ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

BÁO CÁO THỰC TẬP TỐT NGHIỆP

XÂY DỰNG HỆ THỐNG RÚT TRÍCH THÔNG TIN ĐỊA CHỈ TỪ INTERNET

GVHD: **ThS. Dương Ngọc Hiếu**

TS. Nguyễn Thanh Hiên

GVPB: **TS. Võ Thị Ngọc Châu**

---o0o---

SVTH:	**Hoàng Nguyên**	**50801398**
	Lê Hùng Vinh	**50802626**
	Trần Vĩ Kiệt	**50801026**

TP. HỒ CHÍ MINH, THÁNG 6/2012

MỤC LỤC

CHƯƠNG I. GIỚI THIỆU

1.1 Mở đầu

Ngày nay, World Wide Web tạo ra nhiều cơ hội và thách thức cho việc khai thác thông tin trên mạng Internet.

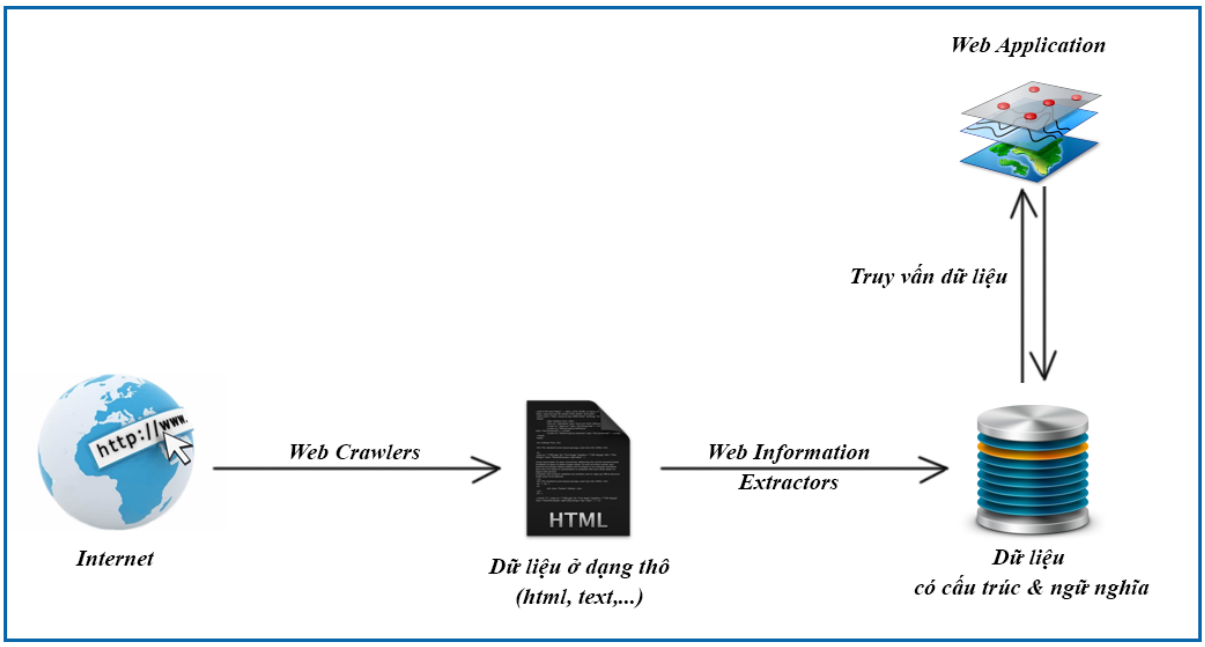
- Lượng thông tin trên Web là rất *lớn, đa dạng và dễ dàng tiếp cận*.
  Lượng thông tin là rất phong phú và đa dạng, thuộc nhiều lĩnh vực khác nhau: khoa học, kinh tế, tin tức…, chỉ cần có Internet và trình duyệt, là chúng ta có thể tiếp cận những thông tin này tại bất cứ ở đâu và lúc nào.
- Lượng thông tin, dữ liệu ở *nhiều dạng khác nhau*: văn bản, bảng có cấu trúc, đa phương tiện(video, audio,…)… Hầu hết định dạng dữ liệu web là semi-structured, cụ thể là nested structure.
- Các thông tin trên web *được liên kết với nhau* bên trong một trang web và giữa các trang khác nhau.
- Thông tin trên Web *chứa nhiều dữ liệu dư thừa*: Một số thành phần của thông tin không có ích hoặc các biến thể của chúng xuất hiện nhiều lần trên nhiều trang(quảng cáo, copyright, thành phần điều hướng..).
- Dữ liệu trên Web là *động*: thông tin trên web thay đổi và cập nhật liên tục. Vì vậy, vấn đề theo dõi các thay đổi và cập nhật là một trong những vấn đề quan trọng.
- Web có *tính xã hội(virtual society)*: không chỉ chứa dữ liệu, thông tin, dịch vụ.. mà Web còn biểu hiện nhiều sự tương tác giữa các người dung, tổ chức, hệ thống khác nhau.

Thông tin là kho tàng lớn nhất mà Internet mang lại, nhưng thông tin thường ở nhiều dạng và chưa được tinh chiết. Vì vậy, việc thu thập lượng thông tin khổng lồ này và xây dựng chúng thành dạng có cấu trúc và ngữ nghĩa là một trong những tiền đề để xây dựng một hệ thống khai thác thông tin trên mạng Internet.

1.2 Yêu cầu & mục tiêu đề tài

1.2.1 Yêu cầu

Yêu cầu đặt ra là *Xây dựng hệ thống rút trích thông tin địa chỉ từ Internet.*Thông tin địa chỉ được hiểu là các thông tin cơ bản liên quan đến địa danh tại một địa chỉ xác định (VD: nếu tại địa chỉ là trụ sở công ty thì thông tin cơ bản bao gồm: tên công ty, địa chỉ chính xác, điện thoại, fax, email, website công ty, ngành nghề kinh doanh của công ty). Sau khi thu thập xong thông tin địa chỉ cần có cơ chế để người dùng xem các thông tin đã thu thập được.



Hình 1 Tổng quan hệ thống

1.2.2 *Mục tiêu*

Mục tiêu của nhóm là xây dựng một hệ thống có khả năng phân tích các dữ liệu dạng thô được thu thập(crawl) trên Internet, trích xuất các thông tin có ích về địa chỉ, lưu trữ chúng dưới dạng có cấu trúc(trong cơ sở dữ liệu), sau đó, hiện thực một ứng dụng sử dụng nguồn dữ liệu đã được rút trích ở bên trên.

Với mục tiêu như trên, hệ thống cần xây dựng bao gồm các thành phần như sau:

(1) *Web Information Extractors*: Xử lý dữ liệu thô thành các đối tượng dữ liệu có cấu trúc và ngữ nghĩa.

(2) *Web Application:* Xây dựng ứng dụng khai thác dữ liệu địa chỉ trên.

CHƯƠNG II. PHÂN TÍCH YÊU CẦU

2.1 Dữ liệu cần extract

Dựa trên yêu cầu của hệ thống: thu thập dữ liệu liên quan đến địa chỉ. Nhóm đã xác định các thuộc tính liên quan đến địa chỉ cần extract

- Tên địa danh (VD: công ty A, ngân hàng B …)
- Địa chỉ chính xác của địa danh (VD: phòng, tầng …)
- E-mail (nếu có)
- Website (nếu có)
- Số điện thoại (nếu có)
- Số fax (nếu có)
- Ngành nghề kinh doanh (nếu có)

2.2 Phương pháp extract

Nhóm xác định nguồn dữ liệu của hệ thống là web pages từ các website khác nhau (VD : website trang vàng). Web page là dạng văn bản có cấu trúc, do đó thuận lợi hơn trong việc tìm ra một phương pháp rút trích thông tin từ chúng. Web page có 2 dạng:

+ List page: chứa 1 hoặc nhiều list, mỗi list gồm nhiều data record được định dạng theo cấu trúc giống nhau.

+ Detail page: page chứa thông tin chi tiết về 1 đối tượng

Qua quá trình khảo sát một số nguồn dữ liệu, nhóm nhận thấy nguồn dữ liệu cho hệ thống bao gồm cả hai dạng page này. Do đó cần tìm ra một phương pháp tự động extract dữ liệu từ cả 2 dạng page này.

Đã có nhiều công trình nghiên cứu về extract data từ web page. Có 3 hướng tiếp cận chính :

+ Manual approach

+ Wrapper induction

+ Automatic extraction

2.2.1 Manual approach

Người lập trình quan sát webpage và source code của nó để tìm ra pattern, sau đó viết chương trình để lấy dữ liệu, có thể sử dụng các công cụ : xpath… để việc lấy dữ liệu dễ dàng hơn. Chỉ ứng với từng website cụ thể, không thể mở rộng.

2.2.2 Wrapper induction

Dựa vào một số page được đánh dấu sẵn vị trí của data record, hệ thống tự động rút ra quy luật và dùng luật này để extract data từ những page có cấu trúc tương tự.

Phương pháp này bộc lộ nhiều nhược điểm:

- Tốn công sức và thời gian cho việc đánh dấu vị trí data record bằng tay.

- Chi phí cao khi áp dụng cho nhiều site khác nhau.

2.2.3 Automatic extraction

Cho trước một hoặc một số page, hệ thống sẽ tự động tìm ra pattern để extract data. Cách tiếp cận này loại bỏ việc đánh dấu vị trí bằng tay, do đó có thể áp đụng cho 1 số lượng lớn page và site.

Một giải thuật thuộc cách tiếp cận này là giải thuật NET của tác giả Bing Liu. Giải thuật này dựa trên sự so trùng DOM tree để tìm ra pattern cho việc extract data. Giải thuật áp dụng cho list page, tuy nhiên có thể dễ dàng điều chỉnh để áp dụng cho detail page bằng cách tạo một DOM tree từ các detail page này.

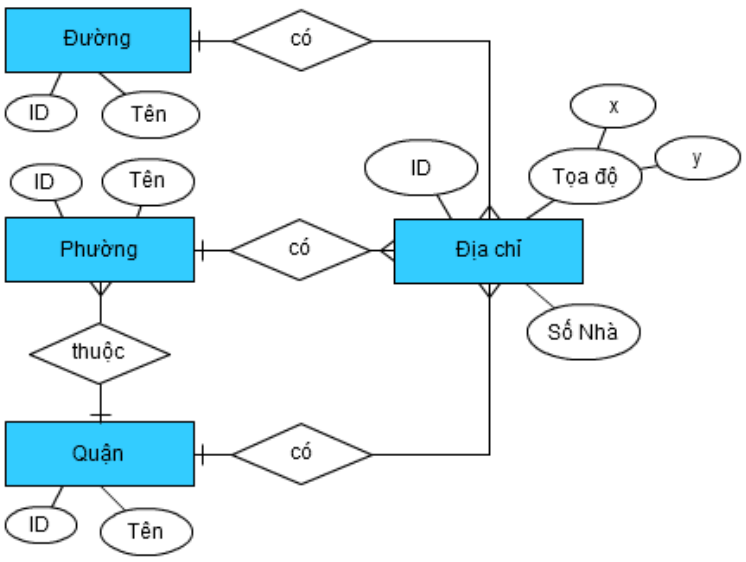
Nhược điểm dễ nhận thấy của giải thuật là chưa thể tạo được attribute name cho dữ liệu được extract => cần có cơ chế nhận dạng, gán nhãn cho dữ liệu. Nhóm sử dụng heuristic dựa trên các đặc điểm của dữ liệu để làm việc này.

2.3 Khớp dữ liệu extract được và dữ liệu hiện có

2.3.1 Cơ sở dữ liệu hiện có

Cơ sở dữ liệu hiện có là một cơ sở dữ liệu chứa đựng thông tin về các khu đất được đánh địa chỉ tại Thành phố Hồ Chí Minh kèm theo tọa độ cụ thể của khu đất.

Chi tiết về cơ sở dữ liệu này được biểu diễn trong ERD dưới đây



Hình 2 ERD của cơ sở dữ liệu hiên có

ERD trên đã được map thành các bảng như sau:

tbl_ConDuong		
column name	*type*	*key*
Id	int	PK
TenConDuong	nvarchar(100)	
TenKhongDau	nvarchar(100)	

tbl_Quan		
column name	*type*	*key*
Id	int	PK
TenQuan	nvarchar(100)	
TenKhongDau	nvarchar(100)	

tbl_Phuong		
column name	*type*	*key*
Id	int	PK
TenPhuong	nvarchar(100)	
TenKhongDau	nvarchar(100)	
QuanId	int	FK(tbl_Quan)

tbl_DiaChi		
column name	*type*	*key*
Id	int	PK
SoNha	nvarchar(100)	
DuongId	int	FK(tbl_Duong)
PhuongId	int	FK(tbl_Phuong)
QuanId	int	FK(tbl_Quan)
X	float	
Y	float	

2.3.2 Cách thức so khớp

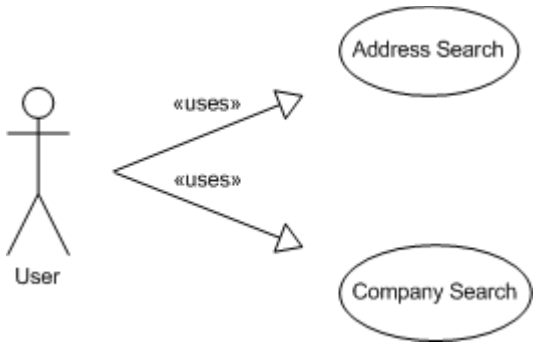
Việc khớp các đối tượng dữ liệu đã được extract vào cơ sở dữ liệu hiện có dựa trên thuộc tính về địa chỉ chính xác của đối tượng dữ liệu. Quá trình map được thực hiện bằng 2 bước:

- Phân tích chuỗi địa chỉ chính xác để có được số nhà, tên đường, phường, quận.

- Map đối tượng dữ liệu vào 1 địa chỉ trong cơ sở dữ liệu hiện có dựa trên 4 thuộc tính vừa được phân tích.

2.4 Xây dựng web application

Dựa trên yêu cầu xây dựng cơ chế cho phép người dùng tìm kiếm trên dữ liệu địa chỉ vừa extract. Nhóm dự định xây dựng website có các chức năng như sơ đồ use-case dưới đây



Hình 3 Lược đồ use-case cho web application

Trong đó:

- Address Search: người dùng nhập vào một địa chỉ cụ thể, hệ thống trả về các địa danh tại địa chỉ này.
- Company Search: người dùng nhập tên một công ty , dịch vụ hay một cửa hàng nào đó, hệ thống trả về vị trí của địa danh và thông tin cơ bản của địa danh đó.

CHƯƠNG III. CƠ SỞ LÝ THUYẾT

3.1 Giới thiệu

Giải thuật NET là giải thuật rút trích thông tin từ web page chứa nhiều data record theo cách tiếp cận automatic extraction. Giải thuật dựa trên phương pháp chi phí chuyển đổi cây (tree edit distance method) và sắp xếp cây (tree alignment).

3.2 Tree Matching và giải thuật Simple Tree Matching

3.2.1 Tree Matching

Việc trích xuất dữ liệu từ các trang Web có thể được thực hiện thông qua việc phân tích cấu trúc của trang Web đó. Cụ thể là nhóm các trang có cùng một cấu trúc thành một *nhóm trang* và tìm những biểu diễn giống nhau của cấu trúc của các trang Web này trong một nhóm.

Để đánh giá mức độ giống nhau giữa các trang ta sử dụng khái niệm *Chi phí chuyển đổi cây (Tree Edit Distance).*

Chúng ta có khái niệm *cây có thứ tự, được gán nhãn, có gốc cố định*(*labeled ordered tree*) cho các cây có đỉnh gốc là cố định, có thứ tự các con là cố định đối với mỗi đỉnh, và với mỗi đỉnh được gán một nhãn l cố định. Các cây được mô tả trong toàn bộ phần giải thuật này là labeled ordered tree.

Chi phí chuyển đổi cây giữa 2 cây *A* và *B* là chi phí phải trả cho một tập tối thiểu các thao tác cần thiết để chuyển đổi A thành B.

Các thao tác có thể, bao gồm: thêm nút, xóa nút, thay thế nút. Mỗi thao tác có một chi phí xác định.

Cho cây *X* bất kì và *X[i]* là node thứ *i* của xây *X* theo thứ tự duyệt *preorder*(NLR).

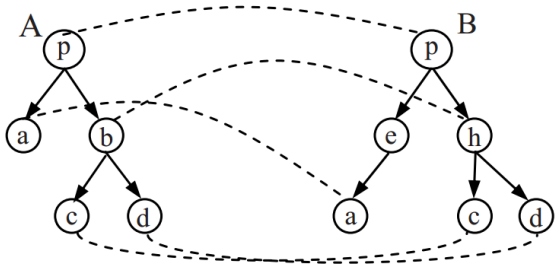
Một *ánh xạ(mapping) M* giữa cây *A* có kích thước n_1 và cây *B* có kích thước n_2 là một tập các cặp (i, j) có thứ tự, sao cho:

Với mọi (i_1, j_1), $(i_2, j_2) \in M$ thì:

(1) $i_1 = i_2$ *khi và chỉ khi* $j_1 = j_2$
(2) $A[i_1]$ *ở bên trái của* $A[i_2]$ *khi và chỉ khi* $A[j_1]$ *ở bên trái của* $A[j_2]$
(3) $A[i_1]$ *là tổ tiên của* $A[i_2]$ *khi và chỉ khi* $A[j_1]$ *là tổ tiên của* $A[j_2]$

Các điều kiện trên xác định một đỉnh của một cây không xuất hiện quá một lần trong một ánh xạ, và bảo toàn thứ tự giữa các nút con và quan hệ phân tầng giữa các nút.

Một số giải thuật đã được đưa ra để giải quyết bài toán tìm tập tối thiểu các thao tác để chuyển đổi từ một cây này sang một cây khác. Tuy nhiên tất cả các công thức đều có độ phức tạp *trên cấp đa thức bậc hai.*



Hình 4 Một ví dụ về phép ánh xạ giữa hai cậy A và B

Với những định nghĩa chung trên, ánh xạ có thể thực hiện giữa các tầng khác nhau(cross levels) giữa hai cây, ví dụ, nút *a* ở trong cây *A* và nút *a* ở trong cây *B*(hình 1). Những thay thế cũng được cho phép, ví dụ, nút *b* ở cây *A* và nút *h* ở cây *B*.

Chúng ta sẽ định nghĩa một ánh xạ giới hạn – được gọi là *Simple Tree Matching(STM)*, trong đó, không có thay thế và sự thay thế ở khác tầng.

3.2.2 *Simple Tree Matching*

Cho hai cây *A*, *B* và $i \in A$ và $j \in B$ là 2 nút trên các cây *A*, *B* bất kỳ. Một phép so trùng(matching) giữa hai cây là ánh xạ *M*, bao gồm tất cả các cặp $(i, j) \in$ M, sao cho i và j không phải là nút gốc, $(parent(i), parent(j)) \in$ M.

Một phép *so trùng tối đa(maximum matching)* là một phép so trùng với số lượng lớn nhất các cặp (i, j) này có thể đạt tới.

Cho cây $A = R_A{:} < A_1, \dots, A_k >$ và $B = B_A{:} < B_1, \dots, B_n >$, trong đó R_A và R_B là gốc của *A* và *B*; và A_i và B_j là cây con tầng đầu tiên thứ *i* và *j* của các cây *A* và *B* bất kì.

Gọi *W(A, B)* là số lượng các cặp trong phép so trùng tối đa giữa hai cây *A* và *B*.

Ta có các công thức sau:

$$W(A,B) = \begin{cases} 0 & \text{nếu } R_A \neq R_B \\ m(< A_1, \dots, A_k >, < B_1, \dots, B_n >) + 1 & \text{còn lại} \end{cases}$$

m(< >, < >) = 0 // < > mô tả một danh sách cây con rỗng.

m(s, < >) = m(< >, s) = 0 // s không match được với bất cứ một danh sách cây con rỗng nào.

$$m(< A_1, \dots, A_k >, < B_1, \dots, B_n >) = \max \begin{pmatrix} m(< A_1, \dots, A_{k-1} >, < B_1, \dots, B_{n-1} >) + W(A_k, B_n), \\ m(< A_1, \dots, A_k >, < B_1, \dots, B_{n-1} >), \\ m(< A_1, \dots, A_{k-1} >, < B_1, \dots, B_n >) \end{pmatrix}$$

Giải thuật *Simple Tree Matching*(*STM)* tính toán giá trị $W(A, B)$ được mô tả như sau:

Algorithm: STM(A, B)

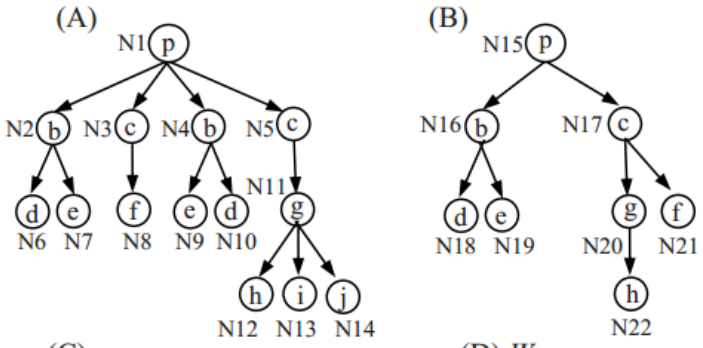
```
1.  if các gốc của hai cây A và B chứa các giá trị khác nhau thì
2.     return(0)
3.  else  k ← số cây con tầng đầu tiên của A;
4.        n ← số cây con tầng đầu tiên của B;
5.        Khởi tạo: m[i, 0] ← 0 cho i = 0, …, k;
                    m[0, j] ← 0 cho j = 0, ... , n;
6.        for i = 1 to k do
7.            for j = 1 to n do
8.                m[i, j] ← max(m[i, j – 1], m[i – 1, j], m[i – 1, j – 1] + W[i, j]),
                      where W[i, j] ← STM(A_i, B_j)
9.            end-for
10.       end-for
11.       return (m[k, n] + 1)
12. end-if
```

STM là một giải thuật top-down, tính toán sự giống nhau bằng cách tìm giá trị *so trùng tối đa* bằng tính toán động(dynamic programming).

Giải thuật này có *độ phức tạp* là $O(n_1, n_2)$, với n_1 và n_2 là kích cỡ của các cây *A* và *B* bất kỳ.

Dòng 1: nút gốc của *A* và *B* sẽ được so sánh đầu tiên. Nếu các nút gốc này chứa các giá trị khác nhau thì hai cây không tồn tại một ánh xạ nào cả. Còn không, giải thuật đệ quy tìm số lượng tối đa các ánh xạ giữa các cây con tầng đầu tiên của *A* và *B*, và các giá trị này trong ma trận *W*. Dựa vào ma trận *W*, một quy tắc tính toán động sẽ được áp dụng để tìm ra số lượng các cặp trong ánh xạ tối đa giữa hai cây *A* và *B*.

Giải thuật được giải thích bằng một ví dụ đơn giản ở Hình 5.



(C) $m_{1,15}$

	0	1 (N16)	2 (N16-N17)
0	0	0	0
1 (N2)	0	3	3
2 (N2-N3)	0	3	5
3 (N2-N4)	0	3	5
4 (N2-N5)	0	3	6

(D) $W_{1,15}$

	1 (N16)	2 (N17)
1 (N2)	3	0
2 (N3)	0	2
3 (N4)	2	0
4 (N5)	0	3

(E) $m_{5,17}$

	0	1 (N20)	2 (N20-N21)
0	0	0	0
1 (N11)	0	2	2

(F) $W_{5,17}$

	1(N20)	2(N21)
1 (N11)	2	0

(G) $m_{11,20}$

	0	1 (N22)
0	0	0
1 (N12)	0	1
2 (N12-N13)	0	1
3 (N12-N14)	0	1

(H) $W_{11,20}$

	1 (N22)
1 (N12)	1
2 (N13)	0
3 (N14)	0

Hình 5 *(A) Cây A; (B) Cây B; (C) ma trận m cho các cây con tầng đầu tiên của N1 và N15; (D) ma trận W cho các cây con tầng đầu tiên của N1 và N15; (E)-(H) ma trận m và W cho các cây con tầng thấp hơn.*

Để tìm *ánh xạ tối đa* giữa hai cây *A* và *B*, các nút gốc của chúng(*N1* và *N15*) được so sánh trước. Vì *N1* và *N15* chứa các giá trị giống nhau, nên ánh xạ tối đa được trả về là $m_{1,15}[4, 2] + 1$. Ma trận $m_{1,15}$ được tính toán dựa trên ma trận $W_{1,15}$. Mỗi giá trị thuộc $W_{1,15}$ là giá trị ánh xạ tối đa giữa cây con tầng đầu tiên thứ i và thứ j của A và B. Ví dụ trên, $W_{1,15}[4, 2]$ được tính toán đệ quy bằng cách xây dựng các ma trận *(E)-(H).* Các ô của hàng 0 và cột 0 của ma trận m được khởi tạo với giá trị 0. Chú ý chỉ số bên dưới(1, 15) ma trận *W* và m là nút mà chúng ta đang tính toán tại đó

Giá trị *normalized simple tree matching* NSTM(A, B) được tính bằng cách chia giá trị *STM* cho trung bình cộng số lượng nút ở hai cây.

$$NSTM(A,B) = \frac{STM(A,B)}{(nodes(A) + nodes(B))/2}$$

Mẫu thức còn có thể sử dụng *max(nodes(A), nodes(B)), nodes(X)* là số lượng nút trên cây *X*.

Sau khi tính toán, chúng ta lần ngược các giá trị trong ma trận m để sắp các nút từ hai cây.

3.3 Multiple Alignment và giải thuật Partial Tree Alignment

3.3.1 Multiple Alignment

Để tìm ra mô hình (pattern) lặp từ chuỗi HTML dựa vào chi phí chuyển đổi chuỗi (string edit distance) hoặc kết hợp cây (tree matching) , chúng ta cần sự sắp xếp(alignment) của các chuỗi và cây đó. Một trang web thường chứa hơn hai data record, do đó có nhiều hơn hai chuỗi hoặc cây cần phải được liên kết. Việc tạo ra một sự liên kết tổng thể của tất cả các chuỗi hoặc cây là rất quan trọng. Đó chính là Multiple Alignment .

Trong giới hạn báo cáo này , nhóm sẽ đề cập đến giải thuật Partial Tree Alignment , một phương pháp sắp xếp theo kinh nghiệm .

3.3.2 Partial Tree Alignment

3.3.2.1. Partial Alignment of Two Trees

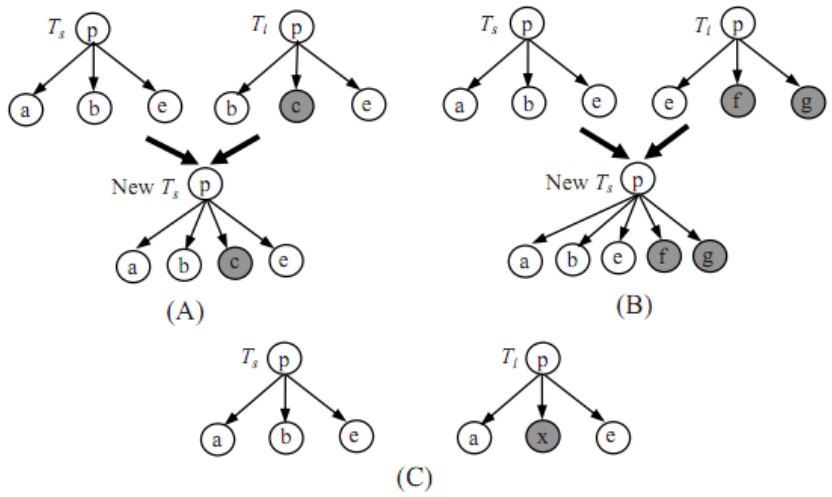
Trước khi đến với một giải thuật đầy đủ , ta sẽ nhìn vào sự liên kết một phần của 2 cây . Sau khi 2 cây Ts và Ti được kết , một số nút trong Ti có thể được liên kết với các nút tương ứng của Ts vì chúng phù hợp với nhau . Đối với những nút trong Ti không phù hợp, ta sẽ chèn chúng vào Ts với các dữ liệu là tùy chọn.Có hai tình huống có thể khi chèn một nút mới vi từ Ti vào Ts, tùy thuộc vào vị trí trong Ts có thể được xác định duy nhất để chèn vi.Thay vì xem xét một nút duy nhất vi, chúng ta có thể xem xét một tập các nút liền kề VJ ... vm chưa được liên kết từ Ti với nhau. Không mất tính tổng quát, ta giả định rằng nút cha của VJ ... vm có sự trùng khớp trong Ts và ta muốn chèn VJ ... vm vào Ts dưới cùng một nút cha. Ta chỉ chèn VJ ... vm vào Ts nếu một vị trí để chèn VJ ... vm có thể được xác định duy nhất trong Ts.Nếu không, chúng sẽ không được chèn vào Ts (unaligned). Các vị trí để chèn VJ ... vm có thể chỉ được quyết định bởi :

1. Nếu vj...vm có 2 nút kết cận trong Ti , một bên phải và một bên trái ,thì chúng trùng với 2 nút liên tiếp trong Ts (Hình A)

2. Nếu vj ... vm chỉ có 1 nút trái x thuộc Ti và x lại trùng với nút phải nhất trong Ts , thì vj ... vm có thể được thêm vào sau nút x trong Ts (Hình B)

3. Nếu vj ... vm chỉ có 1 nút phả x thuộc Ti và x lại trùng với nút trái nhất trong Ts , thì vj ... vm có thể được thêm vào trước nút x trong Ts

Các trường hợp còn lại , ta không thể xác định được vị trí duy nhất để thêm vào Ts (Hình C)



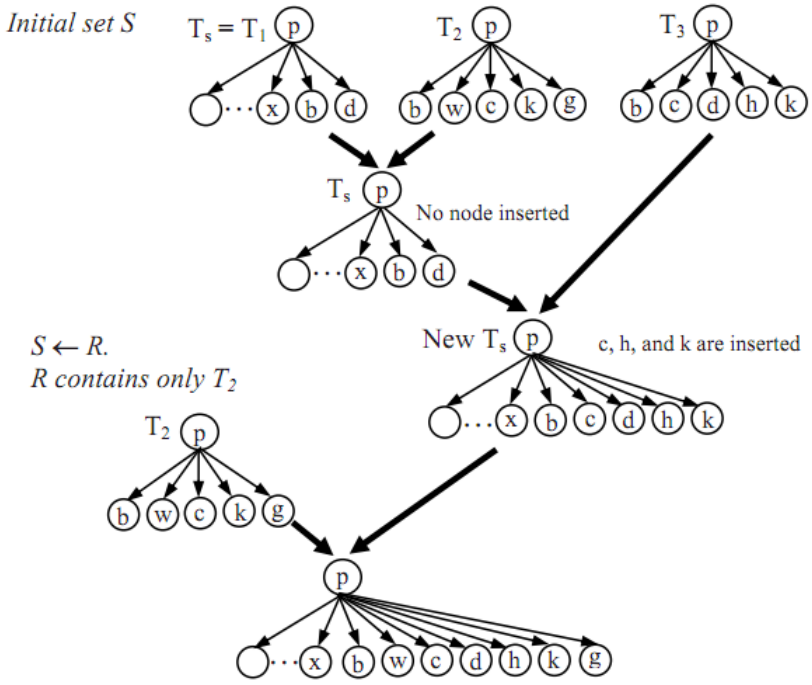
Hình 6 Mở rộng cây seed (A) và (B) có thể chèn, (C) nhập nhằng

3.3.2.2. Partial Alignment of Multiple Trees

Algorithm PartialTreeAlignment(S)

1. Sort trees in S in descending order of the number of unaligned data items;
2. Ts ← the first tree (which is the largest) and delete it from S;
3. R ← ∅;
4. **while** (S ≠ ∅) **do**
5. Ti ← select and delete next tree from S; // follow the sorted order
6. STM(Ts, Ti); // tree matching
7. AlignTrees(Ts, Ti); // based on the result from line 6
8. **if** Ti is not completely aligned with Ts **then**
9. **if** InsertIntoSeed(Ts, Ti) **then** // True: some insertions are done
10. S ← S ∪ R;
11. R ← ∅
12 **endif;**
13. **if** there are still unaligned items in Ti that are not inserted into Ts **then**
14. R ← R ∪ {Ti}
15. **endif;**
16. **endif;**
17. **endwhile;**
18. Output data fields from each Ti to a data table based on the alignment results.

Đây là giải thuật đầy đủ đối với trường hợp nhiều cây dựa trên giải thuật partial alignment trên 2 cây . Dòng 1-2 (Trong giải thuật) sẽ tìm cây chứa nhiều data items nhất . Nó được dùng như cây giống Ts (Seed Tree) . Dòng 3 khởi tạo R , cái sẽ chứa những cây không hoàn toàn được sắp thẳng với Ts trong mỗi vòng lặp . Dòng 4 bắt đầu vòng lặp While để sắp xếp mỗi các cây với Ts . Dòng 5 lấy chưa được sắp xếp kế tiếp , dòng 6 thực hiện việc kết nối . Dòng 7 tìm tất cả các cặp trùng khớp bằng cách lần theo ma trận kết quả từ dòng 6 . Hàm này tương tự như việc sắp xếp 2 chuỗi sử dụng khoảng cách chỉnh sửa . Dòng 8 và dòng 9 cố găng thêm những nút không trùng vào Ts . Đây là giải thuật partial tree alignment đã nêu ở trên . Dòng 13 – 14 thêm T_2 vào R , là danh sách các cây cần được kết hợp lại vì một số data items chưa được sắp xếp và chưa được thêm vào Ts . Dòng 10 – 11 đưa những cây trong R vào S và khởi tạo lại R . Dòng 18 xuất ra các data items từ mỗi cây theo sự sắp xếp . Hình bên dưới minh hoạ cho giải thuật trên



Hình 7 Tree Alignment với hai vòng lặp

Final data table ("1" indicates a data item)

	…	x	b	w	c	d	h	k	g
T_1	…	1	1			1			
T_2			1	1	1			1	1
T_3			1		1	1	1	1	

Hình 8 Bảng dữ liệu sau khi kết thúc giải thuật ("1" để chỉ một data item)

Trong thực tế , để có một giải thuật hoàn thiện , một lời gọi đệ quy nên được thêm vào sau dòng 17 để sửa lý trường hợp $R \neq 0$. Ba dòng sau có thể được dùng

18. **if** $R \neq \emptyset$ **then**
19. PartialTreeAlignment(R)
20. **endif**

Điều này sẽ đảm bảo được trường hợp có những item chưa được sắp xếp và thêm vào . Ta cần lưu ý 2 điểm đối với giải thuật hoàn thiện này . Thứ 1 , lời gọi đệ quy sẽ dừng ngay cả khi không có một sắp xếp nào hay một việc thêm nào được tạo ra đối với cây giống bởi vì cây giống đã bị xoá trong mỗi lần gọi đệ quy và do đó R sẽ càng lúc càng nhỏ . Thứ 2 , giải thuật có thể tìm thấy nhiều mẫu trong dữ liệu . Cây giống từ mỗi lần gọi đệ quy sẽ đại diện cho một mẫu khác nhau .

3.4 Giải thuật NET

3.4.1 Hai điều quan sát về data record

Trước khi giới thiệu giải thuật, chúng tôi xin trình bày 2 điều quan sát của tác giả về data record.

Việc tìm kiếm các data record trong một web page có thể xem như việc tìm ra những cấu trúc lặp đi lặp lại trong page và sắp xếp chúng. Phương pháp so sánh chuỗi / cây là cách tiếp cận tự nhiên. Vấn đề là sự hiệu quả của chúng, bởi một data record có thể được bắt đầu tại bất cứ đâu trong page và có chiều dài bất kỳ. Nếu tất cả các data record có cùng chuỗi tag thì vấn đề trở nên đơn giản. Tuy nhiên trong thực tế, các data record không giống nhau hoàn toàn do một số item là optional. Hai quan sát dưới đây, dựa trên cấu trúc cây của web page, sẽ giúp giải quyết vấn đề này.

(1) Một nhóm các data record chứa thông tin của các đối tượng giống nhau được đặt tại một vùng liên tiếp trên page và được định dạng bằng các thẻ HTML tương tự nhau.

(2) Các data record trong cùng một vùng trên page được tạo ra từ các cây con của cùng một node. Một data record không thể bắt đầu từ giữa một cây con và kết thúc ở giữa một cây con khác. Nó thường

bắt đầu từ phần đầu của một cây con và kết thúc tại cuối cây con đó hoặc cuối một cây con kế sau.

Dựa trên hai quan sát này ta có thể xây dựng một phương pháp trích xuất dữ liệu dựa vào cấu trúc cây của web page.

3.4.2 Giải thuật

Ý tưởng cơ bản của giải thuật là duyệt cây DOM theo thứ tự từ dưới lên (post-order) . Các data record sẽ được tìm ra dựa trên sự lặp lại của cấu trúc cây. Những data record lồng nhau được xử lý ở mức thấp trước khi xử lý ở mức cha cao hơn.

```
Algorithm NET(Root, τ)
1    TraverseAndMatch(Root, τ);
2    for each top level node Node whose children have aligned data records do
3    PutDataInTables(Node);
4    endfor
```

Hàm TraverseAndMatch duyệt cây DOM theo thứ tự từ dưới lên (post-order). Trong khi duyệt, các data record cùng kiểu sẽ được biến đổi. Hàm PutDataInTables sẽ xuất dữ liệu rút trích được thành dạng bảng (một page có thể có nhiều vùng dữ liệu, dữ liệu trong mỗi vùng sẽ được xuất ra thành một bảng riêng).

```
Function TraverseAndMatch (Node, τ)
1    for each Child ∈ Node.Children do
2    TraverseAndMatch(Child, τ);
3    endfor
4    Match(Node, τ);
```

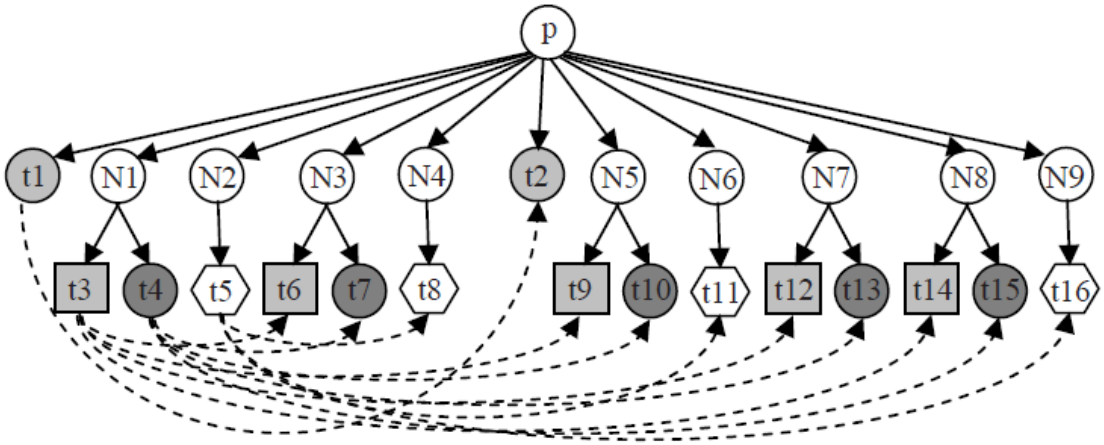
Hàm Match thực hiện tree matching trên các cây con của Node và tạo ra mẫu chung (pattern). τ là ngưỡng để xác định hai cây có giống nhau hay không.

```
Function Match(Node, τ)
1    Children ← Node.Children;
2    while Children ≠ ∅ do
3        ChildFirst ← select and remove the first child from Children;
4        for each ChildR in Children do
5            if TreeMatch(ChildFirst, ChildR) > τ then
6            AlignAndLink();
7            Children ← Children – {ChildR}
```

8	**endfor**
9	**if** some alignments (or links) have been made with *ChildFirst* **then**
10	GenNodePattern(*ChildFirst*)
11	**endwhile**
12	**If** consecutive child nodes in *Children* are aligned **then**
13	GenRecordPattern(*Node*)

Hình bên dưới là một ví dụ. Trong hình này Ni biểu diễn node trong, ti biểu diễn node lá có chứa data item.

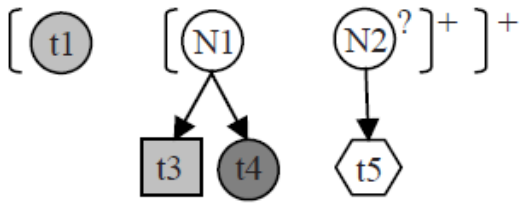
Cho trước đối số đầu vào Node, dòng 1 của hàm Match lấy ra tất cả cây con của Node để so trùng. Trong hình ví dụ, con của p là các cây t1, N1, N2, N3, N4, 2, N5, N6, N7, N8, N9. Dòng 2-4 so trùng từng cặp cây con. Việc so trùng được thực hiện bởi hàm TreeMatch. Hàm này sử dụng giải thuật Simple Tree Matching đã được đề cập ở phần trước. Hàm AlignAndLink ở dòng 6 sắp xếp và nối những data item trùng khớp với nhau. Nếu ChildR trùng khớp với ChildFirst, ChildR sẽ được loại ra khỏi Children để tránh bị so trùng lần nữa. Trong ví dụ của chúng ta, sau dòng 4-11, các đường nối giữa các data item được biểu diễn bằng nét đứt (giả sử chúng đã thỏa điều kiện trùng khớp)



Hình 9 Cây với tất cả item trùng khớp đã được nối.

Tại dòng 9-10, nếu có cây con trùng khớp với ChildFirst, hàm GenNodePattern sẽ đuược gọi để tạo node pattern từ những cây con khớp nhau. Hàm này sử dụng giải thuật Partial Tree Alignment trên tập cây con trùng khớp để tạo pattern.

Tại dòng 12-13, pattern cho data record được tạo ra.



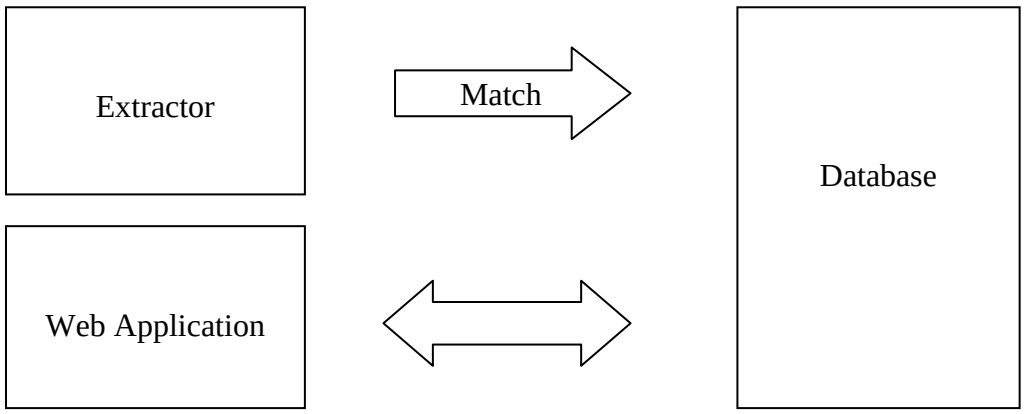
Hình 10 Record pattern được tạo ra từ ví dụ trên

Cuối cùng, hàm PutDataInTables xuất dữ liệu ra bảng. Việc này khá đơn giản khi record pattern đã được tạo ra.

CHƯƠNG IV. THIẾT KẾ VÀ HIỆN THỰC

4.1 Tổng quan hệ thống

Hệ thống gồm 2 hệ thống con Extractor và Web Application như hình vẽ sau



Extractor rút trích thông tin địa chỉ từ nguồn dữ liệu về database.
Web Application sử dụng database để cung cấp thông tin cho người sử dụng.

4.2 Công cụ sử dụng

Ngôn ngữ lập trình :

- Extractor : Python
- Web Application : ASP.NET

Hệ quản trị cơ sở dữ liệu :

- Microsoft SQL Server

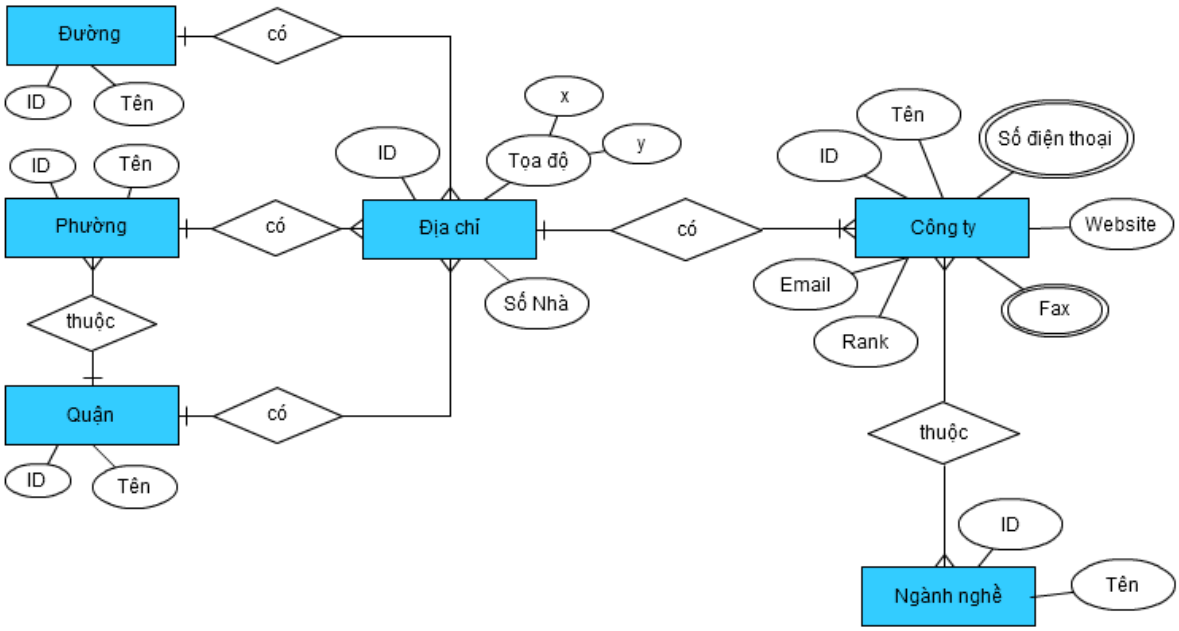
4.3 Extractor

4.3.1 Thiết kế kiến trúc

Sub-system Extractor gồm các module sau
- Matching: hiện thực giải thuật Simple Tree Matching
- Alignment: hiện thực giải thuật Partial Tree Alignment
- Wildcard: chứa các hàm support cho việc hiện thực wildcard trong pattern
- Label: các hàm dùng gán nhãn dữ liệu
- Support: các hàm phụ trợ khác

4.3.2 Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu của hệ thống được thiết kế như trong ERD dưới đây



Hình 11 ERD biểu diễn thiết kế cơ sở dữ liệu của hệ thống

Ngoài 5 bảng *tbl_ConDuong, tbl_Phuong, tbl_Quan, tbl_Duong, tbl_DiaChi*, bổ sung thêm các bảng sau :

tbl_CongTy		
column name	*type*	*key*
Id	int	PK
Ten	nvarchar(4000)	
DiaChiId	int	FK(tlb_DiaChi)
DiaChiChinhXac	nvarchar(4000)	
Email	nvarchar(512)	
Website	nvarchar(512)	
Rank	nvarchar(512)	

tbl_NganhNghe		
column name	*type*	*key*
Id	int	PK
Ten	nvarchar(4000)	

tbl_CongTy_NganhNghe		
column name	*type*	*key*
Id	int	PK
CongTyId	int	FK(tbl_CongTy)
NganhNgheId	int	FK(tbl_NganhNghe)

tbl_CongTy_DienThoai		
column name	*type*	*key*
Id	int	PK
CongTyId	int	FK(tbl_CongTy)
SoDienThoai	nvarchar(50)	

tbl_CongTy_Fax		
column name	*type*	*key*
Id	int	PK
CongTyId	int	FK(tbl_CongTy)
SoFax	nvarchar(50)	

4.3.3 Hiện thực

Dữ liệu đầu vào của Extractor là các page chứa thông tin liên quan địa chỉ cần được rút trích (list page hoặc detail page). Extractor sẽ rút trích thông tin địa chỉ và đưa vào cơ sở dữ liệu. Quá trình này gồm các bước sau

4.3.3.1. Thu thập dữ liệu thô

Dữ liệu thô đầu vào của hệ thống là các web page từ một số website. Dữ liệu được thu thập bằng các Crawlers được viết bằng Python sử dụng thư viện Scrapy.

4.3.3.2. Tạo DOM tree

DOM tree được tạo bằng parser trong thư viện xml.dom.minidom của Python. Tuy nhiên, thực tế các page có thể thiếu sót, không đúng chuẩn html (thiếu thẻ đóng …). Cần phải có quá trình sửa lỗi html trước khi đưa vào parser. Quá trình này gồm :

- Xóa các text node rỗng (chỉ bao gồm dấu cách, tab và xuống dòng)
- Sửa lỗi html sử dụng Cleaner và Parser trong thư viện lxml

Sau khi tạo được DOM tree, nếu input là các detail page, cần tạo một node gốc giả, sau đó nối các DOM tree vào thành cây con của node gốc giả này.

4.3.3.3. Xây dựng record pattern

Record pattern được xây dựng bằng hàm TraverseAndMatch dựa trên giải thuật NET. Pattern là một DOM tree mà mỗi node của nó có một thuộc tính đặc biệt là wildcard. Wildcard biểu diễn tần suất xuất hiện của cây con mà node chứa wildcard là gốc trong pattern, gồm các dạng sau:

- (+) : xuất hiện một hoặc nhiều lần.
- (?) : xuất hiện một lần, có thể không xuất hiện.
- (.) : xuất hiện một lần, bắt buộc phải có.

Hàm GenRecordPattern tạo ra pattern từ danh sách các cây con trùng khớp theo giả sử: Item đầu tiên trong các data record bắt buộc phải xuất hiện. Điều này cho phép đơn giản hóa quá trình tạo pattern.

4.3.3.4. Rút trích dữ liệu theo record pattern

Dữ liệu được rút trích bằng cách duyệt theo pattern và DOM tree của các page có chứa thông tin. Wildcard trong từng node của pattern cho phép xác định tần suất xuất hiện của node đó trong DOM tree. Ứng với từng loại wildcard, ta có các ý nghĩa sau:

- (+) : data region chứa một hoặc nhiều data record.
- (?) : item có thể xuất hiện hoặc không trong data record.
- (.) : item bắt buộc xuất hiện trong data record

4.3.3.5. Gán nhãn dữ liệu

Như phần phân tích đã nên rõ, mục tiêu của extractor là rút trích 7 thuộc tính của một đối tượng bao gồm :

- Tên cơ sở.
- Địa chỉ chính xác của cơ sở
- E-mail
- Website
- Số điện thoại
- Số fax
- Ngành nghề kinh doanh

Tuy nhiên data record rút trích theo giải thuật NET chưa được gán nhãn dữ liệu, do đó cần thiết phải nhận diện từng item trong data record để xác định nhãn của chúng. Việc nhận diện được thực hiện dựa trên các đặc điểm của 7 thuộc tính trên, gồm có:

- Các đặc điểm của cấu trúc chuỗi
    - Số lượng từ viết hoa chữ đầu.

- Số lượng dấu phẩy.
- Sự xuất hiện của các từ chỉ thị (VD: công ty, cơ sở …)
- Số lượng chữ số.
- Số lượng chữ số liên tiếp lớn nhất.

- Các đặc điểm của cấu trúc cây:
  - Text nằm trong thẻ anchor (<a>)

4.3.3.6. Khớp dữ liệu với địa chỉ trong cơ sở dữ liệu

Khớp dữ liệu vừa rút trích được với dữ liệu địa chỉ đã có được thực hiện qua 2 bước:

- Phân tích địa chỉ chính xác để xác định số nhà, đường, phường, quận: thực hiện dựa trên thống kê các dạng biểu diễn của địa chỉ chính xác, sau đó phân tích dựa theo các dạng này.

  VD: một số dạng biểu diễn của địa chỉ

  286 Lý Thường Kiệt, P.14, Q.10
  286, Lý Thường Kiệt, P.14, Q.10
  286 Lý Thường Kiệt, Phường 14, Quận 10
  Kios 100, 142 Tô Hiến Thành, Q.10
- Dựa trên số nhà, đường, phường, quận đã phân tích, xác định địa chỉ đã có trong bảng tbl_DiaChi, insert data record vào các bảng cần thiết với DiaChiId là địa chỉ vừa xác định.

4.4 Web Application

4.4.1 Cơ sở dữ liệu

Phần website sử dụng cơ sở dữ liệu phần thiết kế Extractor đã tạo, để tiện trong việc truy vấn, nhóm đã bổ sung thêm các view .
- View 1 : vw_CongTy_DienThoai

```
CREATE VIEW vw_CongTy_DienThoai
AS
SELECT i.CongTyId, STUFF(g.y, 1, 1, '') AS
SoDienThoai
FROM
      (SELECT CongTyId FROM tbl_CongTy_DienThoai
      GROUP BY CongTyId ) AS i
CROSS APPLY
      (SELECT DISTINCT',' + CAST(SoDienThoai AS
NNVARCHAR(11))
      FROM   tbl_CongTy_DienThoai AS s
      WHERE  s.CongTyId = i.CongTyId
      ORDER BY   ',' + CAST(SoDienThoai AS
NVARCHAR(11))
```

```
      FOR XML PATH('')
) AS g(y);
```

View tạo được có dạng

CongTyId	SoDienThoai
1	0838225223,0839326280
2	0838272811,0838272812,0838276027

- View 2 : vw_CongTy_Fax

```
CREATE VIEW vw_CongTy_Fax
AS
SELECT i.CongTyId, STUFF(g.y, 1, 1, '') AS SoFax
FROM
      (SELECT CongTyId FROM tbl_CongTy_Fax
      GROUP BY CongTyId ) AS i
CROSS APPLY  (
      SELECT DISTINCT ',' + CAST(SoFax AS
NVARCHAR(11))
      FROM  tbl_CongTy_Fax AS s
      WHERE  s.CongTyId = i.CongTyId
      ORDER BY ',' + CAST(SoFax AS NVARCHAR(11))
      FOR XML PATH('')
) AS g(y)
```

View tạo được có dạng :

CongTyId	Sofax
1	0822172675,0838127196
2	0838218590, 0838298762, 0839600015

- View 3 : vw_CongTy_NganhNghe

```
CREATE VIEW vw_CongTy_NganhNghe
AS
SELECT     i.CongTyId, STUFF(g.y, 1, 1, '') AS
NganhNghe
FROM
      (SELECT    m.CongTyId
      FROM  (SELECT CongTyId,Ten
              FROM tbl_CongTy_NganhNghe ctnn
```

```
            LEFT JOIN tbl_NganhNghe nn
            ON ctnn.NganhNgheId = nn.Id) as m
      GROUP BY CongTyId ) AS i
CROSS APPLY      (
      SELECT DISTINCT ',' + CAST(Ten AS
NVARCHAR(11))
      FROM   (SELECT CongTyId,Ten
              FROM tbl_CongTy_NganhNghe ctnn
              LEFT JOIN tbl_NganhNghe nn
              ON ctnn.NganhNgheId = nn.Id) AS s
      WHERE s.CongTyId = i.CongTyId
      ORDER BY   ',' + CAST(Ten AS NVARCHAR(11))
      FOR XML PATH('')
) AS g(y)
```

View tạo được có dạng :

CongTyId	NganhNghe
1	Giày Dép,Mua Sắm
2	Dịch Vụ,Ngành Nước

- View 4 : vw_DiaChi

```
CREATE VIEW vw_DiaChi
AS
SELECT dbo.tbl_DiaChi.Id, dbo.tbl_DiaChi.SoNha,
       dbo.tbl_ConDuong.TenConDuong AS Duong,
       dbo.tbl_ConDuong.TenKhongDau AS
DuongKhongDau,
       dbo.tbl_Phuong.TenPhuong AS Phuong,
       dbo.tbl_Quan.TenQuan AS Quan,
       dbo.tbl_Quan.TenKhongDau AS QuanKhongDau,
       dbo.tbl_DiaChi.X, dbo.tbl_DiaChi.Y
FROM  dbo.tbl_DiaChi LEFT JOIN dbo.tbl_Phuong
    ON dbo.tbl_DiaChi.PhuongId = dbo.tbl_Phuong.Id
    INNER JOIN dbo.tbl_Quan
    ON dbo.tbl_DiaChi.QuanId = dbo.tbl_Quan.Id
    INNER JOIN dbo.tbl_ConDuong
    ON dbo.tbl_DiaChi.DuongId = dbo.tbl_ConDuong.Id
```

- View 5 : vw_CongTy

```
CREATE VIEW vw_CongTy
AS
SELECT ct.Id, Ten, TenKhongDau, DiaChiChinhXac,
SoDienThoai, Email, SoFax, Website, Rank, SoNha,
Duong, Phuong, Quan, DuongKhongDau, QuanKhongDau,
NganhNghe, X, Y
FROM   tbl_CongTy ct
       LEFT JOIN vw_DiaChi dc
       ON ct.DiaChiId = dc.Id
       LEFT JOIN vw_CongTy_DienThoai dt
       ON ct.Id = dt.CongTyId
       LEFT JOIN vw_CongTy_Fax sf
       ON ct.Id = sf.CongTyId
       LEFT JOIN vw_CongTy_NganhNghe nn
       ON ct.Id = nn.CongTyId
```

View này là tổng hợp từ các view trên và phục vụ cho chức năng search .

4.4.2 Hiện thực

Website Location Finder được xây dựng với 2 chức năng chính : Tìm địa chỉ và tìm công ty

- Chức năng 1 : Tìm địa chỉ
Để thực hiện chức năng tìm địa chỉ ta cần nhập vào textbox search theo cú pháp : Số Nhà , Tên đường

- Chức năng 2 : Tìm công ty
Để thực hiện chức năng tìm công ty ta chỉ cần nhập vào textbox search tên công ty cần tìm

Khi chuỗi query địa chỉ (Ví dụ : 133, 3 Thang 2) hay tên công ty (Ví dụ : Cửa hàng) được nhập vào hệ thống sẽ tìm kiếm các cột DuongKhongDau , Ten, TenKhongDau trong view vw_CongTy với chuỗi điều kiện như sau:

```
WHERE DuongKhongDau LIKE N'%133 3 Thang 2%'
      OR Ten LIKE N'%133 3 Thang 2%
      OR TenKhongDau LIKE N'%133 3 Thang 2%
```

Nếu chuỗi nhập vào có ký tự đầu tiên thuộc dạng số , ta sẽ tìm thêm trên cột SoNha và DuongKhongdau (cùng thoả) trên view vw_CongTy , chuỗi điều kiện sẽ như sau :

```
WHERE DuongKhongDau LIKE N'%133 3 Thang 2%'
      OR (SoNha LIKE '%133%'
          AND DuongKhongDau LIKE N'%3 Thang 2%')
      OR Ten LIKE N'%133 3 Thang 2%
      OR TenKhongDau LIKE N'%133 3 Thang 2%
```

4.5 Sử dụng

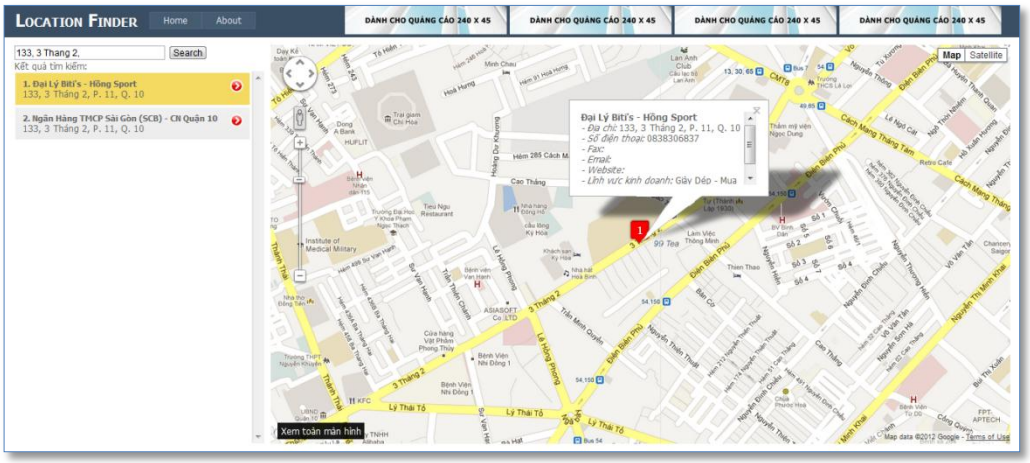
4.5.1 Extractor

Extractor có thể được điều chỉnh bằng các thông số cấu hình sau, các thông số này đặt trong module settings.

- DATA_PATH : đường dẫn đến thư mục chứa các page dữ liệu cần rút trích.
- MODE : 1 : extract từ các list page
  2 : extract từ các detail page
- SIMILARITY_DEGREE : thông số τ trong giải thuật NET, đặc tả ngưỡng xác định 2 cây là giống nhau về cấu trúc

4.5.2 Website :

4.5.2.1. Tìm địa chỉ



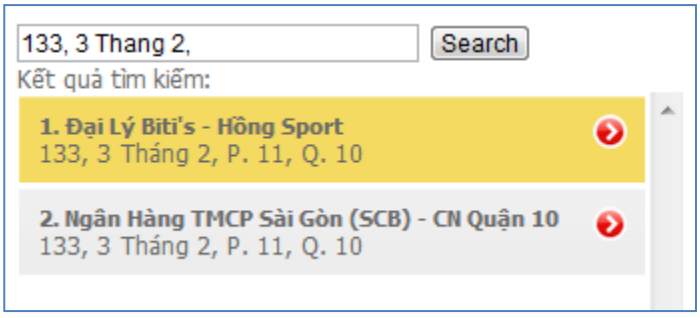
Hình 12 Giao diện tìm địa chỉ

Đầu tiên ta nhập vào ô Search 

Click button Search , kết quả trả về sẽ gồm 1 danh sách thoả điều kiện của chuỗi

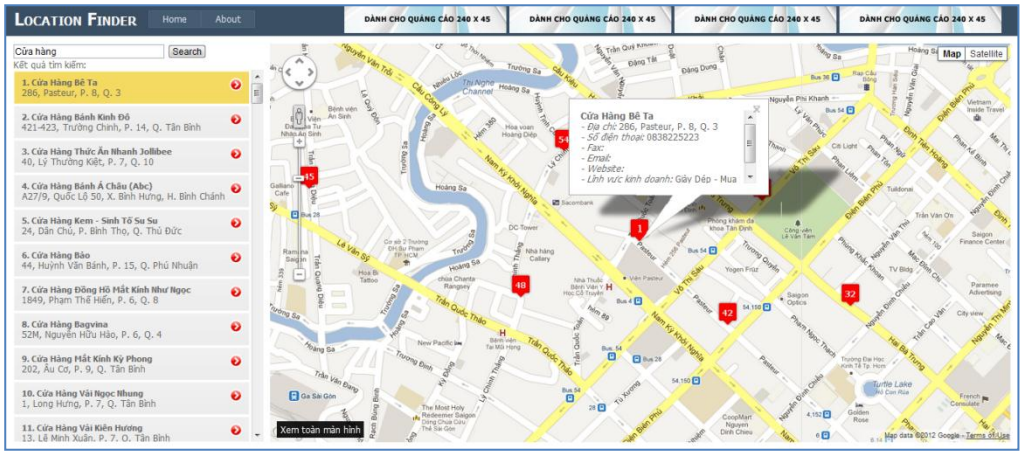
Và vị trí của các địa chỉ được hiển thị trên bản đồ

Khi ta chọn 1 kết quả trong danh sách bên trái



thì thông tin chi tiết sẽ được hiển thị rõ trên bản đồ bên phải

4.5.2.2. Tìm địa danh

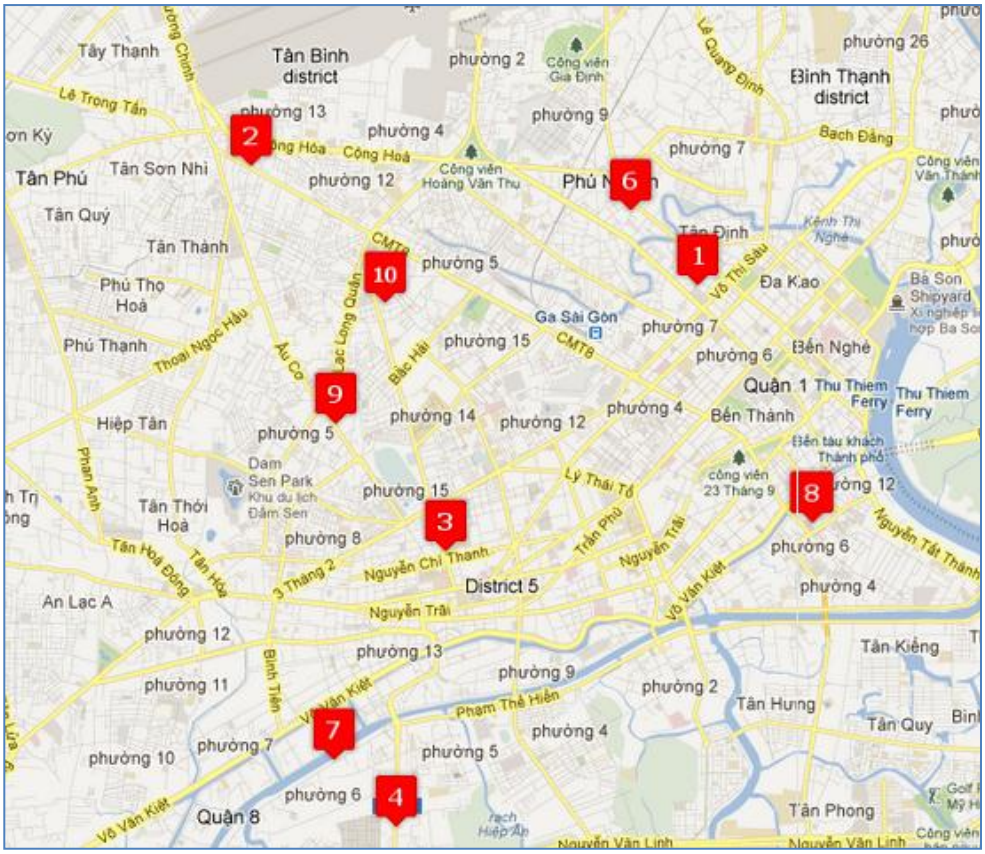


Hình 13 Giao diện tìm kiếm địa danh

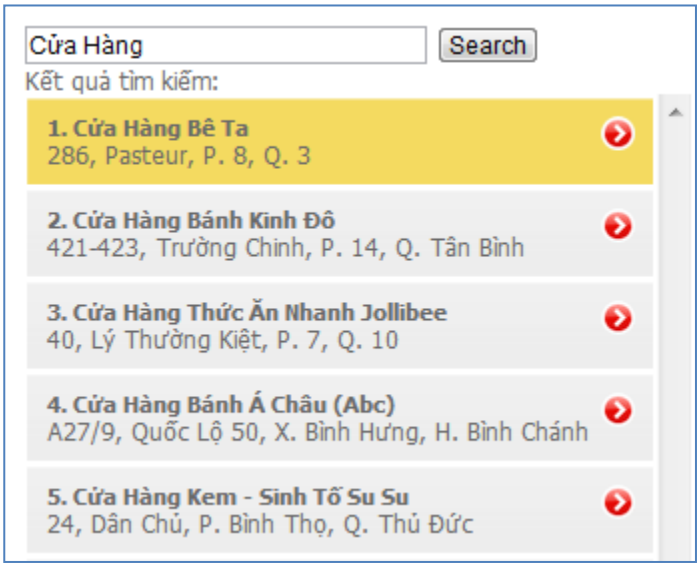
Tương tự như tìm địa chỉ , đầu tiên ta nhập vào ô Search

Click button Search , kết quả trả về sẽ gồm 1 danh sách thoả điều kiện của chuỗi

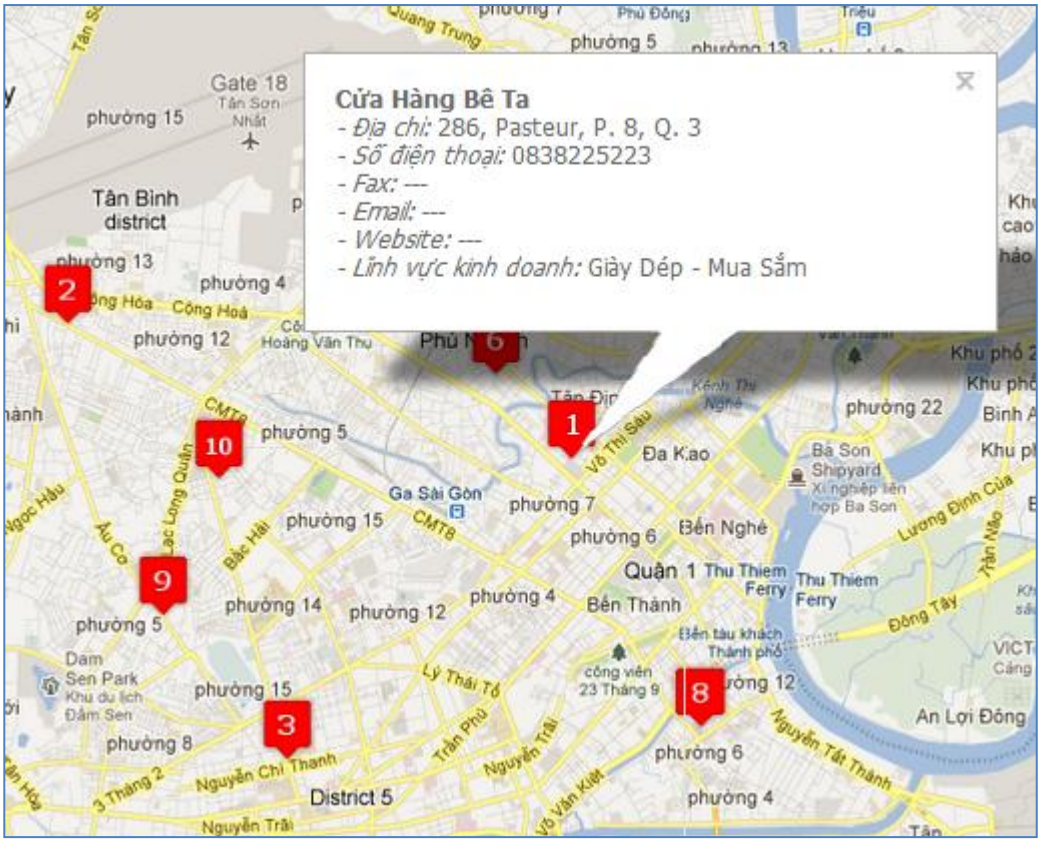
Và vị trí của các địa chỉ được hiển thị trên bản đồ



Khi ta chọn 1 kết quả trong danh sách bên trái



thì thông tin chi tiết sẽ được hiển thị rõ trên bản đồ bên phải


Cửa Hàng Bê Ta
- Địa chỉ: 286, Pasteur, P. 8, Q. 3
- Số điện thoại: 0838225223
- Fax: ---
- Email: ---
- Website: ---
- Lĩnh vực kinh doanh: Giày Dép - Mua Sắm
Gate 18
Tân Sơn Nhất
phường 15
Tân Bình district
phường 13
phường 4
Cộng Hoà
phường 12
Hoàng Văn Thụ
phường 5
Ga Sài Gòn
CMT8
phường 7
Đa Kao
Bến Nghé
phường 6
Quận 1
Thu Thiem Ferry
Bến Thành
phường 14
phường 12
phường 4
Dam Sen Park
Khu du lịch Đầm Sen
phường 15
phường 8
District 5
Nguyễn Chí Thanh
Nguyễn Trãi
Lý Thái Tổ
Trần Phú
Võ Văn Kiệt
Nguyễn Tất Thành
An Lợi Đông
phường 22
Ba Son Shipyard
Bến tàu khách Thành phố
công viên 23 Tháng 9
Đông Tây
Lương Định Của
Bắc Hải
Âu Cơ
Ngọc Hồi
3 Tháng 2
1
2
3
8
9
10

CHƯƠNG V. TỔNG KẾT

5.1 Các kết quả đạt được

Nhóm đã xây được một hệ thống bao gồm các chức năng sau:

- Thu thập thông tin về địa chỉ từ các website về địa chỉ trên Internet.
- Trích xuất thông tin từ dạng dữ liệu thô thành dạng dữ liệu có cấu trúc và ngữ nghĩa với kết quả phân tích khá chính xác.
- Xây dựng ứng dụng tìm kiếm cơ bản từ cơ sở dữ liệu thông tin đã có, với giao diện hiển thị trực quan cho người dùng.

5.2 Các hạn chế của hệ thống

- Thu thập dữ liệu thô:
  - Thông tin về địa chỉ được thu thập từ một số website cố định, chưa phủ được thông tin vô tận trên Internet.
  - Chưa có cơ chế cập nhật nguồn dữ liệu theo thời gian, để đảm bảo CSDL luôn theo kịp với nguồn dữ liệu trên Internet.
- Extractors:
  - Hạn chế của giải thuật NET:
    - Rút trích từ tất cả các vùng chứa dữ liệu trên web page, có thể chứa nhiều thông tin không có giá trị.
    - So trùng cấu trúc cây có thể không hiệu quả nếu số lượng node chứa item optional là khá lớn.
  - Giao diện sử dụng chưa trực quan rõ ràng, chưa có tính tương tác với người dùng.
  - Độ tùy biến không cao.
  - Chưa có cơ chế xử lý khi dữ liệu thu thập từ các nguồn trùng nhau.
- Web Application: chỉ mới hiện thực một ứng dụng cơ bản nhất để sử dụng nguồn thông tin về địa chỉ đã có.

5.3 Hướng phát triển

Trong giai đoạn luận văn, nhóm sẽ hoàn thiện hơn Extractor, xây dựng giao diện người dùng, tăng tính tùy biến, xây dựng cơ chế để phần nào khắc phục hạn chế của giải thuật NET. Đồng thời, search engine trong web application sẽ được hoàn thiện hơn.

Xa hơn nữa, web application có thể được nâng cấp tính năng theo một số hướng như sau:

(1) *Tăng cường chất lượng của thông tin địa chỉ hiện có dựa trên sự cộng tác của người dùng*. Bằng cách phát triển hệ thống cho phép người dùng comment, đánh giá(rating), bổ sung thông tin như ảnh, tọa độ địa lý..

(2) *Cho phép người dùng tạo ra địa chỉ về công ty của họ, quản lý và cập nhật dữ liệu này.* Website có thể trở thành kênh quảng bá cho công ty của họ, với hình thức danh thiếp điện tử chẳng hạn.

(3) *Xây dựng tính xã hội ảo(social) cho hệ thống*: người dùng có thể bookmark lại các địa chỉ ưa thích của mình, và chia sẻ với bạn bè trên hệ thống hoặc trên các mạng xã hội.

Dựa vào đó, ứng dụng có thể phát triển thành:

(1) Mạng xã hội về địa điểm.
(2) Website tìm kiếm thông tin địa chỉ kết hợp tìm đường đi.
(3) Hệ thống danh thiếp điện tử về địa chỉ.

TÀI LIỆU THAM KHẢO

[1] Liu, B. Structured Data Extraction: Wrapper Generation. *Web Data Mining*, 2009, 9: p. 363 – 418.

[2] Liu, B. and Y. Zhai. NET - A System for Extracting Web Data from Flat and Nested Data Records. In *Proceedings of Intl. Conf. on Web Information Systems Engineering (WISE2005)*, 2005.

[3] Zhai, Y. and B. Liu. Structured data extraction from the web based on partial tree alignment. *IEEE Transactions on knowledge and Data Engineering*, 2006: p. 1614-1628.

[4] Zhai, Y. and B. Liu. Web data extraction based on partial tree alignment. In *Proceedings of International conference on World Wide Web (WWW-2005)*, 2005.