho biết số lượng xuất hiện thật sự của mỗi tập phần tử. Trong trường hợp này, chúng ta sẽ không cần phải thực hiện lại việc đếm số lần xuất hiện cho các tập item này.

Dễ thấy rằng số lượng dòng dữ liệu cần quét với một tập gồm nhiều tập item cũng là một vẫn đề ảnh hưởng xấu đến hiệu quả thực hiện. Việc giảm thiểu số transaction cần phải đọc lại và bỏ bớt các item không cần xét rõ ràng cải thiện chất lượng data mining với cơ sở dữ liệu lớn.

Giải thuật được đề nghị PHP sử dụng một bảng băm lý tưởng (perfect hashing) cho mỗi bước phát sinh bảng băng và đồng thời giảm kích thước cơ sở dữ liệu bằng cách cắt bỏ những transaction không chứa bất kỳ một tập phổ biến nào. Thuật giải được mô tả như sau: Trong bước đầu tiên của thuật giải, kích thước của bảng băm bằng với số lượng item trong cơ sở dữ liệu. Mỗi item này được ánh xạ vào một vị trí riêng biệt trong bảng băm, do đó, ta gọi giải thuật này là perfect hashing. Phương thức cộng của bảng băm thêm vào một mục mới nếu mục này chưa tồn tại trong bảng băm và giá trị đếm được khởi tạo là 1; ngược lại biến đếm sẽ được tăng lên 1 đơn vị. Sau bước đầu tiên, bảng băm chứa đúng số lần xuất hiện của mỗi item trong cơ sở dữ liệu. Chỉ cần duyệt một bước qua bảng băm (được đặt trong bộ nhớ chính), thuật giải dễ dàng phát sinh ra các tập phổ biến 1-phần tử. Sau bước này, phương thức prune của bảng băm sẽ loại bỏ tất cả các mục có độ support nhỏ hơn minSupp.

Trong các bước tiếp theo, giải thuật cắt xén bớt cơ sở dữ liệu bằng cách bỏ đi không xét đến các transaction không chứa bất kỳ một tập phổ biến nào cũng như bỏ tất cả các item không tham gia vào một tập phổ biến nào. Kế đó, thuật giải phát sinh các ứng viên k-phần tử và đếm số lần xuất hiện của các tập k-phần tử. Cuối của bước này, D_k là cở sở dữ liệu đã được cắt xén, H_k chứa số lần xuất hiện của các tập k-phần tử, F_k là các tập phổ biến k-phần tử. Quá trình này tiếp tục cho đến khi không còn F_k nào được tìm thêm nữa.

Thuật giải này rõ ràng là tốt hơn DHP vì sau khi tạo bảng băm, chúng ta không cần đếm số lần xuất hiện của các ứng viên k-phần tử như trong trường hợp DHP. Giải thuật này cũng tốt hơn giải thuật Apriori vì tại mỗi vòng lặp, kích thước của cơ sở dữ liệu được giảm, điều này làm tăng hiệu quả của thuật toán trong trường hợp cơ sở dữ liệu lớn và số lượng các tập phổ biến tương đối nhỏ.

Input: Database

Output: Tập phổ biến k-item

```
/* Database = set of transaction;
   Items = set of items;
   transaction = <TID, {x ∈ Items}>;
   F1 là tập phổ biến l-item */
   F1=∅;
/*
   H1 là bảng băm có 1-itemset
*/
for each transaction t ∈ Database do begin
  for each item x in t do
   H1.add(x);
end
 for each itemset y in H1 do
  if H1.hassupport(y)
  then F1=F1 ∪ y
end
 H1.prune(minsupp)
 D1=Database
 /* Tìm Fk tập phổ biến k-item, k ≥ 2 */
k=2;
repeat
   Dk=∅;
    Fk=∅;
   for each transaction t ∈ Dk-1 do begin
   // w là k-1 subset của item in t
   if ∀w | w ∉ Fk-1
   then skip t;
   else
```

items = ∅;
for each k-itemser y in *t* do
if $\neg\exists z \mid$ z = k –subset of y
$\wedge \neg H_{k-1}$.hassupport(z)
then H_k.add(y);
items = item$\cup$ y;
end
$D_k=D_k \cup t$;
end
for each itemset y in H_k do
if Hk.hassport(y)
then $F_k=F_k \cup y$;
end
H_{k+1}.prune(minsupp)
k++;
until $F_{k-1}=\emptyset$;
Answer=$\cup_k F_k$;

Ngoài ra, sau mỗi vòng lặp thì D_k là cở sở dữ liệu chỉ chứa các transaction có chứa tập phổ biến. Giải thuật tạo tất cả các tập con k-phần tử của mỗi item trong mỗi giao tác và chèn phần tử nào có các tập con k-1 phần tử thỏa độ support trong bảng băm. Vì thuật giải thực hiện việc cắt xén trong quá trình thêm các ứng viên k-phần tử vào H_k nên kích thước của bảng băm không quá lớn và có thể đặt trong bộ nhớ chính.

4. Phát sinh luật từ các tập phổ biến

Sau khi có được các tập phổ biến với độ tin cậy minSupp, chúng ta cần rút ra các luật có độ tin cậy minConf. Để sinh các luật, với mỗi tập phổ biến L, ta tìm các tập con khác rỗng của L. Với mỗi tập con s tìm được, ta xuất ra luật s $\Rightarrow$ (L-s) nếu tỉ số supp(L)/supp(a) tối thiểu là minsconf

for mỗi tập phổ biến *L*

tạo tất cả các tập con khác rỗng *s* of *L*

for mỗi tập con khác rỗng *s* of *L*

cho ra luật "*s* $\Rightarrow$ (*L-s)"* nếu *support(L)/support(s)* $\geq$ *min_conf"*

trong đó *min_conf* là ngưỡng độ tin cậy tối thiểu

Ví dụ: tập phổ biến l = {abc}, subsets s = {a, b, c, ab, ac, bc)

$a \Rightarrow b$, $a \Rightarrow c$, $b \Rightarrow c$

$a \Rightarrow bc$, $b \Rightarrow ac$, $c \Rightarrow ab$

$ab \Rightarrow c$, $ac \Rightarrow b$, $bc \Rightarrow a$

Vấn đề ở đây là nếu lực lượng item trong |L| = *n* trở nên lớn, số luật có thể phát sinh từ một tập phổ biến L sẽ không nhỏ chút nào

Số luật phát sinh từ $L = 2^n - 2$ với |L| = *n* (nghĩa là nếu |*L*| = 10, ta cần phải kiểm tra độ tin cậy của 1022 luật được phát sinh).

4.1 Cải tiến 1 – Giảm số lượng các luật được phát sinh và cần phải kiểm tra

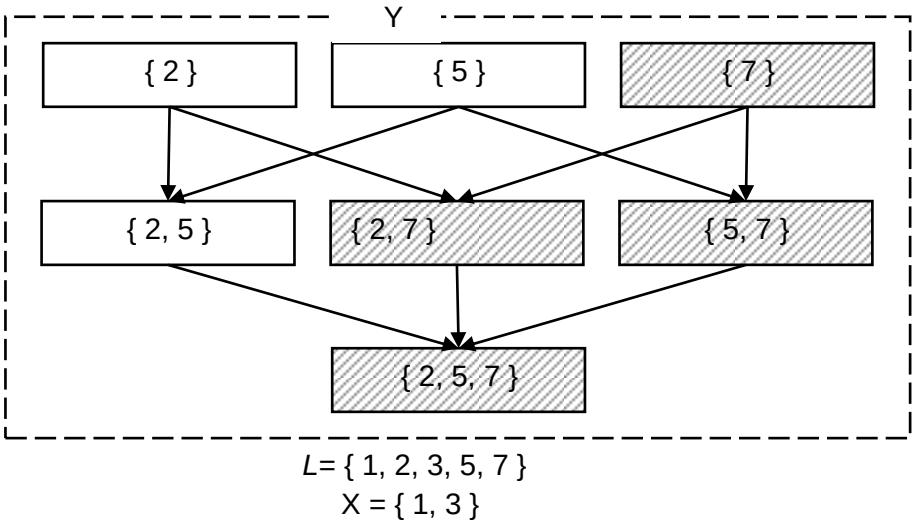
Khó khăn đầu tiên mà chúng ta phải giải quyết trong bài toán là khi |L| chỉ hơi tăng thì số luật phát sinh đã tăng theo cấp số mũ dẫn đến phải kiểm tra nhiều luật hơn.

Xét một luật r: X => Y có không thỏa minConf thì chắc chắn luật r' được phát sinh bằng cách thêm vào vế trái một item $i \in L$ cũng không thể thỏa minConf:

Nếu r: X => Y có conf(r) < minConf thì r': X => Y $\cup$ *i* (với $i \in L$) cũng có conf(r')<minConf.

Như vậy, nếu như ta chỉ xét trên một tập X thì việc phát sinh và kiểm tra các luật r nên bắt đầu với tập Y là tập gồm 1 phần tử, rồi đến các tập 2 phần tử, 3 phần tử… Nếu chúng ta nhìn lại bài toán tìm tập phổ biến thì ta sẽ thấy việc tìm tập Y cũng có tính chất gần tương tự với bài toán đi tìm tập phổ biến L. Chúng ta chỉ phát sinh và kiểm tra độ tin cậy của một phần tử *y* ở mức k nếu mọi tập con của nó đều thỏa minConf (nói một cách khác là mọi tập con của nó phải thuộc Y_k).

```
for each X ⊂ L do
  if X ≠ ∅ then begin
        YS1 = generate_1_itemset_has_confident(X, L\X);
        k = 2
        while YSk-1 ≠ ∅ do
              CYk = generate_k_itemset_from(YSk-1, L\X);
              YSk = DB.check_confident(X, CYk);
        Endwhile
    end
  endfor;
```



Hình 6: ví dụ minh họa

Trong ví dụ trên, luật {1, 3} => {7} không thỏa minConf dẫn đến các luật {1, 3} => {2, 7}, {1, 3} => {5, 7}, {1, 3} => {2, 5, 7} cũng không cần xét nữa.

Với nhận xét này, chúng ta có thể áp dụng được một số cải tiến trong những cải tiến được sử dụng cho bài toán tìm tập phổ biến nhưng ở đây cần lưu ý một điều là ở đây lực lượng |L| không quá lớn và việc tính supp(X∪Y) và supp(Y) có thể xem như đã được lưu lại (xem lại thuật giải PHP) nên có thể một số cải tiến trở nên không cần thiết.

4.2 Cải tiến 1.a – tránh phát sinh các luật không có ý nghĩa

Một tính chất khác mà chúng ta cũng cần lưu lý là nếu chúng ta có một luật r: X => Y thỏa conf(r) ≥ minConf thì luật được phát sinh bằng cách thêm vào vế trái một một item $i \in Y$ cũng thỏa độ tin cậy minConf:

Nếu r: X => Y,conf(r)≥minConf thì r': X∪i => Y cũng có conf(r') ≥minConf

Ở đây, luật r' không đem lại ý nghĩa thực tế nếu ta đã có luật r nên trong phần lớn các ứng dụng tìm luật kết hợp, ta đều có mong muốn bỏ không xét đến nó.

Như vậy thay vì xét tuần từ các X ⊂ L, ta sẽ xét có thứ tự đầu tiên là tập các X có 1 phần tử, rồi đến tập các X có 2 phần tử, …, tập X có |L|-1 phần tử. Việc xét có thứ tự này sẽ giúp cho ta phát hiện sớm và loại bỏ hoàn toàn những luật được phát sinh r: X => Y không có ý nghĩa bằng cách đánh dấu những luật r này như là luật không thỏa minConf nếu như chúng ta phát hiện đã có một luật X' => Y thỏa minConf, với X'⊂X.

Thuật giải được sửa lại như sau:

for k=1 to |L|-1 do

for each X ∈ generate_k_itemset(*f*) do

YS_1 = generate_1_itemset_has_confident(X, L\X);

$YS_{1.}$ = Cache.FilterOutRedundantRules(X, YS_1);

k = 2;

while $YS_{k-1} \neq \varnothing$ do

CY_k = generate_k_itemset_from(YS_{k-1}, L\X);

YS_k = DB.check_confident(X, CY_k);

endwhile
endfor; endfor;

4.3 Một số kỹ thuật khác trong việc tối ưu hóa chi phí tính độ Confident

Để tránh việc phải quét lại cơ sở dữ liệu để tính độ tin cậy (tốn kém chi phí không kém gì việc quét cơ sở dữ liệu để tính độ support), ta có thể áp dụng một hướng tiếp cận nào đó để cache (lưu lại) độ support của các tập phổ biến. Chi phí lưu trữ này rõ ràng là quá nhỏ so với chi phí phải bỏ ra để tính lại độ confident cho luật. Ta cũng có thể tận dụng hash tree được sử dụng trong thuật toán PHP để có thể nhanh chóng tính được độ support của một tập phổ biến bất kỳ.

5. Đánh giá, nhận xét

Phần này chúng ta đã xem xét các giải thuật khai phá tập phổ biến như: Apriori, AprioriTID, ... Các giải thuật này đều tỷ lệ tuyến tính với kích thước CSDL. Nghĩa là tất cả các độ phức tạp về thời gian, bộ nhớ, tính toán thuật toán, . . . đều tỉ lệ thuận với độ lớn CSDL D.

Chương 2: MÔ HÌNH TÌM KIẾM THÔNG TIN

1. Tìm kiếm thông tin

Hãy tưởng tượng việc tìm kiếm một cuốn sách trong thư viện mà không có bảng liệt kê mục lục. Thật không phải là một công việc dễ dàng. Cũng như việc tìm kiếm một thông tin trên Internet. Để bắt đầu người dùng theo các siêu liên kết đến trang web mới rồi xác định các tài liệu liên quan chứa thông tin mình cần. Mỗi liên kết không rõ ràng có thể đưa họ đi xa hơn phạm vi tìm kiếm. Trong một hệ thống nhỏ và cố định việc thiết kế một tài liệu hướng dẫn việc tìm kiếm không thành vấn đề. Nhưng trong môi trường world Wide Web là một môi trường thông tin không tập trung, gồm nhiều loại khác nhau, liên tục thay đổi và phát triển nhanh chống thì việc tìm kiếm thông tin có thể nói là một thách thức đòi hỏi khá nhiều thời gian.

Hiện nay đã có khá nhiều các công cụ hay những bộ máy tìm kiếm thông tin thông minh cho phép giải quyết vấn đề này. Nó cung cấp một cơ chế tìm kiếm nhanh chóng bằng cách duy trì một hệ thống chỉ mục các trang web. Côn việc của bộ chỉ mục là phân loại các trang web thình các nhóm thông tin và đánh chỉ mục full-text cho tất cả các trang web. Do môi trường web liên tục thay đổi nên việc đánh chỉ mục phải được thực theo định kì. Người dùng chỉ việc nhập vào các từ khóa hay chủ đề mình cần, bộ máy tìm kiếm sẽ liệt kê tất cả các tài liệu liên quan theo thứ tự độ chính xác tìm được.

Hiện nay có rất nhiều loại môtơ tìm kiếm. Cơ thế tìm kiếm của nó có thể là tìm kiếm theo một chủ đề hay một loại thông tin nào đó. Ví dụ: tìm kiếm thông tin về phần mềm (www.softseek.com), âm nhạc (www.mp3search.com), …. Hay cũng có thể là các thông tin tổng hợp.

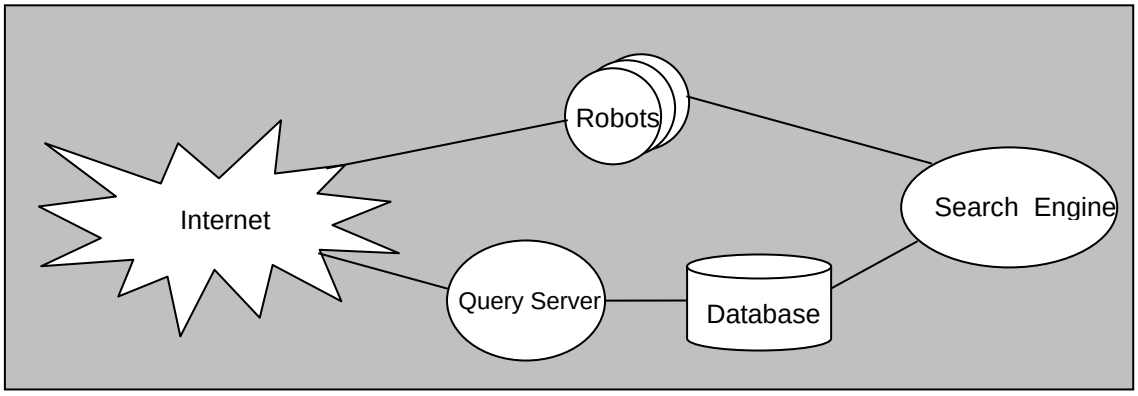
Cùng với nhu cầu tìm kiếm thông tin là nhu cầu nắm bắt những thay đổi trên web. những thay đổi bao gồm việc cập nhật những thông tin về các nhu cầu việc làm mới trên internet, hay những tin tức nóng bỏng … Nó giúp cho các ưng viên tìm được những việc làm phù hợp hay các doanh nghiệp có thể tìm những ứng viên phù hợp với yêu cầu doanh nghiệp, nó cũng giúp cho người dùng biết được những gì đã và đang diễn ra xung quanh.

Như đã nói ở trên việc duy trì hệ thống chỉ mục (bao gồm cả chỉ mục về loại thông tin của tài liệu lẫn chỉ mục full-text các tài liệu) cho các trang web quyết định chất lượng của các search engine. Để duy trì hệ thống chỉ mục này chúng liên tục duyệt qua các trang web bằng cách đi theo các siêu liên kết, qua đó

quyết định xem những tài liệu nào sẽ được thêm vào bảng chỉ mục của mình. Đặc điêm quan trọng nhất của world wide web là mô hình thông tin không tập trung. Bất cứ ai cũng có thể thêm vào các server, các thông tin hay các siêu liên kết. trong môi trường thay đổi như vậy, đối với một search engine cùng với việc thu thập các thông tin liên quan, việc phát hiện các thông tin mới cũng là rất quan trọng.

Các search engine nhận biết các thông tin cần thiết của người dùng thông qua địa chỉ url của chúng. Khi xét một Url, search engine sẽ dựa vào mục đích tìm kiếm quyết định xem nó có nên được dùng để tìm kiếm tiếp hay không và sẽ lưu nội dung của nó lại nếu thích hợp, sau khi lưu một tài liệu, search engine tìm kiếm và đánh dấu tài liệu đã được xét rồi. và tìm tất cả các liên kết có trong tài liệu và lại tiếp tục như vậy đối với các liên kết mới này. Tất cả các bước này đều ảnh hưởng đến việc lưu thông tin trong cơ sở dữ liệu.

2. Mô hình Search engine



Một Search engine bao gồm các thành phần

- Modul chính **Search engine**: điều khiển tất cả hoạt động của hệ thống
- Modul cập nhật thông tin Robots: chịu trách nhiệm tìm kiếm và tái hiện thông tin về các tài liệu trên internet phù hợp với yêu cầu do modul chính đưa ra.
- Phần cơ sở dữ liệu: lưu trữ các thông tin về các tài liệu như: nội dung tài liệu, các siêu liên kết giữa chúng, …

2.1 Search engine

Một search engine phát hiện các tài liệu mới bằng cách bắt đầu với một tập hợp các tài liệu đã biết, kiểm tra các siêu liên kết xuất hiện trong đó, duyệt theo một trong các liên kết đến tài liệu mới, sau đó lặp lại toàn bộ quá trình này. Tưởng tượng web như là một đồ thị có hướng và việc tìm kiếm đơn giản chỉ là duyệt qua đồ thị sử dụng với một thuật toán duyệt đồ thị nào đó. Search engine

không chỉ chịu trách nhiệm quyết định xem tài liệu nào sẽ duyệt mà còn quyết định xem kiểu tài liệu nào mới được duyệt.

2.2 Agents

Để thực hiện việc thu thập tài liệu từ web, search engine gọi đến các "Agent" hay còn gọi là các Robot. Đầu vào của nó là một địa chỉ Url và nhiệm vụ là tái hiện thông tin về tài liệu tại địa chỉ đó. Kết quả trả về cho modul chính là một đối tượng chứ nội dung tài liệu ở địa chỉ đó hoặc một giải thích lý do tại sao tài liệu không được tái hiện. Các Agent này phải có khả năng truy cập được các kiểu nội dung khác nhau với các giao thức phổ biến như HTTP, FTP, …

Việc chờ đợi sự trả lời từ một server ở xa có thể gây tốn tài nguyên của hệ thống, các Agent thường được tổ chức thành các tiến trình khác nhau và chạy song song với nhau. Modul chính làm chức năng quản lý tiến trình này, khi phát hiện ra một địa chỉ mới, nó sẽ tìm một Agent đang rỗi và giao nhiệm vụ cho Agent này. Khi thực hiện xong nó trả lại kết quả cho modul chính và thiết đặt trạng thái rỗi. Quá trình cứ tiếp tục như thế cho đến hết thời gian quy định hay khi không còn có một địa chỉ mới nào nữa.

3. Hoạt động của các Search engine

Như đã nói ở trên Search Engine dùng các robot để xây dựng bảng chỉ mục nội dung các trang Web. Đó là các chương trình tự động đi theo các siêu liên kết trên các trang Web, thu thập các dữ liệu tại các trang Web đó cần thiết cho việc đánh chỉ mục. Chúng được gọi là các robot bởi vì chúng hoạt động độc lập: chúng tự phân tách các siêu liên kết và đi theo các siêu liên kết này. Một số tên khác cho những chương trình kiểu này: spider, spider, worm, wanderer, gatherer, ... Việc các rôbốt đi theo các liên kết cũng giống như một người duyệt Web xem các trang tài liệu trên browser của mình.

Bạn có thể hỏi tại sao các robot lại phải tạo ra bảng chỉ mục các trang Web như vậy, tại sao không chỉ tìm kiếm khi người dùng đã nhập vào yêu cầu tìm kiếm. Đó là vì việc tổ chức bảng chỉ mục tập trung sẽ cho phép giảm khối lượng dữ liệu vào ra trên server, cho phép tìm kiếm một số lượng lớn tài liệu và bởi nhiều người cùng một lúc. Nó còn cho phép liệt kê kết quả theo thứ tự liên quan của tài liệu đối với yêu cầu tìm kiếm.

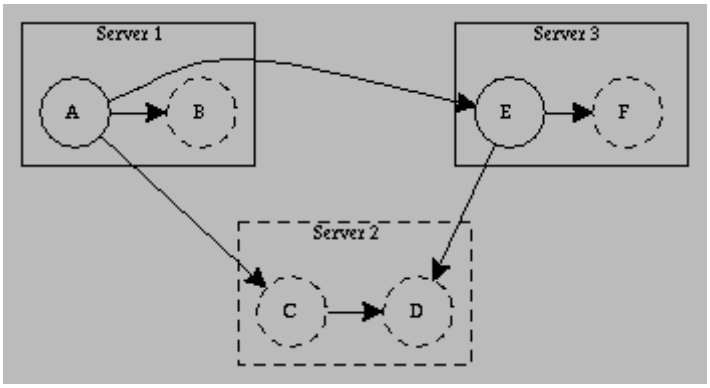
Dưới chúng ta sẽ tìm hiểu kĩ hơn xem các robot tập hợp dữ liệu cho việc xây dựng bảng chỉ mục như thế nào, cách chúng đi theo các liên kết trên Internet, cách chúng đánh chỉ mục tài liệu và cập nhật bảng chỉ mục ...

3.1 Hoạt động của các robot

Các robot bắt đầu từ một trang cho trước, thường thường là trang chủ của một Web site nào đó, nó đọc nội dung của trang đó giống như một trình duyệt Web, và theo các siêu liên kết đến các trang khác. Việc quyết định có đi đến trang khác hay không tuỳ thuộc vào cấu hình của hệ thống. Các robot có thể chỉ cho phép duyệt các trang Web trong phạm vi một server hay một tên miền nào đó.

Một môtơ tìm kiếm phát hiện các tài liệu mới bằng cách bắt đầu với một tập hợp các tài liệu đã biết, kiểm tra các siêu liên kết xuất hiện trong đó, duyệt theo một trong các liên kết đến tài liệu mới, sau đó lặp lại toàn bộ quá trình này. Tưởng tượng Web như là một đồ thị có hướng và việc tìm kiếm đơn giản chỉ là duyệt qua đồ thị sử dụng với một thuật toán duyệt đồ thị nào đó.

Hình dưới chỉ ra một ví dụ. Giả sử rằng để duyệt qua tài liệu A trên Server1 và tài liệu E trên Server3 và bây giờ môtơ quyết định xem tài liệu mới nào sẽ được duyệt tiếp. Tài liệu A có các liên kết đến tài liệu B, C, tài liệu E có các liên kết đến tài liệu D và F. Môtơ tìm kiếm sẽ lựa chọn một trong các tài liệu B, C hoặc D để duyệt tiếp dựa trên yêu cầu tìm kiếm đang được thực hiện.



Hình 7: hoạt động của robot

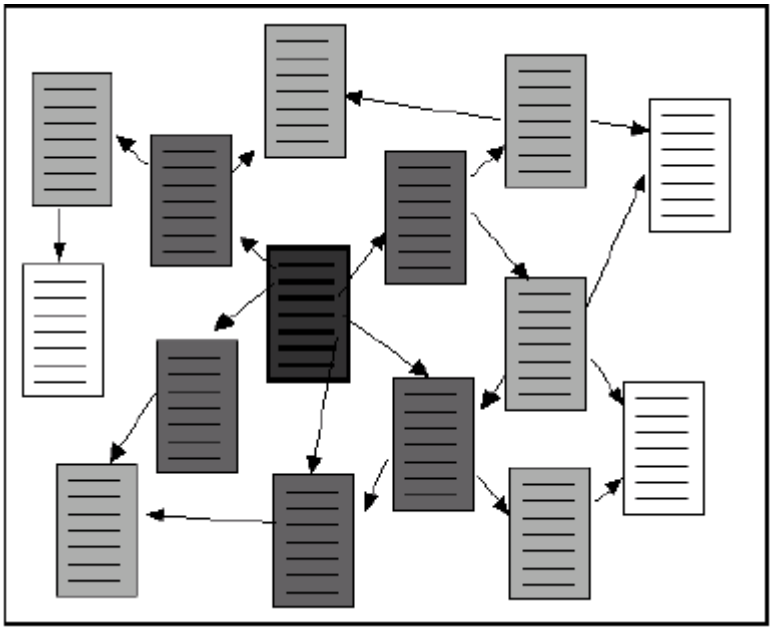
3.2 Duyệt theo chiều rộng

Ý tưởng duyệt theo chiều rộng mục đích là tập hợp được tất cả các trang xung quanh điểm xuất phát trước khi theo các liên kết đi ra xa điểm bắt đầu. Đây là cách thông thường nhất mà các robot hay làm. Nếu việc thực hiện đánh chỉ mục trên một vài server thì khối lượng yêu cầu tới các server được phân phối đều nhau, vì thế làm tăng hiệu quả tìm kiếm. Chiến lược này cũng giúp cho việc cài đặt cơ chế xử lý song song cho hệ thống.

Trong đồ thị dưới đây, trang bắt đầu ở giữa được tô màu đậm nhất. Các trang tiếp theo, tô màu đậm vừa sẽ được đánh chỉ mục đầu tiên, sau đó mới đến các trang được tô màu nhạt hơn cuối cùng đến các trang màu trắng.

Ý tưởng duyệt theo chiều rộng mục đích là tập hợp được tất cả các trang xung quanh điểm xuất phát trước khi theo các liên kết đi ra xa điểm bắt đầu. Đây là cách thông thường nhất mà các robot hay làm. Nếu việc thực hiện đánh chỉ mục trên một vài server thì khối lượng yêu cầu tới các server được phân phối đều nhau, vì thế làm tăng hiệu quả tìm kiếm. Chiến lược này cũng giúp cho việc cài đặt cơ chế xử lý song song cho hệ thống.

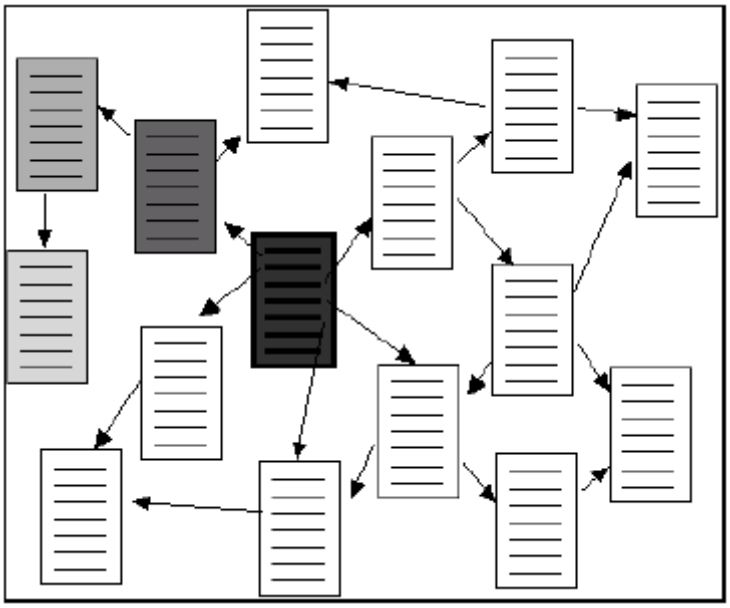
Trong đồ thị dưới đây, trang bắt đầu ở giữa được tô màu đậm nhất. Các trang tiếp theo, tô màu đậm vừa sẽ được đánh chỉ mục đầu tiên, sau đó mới đến các trang được tô màu nhạt hơn cuối cùng đến các trang màu trắng.



Hình 8: mô hình tìm kiếm theo chiều rộng

3.3 Duyệt theo chiều sâu

Theo cách duyệt này, các robot đi theo các liên kết từ liên kết thứ nhất trong trang bắt đầu, sau đó đến liên kết thứ nhất trong trang thứ hai và tiếp tục như thế. Khi nó đánh chỉ mục được các liên kết đầu tiên của mỗi trang, nó tiếp tục tới các liên kết thứ hai và tiếp theo. Một số robot đơn giản dùng phương pháp này vì nó dễ cài đặt.



Hình 9: mô hình tìm kiếm theo chiều sâu

3.4 Độ sâu giới hạn

Một vấn đề đối với các robot là độ sâu giới hạn cho phép chúng trong khi duyệt một Web site. Trong ví dụ về duyệt theo độ sâu ở trên, trang bắt đầu có độ sâu 0, và độ xám của các trang chỉ ra 3 mức liên kết với các độ sâu 1, 2, 3. Đối với một số Web site, thông tin quan trọng nhất thường gần với trang chủ và các trang có độ sâu lớn hơn thường ít liên quan đến chủ đề chính. Một số khác ở vài mức đầu tiên chứa chủ yếu là các liên kết còn nội dung chi tiết lại ở các mức sâu hơn. Trong trường hợp này, các robot phải đảm bảo đánh chỉ mục được các trang chi tiết bởi vì chúng có giá trị đối với những người muốn tìm kiếm trên Web site đó. Cũng có một số robot chỉ đánh chỉ mục ở một vài mức đầu tiên mục đích để tiết kiệm không gian lưu trữ.

3.5 Vấn đề tắc nghẽn đường chuyền

Các Web robot, giống như các trình duyệt, có thể dùng nhiều kết nối tới một Web Server để đọc dữ liệu. Tuy nhiên, điều này có thể làm các server quá tải với việc bắt chúng phải trả lời hàng loạt yêu cầu của robot. Khi kiểm tra hoạt động của server hoặc phân tách các thông báo truy vấn từ bên ngoài, người quản trị mạng có thể phát hiện ra rất nhiều yêu cầu xuất phát từ cùng một địa chỉ IP và có thể ngăn chặn robot không cho nó truy cập thông tin từ đó nữa.

Rất nhiều Web robot đã có cơ chế đặt khoảng thời gian trễ đối với các yêu cầu tới cùng một server. Điều này cực kì quan trọng khi robot xuất phát từ một

địa chỉ đơn và server cần đánh chỉ mục có băng thông hẹp hay có rất nhiều truy vấn cùng lúc.

Đối với các server quá tải, đặt biệt là các server với những trang Web có kích thước lớn và ít thay đổi, thì việc kiểm tra ngày tháng cập nhật thông tin là rất cần thiết. Với tập lệnh trong giao thức HTTP: HEAD hay CONDITIONAL GET, các robot có thể lấy các thông tin META về trang Web trong đó có thông tin về thời gian trang Web đã bị thay đổi. Điều này có nghĩa là robot chỉ lấy về các trang Web đã thay đổi chứ không phải là tất cả các trang, do đó làm giảm khối lượng truy vấn tới server một cách đáng kể.

3.6 Hạn chế của các robot

Mỗi khi robot truy nhập một trang Web từ một server nào đó qua giao thức HTTP, giao thức này bao gồm một số thông tin về đặc điểm của phía client và kiểu thông tin yêu cầu trong phần header. Trong đó có trường User-Agent, nó ghi lại tên của client (chương trình gửi yêu cầu), đó hoặc là một trình duyệt hay là một chương trình robot. Người quản trị mạng qua đó có thể biết được hoạt động của robot.

Cũng do cơ chế bảo mật, người quản trị mạng có thể chỉ định những thư mục có thể cho phép robot truy nhập cũng ngăn không cho robot truy nhập vào một số thư mục ví dụ như: CGI, các thư mục tạm, thư mục cá nhân. Tất cả những thông tin này được lưu trong file robots.txt và được đặt trong thư mục gốc.

3.7 Phân tích các liên kết trong trang web

Đối với rất nhiều trang Web việc tìm kiếm các liên kết đến các trang Web khác rất dễ dàng. Các liên kết có dạng URL chuẩn : “<A HREF = “page.html”> (đối với một file trong cùng một thư mục trên cùng một server) hay “<A HREF = http://ww.domain.com/page.html>” (đối với các file trên các server khác nhau).

Tuy nhiên một số Web site việc phát hiện ra các liên kết này không đơn giản như vậy. Tất cả các thẻ JavaScript, Frames, Image Maps và một số thẻ khác có thể làm cho robot không thể phân biệt được đâu là các liên kết trong đó.

3.8 Nhận dạng mã tiếng việt

Tiếng Việt chưa có một bảng mã thống nhất dùng trong cả nước, mỗi vùng quen dùng một loại mã tiếng Việt riêng như các tỉnh phía Bắc hay dùng ABC,

VietWare, phía Nam hay dùng VNI, ĐHBK tpHCM. Điều này gây ra khó khăn khi trao đổi thông tin trên máy tính. Khi ta nhận tập tin tiếng Việt từ máy khác không dùng chung bảng mã tiếng Việt với máy của ta thì ta phải thực hiện thao tác chuyển mã. Nếu đã biết mã nguồn thì công việc trở nên đơn giản hơn, viết một chương trình nhỏ với dữ liệu mã nguồn đã biết ta có thể chuyển đổi mã nhanh chóng. Các phần mềm tiếng Việt thường dùng như VietWare, VNI đều có chức năng chuyển mã biết mã nguồn này. Vấn đề trở nên phức tạp hơn khi mã nguồn không biết, ta phải tự động đoán ra mã nguồn của đoạn văn tiếng Việt gửi đến. Hiện nay với sự bùng nổ của Internet việc trao đổi thông tin trên mạng thành thường xuyên hơn thì nhu cầu nhận dạng tự động mã tiếng Việt là rất lớn. Ta thử tưởng tượng với bất cứ chương trình nào chạy trên Web server có đầu vào là một đoạn tiếng Việt nhận từ các máy client ở các vùng khác nhau sử dụng các bảng mã khác nhau (như chương trình truy cập thông tin sách báo, chương trình chọn bài nhạc, các chương trình hỏi đáp cơ sở dữ liệu từ xa v.v…) đều cần phải nhận dạng loại mã mà client đã dùng để biết đúng ý nghĩa của xâu gửi đến mà đáp ứng yêu cầu của client. Việc nhận dạng mã tiếng Việt còn giúp ta chuyển đổi tất cả các tài liệu trên mạng về một chuẩn mã thuận tiện cho việc xử lý sau này.

Chương 3: ỨNG DỤNG THỬ NGHIỆM KHAI PHÁ DỮ LIỆU TÍCH HỢP TỪ CÁC WEBSITE TUYỂN DỤNG

1. Bài toán:

1.1 Phát biểu bài toán:

Hiện nay do nhu cầu của xã hội, việc tuyển dụng trên các website tuyển dụng khá phổ biến các thông tin việc tìm người và người tim việc được cập nhật liên tục. Các thông tin về việc tìm người bao gồm: Ngành tuyển, doanh nghiệp cần tuyển, công việc, mức lương, độ tuổi, giới tính. Các thông tin về người tìm việc bao gồm: Ngành tuyển, người tuyển, độ tuổi, giới tính, công việc. các thông tin tổng hợp này sẽ giúp các nhà quản lý, các trường đại học biết được xu hướng tuyển của doanh nghiệp, xu hướng chọn ngành nghề của người học, dánh giá về mực lương của mỗi ngành qua đó có điều chỉnh cho phù hợp…

Trong phạm vi của đồ án này, Em sử dụng các kỹ thuật khai phá dữ liệu đối với CSDL Việc tìm người và Người tìm việc nhằm xác định xu hướng tìm việc của người tìm việc và xu hướng tuyển của doanh nghiệp theo ngành thông qua thuật toán Apriori.

1.2 Một số website tìm việc làm nổi tiếng của việt nam:

http://www.vietnamworks.com	
Người tìm việc	**Việc tìm người**
Tóm lược	**Sơ lược về Công ty**
Họ tên	Sơ lược về công ty
Địa chỉ email	Quy mô công ty
Bằng cấp cao nhất	Địa chỉ công ty
Cấp bậc hiện tại	**Chi tiết công việc**
Tổng số năm	Chức danh
Kinh nghiệm	Mô tả công việc
Công việc gần đây nhất	yêu cầu chung
Công việc mong muốn	Nhận hồ sơ bằng ngôn ngữ
Vị trí	Kỹ năng bắt buộc
Cấp bậc	Loại hình làm việc
Loại hình	Nơi làm việc
Ngành nghề	Ngành nghề
Nơi làm việc	Cấp bậc tối thiểu
Mức lương mong muốn	Mức lương

http://www.tuyendungnhanh.com	
Người tìm việc	**Việc tìm người**
Tóm lược	**Sơ lược về Công ty**
Họ tên	Công ty
Địa chỉ email	Mô tả
Bằng cấp cao nhất	Điện thoại
Kĩ năng cá nhân	Quy mô
	Tiêu chí hoạt động
	Email
	Website
Cấp bậc hiện tại	**Chi tiết công việc**
Tổng số năm	Chức danh/vị trí
Kinh nghiệm	Số lượng tuyển
Công việc gần đây nhất	Lĩnh vực ngành nghề
Công việc mong muốn	Địa điểm làm việc
Vị trí	Mô tả việc làm
Chức danh	Kỹ năng tối thiểu
Mô tả công việc	Trình độ tối thiểu
Mức lương hiện tại	Kinh nghiệm yêu cầu
Mức lương mong muốn	Yêu cầu giới tính
Loại hình công việc	Hình thức làm việc
Ngành nghề muốn	Mức lương
Địa điểm	Thời gian thử việc
	Các chế độ khác
	Yêu cầu hồ sơ
	Hạn nộp hồ sơ

http://www.ungvien.com.vn	
Người tìm việc	**Việc tìm người**
Tóm lược	**Sơ lược về Công ty**
Họ tên	Tên công ty
Địa chỉ email	Tóm lược công ty
Bằng cấp cao nhất	Địa chỉ công ty
Cấp bậc hiện tại	**Chi tiết công việc**
Tổng số năm	Chức danh
Kinh nghiệm	Ngành nghề
Công việc gần đây nhất	Địa điểm làm việc
Công việc mong muốn	Số lượng tuyển
Vị trí	Mô tả công việc
Cấp bậc	Kinh nghiệm kĩ năng
Loại hình	Trình độ học vấn

Ngành nghề	Yêu cầu kinh nghiệm
Nơi làm việc	Loại hình công việc
Mức lương mong muốn	Mức lương

http://works.vn	
Người tìm việc	**Việc tìm người**
Tóm lược	**Sơ lược về Công ty**
Họ tên	Sơ lược
Tuổi	Quy mô
Địa chỉ	Địa chỉ
Chức danh	**Chi tiết công việc**
Yêu cầu	Chức danh
Khả năng	Mô tả công việc
	Yêu cầu
Công việc mong muốn	Loại hình công việc
Loại hình công việc	Nơi làm việc
Nơi làm việc	Ngành nghề
Ngành nghề	Cấp bậc tối thiểu
Mức lương	Mức lương
Trình độ học vấn	Liên hệ
Kĩ năng	Hạn nộp hồ sơ

http://www.timviecnhanh.com	
Người tìm việc	**Việc tìm người**
Tóm lược	**Sơ lược về Công ty**
Họ tên	Công ty
Ngày sinh	Địa chỉ
Giới tính	Mô tả
Tình trạng hôn nhân	Điện thoại
Địa chỉ	Quy mô
Điện thoại	Tiêu chí hoạt động
Trình độ	Website
email	**Chi tiết công việc**
	Chức danh/ vị trí
	Số lượng tuyển
	Lĩnh vực ngành nghề
Công việc mong muốn	Địa điểm làm việc
Chức danh	Kỹ năng tối thiểu
Mô tả công việc	Trình độ tối thiểu
Mức lương	Kinh nghiệm yêu cầu

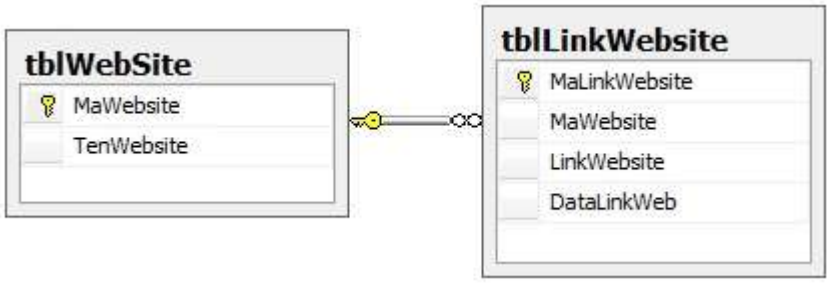
Địa điểm	Yêu cầu giới tính
Trình độ học vấn	Hình thức làm việc
Kinh nghiệm	Mức lương

1.3 Thiết kế cơ sở dữ liệu:

Hiện nay do sự bùng nổ của công nghệ thông tin, nhu cần tuyển dụng trực tuyến trở lên phù hợp hơn với các ứng viên và các nhà tuyển dụng so với cách tuyển dụng truyền thống. Với cách tuyển dụng này các ứng viên hay nhà tuyển dụng chỉ cần truy cập vào các website tuyển dụng tìm các công việc, hay các hồ sơ ứng viên phù hợp với khả năng của các ứng hay, nhà tuyển dụng và các ứng viên sẽ hộp hồ sơ trực tiếp qua email cho các nhà tuyển dụng, cho các ứng viên. Với cách tuyển dụng mới này cũng giúp cho các nhà quản lý đỡ mất thời gian trong việc thu thập thông tin về việc làm của các cơ quan quản lý có thể nắm bắt được nhu cầu việc làm của xã hội và có thể từ các thông tin việc làm trong csdl việc làm có thể rút ra các tri thức hay các xu hướng công việc và là nguồn thông tin giúp trường đại học dân lập hải phòng xác định xu hướng ngành nghề góp phần định hướng đào tạo của trường.

Việc thu thập thông tin việc làm từ các trang web một cách tự động làm cho việc thu thập thông tin một cách nhanh chóng và chính xác. Do các web site được tổ chức dưới dạng phân cấp, chính vì vậy ta phải lưu lại các đường dẫn(url) và một số thông tin quan trọng của website. Việc tạo cơ sở dữ liệu để lưu các thông tin cần thiết phục vụ cho việc lấy dữ liệu một các tự động từ các web site giúp cho công việc lấy thông tin được nhanh hơn. Thông tin cần lưu lại để phục vụ việc lấy thông tin một các tự động từ các website bao gồm: tên website, các liên kết có bên trong website, dữ liệu của các liên kết trong website đó...

Ta có mô hình cơ sở dữ liệu như sau:



Hình 10: mô hình csdl lấy data từ website

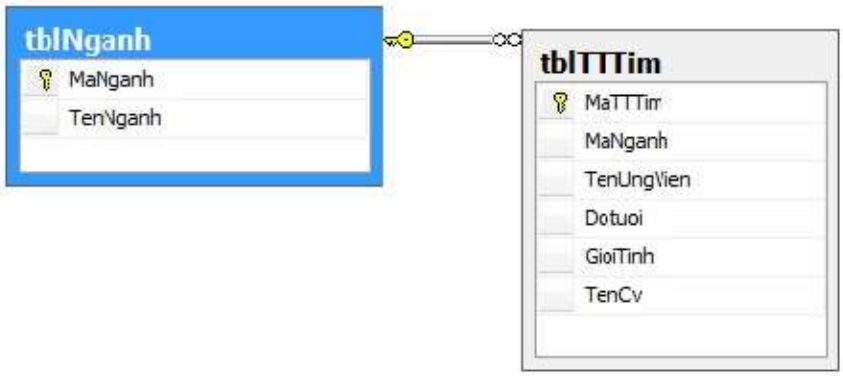
Qua tìm hiểu hồ sơ của các website tuyển dụng nổi tiếng của việt nam có thể chia thành hai loại thông tin như sau: Thông tin việc tìm người và người tìm việc. Các thông tin về việc tìm người bao gồm: Ngành tuyển, doanh nghiệp cần tuyển, công việc, mức lương, độ tuổi, giới tính. Các thông tin về người tìm việc bao gồm: Ngành tuyển, người tuyển, độ tuổi, giới tính, công việc...

Bảng mô hình người tìm việc

Bảng Ngành	
MaNganh	Int
TenNganh	Nvarchar(100)

Bảng thông tin tìm việc	
MaTTTim	Int
MaNganh	Int
TenUngVien	Nvarchar(50)
Dotuoi	Int
Gioitinh	Boolean
TenCv	Nvarchar(30)

Ta có mô hình cơ sở dữ liệu quan hệ:



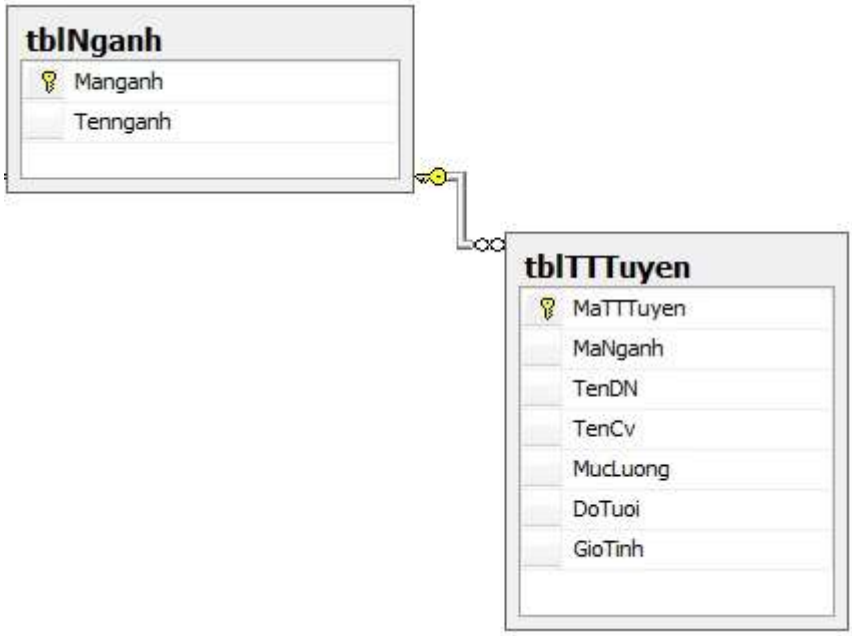
Hình 11: mô hình CSDL tìm việc

Ta có cơ sở dữ liệu Việc tìm người như sau:

Bảng Ngành	
MaNganh	Int
TenNganh	Nvarchar(100)

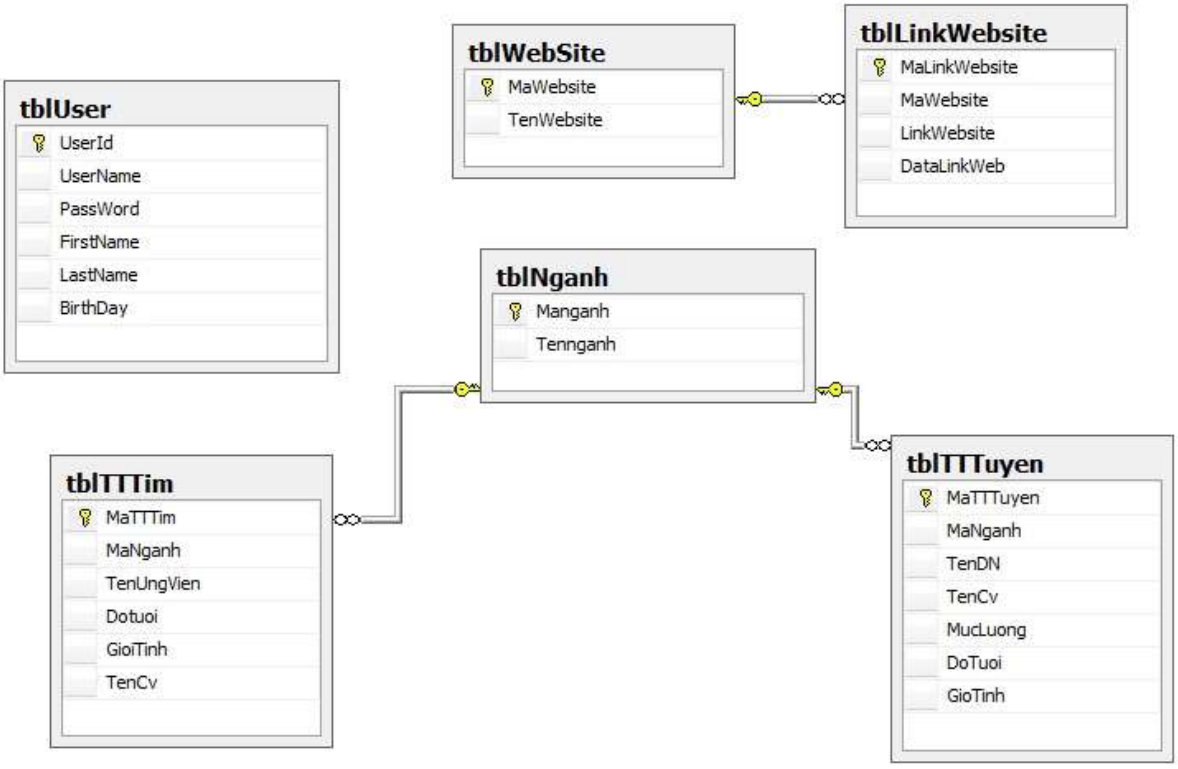
Bảng thông tin tuyên dụng	
MaTTTuyen	Int
MaNganh	Int
TenDN	Nvarchar(50)
MucLuong	Money
Gioitinh	Boolean
TenCv	Nvarchar(30)
Dotuoi	Int

Ta có mô hình cơ sở dữ liệu quan hệ:



Hình 12: mô hình CSDL tuyển dụng

Từ việc phân tích như trên, ta có sơ đồ quan hệ để lưu trữ dữ liệu của bài toán như sau:



Hình 13: mô hình CSDL của chương trình

1.4 Đặc tả dữ liệu:

Một đặc điểm mang tính thực tế là các item không đơn thuần chỉ được xét là "Có" hay "Không" trong khi đếm Support mà mỗi item được kèm theo một trọng số mô tả mức quan trọng của item đó. Các item ta vẫn xem xét thường ở dạng Boolean. Chúng mang giá trị là "1" nếu item có mặt trong giao tác và "0" nếu ngược lại. Các bài toán khai phá dữ liệu như trên người ta vẫn gọi là khai phá dữ kiểu nhị phân (Mining Boolean Association Rules).

Nhưng trong thực tế, các bảng số liệu thường xuất hiện các thuộc tính không đơn giản như vậy. Các thuộc tính có thể ở dạng số (quantitative) như: mức lương, độ tuổi, Các thuộc tính có thể ở dạng Hạng mục (categorical) như: Tên Ngành, Tên Công Việc, Giới tính, … Ta phải rời rạc hóa đưa về dạng bài toán phai phá kết hợp định lượng (Mining Quantitative Association Rules). Cũng như các bài toán khai phá luật kết hợp trước đây, mục tiêu của bài toán khai phá luật kết hợp định lưỡng cũng là kết xuất các luật kết hợp trên các ngưỡng support tối thiểu và các ngưỡng confidence tối thiểu.

Với các thuộc tính hạng mục thì ta phải thực hiện phân đoạn cho các thuộc tính này vì làm như vậy sẽ dễ dàng ánh xạ các thuộc tính tịnh lượng sang các thuộc tính boonlean. Nếu các thuộc tính phân loại hoặc số lượng chỉ có vài giá trị riêng biệt(ví dụ: giới tính) thì có thể ảnh xạ như sau: Mỗi thuộc tính trong bảng dữ

liệu có p giá trị riêng biệt sẽ được lập thành p thuộc tính Boolean mới. Mỗi thuộc tính Boolean mới này tương ứng với một cặp <attribute,value>. Nó có giá trị "1" nếu value có mặt trong dữ liệu gốc và có giá trị "0" nếu ngược lại. Nếu số giá trị riêng biệt của một số thuộc tính khá lớn thì người ta thực hiện việc phân đoạn thuộc tính thành các khoảng và ánh xạ mỗi cặp <attribute,value> thành một thuộc tính. Sau khi ánh xạ, có thể thực hiện khai phá luật kết hợp trên CSDL mới bằng thuật toán khai phá luật kết hợp kiểu Boolean.

Tổng quát, ta có thể đưa ra một số phương pháp rời rạc hoá như sau:

Trường hợp 1 : Nếu A là thuộc tính số rời rạc hoặc là thuộc tính hạng mục có miền giá trị hữu hạng dạng $\{V_1, V_2, \ldots, V_k\}$ và k đủ nhỏ (<100) thì ta biến đổi thuộc tính này thành k thuộc tính nhị phân $A\_V_1, A\_V_2, \ldots, A\_V_k$. Giá trị của bản ghi tại trường A_V_i = True (hoặc 1) Nếu giá trị của bản ghi đó tại thuộc tính A ban đầu bằng v_i, Ngược lại Giá trị của A_V_i = False (hoặc 0).

Trường hợp 2 : Nếu A là thuộc tính số liên tục hoặc A là thuộc tính số rời rạc hay thuộc tính hạng mục có miền giá trị hữu hạng dạng $\{V_1, V_2, \ldots, V_p\}$ (p lớn) thì ta sẽ ánh xạ thành q thuộc tính nhị phân <A:start$_1$. . end$_1$>, <A : start$_2$. . end$_2$>, ... , <A : start$_q$. . end$_q$>. Giá trị của bản ghi tại trường <A : start$_i$. . end$_i$> bằng True (hoặc 1) nếu giá trị của bản ghi đó tại thuộc tính A ban đầu nằm trong khoảng [start$_i$. . end$_i$] , ngược lại giá trị của <A:start$_i$. . end$_i$> = False (hoặc 0).

MaNganh	TenUngVien	Dotuoi	GioiTinh	TenCv
CNTT	Nguyễn Văn dũng	25	1	Lập trình viên
CNTT	Nguyễn Văn hà	27	1	Lập trình viên
CNTT	Nguyễn Thị Linh	24	0	Quản trị mạng
CNTT	Nguyễn Thị Hồng Ngân	23	0	Quản trị mạng
CNTT	Đinh Mạnh Dũng	23	1	Kĩ thuật Viên
CNTT	Phạm thị Linh	23	0	Quản trị mạng
CNTT	Phạm Công Tâm	23	1	Kĩ thuật viên
CNTT	Phạm thị thu hà	23	0	Quản trị mạng
CNTT	Trần thanh tùng	23	1	Đồ họa máy tính
Kt	Đỗ thị hà	22	0	kế toán viên
Kt	Trần bíc thủy	26	0	kế toán trưởng
Kt	Trần thị thủy	23	0	kế toán viên
Kt	Trần thị phượng	23	0	kế toán viên
Kt	Phạm thanh tùng	25	1	kế toán trưởng
Kt	Phạm thanh hưng	25	1	kế toán trưởng
….	….	….	….	….

Bảng 5: CSDL về thông tin tìm việc

Ví dụ: Với bảng số liệu trên đây ta có thể phân chia như sau:

Thuộc tính Độ tuổi là thuộc tính có nhiều giá trị, ta có thể phân thành các khoảng <20, 20..23,24..26, >26. Khi đó, trong tập dữ liệu mới có các thuộc tính

(số người tìm việc có độ tuổi < 20, số người tìm việc có độ tuổi từ 20…23, số người tìm việc có độ tuổi từ 23…26 và số người tìm việc có độ tuổi< 26) tương ứng với thuộc tính độ tuổi. các thuộc tính khác có thể phân chia tương tự nhưng có thể khoảng phân chia khác nhau.

Như vậy CSDL ánh xạ từ CSDL ban đầu sẽ là:

MaNganh	TenUngVien	Dotuoi				GioiTinh	TenCv
		Dotuoi< 20	20<Dotuoi <23	23<Dotuoi <26	26<Dotuoi		
CNTT	Nguyễn Văn dũng	0	0	1	0	Nam	Lập trình viên
CNTT	Nguyễn Văn hà	0	0	0	1	Nam	Lập trình viên
CNTT	Nguyễn Thị Linh	0	0	1	0	Nữ	Quản trị mạng
CNTT	Nguyễn thị Ngân	0	1	0	0	Nữ	Quản trị mạng
CNTT	Đinh Mạnh Dũng	1	0	0	0	Nam	Kĩ thuật Viên
CNTT	Phạm thị Linh	0	1	0	0	Nữ	Quản trị mạng
CNTT	Phạm Công Tâm	1	0	0	0	Nam	Kĩ thuật viên
CNTT	Phạm thị thu hà	0	1	0	0	Nữ	Quản trị mạng
CNTT	Trần thanh tùng	0	1	0	0	Nam	Đồ họa máy tính
Kt	Đỗ thị hà	0	0	1	0	Nữ	kế toán viên
Kt	Trần bíc thủy	0	1	0	0	Nữ	kế toán trưởng
Kt	Trần thị thủy	0	1	0	0	Nữ	kế toán viên
Kt	Trần thị phượng	0	0	1	0	Nữ	kế toán viên
Kt	Phạm thanh tùng	0	0	1	0	Nam	kế toán trưởng
Kt	Phạm thanh hưng	0	0	1	0	Nam	kế toán trưởng
….	….	….	….	….	….	….	….

Bảng 6: Dữ liệu đã chuyển đổi từ dạng số lượng sang dạng boolean

Việc ánh xạ như trên có thể xảy ra vấn đề sau:

"minsup": Nếu số lượng khoảng cho thuộc tính số lượng(hoặc số các giá trị riêng cho các thuộc tính hạng mục) là lớn thì support cho các khoảng có thể là nhỏ. Do đó, việc chia một thuộc tính ra quá nhiều khoảng có thể làm cho luật chứa nó không đạt được support tối thiểu.

"minconf": Một số thông tin có thể bị mất dữ liệu do việc chia khoảng. Một số luật có thể có minconf chỉ khi một item trong chúng có giá trị đơn hoặc

một khoảng rất nhỏ, do đó thông tin có thể bị mất. Sự mất mát thông tin càng tăng khi kích thước khoảng chia càng lớn.

Như vậy. nếu kích thước khoảng là quá lớn(số khoảng nhỏ) thì có nguy cơ một số luật sẽ không có confidence tối thiểu, còn kích thước các khoảng là quá nhỏ (số khoảng lớn) thì một số luật có nguy cơ không có độ support tối thiểu.

Để giải quyết các vấn đề trên, người ta chú ý đến tất cả các vùng liên tục trên thuộc tính số lượng hoặc trên tất cả các phân đoạn. Vấn đề "minsup" sẽ được khắc phục bằng cách liên hợp các khoảng gần kề hoặc các giá trị gần kề. Vấn đề "minconf" sẽ được khắc phục bằng cách tăng số lượng khoảng mà không ảnh hưởng đến vấn đề "minsup".

Người ta có thể thực hiện một phương pháp đơn giản để thực hiện việc chuyển các thuộc tính số lượng và hạng mục về cùng một dạng với nhau. Với thuộc tính phân loại, các giá trị của nó sẽ được ánh xạ vào tập các số nguyên liên tiếp. Với các thuộc tính số lượng không cần khoảng chia (tức là có ít giá trị) thì các giá trị sẽ được ánh xạ vào tập các số nguyên liên tiếp theo thứ tự của các giá trị đó. Còn đối với các thuộc tính số lượng được phân khoảng, thì các khoảng sẽ được ánh xạ vào tập số nguyên liên tiếp, trong đó thứ tự các khoảng này sẽ được bảo tồn. Các ánh xạ này sẽ làm cho mỗi bản ghi trong CSDL trở thành một tập các cặp <Attribute,value>. Bài toán khai phá luật kết hợp lúc này có thể thực hiện qua các bước sau:

Bước 1: Xác định số lượng mỗi phần tử chia cho mỗi thuộc tính số lượng.

Bước 2: Với các thuộc tính hạng mục, ánh xạ các thuộc tính vào tập số nguyên liên tiếp. Với các thuộc tính số lượng không cần sự phụ thuộc khoảng, ánh xạ các giá trị của chúng vào tập các số nguyên liên tiếp theo thứ tự giá trị thuộc tính. Với các thuộc tính số lượng đã được phân khoảng, ánh xạ các khoảng được chia vào tập các số nguyên liên tiếp và bảo tồn thứ tự các khoảng. Bằng cách này, thuật toán chỉ xem các giá trị hoặc các vùng giá trị như là các thuộc tính định lượng.

Bước 3: Tìm độ support cho mỗi giá trị của các thuộc tính hạng mục và thuộc tính số lượng, tiếp theo tìm tất cả các itemset và support của nó lớn hơn minsupport.

Bước 4: Sử dụng các tập tìm được để sinh ra các luật kết hợp.

Bước 5: Xác định luật đáng quan tâm và kết xuất chúng.

Như vậy, khi xét trên CSDL là hồ sơ tìm việc của các ứng viên xin việc (MaNganh, TenUngVien, Dotuoi, GioiTinh, TenCv) trên các website tuyển dụng

lớn trong nước, ta có thể thực hiện việc phân chia các thuộc tính trong bảng thành các khoảng và kí hiệu như nhau:

Mã ngành:

Ngành Xây dựng Điện

Ký hiệu: A B

Ngành Văn hóa du lịch tài chính ngân hàng

Ký hiệu: C D

Ngành Công nghệ thông tin Ngành Kế toán

Ký hiệu: E F

Ngành Quản trị

Ký hiệu: G

Độ tuổi:

Dotuoi <20 20<Dotuoi<23 23<Dotuoi<26 26<Dotuoi

Ký hiệu : H I J K

Giới tính :

Nam Nữ

Ký hiệu N M

TenUngVien	TenCv	MaNganh	Dotuoi				GioiTinh	
			Dotuoi<20	20<Dotuoi<23	23<Dotuoi<26	26<Dotuoi	Nam	Nữ
Nguyễn Văn dũng	Lập trình viên	E			J		N	
Nguyễn Văn hà	Lập trình viên	E				K	N	
Nguyễn Thị Linh	Quản trị	G			J			M
Nguyễn Thị Ngân	Quản trị	G		I				M
Đinh Mạnh Dũng	Kĩ thuật Viên	E	H				N	
Phạm thị Linh	Quản trị mạng	E		I				M
Phạm Công Tâm	Kĩ thuật viên	E	H				N	
Phạm thị thu hà	Quản trị	G		I				M
Trần thanh tùng	Đồ họa máy tính	E		I			N	

Đỗ thị hà	kế toán viên	F			J			M
Trần bíc thủy	kế toán trưởng	F		I				M
Trần thị thủy	kế toán viên	F			J			M
Trần thị phượng	kế toán viên	F		I				M
Phạm thanh tùng	kế toán trưởng	F			J		N	
Phạm thanh hưng	kế toán trưởng	F			J		N	
….	….	….	….	….	….	….	….	….

Chương trình chạy thử nghiệm nhận được là (kết quả này tùy thuộc vào minsupp và minconf, dưới đây là kết quả nhận được với minsupp=0.1 và minconf=0.1):

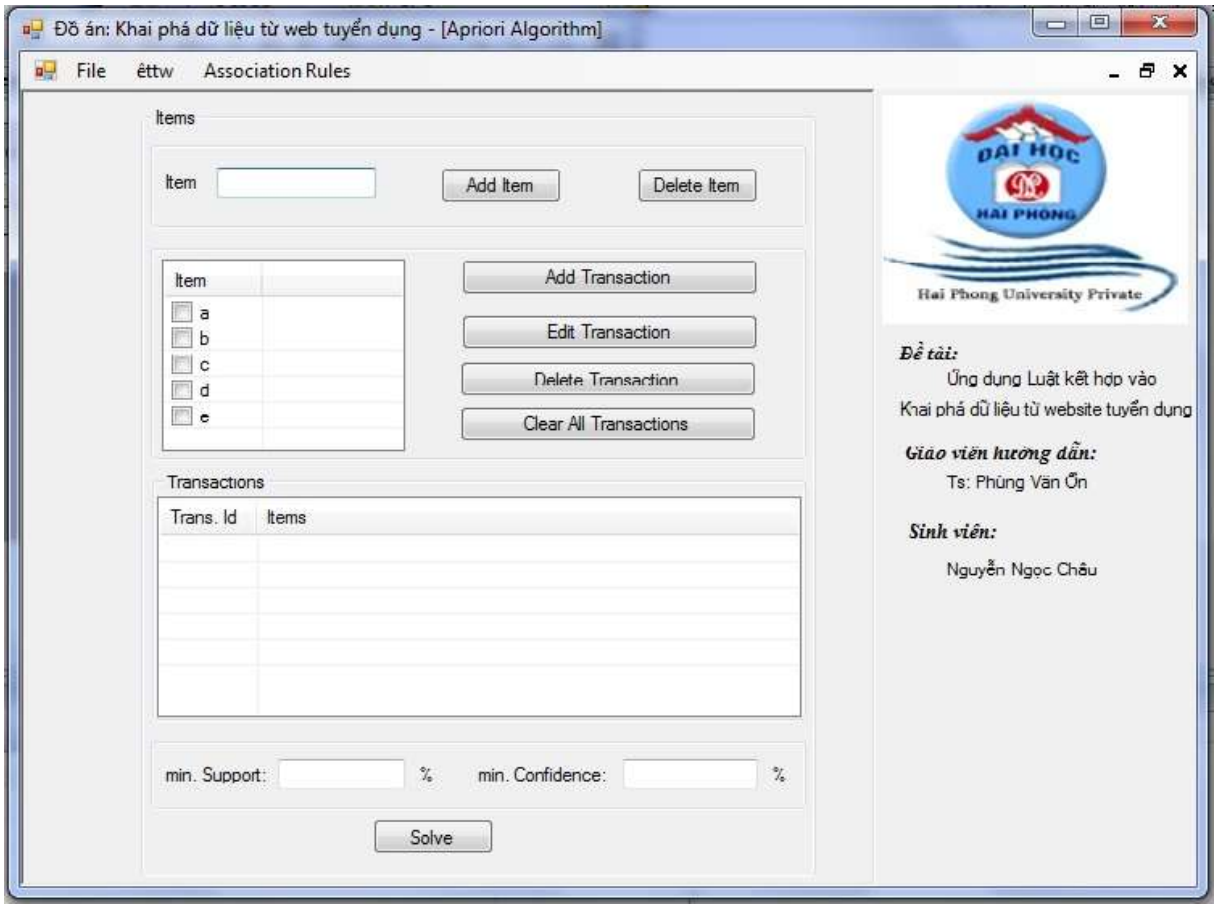
Luật kết hợp	**Supp**	**Conf**
Công nghệ thông tin =>độ tuổi [23-26]	0.7104	0.9023
Công nghệ thông tin =>độ tuổi [<20]	0.4409	0.9266
Công nghệ thông tin =>độ tuổi [20-23]	0.554	0.9687
Công nghệ thông tin => Nam	0.854	0.9885
Công nghệ thông tin =>độ tuổi [>26]	0.5573	0.9765
Công nghệ thông tin => Nữ	0.4901	0.9654
Kế toán =>Dotuoi[20-23]	0.4409	0.9605
Kế toán =>Dotuoi[23-26]	0.6737	0.9722
Kế toán =>Dotuoi[>26]	0.5081	0.9117
Kế toán =>Nam	0.5409	0.9166
Kế toán =>Nữ	0.5737	1
…	…	…

Dựa vào kết quả trên ta nhận thấy rằng ngành công nghệ thông tin có cần tuyển nhiều nhất thường có độ tuổi từ 23 => 26 thường là nam, ngành kế toán kiểm toán số lượng tuyển nhiều nhất thường có độ tuổi 23=>26 chủ yếu là nữ…

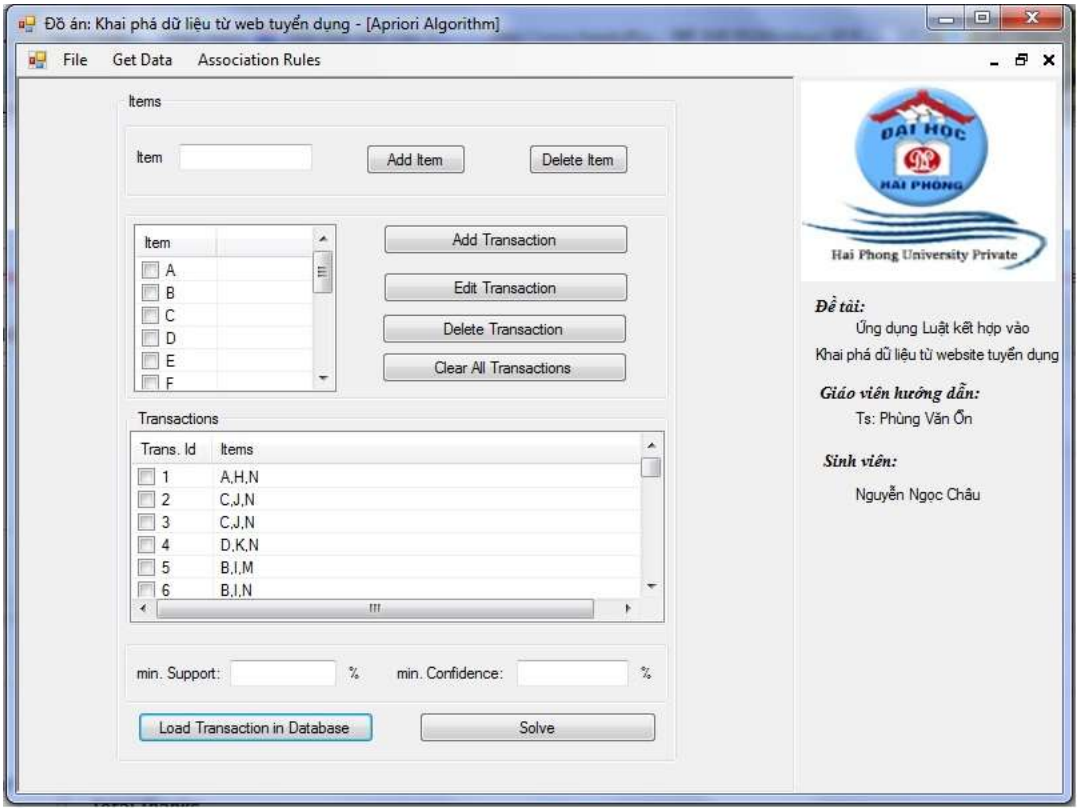
1.5 Minh họa chương trình

Chương trình được cài đặt bằng ngôn ngữ C#.net, CSDL thiết kế trên SQL 2005, hệ điều hành window 7, chíp dua 2 core T6500 2.1 Mhz, RAM 2GB, Ổ cứng 250GB còn trống 50Gb.

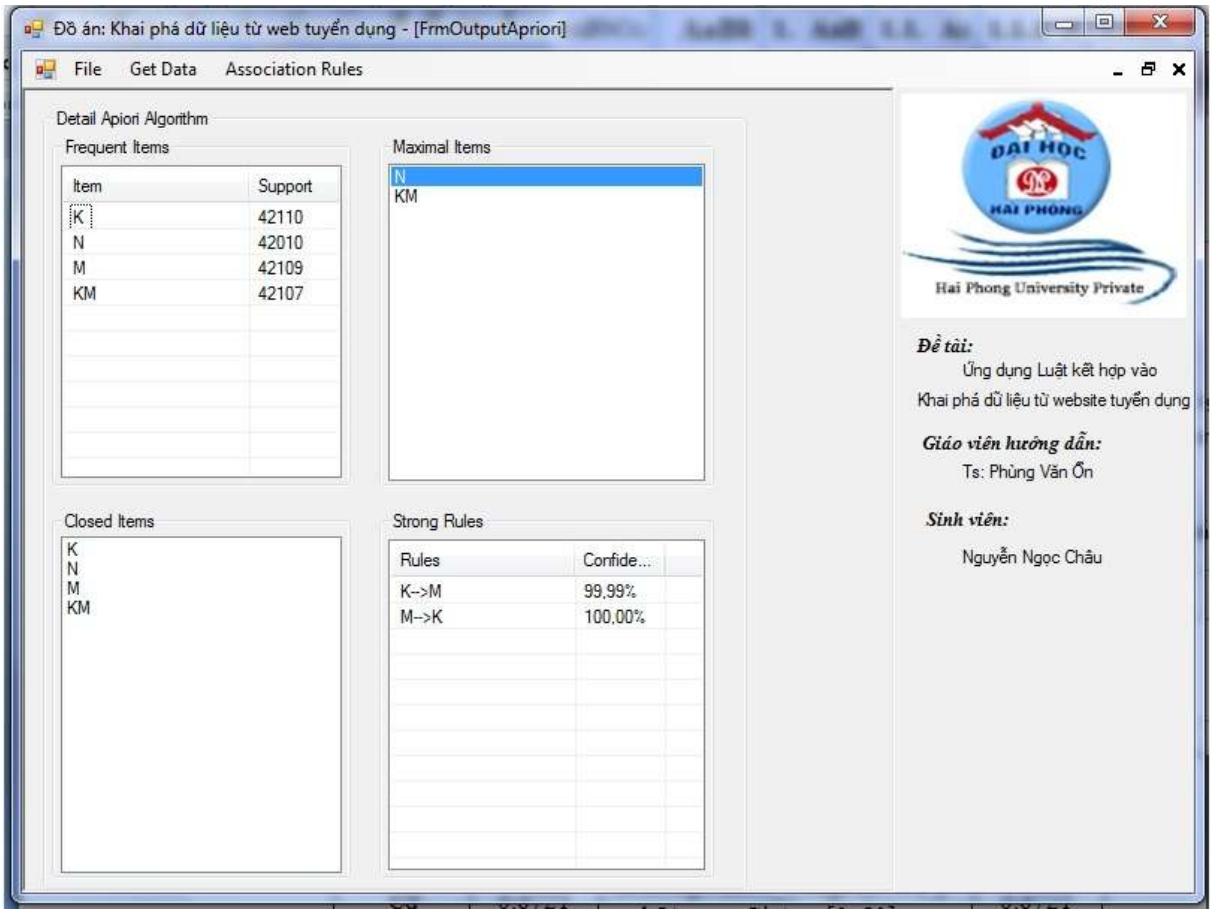
Chương trình có một số giao diện chính sau:



Hinh 14: giao diện chương trình



Hình 15: quá trình tạo luật kết hợp theo thuật toán Apriori



Hình 16: luật thu được

1.6 Phân tích đánh giá

Chương trình thực hiện tìm các tập phổ biến và luật kết hợp thông qua thuật toán Apriori. Ta có một số nhận xét sau:

Để xác định độ Support của các tập ứng viên, thuật toán Apriori luôn luôn phải quét lại toàn bộ các giao tác trong CSDL. Do vậy sẽ tiêu tốn rất nhiều thời gian khi số k-items tăng(số lần duyệt các giao tác tăng).

Trong quá trình xét duyệt khởi tạo thuật toán Apriori, kích thước của C'k là rất lớn và hâu hết tương đương với kích thước của CSDL gốc. Do đó, thời gian tiêu tốn cũng sẽ bằng với thuật toán Apriori.

1.7 Hướng phát triển

- Tiếp tục hoàn thiện và mở rộng chương trình trong đồ án này để có thể áp dụng vào thực tế một cách triệt để. Chương trình thực hiện theo đúng các bước trong quá trình khai phá dữ liệu: **1- Chọn lọc dữ liệu**(chọn lọc, trích rút các dữ liệu cần thiết từ CSDL), **2- làm sạch dữ liệu**(chống trùng lặp và giới hạn vùng giá trị), 3- làm giàu dữ liệu, 4- **khai thác tri thức từ dữ liệu**(tìm tác vụ phát hiện luật kết hợp, trình chiếu báo cáo), **5- chọn dữ liệu có ích áp dụng vào trong hoạt động thực tế**.
- Cho đến nay hầu hết các thuật toán xác định các tập phổ biến đều được xây dựng dựa trên thừa nhận độ hỗ trợ cực tiểu(minsup) là thống nhất, tức là các tập mục được chấp nhận đều có độ support lớn hơn cùng một độ tối thiểu. Điều này không thực tế vì có nhiều ngoại lệ khác được chấp nhận thường có độ hỗ trợ thấp hơn nhiều so với khuynh hướng chung(các tiêu chí phân loại, ưu tiên là khác nhau). Mặt khác, khi xem xét các thuộc tính số lượng rời rạc hóa bằng phương pháp phân khoảng thường tạo ra số khoảng rất lớn. Vì vậy, hướng nghiên cứu tiếp theo của em là luật kết hợp mờ(điều này cũng đang được nhiều người quan tâm).
- Nghiên cứu sâu các thuật toán khai phá dữ liệu và áp dụng vào một số bài toán khai phá dữ liệu phù hợp với giai đoạn hiện nay: dự báo việc làm, định hướng trong kinh doanh…

KẾT LUẬN

Đồ án đề cập đến các nội dung về kho dữ liệu và ứng dụng của lưu trữ và khai phá tri thức trong kho dữ liệu nhằm hỗ trợ ra quyết định.

Về mặt lý thuyết, khai phá tri thức bao gồm các bước: Hình thành, xác định và định nghĩa bài toán, thu thập và tiền xử lý dữ liệu, khai phá dữ liệu, rút ra các tri thức, sử dụng các tri thức phát hiện được. Phương pháp khai phá dữ liệu có thể là: phân lớp, cây quyết định, suy diễn… Các phương pháp trên có thể áp dụng trong dữ liệu thông thường.

Về thuật toán khai phá tri thức, đồ án trình bày một số thuật toán và minh họa một thuật toán kinh điển về phát hiện tập chỉ báo phổ biến và khai phá luật kết hợp là: Apriori

Về mặt cài đặt thử nghiệm, đồ án giới thiệu kĩ thuật khai phá dữ liệu theo thuật toán Apriori áp dụng vào bài toán dự báo xu hướng tìm việc của các ứng viên, xu hướng tuyển dụng của doanh nghiệp.

Trong quá trình thực hiện đồ án, em đã cố gắng tập trung tìm hiểu và tham khảo các tài liệu liên quan. Tuy nhiên, với thời gian và trình độ có hạn nên không tránh khỏi những hạn chế và thiếu sót. Em rất mong nhận được các nhận xét và góp ý của các thầy cô giáo và bạn bè, những người cùng quan tâm để hoàn thiện hơn các kết quả nghiên cứu của mình.

TÀI LIỆU THAM KHẢO

Tiếng việt:

Hoàng Kiếm - Đỗ Phúc, *Giáo trình khai phá dữ liệu* - Trung tâm nghiên cứu phát triển công nghệ thông tin, Đại học Quốc gia thành phố Hồ Chí Minh, 2005.

Nguyễn Lương Thục, *Một số phương pháp khai phá luật kết hợp và cài đặt thử nghiệm* - Luận văn thạc sỹ ngành CNTT, Khoa Tin học, Đại học Sư phạm Huế, 2002.

Tiếng anh:

Bao Ho Tu (1998), *Introduction to Knowledge Discovery and Data mining*, Institute of Information Technology National Center for Natural Science and Technology.

Jean-Marc Adamo (2001), *Data Mining for Association Rule and Sequential Pattens,* With 54 Illustrations. ISBN0-95048-6.

John Wiley & Sons (2003) - *Data Mining-Concepts Models Methods And Algorithms,* Copyright © 2003 The Institute of Electrical and Electronics Engineers, Inc.

John Wiley & Son, *Visual Data Mining: Techniques and Tools for Data Visualization and Mining, by Tom Soukup and Ian Davidson,* ISBN: 0471149993.

John Wiley & Sons (2003), *Data Mining: Concepts, Models, Methods, and Algorithms,* by Mehmed Kantardzic, ISBN:0471228524.

Patrick BOSC - Didier DUBOIS - Henri PRADE, *Fuzzy functional dependencies.*