Automated Extraction of Tree Adjoining Grammars from a Treebank for Vietnamese

Phuong Le-Hong, Thi Minh Huyen Nguyen, Phuong Thai Nguyen, Thi Ha Phan

Trích rút tự động văn phạm LTAG cho tiếng Việt

Lê Hồng Phương[1], Nguyễn Thị Minh Huyền[1], Nguyễn Phương Thái[1], Phan Thị Hà[2]
[1] Đại học Quốc gia Hà Nội, [2] Học viện Công nghệ Bưu chính Viễn thông

Tóm tắt nội dung

Bài báo này giới thiệu hệ văn phạm kết nối cây LTAG (Lexicalized Tree Adjoining Grammars – LTAG) và các thuật toán trích rút tự động LTAG từ kho văn bản gán nhãn cú pháp (treebank). Chúng tôi trình bày kết quả trích rút một văn phạm LTAG cho tiếng Việt. Chương trình trích rút tự động các văn phạm LTAG độc lập với ngôn ngữ và được phân phối dưới dạng mã nguồn mở.

Từ khoá: trích rút, LTAG, treebank, tiếng Việt.

Abstract

In this paper, we present a system that automatically extracts lexicalized tree adjoining grammars (LTAG) from treebanks. We first discuss in detail extraction algorithms and compare them to previous works. We then report the first LTAG extraction result for Vietnamese, using a recently released Vietnamese treebank. The implementation of an open source and language independent system for automatic extraction of LTAG grammars is also discussed.

Keywords: extraction, LTAG, treebank, Vietnamese.

1 Giới thiệu

Phân tích cú pháp là bước quan trọng trong quá trính xử lý ngôn ngữ tự nhiên. Các bộ phân tích cú pháp chất lượng cao giúp tăng tính hiệu quả của các hệ thống xử lí ngôn ngữ tự nhiên như dịch máy, tóm tắt văn bản, các hệ hỏi đáp...

Mọi bộ phân tích cú pháp đều cần một bộ luật cú pháp, hay văn phạm, được biểu diễn bởi một hệ văn phạm hình thức cụ thể nào đó. Việc xây dựng văn phạm thủ công là công việc tốn rất nhiều thời gian và công sức, chính vì vậy nhiều phương pháp tự động hoặc bán tự động để xây dựng văn phạm đã được nghiên cứu trong thời gian qua. Hầu hết các kết quả nghiên cứu về xây dựng văn phạm cho các hệ thống xử lí ngôn ngữ tự nhiên đã được công bố đều tập trung vào các ngôn ngữ phổ dụng như tiếng Anh, các thứ tiếng Ấn-Âu và tiếng Hoa.

Nhìn chung, có hai phương pháp chính để xây dựng tự động văn phạm. Phương pháp thứ nhất sử dụng một hệ thống mô tả văn phạm bậc cao để sinh văn phạm. Các hệ thống như vậy được gọi là các siêu văn phạm (meta-grammar) [10, 20]. Phương pháp thứ hai là phương pháp trích rút tự động văn phạm từ các kho văn bản có chú giải cú pháp (treebank). Ở đây, chúng tôi quan tâm tới phương pháp thứ hai.

Trong bài báo này, chúng tôi trình bày các thuật toán trích rút tự động văn phạm LTAG từ treebank. Chúng tôi phát triển một chương trình để tự động trích rút văn phạm LTAG cho tiếng Việt từ kho văn bản VietTreebank và đánh giá kết quả thu được.

Cấu trúc của bài báo như sau. Trước hết, mục 2 giới thiệu sơ lược hệ văn phạm LTAG. Mục 3 điểm lại một số công trình đã có về trích rút văn phạm từ treebank. Mục 4 nêu khái quát về treebank tiếng Việt. Tiếp theo, mục 5 trình bày chi tiết thuật toán trích rút văn phạm LTAG từ treebank mà chúng tôi sử dụng và so sánh nó với một thuật toán tương tự. Mục 6 trình bày kết quả và chương trình phần mềm trích rút tự động văn phạm LTAG cho tiếng Việt. Cuối cùng là phần kết luận và hướng phát triển tiếp theo.

2 Hệ văn phạm LTAG

Văn phạm kết nối cây (*Tree Adjoining Grammars*–TAG) là hệ văn phạm hình thức được phát minh bởi Aravind Joshi [17, 18] và các đồng nghiệp. Khác với hệ văn phạm phi ngữ cảnh sử dụng các luật viết lại

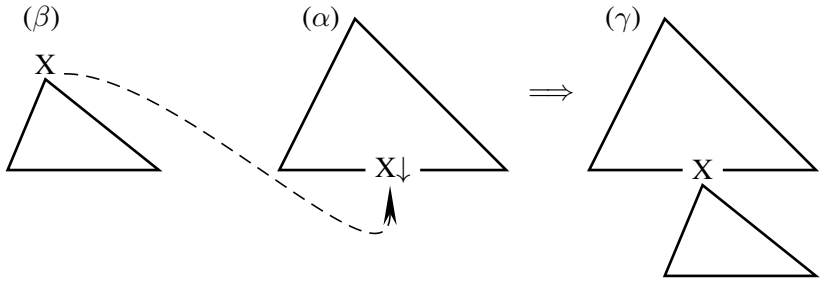
dạng xâu, hệ văn phạm kết nối cây sử dụng các luật viết lại dạng cây. Văn phạm TAG đã được nghiên cứu kỹ về mặt hình thức và khả năng ứng dụng trong việc phân tích nhiều ngôn ngữ tự nhiên khác nhau, ví dụ cho tiếng Anh [5, 13, 30], tiếng Pháp [1, 12, 27], tiếng Đức [19], tiếng Hoa [28]. Trong mục này, chúng tôi giới thiệu khái quát (một cách không hình thức) hệ văn phạm LTAG. Chi tiết về văn phạm LTAG đã được trình bày kỹ lưỡng trong nhiều tài liệu tham khảo khác nhau, ví dụ trong tài liệu [18].

2.1 Các cây cơ bản

Phần tử cơ sở của một văn phạm TAG là các cây cơ bản. Nếu mỗi cây cơ bản đều chứa ít nhất một nút lá có nhãn là kí hiệu kết (nút từ vựng) thì văn phạm được gọi là LTAG (Lexicalized TAG). Có hai kiểu cây cơ bản là *cây khởi tạo* và *cây phụ trợ*. Cây khởi tạo chứa các nút được gán kí hiệu không kết, riêng các nút lá có thể được gán kí hiệu kết. Nếu các nút lá có kí hiệu không kết thì chúng được đánh dấu bằng ký hiệu thay thế $\downarrow$. Cây phụ trợ được đặc trưng bởi hai nút có ký hiệu không kết giống nhau, trong đó có một nút gốc và một nút lá. Nút lá đặc biệt này được gọi là nút chân và được đánh dấu bằng ký hiệu $*$.

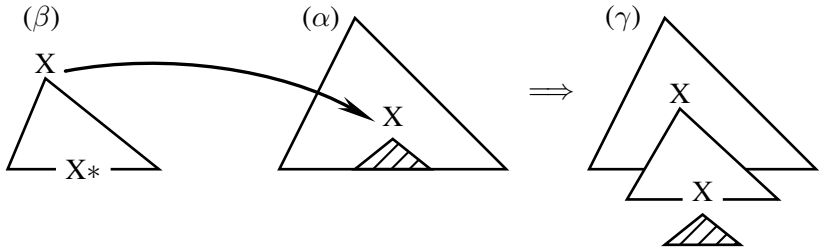
2.2 Hai thao tác viết lại

Các cây cơ bản của văn phạm LTAG được kết hợp với nhau bằng hai thao tác viết lại là *thay thế* và *kết nối*. Thao tác thay thế thực hiện phép thế một nút lá có nhãn X của một cây α bởi một cây β có gốc cũng có nhãn là X. Thao tác thay thế được minh hoạ bởi Hình 1.



Hình 1: Phép thay thế

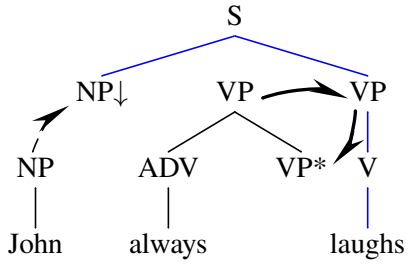
Thao tác kết nối thực hiện phép chèn một cây phụ trợ vào bên trong một cây khác. Như minh hoạ trong Hình 2, cây phụ trợ β có gốc và nút chân có cùng nhãn X được chèn vào trong cây α tại nút cũng có nhãn X, sinh ra cây γ. Chú ý rằng thao tác kết nối không được thực hiện tại các nút được đánh dấu là nút thay thế của α.



Hình 2: Phép kết nối

2.3 Cây phân tích và cây dẫn xuất

Các cây trung gian sinh ra khi áp dụng các phép thế và kết nối được gọi là các *cây phân tích*. Cây phân tích *đầy đủ* là cây phân tích trong đó mọi nút lá đều được gán nhãn kết. Như vậy, việc phân tích cú pháp của một câu là việc xuất phát từ một cây cơ bản có gốc là tiên đề, tìm một cây phân tích đầy đủ có các nút lá tương ứng với dãy các từ trong câu.

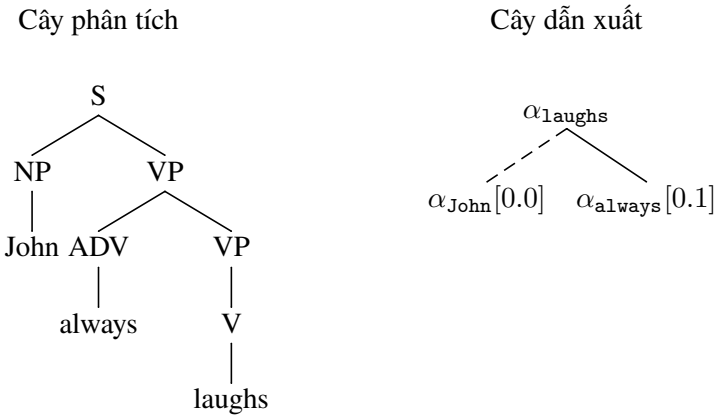


Hình 3: Dẫn xuất của câu *John always laughs.*

Hình 3 minh hoạ ví dụ về dẫn xuất cú pháp của câu "*John always laughs*". Nếu ta kí hiệu $\alpha_{\texttt{John}}$, $\alpha_{\texttt{always}}$ và $\alpha_{\texttt{laughs}}$ tương ứng là các cây gắn với các từ John, always và laughs thì dẫn xuất này sử dụng hai quy tắc viết lại của hệ hình thức LTAG như sau:

- Cây $\alpha_{\texttt{John}}$ được thay thế vào nút lá có nhãn NP của cây $\alpha_{\texttt{laughs}}$, sinh ra cây $\alpha'_{\texttt{laughs}}$;
- Cây phụ trợ $\alpha_{\texttt{always}}$ được kết nối vào nút VP của cây $\alpha'_{\texttt{laughs}}$, sinh ra cây dẫn xuất đầy đủ ở bên trái Hình 4.

Đối với văn phạm phi ngữ cảnh, nhìn vào cây phân tích ta biết được ngay các quy tắc viết lại đã thực hiện. Đối với văn phạm TAG, từ cây phân tích ta không thể biết cụ thể các phép viết lại đã được thực hiện để tạo nên cây đó, chính vì vậy, trong hệ hình thức LTAG, người ta cần dùng một cấu trúc đặc biệt gọi là *cây dẫn xuất* để ghi lại các thao tác tạo nên cây phân tích từ các cây cơ bản. Mỗi nút trên cây dẫn xuất là tên của một cây cơ bản, mỗi cung biểu diễn một phép kết nối (nét liền) hoặc một phép thay thế (nét đứt). Ngoài ra, mỗi nút tại đó có áp dụng thao tác viết lại được đánh dấu bằng một địa chỉ Gorn[1]. Cây dẫn xuất mô tả phân tích của câu *John always laughs* được biểu diễn ở bên phải của Hình 4.



Hình 4: Cây phân tích và cây dẫn xuất của câu *John always laughs*.

Khi xây dựng văn phạm LTAG cho một ngôn ngữ tự nhiên, người ta áp dụng một số nguyên lý ngôn ngữ học sau. Thứ nhất, văn phạm TAG được từ vựng hóa: mỗi cây cơ bản đều có ít nhất một nút lá gắn với một đơn vị từ vựng gọi là từ neo. Thứ hai, mỗi cây khởi tạo của LTAG biểu diễn các thành phần chiếu của một từ neo, hay nói cách khác là các thành phần đối bổ nghĩa cho từ neo. Thứ ba, các cây cơ bản là cực tiểu: cây khởi tạo phải có từ neo là từ trung tâm của một thành phần chính trong câu và chứa tất cả các thành phần đối bắt buộc của từ neo [14]. Tất cả các thành phần phụ của từ neo có thể thêm vào một cách đệ quy bằng cách sử dụng phép kết nối với các cây phụ trợ.

[1]Địa chỉ Gorn được định nghĩa đệ quy như sau: nút gốc có địa chỉ 0, nút con thứ k của một nút có địa chỉ j có địa chỉ là $j.k$.

Như vậy, khi xây dựng câu, các phép thế tương ứng với việc gắn các đối vào vị từ, phép kết nối tương ứng với việc thêm các thành phần phụ. Vì thế, cây dẫn xuất biểu diễn quan hệ phụ thuộc ngữ nghĩa giữa các từ trong câu. Đây là lý do hầu hết các tiếp cận tới ngữ nghĩa trong văn phạm LTAG sử dụng cây dẫn xuất như là giao diện giữa cú pháp và ngữ nghĩa.

LTAG thuộc lớp các văn phạm cảm ngữ cảnh yếu (*mildly context-sensitive grammar*), tức là có khả năng sinh mạnh hơn các văn phạm phi ngữ cảnh, trong khi độ phức tạp thời gian của bộ phân tích cú pháp LTAG vẫn là đa thức ($O(n^6)$). Văn phạm hình thức LTAG rất phù hợp với các ứng dụng ngôn ngữ học. Người ta đã chỉ ra rằng các tính chất của văn phạm LTAG cho phép mô tả các hiện tượng cú pháp một cách tự nhiên. Hơn nữa, khả năng chuyển đổi một văn phạm LTAG sang các hệ hình thức văn phạm hợp nhất khác như LFG (Lexical Functional Grammar) hay HPSG (Head-driven Phrase Structure Grammar) đã được chứng minh [31]. Vì các lí do trên, chúng tôi chọn hệ hình thức LTAG để mô hình hóa văn phạm tiếng Việt. Một mặt chúng tôi điều chỉnh một bộ phân tích cú pháp LTAG tổng quát cho phù hợp với tiếng Việt, mặt khác chúng tôi cố gắng xây dựng một kho ngữ liệu có thể tái sử dụng cho các ứng dụng liên quan đến phân tích cú pháp tiếng Việt cũng như việc đánh giá các công cụ phân tích cú pháp.

Trong các mục tiếp theo, chúng tôi điểm qua các công trình đã có về trích rút tự động văn phạm từ treebank và trình bày thuật toán trích rút LTAG mà chúng tôi sử dụng cho tiếng Việt.

3 Trích rút tự động văn phạm

Có khá nhiều công trình về trích rút tự động văn phạm từ treebank đã được công bố, tất cả các công trình này đều được thực hiện cho các ngôn ngữ thông dụng [21]. Xia phát triển phương pháp trích rút văn phạm tổng quát và áp dụng cho tiếng Anh, tiếng Trung và tiếng Hàn [28, 29]. Chiang đã phát triển một hệ thống trích chọn văn phạm LTAG từ Penn Treebank tiếng Anh và dùng trong phân tích cú pháp thống kê với LTAG [8]. Chen đã trích TAG từ Penn Treebank tiếng Anh [6, 7]. Một số công trình sau đó ứng dụng phương pháp của Chen để trích rút văn phạm cho một số ngôn ngữ khác, như các công trình của Johansen [16] và Nasr [23] cho tiếng Pháp, của Habash cho tiếng Ả-rập [15]. Neumann trích văn phạm cho tiếng Anh từ Penn Treebank tiếng Anh và cho tiếng Đức từ NEGRA Treebank [24]. Bäcker trích rút văn phạm LTAG cho tiếng Đức từ NEGRA Treebank [3]. Park trích rút văn phạm LTAG cho tiếng Hàn từ Sejong Treebank [26].

4 Treebank tiếng Việt

Trong khuôn khổ đề tài KC01.01/06-10, nhóm các chuyên gia ngôn ngữ học đã thực hiện việc chú giải thông tin cú pháp cho một kho văn bản tiếng Việt (VietTreebank). Dữ liệu văn bản được thu thập từ chuyên mục Chính trị - Xã hội của báo Tuổi trẻ Online.

Kho văn bản được chia làm ba tập tương ứng với ba mức gán nhãn là tách từ, gán nhãn từ loại và gán nhãn cú pháp. Tập được gán nhãn cú pháp là tập con của tập được gán nhãn từ loại; tập được gán nhãn từ loại là tập con của tập được tách từ. Tập gán nhãn cú pháp gồm 10471 câu (225085 đơn vị từ vựng). Độ dài của các câu nằm trong khoảng từ 2 tới 105 từ, với độ dài trung bình là 21.75 từ. Có 9314 câu (chiếm 88.95%) có độ dài không lớn hơn 40 từ. Tập nhãn của treebank gồm 38 nhãn cú pháp (18 nhãn từ loại, 17 nhãn cụm từ, 3 nhãn phần tử rỗng) và 17 nhãn chức năng. Các cây cú pháp có chiều cao đa số nằm trong khoảng từ 5 đến 10, phổ biến nhất là bằng 7 (1436 câu). Đặc biệt có 2 câu có chiều cao bằng 27. Các thông tin chi tiết hơn về treebank tiếng Việt được trình bày trong tài liệu [25].

Một chú ý nhỏ là VietTreebank không phân biệt các liên từ đẳng lập và liên từ chính phụ, tất cả các liên từ đều được gán nhãn C. Do việc phân biệt giữa các loại liên từ này là cần thiết khi xây dựng văn phạm LTAG nên chúng tôi xử lý bằng cách thay thế các liên từ đẳng lập trong treebank (“và”, “hoặc”, “&”) bằng nhãn CC. Một số nhãn cú pháp được sử dụng trong các ví dụ của bài báo này được liệt kê trong Bảng 1.

No.	Nhãn	Mô tả
1.	S	câu trần thuật
2.	VP	cụm động từ
3.	NP	cụm danh từ
4.	PP	cụm giới từ
5.	N	danh từ chung
6.	V	động từ
7.	P	đại từ
8.	R	phó từ
9.	E	giới từ
10.	CC	liên từ đẳng lập

Bảng 1: Các nhãn cú pháp được sử dụng trong ví dụ.

5 Thuật toán trích rút LTAG từ treebank

Về cơ bản, quá trình trích rút tự động văn phạm LTAG từ treebank gồm ba bước. Thứ nhất, chuyển các cây cú pháp của treebank thành các cây phân tích của hệ hình thức LTAG. Thứ hai, phân rã các cây phân tích thu được ở bước một thành các cây cơ bản tương ứng với ba mẫu cây được xác định trước. Cuối cùng, sử dụng tri thức ngôn ngữ để lọc bỏ các cây cơ bản không hợp lệ.

Trong các mục tiếp theo, chúng tôi trình bày chi tiết các thuật toán mà chúng tôi phát triển trong ba bước này và so sánh chúng với thuật toán tương tự của Xia [28].

5.1 Xây dựng cây phân tích LTAG

Các cây cú pháp của VietTreebank được mã dưới dạng đặt ngoặc truyền thống. Ở mỗi cụm không có sự phân biệt rõ ràng giữa thành phần trung tâm, thành phần đối bắt buộc và thành phần phụ trợ như trong cấu trúc cây phân tích của hệ hình thức LTAG. Vì vậy, trước tiên ta cần chuyển đổi từ cây cú pháp gốc thành cây phân tích LTAG tương ứng.

Trong bước này, trước tiên ta cần phân mỗi nút của cây cú pháp thành ba loại là nút trung tâm, nút đối và nút phụ. Sau đó ta chèn thêm các nút trung gian vào cây sao cho tại mỗi mức của cây, quan hệ giữa các nút là một trong ba quan hệ sau [28]:

- *quan hệ vị từ–đối*: có một hoặc nhiều nút, một nút là trung tâm, các nút còn lại là các đối của trung tâm;
- *quan hệ phụ trợ*: có đúng hai nút, một nút phụ trợ cho nút kia;
- *quan hệ đẳng lập*: có đúng ba nút, hai nút trái và phải được liên kết với nhau bằng nút liên từ ở giữa.

Chúng tôi đã xây dựng *bảng thành phần trung tâm* [9, 22] cho VietTreebank. Bảng này được sử dụng để chọn nút con trung tâm của một nút bất kì. Chúng tôi cũng xây dựng *bảng đối* để xác định kiểu đối của một thành phần trung tâm. Bảng này được sử dụng để xác định tính chất đối hay phụ trợ của một nút anh em cho thành phần trung tâm dựa trên nhãn của trung tâm và vị trí của các nút này.

Vì cấu trúc của các cụm đẳng lập khác với các cấu trúc đối và phụ trợ nên trước tiên chúng tôi xử lí toàn bộ các cụm đẳng lập của mỗi cây bằng Thuật toán 1. Sau đó xây dựng cây phân tích đầy đủ từ cây thu được bằng Thuật toán 2.

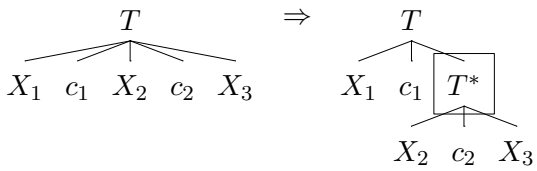
Hình 5 minh hoạ một cây có các cấu trúc liên từ trước và sau khi được xử lí bởi Thuật toán 1, ở đây c_i là các liên từ đẳng lập và X_i là các cụm đẳng lập. Hình 6 minh hoạ việc triển khai Thuật toán 2 trong đó A_i là các đối của thành phần trung tâm H của cây T và M_i là các thành phần phụ trợ cho H.

Hai thuật toán này sử dụng hàm INSERT-NODE$(T, \mathcal{L})$ trong Thuật toán 3 để chèn các nút trung gian vào giữa một nút T và danh sách $\mathcal{L}$ các nút con của T. Nút mới này là nút con của T, có cùng nhãn với

Thuật toán 1 PROCESS-CONJ(T)

Input: Một cây cú pháp T
Output: Cây T với các cụm liên từ được xử lí

for $K \in T.\texttt{kids}$ **do**
  if IS-PHRASAL(K) **then**
    $K \leftarrow$ PROCESS-CONJ(K);
  end if
end for
$(\mathcal{C}_1, \ldots, \mathcal{C}_k) \leftarrow$ CONJ-GROUPS($T.\texttt{kids}$);
for $i = 1$ to k **do**
  if $\|\mathcal{C}_i\| > 1$ **then**
    INSERT-NODE($T, \mathcal{C}_i$);
  end if
end for
if $k > 2$ **then**
  for $i = k$ **downto** 3 **do**
    $\mathcal{L} \leftarrow \mathcal{C}_{i-1} \cup c_{i-1} \cup \mathcal{C}_i$;
    $T^* \leftarrow$ INSERT-NODE($T, \mathcal{L}$);
    $\mathcal{C}_{i-1} \leftarrow T^*$;
  end for
end if
return T;



Hình 5: Xử lí các cụm liên từ bằng Thuật toán 1.

T và có danh sách con là $\mathcal{L}$. Hàm CONJ-GROUPS($\mathcal{L}$) trả về k cụm thành phần $\mathcal{C}_i$ của $\mathcal{L}$ dựa trên $k-1$ liên từ $c_1, \ldots, c_{k-1}$. Hàm NEW-NODE(l) tạo một nút mới có nhãn l.

Thuật toán 2 sử dụng một số hàm phụ trợ sau. Hàm HEAD-CHILD(X) chọn nút con trung tâm của một nút X dựa trên bảng thành phần trung tâm. Bảng 4 là bảng thành phần trung tâm cho VietTreebank. Hàm IS-LEAF(X) kiểm tra tính chất lá của nút X. Hàm IS-PHRASAL(X) kiểm tra xem X có phải là một cụm hay không.[2] Các hàm ARG-NODES($H, \mathcal{L}$) và MOD-NODES($H, \mathcal{L}$) tương ứng trả về danh sách các nút đối và phụ trợ của nút H. Danh sách $\mathcal{L}$ chứa tất cả các nút anh em của nút H.

Ví dụ, Hình 7 minh hoạ cây cú pháp của câu "*Họ sẽ không chuyển hàng xuống thuyền vào ngày mai.*" trích từ VietTreebank với cấu trúc ngoặc như sau:

```
(S
  (NP (P họ))
  (VP (R sẽ) (R không) (V chuyển)
    (NP (N hàng))
    (PP (E xuống)
      (NP (N thuyền)))
    (PP-TMP (E vào)
      (NP (N ngày mai)))))
```

[2]Nút cụm là nút không phải nút lá hoặc nút từ loại, nghĩa là nó phải có ít nhất là hai nút con, hoặc có một nút con không phải nút lá.

Thuật toán 2 BUILD-DERIVED-TREE(T)

Input: Cây cú pháp T đã được xử lí các cụm liên từ
Output: Cây phân tích có gốc là T

if (not IS-PHRASAL(T)) **then**
  return T;
end if
$H \leftarrow$ HEAD-CHILD(T);
if not IS-LEAF(H) **then**
  for $K \in T.\texttt{kids}$ **do**
    $K \leftarrow$ BUILD-DERIVED-TREE(K);
  end for
  $\mathcal{A} \leftarrow$ ARG-NODES$(H, \mathcal{L})$;
  $\mathcal{M} \leftarrow$ MOD-NODES$(H, \mathcal{L})$;
  $m \leftarrow \|\mathcal{M}\|$;
  if $m > 0$ **then**
    $\mathcal{L} \leftarrow \{H\} \cup \mathcal{A}$;
    $T^* \leftarrow$ INSERT-NODE$(T, \mathcal{L})$;
  end if
  $(M_1, M_2, \ldots, M_m) \leftarrow \mathcal{M}$;
  for $i = 1$ **to** $m - 1$ **do**
    $\mathcal{L} \leftarrow \{M_i, T^*\}$;
    $T' \leftarrow$ INSERT-NODE$(T, \mathcal{L})$;
    $T^* \leftarrow T'$;
  end for
end if
return T;

Thuật toán 3 INSERT-NODE$(T, \mathcal{L})$

Input: Cây T và danh sách các nút con $\mathcal{L}$
Output: Nút mới T^* là con của T và là cha của $\mathcal{L}$

$T^* \leftarrow$ NEW-NODE$(T.\texttt{label})$;
$T^*.\texttt{kids} \leftarrow \mathcal{L}$;
$T.\texttt{kids} \leftarrow T.\texttt{kids} \setminus \mathcal{L}$;
$T.\texttt{kids} \leftarrow T.\texttt{kids} \cup \{T^*\}$;
return T^*;

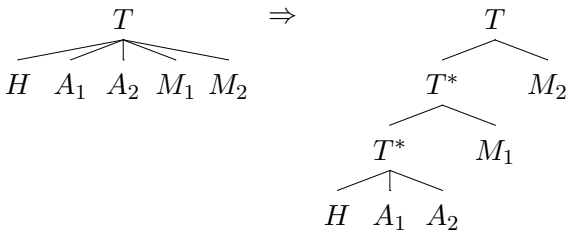
Các nút con trung tâm của các cụm được khoanh tròn. Cây phân tích của câu ví dụ sinh bởi Thuật toán 2 được cho trên Hình 8, các nút trung gian chèn thêm là các nút được đóng khung.

5.2 Trích rút các cây cơ bản

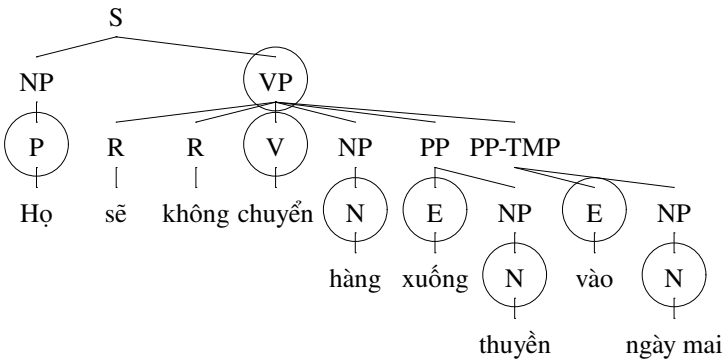
Trong bước này, mỗi cây phân tích được phân rã thành một tập các cây cơ bản. Các cấu trúc đệ quy của cây phân tích được tách ra thành các cây phụ trợ, các cấu trúc không đệ quy còn lại được tách thành các cây khởi tạo. Các cây cơ bản được trích rút ra đều thuộc một trong ba mẫu tương ứng với ba kiểu quan hệ của nút neo với các nút khác. Ba mẫu này được minh hoạ trong Hình 9.

Quá trình trích rút thực hiện việc chép các nút của cây phân tích để xây dựng các cây cơ sở. Kết quả trích rút gồm ba tập cây: tập $\mathcal{S}$ chứa các cây spine (các cây khởi tạo), $\mathcal{M}$ chứa các cây phụ trợ và $\mathcal{C}$ chứa các cây đẳng lập.

Để xây dựng các cơ bản từ một cây phân tích T, trước tiên ta tìm đường đi trung tâm $\{H_0, H_1, \ldots, H_n\}$ của T bằng thủ tục HEAD-PATH(T). Đường đi trung tâm xuất phát từ T là đường đi duy nhất từ T tới một nút lá trong đó mỗi nút trừ T đều là nút con trung tâm của nút cha. Ở đây $H_0 \equiv T$ và H_j là cha của nút



Hình 6: Ví dụ minh hoạ việc xây dựng cây phân tích



Hình 7: Một cây cú pháp

con trung tâm H_{j+1}. Với mỗi nút cha P và nút con trung tâm H, ta lấy danh sách $\mathcal{L}$ các nút anh em của H và xác định quan hệ giữa H và $\mathcal{L}$. Nếu đó là quan hệ đẳng lập thì trích ra một cây đẳng lập; nếu đó là quan hệ phụ trợ thì trích ra một cây phụ trợ, nếu là quan hệ vị từ–đối thì trích ra một cây khởi tạo. Thuật toán 4 là thuật toán trích rút các cây cơ bản từ một cây phân tích. Thuật toán này sử dụng các hàm như sau.

Thuật toán 5 trích rút cây khởi tạo (spine). Hàm MERGE-LINK-NODES(T) ghép các nút liên kết của một cây spine thành một nút (xem Hình 11). Các thuật toán 6 và 7 là các hàm tương ứng trích rút các cây phụ trợ và cây đẳng lập. Ví dụ, từ cây phân tích ở Hình 8, ta trích được 9 cây cơ bản như trên các Hình 10 và 11.

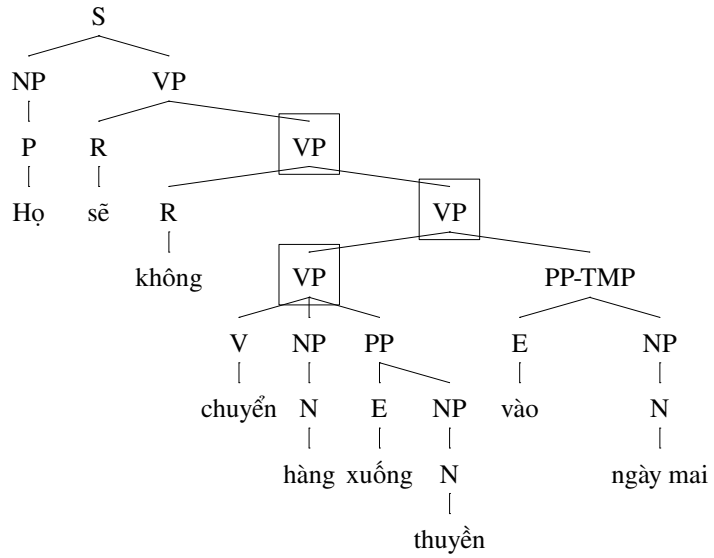
5.3 Lọc bỏ cây không hợp lệ

Các lỗi chú giải là không thể tránh khỏi đối với các treebank lớn, những lỗi trong các cây phân tích cú pháp sẽ dẫn đến các cây cơ bản không hợp lệ. Cây cơ bản được gọi là không hợp lệ nếu nó không thoả mãn một yêu cầu ngôn ngữ học nào đó.

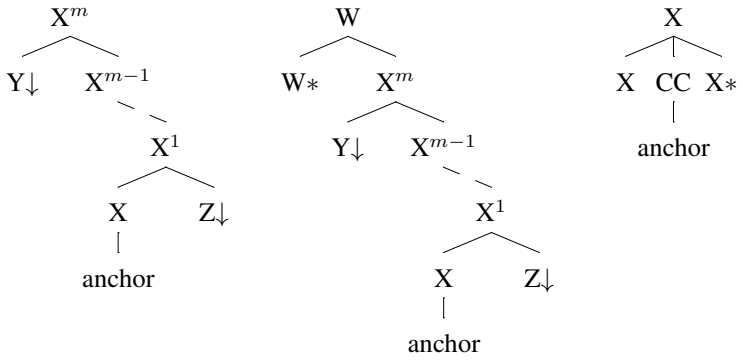
Dựa trên một số tri thức ngôn ngữ tiếng Việt, chúng tôi đã xây dựng một bộ luật để lọc các cây cơ bản không hợp lệ. Ví dụ, trong tiếng Việt, một tính từ (hoặc một cụm tính từ) có thể làm phần phụ của một danh từ (hoặc một cụm danh từ), tuy nhiên nó luôn phải đi sau danh từ. Vì vậy, nếu có cây cơ bản trong đó có tính từ nằm bên trái danh từ thì cây này là không hợp lệ, cần được lọc ra. Một ví dụ khác về kiểu cây không hợp lệ là cây khởi tạo trong đó nút trung tâm có nhiều hơn 4 đối bắt buộc, trường hợp không xảy ra trong tiếng Việt, như cây trên Hình 12. Thông qua việc kiểm tra tính hợp lệ của tập cây cơ bản được trích rút, chúng tôi đã đề xuất nhiều cải tiến và sửa lỗi cho VietTreebank, giúp nâng cao chất lượng của treebank tiếng Việt.

5.4 So sánh với thuật toán của Xia

Cách tiếp cận trích rút văn phạm LTAG mà chúng tôi trình bày tương đối giống với phương pháp trích rút văn phạm được đề xuất bởi Xia [28]. Tuy nhiên, có một số điểm khác nhau về phương pháp thiết kế và cài đặt thuật toán giữa hai cách tiếp cận.



Hình 8: Cây phân tích của cây cú pháp trong Hình 7



Hình 9: Các mẫu cây cơ sở spine (ứng với quan hệ đối–vị từ) và phụ trợ (ứng với quan hệ phụ trợ hoặc đẳng lập

Thứ nhất, trong bước xây dựng cây phân tích, trước tiên chúng tôi xử lí toàn bộ các cụm liên từ đẳng lập của cây cú pháp trước khi phân biệt các thành phần đối và phụ trợ, thay vì xử lí đồng thời cả ba dạng cấu trúc. Việc xử lí tuần tự này dễ hiểu và dễ cài đặt hơn vì các cụm đẳng lập có cấu trúc khác với các cấu trúc đối và phụ trợ. Thứ hai, trong bước trích rút cây cơ bản, chúng tôi không tách mỗi nút của cây thành hai thành phần trên và dưới như trong cách tiếp cận của Xia. Các nút của cây phân tích được sao chép trực tiếp sang các cây cơ bản. Việc sao chép trực tiếp mà không tách nút làm tăng tính hiệu quả thời gian và không gian của các thuật toán. Thứ ba, quá trình trích rút cây được phân rã thành các thủ tục con, gọi tương hỗ qua lại để lặp lại quá trình trích rút trên từng cây con có nút gốc chưa được xử lí. Các hàm đệ quy tương hỗ được thiết kế kỹ lưỡng đảm bảo không có lời gọi thừa, mỗi một nút của cây phân tích chỉ được xử lí một lần. Tính hiệu quả và dễ tối ưu hoá của phương pháp "chia để trị" đã được chứng minh trong lí thuyết thiết kế thuật toán.

6 Kết quả thử nghiệm

Chúng tôi chạy các thuật toán trích rút trên treebank tiếng Việt và trích hai văn phạm. Văn phạm thứ nhất, G_1, sử dụng bộ nhãn gốc của treebank. Văn phạm thứ hai, G_2, sử dụng bộ nhãn thu gọn, trong đó một số nhãn của treebank được ghép thành một nhãn, như trên Bảng 2. Văn phạm G_2 nhỏ hơn G_1, trong

Thuật toán 4 BUILD-ELEMENTARY-TREES(T)

Input: T là một cây phân tích
Output: Các tập cây cơ bản $\mathcal{S}, \mathcal{M}, \mathcal{C}$

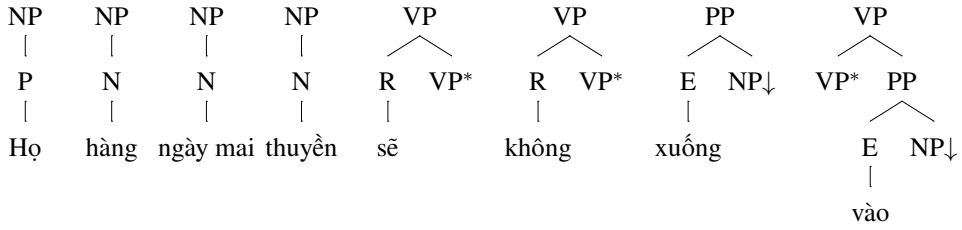
if (not IS-PHRASAL(T)) **then**
  return ;
end if
$\{H_0, H_1, \ldots, H_n\} \leftarrow$ HEAD-PATH(T);
$ok \leftarrow$ `false`;
$P \leftarrow H_0$;
for $j \leftarrow 1$ **to** n **do**
  $\mathcal{L} \leftarrow$ SISTERS(H_j);
  if $|\mathcal{L}| > 0$ **then**
    `Rel` $\leftarrow$ GET-RELATION($H_j, \mathcal{L}$);
    if `Rel` $=$ `Coordination` **then**
      $\mathcal{C} \leftarrow \mathcal{C} \cup$ BUILD-CONJ-TREE(P);
    end if
    if `Rel` $=$ `Modification` **then**
      $\mathcal{M} \leftarrow \mathcal{M} \cup$ BUILD-MOD-TREE(P);
      if $j = 1$ **then**
        $\mathcal{S} \leftarrow \mathcal{S} \cup$ BUILD-SPINE-TREE(P);
        $ok \leftarrow$ `true`;
      end if
    end if
    if `Rel` $=$ `Argument` **then**
      if not ok and not IS-LINK-NODE(P) **then**
        $\mathcal{S} \leftarrow \mathcal{S} \cup$ BUILD-SPINE-TREE(P);
        $ok \leftarrow$ `true`;
      end if
    end if
  else
    if not IS-LINK-NODE(P) and IS-PHRASAL(P) **then**
      $\mathcal{S} \leftarrow \mathcal{S} \cup$ BUILD-SPINE-TREE(P);
    end if
  end if
  $P \leftarrow H_j$;
end for

khi vẫn mô tả được hầu hết các cấu trúc cú pháp cùng loại của G_1; điều này giúp giảm bớt vấn đề dữ liệu thưa, tăng độ chính xác khi sử dụng văn phạm trong các bộ phân tích cú pháp (đơn định hoặc thống kê). Ngoài ra, kích thước của văn phạm cũng đã được chứng minh là một nhân tố quan trọng trong phân tích phụ thuộc và phân tích cú pháp bộ phận [4].

Chúng tôi đếm số cây cơ bản và các mẫu cây[3]. Kích thước của hai văn phạm được cho trong Bảng 3. Có 15035 từ duy nhất trong treebank và trung bình một từ gắn với 3.07 cây cơ bản. Chúng tôi cũng đếm số quy tắc phi ngữ cảnh của các văn phạm trong đó các quy tắc được xây dựng đơn giản bằng cách đọc các mẫu cây (kí hiệu trái của quy tắc là nút gốc, các kí hiệu bên phải của quy tắc là các nút con của gốc). Các văn phạm G_1 và G_2 tương ứng có 851 và 727 quy tắc phi ngữ cảnh.

Để đánh giá độ phủ của treebank tiếng Việt, chúng tôi đếm số mẫu cây ứng với kích thước của treebank. Hình 13 minh hoạ số mẫu cây tăng dần theo kích thước của treebank được sử dụng. Việc hội tụ rất chậm của số mẫu cây cho thấy kích thước hiện tại của VietTreebank là chưa đủ lớn để phủ hết các

[3]Mẫu cây là cây cơ bản bỏ đi nút neo.



Hình 10: Các cây cơ bản

Thuật toán 5 BUILD-SPINE-TREE(T)

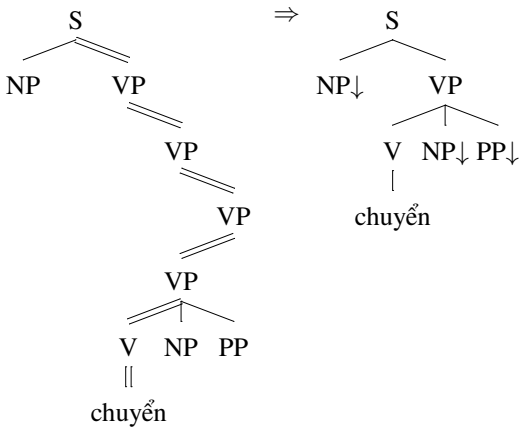
Input: Một cây phân tích T
Output: Cây khởi tạo dạng spine

```
T_c ← COPY(T);
P ← T_c;
H ← NULL;
repeat
    H ← HEAD-CHILD(P);
    L ← SISTERS(H);
    if |L| > 0 then
        Rel ← GET-RELATION(H, L);
        if Rel = Argument then
            for A ∈ L do
                BUILD-ELEMENTARY-TREES(A);
                A.kids ← ∅;
                A.type ← Substitution;
            end for
        else
            for A ∈ L do
                P.kids ← P.kids \ A;
            end for
        end if
    end if
    P ← H;
until (H = NULL)
return MERGE-LINK-NODES(T_c);
```

mẫu cú pháp của tiếng Việt.

Chúng tôi đã phát triển một chương trình phần mềm có tên **LExtractor** cài đặt các thuật toán trích rút văn phạm ở trên. Chương trình được viết bằng ngôn ngữ lập trình Java và được phân phối dưới dạng phần mềm tự do cho cộng đồng nghiên cứu và ứng dụng theo giấy phép GNU/GPL[4]. Chương trình chạy nhanh và hiệu quả: để trích rút toàn bộ văn phạm G_1 ở trên chương trình chỉ cần chạy trong 165 giây trên một máy tính cá nhân thông thường. Chương trình **LExtractor** được thiết kế và cài đặt tương đối tổng quát, dễ dàng sử dụng để trích rút văn phạm từ các treebank của các ngôn ngữ khác. Mọi thông tin liên quan tới đặc trưng của ngôn ngữ được tách ra khỏi lõi chương trình trích rút. Do đó, người sử dụng chương trình trên một ngôn ngữ khác chỉ cần cung cấp thông tin liên quan tới treebank của ngôn ngữ đó dưới dạng các bảng nhãn, bảng thành phần tử trung tâm và bảng đối.

[4]Chương trình chạy và mã nguồn có thể tải từ địa chỉ *http://www.loria.fr/∼lehong/tools/vnLExtractor.php*

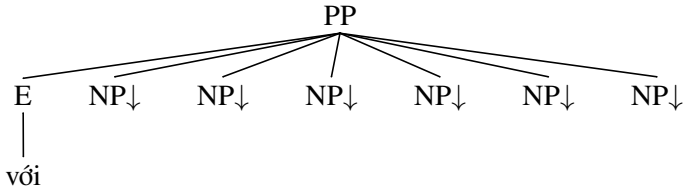


Hình 11: Ghép các nút liên kết. Đường đi trung tâm được đánh dấu bằng nét đôi

Thuật toán 6 BUILD-MOD-TREE(T)

Input: Một cây phân tích T
Output: Một cây phụ trợ

```
T_c ← COPY(T);
H ← HEAD-CHILD(T_c);
H.kids ← ∅;
H.type ← Foot;
M ← MODIFIER(H);
T' ← BUILD-SPINE-TREE(M);
if |M.kids| > 1 then
    BUILD-ELEMENTARY-TREES(M);
end if
M ← T';
return T_c;
```



Hình 12: Một cây không hợp lệ

7 Kết luận và hướng phát triển

Trong bài báo này, chúng tôi đã giới thiệu các thuật toán và một hệ thống tự động trích rút các văn phạm LTAG từ treebank. Hệ thống được thử nghiệm để trích rút văn phạm LTAG cho tiếng Việt từ treebank tiếng Việt.

Mặc dù văn phạm LTAG thu được đã phủ hoàn toàn các cấu trúc cú pháp của treebank, số mẫu cây của văn phạm hội tụ rất chậm cho thấy có nhiều cấu trúc cú pháp chưa được mã hoá trong treebank, nói cách khác là treebank tiếng Việt chưa đủ lớn hoặc chưa đủ điển hình để phủ hết các mẫu cú pháp của tiếng Việt.

Khi tiến hành đánh giá kết quả phân tích cú pháp tiếng Việt sử dụng bộ phân tích cú pháp LLP2 [11] chúng tôi nhận thấy tiếng Việt có độ nhập nhằng cú pháp rất cao. Ví dụ, khi sử dụng một văn phạm LTAG kích thước trung bình để phân tích cú pháp 70 câu có độ dài nhỏ hơn hay bằng 15 từ thì kết quả thu được là mỗi câu có trung bình 49.6 cách phân tích, trong đó chỉ có 14 câu có duy nhất một cách phân tích.

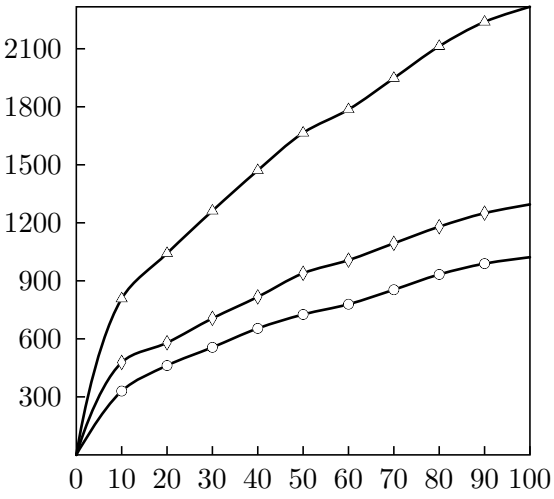
Thuật toán 7 BUILD-CONJ-TREE(T)

Input: Một cây phân tích T
Output: Một cây đẳng lập

1: $T_c \leftarrow$ COPY(T);
2: $H \leftarrow$ HEAD-CHILD(T_c);
3: BUILD-ELEMENTARY-TREES(H);
4: $K \leftarrow$ COORDINATOR(H);
5: BUILD-ELEMENTARY-TREES(K);
6: H.`kids` $\leftarrow \emptyset$;
7: H.`type` $\leftarrow$ `Foot`;
8: K.`kids` $\leftarrow \emptyset$;
9: K.`type` $\leftarrow$ `Substitution`;
10: **return** T_c;

Loại	**Nhãn gốc**	**Nhãn trong G_2**
cụm danh từ	NP/WHNP	NP
cụm tính từ	AP/WHAP	AP
cụm phó từ	RP/WHRP	RP
cụm giới từ	PP/WHPP	PP
mệnh đề	S/SQ	S

Bảng 2: Ghép một số nhãn cú pháp của VietTreebank thành một



Hình 13: Số mẫu cây tăng dần theo kích thước của treebank. Trục x biểu diễn phần trăm kho văn bản được sử dụng để trích rút văn phạm, trục y biểu diễn số lượng mẫu cây tổng thể ($\triangle$), mẫu cây khởi tạo (o) và mẫu cây phụ trợ ($\diamond$) thu được.

Trong thời gian tới, chúng tôi dự định đánh giá một cách toàn diện hiệu quả cũng như độ chính xác của chương trình phân tích cú pháp và văn phạm mà chúng tôi đã xây dựng.

Hiện tại, chúng tôi cũng đang thử nghiệm việc trích chọn một văn phạm LTAG cho tiếng Pháp, từ treebank tiếng Pháp [2]. Dựa trên các kết quả trích rút, chúng tôi cũng dự định so sánh các cấu trúc cú pháp của tiếng Pháp và tiếng Việt một cách định lượng để tìm các điểm chung, phục vụ một số nghiên cứu đối chiếu giữa hai ngôn ngữ.

Kiểu	Số cây	Số mẫu cây
G_1	**46382**	**2317**
Cây khởi tạo	24973	1022
Cây phụ trợ	21309	1223
Cây đẳng lập	100	72
G_2	**46102**	**2113**
Cây khởi tạo	24884	952
Cây phụ trợ	21121	1093
Cây đẳng lập	97	68

Bảng 3: Hai văn phạm LTAG được trích rút từ VietTreebank

Nhãn	Hướng chọn	Danh sách ưu tiên
S	Trái	S VP AP NP
SBAR	Trái	SBAR S VP AP NP
SQ	Trái	SQ VP AP NP
NP	Trái	NP Nc Nu Np N P
VP	Trái	VP V A AP N NP S
AP	Trái	AP A N S
RP	Phải	RP R T NP
PP	Trái	PP E VP SBAR AP QP
QP	Trái	QP M
XP	Trái	XP X
YP	Trái	YP Y
MDP	Trái	MDP T I A P R X
WHNP	Trái	WHNP NP Nc Nu Np N P
WHAP	Trái	WHAP A N V P X
WHRP	Trái	WHRP P E T X
WHPP	Trái	WHPP E P X
WHXP	Trái	XP X

Bảng 4: Bảng thành phần trung tâm cho treebank tiếng Việt

Lời cảm ơn

Công trình này được sự hỗ trợ tài chính của đề tài nghiên cứu khoa học QT-09-01 thuộc Đại học Quốc gia Hà Nội.

Tài liệu tham khảo

[1] A. Abeillé. *Une grammaire électronique du français*. CNRS, Paris, 2002.

[2] A. Abeillé, L. Clément, and F. Toussenel. Building a treebank for French. In *Treebanks: Building and Using Parsed Corpora*. Kluwer, Dordrecht, 2003.

[3] J. Bäcker and K. Harbusch. Hidden Markov model-based supertagging in a user-initiative dialogue system. In *Proceedings of TAG+6*, pages 269–278, Universita di Venezia, 2002.

[4] S. Bangalore. Performance evaluation of supertagging for partial parsing. In *Advances in probabilistic and other parsing technologies*, pages 203–220. Kluwer Academic Publishers, 2000.

[5] X. Carreras, M. Collins, and T. Koo. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of COLING 2008*, Manchester, 2008.

[6] J. Chen, S. Bangalore, and K. Vijay-Shanker. Automated extraction of tree-adjoining grammars from treebanks. *Natural Language Engineering*, 12(3):251–299, 2006.

[7] J. Chen and K. Vijay-Shanker. Automated extraction of TAGs from the Penn treebank. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, 2000.

[8] D. Chiang. Statistical parsing with an automatically-extracted tree adjoining grammar. In *ACL'00*, pages 456–463, Morristown, NJ, USA, 2000.

[9] M. Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL*, 1997.

[10] B. Crabbé. Grammatical development with XMG. In *Proceedings of the 5th International Conference on Logical Aspects of Computational Linguistics*, 2005.

[11] B. Crabbé, B. Gaiffe, and A. Roussanaly. Représentation et gestion de grammaires d'arbres adjoints lexicalisées. *Traitement Automatique des Langues*, 44(3):67–91, 2003.

[12] E. V. de la Clergerie, B. Sagot, L. Nicolas, and M.-L. Guénot. FRMG: évolution d'un analyseur syntaxique TAG du français. In *Workshop ATALA de IWPT 2009*, Paris, 2009.

[13] C. Doran, B. Hockey, A. Sarkar, and B. Srinivas. Evolution of the XTAG system. In A. Abeillé and O. Rambow, editors, *Tree adjoining grammars*, pages 371–404. Stanford CSLI, 2000.

[14] R. Frank. *Phrase Structure Composition and Syntactic Dependencies*. MIT Press, Boston, 2002.

[15] N. Habash and O. Rambow. Extracting a tree adjoining grammar from the Penn Arabic treebank. In *Proceedings of TALN'04*, Morocco, 2004.

[16] A.-D. Johansen. Extraction des grammaires LTAG à partir d'un corpus étiquetté syntaxiquement. Master's thesis, Université Paris 7, 2004.

[17] A. K. Joshi, L. S. Levy, and M. Takahashi. Tree adjunct grammars. *Journal of the Computer and System Sciences*, 10:136–165, 1975.

[18] A. K. Joshi and Y. Schabes. *Handbooks of Formal Languages and Automata*, chapter Tree Adjoining Grammars. Springer-Verlag, 1997.

[19] L. Kallmeyer, T. Lichte, W. Maier, Y. Parmentier, and J. Dellert. Developping an MCTAG for German with an RCG-based parser. In *Proceedings of LREC 2008*, Marrakech, Morocco, 2008.

[20] A. Kinyon and C. A. Prolo. A classification of grammar development strategies. In *Proceedings of the Workshop on Grammar Engineering and Evaluation*, pages 43—49, Taipei, Taiwan, 2002.

[21] P. Le-Hong, T. M. H. Nguyen, P. T. Nguyen, and A. Roussanaly. Automated extraction of tree adjoining grammars from a treebank for Vietnamese. In *Proceedings of TAG+10*, Yale University, New Haven, CT, USA, 2010.

[22] D. M. Magerman. Statistical decision tree models for parsing. In *Proceedings of ACL*, 1995.

[23] A. Nasr. *Analyse syntaxique probabiliste pour grammaires de dépendances extraites automatiquement*. Habilitation à diriger des recherches, Université Paris 7, 2004.

[24] G. Neumann. A uniform method for automatically extracting stochastic lexicalized tree grammar from treebank and HPSG. In *Treebanks: Building and Using Parsed Corpora*. Kluwer, Dordrecht, 2003.

[25] P. T. Nguyen, L. V. Xuan, T. M. H. Nguyen, V. H. Nguyen, and P. Le-Hong. Building a large syntactically-annotated corpus of Vietnamese. In *Proceedings of the 3rd Linguistic Annotation Workshop, ACL-IJCNLP*, Singapore, 2009.

[26] J. Park. Extraction of tree adjoining grammars from a treebank for Korean. In *COLING ACL'06 Student Research Workshop*, pages 73–78, Morristown, NJ, USA, 2006.

[27] Y. Parmentier. *SemTAG: Une plate-forme pour le calcul sémantique à partir de grammaires d'arbres adjoints*. PhD thesis, Université Henri Poincaré, Nancy I, 2007.

[28] F. Xia. *Automatic grammar generation from two different perspectives*. PhD thesis, University of Pennsylvania, 2001.

[29] F. Xia, M. Palmer, and A. Joshi. A uniform method of grammar extraction and its applications. In *Proceedings of the joint SIGDAT conference on empirical methods in NLP and very large corpora*, pages 53–62, Morristown, NJ, USA, 2000.

[30] XTAG-Research-Group. A lexicalized tree adjoining grammar for English. Technical report, Institute for Research in Cognitive Science, University of Pennsylvania, 2001.

[31] N. Yoshinaga, Y. Miyao, K. Torisawa, and J. Tsuji. Parsing comparison across grammar formalisms using strongly equivalent grammars. *Traitement Automatique des Langues*, 44(3):15–39, 2003.