NHIỆM VỤ ĐỀ TÀI:

Chương 1: ĐẶT VẤN ĐỀ VÀ PHÁT BIỂU BÀI TOÁN

+ Đặt vấn đề

+ Phát biểu bài toán

+ Cách tiếp cận

Chương 2: CƠ SỞ LÝ THUYẾT

+Tìm hiểu tổng quan XML

+ Nghiên cứu các phương pháp hợp nhất các bản tin có cấu trúc XML

Chương 3: ĐÁNH GIÁ THỰC NGHIỆM

+ Thực nghiệm trên các bản tin mẫu và đánh giá kết quả

Kết luận.

Đề hướng phát triển trong tương lai.

Tài liệu tham khảo.

LỜI CẢM ƠN

Trong suốt khóa học 2005 – 2009 tại trường Đại Học Dân Lập Hải Phòng với sự giúp đỡ của quý thầy cô và giáo viên hướng dẫn về mọi mặt, từ nhiều phía nhất là trong thời gian thực hiện đề tài, nên đề tài của em đã được hoàn thành đúng thời gian quy định.

Em xin gửi lời cảm ơn chân thành nhất tới thầy giáo hướng dẫn Th.s Nguyễn Trịnh Đông đã tận tình hướng dẫn, giúp đỡ, tạo điều kiện để em hoàn thành khóa luận này.

Em xin gửi lời cảm ơn chân thành tới Bộ môn Công Nghệ Thông Tin cùng toàn thể các thầy cô trong khoa cũng như toàn thể các thầy cô trong Trường đã giảng dạy những kiến thức chuyên môn làm cơ sở để em thực hiện tốt cuốn luận văn tốt nghiệp này và đã tạo điều kiện thuận lợi để em hoàn thành khóa học.

Em xin chân thành cảm ơn!

Hải Phòng, ngày 1 tháng 7 năm 2009

Sinh Viên

Vũ Thị Lệ

MỤC LỤC

BẢNG CÁC TỪ VIẾT TẮT

STT	Tên viết tắt	Tên đầy đủ
1	CSDL	Cơ sở dữ liệu
2	XML	eXtensible Markup Language
3	DOM	Document Object Model
4	HTTP	Hypertext Transfer Protocoly
5	DTD	Document Type Definition
6	XSLT	eXtensible Stylesheet Language transformations
7	XSL	XML- Schema XML Definitiom
8	SQL	Structured Query Language - SQL
9	T_b	Tập tin cơ sở
10	T_1, T_2	Tập tin nhánh
11	MBCS	Mixed Byte Character Set

CHƯƠNG 1: ĐẶT VẤN ĐỀ VÀ PHÁT BIỂU BÀI TOÁN

1.1 Đặt vấn đề

Trong tương lai gần đây, khi máy tính trở nên phổ biến đến mức nó chuyển từ khuynh hướng sử dụng ý thức sang tiềm thức. Con người chỉ sử dụng máy tính theo nghĩa thông thường là dùng một máy tính PC, hay Laptop để thực hiện công việc của mình mà có một khái niệm mới sẽ nảy sinh trong tương lai, đó là thông tin di động. Hệ thống thông tin di động đang bước đầu hình thành với sự xuất hiện đa dạng của các hình thức Smart phone, PDA…

Một trong những cách thức trao đổi thông tin trong tương lai là sẽ truyền thông tin dưới dạng các bản tin có cấu trúc, chẳng hạn các bản tin XML. Bản tin có cấu trúc là một khái niệm tổng quát ẩn chứa trong cách tiếp cận khác nhau nhằm quản lí thông tin. Về mặt cú pháp một thành phần của bản tin bao gồm một cụm từ và một nhãn ngữ nghĩa. Các thành phần của bản tin có thể lồng vào nhau trong các thành phần lớn hơn. Hầu hết các thông tin được thể hiện ở dạng bản tin, chẳng hạn thẻ trong XML, kiểu text trong các cơ sở dữ liệu quan hệ và hướng đối tượng và các kết quả từ các hệ thống xử lí thông tin.

Việc gia tăng số người dùng muốn áp dụng công nghệ tính toán song song dựa trên nền tảng trao đổi dữ liệu thông qua XML, nghĩa là công nghệ cho phép nhiều người dùng thêm vào cùng một tập dữ liệu đơn đồng thời, dẫn đến phát sinh nhu cầu phải có công cụ hợp nhất dữ liệu XML đủ mạnh để điều quản quá trình cộng tác này. Việc đưa ra một giải pháp nhất quán, linh động và tương thích cho cơ chế tự động hợp nhất là vấn đề được đặt ra trước tiên.

Em đã chọn đề tài làm đồ án tốt nghiệp là***: "Phương pháp hợp nhất các bản tin có cấu trúc XML".*** Với mục đích nghiên cứu các phương pháp hợp nhất các bản tin có cùng cấu trúc một cách nhanh nhất

1.2 Phát biểu bài toán

Trên thực tế ngày nay có nhiều loại nhiều công văn và bản tin sử dụng các định dạng riêng của nó. Chúng ta có nhiều phương pháp hợp nhất khác nhau nhưng việc hợp nhất các bản tin này thành 1 loại bản tin có cấu trúc chung là phương pháp tối ưu nhất. Phương pháp giúp chúng ta xác định ngay tất cả các thay đổi giữa các bản tin, giúp so sánh, hiểu và kết hợp các tập tin mã nguồn khác nhau một cách dễ dàng, nhanh chóng chính xác. Vì vậy việc hợp nhất các bản tin trở nên cần thiết và quan trọng.

Hiện nay phương pháp hợp nhất các bản tin có cấu trúc XML để lưu trữ và trao đổi thông tin là giải pháp được đánh giá cao. XML là một chuẩn định dạng dữ liệu cho nhiều ứng dụng, do bản chất đơn giản và tự giải thích của mình và nó độc lập giữa dữ liệu với ứng dụng.

1.3 Cách tiếp cận

Bản tin có cấu trúc XML đã có cùng cấu trúc hoặc có cấu trúc tương tự nhau, nghĩa là cùng các từ khóa và nội dung. Để giải quyết bài toán hợp nhất ta có hai phương án là hợp nhất 3-way và hợp nhất 2 – way. Nhưng bài toán hợp nhất 3 – way được nghiên cứu chính trong đồ án này.

Bài toán hợp nhất 3-way được phát biểu cụ thể như sau:

Giả sử T_1 và T_2 là hai cây có thứ tự được dẫn xuất từ cây T_b. Chúng ta sẽ phân tích và thiết kế một công cụ có thể:

1 Thực hiện việc hợp nhất 3-way theo cấu trúc các cây T_1 ,T_2 và T_b và phát hiện diễn tả mọi đụng độ xảy ra trong khi hợp nhất. Gọi là bài toán hợp nhất cây.

2 Sinh ra tập khác biệt giữa hai cây T_1 và T_2 dưới dạng một kịch bản chỉnh sửa. Sử dụng tập khác biệt và thông tin của cây T_1 nhận lại được cây T_2. Gọi là bài toán khác biệt và ráp cây.

CHƯƠNG 2: NGHIÊN CỨU CÁC PHƯƠNG PHÁP HỢP NHẤT CÁC BẢN TIN XML

2.1 Tổng quan về XML.

2.1.1 Giới thiệu XML

XML(Extensible Markup Language)ra đời vào tháng 2/1998, là ngôn ngữ có kiến trúc gần giống với HTML nhưng XML nhanh chóng trở thành một chuẩn phổ biến trong việc chuyển đổi thông tin qua các trang web sử dụng giao thức HTTP. Trong khi HTML là ngôn ngữ chủ yếu về hiển thị dữ liệu thì XML lại đang phát triển mạnh về việc chuyển tải, trao đổi và thao tác dữ liệu bằng XML. XML đưa ra một định dạng chuẩn cho cấu trúc của dữ liệu hoặc thông tin bằng việc tự định nghĩa định dạng của tài liệu. Bằng cách này, dữ liệu được lưu trữ bằng XML sẽ độc lập với việc xử lý. Vì vậy XML ra đời sẽ đáp ứng được yêu cầu ngày càng cao của các nhà lập trình trong vấn đề trao đổi và xử lý thông tin.

2.1.2 Khái niệm XML

XML là một chuẩn ngôn ngữ nhằm mục đích cung cấp việc chia sẻ dữ liệu giữa các hệ thống phần mềm theo hướng thân thiện người dùng. XML đang được đẩy mạnh để trở thành ngôn ngữ chung cho việc trao đổi dữ liệu trên internet. XML được hỗ trợ bởi tổ chức World wide web Consortium-W3C và các tập đoàn lớn.

2.1.3 Mục tiêu ra đời của XML

Ngày nay, XML đang trở thành một chuẩn chung cho việc trao đổi dữ liệu cho những ứng dụng chạy trên môi trường Internet. Vì XML cho phép người dùng có thể tự định nghĩa các thẻ (tag) - những thẻ này làm cho tài liệu XML đa dạng hơn những ngôn ngữ thông thường như HTML. Như vậy mục tiêu đặt ra cho sự ra đời XML là gì? Đầu tiên nó sẽ tương thích với SGML và dễ dàng viết những chương trình để xử lý cho những tài liệu XML. Kế tiếp, những tài liệu XML rõ ràng, dễ đọc, dễ dàng tạo lập. Và điều quan trọng là nó được hỗ trợ trong nhiều ứng dụng. Tóm

lại, XML dễ dàng chia sẻ thông tin qua những định dạng khác nhau thông qua môi trường web. XML được thiết kế dành cho mọi người, được mọi người sử dụng.

2.1.4 Lợi ích ưu điểm và hạn chế khi sử dụng XML

Một số lợi ích khi sử dụng XML

XML có thể tách rời dữ liệu. Sử dụng XML, dữ liệu được chứa trong các tập tin XML riêng biệt.

XML có thể mô tả thông tin của những đối tượng phức tạp mà cơ sở dữ liệu quan hệ không thể giải quyết được.

XML có thể dùng để chuyển đổi dữ liệu giữa các hệ thống không tương thích.

XML dùng để chia sẻ dữ liệu với những tập tin bản tin đơn giản dễ hiểu.

XML cũng được dùng để lưu trữ dữ liệu, có thể làm cho dữ liệu của chúng ta hữu ích hơn.

Như vậy, chúng ta đã biết được lợi ích và vai trò của XML trong vấn đề lưu trữ và trao đổi thông tin. Tuy nhiên hạn chế của XML cũng có :

+ Chuẩn hoá: Trong khi đã tồn tại các định nghĩa tên thẻ của ngành, bạn vẫn có thể định nghĩa các thẻ không phải là tiêu chuẩn.

+ Dung lượng lớn.

2.1.5 Cấu trúc chung

Chúng ta có thể sử dụng trình soạn thảo bất kỳ để soạn thảo tài liệu XML, nhưng phải tuân thủ theo nguyên tắc sau:

```
<root>
<child>
<subchild>……</subchild>
…….
</child>
……
</root>
```

Theo định dạng trên, chúng ta thấy tuy tài liệu XML rất đơn giản nhưng quy định cũng rất chặt chẽ, tức là các tài liệu XML đều xuất phát từ nút gốc (root), và mỗi phần tử phải có thẻ mở và thẻ đóng "<tên thẻ > … </ tên thẻ>"

2.1.6 Những thành phần của một tài liệu XML

Khai báo: Mỗi một tài liệu XML có một chỉ thị khai báo

xml version="1.0"?>

Định nghĩa tài liệu XML tuân theo chuẩn của W3C và đây là phiên bản "1.0"

Chú thích: được khai báo như sau:

<!-- chú thích -- >

Phần tử (Elements): Một tài liệu XML được cấu thành từ những phần tử. Một phần tử có thẻ mở và thẻ đóng. Giữa thẻ mở và thẻ đóng là nội dung của phần tử đó. Phần tử có thể chứa dữ liệu hoặc có thể lồng vào một phần tử khác.

Phần tử gốc (root): Trong tài liệu XML, chỉ có một phần tử gốc và phần tử này sẽ chứa tất cả những phần tử của tài liệu XML do chúng ta tạo ra .

Thuộc tính (Attributes): Như đã trình bày ở trên, một phần tử có thể chứa dữ liệu hoặc chứa phần tử khác hoặc cả hai. Bên cạnh đó, phần tử có thể rỗng khi đó nó có thể chứa thuộc tính. Một thuộc tính chỉ là một sự lựa chọn để gắn dữ liệu đến phần tử. Một thuộc tính đặt trong thẻ mở của phần tử và chỉ ra giá trị của nó bằng cách sử dụng cặp name=value".

2.1.7 Lược đồ XML

DTD(Document Type Definition)và Schema là hai cách khác nhau để quy định những luật về nội dung của một tài liệu XML.Tuy nhiên DTD có hạn chế là không sử dụng định dạng XML vì bản thân DTD không phải là một tài liệu XML và kiểu dữ liệu có sẵn dùng để định nghĩa nội dung của một thuộc tính hoặc một phần tử thì rất giới hạn trong DTD mặt khác DTD không có khả năng mở rộng và không hỗ trợ namespace. Do đó tài liệu không viết theo định dạng XML nên DTD khó viết

và khó hiểu.Vì vậy việc sử dụng DTD để kiểm tra sự hợp lệ của một tài liệu XML là không khả thi. Chúng ta cần một sự lựa chọn khác khả thi hơn để kiểm tra sự hợp lệ của một tài liệu XML. Đó là chúng ta sử dụng lược đồ XML-Schema XML Definition(XSD)

Một lược đồ đơn giản chỉ là một tập hợp những luật được định nghĩa lại mô tả nội dung dữ liệu của một tài liệu XML, nó tương tự như một định nghĩa cấu trúc bảng trong cơ sở dữ liệu quan hệ. Trong lược đồ XML, chúng ta định nghĩa một tài liệu XML, những phần tử của nó, những kiểu dữ liệu của phần tử và những thuộc tính liên quan và điều quan trọng nhất là mối quan hệ "cha con" giữa những phần tử. Chúng ta có thể tạo lược đồ trong nhiều cách khác nhau. Cách đơn giản nhất là sử dụng Notepad.

Các kiểu dữ liệu trong lược đồ XML

Có hai loại kiểu dữ liệu trong lược đồ XML đó là kiểu dữ liệu cơ bản và kiểu dữ liệu mở rộng. Kiểu dữ liệu cơ bản là kiểu dữ liệu không bắt nguồn từ kiểu dữ liệu nào ví dụ như kiểu dữ liệu float. Kiểu dữ liệu mở rộng dựa trên những kiểu dữ liệu khác như kiểu integer dựa trên kiểu decimal.

Kiểu dữ liệu cơ bản được định nghĩa cho mục đích của lược đồ XML thì không nhất thiết phải giống với một số cơ sở dữ liệu khác.

XPath

Qua phần trình bày trên, chúng ta biết được cấu trúc và cú pháp của XML tương đối đơn giản. Bước tiếp theo là tìm hiểu cách nào để xử lý một tài liệu XML

Như vậy để xử lý một tài liệu XML, chương trình ứng dụng phải có cách di chuyển bên trong tài liệu để lấy ra giá trị của các phần tử hay thuộc tính. Do đó ngôn ngữ XML Path được ra đời mà chúng ta gọi tắt là XPath. XPath đóng vai trò quan trọng trong việc truy vấn dữ liệu cho các chương trình ứng dụng vì nó cho phép ta lựa chọn hay sang lọc ra những phần tử nào mình muốn để trao đổi hay hiển thị.

Xpath là một ngôn ngữ dùng để xử lý truy vấn trên tài liệu XML, cũng giống như SQL là một chuẩn để làm việc với cơ sở dữ liệu. Một biểu thức XPath có thể chỉ ra vị trí và mẫu nào để kết hợp. Chúng ta có thể áp dụng toán tử Boolean, hàm string

và toán tử số học trong biểu thức XPath để xây dựng câu truy vấn phức tạp trên tài liệu XML. XPath cũng cung cấp một số hàm về số như tính tổng, hàm làm tròn (round),v.v…..

2.1.8 Đọc và phân tích tài liệu XML

Sử dụng lớp XMLTextReader

Lớp XMLTextReader cung cấp một cursor được sử dụng để lấy dữ liệu từ một tài liệu XML

Cách khai báo:

```
XmlTextReader myRdr =
      new XmlTextReader(Server.MapPath("catalog2.xml"));
```

Khi một thể hiện được tạo ra, con trỏ *cursor* sẽ được đặt ở đầu tài liệu. Chúng ta có thể sử dụng phương thức Read() để lấy những phần dữ liệu một cách tuần tự. Mỗi phần tử dữ liệu tương tự như một nút trong cây XML. Thuộc tính NodeType sẽ lấy giá trị của nút. Vì thế, khi một phần dữ liệu được đọc, chúng ta có thể sử dụng câu lệnh sau để hiển thị tên, giá trị và kiểu của nút

```
Response.Write(myRdr.NodeType.ToString()
      +" " + myRdr.Name + ":" + myRdr.Value);
```

Nếu muốn kiểm tra nút đó có thuộc tính hay không, chúng ta có thể sử dụng phương thức ***HasAttributes***. Nếu giá trị trả về của phương thức ***HasAttributes*** là *true*, chúng ta áp dụng phương thức ***MoveToAttribute(i)*** để lặp qua các thuộc tính của nút.

Sử dụng mô hình DOM(Document Object Module)

Mô hình DOM để đọc và trình bày nội dung của một tệp tin XML. Việc sử dụng mô hình DOM sẽ thông qua một số đối tượng như XMLDocument, XMLDataDocument.

Khi một XMLDocument được tạo ra, nó tổ chức nội dung của một tệp tin XML thành một cây. XMLDocument cung cấp việc truy xuất nhanh và trực tiếp đến

một nút. Tuy nhiên việc sử dụng mô hình DOM rất tốn bộ nhớ để lưu trữ thành một cây và thật sự sẽ khó khăn khi tài liệu XML có kích thước lớn.

2.1.9 Định hướng qua tài liệu XML để rút trích dữ liệu

Sử dụng lớp XMLTexeReader

Trong phần trên, chúng ta đã biết cách để đọc vào một tài liệu XML, phần này chúng ta sẽ định hướng qua tài liệu XML và chỉ lấy những dữ liệu nào cần thiết cho ứng dụng của mình.

Sử dụng mô hình DOM

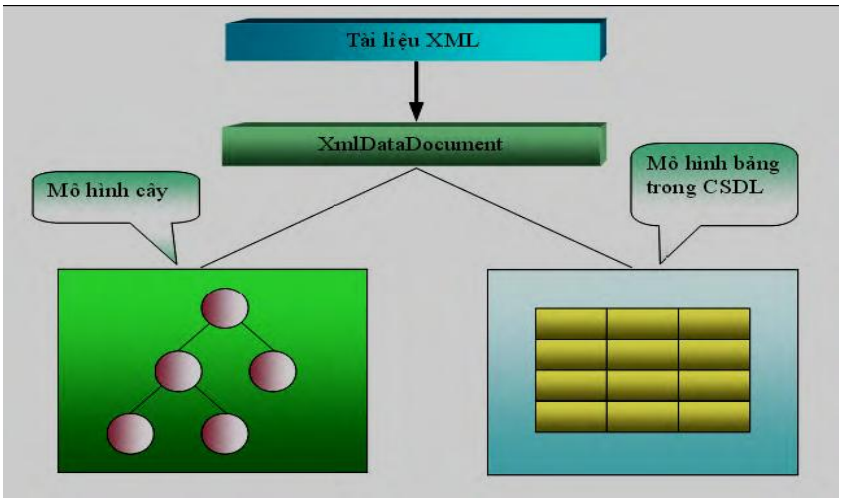
Bên cạnh XMLTextReader thì môi trường Visual Studio.NET cũng hỗ trợ mô hình DOM để đọc và trình bày nội dung của một tập tin XML. Việc sử dụng mô hình DOM sẽ thông qua một số đối tượng như XMLDocment, XMLDataDocument. Khi một XMLDocment được tạo ra, nó tổ chức nội dung của một tập tin XML thành một cây. Trong khi đối tượng XMLTextReader cung cấp một *cursor* định vị trí theo một hướng thì XMLDocumemt cung cấp việc truy xuất nhanh và trực tiếp đến một nút. Tuy nhiên, việc sử dụng mô hình DOM rất tốn bộ nhớ để lưu trữ thành một cây và thất sự sẽ khó khăn khi tài liệu XML có kích thước lớn.

Có nhiều cách khác nhau để tạo một đối tượng XMLDocument. Sau đây chúng ta sử dụng đối tượng XMLTextReader để tạo một XMLDocumnet.

Một cây được tạo từ nhiều nút và một nút cũng là một cây chứa những nút khác. Nút lá thì không có nút con, vì thế nút này dùng để hiển thị dữ liệu bản tin.
Lớp XMLDataDocument kế thừa từ lớp XMLDocument vì thế nó cũng có một số phương thức như lớp XMLDocument. XMLDataDocument cung cấp hai cái nhìn trên cùng một dữ liệu đó là XML view và relational view.

XMLDataDocument có một thuộc tính tên là DataSet, thông qua DataSet, XMLDataDocument trình bày dữ liệu như một hoặc nhiều bảng có quan hệ hoặc không có quan hệ. Khi chúng ta sử dụng phương thức Load() để tải một đối tượng XMLDataDocument, chúng ta có thể xem nó như một cây hoặc như một bảng. Sau

đây là minh họa cho hai cách nhìn về một tài liệu XML khi sử dụng XMLDataDocument

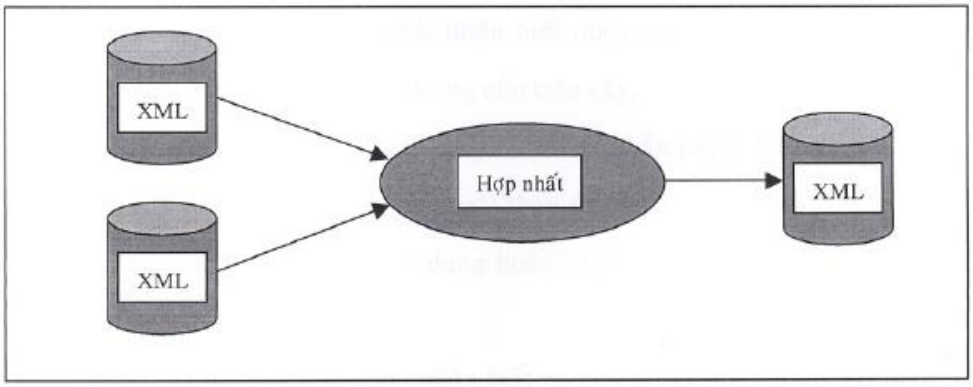


2.1.10 XSLT(eXtensible Stylesheet Language transformations)

XSLT là một ngôn ngữ dựa trên XML dùng để biến đổi các tài liệu XML. Tài liệu gốc thì không bị thay đổi mà thay vào đó một tài liệu XML mới được tạo ra dựa trên nội dung của tài liệu cũ. Tài liệu mới có thể là có định dạng XML hay một định dạng nào đó khác, như HTML hay bản tin thuần. XSLT thường dùng nhất trong việc chuyển đổi giữ liệu giữa các lược đồ XML hay để chuyển đổi dữ liệu XML thành các trang web hay tài liệu dạng PDF.

2.2 Các bản tin có cấu trúc XML

Bài toán đề nghị cách tiếp cận hợp nhất 3-way, lấy một tập tin làm cơ sở, với thông tin đầu vào là ba tập tin XML T_b, T_1, T_2 với T_b là tập tin cơ sở, T_1 và T_2 lần lượt là tập tin nhánh có thể chứa các thay đổi so với tập tin cơ sở T_b. Có hai bài toán riêng biệt nhưng có liên quan đến nhau: Hợp nhất 2-way và hợp nhất 3-way. Sự khác nhau giữa chúng phụ thuộc vào vấn đề có hai tập tin hợp nhất hay có một tập tin "cơ sở", mà từ đó các tập tin khác được dẫn suất ra. Hợp nhất 3-way có tiềm năng về một giải pháp chính xác hơn khi có tập tin cơ sở hiện hữu, nhưng nó khá phức tạp. Trong bài toán chúng ta tập trung vào cách hợp nhất 3-way



Hình 1.1. Cách hợp nhất 2-way

Với bài toán hợp nhất 2-way, các yêu cầu cơ bản có thể phát biểu một cách đơn giản theo cách không hình thức như sau: tài liệu được hợp nhất phải chứa mọi thứ từ cả hai tài liệu nguyên thủy, nhưng không được trùng lặp nếu có những phần chung. Vấn đề ở đây là định nghĩa phần chung nhau là gì và việc đảm bảo điều này được điều quản một cách đúng đắn. Mặc dù mọi người dùng có cùng ý tưởng rất rõ về các yêu cầu của mình nhưng việc định nghĩa phần chung nhau có thể dẫn đến các kịch bản sử dụng khác nhau.

Ta có thể tóm tắt các yêu cầu như sau:

a. Các đặc tính của một phần tử bất kỳ trong tập tin kết quả phải là hợp nhất của các đặc tính trong hai tập tin với các phần tử tương ứng với phần tử hợp nhất, nếu có đụng độ này phải được xử lí.

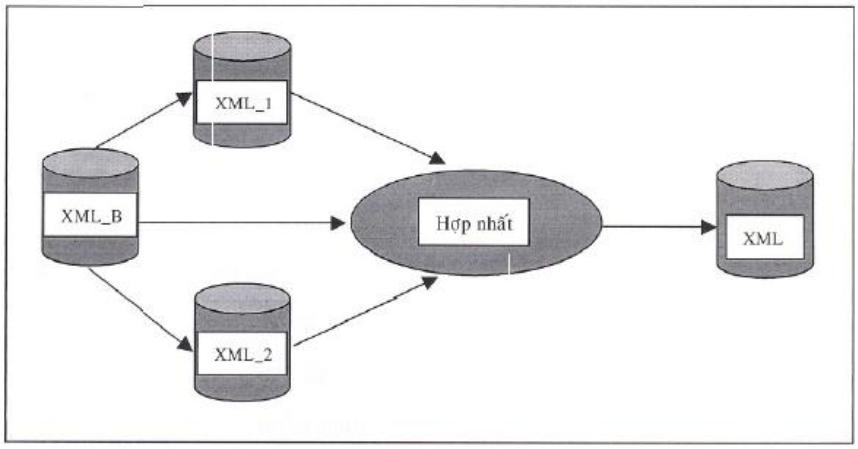
b. Các phần tử con của một phần tử cho trước bất kỳ(phần tử cha)phải được hợp nhất theo thứ tự của các phần tử con của các phần tử tương ứng với phần tử cha đó trong hai tập tin.

c. Thuật toán so sánh phải nhận biết được các phần tử tương ứng trong hai tập tin tại mỗi mức trong cấu trúc cây.

d. Quá trình hợp nhất phải xử lý được các phần tử có thứ tự.

e. Khi các node PCDATA(text)thay đổi, phải có sự lựa chọn giữa việc thực hiện hợp nhất nội dung hoặc chọn một trong các hình thức trình bày text.

f. Các đụng độ phải được nhận biết và xử lý ở các mức độ khác nhau.



Hình 1.2. Cách hợp nhất 3-way

Bên cạnh các yêu cầu giống như trường hợp hợp nhất 2-way, hợp nhất 3-way còn đòi hỏi:

g. Mọi thay đổi của một phần tử thuộc một trong hai nhánh phải được thể hiện trong tập tin hợp nhất.

h. Các phần tử bị xóa thuộc một trong hai nhánh không được xuất hiện trong tập tin hợp nhất.

Xét hai cây có thứ tự, liên quan với nhau T_1 và T_b, giả sử T_b thay đổi thành T_2 và ta muốn truyền các thay đổi này đến các thành phần của T_b hiện hữu trong T_1. Hiển nhiên T_1 và T_2 sẽ tham gia vào quá trình xử lý. Còn T_b tham gia vào để nhận biết các phần chung của T_1 và T_2. Tác vụ này được gọi là hợp nhất 3-way theo cấu trúc để nhấn mạnh rằng bản chất của việc hợp nhất là trên dữ liệu có cấu trúc (tức là trên các cây có thứ tự).

Hợp nhất 3-way theo cấu trúc: Giả sử T_1,T_2 là các cây có thứ tự được dẫn ra từ cây T_b. Việc hợp nhất 3-way các cây T_1,T_b và T_2 hình thành cây có thứ tự T_m,với T_m chứa các thay đổi giữa T_b và T_1 và các thay đổi giữa T_b và T_2. Cây T_b gọi là cây cơ sở và các cây T_1,T_2 gọi là các nhánh.

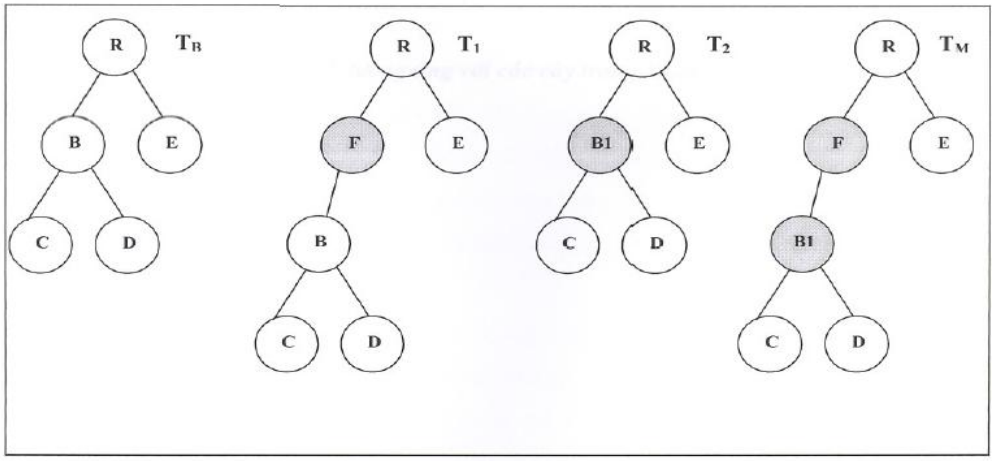
Nếu chúng ta phát sinh tập khác biệt T_b - T_1 và áp dụng cho T_2, chúng ta sẽ cần một phương cách nào đó để nhận biết các node trong T_2 mà các thao tác sửa đổi trong tập khác biệt sẽ được áp dụng. Điều này thật không dễ dàng, do cấu trúc của T_2 có thể khác hoàn toàn T_b và chúng ta không giả thiết về sự tồn tại của bất kỳ node nào. Mặt khác chúng ta không có tên cố định cho các phần của T_2, mà chúng ta muốn áp dụng các thay đổi đến các phần đó. Giải pháp cho vấn đề này là đưa thêm một số dữ liệu vào trong phần khác nhau như là ngữ cảnh cho các thao tác sửa đổi và do đó chúng ta cần phải định nghĩa một loại "cây ngữ cảnh" nào đó. Các giải

pháp hợp nhất khác nhau sẽ có những yêu cầu khác nhau nhưng có hai yêu cầu chính ảnh hưởng lên mọi giải pháp.

Trước tiên đó là các trường hợp hợp nhất và các kết quả mong đợi. Ví dụ việc hợp nhất dữ liệu với cấu trúc phân cấp chặt chẽ dẫn tới quá trình xử lí đơn giản việc hợp nhất các tài liệu mà thông tin trong đó có thể di chuyển trong cấu trúc cây XML.

Thứ hai là ảnh hưởng của kết quả của bài toán ánh xạ được lựa chọn để đưa ra tương ứng giữa hai tập tin: đây là vấn đề mấu chốt

Ví dụ về bài toán hợp nhất: Cây T_b là cây cơ sở, các cây T_1, T_2 lần lượt có các thay đổi như sau: T_1 chèn node F vào vị trí node B, T_b cập nhật node B thành B_1. Chúng ta muốn hợp nhất T_m phải thể hiện được các thay đổi trong các cây T_1 và T_2 so với T_b.



Hình 1 : Ví dụ một trường hợp hợp nhất cây

Các yêu cầu chức năng trong việc hợp nhất các bản tin có cấu trúc XML

Công cụ hợp nhất 3-way và tạo sự khác biệt –ráp cây để đồng bộ hóa phải thỏa mãn một số yêu cầu sau:

1 Thao tác hợp nhất phải dễ hiểu.

2 Mọi thay đổi trên các node trong các nhánh, như di chuyển, xóa, cập nhật, chèn, hay sao chép cần phải được xuất hiện trong cây hợp nhất. Các thao tác di chuyển và sao chép phải không bị hạn chế.

3 Các thao tác trên node phải được xem xét đến tính tương đối thay cho việc xem xét tuyệt đối. Ví dụ, nếu node con của một node n được sắp thứ tự lại

trong một nhánh và cây con gốc tại node n được di chuyển trong một nhánh khác, thì cả hai thao tác di chuyển của n và sắp thứ tự lại của các node con của node n phải được thể hiện trong cây hợp nhất. Như thế, các node con của node n đã được di chuyển một cách tương đối đối với n, không phải là di chuyển một cách tuyệt đối.

4 Bằng cách di chuyển, sao chép hay chèn một node, chúng ta đặt node đó trong một ngữ cảnh có các node bao quanh nó. Chúng ta muốn giữ cảnh đó sẽ được giữ nguyên trong cây hợp nhất.

5 Nếu một cây con được sao chép trong một nhánh, và được chỉnh sửa trong một nhánh khác, các chỉnh sửa phải truyền đến tất cả các sao chép của cây con đó trong phiên bản hợp nhất. Tuy nhiên, cũng có một phản ví dụ. Nếu cây con được sao chép là nhỏ, hay các bản sao chép chỉ là ánh xạ một cách gần đúng với nguyên gốc, các chỉnh sửa không nhất thiết phải được truyền đi.

6 Giả sử một node n tồn tại trong cả hai nhánh và các node mới được nối vào nhau như các node con đối với node n (chẳng hạn T_b=(R,a).T_1=(r;a b) và T_2=(R;a c))

Trong một số trường hợp chúng ta muốn danh sách con hợp nhất chứa các node được nối từ hai nhánh hoặc trong trường hợp khác điều này là một tính huống đụng độ.

7 Việc hợp nhất phải có tính tương đối.

8 Các đụng độ phải được phát hiện và xử lý

a- Cập nhật/Cập nhật. Đụng độ xảy ra nếu cùng một node được cập nhật ở cả hai nhánh.

b- Xóa/Di chuyển, Xóa/Cập nhật và Xóa/sao chép. Tình huống này xảy ra nếu một node bị xóa trong một nhánh và được "chỉnh sửa", tức là di chuyển hay cập nhật trong một nhánh khác

c- Di chuyển/Di chuyển. Node được di chuyển đến các vị trí khác nhau trong các nhánh.

9 Công cụ tạo khác biệt-ráp cây phải cho ra tập khác biệt cực tiểu để giảm băng thông khi truyền trên mạng

2.3 Cây và XML

2.3.1 Cây

Giả sử tập các node V và tập E ⊂ V×V các cạnh nối các node. Nếu (u, v) Є E ta nói rằng (u, v) là một cạnh và u là cha của v. T=(V,E) là cây gốc nếu tập các cạnh thỏa mãn các điều kiện sau:

1 Có đúng một node RЄ V không có cha. Node này gọi là gốc của cây T.

2 Mọi node, ngoại trừ node gốc, có đúng một node cha

3 Mỗi node trong V đều có thể được tham gia chiếu đến từ gốc. Ta nói rằng một cây là có thứ tự nếu các node con v1,…vk của nột node được đánh số duy nhất từ 1 đến k. Danh sách con của node u, kí hiệu u, là danh sách các node con của u theo thứ tự v1,v2,…vk. Ta kí hiệu u(i) là một thành phần trong v_i. Đối với cây có thứ tự, ta định nghĩa khái niệm node kề trái và node kề phải. Một cây không có thứ tự gọi là cây không thứ tự. Các node của cây đều được gán nhãn

Các cây được diễn tả thao các quy ước sau:

1 - Các node được gán nhãn R, a, b, c…, a1… an….Node gốc thường được gán nhãn R.

2- Cây bao gồm một node được ký hiệu bởi hàm nhãn node đó một cây T có nhiều hơn một node được kí hiệu bởi$(a,T_{a1},T_{a2}…T_{an})$với T_{an} là các cây con có gốc tại node an sao cho an là con của a.

Giả sử T_1=(R; a b) và T_2=(R; a c). Để phân biệt node a trong cây T_1 với node a trong cây T_2, ta kí hiệu $T_1(a)$, $T_2(a)$.

2.3.2 Ánh xạ cây

Ánh xạ: Một ánh xạ giữa hai cây T và T' là một tập các cạnh(n, m)n∈ T và m∈ T'. Tập ánh xạ được kí hiệu là M, với chỉ số tùy chọn nhằm nhận biết các cây liên quan.

Trước khi tiến hành hợp nhất chúng ta phải tìm được một ánh xạ giữa hai cây. Ta sử dụng ánh xạ để phát hiện các thay đổi làm biến đổi một cây thành một cây khác. Khi chúng ta nhận được các thay đổi giữa các cây chúng ta có thể thực hiện việc hợp nhất bằng cách tích hợp các thay đổi này vào trong một cây đơn.

Khi hợp nhất hai nhánh, ta kiểm tra từng node khi nó truyền đến cây hợp nhất. Tuy nhiên, có một số trường hợp ta không thể xác định các thay đổi đối với một node

Các kiểu ánh xạ:

Các node có kiểu ánh xạ nội dung: Hai node n ∈ T và m ∈ T' được gọi là có kiểu ánh xạ nội dung nếu cạnh (n, m) ∈ $M_{TT'}$, là kiểu ánh xạ nội dung

Các node có kiểu ánh xạ cấu trúc: Hai node n ∈ T và m ∈ T' được gọi là có kiểu ánh xạ cấu trúc nếu cạnh (n, m) ∈ $M_{TT'}$, là kiểu ánh xạ cấu trúc

Các node có kiểu ánh xạ đầy đủ: Hai node n ∈ T và m Є T' được gọi là có kiểu ánh xạ đầy đủ nếu và chỉ nếu chúng có kiểu ánh xạ cấu trúc và nội dung

Khái niệm các kiểu ánh xạ khác nhau cho phép ta xử lí các sao chép node theo cả hai nghĩa truyền dẫn thay đổi và không truyền dẫn thay đổi.

Các thay đổi đối với nội dung và cấu trúc truyền đi phải một cách tường minh. Nghĩa là mọi kiểu ánh xạ nội dung của một node cơ sở phải phản ánh cùng thay đổi trong nội dung và tất cả các kiểu ánh xạ cấu trúc phải phản ánh cùng thay đổi trong nội dung danh sách con.

Ánh xạ tự nhiên: Một tập ánh xạ M giữa hai cây T_b và T_1 được gọi là tự nhiên nếu và chỉ nếu:

1) Mỗi node m Є T_1 có 0 hay 1 node ánh xạ tương ứng trong T_b
2) Nếu và chỉ nếu n Є T_b có các node ánh xạ tương ứng trong T_1, thì có ít nhất một node trong số đó có kiểu ánh xạ đầy đủ.
3) Nếu và chỉ nếu n Є T_b có các node ánh xạ trong T_1 tất cả các node có kiểu ánh xạ cấu trúc có danh sách con đồng nhất.
4) Nếu và chỉ nếu n Є T_b có các node ánh xạ tương ứng trong T_1, tất cả các node có kiểu ánh xạ nội dung có nội dung đồng nhất.
5) Các cạnh (n, m)Є M được xác định kiểu. Các kiểu bao gồm kiểu ánh xạ nội dung, kiểu ánh xạ cấu trúc và kiểu ánh xạ đầy đủ.

2.3.3 Hợp nhất cây

Trong tất cả các định nghĩa T_1 và T_2 có thể tráo đổi cho nhau. Chúng ta giả sử rằng các tập ánh xạ M_{B1} và M_{B2} tồn tại và chúng là các tập ánh xạ tự nhiên

Node cộng sự: Hai node n Є T_1 và m Є T_2 là node cộng sự nếu và chỉ nếu tồn tại một node b Є T_B sao cho n và b là các node được ánh xạ ,và b và n và m cũng là các node được ánh xạ

Hai hình thức đặc biệt của node cộng sự là cộng sự nội dung và cộng sự cấu trúc. Nếu hai node là cộng sự nội dung, chúng thể hiện các thay thế về nội dung của node trong cây hợp nhất. Nếu các node là cộng sự cấu trúc, chúng thể hiện các thay thế về danh sách con của node đó trong cây hợp nhất.

Node cộng sự nội dung: Hai node n Є T_1 và m Є T_2 là các node cộng sự cấu trúc nếu và chỉ nếu tồn tại một node b Є T_B sao cho n và b là các node ánh xạ nội dung và b và m là các node ánh xạ nội dung.

Node cộng sự cấu trúc: Hai node n Є T_1 và m Є T_2 là các node cộng sự cấu trúc nếu và chỉ nếu tồn tại một node b Є T_B sao cho n và b là các node ánh xạ cấu trúc và b và m là các node ánh xạ cấu trúc.

Chúng ta tiếp tục bằng cách hình thức hóa khái niệm cập nhật, chèn, xóa, di chuyển, sao chép node.

Chèn node: Giả sử m Є T_1. Node m được gọi là được chèn nếu và chỉ nếu nó không có node nào ánh xạ trong T_B.

Xóa node: Giả sử b Є T_B. Node b gọi là bị xóa nếu và chỉ nếu nó không có node nào ánh xạ trong T_1

Node cập nhật: Giả sử b Є T_B và m Є T_1. Nếu m và b ánh xạ nội dung nhưng nội dung của chúng không giống nhau, hay m và b ánh xạ cấu trúc, m gọi là được cập nhật đối với T_B

Số sequence: Giả sử rằng n là node trong 1 cây bất kì. Mỗi node n_i trong danh sách con n= $n_{1...}n_k$ có một số sequence được kí hiệu là $S_n(x)$. Số sequence đó được định nghĩa như sau: $S_n(n_i) = I$ và $S_n(x) = -1$, $x \notin n$.

Cộng sự danh sách: Giả sử b ∈ T_B và m ∈ T_1. Nếu và chỉ nếu m(i) có một hay một số node ánh xạ y_i ∈ b, danh sách cộng sự của nó là các node ánh xạ với y_i, với số sequence nhỏ nhất. Ngược lại m(i) không có danh sách cộng sự.

In sequence: Giả sử b ∈ T_B và m ∈ T_1. m(k),k>1 được gọi là in sequence đối với b nếu và chỉ nếu m(k-1) có một danh sách cộng sự x trong b và m(k) có một danh sách cộng sự y trong b và S(x)< S(y) và với mọi i, $S_n(x)<i<S_n(y)$, thì b(i) bị xóa trong T_1.m(0) là in sequence đối với b nếu có một danh sách cộng sự x trong b và với mọi i, i<S(x) thì b(i) bị xóa trong T_1.

Node sao chép: Giả sử m ∈ T_1 và node ánh xạ của m trong T_B là b. Node m bị sao chép đối với T_B nếu nó bị di chuyển và b có nhiều hơn 1 node ánh xạ trong T_1. Nếu m bị sao chép và một số node ánh xạ hiện hữu trong danh sách con m_p, lúc đó node ánh xạ của m trong m_p với số sequence nhỏ nhất là node sao chép chính trong m_p và mọi node sao chép khác trong m_p gọi là các node sao chép phụ trong m_p. Nếu một node b ∈ T_B có các node ánh xạ mà các node này bị sao chép, ta nói rằng b bị sao chép.

Node hợp nhất: Các node trong T_M là các node hợp nhất. Mỗi node hợp nhất là kết quả của việc hợp nhất một node đơn hay hai node, trong trường hợp đó các node là cộng sự. Một node hợp nhất p là kết quả việc hợp nhất node n được gọi là node hợp nhất của n.

Node ngữ cảnh trái(phải): Giả sử ni ∈ n và T_B là cây cơ sở. Node ngữ cảnh trái (phải) của n_i là $n_k n_i(n_i n_1)$ với $n_k(n_1)$ là node đầu tiên ở ngay trước(ngay sau)n_i trong n mà không bị xóa đối với T_B. Nếu $n_k(n_i)$ không tồn tại, node n_i không có node ngữ cảnh trái (phải).

Node ngữ cảnh: Node ngữ cảnh của n_i là $n_k n_i n_1$, với $n_k n_1$ là node ngữ cảnh trái và $n_i n_1$ là node ngữ cảnh phải. Nếu một trong hai node ngữ cảnh trái hay phải của n_i không tồn tại, node n_i chỉ có ngữ cảnh phải hay trái. Trong các trường hợp này chúng ta nói rằng ngữ cảnh là trái hay phải.

2.3.4 Giải quyết bài toán hợp nhất cấu trúc để đồng bộ hóa

Giống nhau về nội dung- ngữ nghĩa:

Xác định độ giống nhau nội dung bằng Q-GRAM

Khoảng cách chuỗi q-gram giữa hai chuỗi được xác định như sau:

Cho Σ là bộ kí tự hữu hạn, Σ* là tập tất cả các chuỗi sinh ra từ Σ và Σq là tất cả các chuỗi có chiều dài q sinh ra từ Σ. Một q-gram là một chuỗi v = a1a2…aq Є Σq

Cho v là 1 q-gram và x = a1a2…an là một chuỗi trong Σ*. Nếu v = aiai+1…ai+q-1 ⊂ x với I nào đó, thì v xuất hiện trong x. Chúng ta kí hiệu số lần xuất hiện của v trong x là Gq(x)[v]. q-gram profile của x là vectơ Gq(x) = (G(x)[v]), v ϵ Σq

Khoảng cách q-gram giữa các chuỗi x và y là chuẩn L1 của Gq(x) – Gq(y): $Dq(x,y) = \Sigma_v \in \Sigma^{q|Gq(x)[v] - Gq(y)[v]|}$.

Để tránh ảnh hưởng các q-gram profile giống nhau giữa các bản tin không liên quan nhưng đủ dài, chúng ta tăng giá trị của q với chiều dài tổ hợp 1 của chỗi theo công thức:

$$q = \begin{cases} 1 & l < 50 \\ 2 & 50 \le l < 150 \\ 4 & l \ge 150 \end{cases}$$

Các hằng 50 và 150 là các giá trị gần đúng. Chúng ta kí hiệu hàm khoảng cách này là qdist(...).

Đo độ giống nhau về nội dung giữa các node được chuẩn hóa đến khoảng cách [0,1], với 1 nghĩa là khác nhau hoàn toàn. Chúng ta không sử dụng độ đo khoảng cách trực tiếp để đo độ giống nhau, do chúng ta không muốn các chuỗi ánh xạ ngắn được xem là các ánh xạ rất tốt. Điều này là do chúng ta sử dụng sự giống nhau như là độ đo chắc chắn mà một node thực sự ánh xạ với một node khác. Mặc dù khoảng cách là 0 giữa hai node mà nội dung của nó là cùng kí tự đơn thực ra không chuyển tải nhiều thông tin. Nếu nội dung của các node là hai câu dài đồng nhất, chúng ta hình thành công thức tính sự giống nhau của hai node từ trọng số trung bình của khoảng cách q- gram và một số hạng về mức độ vi phạm sao cho nội dung càng ngắn thì càng bị gán số hạng có trị số mức độ vi phạm cao

Chúng ta định nghĩa khoảng cách nội dung giữa hai node là:

$$\text{infoSize}(n) = \begin{cases} \text{Max}(|\text{ text}(n)|\ c_t, 1) & \text{(i)} \\ c_e + \Sigma_{\text{attr}(a,i)}\, c_a + \max(|\text{ val}(n,a)| - c_v, 1)) & \text{(ii)} \end{cases}$$

chú ý: (i) : n là text node

(ii): n là element node

$$\text{attrInfor}(a) = \min(\text{val}|n_1,a|- c_e,1)+ \min(\text{val}(n_2,a)-c_v,1)$$

$$\text{contentDist}(n_1,n_2) = \begin{cases} \min\left(\frac{InfoSize(n_1)+InfoSize(n_2)}{2}, qdist(text(n_1),text(n_2))\right) & n_1,n_2 : text\_node \\ sametag(n_1,n_2) + \sum_C \min\left(qdist(val(n_1,a),val(n_2,a)), attrInfo(a)\right) + c_a D & n_1,n_2 : elemenent\_node \\ 1 & khac \end{cases} \quad (5.4)$$

Với c_t, c_v là các hằng nhằm giảm sự đóng góp đối với sự giống nhau của bản tin ngắn và các giá trị thuộc tính.

C_a là nội dung thông tin trong tên thuộc tính

C_e là nội dung thông tin trong tên phần tử

C là tập các thuộc tính trong cả hai n_1 và n_2

D là số thuộc tính của n_1 và n_2 không hiện diện trong cả hai node

Hàm sametag (.,.) trả về c_e nếu các tên phần tử bằng nhau, ngược lại trả về 0. Các hàm infoSize (.,.) và attrInfo(.,.) trả về chiều dài đã xén của các giá trị thuộc tính và nội dung của một node. Mục đích của các hàm này làm giảm đóng góp của các bản tin ngắn và các giá trị thuộc tính đối với độ đo sự tương đồng.

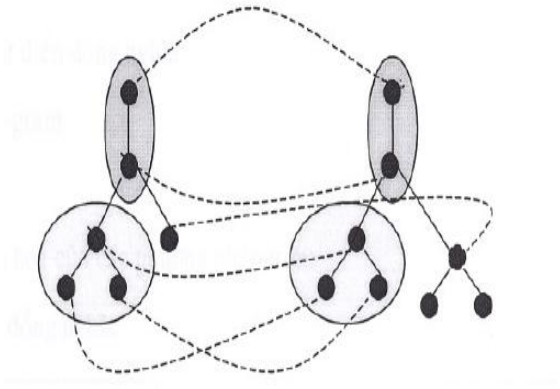
Khoảng cách tối đa giữa hai node bây giờ cho bởi công thức:

$$\text{maxDist}(n_1,n_2) = \begin{cases} \frac{InfoSize(n_1)+InfoSize(n_2)}{2} & n_1,n_2 : text\_node \\ c_t + \sum_C attrInfo(a) + c_a D & n_1,n_2 : elemenent\_node \\ 1 & khac \end{cases} \quad (5.5)$$

2.3.5 Giải thuật tìm kiếm ánh xạ giữa hai cây

Giải thuật tìm kiếm ánh xạ cây con

Chúng ta sẽ tìm kiếm các cây con lớn nhất có thể có của T_B và T_1 ánh xạ được với nhau bằng giải thuật heuristic greedy



Ánh xạ giữa hai cây

Ánh xạ giữa hai cây là tìm các cây con tương ứng nhau trong 2 cây. Sự tương ứng được định nghĩa như sau:

Ánh xạ chính xác giữa 2 cây con:

Nội dung của node gốc và các node con cũng như thứ tự các node con cũng phải giống nhau hoàn toàn. Nói chung có thể chồng khít 2 cây lên nhau được.

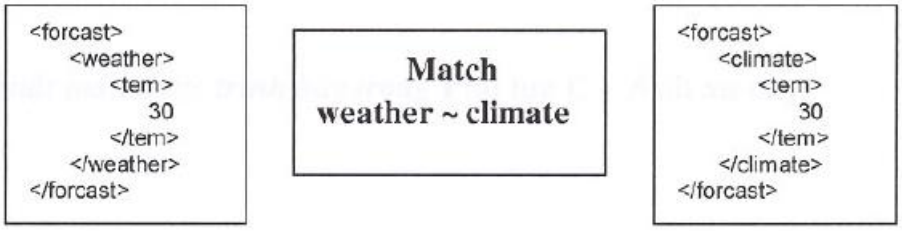
Ánh xạ tương đồng cấu trúc:

Nội dung nút gốc có thể khác nhau nhưng nội dung cũng tương tự của các node con phải giống nhau hoàn toàn. Vậy các chiến lược chấp nhận ánh xạ tương đồng sẽ phụ thuộc cách chập nhận độ sai lệch giữa 2 nút gốc với nhau. Chính tại yếu tố này chúng ta sẽ can thiệp vào bằng 2 cách đo độ sai lệch đó là 2 phương pháp sử dụng:

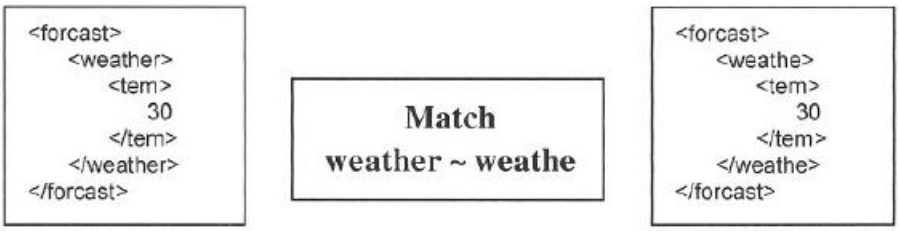
- Từ điển đồng nghĩa
- Q-gram

Sau đây là minh họa của các phương pháp trên

* Từ điển đồng nghĩa



* Q-gram



Để thực hiện việc ánh xạ cây con ta sử dụng giải thuật greendy:lấy một node trong T_1 và toàn bộ cây T_B làm thông tin vào và trả về một node trong T_B là gốc của cây con ánh xạ lớn nhất

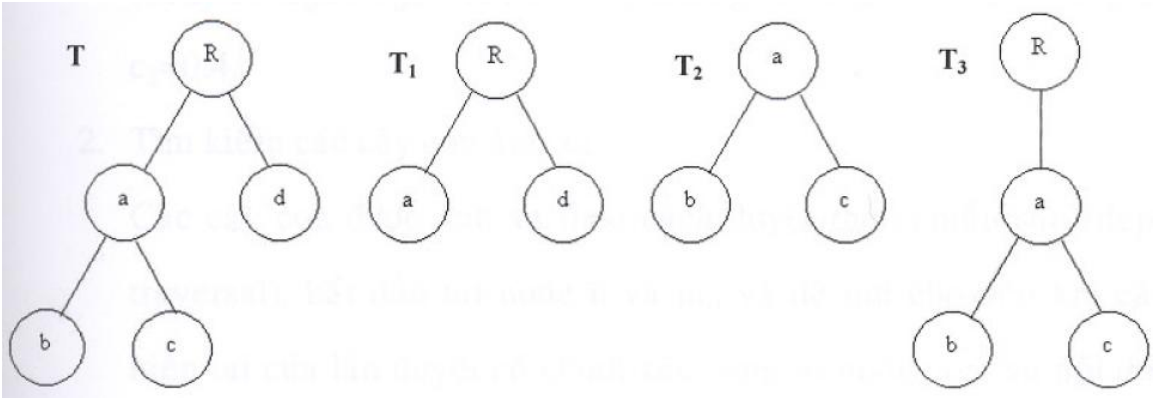
Toàn bộ các công đoạn của thuật toán được trình bày trong bảng

Procedure buildMatching(T_B,T_1: cây)
1. matchSubtrees(T_B,$T1_{(R)}$)
2. removeSmallCopies(T_B,T_1)
3. matchSimilarUnmatched(T_B,T_1)
4. setMatchTypes(T_B,T_1)

End Procedure

Ánh xạ cây con:

Trước hết chúng ta định nghĩa khái niệm cây con bị xén. Một cây con bị xén của cây T là một cây con của T có tất cả các hậu duệ của một số node trong cây con đó bị xóa đi.



Hình ảnh cây con bị xén

Cây con bị xén: Cho T là một cây, và T' là một cây có tất cả các node $n \in T'$: $n \in T$. Cho T_s là cây nhận được từ T' là một cây con bị xén của T nếu và chỉ nếu T_s là một cây con của T

Thủ tục matchSubtrees duyệt đệ quy cây T_1 và tìm các cây con bị xén trong T_B. Thường thì một loạt các cây con ánh xạ được tìm thấy và cây con có nhiều node nhất được sử dụng. Trong trường hợp có nhiều ánh xạ với cùng số tối đa node, ta sử dụng giải thuật heuristic để quyết định ánh xạ tốt nhất

Khi cây con ánh xạ tốt nhất được chọn, các node của cây con đó trong T_1 và T_B được ánh xạ với nhau. Các node của cây con trong T_1 cũng được thẻ hóa với một thẻ của cây con, đồng nhất một cách duy nhất cây con đó. Điều này được thực hiện để theo vết các cây con ánh xạ cần cho giai đoạn tiếp theo. Cuối cùng thủ tục đệ quy đối với mỗi node con của các node lá của cây con ánh xạ, tức là các node chưa được ánh xạ.

Thuật toán được trình bày chi tiết

1. Tìm kiếm các ánh xạ

Khi tìm kiếm các node ánh xạ đối với node n trong T_B, các node mà nội dung của chúng ánh xạ một cách chính xác được xem xét trước.Nếu không, chúng ta tiếp tục tìm các node đồng nghĩa, nếu vẫn không có node nào tìm thấy. Chúng ta trả về tất cả các node $m_i \in T_B$ sao cho nodeDistance (n, m_i) ≤ min (c_1* minDist, c_2), với minDist là khoảng cách node nhỏ nhất giữa một node trong T_B và node n, c_1 là một hằng dùng để chỉ độ sai lệch ánh xạ so với mong đợi và c_2 là hằng cutoff(xén) để ngăn ngừa các ánh xạ không rõ ràng. Thông thường $c_1 = 2$ và $c_2 = 0.4$

2. Tìm kiếm các cây con ánh xạ

Các cây con được ánh xạ theo cách duyệt theo chiều sâu (DFS) bắt đầu tại node n và m_i và đệ quy cho đến khi các node hiện tại của lần duyệt có chính xác cùng số node con và nội dung của các node con này ánh xạ một cách chính xác. Chúng ta gọi một cây bị xén như thế là cây con ánh xạ

3 Chọn cây con ánh xạ tốt nhất

Giai đoạn này bao gồm hai bước: Chọn cây con có số node tối đa nhận được kết quả của giải thuật greendy và nếu có nhiều cây con có cùng số node tối đa, chọn ra cây con ánh xạ tốt nhất trong số đó. Trong trường hợp tất cả các ánh xạ đều xấu, không có cây con nào được chọn là tốt nhất

4 Ánh xạ và đệ quy

Sau khi cây con bị xén ánh xạ tốt nhất được chọn ra, các node của cây con đó được ánh xạ với các node tương ứng trong T_1. Các node của cây con trong T_1 cũng được gán giá trị thẻ. Cuối cùng, thủ tục tiếp tục đệ quy

Procedure matchSubtrees(T_B:cây,n: node trong cây T_1)

1. (m_1,…,m_i):= các ánh xạ của n trong T_B
2. <T_1,…..T_i>: =các cây con có gốc tại m_i thuộc ánh xạ với cây con bị xén có gốc tại n
3. <T_{j1},…,T_{jk}>: =các cây con có số node tối đa
4. T:= ánh xạ tốt nhất của <T_{j1},…T_{jk}>
5. **If** ánh xạ tốt nhất T tồn tại **then**
6. ánh xạ các node trong T với các node tương ứng trong T_1
7. gán cây con trong T_1 thẻ cây con duy nhất
8. **End if**
9. **for** mỗi node lá 1 trong T **do**
10. **fo**r mỗi node con l_i của l **do**
11. matchSubtrees(T_B,l_i)

12. end for
13. end for
End Procedure

Thuật toán ánh xạ cây con (chọn ra cây con ánh xạ tốt nhất trong số các cây con ứng viên có độ lớn bằng nhau)

Độ đo khoảng cách ngữ cảnh n và m_i là:

ContextDistance (n, m_i) =

Min (nodeDistance (n_1, n_{ii}), nodeDistance (n_1, m_{ri}), nodeDistance (n.m_i))

Nếu chúng ta tìm một ứng viên m_i sao cho đối với ứng viên đó n_1 và m_{1i} ánh xạ với nhau, chúng ta sẽ chọn m_i. Nếu không có một ứng viên như vậy thì node có khoảng cách ngữ cảnh nhỏ nhất được sử dụng miễn là nó là node ánh xạ đủ tốt. Nghĩa là nếu kích thước của ứng viên tốt nhất chỉ là một node, khoảng cách ngữ cảnh cần phải nhỏ hơn một hằng số nào đó, ngược lại sẽ không có một ánh xạ tốt nhất nào được chọn.

2.3.6 Xử lý đụng độ

Đụng độ element node

Trường hợp này sẽ được xử lí bằng 3 option:

- Tự động chọn theo T_1
- Để người dùng tự chọn
- Chọn bằng cách đo khoảng cách ngữ nghĩa của tag name bằng cách sử dụng WordNet

Đụng độ text node

Trường hợp này sẽ được xử lí bằng 3 option

- Tự động chọn theo T_1
- Để người dùng tự chọn
- Cho phép chỉnh sửa nội dung của Text node

Để tăng tính hiệu quả cho công cụ, quá trình xử lí đụng độ cập nhật/cập nhật Text node sẽ được chia làm hai giai đoạn:

- Tính khác biệt giữa Text node thuộc T_B và T_1 và giữa T_B và T_2
- Chọn lựa chỉnh sửa

2.4 Chọn lựa mô hình

Để xử lí tài liệu XML ta có thể dùng 2 mô hình, nói chính xác hơn là 2 kĩ thuật đó là mô hình SAX và mô hình DOM

DOM (data Object Model) là kĩ thuật cho ta khả năng thao tác tài liệu XML như thao tác trên một cấu trúc cây. Tuy nhiên thông tin ở mỗi node của cấu trúc cây đó đã được định nghĩa sẵn, chính điều này đã hạn chế việc sử dụng DOM trong việc xử lí các nút có nhiều thông tin.

SAX (Simple API for XML) là kĩ thuật phức tạp hơn DOM. SAX dựa trên mô hình xử lý theo sự kiện. SAX sẽ đọc vào file XML và phát sinh ra những sự kiện khi gặp các loại phần tử của tài liệu XML. Dựa vào những sự kiện này chúng ta sẽ tái tạo mô hình cây của tài liệu XML với thông tin mỗi nút (node) là do chúng ta

quyết định. Vậy SAX trao quyền định nghĩa mô hình cây của tài liệu cho người xử dụng và cũng chính vì điều này làm cho mô hình xử lí SAX linh động hơn DOM. Thực chất mô hình DOM cũng phải dùng mô hình xử lí SAX trước ,nhưng bước này là "trong suốt" với người dùng.

Và do đó chúng ta sẽ xử dụng mô hình SAX để thực hiện chương trình hợp nhất cấu trúc

2.5 Các thuật toán ứng dụng trong hợp nhất bản tin

Trong việc hợp nhất bản tin có cấu trúc việc xác định các từ, ngữ nghĩa của một tập tin được dựa vào các từ điển Tiếng Việt

2.5.1 Từ điển đồng nghĩa Tiếng Việt

Cũng như Tiếng anh, Tiếng Việt có rất nhiều từ đồng nghĩa và trái nghĩa với nhau. Các từ đồng nghĩa và phản nghĩa có thể được tìm thấy rất nhiều trong các bản tin tiếng việt và trong ngôn ngữ giao tiếp hằng ngày. Việc xử lí đụng độ trong quá trình tổng hợp các nguồn thông tin bản tin Tiếng Việt yêu cầu phải nhận biết được các cặp từ đồng nghĩa và trái nghĩa với nhau. Do đó, việc xây dựng từ điển đồng nghĩa- trái nghĩa Tiếng Việt là rất cần thiết.

2.5.2 Nguồn dữ liệu

Nguồn dữ liệu là một tập dạng Text gồm nhiều dòng tổ chức theo nhiều mục. Cấu trúc của một mục dữ liệu trong tập tin này được mô tả như sau:

W

+ {Danh sách các từ đồng nghĩa của W cách nhau bởi dấu phẩy} (ký hiệu S)

++ {Danh sách các từ trái nghĩa của W cách nhau bởi dấu phẩy} (ký hiệu A)

**** Nhận xét:**

Theo quy luật phát sinh quan hệ, các từ trong tập S đều là các từ đồng nghĩa với nhau. Tương tự, các từ trong tập A đều là các từ đồng nghĩa với nhau. Tương tự, các từ trong tập A đều là các từ đồng nghĩa với nhau.

2.5.3 Chuyển đổi từ điển đồng nghĩa – trái nghĩa Tiếng Việt sang dạng thích hợp

Việc tra cứu thông tin trực tiếp vào nguồn dữ liệu ban đầu rất chậm vì cấu trúc dữ liệu của nguồn dữ liệu chỉ là dạng text đơn giản. Để tăng tốc độ truy vấn thông tin, chúng ta phải mã hóa các từ thành các con số.

Các từ đồng nghĩa với nhau thì có mã số chênh lệch nhau 1 đơn vị.

Mối quan hệ giữa 2 từ như sau:

Synonym (W_1, W_2) <=> (ID (W_1) = ID (W_2))

Synonym(W_1, $W_{2)}$ <=> ((ID(W_1) lẽ $\wedge$ ID(W_2)= ID(W_1)+ 1) $\vee$ (ID(W_2) lẽ ID(W_1) = ID(W_2)+1++

Với ID (W_1), ID (W_2) lần lượt là mã số của từ W_1 và W_2.

2.5.4 Thuật toán xây dựng từ điển đồng nghĩa – trái nghĩa Tiếng Việt

In put: dữ liệu vào

Output:

+Bước 1: ID =1

Cây hậu tố T_s rỗng

+ Bước 2: Lặp

Xét từng mục trong tập tin nguồn dữ liệu. Với mỗi mục:

W

+ S

++ A

-$\forall w \in S \cup \{W\}$ thực hiện: thêm cặp(w, ID)vào cây hậu tố.

- $\forall w \in A$ thực hiện: thêm cặp (w,ID + 1) vào cây hậu tố.

ID = ID +2

2.5.5 Thuật toán xác định quan hệ giữa 2 từ Tiếng Việt:

Input: Cho 2 từ Tiếng Việt $word_1$ và $word_2$.

Output: Mối quan hệ giữa chúng được xác định qua các bước sau:

+ Bước 1:

Xác định mã số của $word_1$, $word_2$ thông qua cây hậu tố T_s.

$$\text{Word}_1 \xrightarrow{T_s} ID_1$$

$$word_2 \xrightarrow{T_s} ID_2$$

+ Bước 2:

Nếu $ID_1 = ID_2$ thì $word_1$ và $word_2$ là hai từ đồng nghĩa với nhau và dừng thuật toán. Ngược lại xuống bước 3 .

+ Bước 3:

- Nếu ID_1 lẽ và $ID_2 = ID_1 + 1$ thì $word_1$ và $word_2$ là 2 từ trái nghĩa với nhau và dừng thuật toán

- Nếu ID_2 lẽ và $ID_1 = ID_2+1$ thì $word_1$ và $word_2$ là 2 từ trái nghĩa với nhau và dừng thuật toán.

- Thông báo $word_1$ và $word_2$ không có quan hệ với nhau. Thuật toán kết thúc

* Chú ý:

Do sự phức tạp và đa dạng của các Font chữ Tiếng Việt nên trong quá trình xây dựng và truy vấn từ điển, các từ Tiếng Việt nguyên thủy ban đầu sẽ được chuyển sang dạng mã ngõ để việc lưu trữ và truy vấn dữ liệu không bị sai sót.

Cách chuyển mã căn cứ vào bảng sau:

Sắc	Huyền	Hỏi	Ngã	Nặng	â	,	ă	đ
1	2	3	4	5	6	7	8	9

2.5.6 Ánh xạ cây

Giải thuật tìm kiếm ánh xạ cây con

Xóa các sao chép nhỏ: Giai đoạn tinh chỉnh đầu tiên là xác định một ngưỡng trên số lượng dữ liệu nhân bản được xem xét là kết quả của các thao tác sao chép, chứ không phải là kết quả của thao tác chèn.

Procedure builMatching (T_B,T_1: cây)
1. matchSubtrees(T_B,$T1_{(R)}$)
2. removeSmallCopies(T_B,T_1)
3. matchSimilarUnmatched(T_B,T_1)
4. setMatchType(T_B,T_1)

End Procedure

Các công đoạn ánh xạ giữa hai cây

Thủ tục removeSmallCopies kiểm tra tất cả các nút trong T_1 có nút ánh xạ cơ sở của nó có nhiều nút ánh xạ trong T_1. Nếu cây con bản sao của T_B có nội dung thông tin nhỏ hơn một ngưỡng cho trước, ánh xạ đó sẽ bị xóa và như thế nút được xem là nút được chèn vào. Nội dung thông tin của một cây con là độ đo lượng thông tin chứa đựng trong một cây con.

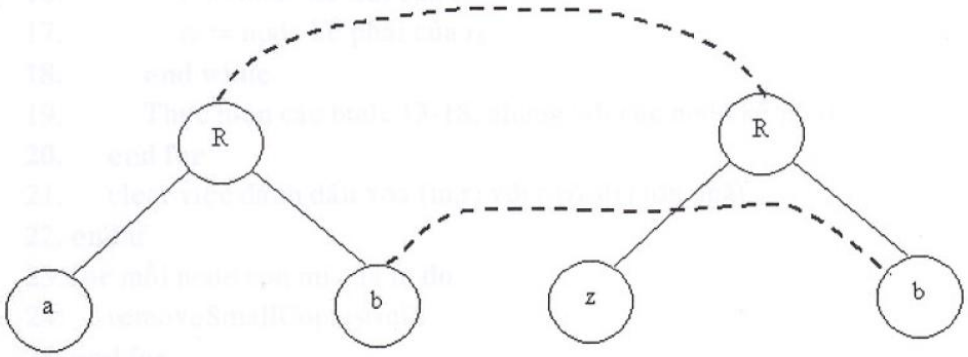
Chúng ta đã xử dụng công thức heuristic sau đây để tính nội dung thông tin cho một cây con:

$$\text{treeInfoContent(T)} = \sum_{n_i \in T} InfoSize\ (n_i) + c_{ed} * numEdges\ (T) \qquad \text{(C.1)}$$

Hằng số c_{ed} là nội dung thông tin trên một cạnh. Ở đây chúng ta đã xử dụng giá trị c_{ed}=4.

Sau khi thực hiện thủ tục matchSubtrees tất cả các nút ánh xạ trong T_1 được thẻ hóa với thẻ cây con để đồng nhất cây con ánh xạ mà chúng là một phần của cây con đó. Việc thẻ hóa này cho thấy rất thuận lợi khi thực hiện thủ tục removeSmallCopies .

Đôi khi, các cây con của tất cả các ánh xạ đều nhỏ hơn ngưỡng sao chép. Trong trường hợp này chúng ta không muốn xóa tất cả các ánh xạ, giải pháp tốt hơn là cố gắng nhận biết ánh xạ tốt nhất. Chúng ta thực hiện điều này bằng cách xem xét các nút láng giềng của nút gốc của câu con ánh xạ: nếu các nút bên trái và bên phải ánh xạ với các nút ở bên trái và bên phải trong cây cơ sở, chúng ta thêm các nội dung thông tin của các cây ánh xạ vào các nút đó của ứng viên để xóa nút ánh xạ, ứng viên có nhiều byte nhất sẽ không được đánh dấu không ánh xạ



***Hình C.3** : một ánh xạ có thể được mở rộng. Các node a và z không được ánh xạ, mặc dù chúng có thể được ánh xạ. Thủ tục matchSimilarUnmatched sẽ giải quyết trường hợp này.*

Giá trị của hằng số ngưỡng sao chép $c_{thresold}$ được xử dụng ngầm định là 128 byte

```
Procedure removeSmallCopies(m: node trong T_B)
1.  If m có nhiều ánh xạ trong T_1 then
2.      N := các ánh xạ của m trong T_1
3.      for mỗi n_i ∈ N do
4.          if nội dung thông tin của cây con mà n_i được gán < c_threshold then
5.              đánh dấu ánh xạ (m,n_i) để xoá
6.          end if
7.      end for
8.  end if
9.  if tất cả các ánh xạ của m được đánh dấu xóa then
10.     R := các gốc của các cây con mà các node n_i được gán
11.     for mỗi r ∈ R do
12.         s[r] := 0
13.         r_b:= ánh xạ cơ sở của r
14.         while r và r_b được ánh xạ và s[r] ≤ c_threshold do
15.             thêm treeInfoSize của r vào s[r]
16.             r  := node kề trái của r
17.             r_b := node kề phải của r_b
18.         end while
19.         Thực hiện các bước 13-18, nhưng với các node kề phải
20.     end for
21.     clear việc đánh dấu xóa (m,r) với r có s[r] lớn nhất
22. end if
23. for mỗi node con mi của m do
24.     removeSmallCopies(mi)
25. end for
End Procedure
```

Bảng C.2. Thuật toán xoá các sao chép nhỏ.

2.5.7 Thuật toán hợp nhất 3- way theo cấu trúc

*** Hợp nhất 3 – way theo cấu trúc**

Chúng ta sử dụng các ánh xạ M_{B1} và M_{B2} được sinh ra trong giai đoạn ánh xạ cây, để sinh ra cây hợp nhất.Do không phát sinh kịch bản chỉnh sửa, chúng ta sẽ không bị hạn chế trên các thao tác của kịch bản chỉnh sửa. Hạn chế duy nhất áp đặt là các ánh xạ tự nhiên và không cho phép các thao tác "uncopy". Ý chính của thuật toán này là duyệt các cây T_1 và T_2 đồng thời để các node *partner* được thăm cùng một lúc. Mỗi bước duyệt sẽ xuất ra một node đến T_M. Node được xuất là sự hợp nhất các node trong T_1 và T_2 đang được thăm

Tiến trình hợp nhất gồm 3 giai đoạn:

- Giai đoạn tạo danh sách hợp nhất của mỗi cây con dựa vào cây T_B. Giai đoạn này các thay đổi đặc trưng của cây con so với cây T_B sẽ được đánh dấu bằng 2 thao tác là treo những node được thêm mới và khóa những node bị di chuyển
- Giai đoạn tạo thành các cặp hợp nhất. Input của giai đoạn này chính là 2 danh sách hợp nhất mà ta có được từ giai đoạn 1 của 2 cây con T_1 và T_2. Output là danh sách các cặp node sẽ được thực hiện hợp nhất ở giai đoạn sau. Mọi thay đổi đặc trưng của 2 cây con mà ta xác định ở giai đoạn 1 sẽ được bảo lưu trong giai đoạn này
- Giai đoạn hợp nhất các cặp lấy từ danh sách có được ở giai đoạn 2, ta sẽ thực hiện hợp nhất từng cặp trong danh sách này, kết quả sẽ là một node mới trong cây hợp nhất

Để hỗ trợ việc trình bày thuật toán chúng ta giới thiệu khái niệm con trỏ cây (tree cursos). Con trỏ cây chỉ ra vị trí hiện tại trong cây theo cách tương tự như con trỏ trong trình xử lí bản tin chỉ thị vị trí hiện tại của bản tin. Con trỏ cây cũng có thể được định vị tại một vị trí đặc biệt là NULL, khi con trỏ không hoạt động. Chúng ta sẽ kí hiệu các con trỏ là C_n với n là tên con trỏ và kí hiệu tham chiếu đến node mà con trỏ đang trỏ đến là node(C_n). Nút NULL được kí hiệu là 0. Quy ước

việc con trỏ đang trỏ tại một node m là C_n = m. Cho C_1 và C_2 và C_M lần lượt là con trỏ của các cây T_1 và T_2, T_M.

Procedure treemerge

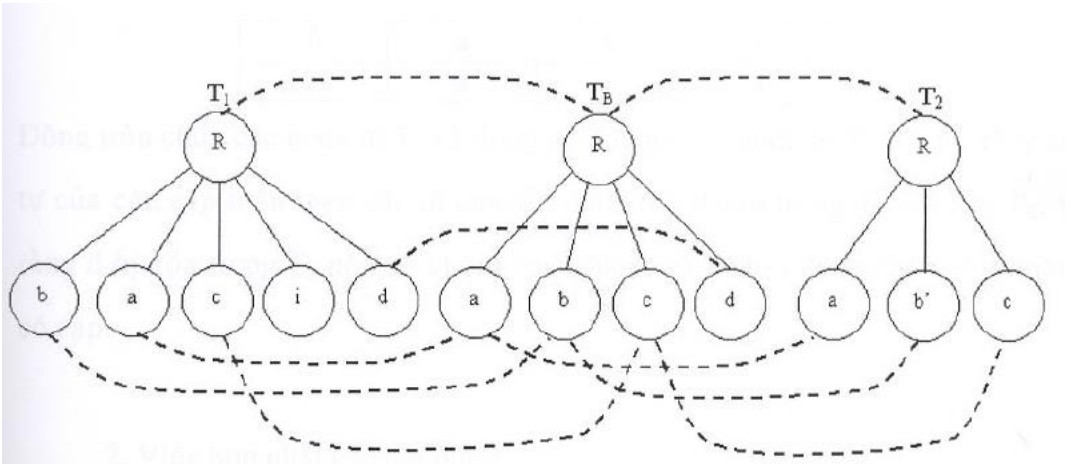
1. phát sinh danh sách cặp hợp nhất của các node con của node(C_1) và node(C_2)
2. p:= node(C_M)
3. **for** với mỗi cặp := {u_i,v_j} trong danh sách cặp hợp nhất **do**
4. hợp nhất các nội dung của u_i và v_j thành một node mới w
5. thêm w như một node con của p và định vị lại C_M tại w
6. định vị lại C_1 đến u và C_2 đến v
7. call treemerge
8. **end for**

End Procedure

Thuật toán hợp nhất

Các node *partner* được sắp thành cặp để hợp nhất bởi thuật toán sẽ được tham chiếu như cặp hợp nhất. Các cặp hợp nhất được kí hiệu bởi {n, m}, với n m là các node trong cặp. Các cặp hợp nhất cũng có thể chỉ chứa một node, trong trường hợp đó ta kí hiệu là {n, .}. Danh sách cặp hợp nhất là danh sách các cặp hợp nhất được phát sinh bằng cách tổ hợp các danh sách con của các node được trỏ tới bởi C_1 và C_2.

Ví dụ hợp nhất cây đơn giản.



Hình 5.5 : Trường hợp hợp nhất đơn giản

Chúng ta giả sử T_1 và T_2,T_B là các cây trong hình trên,chúng được ánh xạ theo hình vẽ, và tất cả các con trỏ được định vị tại gốc của mỗi cây.

1 Phát sinh danh sách cặp hợp nhất

Trong bước này chúng ta sắp cặp các con của node (C_1) và node (C_2) theo các ánh xạ đó, sau đó chúng ta xác định dãy các cặp này, theo các di chuyển bất kì được hình thành trong các cây. Các node bị xóa sẽ được loại bỏ khỏi danh sách. Trong trường hợp này chúng ta sắp cặp các danh sách con b a c i d (của T_1(R)) và a b' c (của T_2(R)). Danh sách cặp kết quả là:

b	a	c	i
b'	a	c	•

Dòng trên chứa các node từ T_1 và dòng dưới chứa các node từ T_2. Chú ý rằng thứ tự của các cặp tuân theo các di chuyển được tạo thành trong T_1 đối với T_B, và d bị xóa trong T_2 nên sẽ không xuất hiện và node i được chèn vào không có cặp

2. Việc hợp nhất các nội dung

Bây giờ chúng ta xác định việc hợp nhất nội dung của cặp $\{u_i,v_i\}$phải thực hiện như thế nào. Nói chung, chúng ta luôn lấy nội dung mà thể hiện sự thay đổi đối với T_B. Trong trường hợp này việc hợp nhất nội dung sẽ là b' a c i.

3. Thêm w

Chúng ta đã hợp nhất thành công các node u_i và v_j thành node w. Node hợp nhất được thêm vào cây hợp nhất và vị trí con trỏ được cập nhật để nối các node con đến node mới bằng cách thêm b' a c i cây hợp nhất của chúng ta bây giờ là:

T_M = (R; b' a c i)

4. Định vị lại C_1 và C_2

Các con trỏ của T_1 và T_2 được cập nhật để trỏ đến các node trong cặp hợp nhất mà nội dung hợp nhất của chúng đã được thêm vào T_M. Thông thường các con trỏ được trỏ trực tiếp đến u_i và v_j không phải là các *partner* cấu trúc. Con trỏ cập nhật đối với các cặp hợp nhất trong cùng trường hợp là:

{b,b'} => $C_1 = T_1(b) \wedge C_2 = T_2(b')$

{a ,a} => $C_1 = T_1(a) \wedge C_2 = T_2(a)$

{c ,c } => $C_1 = T_1(c) \wedge C_2 = T_2(c)$

{i , •} => $C_1 = T_1(i) \wedge C_2 = 0$

5 Gọi thực hiện treemerge

Đây là bước đệ quy của thuật toán. Chú ý rằng chúng ta đã thêm một node hợp nhất của các node con của các node được trỏ đến bởi các giá trị nguyên thủy của C_1 và C_2 đến T_M. Các con trỏ C_1, C_2 đã được reset để trỏ đến một tập các *partner* mới và C_m được cập nhật đến vị trí "chèn " mới trong T_M.

```
Procedure makeMergeList(n: Node)
1.  M := danh sách hợp nhất rỗng
2.  append α vào M
3.  đặt b = ánh xạ của n trong T_B
4.  đặt b = danh sách con của b
5.  for với mỗi node con n_i của n do
6.     if n_i được chèn vào then
7.        thêm n_i như một node bị treo đối với entry cuối cùng trong M
8.        đánh dấu entry cuối cùng là bị khoá trái
9.     else if n_i là node far move then
10.       thêm n_i như node bị treo đối với entry cuối cùng trong M
11.       đánh dấu entry cuối cùng là bị khóa trái
12.    else if n_i là bản sao thứ cấp của a then
13.       thêm n_i như node bị treo đối với entry cuối cùng trong M
14.       khóa bản sao chính của a ở bên trái và phải
15.    else
16.       append vào M một entry mới e mà node của nó là n_i
17.       if n_i không in sequence với b then
18.          khóa e ở bên trái
19.       end if
20.    end if
21. end for
22. đặt n_i là node của entry cuối cùng trong M
23. append entry kết thúc w vào M
24. if ánh xạ của n_i không phải là node cuối cùng trong b then
25.    khóa w ở bên trái
26. end if
end procedure
```

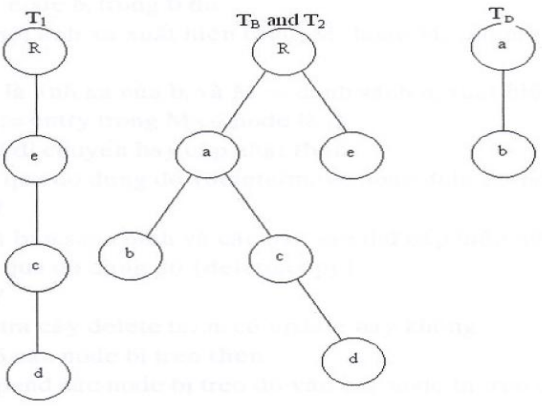
Bảng D.2. Phát sinh danh sách hợp nhất

2.5.8 Kiểm tra các node bị xoá và di chuyển xa:

Cho các danh sách hợp nhất được phát sinh từ các node con của u và v là M_1 và M_2. Node cơ sở chung của u và v là node b. Chúng ta bắt đầu tổ hợp các danh sách hợp nhất bằng cách kiểm tra các node xuất hiện trong b nhưng không có trong M_1 hoặc M_2 ,tức là các node mà đã bị xóa hoặc di chuyển xa khỏi b trong cả hai nhánh. Nói tóm lại, một node bất kì $b_i \in b$ không xuất hiện như là một node của một entry trong M_1 mà node đó có một entry ánh xạ được trong M_2 có một entry của nó bị loại bỏ khỏi M_2 và ngược lại.

Tuy nhiên, chúng ta không thể chỉ đơn giản loại bỏ các entry từ cả hai danh sách hợp nhất mà không cân nhắc. Chúng ta cần kiểm tra rằng không có các chỉnh xửa trong một cây của các node bị xóa có gốc tại entry bị xóa, trong trường hợp đó các chỉnh xửa này sẽ được bỏ qua. Thêm nữa, một entry có thể không bị xóa nếu nó được cập nhật, di chuyển hoặc là bản sao chép chính khi các bản sao chép thứ cấp tồn tại sẽ vi phạm các yêu cầu 1, 4, 5. Nếu một *entry* với các node bị treo được loại bỏ, các node bị treo phải được di chuyển đến *entry* ở trước .

Việc cây delete T_D của node bị xóa n chứa n và tất cả các hậu duệ bị xóa của n, mà không có bất kì node nào không bị xóa ở giữa. Chúng ta định nghĩa cây delete của n là cây con T_n mà từ đó các cây con $T_{n1}…T_{nk}$ bị loại bỏ, với các node $n_i \in T_n$ được ánh xạ đầy đủ với node cơ sở và có một *partner*, tức là các node mà được đảm bảo sẽ được viếng tại một giai đoạn khác của thuật toán.



Hình D.2. Cây delete T_D. Cây delete của $T_2(a)$ có kiểm tra sự cập nhật khi hợp nhất các danh sách con của $T_1(R)$ và $T_2(R)$ là cây T_D.

Bây giờ T_D là tập các node sẽ không xuất hiện trong cây hợp nhất, do việc loại bỏ node n khỏi danh sách hợp nhất. Chúng ta cần kiểm tra rằng không có node nào trong các node này thể hiện một thao tác chỉnh xửa và nếu một node này là thể hiện một thao tác chỉnh xửa, chúng ta có thể hoặc bỏ qua hoặc đưa ra một cảnh báo về việc cập nhật có khả năng mất. Việc kiểm tra các node chỉnh xửa được thực hiện theo các định nghĩa thao tác chỉnh xửa trong cây hợp nhất.

```
Procedure removeDeletedOrMoved(M1, M2, b)
    1.  for với mỗi node bi trong b do
    2.      if bi có một ánh xạ xuất hiện trong M1 hoặc M2 (nhưng không trong cả
        hai) then
    3.          đặt ni là ánh xạ của bi và M := danh sách ni xuất hiện bên trong
    4.          đặt e := entry trong M có node là ni
    5.          if ni bị di chuyển hay cập nhật then
    6.              bỏ qua do đụng độ {delete/move hoặc delete/update}
    7.          end if
    8.          if ni là bản sao chính và các bản sao thứ cấp hiện hữu trong M then
    9.              bỏ qua do đụng độ {delete/copy}
    10.         end if
    11.         kiểm tra cây delete tại ni có update hay không
    12.         if e có các node bị treo then
    13.             append các node bị treo đó vào các node bị treo của entry ở trước
        e trong M
    14.         end if
    15.         if e bị khóa then
    16.             khóa entry ở trước e
    17.         end if
    18.         xóa e khỏi M
    19.     end if
    20. end for
end procedure
```

Bảng D.3. Thủ tục Xóa hay di chuyển Nút

Ví dụ trên kết quả của việc thực hiện thủ tục removeDeletedOrMoved trên các danh sách như sau:

$M_1^D =$ α b c e f g h w (D.5)

$M_2^D =$ α b' c e h f g w (D.6)

Với M_1^D và M^D_2 là các danh sách hợp nhất M_1 và M_2 sau khi thủ tục removeDeleteOrMoved được thực hiện

Như ta có thể thấy node d đã bị loại bỏ khỏi M_1, do nó đã bị xóa trong T_2 và node a đã bị loại bỏ khỏi M_2,do di chuyển xa trong T_1. Chú ý rằng kết quả này có đúng các node cùng *entry* trong cả hai danh sách : b ,c,e,h,f,và g

2.5.9 Tổ hợp các danh sách hợp nhất thành danh sách hợp nhất

Trong bước phát sinh danh sách hợp nhất cuối cùng, chúng ta tổ hợp các danh sách hợp nhất M_1^D và M_1^D thành một danh sách cặp hợp nhất tuân theo việc sắp thành dãy các node bị treo và các *entry* bị khóa. Bây giờ chúng ta có thể thấy lợi ích của các node bị treo và giai đoạn xóa trước: các *entry* trong M_1^D và M_2^D có tương ứng 1-1 tức là node của mỗi *entry* trong M_1^D là *partner* của đúng một node *entry* trong M_2^D và ngược lại. Sự tương ứng 1-1 này làm đơn giản hóa đáng kể việc hợp nhất các danh sách.

Chúng ta xử dụng việc duyệt đồng thời các danh sách hợp nhất, tương ứng việc duyệt node đồng thời được mô tả ở trên. Cho p_1 và p_2 là vị trí hiện tại trong M_1^D và M_2^D, cả hai được khởi đầu tại vị trí đầu tiên của danh sách. Chúng ta xuất ra các node p_1 và p_2 trỏ đến các node bị treo cũng như các cặp hợp nhất. Vị trí của p_1 và p_2 lúc đó được cập nhật để chúng ta luôn đi theo sau một khóa phải nếu tồn tại. Điều này được lập lại cho đến khi đạt đến cuối danh sách hợp nhất

Trong ví dụ chúng ta thực hiện thủ tục makeMergePairList trên các danh sách hợp nhất M_1^D và M_2^D được phát sinh trong giai đoạn trước. Kết quả danh sách cặp hợp nhất là:

$$M = \begin{bmatrix} b & c & e & h & f & i & b & g & b_1 \\ b' & c & e & h & f & \bullet & b' & g & b_1 \end{bmatrix} \quad \text{(D.7)}$$

Với các node trong hàng trên từ T_1 và các node trong hàng dưới từ T_2(đây cũng là vấn đề định dạng danh sách,thuật toán thêm các cặp {n,m} sao cho $n \in T_1$ và $m \in T_2$ hoặc $n \in T_2$ và $m \in T_1$.

Chúng ta chú ý rằng thứ tự của các cặp trong danh sách hợp nhất thỏa các khóa trong M_1^D và M_2^D,cũng như treo các node không khóa trong thứ tự nguyên thủy của nó. Thêm nữa, node i được chèn trong T_1 không có cặp hợp nhất và cả hai bản sao chép của node b trong T_1 được sắp với phiên bản cập nhất của node trong T_2

Mặc dù rõ ràng thứ tự của các cặp tuân theo thứ tự ngầm định bởi các *entry* bị khóa trong các danh sách hợp nhất, một số thuộc tính, chẳng hạn tính dừng của vòng lặp là không rõ ràng.

CHƯƠNG 3: ĐÁNH GIÁ THỰC NGHIỆM VÀ KẾT LUẬN

3.1 Giới thiệu về phần mềm Tree Way Merge

Merge là các tập tin hình ảnh, kết hợp ứng dụng và thư mục đồng bộ hóa từ Araxis. Xử dụng nó để so sánh và hợp nhất các mã nguồn, các trang web, XML và các tập tin dạng bản tin với các ứng dụng hiệu quả. Trực tiếp mở và so sánh các bản tin từ Microsoft Ofice, OpenDocument, PDF và các tập tin RTF. Hierarchies làm việc với các thư mục chứa hàng ngàn tập tin. Merge tích hợp nhiều SCM và các hệ thống khác

Ưu điểm:

- Đối với các chuyên gia pháp lí và xuất bản: Xác định ngay tất cả các thay đổi khác nhau giữa các hợp đồng hoặc bản thảo. Trực tiếp mở và so sánh các bản tin từ Microsoft Office, OpenDocument, PDF…. Sao chép bản tin từ các ứng dụng khác và dán nó trực tiếp vào một cửa sổ so sánh bản tin.

- Đối với kỹ sư phần mềm và các nhà phát triển web: So sánh và hiểu kết hợp các tập tin mã nguồn phiên bản khác nhau. Làm việc một cách nhanh chóng và chính xác, cho dù bạn đang so sánh các tập tin cá nhân hoặc chi nhánh của reconciling toàn bộ mã nguồn.

- Những người dùng khác: Cần phải giữ nhiều thư mục trong đồng bộ. Hợp nhất giúp tiết kiệm và giảm các lỗi bằng cách giúp bạn làm việc một cách nhanh chóng và chính xác.

Merge cho phép bạn có thể so sánh và làm việc với các phiên bản khác nhau của tập tin bản tin, chẳng hạn như chương trình mã nguồn, html, xml và các tập tin. Merge có thể trích xuất và so sánh các bản tin từ Microsoft Office, OpenDocument, PDF và các tập tin rtf. Tệp tin XML có thể được hiển thị với các định dạng đặc biệt, giúp bạn xem các thay đổi một cách rõ ràng hơn. Nó hỗ trợ các tệp tin với mã ascii, mbcs (Mixed Byte Character Set) và ký tự Unicode Encodings. Liên kết giữa các dòng được trích ra các tài liệu được hiển thị rõ ràng như thế nào khi có liên quan

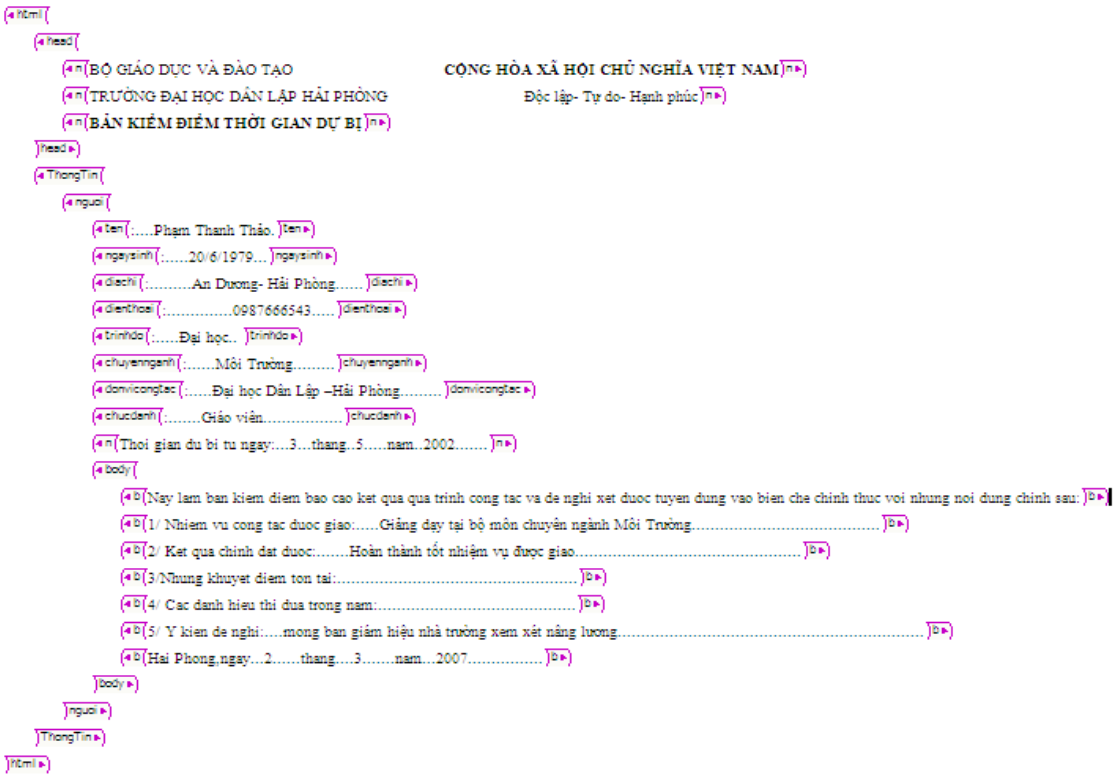
Merge cho thấy chi tiết nổi bật những thay đổi trong dòng. Nó có thể được cấu hình để bỏ qua sự khác biệt trong dòng và các hậu tố, cũng như các thay đổi trong dòng phù hợp với quy định.

Nhược điểm:

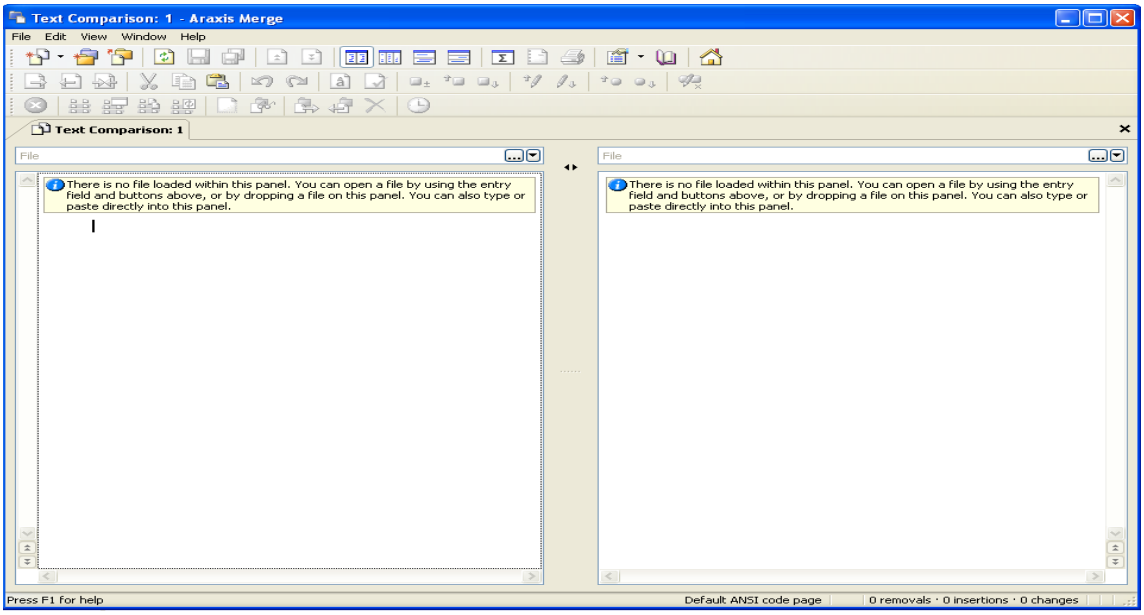
Trong quá trình hợp nhất các bản tin có cấu trúc vẫn còn một số nhược điểm chưa khắc phục được và chưa thân thiện với người dùng, đòi hỏi cần phải có một chương trình dễ hơn để hợp nhất các bản tin Tiếng Việt.

3.2 Mô hình thử nghiệm và đánh giá

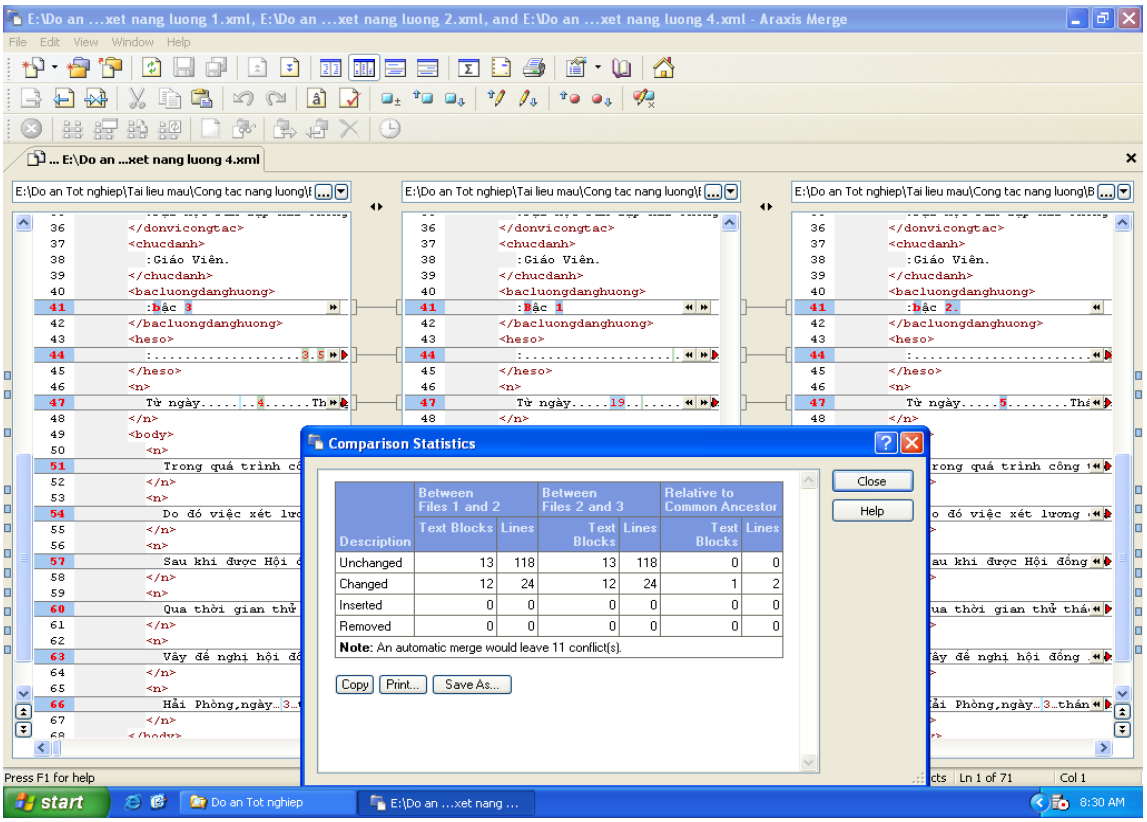
Cấu trúc của tài liệu XML



Mô hình của phần mềm Tree Way Merge Demostration

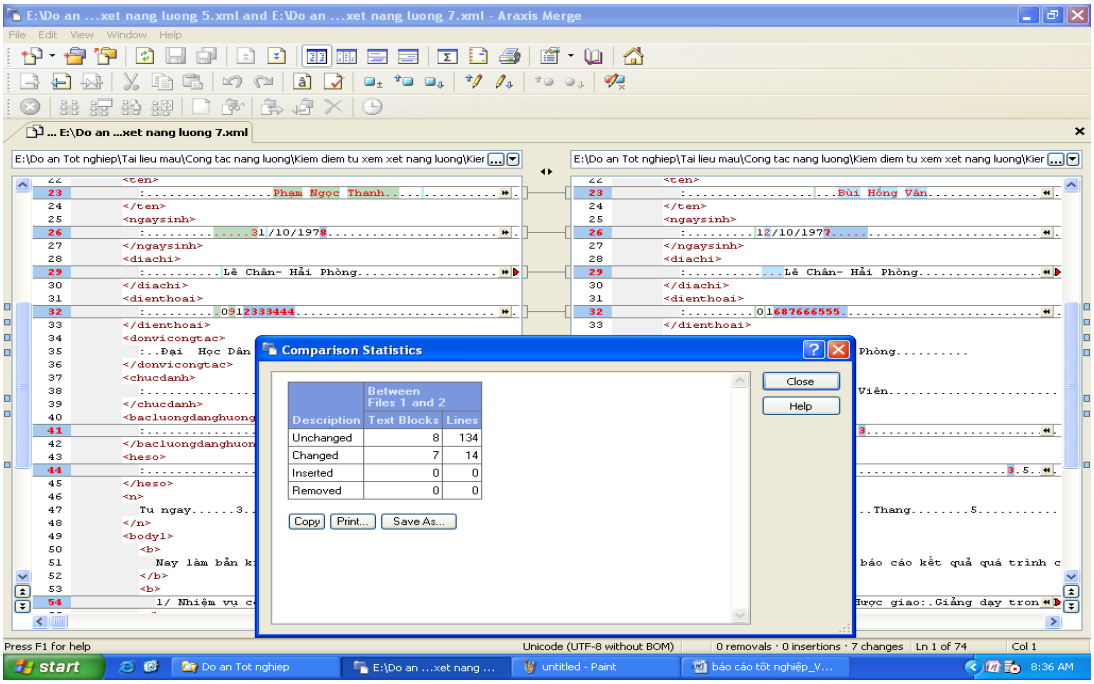


Một số các mô hình ví dụ hợp nhất 3-way



Description	Between Files 1 and 2		Between Files 2 and 3		Relative to Common Ancestor	
	Text Blocks	Lines	Text Blocks	Lines	Text Blocks	Lines
Unchanged	13	118	13	118	0	0
Changed	12	24	12	24	1	2
Inserted	0	0	0	0	0	0
Removed	0	0	0	0	0	0

Ví dụ về hợp nhất 2-way



Tập dữ liệu để đánh giá:

STT	Loại tài liệu	Số lượng	Nội dung
1	Quyết định	20	Thông báo quyết định của trường ĐHDLHP
2	Bản kiểm điểm	30	Công tác nâng lương
3	Biên bản	30	Quản lý bộ môn

Kết quả đánh giá:

Các kết qủa thử nghiệm trên các tập tin XML thực tế cho thấy chương trình chạy khá chính xác. Đặc biệt các trường hợp có liên quan đến ngữ nghĩa trên Tag Name cũng như các đụng độ trên Text Node đã được giải quyết theo hướng thân thiện người dùng và tỏ ra có ý nghĩa thực sự

- Chương trình giúp tiết kiệm thời gian và giảm các lỗi làm việc một cách nhanh chóng và chính xác

- Dễ dàng xử dụng đối với mọi người

- Thích hợp mở với các loại file

Kết luận

Hợp nhất thông tin có cấu trúc là bài toán rất cần thiết, đặc biệt là trong môi trường cộng tác nhiều người cùng chia sẻ một số thông tin hoặc trong môi trường một người dùng chia sẻ thông tin trên nhiều thiết bị. Nếu hệ thống của chúng ta là hệ thống mạng mạnh, bài toán đồng bộ hóa đã được giải quyết khá tốt, mục đích của đề tài này là nghiên cứu và phát triển công cụ hợp nhất để đồng bộ hóa trong môi trường mạng yếu với các tập tin có cấu trúc, được hỗ trợ tối thiểu của hệ thống.

Các điểm mới của luận văn bao gồm:

1 Về phương pháp tiếp cận: Điểm khác biệt căn bản của đồ án so với các tiếp cận hiện đó là chọn lựa cách hợp nhất 3-way nhưng mã hóa tập khác biệt dưới dạng một kịch bản chỉnh xửa để đồng bộ hóa dữ liệu có dạng XML. Cách tiếp cận này cho phép hợp nhất các bản tin có cấu trúc khác nhau và sinh tập khác biệt có kích thước cực tiểu.

2 Về kĩ thuật ánh xạ: Tận dụng tính gợi ý của các thẻ XML để tăng tính chính xác của thuật toán ánh xạ.

3 Về xử lí đụng độ: Đụng độ Tag Name được tinh tế hóa thông qua chọn lựa tự động-Đụng độ Text Node được xử lí linh động hơn thông qua thuật toán LCS, cho phép người dùng nhận biết các thay đổi trong Text Node

4 Hiện thực công cụ xử lí quá trình hợp nhất và đồng bộ hóa có tính ứng dụng cao, ngoài ra còn chứng minh khả năng hỗ trợ đa ngôn ngữ, cũng cho phép mở rộng ứng dụng hệ thống không chỉ trên các bản tin XML mà trên các dữ liệu có cấu trúc bất kì của công cụ.

Tuy nhiên vẫn còn nhiều vấn đề chưa được đề cập và giải quyết về vấn đề hợp nhất thông tin, chẳng hạn việc xem xét DTD của tập tin XML xử lí tự động việc hợp nhất không chỉ cấu trúc XML mà còn hợp nhất nội dung của Text Node. Tuy nhiên các nỗ lực của đồ án cho thấy có thể xây dựng một phần mềm thương mại dựa trên các vấn đề đã được phát triển trong đồ án.

Đề hướng phát triển trong tương lai

Đồ án đã giải quyết việc hợp nhất để đồng bộ hoá các bản tin có cấu trúc dạng XML và thử nghiệm cho thấy công cụ có khả năng đồng bộ hoá một cách hiệu quả trong môi trường mạng yếu

Tuy nhiên đây là những ý tưởng cải tiến bước đầu còn phải hoàn thiện nhiều hơn nữa mới có thể trở thành sản phẩm thương mại

Các vấn đề cần được giải quyết tiếp theo để hoàn thiện công cụ bao gồm:

->Nghiên cứu cải tiến thuật toán hợp nhất 3-way, nhất là các trường hợp đụng độ cấu trúc.

-> Nghiên cứu ứng dụng các thuật toán tạo khác biệt mới để có tập khác biệt càng nhỏ càng tốt.

-> Thể hiện kịch bản chỉnh xửa cũng như các bản tin XML dưới dạng thân thiện người dùng.

-> Xử lí DTD: hai tập tin có cấu trúc giồng nhau nhưng DTD khác nhau cần phải được nhận biết.

Tài liệu tham khảo

[1] Asklund U. – Identifying Conflicts During Structural Merge – Proceeding of the Nordic Workshop on Programming Environment Research ‘ 94 . Lund Universit y, 1994.

[2] Cederqvist P. Et al. – Version Management with CVS – Signum Support AB, Linkoping, Swenden, 1993. http://www.loria.fr/~molli/cvs/doc/cvs.pdf

[3] Eric Amstrong - Working with XML – http://java.sun.com/xml/jaxp-1.1/docs/tutorial/index.html

[4] IBM Alphaworks. – XML diff and merge tôl home page http:// www.alphaworks.ibm.com/tech/xmldiffmerge

[5] http:// www.W3c.org – World wide web consortium (W3C)