TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
VIỆN TRÍ TUỆ NHÂN TẠO

BÁO CÁO BÀI TẬP LỚN MÔN HỌC XỬ LÝ NGÔN NGỮ TỰ NHIÊN

PHÂN TÍCH CÚ PHÁP PHỤ THUỘC TIẾNG VIỆT

Nhóm sinh viên thực hiện:

1: Nguyễn Công Hiếu (MSV: 22022510)

2: Hoàng Đăng Khoa (MSV: 22022548)

3: Long Trí Thái Sơn (MSV: 22022653)

4: Trịnh Đắc Phú (MSV: 22022597)

HÀ NỘI - 2024

Mục lục

1 Giới thiệu bài toán **1**

1.1 Phân tích cú pháp phụ thuộc 2

1.1.1 Định nghĩa cú pháp phụ thuộc 2

1.1.2 Bài toán phân tích cú pháp phụ thuộc 3

1.1.3 Đặc điểm của ngôn ngữ tiếng Việt 3

2 Phương pháp **5**

2.1 Phân tích cú pháp phụ thuộc dựa vào đồ thị 5

2.1.1 Tìm cây bao trùm tối đa 7

2.1.2 Cách học hàm tính điểm s 10

2.2 Phân tích cú pháp phụ thuộc dựa vào bước chuyển 13

2.2.1 Các thuật toán phân tích cú phụ phụ thuộc vào bước chuyển 14

2.2.2 Thực hiện từng bước 15

3 Thí nghiệm **20**

3.1 Các công cụ sử dụng 20

3.1.1 MSTParser 20

3.1.2 MaltParser 21

3.2 Dữ liệu 21

3.3 Nhãn Dữ liệu 21

3.3.1 Kho ngữ liệu tiếng Việt - Viettreebank 21

3.3.2 Tập nhãn quan hệ phụ thuộc đa ngôn ngữ 24

3.4 Kết quả và phân tích 25

4 Kết luận **27**

Chương 1

Giới thiệu bài toán

Xử lý Ngôn ngữ Tự nhiên (Natural Language Processing - NLP) là một nhánh của trí tuệ nhân tạo và ngôn ngữ học, tập trung vào việc phát triển các hệ thống có khả năng hiểu, diễn giải và tạo ra ngôn ngữ tự nhiên của con người. Trong quá trình xử lý ngôn ngữ tự nhiên, máy tính phải đối mặt với nhiều thách thức, từ việc nhận dạng và phân tích các thành phần cơ bản của ngôn ngữ như từ và câu, đến việc hiểu ngữ cảnh, ý nghĩa và cảm xúc ẩn chứa trong văn bản. Phân tích cú pháp phụ thuộc (Dependency Parsing) là một trong những nhiệm vụ quan trọng và cơ bản trong lĩnh vực Xử lý Ngôn ngữ Tự nhiên.

Phân tích cú pháp phụ thuộc đóng vai trò then chốt trong việc giải quyết một trong những thách thức cơ bản nhất của NLP: hiểu cấu trúc ngữ pháp của câu. Bằng cách xác định mối quan hệ phụ thuộc giữa các từ trong câu, phân tích cú pháp phụ thuộc cung cấp một biểu diễn cấu trúc chi tiết, giúp máy tính "hiểu" cách các thành phần trong câu liên kết với nhau để tạo nên ý nghĩa. Điều này là nền tảng cho nhiều ứng dụng NLP cao cấp hơn, như dịch máy, hệ thống hỏi đáp, tóm tắt văn bản, và phân tích tình cảm.

Đối với tiếng Việt, một ngôn ngữ có cấu trúc ngữ pháp phức tạp và đa dạng, thuộc nhóm ngôn ngữ đơn lập, trong đó mối quan hệ ngữ pháp giữa các từ được thể hiện chủ yếu thông qua trật tự từ và các từ chức năng, thay vì thông qua biến đổi hình thái như trong nhiều ngôn ngữ Ấn-Âu. Do đó phân tích cú pháp phụ thuộc là một nhiệm vụ cơ bản nhưng đóng vai trò quyết định trong việc xử lý ngôn ngữ tự nhiên tiếng Việt. Nó không chỉ giúp máy tính "hiểu" cấu trúc ngữ pháp của câu mà còn tạo nền tảng cho việc phát triển các ứng dụng NLP tiên tiến, đóng góp vào sự phát triển của công nghệ xử lý ngôn ngữ tự nhiên cho tiếng Việt nói riêng và ngôn ngữ học tính toán nói chung.

1.1 Phân tích cú pháp phụ thuộc

Cú pháp là tập hợp các quy tắc chi phối cấu trúc của các câu trong một ngôn ngữ. Nó quy định cách các từ và cụm từ được sắp xếp để tạo thành câu có nghĩa. Ví dụ, trong tiếng Việt, một câu cơ bản thường có cấu trúc Chủ ngữ - Động từ - Tân ngữ (SVO), như trong câu "Tôi ăn cơm" (Chủ ngữ: "Tôi", Động từ: "ăn", Tân ngữ: "cơm").

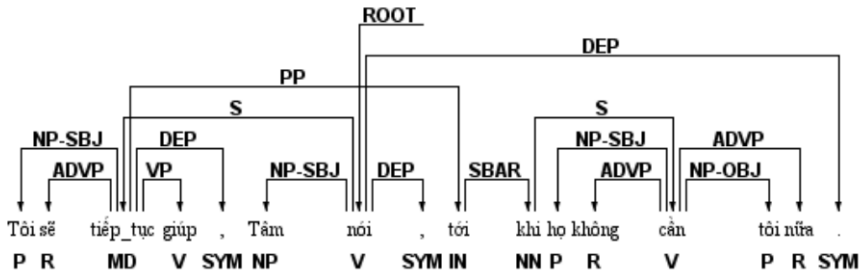
Phân tích cú pháp (syntactic analysis) là quá trình xác định và phân tích cấu trúc cú pháp của một câu hoặc một đoạn văn bản. Mục tiêu của phân tích cú pháp là hiểu được cách mà các thành phần trong câu (chủ ngữ, động từ, tân ngữ,...) liên kết với nhau để tạo ra ý nghĩa. Có hai dạng phân tích cú pháp chính: phân tích cú pháp thành phần và phân tích cú pháp phụ thuộc.

1.1.1 Định nghĩa cú pháp phụ thuộc

Cú pháp phụ thuộc là cấu trúc cú pháp mô tả các mối quan hệ giữa các từ trong một câu, dựa trên nguyên tắc "phụ thuộc". Thay vì chia câu thành các cụm từ như trong cú pháp thành phần, cú pháp phụ thuộc liên kết trực tiếp các từ với nhau thông qua các quan hệ phụ thuộc. Quan hệ phụ thuộc này có thể được đặt tên để làm rõ quan hệ giữa hai từ.

Một quan hệ phụ thuộc thể hiện bằng 1 mũi tên có hướng, trong đó:

- Head (Đầu): Là từ chính trong một cặp quan hệ phụ thuộc, thường là động từ hoặc danh từ. Đây là từ có chức năng ngữ pháp chính trong cấu trúc câu.
- Dependent (Phụ thuộc): Là từ phụ thuộc vào từ "head". Từ này thường cung cấp thêm thông tin về từ "head", ví dụ như bổ ngữ hoặc trạng từ .
- Arc (Cung): Là mối quan hệ hướng giữa từ "head" và từ "dependent". Mối quan hệ này thể hiện loại phụ thuộc ngữ pháp, ví dụ: chủ ngữ, tân ngữ, hoặc định ngữ.



Hình 1.1: Đồ thị phụ thuộc câu Tiếng Việt.

Về cơ bản, chúng ta có thể biểu diễn các phần phụ thuộc dưới dạng đồ thị có hướng $G = (V, E)$ trong đó V (tập các đỉnh) đại diện cho các từ (và cả các dấu câu) trong câu và E (tập các cung) đại diện cho mối quan hệ ngữ pháp giữa các phần tử của V. Theo quy ước phổ biến thì từ nằm

ở gốc (ROOT) là từ chính của toàn bộ câu, thường là động từ chính và không phụ thuộc vào từ nào khác. Đồ thị phụ thuộc này có các tính chất sau:

- 1. Liên thông yếu
- 2. Một từ chỉ có duy nhất 1 head (một cạnh đi vào)
- 3. Không có chu trình
- 4. Nếu câu có n từ (kể cả root) thì đồ thị có (n-1) cạnh

Cấu trúc phụ thuộc giúp đơn giản hóa, dễ dàng xử lý hơn đối với các ngôn ngữ có cấu trúc câu phức tạp trong đó có Tiếng Việt. Ngoài ra cấu trúc phụ thuộc giúp xác định mối quan hệ ngữ nghĩa giữa các thành phần trong câu, hỗ trợ cho các tác vụ như trích xuất thông tin và hiểu ngữ nghĩa, được sử dụng rộng rãi trong các ứng dụng NLP hiện đại như dịch máy, phân tích tình cảm, và nhận diện thực thể có tên.

1.1.2 Bài toán phân tích cú pháp phụ thuộc

Phân tích cú pháp phụ thuộc chính là tìm ra đồ thi phụ thuộc của một câu, từ đó có thể mô tả về mối quan hệ giữa các từ trong câu. Mục tiêu của bài tập lớn này là cần tìm ra phương pháp sinh đồ thị phụ thuộc chính xác nhất cho câu đầu vào Tiếng Việt, tức là làm cực đại số cung chính xác trong đồ thị và số nhãn gán đúng cho các cung.

- **Đầu vào:** Câu văn bản $x = (w_1, w_2, \ldots, w_n)$ đã được tiền xử lý, tách từ và gán nhãn từ loại.
- **Đầu ra:** Đồ thị phụ thuộc $G_x = (V_x, E_x)$

Tập các đỉnh là tập các số nguyên không âm tăng dần đến n: $V_x = Z_{n+1} = \{0, 1, 2, ..., n\}, n \in \mathbb{Z}^+$. Tức là mỗi từ trong câu tương ứng với một đỉnh $(1 \leq i \leq n)$ và đỉnh ROOT là 0. Ký hiệu V_x^+ là tập tất cả các đỉnh không chứa root.

Tập các cung E_x là tập các cặp (i, j) trong đó i, j là các đỉnh, kí hiệu $i \to j$ tương ứng với cung nối đỉnh i và $i \to^* j$ là một mối quan hệ bao hàm cả trường hợp tự phụ thuộc $(i = j)$ và mối quan hệ phụ thuộc $i \to j$.

1.1.3 Đặc điểm của ngôn ngữ tiếng Việt

Tính phân tích

Tiếng Việt được xếp vào nhóm ngôn ngữ có tính phân tích cao. Điều này có nghĩa là tiếng Việt chủ yếu sử dụng từ rời rạc và từ không biến đổi hình thái để biểu thị ngữ pháp và ý nghĩa, thay vì sử dụng sự biến đổi của từ gốc (như chia động từ, biến đổi danh từ số ít/số nhiều, hoặc biến đổi tính từ) như trong các ngôn ngữ tổng hợp như tiếng Hi Lạp, tiếng La-tinh, tiếng Đức.

Tính đơn hình

Hầu hết các từ trong tiếng Việt là đơn tiết (chỉ bao gồm một âm tiết) và thường tương ứng với một ý nghĩa cụ thể. Tuy nhiên, điều này không có nghĩa là tiếng Việt hoàn toàn đơn hình, mà đúng hơn là phần lớn từ vựng cơ bản trong tiếng Việt có cấu trúc đơn âm tiết.

Trật tự từ

Các loại trật tự từ khác nhau thường được mô tả dựa trên vị trí của ba thành phần chính trong câu: Chủ ngữ (Subject - S), Động từ (Verb - V), và Tân ngữ (Object - O). Có 6 trật tự từ cơ bản: SVO, SOV, VSO, VOS, OSV, OVS, trong đó tiếng Việt thuộc loại SVO.

Điều kiện xạ ảnh

Câu được gọi là "xạ ảnh" (projective) nếu các cung của nó không cắt nhau khi vẽ trên mặt phẳng. Điều này có nghĩa là trong cây phụ thuộc của câu, nếu có một từ phụ thuộc vào một từ khác, thì tất cả các từ giữa chúng trong câu cũng phải phụ thuộc vào các từ trong khoảng đó hoặc là con cháu của từ đó. Tức là nếu $i \to j$ thì $i \to^* k$ bất kỳ thỏa mãn $i < k < j$ hoặc $j < k < i$.

Trong tiếng Việt, đa số các câu thỏa mãn tính chất xạ ảnh. Điều này thường xảy ra khi câu có trật tự từ tương đối đơn giản và không có sự đảo ngữ hay cấu trúc câu phức tạp. Tuy nhiên vẫn tồn tại những câu ghép không có tính xạ ảnh, do đó cần quan tâm đến khi phân tích cú pháp phú thuộc.

Chương 2

Phương pháp

2.1 Phân tích cú pháp phụ thuộc dựa vào đồ thị

Sử dụng đồ thị là một trong những hướng tiếp cận hiệu quả và phổ biến nhất cho bài toán phân tích cú pháp [7]. Ý tưởng chính của phương pháp này bao gồm ba phần chính:

- **Xác định không gian ứng viên:** Xác định không gian các cây phụ thuộc ứng viên cho một câu.
- **Huấn luyện:** Tạo một mô hình để đánh giá điểm số cây phụ thuộc của câu đầu vào.
- **Phân tích:** Tìm trong không gian các cây phụ thuộc cây có kết quả tốt nhất dựa vào điểm số từ mô hình nói trên.

Nhắc lại, với một câu đầu vào $x = (w_1, w_2, \ldots, w_n)$, ta định nghĩa $G_x = (V_x, E_x)$ là một đồ thị có hướng với tập đỉnh là các từ trong câu x cùng với ký hiệu gốc "ROOT", tập cạnh bao gồm các cạnh có hướng kết nối các cặp từ trong câu, cũng như các cạnh từ "ROOT" đến tất cả các từ trong câu [7].

$$\begin{aligned} V_x &= \{x_0 = \text{ROOT}, x_1, x_2, \ldots, x_n\} \\ E_x &= \{(i,j) : i \neq j, (i,j) \in [0,n] \times [1,n]\} \end{aligned} \tag{2.1}$$

Khi đó ta có thể thấy rõ rằng vì G_x chứa tất cả những cung được gán nhãn nên không gian các cây phụ thuộc ứng viên cho câu x sẽ nằm trong tập tất cả các đồ thị con của G_x, được ký hiệu là $D(G_x)$. Vấn đề tìm kiếm cây phụ thuộc tốt nhất trở thành bài toán tìm kiếm cây bao trùm tối đa (maximum spanning tree) trong không gian $D(G_x)$ [7]. Tuy nhiên, không phải tất cả các phần tử của $D(G_x)$ đều là cây phụ thuộc hợp lệ. Một đồ thị phụ thuộc G phải là hợp lệ khi và chỉ khi nó thỏa mãn các điều kiện sau:

- Đỉnh 0 là gốc "ROOT".

- G liên thông yếu (weakly connected), nghĩa là trong **đồ thị vô hướng tương ứng với** G, luôn tồn tại đường đi giữa hai đỉnh bất kỳ.
- Mọi đỉnh đều có nhiều nhất một từ trung tâm (single-head), tức là nếu $i \to j$ thì $\nexists k$ thỏa mãn $k \neq i$ và $k \to j$.
- G không có chu trình, tức là nếu có $i \to j$ thì $\nexists j \to^* i$ (acyclicity).

Để đánh giá điểm số của cây phụ thuộc, chúng ta sử dụng một phương pháp tiếp cận phân tách cây phụ thuộc thành các đồ thị con nhỏ hơn [7]. Sau đó, học một hàm tính điểm s cho từng đồ thị con này và xác định điểm số tổng quát của cây phụ thuộc dựa trên tổng điểm của tất cả các đồ thị con. Cụ thể, nếu cây phụ thuộc T được phân tách thành các đồ thị con $G_1, G_2, \ldots, G_m$, điểm số của cây phụ thuộc sẽ được tính theo công thức:

$$s(T) = \sum_{i=1}^{m} s(G_i) \tag{2.2}$$

Kích thước của các đồ thị con G_i phụ thuộc vào phương pháp cụ thể mà chúng ta sử dụng. Trong bài báo cáo này, chúng tôi sẽ chỉ tập trung vào First-order method – một phương pháp mà mỗi đồ thị con G_i chỉ bao gồm hai đỉnh, một cạnh và nhãn của cạnh đó [7]. Nói cách khác, các đồ thị con G_i chính là các phần tử trong tập cạnh đã được gán nhãn:

$$E_x = \{(i, j, r) : i \neq j, (i, j) \in [0, n] \times [1, n], r \in L\}. \tag{2.3}$$

Khi đó, điểm số của cây phụ thuộc T sẽ là:

$$s(T) = \sum_{(i,j,r) \in E_x} s(i, j, r) \tag{2.4}$$

Sau khi đã huấn luyện được hàm $s : V \times V \times L \to \mathbb{R}$ để tính trọng số cho từng cạnh, bước cuối

(a) First-order (b) Second-order

Hình 2.1: Phương pháp first-order và second-order. Ở đây, h là từ chính, m là từ bổ nghĩa và s là từ đồng nghĩa của m.

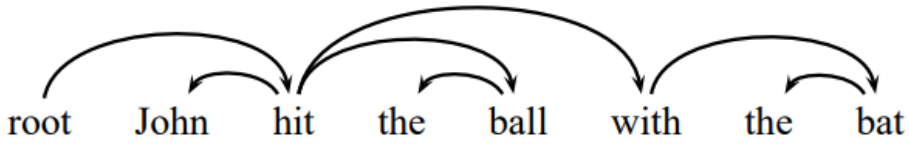
cùng là tìm kiếm cây phụ thuộc tốt nhất trong không gian $D^*(G_x)$ của tất cả các cây phụ thuộc hợp lệ. Bài toán này có thể được biểu diễn dưới dạng:

$$T^* = \arg\max_{T \in D^*(G_x)} s(T) = \arg\max_{T \in D^*(G_x)} \sum_{(i,j,r) \in E_x} s(i, j, r) \tag{2.5}$$

Các phương pháp tìm kiếm cây bao trùm tối đa (maximum spanning tree) và cách huấn luyện hàm đánh giá s cho first-order method sẽ được trình bày cụ thể ở hai mục 2.1.1 và **??**.

2.1.1 Tìm cây bao trùm tối đa

Trong phân tích cú pháp phụ thuộc, hai loại cây phụ thuộc thường được sử dụng là cây xạ ảnh (projective trees) và cây không xạ ảnh (non-projective trees) [7]. Một đồ thị là xạ ảnh nếu như có $i \rightarrow j$ thì $i \rightarrow k \; \forall i \leq k \leq j$ hoặc $j \leq k \leq i$ (projectivity), nghĩa là khi ta sắp xếp các từ trong câu theo thứ tự của nó thì các cạnh phụ thuộc có thể được vẽ mà không giao nhau. Ví dụ về câu xạ ảnh và không xạ ảnh có thể thấy ở hình sau.



(a) Cây bao trùm cho câu xạ ảnh

(b) Cây bao trùm cho câu không xạ ảnh

Hình 2.2: Cây bao trùm cho câu xạ ảnh và không xạ ảnh. Có thể thấy cây xạ ảnh có các cạnh phụ thuộc không giao nhau

Việc áp dụng các thuật toán tìm kiếm cây bao trùm tối đa cho bài toán phân tích cú pháp trước đây mới chỉ dừng lại ở tìm kiếm cây xạ ảnh. Mặc dù cây xạ ảnh là đủ để phân tích cú pháp phần lớn các câu nhưng trong một số trường hợp câu với cấu trúc phức tạp thì cây không xạ ảnh trở nên cần thiết [7].

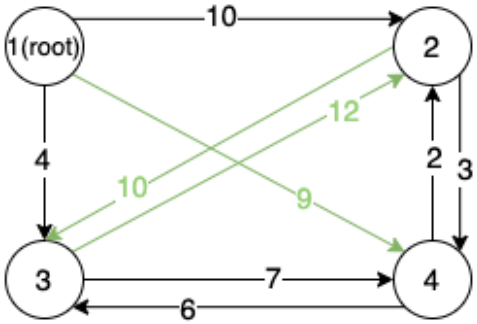
Trong mục này chúng tôi sẽ trình bày về 2 thuật toán chính cho cả 2 hướng tiếp cận: Thuật toán Chu-Liu-Edmonds cho bài toán tìm kiếm cây không xạ ảnh [3, 5] và thuật toán Eisner cho bài toán tìm kiếm cây xạ ảnh [6].

Thuật toán Chu-Liu-Edmonds

Thuật toán Chu-Liu-Edmonds là một phương pháp hiệu quả để tìm cây không xạ ảnh bao trùm tối đa trong đồ thị có hướng [3, 5]. Thuật toán hoạt động dựa trên nguyên tắc tham lam, theo đó, đối với mỗi đỉnh, thuật toán sẽ chọn cạnh có trọng số cao nhất nối đến đỉnh đó. Khi xuất hiện chu trình, thuật toán sẽ phá bỏ chu trình bằng cách thay thế các cạnh sao cho tổng trọng số bị loại bỏ là ít nhất. Quá trình này được lặp lại đệ quy cho đến khi tìm được cây bao trùm tối đa, đảm bảo phân tích cú pháp phụ thuộc một cách chính xác.

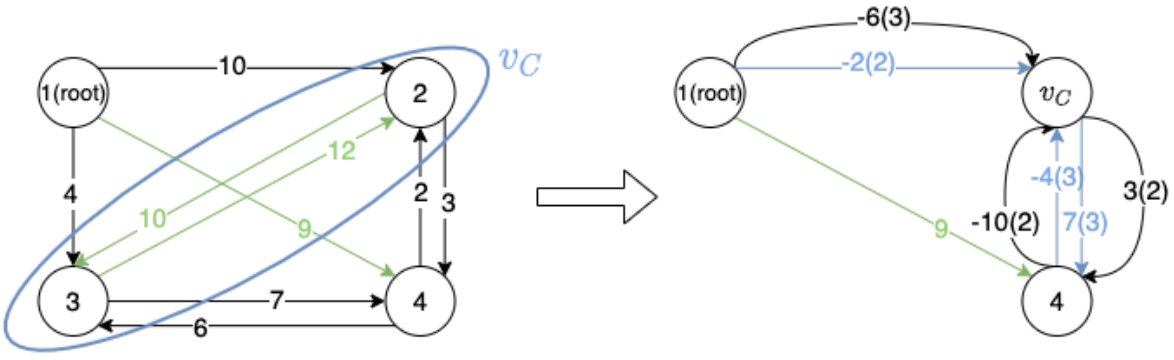
Cụ thể hơn, để sử dụng thuật toán Chu-Liu-Edmonds tìm cây bao trùm tối đa trong đồ thị $G = \{V, E\}$ ta sẽ cần thực hiện ba bước chính:

- **Khởi tạo tham lam:** Với mỗi đỉnh v_i khác ROOT ta chỉ giữ lại cạnh nối vào v_i có trọng số cao nhất, ký hiệu cạnh đó là $e_{\pi(v_i)v_i}$.



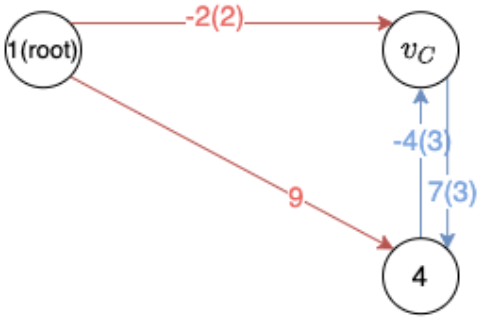
Hình 2.3: Minh họa bước khởi tạo tham lam cho thuật toán Chu-Liu-Edmonds, chỉ giữ lại các cạnh có trọng số cao nhất (màu xanh lá)

- **Phá chu trình**: Để phá một chu trình C_{node} gồm các đỉnh $\{v_{c_1}, v_{c_2}, \ldots, v_{c_k}\}$, thuật toán tạo ra một đồ thị mới $G' = \{V', E'\}$ trong đó toàn bộ các đỉnh thuộc chu trình C_{node} được xem như một đỉnh duy nhất v_C trong G' $(V' = V \backslash C \cup v_C)$. Tập cạnh E' được xây dựng như sau:
  - Với mỗi cạnh $e_{sd} \in E$. Nếu $s \notin C_{node}$ và $d \in C_{node}$ thì ta thêm e'_{sv_C} vào E' với trọng số $w(e'_{sv_C}) = w(e_{sd}) - w(e_{\pi(v_d)v_d})$.
  - Với mỗi cạnh $e_{sd} \in E$. Nếu $s \in C_{node}$ và $d \notin C_{node}$ thì ta thêm e'_{v_Cd} vào E' với trọng số $w(e'_{v_Cd}) = w(e_{sd})$.
  - Với mỗi cạnh $e_{sd} \in E$. Nếu $s \notin C_{node}$ và $d \notin C_{node}$ thì ta thêm e'_{sd} vào E' với trọng số $w(e'_{sd}) = w(e_{sd})$.



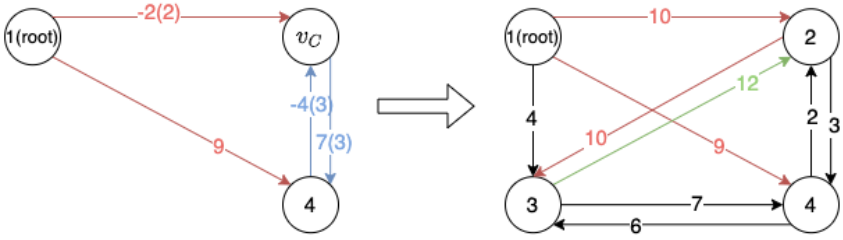
Hình 2.4: Minh họa bước phá bỏ chu trình trong thuật toán Chu-Liu-Edmonds.

Giữa v_C và các đỉnh khác trong G' có thể tồn tại nhiều cạnh; trong trường hợp này, chỉ giữ lại cạnh có trọng số lớn nhất. Sau đó, thuật toán tiếp tục xây dựng cây bao trùm tối đa trong G' một cách đệ quy [3, 5].

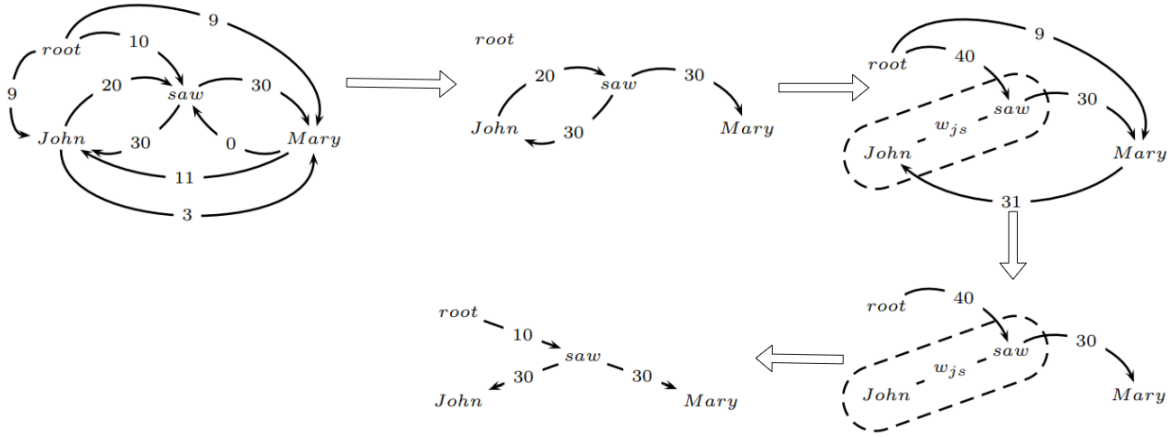


Hình 2.5: Minh họa bước tiếp tục tìm cây khung tối đa trong đồ thị đã phá chu trình một cách đệ quy

- **Khôi phục cây bao trùm tối đa:** Giả sử cạnh nối vào v_C trong cây A của đồ thị G' xuất phát từ đỉnh v_s, và cạnh tương ứng trong đồ thị gốc G là e_{sk} với $v_k \in C_{node}$. Để khôi phục cây cuối cùng, ta thay thế các cạnh từ/tới v_C trong cây A bằng các cạnh tương ứng trong đồ thị gốc. Sau đó, giữ lại các cạnh trong chu trình C_{node}, trừ cạnh nối vào v_k. Kết quả là cây bao trùm tối đa trong đồ thị gốc [3, 5].



Hình 2.6: Minh họa bước khôi phục cây bao trùm tối đa trọng thuật toán Chu-Liu-Edmonds



Hình 2.7: Ví dụ sử dụng thuật toán Chu-Liu-Edmonds để phân tích cú pháp cho câu x = John saw Mary

Thuật toán Eisner

Giải thuật Eisner là một giải thuật dùng để phân tích cú pháp phụ thuộc cho các cấu trúc cây không xạ ảnh sử dụng kỹ thuật quy hoạch động với độ phức tạp $O(n^3)$, hiệu quả hơn so với một số thuật toán khác. Cơ chế của giải thuật Eisner bao gồm các bước sau:

- **Khởi tạo bảng:** Giải thuật sử dụng một bảng 4 chiều C[s][t][d][c] để lưu trữ các cấu trúc phụ thuộc giữa các từ trong câu. Các chiều của bảng: start, target, direction (trái/phải), completeness (hoàn chỉnh/không hoàn chỉnh). Bảng quy hoạch động lưu điểm số của cây con tốt nhất từ vị trí s đến t, biến $d \in \{\leftarrow, \rightarrow\}$, nếu $d = \leftarrow$ thì t là head của cây con, nếu $d = \rightarrow$ thì s là head của cây con. Biến $c = 1$ hàm ý một cây con hoàn chỉnh (không thể thêm phụ thuộc), và $c = 0$ tức là chưa hoàn chỉnh.
- **Xét các span (khoảng):** Thuật toán bắt đầu với các span bằng 1, sau đó tăng dần lên đến n. Mỗi span được xác định bởi điểm bắt đầu và kết thúc. Với mỗi span, xét tất cả các cách chia nhỏ có thể để tìm cách chia tốt nhất để xây dựng cấu trúc phụ thuộc.
- **Cập nhật điểm:** Với mỗi cách chia, cập nhật 4 trường hợp:a. Span trái không hoàn chỉnh, b. Span phải không hoàn chỉnh, c. Span trái hoàn chỉnh, d. Span phải hoàn chỉnh. Cập nhật bảng nếu tìm thấy điểm số tốt hơn.
- **Lặp lại:** Tiếp tục quá trình này cho đến khi xét hết tất cả các span có thể.

```
1  for i : 0..n and all d, c
2      C[i][i][d][c] ← 0.0
3  for m : 1..n
4      for i : 0..n−m
5          j ← i+m
6          C[i][j][←][0] ← max_{i≤k<j} C[i][k][→][1] + C[k+1][j][←][1] + Score(j, i)
7          C[i][j][→][0] ← max_{i≤k<j} C[i][k][→][1] + C[k+1][j][←][1] + Score(i, j)
8          C[i][j][←][1] ← max_{i≤k<j} C[i][k][←][1] + C[k][j][←][0]
9          C[i][j][→][1] ← max_{i<k≤j} C[i][k][→][0] + C[k][j][→][1]
10 return C[0][n][→][1]
```

Hình 2.8: Mã giả thuật toán Eisner

2.1.2 Cách học hàm tính điểm *s*

Phương pháp sử dụng hand-crafted features và online learning

Phần này được trình bày dựa trên nội dung từ bài báo [4]. Trong bài báo, nhóm tác giả đề xuất một hướng tiếp cận mới, trong đó trọng số của một cạnh được gán nhãn được coi là một

hàm tuyến tính, được biểu diễn như sau:

$$s(i,j,r) = \mathbf{w} \cdot \mathbf{f}(i,j) \tag{2.6}$$

Trong đó, $\mathbf{w}$ là tham số cần được học trong quá trình huấn luyện, còn $\mathbf{f}(i,j)$ là véc tơ đặc trưng của cạnh nối từ đỉnh i đến đỉnh j. Các đặc trưng này thường được thiết kế thủ công (hand-crafted features) dựa trên hiểu biết về bài toán và dữ liệu cụ thể.

Cách học véc tơ tham số w: Nhóm tác giả đề xuất sử dụng thuật toán MIRA (Margin Infused Relaxed Algorithm) để học véc tơ tham số $\mathbf{w}$ [4]. MIRA thuộc nhóm các thuật toán học trực tuyến (online learning), nghĩa là việc cập nhật véc tơ $\mathbf{w}$ được thực hiện liên tục với từng mẫu dữ liệu theo thứ tự, không cần phải xử lý toàn bộ dữ liệu trước khi bắt đầu quá trình huấn luyện. Quy trình của một thuật toán học trực tuyến thường được mô tả như sau:

Tại mỗi bước của quá trình huấn luyện, thuật toán MIRA sẽ cập nhật giá trị $\mathbf{w}^{(i+1)}$ dựa trên giá trị hiện tại $\mathbf{w}^{(i)}$ bằng việc giải bài toán tối ưu sau:

$$\begin{aligned} &\min ||w^{(i+1)} - w^{(i)}|| \\ &\text{s.t.} \quad s(x_t, y_t) - s(x_t, y') \geq L(y_t, y') \quad \forall y' \in D^*(G_{x_t}) \end{aligned} \tag{2.7}$$

Ý tưởng chính của bài toán tối ưu này là đảm bảo rằng khi sử dụng véc tơ tham số $\mathbf{w}^{(i+1)}$ để tính điểm số (score), điểm số của nhãn đúng y_t phải cao hơn điểm số của tất cả các nhãn sai y' một khoảng ít nhất bằng giá trị lỗi $L(y_t, y')$. Trong ngữ cảnh bài toán phân tích cú pháp, hàm lỗi thường được chọn là số từ có quan hệ cha con sai lệch so với cây phân tích đúng. Do đó, giá trị lỗi lớn nhất mà một cây phụ thuộc có thể đạt được chính là độ dài của câu. Bài toán tối ưu trên thuộc nhóm bài toán quy hoạch bậc hai (quadratic programming problem) và có thể được giải sử dụng thuật toán Hildreth [**?**].

Tuy nhiên việc giải bài toán tối ưu với $|D^*(G_x)|$ ràng buộc có thể dẫn đến khó khăn về mặt tính toán do số lượng ràng buộc lớn. Để khắc phục vấn đề này, một biến thể của MIRA được tác giả đề xuất là Single-best MIRA [4]. Trong biến thể này, thay vì sử dụng tất cả các ràng buộc từ $D^*(G_x)$, chỉ ràng buộc tương ứng với cây y' có điểm số cao nhất được xét đến:

$$\begin{aligned} &\min ||\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}|| \\ &s.t.\, s(x_t, y_t) - s(x_t, y') \geq L(y_t, y') \\ &\text{where } y' = \arg\max_{y'} s(x_t, y') \end{aligned} \tag{2.8}$$

Cách chọn feature set f: Bài báo đề xuất ba cách chọn đặc trưng, được trình bày trong ba bảng trong hình 2.9:

Trong hai bảng a và b của hình 2.9, tập đặc trưng chỉ bao gồm các thông tin liên quan đến hai từ i và j, tương ứng với nút cha và nút con trong cây phụ thuộc. Do chỉ dựa trên thông tin

a)

Basic Uni-gram Features
p-word, p-pos
p-word
p-pos
c-word, c-pos
c-word
c-pos

b)

Basic Big-ram Features
p-word, p-pos, c-word, c-pos
p-pos, c-word, c-pos
p-word, c-word, c-pos
p-word, p-pos, c-pos
p-word, p-pos, c-word
p-word, c-word
p-pos, c-pos

c)

In Between POS Features
p-pos, b-pos, c-pos
Surrounding Word POS Features
p-pos, p-pos+1, c-pos-1, c-pos
p-pos-1, p-pos, c-pos-1, c-pos
p-pos, p-pos+1, c-pos, c-pos+1
p-pos-1, p-pos, c-pos, c-pos+1

Hình 2.9: Các đặc trưng được hệ thống sử dụng bao gồm: p-word (từ của nút cha trong cây phụ thuộc), c-word (từ của nút con), p-pos (nhãn từ loại của nút cha), c-pos (nhãn từ loại của nút con), p-pos+1 (nhãn từ loại của từ đứng ngay sau nút cha trong câu), p-pos-1 (nhãn từ loại của từ đứng ngay trước nút cha), c-pos+1 (nhãn từ loại của từ đứng ngay sau nút con), c-pos-1 (nhãn từ loại của từ đứng ngay trước nút con), b-pos (nhãn từ loại của từ nằm giữa nút cha và nút con).

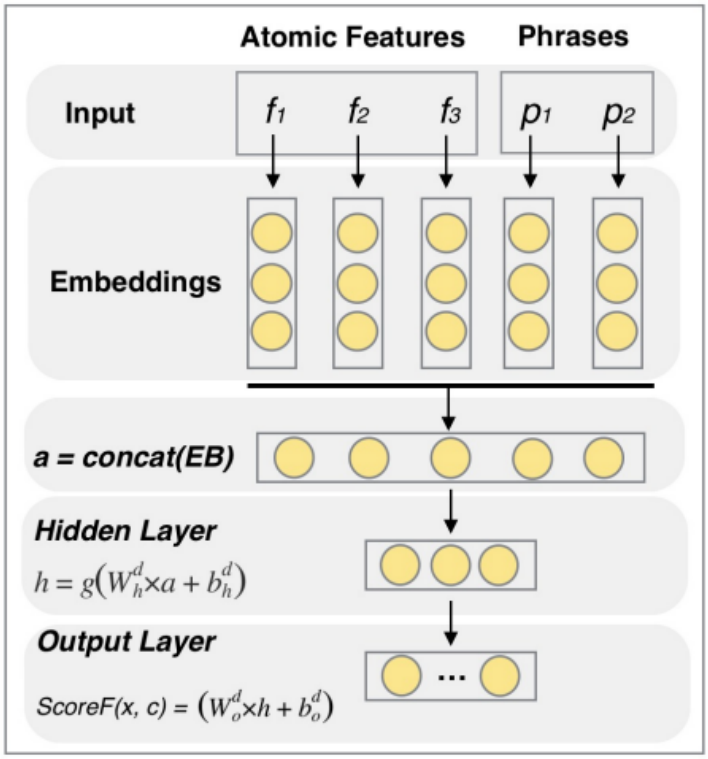
từ hai từ này, hiệu suất của mô hình không cao, bởi vì thông tin ngữ cảnh không được khai thác đầy đủ [4].

Để khắc phục hạn chế này, nhóm tác giả đã đề xuất một tập đặc trưng mở rộng, được trình bày trong bảng *c* của hình 2.9. Tập đặc trưng này không chỉ sử dụng thông tin của hai từ *i* và *j* mà còn bao gồm cả các từ nằm giữa và xung quanh hai từ đó. Việc mở rộng tập đặc trưng giúp mô hình nắm bắt được nhiều thông tin ngữ cảnh hơn, từ đó cải thiện đáng kể độ chính xác trong quá trình phân tích cú pháp [4].

Phương pháp Mạng Nơ-ron

Việc xây dựng thủ công hàng triệu đặc trưng để nắm bắt thông tin ngữ cảnh và cấu trúc trong đồ thị con không chỉ giới hạn khả năng khái quát hóa của mô hình mà còn làm chậm tốc độ phân tích cú pháp. Để giải quyết vấn đề này, nhiều nghiên cứu đã tìm cách sử dụng mạng nơ-ron để giảm bớt gánh nặng của việc thiết kế đặc trưng [2]. Một ví dụ điển hình là công trình được trình bày trong "A Fast and Accurate Dependency Parser using Neural Networks" [2], trong đó tác giả chỉ sử dụng một số đặc trưng nguyên tử của đồ thị con, sau đó ánh xạ chúng thành các véc-tơ và tự động học các thông tin hữu ích từ dữ liệu.

Phương pháp này không chỉ giảm thiểu yêu cầu về thiết kế đặc trưng mà còn giúp mô hình có khả năng khái quát hóa tốt hơn nhờ vào việc học trực tiếp từ dữ liệu thay vì dựa trên các quy tắc được xác định trước. Điều này đặc biệt quan trọng trong các bài toán phân tích cú pháp, nơi mà sự đa dạng và phức tạp của ngôn ngữ tự nhiên đòi hỏi một cách tiếp cận linh hoạt và hiệu quả hơn [2].



Hình 2.10: Kiến trúc của mạng nơ-ron được đề xuất trong [2]

2.2 Phân tích cú pháp phụ thuộc dựa vào bước chuyển

Phân tích cú pháp phụ thuộc dựa trên chuyển trạng thái (Transition-Based Dependency Parsing) là một trong những phương pháp phổ biến và hiệu quả để phân tích cấu trúc cú pháp của câu trong ngôn ngữ tự nhiên. Dưới đây là mô tả chi tiết về phương pháp này, bao gồm các bước xác định không gian ứng viên, huấn luyện mô hình, và quá trình phân tích.

Trong phân tích cú pháp dựa trên chuyển trạng thái, cây phụ thuộc của một câu được xây dựng dần dần thông qua một chuỗi các trạng thái (states). Mỗi trạng thái đại diện cho một phần của câu đã được phân tích. Không gian ứng viên là tập hợp tất cả các cây phụ thuộc có thể được tạo ra bằng cách áp dụng các chuyển trạng thái khác nhau từ một trạng thái ban đầu. Phương pháp này thường gồm ba phần chính:

- **Không gian trạng thái:** Tập hợp các trạng thái có thể có trong quá trình phân tích. Mỗi trạng thái biểu diễn một phần của cây phụ thuộc đang được xây dựng.
- **Hành động chuyển trạng thái (Transitions):** Các hành động chuyển trạng thái được áp dụng để di chuyển từ một trạng thái này sang trạng thái khác. Mỗi hành động tương ứng với việc thêm một cạnh vào cây phụ thuộc.
- **Mô hình dự đoán hành động:** Một mô hình học máy được huấn luyện để dự đoán hành

động chuyển trạng thái tiếp theo dựa trên trạng thái hiện tại.

2.2.1 Các thuật toán phân tích cú phụ phụ thuộc vào bước chuyển

Thuật toán Nvire[10]

Thuật toán Nivre là một trong những thuật toán nổi bật trong lĩnh vực phân tích phụ thuộc (dependency parsing). Đây là một loại phân tích cú pháp, nơi cấu trúc của câu được biểu diễn dưới dạng một cây với các từ làm các nút và các quan hệ ngữ pháp làm các cạnh nối giữa các từ.

Trong một hệ thống Nvire, cho tập $L = (r_0, ..., r_m)$ là tập nhãn phụ thuộc và một câu $x = (w_0, ..., w_n)$, một cấu hình phân tích cú pháp phụ thuộc là một bộ ba: $c = \{\delta, \beta, A\}$. Trong đó c chứa một stack δ, một buffer (queue) β và một tập các cung phụ thuộc A.

Nivre sử dụng một ngăn xếp (stack) và một hàng đợi (queue) để duyệt qua các từ trong câu và xây dựng cây phụ thuộc theo từng bước chuyển đổi:

- **Stack δ:** Chứa các từ từng được xét, và sẽ còn được xét tiếp.
- **Buffer β:** Chứa các từ chưa được xét, hoặc mới xét đến.
- **List A:** Chứa các quan hệ phụ thuộc đã tìm ra.

Các thao tác cơ bản của thuật toán:

- **SHIFT:** Chuyển từ ở đầu buffer lên đỉnh của stack, không thêm quan hệ.
- **RIGHT-ARC:** Thêm từ ở đầu buffer vào đỉnh stack, thêm quan hệ phụ thuộc.
- **LEFT-ARC:** Bỏ từ ở đỉnh stack ra, giữ nguyên buffer, thêm quan hệ phụ thuộc.
- **REDUCE:** Bỏ từ ở đỉnh stack, không thêm bất kì quan hệ nào.

Câu đầu vào $x = (w_0, ..., w_n)$.

Cấu hình khởi tạo :

- δ**:** Chỉ chứa ROOT.
- β**:** $w_1, w_2, w_3 \ldots, w_n$.
- A**:** $\emptyset$.

Cấu hình kết thúc:

- δ**:** Chỉ chứa ROOT.
- β**:** $\emptyset$.

- A**:** Chứa các quan hệ phụ thuộc.

Thuật toán được thực hiện như sau:

```
Input: sentence W, parameter-vector w
// Bước 1: Khởi tạo trạng thái ban đầu từ câu W
c ← INITIAL(W)
// Bước 2: Lặp lại cho đến khi trạng thái kết thúc
while not TERMINAL(c) do
    // Bước 3: Chọn hành động tối ưu
    t_opt ← arg max tLEGAL(c) w · (c, t)
    // Bước 4: Cập nhật trạng thái c bằng cách áp dụng t_opt
    c ← t_opt(c)
// Bước 5: Trả về chuỗi các hành động đã thực hiện
return Ac
```

Ví dụ: Câu đầu vào: **"Mèo ăn cá"**

Trong phân tích cú pháp phụ thuộc, nhãn phụ thuộc mô tả loại quan hệ giữa các từ trong câu. Ví dụ về các nhãn phụ thuộc có thể bao gồm:

- **nsubj**: Chủ ngữ
- **obj**: Tân ngữ
- **root**: Từ gốc của câu (thường là động từ chính)
- **amod**: Tính từ bổ nghĩa cho danh từ

Trạng thái ban đầu được thiết lập như sau:

- **Stack** (Ngăn xếp): rỗng `["Root"]`
- **Buffer** (Bộ đệm): `["Mèo", "ăn", "cá"]`
- **Arcs** (Các cung phụ thuộc): rỗng chỉ chứa mỗi Root `["Root"]`

2.2.2 Thực hiện từng bước

Bước 1: SHIFT

- **Mô tả**: Di chuyển từ "Mèo" từ Buffer vào Stack.
- **Stack**: `["Mèo"]`
- **Buffer**: `["ăn", "cá"]`

- **Arcs**: `[]`

Bước 2: LEFT-ARC

- **Mô tả**: Xóa "Mèo" khỏi Stack, thêm $(\text{ăn}, \text{nsubj}, \text{Mèo})$ vào tập Arcs
- **Stack**: `["Root"]`
- **Buffer**: `["ăn","cá"]`
- **Arcs**: `[(ăn, nsubj, Mèo)]`

Bước 3: RIGHT-ARC

- **Mô tả**: Thêm "ăn" từ Buffer vào Stack, thêm $(\text{Root}, \text{root}, \text{ăn})$ vào tập Arcs
- **Stack**: `["Root","ăn"]`
- **Buffer**: `["cá"]`
- **Arcs**: `[(ăn, nsubj, Mèo), (Root, root, ăn)]`

Bước 4: RIGHT-ARC

- **Mô tả**: Thêm "cá" từ Buffer vào Stack, thêm $(\text{ăn}, \text{dobj}, \text{cá})$ vào tập Arcs
- **Stack**: `["Root","ăn", "cá"]`
- **Buffer**: `[]`
- **Arcs**: `[(ăn, nsubj, Mèo), (Root, root, ăn), (ăn, dobj, cá)]`

Bước 5: REDUCE

- **Mô tả**: Xóa "cá" ra khỏi Stack
- **Stack**: `["Root","ăn"]`
- **Buffer**: `[]`
- **Arcs**: `[(ăn, nsubj, Mèo), (Root, root, ăn), (ăn, dobj, cá)]`

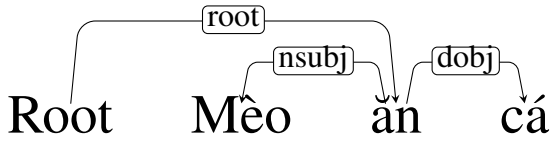
Bước 6: REDUCE

- **Mô tả**: Loại bỏ từ "ăn" khỏi Stack.
- **Stack**: `["Root"]`

- **Buffer**: `[]`
- **Arcs**: `[(ăn, nsubj, Mèo), (Root, root, ăn), (ăn, dobj, cá)]`

Hiện tại đã đến cấu hình cuối cùng, Stack chỉ có Root, Buffer rỗng. Hệ thống trả lại tập quan hệ phụ thuộc Arcs.

→ Thuật toán hoàn tất



Thuật toán Covington

Vào năm (2001), Covington trình bày một thuật toán "cơ bản" cho việc phân tích cú pháp phụ thuộc kết quả cuối cùng là một cây đơn(có gốc duy nhất) bao gồm tất cả các từ trong chuỗi đầu vào

Thuật toán Covington là một thuật toán kiểm tra mọi cặp từ trong câu để xác định các mối quan hệ phụ thuộc giữa chúng. Mục tiêu của thuật toán là xây dựng một cây phụ thuộc, trong đó mỗi từ trong câu sẽ có một từ chủ (head) và có thể có một hoặc nhiều từ phụ thuộc (dependents). Nó có thể sử dụng để xử lý các cấu trúc cú pháp phức tạp, bao gồm cả cấu trúc không xạ ảnh (non -projective).

Xạ ảnh (projective)

- xạ ảnh được định nghĩa một cách không chính thức là "không giao nhau". Chính thức hơn:
  - Cây có tính xạ ảnh khi và chỉ khi mỗi từ trong nó bao gồm một chuỗi con liên tục.
  - Một từ bao gồm một chuỗi con liên tục khi và chỉ khi, với hai từ bất kỳ mà nó bao gồm, nó cũng bao gồm tất cả các từ giữa chúng.
- Nó phải là một chuỗi con liên tục. Một chuỗi con liên tục là một chuỗi sao cho mọi thứ giữa bất kỳ phần tử nào của nó cũng là một phần của nó.

Để xây dựng tính xạ ảnh trong trình phân tích cú pháp phụ thuộc từ dưới lên, chúng ta cần đảm bảo các điều kiện sau:

1. Đừng bỏ qua một từ phụ thuộc tiềm năng của W. Nghĩa là, đính kèm mọi từ trước liên tiếp mà vẫn độc lập hoặc ngừng tìm kiếm.
2. Khi tìm kiếm phần đầu của W, chỉ xem xét từ trước đó, phần đầu của nó, phần đầu của từ

đó, v.v. cho đến gốc của cây.

Điều kiện **b** nói rằng nếu phần đầu của W (gọi là H) đứng trước W thì nó cũng phải chứa từ đứng ngay trước W; do đó, nó có thể truy cập được bằng cách trèo cây từ từ đó. Điều này tuân theo định nghĩa về tính xạ ảnh: chuỗi con H...W phải liên tục.

Ràng buộc **a** nói rằng các phụ thuộc trước của W là một chuỗi liên tục các từ vẫn độc lập tại thời điểm W được gặp. Hãy xem xét các từ mà ở bất kỳ giai đoạn nào vẫn không có *head*, tức là nội dung của Danh sách tiêu đề trong thuật toán phân tích cú pháp dựa trên danh sách. Mỗi từ như vậy là phần đầu của một thành phần, tức là một chuỗi con liên tục. Nghĩa là, mỗi từ vẫn độc lập là viết tắt của chuỗi từ mà nó bao gồm. Mục tiêu của trình phân tích cú pháp là tập hợp không hoặc nhiều chuỗi này thành một chuỗi liên tục kết thúc bằng W. Rõ ràng, nếu bất kỳ phần tử nào bị bỏ qua, chuỗi kết quả không thể liên tục. q.e.d.

Dưới đây là mã giả của giải thuật :

Initialize:

- Headlist := [] {Words that do not yet have heads}
- Wordlist := [] {All words encountered so far}

Repeat:

1. Accept a word and add it to Wordlist
    - W := the next word to be parsed
    - Wordlist := W + Wordlist
2. Look for dependents of W; they can only be consecutive elements of Headlist starting with the most r
    - **For** D := each element of Headlist, starting with the first
        - **begin**
        - **if** D can depend on W **then**
            - link D as dependent of W
            - delete D from Headlist
        - **else**
        - terminate this for loop
        - **end**

3. Look for the head of W; it must comprise the word preceding W

    - H := the word most recently preceding W in the input string that is not already subordinate to W
    - **loop**
        - **if** W can depend on H **then**
            - link W as dependent of H
            - terminate the loop
        - **if** H is independent then terminate the loop
        - H := the head of H
    - **end loop**

4. **if** no head for W was found then

    - Headlist := W + Headlist

Until all words have been parsed.

Chương 3

Thí nghiệm

3.1 Các công cụ sử dụng

Trong quá trình thực hiện thí nghiệm, chúng tôi đã sử dụng hai công cụ phân tích cú pháp phổ biến: MSTParser và MaltParser. Hai công cụ này được lựa chọn vì khả năng cung cấp hai phương pháp tiếp cận khác nhau (dựa vào đồ thị và dựa vào bước chuyển) cho bài toán phân tích cú pháp phụ thuộc [7, 9].

3.1.1 MSTParser

MSTParser là một công cụ mã nguồn mở viết bằng Java, được phát triển bởi Ryan McDonald để thực hiện các thí nghiệm trong bài báo [7]. Công cụ này được thiết kế chuyên biệt cho phân tích cú pháp phụ thuộc dựa trên lý thuyết đồ thị và hỗ trợ định dạng dữ liệu đầu vào chuẩn CoNLL-X, nơi mỗi từ trong câu được gán nhãn từ loại (POS) và thông tin phụ thuộc [1].

Một trong những điểm mạnh của MSTParser là khả năng tùy chỉnh linh hoạt, cho phép người dùng lựa chọn giữa hai thuật toán chính để tìm cây bao trùm: thuật toán **Eisner** cho cây xạ ảnh (projective) [6] và thuật toán **Chu-Liu/Edmonds** cho cây không xạ ảnh (non-projective) [3, 5]. Thuật toán Eisner là lựa chọn tốt khi yêu cầu cây bao trùm tuân thủ các ràng buộc ngữ pháp xạ ảnh, trong khi Chu-Liu/Edmonds được sử dụng khi không có yêu cầu này, giúp đảm bảo phân tích chính xác ngay cả khi cấu trúc câu phức tạp hơn.

Ngoài ra, MSTParser còn cung cấp tùy chọn giữa hai phương pháp: **first-order** và **second-order**. Phương pháp first-order tập trung vào các mối quan hệ trực tiếp giữa các từ trong câu, chỉ xét đến các cặp từ liền kề, giúp giảm thiểu độ phức tạp tính toán và tăng tốc độ xử lý. Ngược lại, phương pháp second-order xem xét thêm ngữ cảnh của các từ liên quan khác trong câu, từ đó cải thiện độ chính xác của phân tích nhưng cũng đòi hỏi nhiều tài nguyên tính toán hơn [7].

Trong bài báo cáo này, chúng tôi sẽ thực hiện các thí nghiệm với phương pháp first-order trên cả hai thuật toán Chu-Liu/Edmonds và Eisner để đánh giá hiệu quả của chúng trong phân tích cú pháp phụ thuộc.

3.1.2 MaltParser

MaltParser là một hệ thống phân tích cú pháp phụ thuộc (dependency parsing) được phát triển bởi Joakim Nivre và nhóm nghiên cứu của ông tại Đại học Uppsala, Thụy Điển.

Hệ thống của MaltParser hoạt động qua hai giai đoạn chính: giai đoạn huấn luyện và giai đoạn phân tích cú pháp. Trong giai đoạn huấn luyện, hệ thống cung cấp một tập hợp các câu với phân tích cú pháp đúng để đào tạo mô hình, và huấn luyện các lớp dữ liệu này. Trong giai đoạn phân tích, hệ thống nhận một tập hợp các câu và một lớp mô hình đã đào tạo, sau đó phân tích cú pháp các câu này dựa trên mô hình học được.

MaltParser thực hiện việc huấn luyện để đưa ra lựa chọn cho phân tích cú pháp phụ thuộc bằng cách sử dụng các thuật toán như **SVM**, **LIBLINEAR**, hoặc **TiMBL**. Để phân tích, MaltParser cung cấp 35 thuật toán phân tích khác nhau, bao gồm thuật toán **Nivre** và thuật toán **Covington**. Thuật toán Nivre là thuật toán dựa trên chuyển đổi và phù hợp cho các đồ thị phụ thuộc không vòng, trong khi thuật toán Covington thích hợp cho các đồ thị phụ thuộc có vòng. Mặc định, MaltParser sử dụng thuật toán Nivre và SVM cho việc phân tích và huấn luyện, vì thường mang lại hiệu quả cao.

3.2 Dữ liệu

Dữ liệu mà chúng tôi tiến hành thí nghiệm là bộ dữ liệu **VnDT: A Vietnamese dependency treebank** Bộ dữ liệu này bao gồm các câu tiếng Việt đã được gán nhãn với cấu trúc phụ thuộc cú pháp, phù hợp cho các bài toán phân tích cú pháp phụ thuộc. Bộ dữ liệu được tạo ra từ nhiều nguồn khác nhau, bao gồm văn bản báo chí, văn học, và các văn bản chuyên môn, nhằm cung cấp một cái nhìn toàn diện về cấu trúc cú pháp của tiếng Việt.

3.3 Nhãn Dữ liệu

3.3.1 Kho ngữ liệu tiếng Việt - Viettreebank

Tập nhãn mà chúng tôi sự dũng cũng được xây dựng trên tập dữ liệu Vietreebank[8]. tập nhãn của Vietreebank được thiết kế gồm có:

- Tập nhãn từ loại: Về nguyên tắc, các thông tin về từ có thể được chứa trong nhãn từ loại bao gồm: từ loại cơ sở (danh từ, động từ,...), thông tin hình thái (số ít, số nhiều, thì,

ngôi,...), thông tin về phân loại con (ví dụ động từ đi với danh từ, động từ đi với mệnh đề,...), thông tin ngữ nghĩa, hay một số thông tin cú pháp khác. Với đặc điểm của tiếng Việt, tập nhãn từ loại chỉ chứa thông tin về từ loại cơ sở mà không bao gồm các thông tin như hình thái, phân loại con,... Bảng 3.1

- Tập nhãn các thành phần cú pháp: Tập nhãn này chứa các nhãn mô tả các thành phần cú pháp cơ bản là cụm từ và mệnh đề. Nhãn thành phần cú pháp là thông tin cơ bản nhất trên cây cú pháp, nó tạo thành xương sống của cây cú pháp.

  Các nhãn cụm từ của tiếng Việt được đưa ra trong Bảng 3.2

  Các nhãn mệnh đề của tiếng Việt được đưa ra trong Bảng3.3

- Tập nhãn chức năng ngữ pháp: Nhãn chức năng của một thành phần cú pháp cho biết vai trò của nó trong thành phần cú pháp mức cao hơn. Nhãn chức năng cú pháp được gán cho các thành phần chính trong câu như chủ ngữ, vị ngữ, tân ngữ. Nhờ thông tin do nhãn chức năng cung cấp ta có thể xác định các loại quan hệ ngữ pháp cơ bản sau đây: – Chủ-vị – Đề-thuyết – Phần chêm – Bổ ngữ – Phụ ngữ – Sự kết hợp

Bảng 3.1: Tập nhãn từ loại tiếng Việt

STT	**Tên**	**Chú thích**
1	N	Danh từ
2	Np	Danh từ riêng
3	Nc	Danh từ chỉ loại
4	Nu	Danh từ đơn vị
5	V	Động từ
6	A	Tính từ
7	P	Đại từ
8	L	Định từ
9	M	Số từ
10	R	Phụ từ
11	E	Giới từ
12	C	Liên từ
13	I	Thán từ
14	T	Trợ từ, tiểu từ, từ tình thái
15	U	Từ đơn lẻ
16	Y	Từ viết tắt
17	X	Các từ không phân loại được

Bảng 3.2: Tập nhãn cụm từ tiếng Việt

STT	Tên	Chú thích
1	NP	Cụm danh từ
2	VP	Cụm động từ
3	AP	Cụm tính từ
4	RP	Cụm phụ từ
5	PP	Cụm giới từ
6	QP	Cụm từ chỉ số lượng
7	MDP	Cụm từ tình thái
8	WHNP	Cụm danh từ nghi vấn (ai, cái gì, con gì,...)
9	WHAP	Cụm tính từ nghi vấn (lạnh thế nào, đẹp ra sao,...)
10	WHRP	Cụm từ nghi vấn dùng khi hỏi về thời gian, nơi chốn,...
11	WHPP	Cụm giới từ nghi vấn (với ai, bằng cách nào,...)

Bảng 3.3: Tập nhãn mệnh đề tiếng Việt

STT	Tên	Chú thích
1	S	Câu trần thuật (khẳng định hoặc phủ định)
2	SQ	Câu hỏi
3	SBAR	Mệnh đề phụ (bổ nghĩa cho danh từ, động từ, và tính từ)

Bảng 3.4: Tập nhãn chức năng cú pháp tiếng Việt.

STT	Tên
1	SUB: Nhãn chức năng chủ ngữ
2	DOB: Nhãn chức năng tân ngữ trực tiếp
3	IOB: Nhãn chức năng tân ngữ gián tiếp
4	TPC: Nhãn chức năng chủ đề
5	PRD: Nhãn chức năng vị ngữ không phải cụm động từ
6	LGS: Nhãn chức năng chủ ngữ logic của câu ở thể bị động
7	EXT: Nhãn chức năng bổ ngữ chỉ phạm vi hay tần suất của hành động
8	HN: Nhãn phần tử trung tâm (của cụm từ hoặc mệnh đề)
9-12	TC, CMD, EXC, SPL: Nhãn phân loại câu: đề-thuyết, mệnh lệnh, cảm thán, đặc biệt
13	TTL: Tít báo hay tiêu đề
14	VOC: Thành phần than gọi

3.3.2 Tập nhãn quan hệ phụ thuộc đa ngôn ngữ

Tập nhãn quan hệ phụ thuộc đa ngôn ngữ (Universal Dependency - UD) được xây dựng bởi nhóm nghiên cứu của trường đại học Stanford là Marneffe và cộng sự là tập hợp các nhãn được sử dụng để biểu diễn các mối quan hệ phụ thuộc cú pháp giữa các từ trong câu, áp dụng cho nhiều ngôn ngữ khác nhau. Tập nhãn này thường được sử dụng trong các hệ thống phân tích cú pháp phụ thuộc để giúp phân tích cấu trúc cú pháp của câu trong các ngôn ngữ khác nhau.

Tập nhãn phụ thuộc Stanford đã được xây dựng dựa vào những ý tưởng mô tả mối quan hệ ngữ pháp chung có thể thấy trong nhiều ngôn ngữ khác nhau. Tập nhãn này được tổ chức theo các nhóm về chủ ngữ, tân ngữ, các mệnh đề, từ hạn định của danh từ, hoặc các từ bổ nghĩa cho danh từ,... Stanford đưa ra gần 50 loại quan hệ phụ thuộc cho tiếng Anh dựa vào kho ngữ liệu PennTreebank.Mỗi một quan hệ được đưa ra bởi ba thành phần: tên quan hệ phụ thuộc, từ trung tâm và từ phụ thuộc.

Nhóm tác giả đã xây dựng một tập nhãn đa ngôn ngữ gồm có 40 nhãn. Được chia thành các nhóm sau:

- Những phụ thuộc cốt lõi của vị từ: nsubj, csubj, nsubjpass, csubjpass, dobj,22ccomp, xcomp, iobj.
- Những phụ thuộc không cốt lõi của vị từ: nmod, advcl, advmod, neg.
- Những phụ thuộc mệnh đề đặc biệt: vocative, aux, mark, discourse, auxpass, punct, expl, cop.
- Những phụ thuộc danh từ: nummod, acl, amod, appos, det, nmod, neg.
- Những phụ thuộc về các từ không thể phân tích và các nhóm từ ghép: compound, mwe, goeswith, name, foreign.
- Những phụ thuộc về sự liên hợp: conj, cc, punct.
- Những phụ thuộc về sợ sở hữu, các giới từ, hoặc các trường hợp đặc biệt được đánh dấu: case.
- Những phụ thuộc về các thành phần tham gia: list, parataxis, remnant, dislocated, reparandum.
- Và những phụ thuộc khác: root, dep.

Ví dụ: Bell, based in Los Angeles, makes and distributes electronic, com- puter and building products.

Các quan hệ phụ thuộc của Stanford đưa ra cho câu trên là:

```
nsubj(makes-8, Bell-1)
nsubj(distributes-10, Bell-1)
vmod(Bell-1, based-3)
nn(Angeles-6, Los-5)
prep in(based-3, Angeles-6)
root(ROOT-0, makes-8)
conj and(makes-8, distributes-10)
amod(products-16, electronic-11)
conj and(electronic-11, computer-13)
amod(products-16, computer-13)
conj and(electronic-11, building-15)
amod(products-16, building-15)
dobj(makes-8, products-16)
dobj(distributes-10, products-16)
```

Các quan hệ trên đều có ý nghĩa Ví dụ: quan hệ phụ thuộc dobj(makes-8, products-16) có nghĩa là: products là tân ngữ trực tiếp của makes.

3.4 Kết quả và phân tích

Kết quả phân tích cú pháp phụ thuộc trên bộ dữ liệu Viettreebank của hai hướng tiếp cận dựa trên đồ thị và bước chuyển được trình bày ở bảng 3.5.

Category	**Variants**	**UAS**	**LAS**
Graph-based	Eisner	**77.44**	71.70
	Chu-Liu-Edmonds	75.80	69.80
Transition-based	Nivre-eager	72.94	**72.46**
	Nivre-standard	60.97	60.61
	Covington non-projective	71.72	60.05
	Covington projective	71.76	60.08

Bảng 3.5: Kết quả phân tích cú pháp phụ thuộc trên bộ dữ liệu Viettreebank [8]

Nhìn chung, các phương pháp dựa trên đồ thị (đặc biệt là Eisner) có xu hướng đạt hiệu suất cao hơn so với các phương pháp dựa trên bước chuyển trong phân tích cú pháp phụ thuộc cho bộ dữ liệu Viettreebank.

Trong nhóm phương pháp dựa trên bước chuyển, Nivre-eager có hiệu suất gần bằng Eisner, đặc biệt nổi bật ở khả năng gán nhãn với LAS gần bằng UAS, cho thấy đây là một lựa chọn mạnh mẽ khi cần cân bằng giữa tốc độ và độ chính xác. Ngược lại, Nivre-standard và các biến thể Covington có hiệu suất thấp hơn, đặc biệt là ở LAS, điều này có thể chỉ ra rằng các phương pháp này gặp khó khăn hơn trong việc duy trì độ chính xác cao trong gán nhãn.

Chương 4

Kết luận

Trong bài báo cáo này, chúng tôi đã trình bày lý thuyết và so sánh kết quả thực nghiệm của hai phương pháp phân tích cú pháp phụ thuộc phổ biến: dựa trên đồ thị và dựa trên bước chuyển.

Những kết quả này khẳng định rằng trong các ứng dụng đòi hỏi độ chính xác cao, phương pháp dựa trên đồ thị, đặc biệt với thuật toán Eisner, là lựa chọn ưu việt. Tuy nhiên, phương pháp dựa trên bước chuyển vẫn có vai trò quan trọng trong các tình huống yêu cầu cân bằng giữa tốc độ và độ chính xác, từ đó nhấn mạnh tầm quan trọng của việc lựa chọn đúng phương pháp phân tích cú pháp phụ thuộc tùy theo đặc điểm cụ thể của ngữ liệu và yêu cầu của ứng dụng.

Tài liệu tham khảo

[1] Travis Brown. Mstparser. https://github.com/travisbrown/mstparser, 2008. Accessed: 2024-08-11.

[2] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[3] Y J Chu and T H Liu. On the shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400, 1965.

[4] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[5] Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71:233–240, 1967.

[6] Jason Eisner. Three new probabilistic models for dependency parsing: An exploration. *Proceedings of the 16th conference on Computational linguistics-Volume 1*, pages 340–345, 1996.

[7] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530, 2005.

[8] Dat Quoc Nguyen, Dai Quoc Nguyen, Son Bao Pham, Phuong-Thai Nguyen, and Minh Le Nguyen. From Treebank Conversion to Automatic Dependency Parsing for Vietnamese. In *Proceedings of 19th International Conference on Application of Natural Language to Information Systems*, pages 196–207, 2014.

[9] Joakim Nivre, Johan Hall, and Jens Nilsson. *MaltParser: A data-driven parser-generator for dependency parsing*. International Computer Science Institute, 2007.

[10] Joakim Nivre and Mario Scholz. Deterministic dependency parsing of English text. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 64–70, Geneva, Switzerland, aug 23–aug 27 2004. COLING.

Bảng 4.1: NHIỆM VỤ CỦA CÁC THÀNH VIÊN

Họ và Tên	**Công việc**
Nguyễn Công Hiếu	+ Tìm hiểu tổng quan phân tích cú pháp phụ thuộc. + Tìm hiểu phân tích cú pháp dựa trên đồ thị. + Tìm hiểu thuật toán Eisner. + Phân tích kết quả về đánh giá. + Làm slide. + Làm báo cáo.
Hoàng Đăng Khoa	+ Tìm hiểu và lựa chọn bộ dữ liệu. + Tìm hiểu phân tích cú pháp dựa trên đồ thị. + Tìm hiểu thuật toán Chu-Liu-Edmonds + Tìm hiểu và sử dụng công cụ MSTParser + Làm slide + Làm báo cáo
Long Trí Thái Sơn	+ Tìm hiểu dữ liệu + Tìm hiểu phân tích cú pháp dựa trên bước chuyển + Tìm hiểu thuật toán Nvire + Tìm hiểu và sử dụng công cụ MaltParser + Làm slide + Làm báo cáo
Trịnh Đắc Phú	+ Tìm hiểu và phân tích nhãn cho tiếng Việt + Tìm hiểu thuật toán Covington + Tìm hiểu phân tích cú pháp dựa trên bước chuyển + Làm slide. + Làm báo cáo