TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO MÔN HỌC
XỬ LÝ NGÔN NGỮ TỰ NHIÊN

SPEECH AND LANGUAGE PROCESSING

Giảng viên hướng dẫn: **TS. Nguyễn Kiêm Hiếu**

Học viên thực hiện: **Nguyễn Đức Thắng**

Mã số học viên: **20166769**

HÀ NỘI, 07/2020

Mục lục

1 Hidden Markov Models **3**
- 1.1 Xích Markov . . . 3
- 1.2 Mô hình Markov ẩn . . . 5
- 1.3 Thuật toán Forward . . . 7
  - 1.3.1 Khởi tạo (Initialization) . . . 8
  - 1.3.2 Đệ quy (Recursion) . . . 8
  - 1.3.3 Kết thúc (Termination) . . . 10
- 1.4 Thuật toán Backward . . . 11
  - 1.4.1 Khởi tạo (Initialization) . . . 11
  - 1.4.2 Đệ quy (Recursion) . . . 11
  - 1.4.3 Kết thúc (Termination) . . . 14
- 1.5 Decoding: Thuật toán Viterbi . . . 15
  - 1.5.1 Khởi tạo (Initialization) . . . 16
  - 1.5.2 Đệ quy (Recursion) . . . 17
  - 1.5.3 Kết thúc (Termination) . . . 19
  - 1.5.4 Quay lui (Backtracking) . . . 19
- 1.6 Thuật toán Forward-Backward . . . 20

2 Part-of-Speech Tagging **23**
- 2.1 Bài toán gán nhãn từ loại . . . 23
- 2.2 Hidden Markov Model POS Tagging . . . 25
  - 2.2.1 Phương trình cơ bản của HMM tagging . . . 25
  - 2.2.2 Ước lượng các xác suất . . . 26
  - 2.2.3 Ví dụ và thuật toán . . . 27
  - 2.2.4 Mở rộng HMM cho Trigrams . . . 30

3 Formal Grammars of English **32**
- 3.1 Văn phạm . . . 32
  - 3.1.1 Định nghĩa văn phạm . . . 32
  - 3.1.2 Ngôn ngữ sinh bởi văn phạm . . . 34
- 3.2 Phân loại văn phạm . . . 34
  - 3.2.1 Văn phạm không hạn chế . . . 35

3.2.2 Văn phạm cảm ngữ cảnh . . . 35
3.2.3 Văn phạm phi ngữ cảnh . . . 35
3.2.4 Văn phạm chính quy . . . 36
3.3 Một số quy tắc ngữ pháp trong tiếng Anh . . . 37
3.3.1 Sentence-Level Constructions . . . 37
3.3.2 The Noun Phrase . . . 38
3.3.3 The Verb Phrase . . . 41
3.3.4 Coordination . . . 42
3.4 Treebanks . . . 43
3.5 Grammar Equivalence and Normal Form . . . 45

4 Syntactic Parsing **47**
4.1 Ambiguity . . . 47
4.2 CKY Parsing: A Dynamic Programming Approach . . . 48
4.2.1 Rút gọn văn phạm . . . 48
4.2.2 Chuyển CFG sang dạng chuẩn Chomsky . . . 51
4.2.3 Thuật toán CYK . . . 53
4.3 Phân tích cú pháp một phần (Partial Parsing) . . . 57
4.3.1 Tiếp cận dựa trên học máy cho Chunking . . . 57
4.3.2 Đánh giá hệ thống Chunking . . . 58

5 Statistical Parsing **60**
5.1 Văn phạm phi ngữ cảnh xác suất . . . 60
5.1.1 PCFG for Disambiguation . . . 62
5.1.2 PCFG for Language Modeling . . . 63
5.2 Probabilistic CKY Parsing of PCFG . . . 64
5.3 Các vấn đề với PCFG . . . 66
5.4 Văn phạm phi ngữ cảnh xác suất từ vựng (Probabilistic Lexicalized CFGs) 67
5.4.1 Phân tích cú pháp Collins . . . 67
5.4.2 Nâng cao: Các bổ sung của trình phân tích Collins . . . 70
5.5 Đánh giá độ chính xác . . . 71

6 Dependency Parsing **73**
6.1 Cú pháp phụ thuộc . . . 73
6.1.1 Định nghĩa cú pháp phụ thuộc . . . 73
6.1.2 Bài toán phân tích cú pháp phụ thuộc . . . 75
6.1.3 Biểu diễn cú pháp phụ thuộc . . . 75
6.2 Các thuật toán phân tích cú pháp phụ thuộc . . . 77
6.2.1 Phân tích cú pháp phụ thuộc dựa trên bước chuyển . . . 78
6.2.2 Phân tích cú pháp phụ thuộc dựa trên đồ thị . . . 82

Chương 1

Hidden Markov Models

1.1 Xích Markov

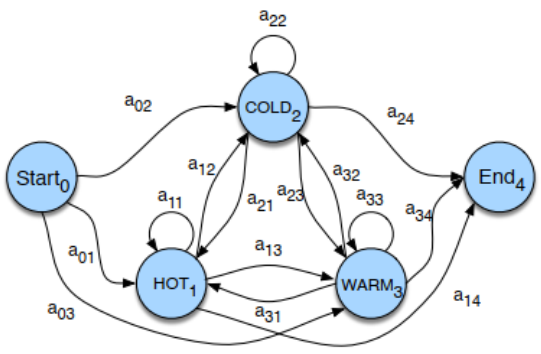
Quá trình Markov là quá trình mà không lưu giữ kí ức về các trạng thái của nó trong quá khứ, hay nói cách khác là chỉ có trạng thái hiện tại của quá trình mới có ảnh hưởng đến diễn tiến của nó trong tương lai. Quá trình Markov ngày một trở nên quan trọng vì hai lý do cơ bản:

1. Rất nhiều các hiện tượng trong vật lý, sinh vật, kinh tế, tài chính, xã hội có thể được mô hình bởi quá trình Markov.

2. Sự phát triển mạnh mẽ của lý thuyết cho phép chúng có thể được thực hiện tốt các thao tác tính toán, phân tích, dự báo trên mô hình.

Quá trình Markov mà thời gian rời rạc gọi là **xích Markov/chuỗi Markov**.

Chuỗi Markov còn được gọi là **mô hình Markov hiện**. Cả chuỗi Markov ẩn và hiện đều là mở rộng của Otomat hữu hạn được. Ta gọi **weighted finite automaton** là một tập các trạng thái (states) và một tập các phép chuyển (transition) giữa các trạng thái, mỗi phép chuyển có một trọng số (weight). Một chuỗi Markov là một trường hợp đặc biệt của weighted automaton với các trọng số ở đây là các xác suất (tổng xác xuất ra khỏi một nút phải bằng 1).



Hình 1.1: Chuỗi Markov cho thời tiết

Hình 1.1 mô phỏng chuỗi Markov cho các sự kiện thời tiết với 3 trạng thái cold, hot và warm. Trạng thái start và end là 2 trạng thái đặc biệt lần lượt gọi là **trạng thái bắt đầu** và **trạng thái kết thúc**. Một chuỗi Markov được biểu diễn với các thành phần sau:

- Q = $\{q_1, q_2, \ldots, q_N\}$: Một tập N trạng thái.
- $A = \begin{bmatrix} a_{01} & \cdots & a_{0n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$: **Ma trận xác suất chuyển**, mỗi a_{ij} biểu diễn xác suất chuyển từ trạng thái i sang trạng thái j, và có $\sum_{j=1}^{n} a_{ij} = 1, \forall i$
- q_0, q_F: trạng thái bắt đầu và trạng thái kết thúc.

Hình 1.1 biểu diễn các trạng thái (bao gồm trạng thái bắt đầu và trạng thái kết thúc) như các đỉnh của đồ thị, các phép chuyển như các cạnh có hướng và có trọng số của đồ thị. Một chuỗi Markov thể hiện một giả định quan trọng về các xác suất này. Trong chuỗi Markov bậc 1 (**first-order Markov chain**), xác suất một trạng thái nào đó chỉ phụ thuộc vào trạng thái trước. Ta có **giả thuyết Markov** như sau:

$$P(q_i|q_1, \ldots, q_{i-1}) = P(q_i|q_{i-1})$$

trong đó $a_{ij} = P(q_j|q_i)$. Thông thường, thay vì sử dụng trạng thái bắt đầu và trạng thái kết thúc, chuỗi Markov sử dụng **phân phối ban đầu** $\pi = (\pi_1, \ldots, \pi_n)$ để thay thế, với π_i biểu diễn xác suất để xuất phát ở trạng thái i (hay $p(q_i|START)$) và có $\sum_{i=1}^{n} \pi_i = 1$.

Ví dụ 1. *Với hình 1.1, xác suất xảy ra chuỗi "hot cold hol cold" là:* $\pi_1 a_{12} a_{21} a_{12}$

Ví dụ 2. *Cho mô hình Markov gồm 3 trạng thái mô tả thời tiết như sau:*

Thời tiết quan sát được có thể thuộc các trạng thái {mưa, mây, nắng} đánh số tương ứng là {1,2,3}. Ma trận xác suất chuyển trạng thái:

$$A = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

Cho trước trạng thái ban đầu ($t = 1$) là "nắng". Tìm xác suất để chuỗi 7 ngày tiếp theo là "nắng, nắng, mưa, mưa, nắng, mây, mưa".
Gọi O là chuỗi thời tiết quan sát được trong 8 ngày:

$$O = \{\text{nắng, nắng, nắng, mưa, mưa, nắng, mây, mưa}\}$$

tương ứng với $t = 1, 2, \ldots, 8$. Ta cần xác định $P(O|Markov)$ - xác suất của O cho mô hình Markov:

$$P(O|Markov) = P(\text{nắng, nắng, nắng, mưa, mưa, nắng, mây, mưa}|Markov)$$

Do tính chất Markov nên ta có:

$$\begin{aligned} P(O|Markov) &= P(\text{nắng}).P(\text{nắng}|\text{nắng}).P(\text{nắng}|\text{nắng}).P(\text{mưa}|\text{nắng}). \\ &\quad P(\text{mưa}|\text{mưa}).P(\text{nắng}|\text{mưa}).P(\text{mây}|\text{nắng}).P(\text{mưa}|\text{mây}) \\ &= \pi_3.a_{33}.a_{33}.a_{31}.a_{11}.a_{13}.a_{32}.a_{23} \\ &= 1 \times 0.8 \times 0.8 \times 0.4 \times 0.3 \times 0.1 \times 0.2 \\ &= 1.536 \times 10^{-4} \end{aligned}$$

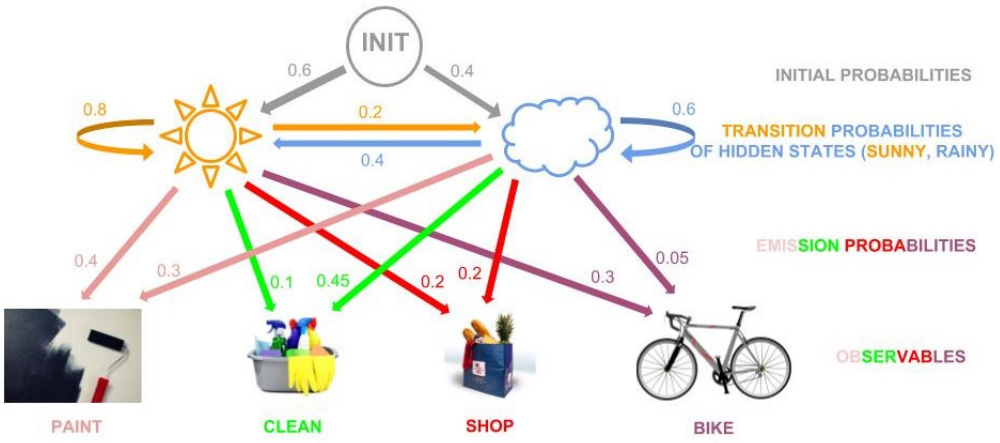
1.2 Mô hình Markov ẩn

Mô hình Markov hữu ích khi chúng ta chúng ta tính toán xác suất cho một chuỗi các sự kiện mà chúng ta có thể quan sát. Tuy nhiên, trong nhiều trường hợp, các sự kiện không thể quan sát trực tiếp nữa. Ví dụ như bài toán gán nhãn từ loại thì các tag như noun, verb, ... là không thể quan sát, chúng ta chỉ nhìn vào các từ và suy ra các thẻ đúng của các từ, trong bài toán này, các thẻ từ loại được gọi là ẩn (hidden).

Mô hình Markov ẩn (Hidden Markov Model) (gọi tắt là **HMM**) là kết quả mở rộng của mô hình Markov bằng cách mỗi trạng thái được gán với một **hàm phát xạ quan sát (observation distribution)**. Ngoài các quá trình ngẫu nhiên chuyển giữa các trạng thái, tại mỗi trạng thái còn sinh ra một quá trình ngẫu nhiên nữa đó là **quá trình ngẫu nhiên sinh ra một quan sát**. Như vậy, trong mô hình Markov ẩn có một quá trình ngẫu nhiên kép, trong đó có một quá trình ngẫu nhiên không quan sát được.
Một số ký hiệu:

- T: Độ dài chuỗi quan sát.
- N: Số trạng thái trong mô hình.
- M: Số sự kiện quan sát.
- $Q = \{q_1, q_2, \ldots, q_N\}$: Các trạng thái ẩn của quá trình Markov.
- $V = \{v_1, v_2, \ldots, v_M\}$: Tập quan sát khả thi.
- A: ma trận xác suất chuyển trạng thái ẩn hay gọi là **Transition matrix**.
- B: ma trận xác suất chuyển trạng thái quan sát hay gọi là **Emision matrix**.
- π: phân phối xác suất các trạng thái ban đầu.
- $O = \{o_1, o_2, \ldots, o_T\}$: Chuỗi quan sát.

Ví dụ 3. *Xét mô hình Markov như hình 1.2.*
Trong đó, các sự kiện thời tiết nắng (sunny) hoặc mưa (rainy) là các trạng thái ẩn. Chúng ta chỉ có thể quan sát hành động của một người như đi vẽ (paint), dọn dẹp (clean), đi mua sắm (shop), đi xe đạp (bike), ... Từ các quan sát này, chúng ta phải dự đoán ra thời tiết.



Hình 1.2: Ví dụ về chuỗi Markov ẩn

Trong hình 1.2 thì ta có:

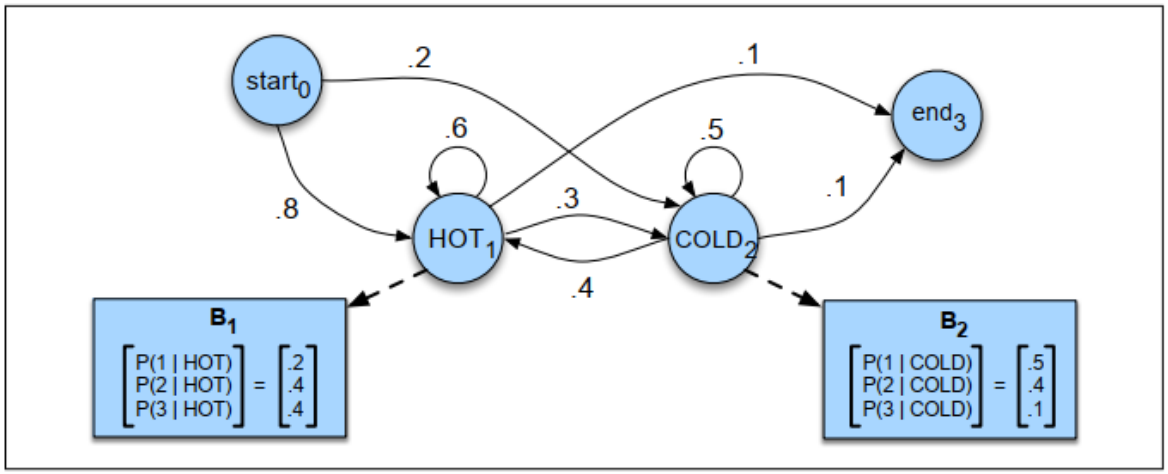
- Q: Các trạng thái ẩn là {sunny, rainy} , $N = 2$.
- V: Tập quan sát khả thi là {paint, clean, shop, bike}, $M = 4$.
- π: Phân phối ban đầu $\pi = (0.6, 0.4)$.
- A: Ma trận xác suất chuyển

$$A = \begin{bmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{bmatrix}$$

- B: Emision matrix

$$B = b_i(o_t) = \begin{bmatrix} 0.4 & 0.1 & 0.2 & 0.3 \\ 0.3 & 0.45 & 0.2 & 0.05 \end{bmatrix}$$

Ví dụ 4. *Giả sử chúng ta có thể quan sát số kem Jason ăn mỗi ngày nhưng không thể biết thời tiết nơi Jason ở như thế nào. Cho một chuỗi các quan sát O, mỗi quan sát tương ứng là một số nguyên biểu diễn số kem Jason ăn trong ngày, các trạng thái ẩn là thời tiết nóng (hot - H) hoặc lạnh (cold - C). Hình 1.3 là ví dụ về một HMM cho bài toán này.*



Hình 1.3: Mô hình Markov ẩn biểu thị mỗi quan hệ giữa kem mà Jason ăn với thời tiết

Với HMM, chúng ta có 3 bài toán cơ bản sau:

1. **Bài toán Likelihood**: Cho HMM $\lambda = (A, B)$ và một chuỗi các quan sát O. Xác định xác suất $P(O|\lambda)$.
2. **Bài toán Decoding**: Cho một chuỗi các quan sát O và một HMM $\lambda = (A, B)$. Xác định dãy trạng thái tốt nhất Q.
3. **Bài toán Learning**: Cho một chuỗi quan sát O và một tập các trạng thái trong HMM, học các tham số A và B của HMM.

1.3 Thuật toán Forward

Thuật toán Forward để giải quyết bài toán Likelihood, bao gồm 3 bước chính:

- Khởi tạo (Initialization)
- Đệ quy (Recursion)
- Kết thúc (Termination)

Để dễ trình bày, chúng ta sẽ đi từ ví dụ sau đó đưa ra thuật toán. Gọi $\alpha_t(j)$ là biến chuyển tiếp (forward variable) có dạng:

$$\alpha_t(j) = p(o_1, o_2, \ldots, o_t, q_t = j|\lambda)$$

Với $q_t = j$ có nghĩa là trạng thái thứ t trong chuỗi các trạng thái là trạng thái j.
Giả sử với ví dụ 3, chúng ta có chuỗi quan sát: paint, clean, shop, bike.

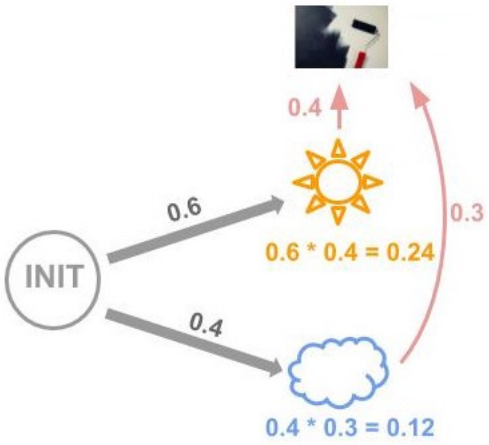
$$O = \{O_{\text{paint}}, O_{\text{clean}}, O_{\text{shop}}, O_{\text{bike}}\}$$

Chúng ta cần tính một cách hiệu quả $P(O|\lambda)$. Có 2 phương pháp để giải quyết bài toán này là thuật toán Forward và thuật toán Backward. Trong phần này, chúng ta sẽ tìm hiểu về thuật toán Forward.

1.3.1 Khởi tạo (Initialization)

$$\alpha_1(j) = \pi_j b_j(o_1)$$

Phương trình trên có nghĩa là biến chuyển tiếp đầu tiên được tính bằng cách nhân xác suất ban đầu của trạng thái thứ j với xác suất phát xạ quan sát b của trạng thái đó với O có thể quan sát tại thời điểm 1. Hình 1.4 mô tả khởi tạo thuật toán Forward cho ví dụ 3.



Hình 1.4: Khởi tạo thuật toán Forward

Có thể thấy, xác suất để bắt đầu trạng thái Sunny là 0.6, xác suất từ Sunny đến paint là 0.4. Từ đó:

$$\alpha_1(sunny) = \alpha_1(1) = \pi_1 b_1(o_1) = 0.6 \times 0.4 = 0.24$$

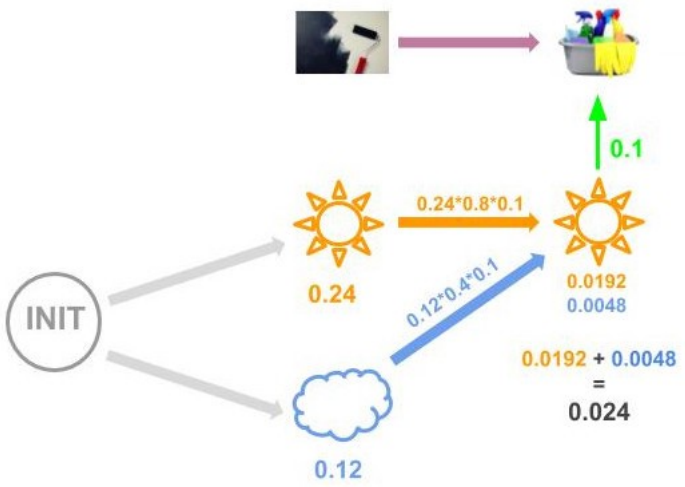
Tương tự như vậy:

$$\alpha_1(rainy) = \alpha_1(2) = \pi_2 b_2(o_1) = 0.4 \times 0.3 = 0.12$$

1.3.2 Đệ quy (Recursion)

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i).a_{ij}.b_j(o_{t+1})$$

với $t = 2, \ldots, T$. Với ví dụ 3, ta tính toán $\alpha_2(sunny)$ cho quan sát O_{clean} được mô phỏng như ở hình 1.5.

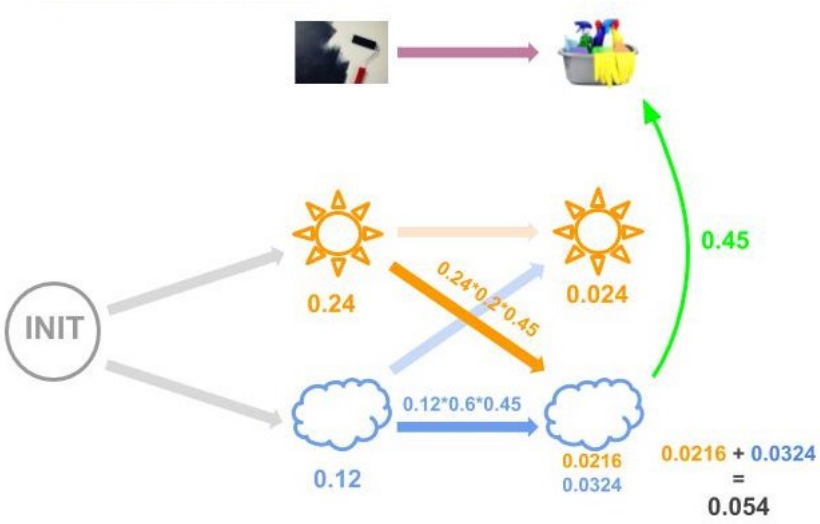


Hình 1.5: Recursion của thuật toán Forward 1

Biến forward trước đó là $\alpha_1(sunny) = 0.24$ và $\alpha_1(rainy) = 0.12$. Xác suất chuyển trạng thái từ sunny và rainy sang sunny lần lượt là 0.8 và 0.4. Xác suất phát xạ quan sát từ sunny sang clean là 0.1. Từ đó ta có:

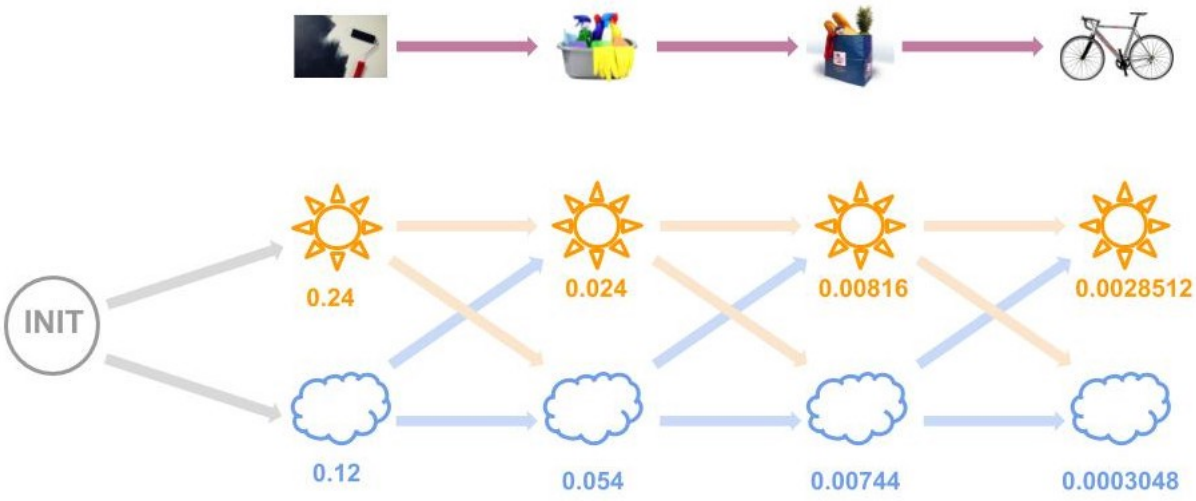
$$\alpha_2(sunny) = \alpha_2(1) = 0.24 \times 0.8 \times 0.1 + 0.12 \times 0.4 \times 0.1 = 0.0192 + 0.0048 = 0.024$$

Tương tự như vậy, chúng ta sẽ tính được biến forward của rainy là $\alpha_2(rainy) = \alpha_2(2) = 0.054$ (Hình 1.7).



Hình 1.6: Recursion của thuật toán Forward 2

Lặp lại như vậy, ta tính toán được kết quả như hình 1.7



Hình 1.7: Recursion của thuật toán Forward 3

1.3.3 Kết thúc (Termination)

$$P(O|\lambda) = \alpha_T(q_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF}$$

Áp dụng với ví dụ 3, do vì chúng ta không có trạng thái kết thúc nên coi $a_{iF} = 1$. Từ đó:

$$P(O|\lambda) = 0.0028512 + 0.0003048 = 0.003156$$

Qua các bước và ví dụ bên trên, chúng ta có thể tổng kết ngắn gọn thuật toán Forward như sau:

1. Initialization:
$$\alpha_1(j) = \pi_j b_j(o_1); 1 \leq j \leq N$$
2. Recursion:
$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); 1 \leq j \leq N, 1 < t \leq T$$
3. Termination:
$$P(O|\lambda) = \alpha_T(q_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF}$$

Ngoài thuật toán Forward ra thì thuật toán Backward cũng có thể được sử dụng cho bài toán Likelihood. Trong thực tế, để xử lý bài toán Likelihood của mô hình Markov ẩn, chúng ta không cần thiết phải sử dụng thuật toán Backward, tuy nhiên, thuật toán này kết hợp cùng forward sẽ giúp chúng ta xử lý bài toán learning. Chúng ta sẽ tìm hiểu về thuật toán Backward trong phần dưới đây để có thể so sánh và đánh giá 2 thuật toán với nhau.

1.4 Thuật toán Backward

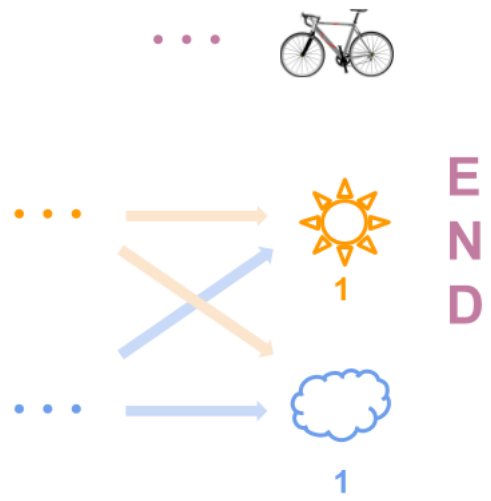
Thuật toán Backward có các bước thực hiện tương tự như thuật toán forward: khởi tạo, đệ quy và kết thúc, nhưng xét theo chiều lùi ngược lại. Gọi $\beta_t(i)$ là biến backward của trạng thái thứ i tại thời điểm t.

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T, q_t = i|\lambda)$$

1.4.1 Khởi tạo (Initialization)

$$\beta_T(i) = a_{iF}$$

Tại thời điểm T (ở cuối chuỗi quan sát), các biến backward của mọi trạng thái bằng a_{iF}, trong trường hợp không có trạng thái kết thúc, chúng ta mặc định là 1. Bước khởi tạo của ví dụ 3 được mô phỏng như ở hình 1.8.

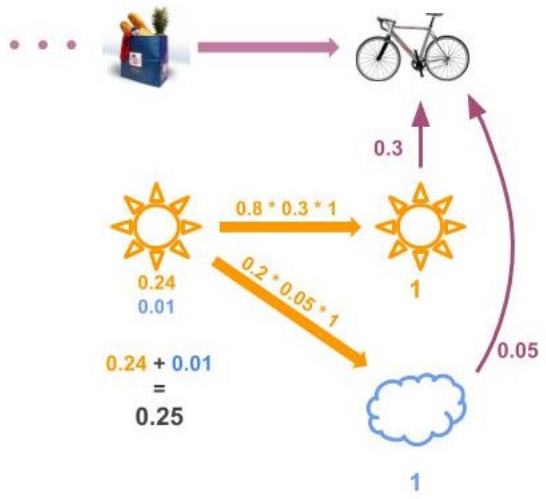


Hình 1.8: Khởi tạo thuật toán backward

1.4.2 Đệ quy (Recursion)

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

Phương trình trên là tổng tất cả các xác suất chuyển từ trạng thái i sang trạng thái j, nhân với xác suất phát xạ của O có thể quan sát được tại thời điểm $t+1$ từ trạng thái j, nhân với biến backward của trạng thái j tại thời điểm $t+1$. Với ví dụ 3, chúng ta có thể tính biến backward của trạng thái ẩn sunny như hình 1.9.



Hình 1.9: Recursion của thuật toán Backward 1

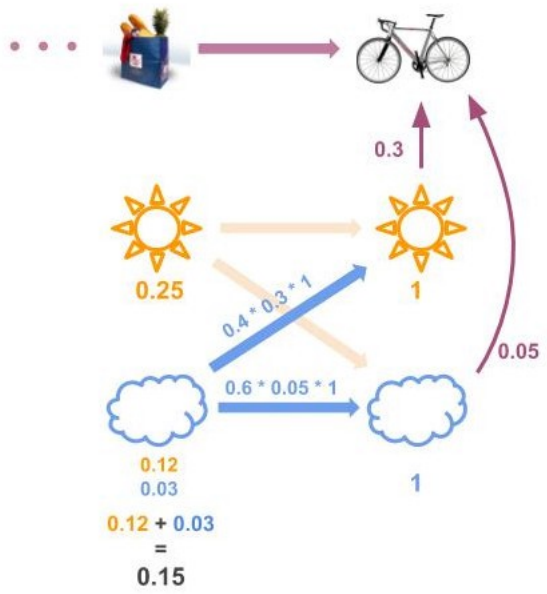
Tại thời điểm $T = 4$ cuối cùng của chuỗi quan sát là bike, các biến backward đã được khởi tạo là 1. Ta sẽ tính toán biến backward của trạng thái sunny trước đó $\beta_3(sunny)$. Các bước tính toán như sau:

- Xác suất chuyển trạng thái từ sunny sang sunny là 0.8, xác suất chuyển trạng thái sunny sang quan sát bike là 0.3, biến backward $\beta_4(sunny) = 1$. Do đó, ta có: $0.8 \times 0.3 \times 1 = 0.24$.
- Xác suất chuyển trạng thái từ sunny sang rainy là 0.2, xác suất chuyển trạng thái rainny sang quan sát bike là 0.05, biến backward $\beta_4(rainy) = 1$. Do đó, ta có: $0.2 \times 0.05 \times 1 = 0.01$.

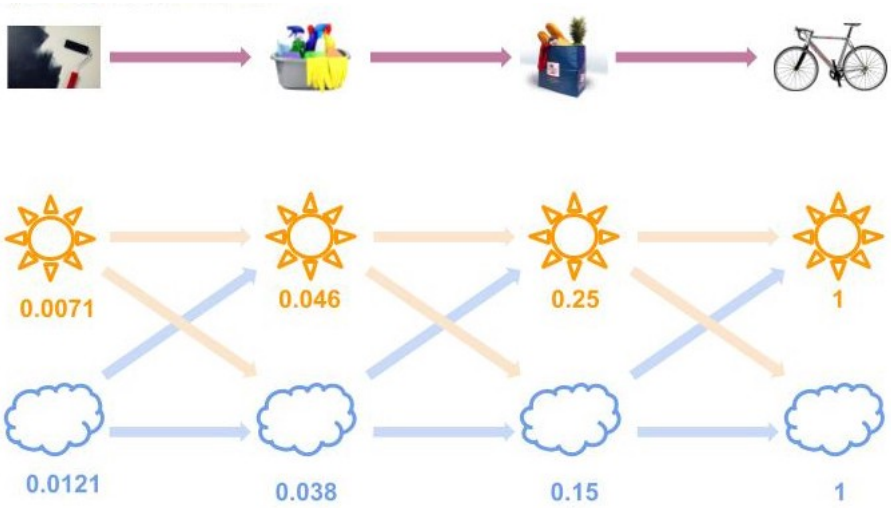
Sau đó tổng hợp các kết quả, ta có:

$$\beta_3(sunny) = \beta_3(1) = 0.24 + 0.01 = 0.25$$

Tương tự như vậy, chúng ta tính được $\beta_3(rainy)$, kết quả thu được như ở hình 1.10. Lặp lại các bước tính đệ quy, chúng ta thu được kết quả của các biến backward như hình 1.11.



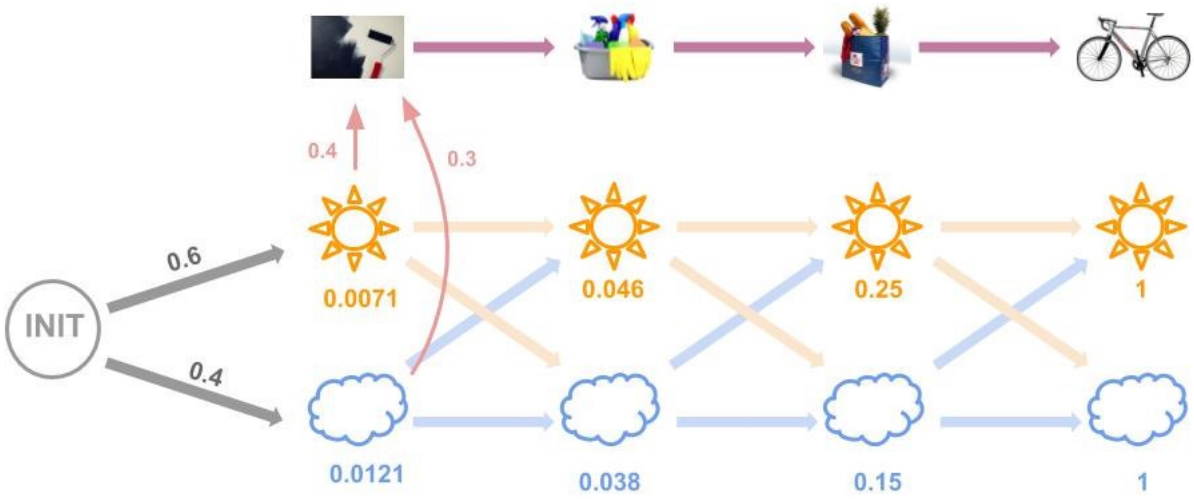
Hình 1.10: Recursion của thuật toán Backward 2



Hình 1.11: Recursion của thuật toán Backward 3

1.4.3 Kết thúc (Termination)

$$P(O|\lambda) = \alpha_T(q_F) = \beta_1(q_1) = \sum_{j=1}^{N} \pi_j b_j(o_1)\beta_1(j)$$



Hình 1.12: Terminal của thuật toán Backward

Áp dụng với ví dụ 3, ta có:

- Xác suất ban đầu của trạng thái sunny là 0.6, biến backward của sunny tại thời điểm $t = 1$ là $\beta_1(sunny) = 0.0071$, xác suất phát xạ từ sunny sang quan sát paint tại thời điểm $t = 1$ là 0.4. Do đó, ta có: $0.6 \times 0.0071 \times 0.4 = 0.001704$.
- Xác suất ban đầu của trạng thái rainy là 0.4, biến backward của rainy tại thời điểm $t = 1$ là $\beta_1(rainy) = 0.0121$, xác suất phát xạ từ rainy sang quan sát paint tại thời điểm $t = 1$ là 0.3. Do đó, ta có: $0.4 \times 0.0121 \times 0.3 = 0.001452$.

Tổng của 2 thành phần trên, ta thu được:

$$P(O|\lambda) = 0.001704 + 0.001452 = 0.003156$$

Vậy qua 2 thuật toán Forward và Backward, chúng ta có thể kết luận rằng xác suất để xảy ra chuỗi quan sát $O = \{O_{\text{paint}}, O_{\text{clean}}, O_{\text{shop}}, O_{\text{bike}}\}$ là 0.003156 với mô hình Markov ẩn mà ta đã xây dựng.

Qua các bước bên trên, chúng ta có thể tổng kết ngắn gọn thuật toán backward như sau:

1. Initialization:
$$\beta_T(i) = a_{iF}; 1 \leq i \leq N$$

2. Recursion:
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j); 1 \leq i \leq N, 1 \leq t \leq T$$

3. Termination:
$$P(O|\lambda) = \alpha_T(q_F) = \beta_1(q_1) = \sum_{j=1}^{N} \pi_j b_j(o_1) \beta_1(j)$$

1.5 Decoding: Thuật toán Viterbi

Bài toán giải mã (Decoding) là bài toán cho một chuỗi các quan sát O và một HMM $\lambda = (A, B)$. Xác định một dãy các trạng thái tốt nhất Q. Giả sử với ví dụ 3, chúng ta có chuỗi quan sát như sau:

$$O = \{O_{\text{shop}}, O_{\text{clean}}, O_{\text{bike}}, O_{\text{paint}}\}$$

Chúng ta cần tìm thời tiết tương ứng với 4 quan sát này. Để thực hiện bài toán này, tư tưởng đơn giản là có thể sử dụng thuật toán forward hoặc backward tính toán likelihood của các chuỗi quan sát dựa trên các chuỗi trạng thái ẩn, các chuỗi trạng thái ẩn này được liệt kê có dạng như $\{SSSS, SSSR, \ldots, \}$ (trong đó S tương ứng với *Sunny*, R tương ứng với *Rainy*). Sau đó, chúng ta chọn chuỗi trạng thái ẩn mà cho likelihood của chuỗi quan sát đang xét lớn nhất. Ý tưởng này đơn giản và rõ ràng, nhưng độ phức tạp tính toán rất lớn.

Thay vào đó, một thuật toán giải mã (decoding) phổ biến nhất cho mô hình markov ẩn gọi là **thuật toán Viterbi**. Viterbi là một thuật toán quy hoạch động. Thuật toán cũng bao gồm các bước như thuật toán forward: khởi tạo (Initialization), đệ quy (Recursion), kết thúc (Termination) và thêm các bước quay lui (backtracking) để tìm chuỗi các trạng thái ẩn.

Chúng ta sẽ tìm hiểu thuật toán Viterbi và đồng thời thực hiện tìm chuỗi trạng thái ẩn hợp lý nhất cho chuỗi quan sát $O = \{O_{\text{shop}}, O_{\text{clean}}, O_{\text{bike}}, O_{\text{paint}}\}$ trong phần này. Gọi $v_t(j)$ là **biến Viterbi** là xác suất của HMM ở trạng thái j sau khi nhìn thấy t quan sát đầu tiên và đi qua chuỗi trạng thái có thể xảy ra nhất $q_0, q_1, \ldots, q_{t-1}$:

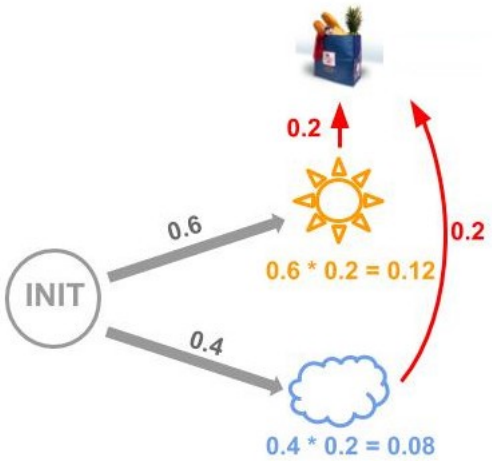
$$v_t(j) = \max_{q_0,\ldots,q_{t-1}} P(q_0, q_1, \ldots, q_{t-1}, o_1, o_2, \ldots, o_t, q_t = j|\lambda)$$

Đại lượng này biểu diễn đường đi có thể xảy ra nhất bằng cách lấy max các chuỗi trạng thái có thể xảy ra trước đó.

1.5.1 Khởi tạo (Initialization)

$$v_1(j) = \pi_j b_j(o_1)$$

Phương trình này giống như khởi tạo của thuật toán forwar. Với ví dụ đang xét, khởi tạo của thuật toán viterbi chúng ta tính toán được như hình 1.13.



Hình 1.13: Khởi tạo của thuật toán Viterbi

Xác suất ban đầu để trạng thái là sunny là 0.6, xác suất phát xạ từ trạng thái sunny đến quan sát shop là 0.2. Do đó, ta có:

$$v_1(sunny) = v_1(1) = \pi_1 b_1(o_1) = 0.6 \times 0.2 = 0.12$$

Tương tự, ta cũng có:

$$v_1(rainy) = v_1(2) = \pi_2 b_2(o_1) = 0.4 \times 0.2 = 0.08$$

Thuật toán Forward chỉ cần truy xuất xác suất cho một chuỗi quan sát, trong khi thuật toán Viterbi vừa phải truy xuất ra xác suất, đồng thời cũng truy xuất ra chuỗi trạng thái có khả năng nhất. Vì vậy, chúng ta cần mảng **backpointer** để theo dõi các đối số mà tối đa hoá cho cho mỗi t và j. Gọi là $bt_t(j)$, trong bước khởi tạo đầu tiên, chúng ta khởi tạo:

$$bt_1(j) = 0$$

vì không có đối số cụ thể từ xác suất ban đầu tối đa hoá giá trị của trạng thái đầu tiên. Từ đó ta có:

$$bt_1(sunny) = bt_1(1) = [0]$$

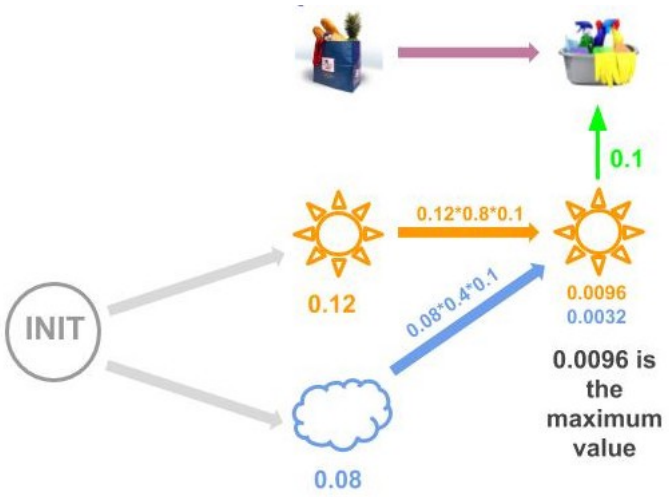
$$bt_1(rainy) = bt_1(2) = [0]$$

1.5.2 Đệ quy (Recursion)

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

$$bt_t(j) = \text{argmax}_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

Áp dụng với ví dụ 3 và chuỗi quan sát chúng ta đang xét, chúng ta tính $v_2(sunny)$ như hình 1.14.



Hình 1.14: Recurrsion của thuật toán Viterbi 1

Để tính $v_2(sunny)$, chúng ta thực hiện lấy max của 2 giá trị 0.0096 và 0.0032. Từ đó:

$$v_2(sunny) = v_2(1) = 0.0096$$

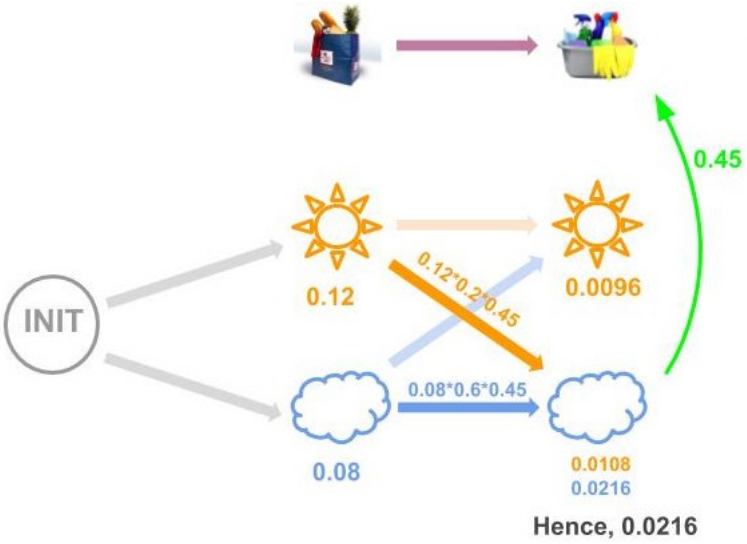
Đồng thời, trạng thái làm cực đại $v_2(sunny)$ là sunny, vì thế ta có:

$$bt_2(sunny) = bt_2(1) = [0, sunny]$$

Tương tự với rainy, ta thu được kết quả như hình1.15. Hay:

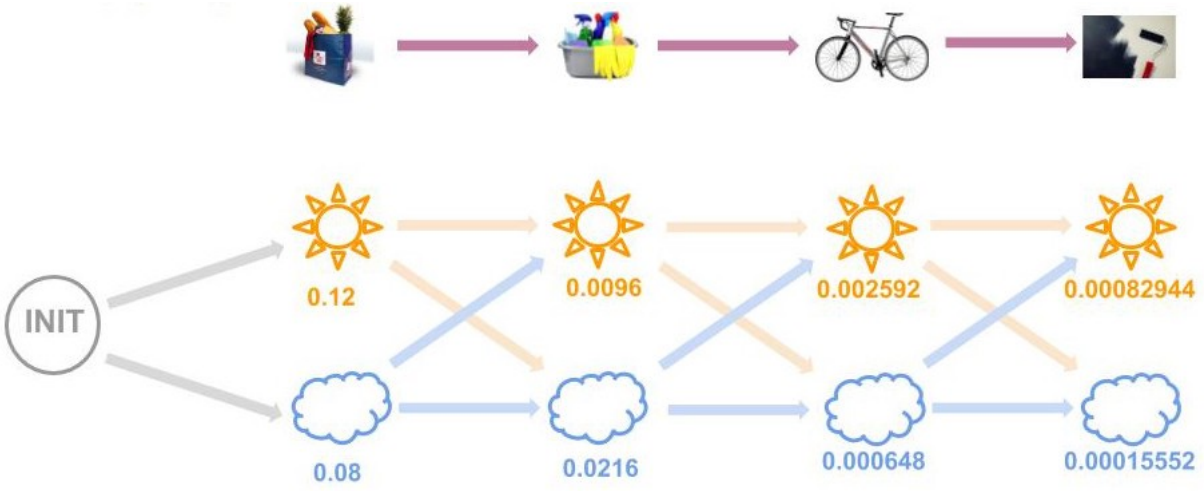
$$v_2(rainy) = v_2(2) = 0.0216$$

$$bt_2(rainy) = bt_2(2) = [0, rainy]$$



Hình 1.15: Recurrsion của thuật toán Viterbi 2

Lặp lại bước đệ quy, kết quả thu được như hình 1.16.



Hình 1.16: Recurrsion của thuật toán Viterbi 3

Mảng $bt_t(j)$ cuối cùng là:

$$bt_4(sunny) = [0, sunny, rainy, sunny]$$

$$bt_4(rainy) = [0, rainy, rainy, sunny]$$

1.5.3 Kết thúc (Termination)

$$v_{T+1}(j) = \max_{i=1}^{N} v_T(i) a_{iF}$$
$$bt_{T+1}(j) = \text{argmax}_{i=1}^{N} v_T(i) a_{iF}$$

Nếu không có trạng thái kết thúc, ta mặc định $a_{iF} = 1$. Tại thời điểm cuối cùng, biến viterbi của sunny là 0.00082944 và biến viterbi của rainy là 0.00015552. Do $0.00082944 > 0.00015552$ nên $P^* = 0.00082944$ và phần tử cuối cùng của mảng bt là sunny bởi vì sunny là tham số làm cực đại hoá phương trình v_T. Mảng bt của sunny và rainy tương ứng là:

$$bt_5(sunny) = [0, sunny, rainy, sunny, sunny]$$

$$bt_5(rainy) = [0, rainy, rainy, sunny, sunny]$$

1.5.4 Quay lui (Backtracking)

$$q_t^* = bt_{t+1}(q_{t+1}^*)$$

Chúng ta bắt đầu quay lui từ trạng thái cuối cùng của chuỗi trạng thái ẩn qua mảng bt, khi $t + 1 = T$, trạng thái q^* cuối cùng là sunny. Dãy trạng thái ẩn tối ưu là:

$$Q = \{\ldots, \ldots, \ldots, sunny\}$$

Để tìm trạng thái ẩn tối ưu tại thời gian t, ta tìm kiếm trong mảng bt của trạng thái q^* tại thời điểm $t + 1$. q^* tại thời điểm 4 là sunny, để tìm trạng thái ẩn tối ưu tại thời điểm $t = 3$, chúng ta tìm kiếm mảng bt của mảng sunny tại thời điểm $t+1 = 3+1 = 4$ là $bt_5(sunny) = [0, sunny, rainy, \textbf{sunny}, sunny]$. Từ đó:

$$Q = \{\ldots, \ldots, sunny, sunny\}$$

Lặp lại quá trình trên, ta thu được chuỗi trạng thái ẩn tối ưu là:

$$Q = \{rainy, rainy, sunny, sunny\}$$

1.6 Thuật toán Forward-Backward

Thuật toán Forward-Backward (còn được gọi là **thuật toán Baum-Welch** là trường hợp đặc biệt của **thuật toán EM**, sử dụng để xử lý bài toán ước lượng tham số cho HMM (bài toán Learning). Thuật toán này sẽ được chúng ta huấn luyện để tìm ma trận xác suất chuyển (transition probabities) A và ma trận esmission B của HMM. EM là một thuật toán lặp. Nó hoạt động bằng cách tính toán ước lượng ban đầu cho các xác suất, sau đó sử dụng các ước lượng để tính toán một ước lượng tốt hơn, và như vậy, lặp đi lặp lại việc nâng cao các xác suất mà nó học.

Chúng ta ước lượng tối đa hoá khả năng của xác suất a_{ij} (xác suất chuyển trạng thái từ trạng thái i sang trạng thái j) bằng cách đếm số lần chuyển trạng thái, gọi là $C(i \to j)$, sau đó chia cho tổng số lần chuyển trạng thái từ i:

$$a_{ij} = \frac{C(i \to j)}{\sum_{q \in Q}(C \to q)}$$

Chúng ta có thể tính toán xác suất trên với chuỗi Markov vì các trạng thái đều quan sát được. Với HMM, chúng ta không thể tính toán theo công thức trên vì không quan sát được hướng chuyển của các trạng thái. Thuật toán Baum-Welch dựa trên tư tưởng của thuật toán EM để giải quyết vấn đề này. Theo thuật toán EM, chúng ta ước lượng $\hat{a}_{ij}$:

$$\hat{a}_{ij} = \frac{\text{Kỳ vọng số lần chuyển từ trạng thái i sang j}}{\text{Kỳ vọng số lần chuyển từ trạng thái i}}$$

Định nghĩa $\xi_t(i, j)$ là xác suất hệ ở trạng thái i thời điểm t và ở trạng thái j thời điểm $t + 1$:

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j | O, \lambda)$$

Ta có:

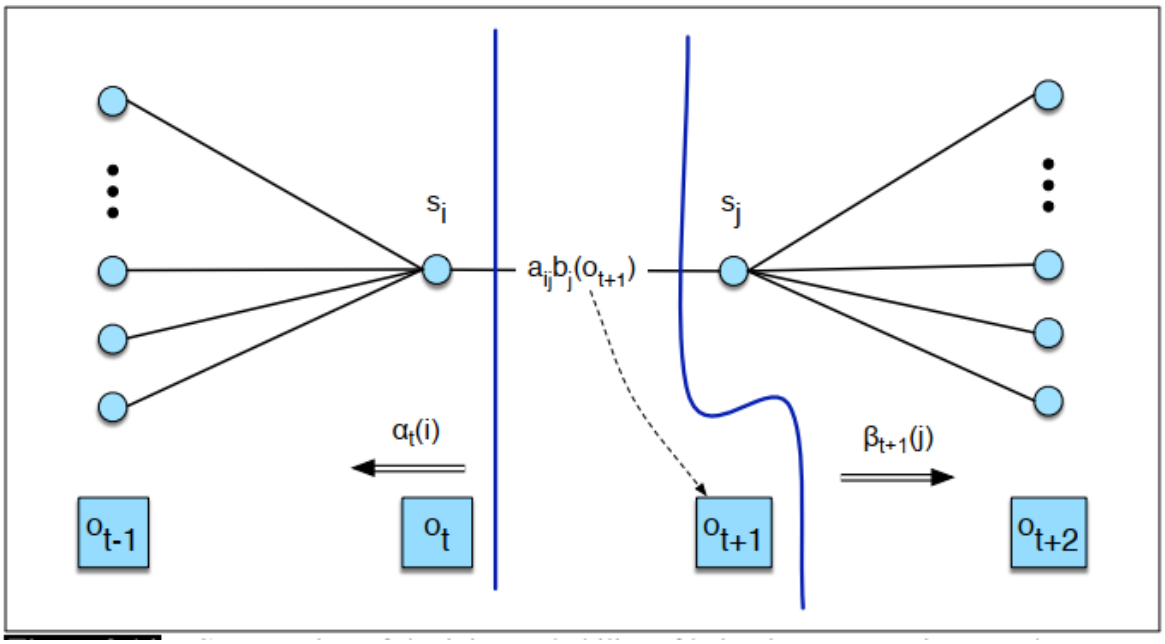
$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i, k)}$$

Để tính được $\xi_t(i, j)$, chúng ta áp dụng tính chất xác suất:

$$P(X|Y, Z) = \frac{P(XYZ)}{P(YZ)} = \frac{P(Z)P(X, Y|Z)}{P(Z)P(Y|Z)}$$

Từ đó, chúng ta định nghĩa *not-quite-*ξ_t như sau:

$$\textit{not-quite-}\xi_t(i, j) = P(q_t = i, q_{t+1} = j, O|\lambda)$$



Hình 1.17: Mô hình chuyển từ trạng thái i thời điểm t sang trạng thái j thời điểm $t+1$

Hình 1.17 cho thấy mô hình chuyển từ trạng thái s_i tạithời điểm t sang trạng thái s_j tại thời điểm $t+1$. Từ đó:

$$not\text{-}quite\text{-}\xi_t(i,j) = \alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)$$

Lại có:

$$P(O|\lambda) = \alpha_T(q_F) = \beta_T(q_0) = \sum_{j=1}^{N} \alpha_t(j)\beta_t(j)$$

Cuối cùng, phương trình của ξ_t là:

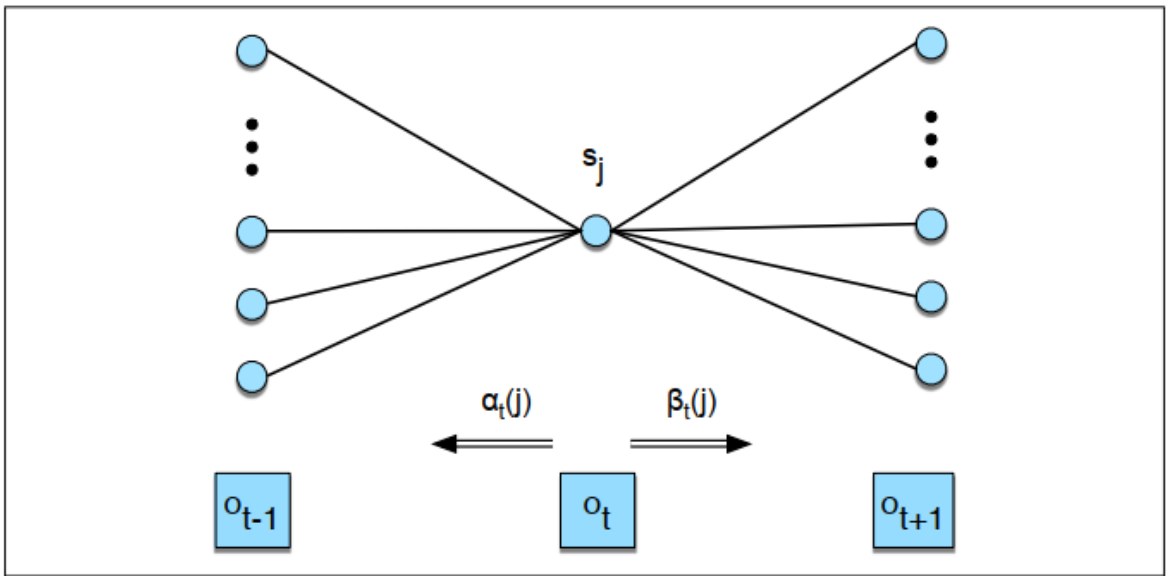
$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)}$$

Tiếp theo, chúng ta cũng cần một công thức để ước lượng các tham số của ma trận emission, tương tự như ước lượng $\hat{a}_{ij}$, ta định nghĩa:

$$\hat{b}_j(v_k) = \frac{\text{Kỳ vọng của số lần ở trạng thái } j \text{ và quan sát } v_k}{\text{Kỳ vọng của số lần ở trạng thái } j}$$

Chúng ta cần biết xác suất ở trạng thái j thời điểm t, gọi là $\gamma_t(j)$:

$$\gamma_t(j) = P(q_t = j|\lambda, O) = \frac{P(q_t = j, O|\lambda)}{P(O|\lambda)}$$



Hình 1.18: Mô hình ở trạng thái s_j tại thời điểm t

Hình 1.18 cho thấy mô hình khi ở trạng thái s_j tại thời điểm t, từ đây ta có:

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)}$$

Vậy:

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 \text{ s.t } O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Qua các phân tích, ta có thuật toán Forward-Backward như sau:

1. Khởi tạo A, B
2. Lặp cho đến khi hội tụ:
   - **E-step**

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)}$$
$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)}$$

   - **M-step**

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1}\sum_{k=1}^{N} \xi_t(i,k)}$$
$$\hat{b}_j(v_k) = \frac{\sum_{t=1 \text{ s.t } O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

3. Trả về A,B

Chương 2

Part-of-Speech Tagging

2.1 Bài toán gán nhãn từ loại

Gán nhãn từ loại (Part of Speech Tagging - POS tagging) là bài toán mà mỗi từ trong câu được gán một thẻ tương ứng. Đầu vào là một tập văn bản đã tách từ và tập nhãn, đầu ra là các nhãn tương ứng với mỗi từ một cách chính xác nhất. Chúng ta có thể chia tất cả các từ thành một số loại như danh từ, tính từ, động từ, ... Bài toán gán nhãn từ loại có thể giúp ta phân tích văn bản, có nhiều ứng dụng trong text-to-speech, tiền xử lý cho phân tích cú pháp, nhận dạng giọng nói, ... Trong bài toán gán nhãn từ loại, một từ vừa có thể là danh từ, vừa có thể là động từ, các từ là nhập nhằng (ambious) - có thể có nhiều nhãn, mục tiêu của chúng ta là tìm tag đúng cho mỗi vị trí. Ví dụ với từ *book*, vừa có thể là động từ trong câu:

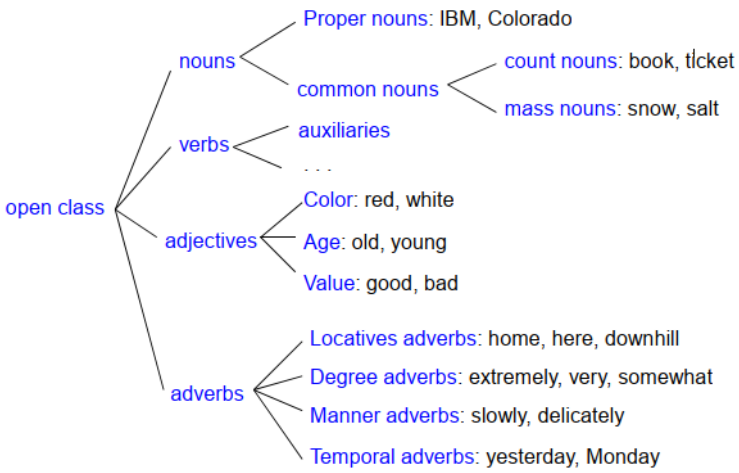
book that flight

Vừa có thể là danh từ trong câu:

hand me that book

vì vậy, tuỳ theo ngữ cảnh mà một từ có thể mang những thẻ từ loại khác nhau. Trong tiếng Anh, tập từ loại có thể chia thành 2 lớp chính là **lớp đóng** và **lớp mở**:

- **Lớp đóng:** (các từ chức năng) chứa số lượng cố định.
  - Giới từ (prepositions): on, under, over, ...
  - Tiểu từ (particles): abroad, about, before, in, ...
  - Mạo từ (articles): a, an, the
  - Liên từ (conjunctions): and, or, but, that, ...
  - Đại từ (pronouns): you, me, I, your, what, who, ...
  - Trợ động từ (auxiliary verbs): can, may, should, ...
- **Lớp mở:** có thể thêm từ mới (Hình 2.2)



Hình 2.1: Lớp từ mở trong tiếng Anh

Có thể thấy rằng, từ loại trong tiếng Anh được chia thành nhiều nhãn. Có một số tập nhãn phổ biến như: tập nhãn Brown (87 nhãn), Penn Treebank (45 nhãn), British national corpus (61 nhãn), C7 (146 nhãn).

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	$
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Hình 2.2: Tập nhãn Penn Treebank

Một số ví dụ về gán nhãn từ loại trong tiếng Anh:

1. The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.
2. **There**/EX are/VBP 70/CD children/NNS **there**/RB
3. Preliminary/JJ findings/NNS were/VBD reported/VBN in/IN today/NN 's/POS New/NNP England/NNP Journal/NNP of/IN Medicine/NNP ./.

Có nhiều phương pháp để gán nhãn từ loại, trong phần này, chúng ta sẽ tìm hiểu hai phương pháp là: gán nhãn từ loại bằng **mô hình Markov ẩn (Hidden Markov Model - HMM)** và gán nhãn từ loại bằng **cực đại Entropy mô hình Markov (Maximum Entropy Markov Model - MEMM)**.

Để đánh giá độ chính xác của một mô hình, chúng ta có thể dùng độ đo **accuracy** - Tỷ lệ thẻ gán nhãn chính xác trên tập kiểm thử. Ngoài ra, hiện nay có một số độ đo phổ biến khác như **precision, recall, F1 Score**,...

2.2 Hidden Markov Model POS Tagging

Trong phần này, chúng ta tìm hiểu về mô hình Markov ẩn cho bài toán POS tagging. Để mô hình hoá bài toán POS tagging bằng mô hình Markov ẩn, thì tập các trạng thái ẩn là các nhãn chúng ta cần gán cho các từ, tập các quan sát chính là các từ trong chuỗi đã cho. Tuy nhiên, khi sử dụng HMM cho POS tagging, chúng ta thường không sử dụng thuật toán Baum-Welch để học các tham số của HMM mà sử dụng thuật toán **Viterbi** để decoding, và chúng ta sẽ cần xem làm thế nào để thiết lập các tham số HMM từ dữ liệu huấn luyện.

2.2.1 Phương trình cơ bản của HMM tagging

Mục tiêu của chúng ta là chọn chuỗi tag có xác suất cao nhất cho chuỗi quan sát n từ w_1^n:

$$\hat{t}_1 = \underset{t_1^n}{\text{argmax}} P(t_1^n|w_1^n) = \underset{t_1^n}{\text{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)} = \underset{t_1^n}{\text{argmax}} P(w_1^n|t_1^n)P(t_1^n)$$

Dấu bằng cuối cùng xảy ra do mẫu số không phụ thuộc vào nhãn. Để đơn giản hoá và giải phương trình trên, chúng ta thêm hai giả thiết:

1. Các xác suất của một từ xuất hiện chỉ phụ thuộc vào thẻ riêng của nó, không phụ thuộc vào từ và thẻ láng giềng:

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

2. [Giả định bigram] Xác suất một tag chỉ phụ thuộc vào một thẻ trước đó, không phụ thuộc vào toàn bộ chuỗi thẻ:

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

Từ các giả thiết này, ta có:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P\left(t_1^n \mid w_1^n\right) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P\left(w_i \mid t_i\right)}^{\text{emission}} \overbrace{P\left(t_i \mid t_{i-1}\right)}^{\text{transition}}$$

2.2.2 Ước lượng các xác suất

Trong HMM Tagging, thay vì ước lượng các tham số của HMM, các xác suất được ước lượng bằng cách đếm trên corpus train đã gán thẻ.

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

Ví dụ trong corpus WSJ, modal (MD) xuất hiện 13124 lần trong đó đi theo verb base form (VB) là 10471, ước lượng hợp lý cực đại của:

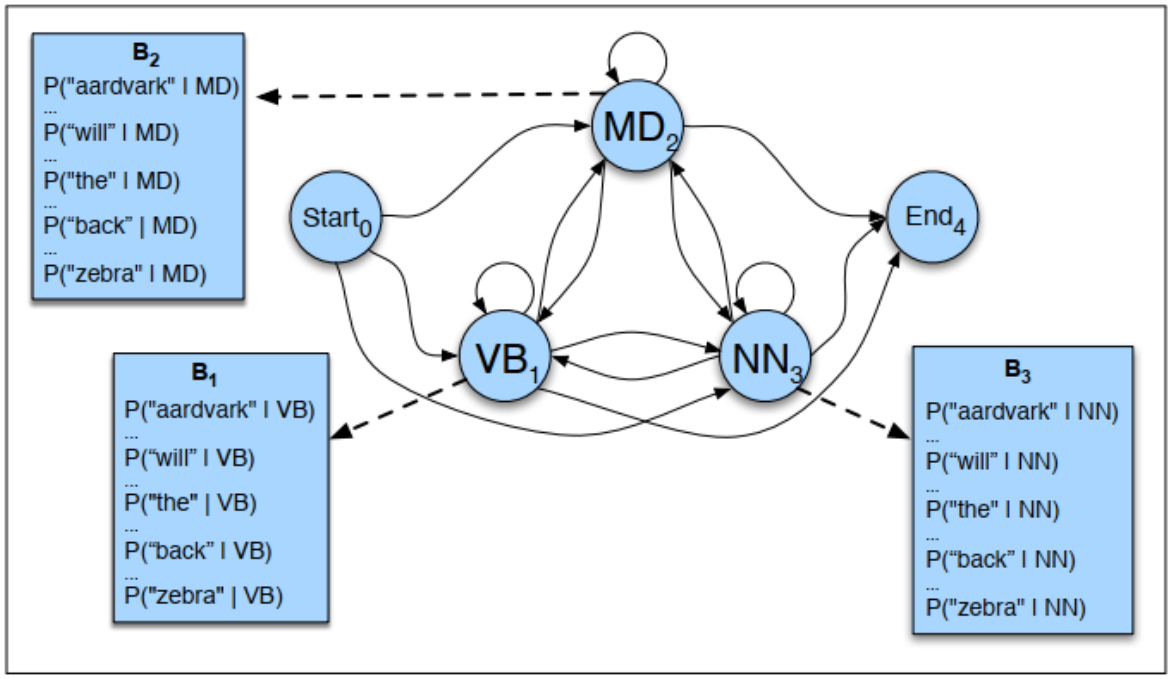
$$P(VB|MD) = \frac{C(MD, VB)}{C(MD)} = \frac{10471}{13124} = 0.80$$

Tương tự vậy với xác suất emission:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

Ví dụ trong corpus WSJ, thẻ MD xuất hiện 13124 lần, trong đó từ *will* được gán thẻ này là 4046 lần, khi đó:

$$P(will|MD) = \frac{C(MD, will)}{C(MD)} = \frac{4046}{13124} = 0.31$$



Hình 2.3: Mô phỏng HMM cho POS Tagging với 3 tag và một số từ quan sát

2.2.3 Ví dụ và thuật toán

Trong phần này chúng ta sẽ đi vào một ví dụ và đưa ra thuật toán Viterbi cho bài toán POS tagging. Giả sử chúng ta có một chuỗi các từ:

Jane will back the will

Chuỗi tag đúng sẽ là:

Jane/NNP will/MD back/VB the/DT bill/NN

Chúng ta cần thiết lập các tham số cho HMM từ corpus WSJ. Hình 2.4 cho thấy một phần của ma trận xác suất chuyển (transition matrix) được tính toán từ corpus này, trong đó hàng đầu tiên (<s>) đại diện cho xác suất ban đầu π. Hình 2.5 biểu diễn ma trận cho chuỗi quan sát (emission matrix) của chúng ta. Dựa vào thuật toán viterbi, chúng ta thiết lập bảng V trong đó $V_t(j)$ đại diện cho mỗi ô của bảng như hình 2.6 (t đại diện cho cột - tương ứng là các quan sát (từ của câu), j đại diện cho hàng - tương ứng là các tag). Theo thuật toán viterbi, ta có:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t)$$

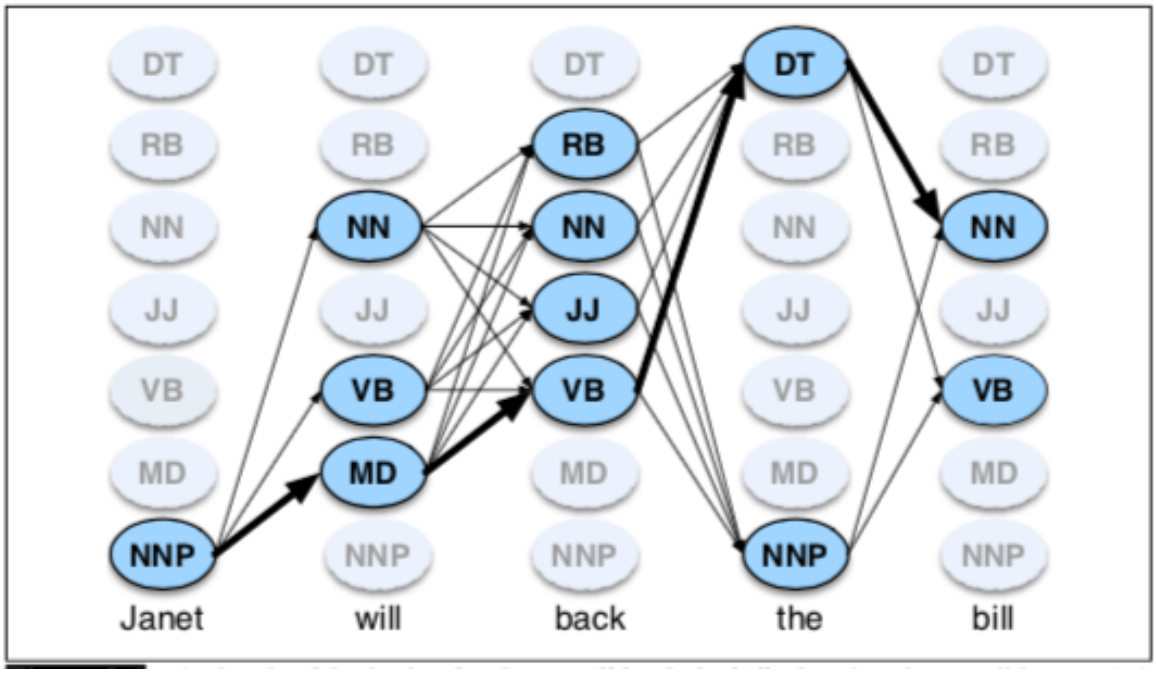
Luồng tính toán với ví dụ của chúng ta được mô phỏng ở hình 2.7, ở đây chúng ta chỉ xét mô hình gồm 7 tag (7 trạng thái ẩn). Tổng kết lại, chúng ta có thuật toán Viterbi như hình 2.8.

	NNP	MD	VB	JJ	NN	RB	DT
<*s*>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

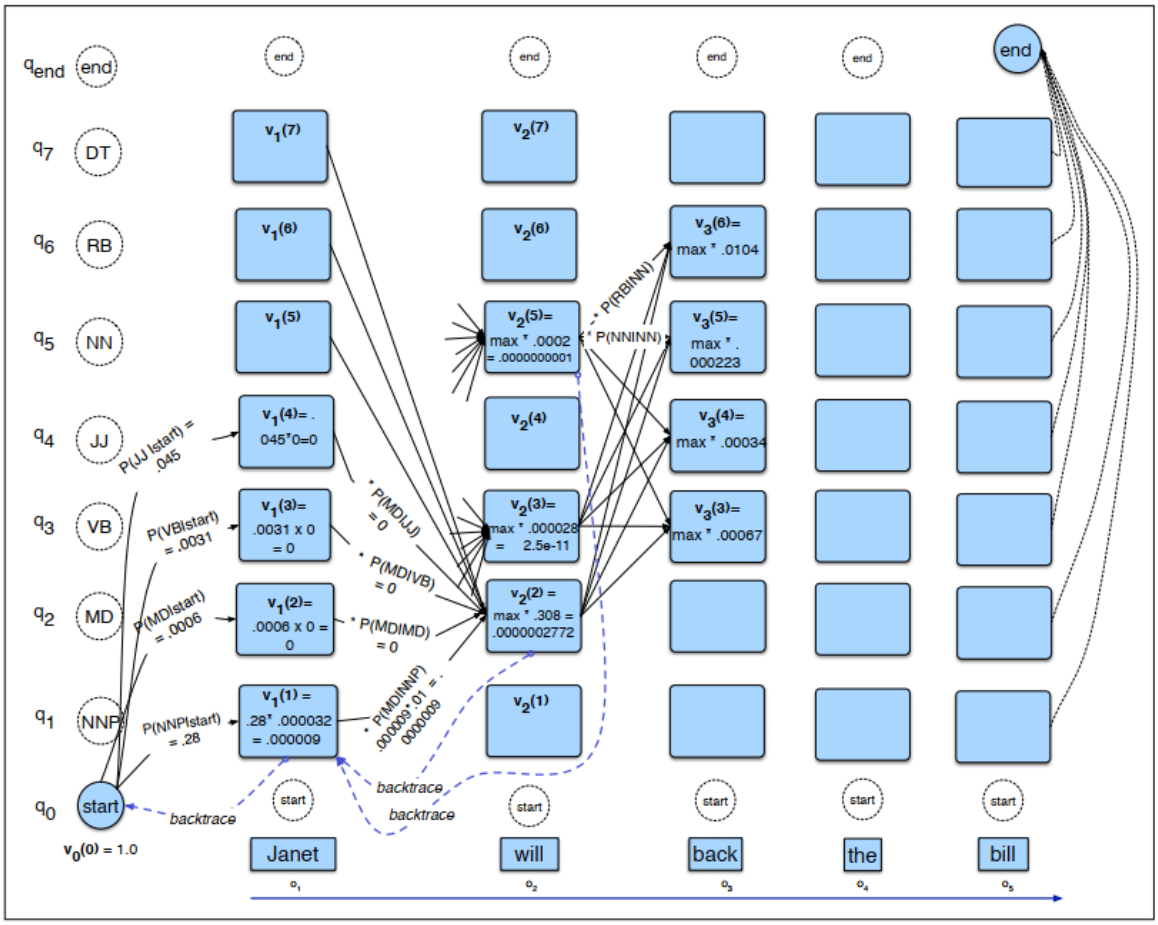
Hình 2.4: Ma trận xác suất chuyển được tính toán từ corpus WSJ

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Hình 2.5: Ma trận cho chuỗi quan sát được tính toán từ corpus WSJ



Hình 2.6: Bảng tính toán theo thuật toán Viterbi



Hình 2.7: Luồng tính toán của thuật toán Viterbi

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*

create a path probability matrix *viterbi*$[N+2,T]$
for each state s **from** 1 **to** N **do** ; initialization step
  $viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
  $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$
$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

Hình 2.8: Thuật toán Viterbi

2.2.4 Mở rộng HMM cho Trigrams

Mở rộng cho bối cảnh (context) rộng hơn, trong phần trên, xác suất của 1 thẻ chi phụ thuộc vào thẻ trước đó (bigram):

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

Trên thực tế, chúng ta có thể sử dụng nhiều hơn các thẻ quá khứ, cho phép xác suất 1 thẻ phụ thuộc vào 2 thẻ trước đó và đây gọi là trigram:

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1}, t_{i-2})$$

Việc mở rộng bigram thành trigram giúp cải thiện hiệu suất, tuy nhiên nó làm thay đổi đáng kể thuật toán Viterbi. Với mỗi ô của mảng Viterbi, thay vì phải lấy max của các ô 1 cột trước đó thì giờ phải lấy max của các ô của 2 cột trước đó. Mỗi bước tính toán phải xét N^2 trạng thái ẩn thay vì N trạng thái ẩn như trước.

Ngoài việc tăng cửa sổ ngữ cảnh, kiến trúc trạng thái của HMM trigram còn thêm một số đặc trưng khác như đánh dấu kết thúc câu bằng cách thêm tag t_{n+1}. Điều này được cho bởi phương trình sau:

$$\hat{t}_1^n = \underset{t_1^n}{\text{argmax}} P(t_1^n|w_1^n) \approx \left[\prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}, t_{i-2})\right] P(t_{n+1}, t_n)$$

Ta cần thêm các thẻ t_{-1}, t_0, t_{n+1} để tránh hiện tượng tính toán tràn ra khỏi biên của câu. Một vấn đề nữa đó là $P(t_i|t_{i-1}, t_{i-2})$ không thể tính trực tiếp như sau:

$$P(t_i|t_{i-1}, t_{i-2}) = \frac{C(t_{i-2}, t_{i-1}, t_i)}{C(t_{i-2}, t_{i-1})}$$

Vì lý do thưa thớt dữ liệu, tức là không phải bộ t_{i-2}, t_{i-1}, t_i lúc nào cũng xuất hiện trong văn bản. Ý tưởng để giải quyết vấn đề trên đó là sử dụng nội suy tuyến tính:

$$P(t_i|t_{i-1}, t_{i-2}) = \lambda_1 \hat{P}(t_i) + \lambda_2 \hat{P}(t_i|t_{i-1}) + \lambda_3 \hat{P}(t_i|t_{i-1}, t_{i-2})$$

trong đó:

$$\begin{cases} \lambda_1 + \lambda_2 + \lambda_3 = 1 \\ \hat{P}(t_i) = \frac{C(t_i)}{N} \\ \hat{P}(t_i|t_{i-1}) = \frac{C(t_i, t_{i-1})}{C(t_{i-1})} \\ \hat{P}(t_i|t_{i-1}, t_{i-2}) = \frac{C(t_i, t_{i-1}, t_{i-2})}{C(t_{i-1}, t_{i-2})} \end{cases}$$

Các λ được gọi là **nội suy (deteted interpolation)**. Hình 2.9 mô tả thuật toán nội suy này.

function DELETED-INTERPOLATION(*corpus*) **returns** $\lambda_1, \lambda_2, \lambda_3$

$\lambda_1 \leftarrow 0$
$\lambda_2 \leftarrow 0$
$\lambda_3 \leftarrow 0$
foreach trigram t_1, t_2, t_3 with $C(t_1, t_2, t_3) > 0$
  depending on the maximum of the following three values
    case $\frac{C(t_1,t_2,t_3)-1}{C(t_1,t_2)-1}$: increment λ_3 by $C(t_1, t_2, t_3)$
    case $\frac{C(t_2,t_3)-1}{C(t_2)-1}$: increment λ_2 by $C(t_1, t_2, t_3)$
    case $\frac{C(t_3)-1}{N-1}$: increment λ_1 by $C(t_1, t_2, t_3)$
  end
end
normalize $\lambda_1, \lambda_2, \lambda_3$
return $\lambda_1, \lambda_2, \lambda_3$

Hình 2.9: Thuật toán nội suy

Chương 3

Formal Grammars of English

3.1 Văn phạm

3.1.1 Định nghĩa văn phạm

Định nghĩa 1 (Văn phạm)**.**

Văn phạm (Grammar) G là một bộ gồm 4 thành phần:

$$G =< \Sigma, \Delta, S, P >$$

Trong đó:

- Σ là tập **các ký hiệu kết thúc (terminal symbols)**
- Δ là tập **các ký hiệu không kết thúc (non-terminal symbols)** hoặc **các biến (varibles)**
- $S \in \Delta$ là **ký hiệu xuất phát (start symbol)**
- P là tập các **quy tắc (rules)** có dạng $\alpha \to \beta$, α được gọi là vế trái, β được gọi là vế phải của quy tắc này, trong α phải chứa ít nhất một ký hiệu không kết thúc.

$$P = \{\alpha \to \beta | \alpha = \alpha^{'} A \alpha^{''}; A \in \Delta; \alpha^{'}, \alpha^{''}, \beta \in (\Sigma \cup \Delta)^{*}\}$$

Ví dụ 5 (Ví dụ về văn phạm)**.**

Với $\Sigma = \{0, 1\}, \Delta = \{S, A, B\}$, S là trạng thái bắt đầu và có P như sau:

$$S \to 0S1A$$
$$0AB \to 1A1B$$
$$A \to \varepsilon$$

là một văn phạm

Ví dụ 6 (Ví dụ không phải là văn phạm)**.**

Như ví dụ trên, nếu tập các quy tắc P có dạng:

$$0 \to A$$
$$01 \to B$$

thì đây không phải là một văn phạm vì tồn tại quy tắc mà không chứa ký hiệu không kết thúc ở vế trái.
Chú ý: Nếu các quy tắc có vế trái giống nhau có thể viết gọn 2 quy tắc $\alpha \to \beta, \alpha \to \gamma$ thành $\alpha \to \beta|\gamma$

Ví dụ 7.

Cho bộ từ điển (lexicon) như hình 3.1 và ta gọi là $\mathcal{L}_0$, hình 3.2 biểu diễn một văn phạm cho $\mathcal{L}_0$. Đặc biệt là trong tìm hiểu ở phần sau, chúng ta sẽ biết đây còn được gọi là văn phạm phi ngữ cảnh (context-free language).

Noun →	*flights* | *breeze* | *trip* | *morning*
Verb →	*is* | *prefer* | *like* | *need* | *want* | *fly*
Adjective →	*cheapest* | *non-stop* | *first* | *latest* | *other* | *direct*
Pronoun →	*me* | *I* | *you* | *it*
Proper-Noun →	*Alaska* | *Baltimore* | *Los Angeles* | *Chicago* | *United* | *American*
Determiner →	*the* | *a* | *an* | *this* | *these* | *that*
Preposition →	*from* | *to* | *on* | *near*
Conjunction →	*and* | *or* | *but*

Hình 3.1: Lexicon cho $\mathcal{L}_0$

Grammar Rules		Examples
S →	*NP VP*	I + want a morning flight
NP →	*Pronoun*	I
|	*Proper-Noun*	Los Angeles
|	*Det Nominal*	a + flight
Nominal →	*Nominal Noun*	morning + flight
|	*Noun*	flights
VP →	*Verb*	do
|	*Verb NP*	want + a flight
|	*Verb NP PP*	leave + Boston + in the morning
|	*Verb PP*	leaving + on Thursday
PP →	*Preposition NP*	from + Los Angeles

Hình 3.2: Văn phạm cho $\mathcal{L}_0$

3.1.2 Ngôn ngữ sinh bởi văn phạm

Định nghĩa 2 (Suy dẫn trực tiếp)**.**

Cho văn phạm $G = <\Sigma, \Delta, S, P>$ và $\eta, \omega \in (\Sigma \cup \Delta)^*$. Ta nói ω được **suy dẫn trực tiếp (directly derives)** từ η trong G nếu tồn tại quy tắc $\alpha \to \beta \in P$ và $\gamma, \delta \in (\Sigma \cup \Delta)^*$ sao cho $\eta = \gamma\alpha\delta, \omega = \gamma\beta\delta$.

Ký hiệu: $\eta \Rightarrow \omega$

Định nghĩa 3 (Suy dẫn)**.**

Cho văn phạm $G = <\Sigma, \Delta, S, P>$ và $\eta, \omega \in (\Sigma \cup \Delta)^*$. Ta nói ω được **suy dẫn (derives)** từ η trong G nếu tồn tại một dãy $D = \omega_0, \omega_1, .., \omega_k \in (\Sigma \cup \Delta)^*$ sao cho $\omega_0 = \eta, \omega_k = \omega$ và ω_i được suy dẫn trực tiếp từ ω_{i-1} với $i = 1, 2, .., k$.

Ký hiệu: $\eta \Rightarrow^* \omega$

Dãy $D = \omega_0, \omega_1, .., \omega_k$ được gọi là một *dẫn xuất* của ω từ η trong G và số k được gọi là *độ dài* của dẫn xuất này. Nếu $\omega_0 = S$ và $\omega_k \in \Sigma^*$ thì dãy D gọi là dẫn xuất đầy đủ.

Nếu ω_i được suy dẫn trực tiếp từ ω_{i-1} bằng việc áp dụng một quy tắc p nào đó trong G thì ta nói quy tắc p được *áp dụng* ở bước thứ i.

Định nghĩa 4 (Ngôn ngữ sinh bởi văn phạm)**.**

Cho văn phạm $G = <\Sigma, \Delta, S, P>$. Từ $\omega \in \Sigma^*$ được gọi là *sinh bởi* văn phạm G nếu tồn tại suy dẫn từ S vào ω. **Ngôn ngữ sinh bởi văn phạm** G, ký hiệu $L(G)$, là tập hợp tất cả các từ sinh bởi văn phạm G:

$$L(G) = \{\omega \in \Sigma^* | S \Rightarrow^* \omega\}$$

Định nghĩa 5 (Văn phạm tương đương)**.**

Hai văn phạm $G_1 =< \Sigma_1, \Delta_1, S_1, P_1 >$ và $G_2 =< \Sigma_2, \Delta_2, S_2, P_2 >$ được gọi là tương đương nếu $L(G_1) = L(G_2)$

3.2 Phân loại văn phạm

Dựa vào đặc điểm của tập quy tắc mà người ta chia các văn phạm thành các nhóm khác nhau. Chomsky đã phân loại văn phạm thành 4 nhóm:

- Nhóm 0: Văn phạm không hạn chế (hay văn phạm ngữ cấu, văn phạm tổng quát).
- Nhóm 1: Văn phạm cảm ngữ cảnh.
- Nhóm 2: Văn phạm phi ngữ cảnh.
- Nhóm 3: Văn phạm chính quy.

3.2.1 Văn phạm không hạn chế

Định nghĩa 6 (Văn phạm không hạn chế)**.**

Văn phạm $G = <\Sigma, \Delta, S, P>$ mà không có một ràng buộc nào đối với các quy tắc của nó được gọi là **văn phạm tổng quát** hay **văn phạm không hạn chế**.
Như vậy, các quy tắc trong văn phạm nhóm 0 có dạng:

$$\alpha \to \beta \text{ với } \alpha = \alpha^{'} A \alpha^{"}, A \in \Delta, \alpha^{'}, \alpha^{"}, \beta \in (\Sigma \cup \Delta)^*$$

Các quy tắc của văn phạm nhóm 0 được gọi là **quy tắc không hạn chế**. Ngôn ngữ do văn phạm nhóm 0 sinh ra được gọi là **ngôn ngữ tổng quát**.

3.2.2 Văn phạm cảm ngữ cảnh

Định nghĩa 7 (Văn phạm cảm ngữ cảnh)**.**

Văn phạm $G =< \Sigma, \Delta, S, P >$ mà các quy tắc của nó đều có dạng:

$$\alpha \to \beta \text{ với } \alpha = \alpha^{'} A \alpha^{"}, A \in \Delta, \alpha^{'}, \alpha^{"}, \beta \in (\Sigma \cup \Delta)^*, \text{ và } |\alpha| \leq |\beta|$$

được gọi là văn phạm nhóm 1 hay **văn phạm cảm ngữ cảnh**.

Ví dụ 8.

Cho văn phạm:

$$G =< \{a, b, c\}, \{S, A, B, C\}, S, P >$$

Trong đó:

$$P = \{S \to aSAC, S \to abC, CA \to BA, BA \to BC, BC \to AC, bA \to bb, C \to c\}$$

Khi đó G là văn phạm cảm ngữ cảnh.
Chú ý:

1. Các quy tắc trong văn phạm nhóm 1 được gọi là **quy tắc cảm ngữ cảnh**. Ngôn ngữ do văn phạm cảm ngữ cảnh sinh ra được gọi là **ngôn ngữ cảm ngữ cảnh**.

2. Các văn phạm mà các quy tắc của chúng có dạng trên, đồng thời chứa thêm quy tắc rỗng $S \to \varepsilon$, cũng được xếp vào lớp văn phạm nhóm 1.

3.2.3 Văn phạm phi ngữ cảnh

Định nghĩa 8 (Văn phạm phi ngữ cảnh)**.**

Văn phạm $G =< \Sigma, \Delta, S, P >$ mà các quy tắc của nó có dạng:

$$A \to \omega \text{ với } A \in \Delta, \omega \in (\Sigma \cup \Delta)^*$$

được gọi là văn phạm nhóm 2, hay **văn phạm phi ngữ cảnh**.

Ví dụ 9. *Các văn phạm dưới đây là văn phạm phi ngữ cảnh:*

1. Văn phạm $G_1 =< \{a, b\}, \{S, A\}, S, P_1 >$, trong đó:

$$P_1 = \{S \to Sa, S \to Aa, A \to aAb, A \to ab\}$$

2. Văn phạm $G_2 =< \{0, 1\}, \{S\}, S, P_2 >$, trong đó:

$$P_2 = \{S \to SS, S \to 0S1, S \to 1S0, S \to \varepsilon\}$$

3. Văn phạm $G_3 =< \{a, b\}, \{S\}, S, P_3 >$, trong đó:

$$P_3 = \{S \to \varepsilon, S \to aSa, S \to bSb, S \to aa, S \to bb\}$$

3.2.4 Văn phạm chính quy

Định nghĩa 9 (Văn phạm tuyến tính phải)**.**

Một văn phạm được gọi là **văn phạm tuyến tính phải** nếu mỗi quy tắc của nó ở một trong 2 dạng sau:

- $A \to wB$, với $A, B \in \Delta, w \in \Sigma^*$
- $A \to w$, với $A \in \Delta, w \in \Sigma^*$

Định nghĩa 10 (Văn phạm tuyến tính trái)**.**

Một văn phạm được gọi là **văn phạm tuyến tính trái** nếu mỗi quy tắc của nó ở một trong 2 dạng sau:

- $A \to Bw$, với $A, B \in \Delta, w \in \Sigma^*$
- $A \to w$, với $A \in \Delta, w \in \Sigma^*$

Định nghĩa 11 (Văn phạm chính quy)**.**

Các văn phạm tuyến tính phải và các văn phạm tuyến tính trái được gọi chúng là các văn phạm loại 3 hay **văn phạm chính quy**. Các quy tắc trong văn phạm chính quy được gọi là **quy tắc chính quy**. Ngôn ngữ do văn phạm chính quy sinh ra được gọi là **ngôn ngữ chính quy**.

Ví dụ 10. *Các văn phạm dưới đây là văn phạm chính quy:*

1. Văn phạm $G_1 =< \{1\}, \{S, A, B\}, S, P_1 >$, với:

$$P_1 = \{S \to \varepsilon, S \to 1A, A \to 1B, B \to 1A, A \to 1\}$$

2. Văn phạm $G_2 =< \{0, 1\}, \{S, A\}, S, P_2 >$, với:

$$P_2 = \{S \to 0A, A \to 0A, A \to 1A, A \to 0\}$$

3.3 Một số quy tắc ngữ pháp trong tiếng Anh

3.3.1 Sentence-Level Constructions

Trong tiếng anh có 4 loại câu phổ biến: câu tường thuật (declaratives), câu mệnh lệnh (imperatives), câu hỏi yes-no (yes-no questions), câu hỏi wh (wh-questions).

Câu với cấu trúc **tường thuật (declaratives)** có cụm danh từ là chủ ngữ theo sau bởi cụm động từ. Ví dụ như *I prefer a morning flight*. Các câu với cấu trúc này có một số lượng lớn các công dụng khác nhau. Dưới đây là một số ví dụ:

- I want a flight form Ontario to Chicago
- The flight should be eleven a.m. tomorrow
- The return flight should leave at around seven p.m.

Câu với cấu trúc **mệnh lệnh (imperative)** thường bắt đầu bởi một cụm động từ và không có chủ ngữ. Chúng được gọi là mệnh lệnh bởi vì chúng luôn sử dụng để ra lệnh và đề nghị. Ví dụ:

- Show the lowest fare
- Give me Sunday's flights arriving in Las Vegas from New York City
- List all flights between five and seven p.m.

Chúng ta có thể mô hình cấu trúc các loại câu này như sau:

$$S \rightarrow VP$$

Các câu với cấu trúc **câu hỏi yes-no (yes-no question)** thường (không phải luôn luôn) được sử dụng để hỏi; chúng bắt đầu với một trợ động từ, theo sau là danh từ là chủ ngữ NP, theo sau nữa là một động từ VP. Ví dụ:

- Do any of these flights have stops?
- Does American's flight eighteen twenty five serve dinner?
- Can you give me the same information for United?

Chúng ta có quy tắc như sau:

$$S \rightarrow \text{Aux NP VP}$$

Các cấu trúc câu phức tạp nhất chúng ta xem xét ở đây là cấu trúc **wh-**. Chúng được đặt tên như vậy bởi một trong số các thành phần của chúng là **cụm wh (wh-phrase)**, các cụm này bao gồm **từ wh (wh-word)** (*who, whose, when, where, what, which, how, why*). Phần này có thể được nhóm rộng rãi thành 2 cấu trúc câu: cấu trúc **wh-subject question** và cấu trúc **wh-non-subject question**

wh-subject question

Giống với cấu trúc tường thuật, ngoại trừ việc các cụm danh từ đầu tiên chứa wh-word.
Ví dụ:

- What airlines fly from Burbank to Denver?
- Which flights depart Burbank after noon and arrive in Denver by six p.m?
- Whose flights serve breakfast?

Quy tắc:

$$S \rightarrow \text{Wh-NP vP}$$

wh-non-subject question

Trong cấu trúc câu wh-non-subject question thì wh-parase không là chủ ngữ của câu, và vì vậy trong câu xuất hiện chủ ngữ khác. Trong các câu loại này, trợ từ xuất hiện trước chủ ngữ NP, giooóng như cấu trúc câu hỏi yes-no.
Ví dụ:

What flights do you have from Burbank to Tacoma Washington?

Quy tắc:

$$S \rightarrow \text{Wh-NP Aux NP VP}$$

Các cấu trúc câu như wh-non-subject question bao gồm **long-distance dependencies** bởi vì các từ trong câu có mối quan hệ phụ thuộc xa với nhau.

3.3.2 The Noun Phrase

The Determiner

Từ hạn định (determiner) là các từ như, my, this, some, every, any, được sử dụng trước danh từ: *the* countryside, *some* paper, *this* old sofa, *my* father, *five* green chairs, *each* person, ...
Từ hạn định bao gồm các loại phổ biến sau:

- Mạo từ (Articles): a, an, the
- Đại từ chỉ định (Demonstratives): this, that, those, these
- Đại từ sở hữu (Possessives): my, your, his, her, its, our, their, x's
- Từ định lượng (Quantifiers): (a) few, fewer, (a) little, many, much, more, most, some, any, etc.
- Số (Numbers): one, two, three, etc.

The Nominal

Cấu trúc **danh ngữ (nominal)** theo sau từ hạn định và chứa từ bổ ngữ (modifiers) đằng trước hoặc sau danh từ. Như đã được nêu ra ở ngữ pháp, ở dạng đơn giản nhất, một danh ngữ có thể chỉ chứa 1 danh từ.

$$Nominal \rightarrow Noun$$

Ta sẽ thấy ở dưới đây, quy tắc này đồng thời cung cấp nền tảng cho nhiều quy tắc hồi quy được cùng để biểu diễn các cấu trúc danh ngữ phức tạp hơn.

Before the Head Noun - Trước danh từ chính

Một số loại từ có thể xuất hiện phía trước chủ ngữ chính (the "postdeterminers") trong một danh ngữ. Chúng bao gồm **số đếm, số thứ tự, lượng từ** và tính từ. Ví dụ của số đếm:

two friends one stop

Số thứ tự bao gồm *first, second, third* và vân vân, và đồng thời những từ như *next, last, past, other,* và *another.*

the first one the next day the second leg
the last flight the other American flight

Một số lượng từ *(many, (a) few, several)* chỉ xuất hiện sau danh từ đếm số nhiều

many fares

Tính từ xuất hiện sau lượng từ nhưng trước danh từ

a *first-class* fare a *non-stop* flight
the *longest* layover the *earliest* lunch flight.

Tính từ đồng thời cũng có thể được nhóm vào một cụm từ, gọi là cụm tính từ (adjective phrase hay AP). APs có thể có một trạng từ trước tính từ (xem chương 10 để xem định nghĩa của tinh từ và trạng từ):

the *least expensive* fare

After the Head Noun - Sau giới từ chính

Một danh từ chính có thể được theo sau bởi **postmodifiers**. Ba loại danh ngữ postmodifiers phổ biến trong tiếng Anh:

Cụm giới từ all flights *from Cleveland*
Mệnh đề không hữu hạn any flights *arriving after eleven a.m*
Mệnh đề quan hệ a flight *that serves breakfast* phổ biến trong kho dữ liệu ATIS vì chúng thường được sử dụng để đánh đấu điểm cất cánh và hạ cánh của các chuyến bay.

Dưới đây là một số ví dụ của cụm postmodifier giới từ, với dấu ngoặc được thêm vào để thể hiện ranh giới của mỗi cụm giới từ, với chú ý rằng có thể có 2 hoặc hơn cụm giới từ trong 1 mệnh đề danh ngữ.

all flights *[from Cleveland][to Neward]*
arrival *[in San Jose][before seven p.m]*
a reservation *[on flight six oh six][from Tampa][to Montreal]*
Dưới đây là một quy tắc mới cho cụm giới từ postnominal:

$$Nominal \rightarrow NominalPP$$

Ba loại phổ biến nhất của postmodifiers không xác định là danh động từ (-ing), -ed và các dạng không xác định khác.

Postmodifiers danh động từ thường được dùng phổ biến vì chúng chứa một cụm động từ bắt đầu với dạng danh động từ (-ing). Dưới đây là một số ví dụ:
any of those *[leaving on Thursday]*
any flights *[arriving after eleven a.m]*
flights *[arriving within thirty minutes of each other]*
Chúng ta có thể định nghĩa *Danh ngữ* với bổ ngữ danh động từ như sau: sử dụng một $GerundVP$ không kết thúc mới:

$$Nominal \rightarrow NominalGerundVP$$

Chúng ta có thể tạo ra các quy tắc cho các thành phần của $GerundVP$ bằng việc lặp lại tất các luật sinh VP, thay thế $GerundV$ cho V.

$$GerundVP \rightarrow GerundVNP$$

$$|GerundVPP|GerundV|GerundVNPPP$$

$GerundV$ có thể được định nghĩa như sau:

$$GerundV \rightarrow being|arriving|leaving|...$$

Cụm từ được in nghiêng phía dưới đây là ví dụ của 2 loại mệnh đề không xác định, dạng vô định và $-ed$.
the last flight *to arrive in Boston*
I need to have dinner *served*
Which is the aircraft *used by this flight?*
Một mệnh đề quan hệ hậu danh ngữ (Chính xác hơn: Mệnh đề quan hệ bị giới han), là mệnh đề thường bắt đầu với một **chủ ngữ quan hệ** (thường là *that* và *who*). Chủ ngữ quan hệ được dùng làm chủ ngữ của từ được nhúng, như ở các ví dụ sau:
a flight *that serves breakfast*
flights *that leave in the morning*
the one *that leaves at ten thirty five*
Chúng ta cũng có thể thêm các quy tắc mới như sau:

$$Nominal \rightarrow NominalRelClause$$

$$RelClause \rightarrow (who|that)VP$$

Chủ ngữ tương đối cũng có chức năng như tân ngữ của từ nhúng, như ở ví dụ dưới đây;

the earliest American Airlines flight that I can get

Nhiều bổ ngữ hậu danh ngữ khác có thể được kết hợp, như ở các ví dụ sau:

a flight *[from Phoenix to Detroit] [leaving Monday evening]*

evening flights *[from Nashville to Houston] [that serve dinner]*

a friend *[living in Denver] [that would like to visit me here in Washington DC]*

Before the Noun Phrase - Trước cụm danh từ

Loại từ mà bổ nghĩa và đứng trước cụm danh từ được gọi là tiền định từ (predeterminers). Trong đó, bao gồm số đếm hoặc số lượng. Một tiền định từ phổ biến là *all*:

all the flights all flights all non-stop flights

Ví dụ cụm danh từ được cho bởi hình 11.5 thể hiện một số vấn đề khi kết hợp những quy tắc trên.

3.3.3 The Verb Phrase

Các **cụm động từ (Verb Phrase)** bao gồm động từ và một số các thành phần khác. Những thành phần khác bao gồm NPs và PPs và kết hợp cả hai:

$VP \rightarrow$ *Verb* disappear

$VP \rightarrow$ *Verb NP* prefer a morning flight

$VP \rightarrow$ *Verb NP PP* leave Boston in the morning

$VP \rightarrow$ *Verb PP* leaving on Thursday

Cụm động từ có thể phức tạp hơn đáng kể so với các ví dụ trên, nhiều thành phần khác có thể được nhúng trong câu kèm theo động từ hoặc nhiều thành phần khác, chúng được gọi là **sentential complements**:

You [VP [V said [S you had a two hundred sixty six dollar fare]]
[VP [V Tell] [NP me] [S how to get from the airport in Philadelphia to downtown]]
I [VP [V think [S I would like to take the nine thirty flight]]

Tại đây chúng ta có quy tắc:

$$VP \rightarrow \text{Verb S}$$

VP cũng có thể đi kèm với các động từ như *want, would like, try, intend, need*:

I want [VP to fly from Milwaukee to Orlando]
Hi, I want [VP to arrange three flights]

Trong khi các cụm động từ có thể có nhiều thành phần, không phải tất cả động từ là phù hợp với các cụm động từ. Ví dụ, động từ *want* có thể được sử dụng như là một thành phần của cụm danh từ (*I want a flight*, ...) hoặc như một thành phần của VP (*I want to fly to ...*). Ngược lại, một động từ như *find* không thể là một thành phần của cụm động từ (*I found to fly to Dallas*).

3.3.4 Coordination

Các loại cụm từ được thảo luận ở phần này có thể được nối với nhau bằng các **liên từ (conjunctions** như: *and, or* hoặc *but* để tạo thành các form cùng loại lớn hơn. Ví dụ:

Please repeat $[_{NP}[_{NP}$ the flights] and $[_{NP}$ the costs]]

I need to know $[_{NP}[_{NP}$ the aircraft] and $[_{NP}$ the flight number]]

Tại đây, ta có quy tắc:

$$NP \rightarrow \text{NP and NP}$$

Chúng ta cũng có thể kết hợp các danh ngữ với nhau:

Please repeat the $[_{Nom}$ $[_{Nom}$ flights] and $[_{Nom}$ costs]]

I need to know the $[_{Nom}$ $[_{Nom}$ aircraft] and $[_{Nom}$ flight number]]

Tại đây ta có quy tắc:

$$Nominal \rightarrow \text{Nominal and Nominal}$$

Chúng ta cũng có thể nối các VP với nhau, hoặc các S với nhau:

What flights do you have $[_{VP}$ $[_{VP}$ leaving Denver] and $[_{VP}$ arriving in San Francisco]]

$[_S$ $[_S$ I'm interested in a flight from Dallas to Washington] and $[_S$ I'm also interested in going to

Từ đó ta có quy tắc:

$$VP \rightarrow \text{VP and VP}$$

$$S \rightarrow \text{S and S}$$

Từ các ví dụ trên, chúng ta có quy tắc tổng quát gọi là **metarules** như sau:

$$X \rightarrow \text{X and X}$$

trong đó, X là ký hiệu không kết thúc, từ *and* có thể thay thế bằng các liên từ khác.

3.4 Treebanks

Một vấn đề cốt lõi của xử lý ngôn ngữ tự nhiên là việc gán nhãn cho các dữ liệu văn bản thô, để có được dữ liệu được gán nhãn để có thêm ý nghĩa để nghiên cứu. Dữ liệu được gán nhãn mang ý nghĩa về mặt cú pháp tức là các từ trong một đoạn văn bản dữ liệu ứng với một từ loại (POS tagging), và về mặt ngữ nghĩa tức là văn bản được phân cụm với các chức năng ngữ pháp, hai cách biểu diễn này gọi là gán nhãn từ loại và gán nhãn chức năng, cả hai đều được tổng hợp lại và thể hiện thông qua một cây dẫn xuất có kết quả là văn bản dữ liệu.

Ví dụ 11.

Gán nhãn từ loại văn bản "Tôi đi học" sau khi được gán nhãn là *Tôi/P đi/V học/N* với P: đại từ, V: động từ, N: danh từ.
Gán nhãn chức năng cho biết vai trò của thành phần chức năng cú pháp trong cả văn bản, các thành phần chính trong câu như chủ ngữ, vị ngữ, tân ngữ.

(S (NP (Np Quang))
(VP (V học)
(NP (N bài))
(PP (E trong)
(NP (N thư viện)))))

Stt	idPOS	vnPOS	enPOS
1	N	danh từ	noun
2	V	động từ	verb
3	A	tính từ	adjective
4	P	đại từ	pronoun
5	M	số từ	numeral
6	D	định từ (những, các, vài...)	determiner
7	R	phụ từ	adverb
8	E	giới từ	preposition
9	C	liên từ	conjunction
10	I	trợ từ	auxiliary word
11	O	cảm từ	emotivity word
12	Z	yếu tố cấu tạo từ (bất, vô...)	component stem
13	X	không (hoặc chưa) xác định	undetermined

Hình 3.3: Nhãn từ loại trong Tiếng Việt

Trong ví dụ thứ 2 bên trên, các từ không chỉ được gán nhãn từ loại mà còn được gán nhãn chức năng thể hiện vai trò của chúng trong câu.

Một TreeBank cho một ngôn ngữ là một bộ các văn bản được gán nhãn chức năng cú pháp. Vấn đề xây dựng TreeBank vẫn là một bài toán đang được nghiên cứu để mở rộng kho từ và tăng độ chính xác. Kho dữ liệu TreeBank Tiếng Anh được gán nhãn thành công là kho dữ liệu Peen TreeBank. Kho dữ liệu này được gán nhãn nhờ sử dụng 3 lược đồ chú thích cơ bản: Gán nhãn từ loại, gán nhãn cú pháp, gán nhãn gián đoạn. Tuy đã sử dụng nhiều hình thức gán nhãn để đảm bảo độ chính xác cao, nhưng vẫn không thể tránh việc xảy ra lỗi. Đối với văn bản tiếng Việt, ngày nay cũng đã có nhiều nghiên cứu phục vụ gán nhãn tiếng Việt, ta cũng có VietTreeBank.
Có hai hình thức xây dựng nên TreeBank chủ yếu:

- *Xây dựng dựa trên phương pháp thủ công:* Nhà nghiên cứu (nhà ngôn ngữ học hoặc nhà xử lý ngôn ngữ tự nhiên) phân tích và gán nhãn cú pháp cho các câu theo một định dạng chuẩn. Định dạng thường sử dụng là định dạng của Penn TreeBank.

- *Xây dựng dựa trên phương pháp bán tự động:* Sử dụng các hệ thống phân tích cú pháp (Parse) để phân tích và sửa đổi nếu cần. Sau đó lưu vào TreeBank cần xây dựng.

Để bổ sung dữ liệu vào kho TreeBank, chúng ta có thể sử dụng thuật toán Parsing có đầu ra là một cây dẫn xuất. Đầu vào là một văn phạm phi ngữ cảnh với ký tự bắt đầu là S ứng với vai trò câu, các ký tự không kết thúc là các cấu trúc trong câu, các ký tự kết thúc là các từ trong văn bản. Dưới đây là một ví dụ về văn phạm đầu vào:

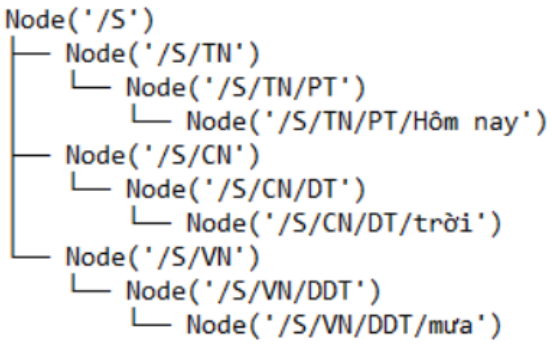
Ví dụ 12.

1. <Câu> → <Chủ ngữ><Vị ngữ>
2. <Câu → <Trạng ngữ><Chủ ngữ><Vị ngữ>
3. <Trạng ngữ> → <Phó từ>
4. <Chủ ngữ> → <Danh ngữ>
5. <Chủ ngữ> → <Danh từ>
6. <Vị ngữ> → <Động ngữ>
7. <Vị ngữ> → <Động từ>

Sau đó gán nhãn từ loại cho câu đầu vào, ta có thêm các luật cú pháp để hoàn chỉnh văn phạm. Câu đầu vào sẽ được phân tách thành các từ token rồi mới gán nhãn.
Ví dụ câu đầu vào là *"Hôm này trời mưa"*, ta có thêm các luật sau:

8. <Danh từ> → trời
9. <Danh từ> → Hôm nay
10. <Phó từ> → Hôm nay
11. <Động từ> → mưa

Khi đó ta được đầu vào là một văn phạm phi ngữ cảnh với các trạng thái kết thúc gồm các ký tự trong xâu đầu vào cần phân tích. Sau đó sử dụng một trong số những thuật toán phân tích để tạo nên một bộ phân tích để đưa ra cây dẫn xuất cho xâu đầu vào. Khi đó ta được một tree để thêm vào TreeBank như hình 3.4.



Hình 3.4: Tree trong TreeBank

Việc có một kho TreeBank lớn sản sinh từ các thuật toán và có độ chính xác cao sẽ là một công cụ hữu hiệu cho các bài toán:

- NLP để số hoá ngôn ngữ, dựa vào mô hình học mát
- Xây dựng máy dịch ngôn ngữ, bán hàng tự động, ...
- Trích rút tự động văn phạm từ Treebank

3.5 Grammar Equivalence and Normal Form

Một ngôn ngữ hình thức được định nghĩa là một chuỗi các từ. Liệu chúng ta có thể biết hai văn phạm tương đương nếu như chúng sinh ra cùng tập các chuỗi như nhau. Trong thực tế, hai văn phạm phi ngữ cảnh khác nhau có thể tạo ra cùng một ngôn ngữ.

Định nghĩa 12 (Tương đương yếu và Tương đương mạnh)**.**

Hai văn phạm được gọi là **tương đương mạnh (strong equivalence)** nếu chúng cùng sinh một bộ xâu và nếu chúng gán cấu trúc cụm từ giống nhau cho mỗi câu (Cho phép chỉ cho đổi tên của những biểu tượng không terminal).

Hai văn phạm được gọi là **tương đương yếu (weak equivalence)** nếu chúng tạo ra cùng một bộ xâu nhưng không gán cùng cụm từ cấu trúc cho mỗi câu.

Định nghĩa 13 (Dạng chuẩn Chomsky)**.**

Văn phạm phi ngữ cảnh $G =< \Sigma, \Delta, S, P >$ được gọi là văn phạm ở dạng **chuẩn chomsky (Chomsky Normal Form - CNF)** nếu mọi quy tắc đều có dạng: $A \to BC$ hoặc $A \to a$ hoặc $A \to \varepsilon$, với $A, B, C \in \Delta, a \in \Sigma$

Ví dụ 13.

Văn phạm $G =< \{a, b\}, \{S, A, B\}, S, P >$ với tập quy tắc P có dạng:

$$S \to SA|AB|BA$$
$$A \to b$$
$$B \to a$$

là một văn phạm ở dạng chuẩn chomsky.

Nhận xét: Các văn phạm ở dạng chuẩn Chomsky thoả mãn điều kiện sau:

a. Không chứa ε-quy tắc với vế trái không phải là ký tự bắt đầu.

b. Không chứa các quy tắc đơn dạng $A \to B$, với $A, B \in \Delta$.

c. Không chứa các quy tắc mà vế phải có cả ký hiệu kết thúc và ký hiệu không kết thúc.

d. Không chứa các quy tắc có vế phải nhiều hơn hai ký hiệu.

Chương 4

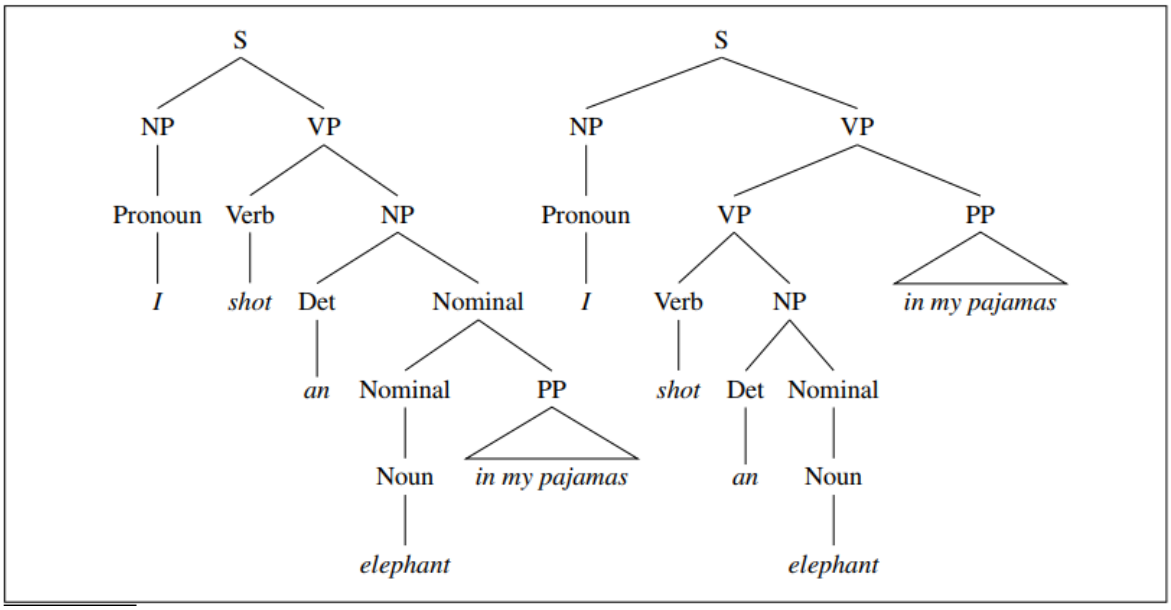
Syntactic Parsing

4.1 Ambiguity

Nhập nhằng (ambiguity) có lẽ là vấn đề nghiêm trọng nhất mà các trình phân tích cú pháp phải đối mặt. Trong phần này, chúng ta sẽ tìm hiểu về **nhập nhằng cấu trúc (structural ambiguity)**, phát sinh từ các quy tắc thường được sử dụng trong cấu trúc ngữ pháp. Sự mô hồ về cấu trúc xảy ra khi có thể gán nhiều hơn một phân tích cho một câu. Ví dụ như câu:

I shot an elephant in my pajamas

chúng ta có 2 cách phân tích như hình 4.1.



Hình 4.1: Hai cây phân tích cho một câu nhập nhằng

Sự nhập nhằng về cấu trúc có thể chia làm 2 loại: **attachment ambiguity** và **coordination ambiguity**.

Một câu được gọi là **attachment ambiguity** nếu như một thành phần cụ thể có thể được gán vào các cây phân tích nhiều hơn một vị trí. Ví dụ như câu:

We saw the Eiffel Tower flying to Paris

từ *flying to Paris* có thể là một cụm động từ cho chủ ngữ là *Eiffel Tower* hoặc nó cũng có thể là một cụm từ bổ trợ cho động từ *saw*

Trong **coordination ambiguity**, các cụm từ có thể được nối với nhau bằng các cách khác nhau. Ví dụ như cụm từ *old men and women* có thể được phân tích là *[old [men and women]]*, có nghĩa là *old men* và *old women*, hoặc cũng có thể phân tích là *[old men] and [women]*, có nghĩa là chỉ đàn ông là già.

Hầu hết các hệ thống xử lý ngôn ngữ tự nhiên cần có khả năng chọn một phân tích chính xác duy nhất từ vô số các phân tích cú pháp có thể thông qua một quá trình phân định cú pháp. Thuật toán CKY được trình bày trong phần tiếp theo được thiết kế để xử lý hiệu quả sự nhập nhằng về cấu trúc. Và như chúng ta sẽ thấy trong chương sau, có những cách đơn giản để tích hợp các kỹ thuật thống kê vào khung CKY cơ bản để tạo ra các trình phân tích cú pháp có độ chính xác cao.

4.2 CKY Parsing: A Dynamic Programming Approach

4.2.1 Rút gọn văn phạm

Loại bỏ các quy tắc ε ($\varepsilon - production$)

Trong CFG, các quy tắc ε trong đa số trường hợp có thể loại bỏ, trừ trường hợp văn phạm đó sinh ra xâu rỗng ($S \to \varepsilon$). Một ký tự không kết thúc A được gọi là một biến null nếu có một quy tắc dạng $A \to \varepsilon$ hoặc có dẫn xuất bắt đầu từ A và kết thúc là ε ($A \to ... \to \varepsilon$).

Thuật toán:

Step 1: Tìm các biến không kết thúc dẫn xuất đến ε ($A \to ... \to \varepsilon$).

Step 2: Với mỗi quy tắc $A \to a$, xây dựng tất cả các quy tắc $A \to x$ với x thu được từ a bằng cách xóa một hoặc nhiều non-terminal ở Step 1.

Step 3: Tổ hợp các quy tắc với kết quả thu được ở bước 2 và xóa bỏ $\varepsilon - production$

Ví dụ 14. *Xóa bỏ các quy tắc ε từ tập các quy tắc sau:*

$$S \to ASA|aB|b$$
$$A \to B$$
$$B \to b|\varepsilon$$

Lời giải:
Các biến null là: A, B
Đầu tiên, chúng ta xóa bỏ quy tắc $B \to \varepsilon$, thu được tập quy tắc như sau:

$$S \to ASA|aB|b|a$$
$$A \to B|\varepsilon$$
$$B \to b$$

Tiếp theo, chúng ta xóa bỏ quy tắc $A \to \varepsilon$:

$$S \to ASA|aB|b|a|SA|AS|S$$
$$A \to B$$
$$B \to b$$

Đây là tập các quy tắc cuối cùng và không còn quy tắc ε.

Loại bỏ các quy tắc đơn (unit production)

Quy tắc đơn là quy tắc có dạng $A \to B$ trong đó B là một ký hiệu không kết thúc. Trên phương diện cú pháp thuần túy thì quy tắc đơn thường làm kéo dài các suy dẫn, cho nên cũng có khi ta đặt vấn đề loại bỏ chúng. Tuy nhiên, trên phương diện ngữ nghĩa, thì quy tắc đơn có khi cũng hữu ích. Có thể loại bỏ các quy tắc đơn ra khỏi văn phạm mà không làm thay đổi ngôn ngữ.
Thuật toán:
Step 1: Loại bỏ các quy tắc đơn $A \to B$ trong P, các quy tắc đơn $A \to B$ mà quy tắc $B \to A$ cũng thuộc P cần được loại bỏ trước.
Step 2: Thêm vào P quy tắc $A \to \alpha$ nếu có quy tắc $B \to \alpha$ trong P $(A \neq \alpha)$
Step 3: Nếu trong P xuất hiện các quy tắc đơn mới, lặp lại bước 1.

Ví dụ 15.

Cho CFG $G =< \{a, b\}, \{S, A, B\}, S, P >$ với:

$$P = \{S \to Aa|B, A \to b|B, B \to A|a\}$$

Lời giải:

1. Loại bỏ quy tắc đơn $A \to B$ trong P.
2. Bổ sung các quy tắc $A \to a$ vào trong P, ta được $P = \{S \to Aa|B, A \to b|a, B \to A|a\}$
3. **Lặp lại từ bước 1**.

1. Loại bỏ các quy tắc đơn $B \to A$ trong P.

2. Bổ sung quy tắc $B \to b|a$ vào trong P, ta được $P = \{S \to Aa|B, A \to b|a, B \to b|a\}$

3. **Lặp lại từ bước 1**.

1. Loại bỏ các quy tắc đơn $S \to B$ trong P.

2. Bổ sung quy tắc $S \to b|a$ vào P, ta được $P = \{S \to Aa|b|a, A \to b|a, B \to b|a\}$

3. Ta thấy P không còn quy tắc đơn, kết thúc thuật toán.

Loại bỏ các dẫn xuất vô sinh

Dẫn xuất không tham gia vào quá trình sinh chuỗi gọi là dẫn xuất vô sinh.

Thuật toán

Step 1: Đặt $T = \sum$ (bao gồm tất cả các ký tự bảng chữ)

Step 2: $\mathrm{W}_1 = \{A : A \to x, x \in T, A \in \Delta\}$

Step 3: Tính W_{i+1}, bao gồm tất cả các ký tự không kết thúc dẫn xuất đến các ký tự không kết thúc của W_i.

Step 4: $i = i + 1$ và lặp lại Step 3 cho đến khi $W_{i+1} = W_i$.

Step 5: $G' = \{\sum, \Delta', P', S\}$. Trong đó:

Δ' bao gồm các ký tự không kết thúc có trong W_i

P' bao gồm các quy tắc mà có xuất hiện phần tử của W_i trong đó.

Ví dụ 16.

Cho $G = \{\sum, \Delta, P, S\}$, trong đó:

$\sum = \{a, b\}$

$\Delta = \{S, A, B, C\}$

$P = \{S \to aA|bAB|abA, A \to aB|ba|a, B \to aB|bB, C \to aA|bS|a\}$

Lời giải:

Đầu tiên, ta có: $T = \{a, b\}$

$W_1 = \{A, C\}$ (Do tồn tại quy tắc $A \to a$ và $C \to a$)

$W_2 = \{A, C, S\}$ (Do tồn tại quy tắc $S \to aA$ hoặc $S \to abA$)

$W_3 = \{A, C, S\} = W_2$ nên dừng thuật toán.

Từ đó thu được $G' = \{\sum, \Delta', P', S\}$ với:

$\Delta' = \{S, A, C\}$

$P' = \{S \to aA|abA, A \to ba|a, C \to aA|bS|a\}$

Ví dụ 17.

Cho $G = \{\sum, \Delta, P, S\}$, với:

$P = \{S \to AC|B, A \to a, C \to c|BC, E \to aA|e\}$

Lời giải:

Ta có, $T = \{a, c, e\}$

$W_1 = \{A, C, E\}$ (Do tồn tại quy tắc $A \to a$, $C \to c$ và $E \to e$)
$W_2 = \{A, C, E\} \cup \{S\}$ (Do tồn tại quy tắc $S \to AC$)
$W_3 = \{A, C, E, S\} = W_2$, nên dừng thuật toán.
Từ đó thu được $G' = \{\{A, C, E, S\}, \{a, c, e\}, P', \{S\}\}$, với:
$P' = \{S \to AC, A \to a, C \to c, E \to aA|e\}$

Loại bỏ các dẫn xuất không đến được

Dẫn xuất không đến được là dẫn xuất mà từ trạng thái đầu S, không thể đi đến và sử dụng dẫn xuất đó để sản sinh chuỗi.
Thuật toán:
Step 1: $i = 1$, $Y = \{S\}$
Step 2: $Y_{i+1} = \{$Tập các ký tự có thể dẫn xuất từ $Y_i\}$
Step 3: $i = i + 1$, lặp lại Step 2 cho đến khi $Y_{i+1} = Y_i$

Ví dụ 18.

Từ kết quả ví dụ trên, ta có:
$Y_1 = \{S\}$
$Y_2 = \{S, A, C\}$ (Từ quy tắc $S \to AC$
$Y_3 = \{S, A, C, a, c\}$ (Từ quy tắc $A \to a$, và $C \to c$)
$Y_4 = \{S, A, C, a, c\} = Y_3$, nên dừng thuật toán và thu được tập các quy tắc sau khi rút gọn là:
$P = \{S \to AC, A \to a, C \to c\}$

4.2.2 Chuyển CFG sang dạng chuẩn Chomsky

Thuật toán này được sử dụng như một bước tiền xử lý cho nhiều thuật toán, như là thuật toán CYK. Đầu vào là một văn phạm phi ngữ cảnh, đầu ra là văn phạm ở dạng chuẩn Chomsky.
Thuật toán:
Step 1: Nếu ký tự bắt đầu S xuất hiện ở bên phải một quy tắc nào đó thì tạo một trạng thái mới S' và một quy tắc $S' \to S$.
Step 2: Xóa bỏ các $\varepsilon - production$.
Step 3: Xóa bỏ các quy tắc đơn (unit production).
Step 4: Thay mỗi quy tắc $A \to B_1B_2...Bn, n > 2$ thành $A \to B_1C$ và $C \to B_2...B_n$.
Lặp lại cho tất cả các quy tắc có nhiều hơn 2 ký tự ở bên phải dẫn xuất.
Step 5: Nếu quy tắc có dạng $A \to aB$ với a là terminal và B là non-terminal. Thì thay thành $A \to XB$ và $X \to a$.
Lặp lại cho tất cả các quy tắc có dạng $A \to aB$.

Ví dụ 19. *Chuyển CFG với các quy tắc sau sang dạng chuẩn Chomsky:*

$$S \to ASA|aB$$
$$A \to B|S$$
$$B \to b|\varepsilon$$

Lời giải:
Step 1: Do tồn tại quy tắc $A \to S$ nên ta thêm trạng thái S' và $S' \to S$, thu được tập các quy tắc như sau:

$$S' \to S$$
$$S \to ASA|aB$$
$$A \to B|S$$
$$B \to b|\varepsilon$$

Step 2: Xóa bỏ các quy tắc ε:
Xóa bỏ $B \to \varepsilon$, tập quy tắc trở thành:

$$S' \to S, S \to ASA|aB|a, A \to B|S|\varepsilon, B \to \varepsilon$$

Xóa bỏ $A \to \varepsilon$, tập các quy tắc trở thành:

$$S' \to S, S \to ASA|aB|a|AS|SA|S, A \to B|S, B \to b$$

Step 3: Xóa bỏ các unit production (quy tắc đơn):
Các trạng thái đơn cần xóa bỏ là:

$$S' \to S, S \to S, A \to B, A \to S$$

* Xóa $S \to S$:

$$S' \to S, S \to ASA|ab|AS|SA, A \to B|S, B \to b$$

* Xóa $S' \to S$:

$$S' \to ASA|aB|a|AS|SA, S \to ASA|aB|a|AS|SA, A \to B|S, B \to b$$

* Xóa $A \to B$:

$$S' \to ASA|aB|a|AS|SA, S \to ASA|aB|a|AS|SA, A \to S|b, B \to b$$

* Xóa $A \to S$:

$$S' \to ASA|aB|a|AS|SA$$
$$S \to ASA|aB|a|AS|SA$$
$$A \to ASA|aB|a|AS|SA|b$$
$$B \to b$$

Step 4: Chúng ta sẽ tìm các quy tắc mà có nhiều hơn 2 biến ở vế phải:

$$S^{'} \to ASA, S \to ASA, A \to ASA$$

Thực hiện thuật toán, thu được tập quy tắc sau:

$$\begin{aligned}
S^{'} &\to AX|aB|a|AS|SA \\
S &\to AX|aB|a|AS|SA \\
A &\to AX|aB|a|AS|SA|b \\
B &\to b \\
X &\to SA
\end{aligned}$$

Step 5: Thay đổi các quy tắc:

$$S^{'} \to aB, S \to aB, A \to aB$$

Thực hiện thuật toán, thu được tập các quy tắc sau:

$$\begin{aligned}
S^{'} &\to AX|YB|a|AS|SA \\
S &\to AX|YB|a|AS|SA \\
A &\to AX|YB|a|AS|SA|b \\
B &\to b \\
X &\to SA \\
Y &\to a
\end{aligned}$$

4.2.3 Thuật toán CYK

Phân tích ngôn ngữ phi ngữ cảnh (Parsing of context-free language) là một trong những ứng dụng quan trọng của lí thuyết ngôn ngữ. Một trong những bài toán quyết định quan trọng trong ngôn ngữ phi ngữ cảnh là:

Cho văn phạm phi ngữ cảnh G và một xâu w, xác định phải chăng $w \in G$?

Một trong những thuật toán để giải quyết bài toán trên là **thuật toán CYK (Cocke-Younger-Kasami)**, nó cho phép xác định liệu $w \in L(G)$ không với độ phức tạp thời gian là $O(|w|^3)$.

Thuật toán CYK nhận đầu vào là một văn phạm G đã ở dạng chuẩn Chomsky và xâu $w = a_1a_2 \dots a_n$, thuật toán sử dụng một bảng kích thước $(n+1) \times (n+1)$ gọi là **ma trận tam giác trên ngặt (strictly upper triangular matrix)**, các phần tử của bảng (ma trận) này là các tập hợp. Đầu ra là yes/no xác định xem xâu $w \in L(G)$ hay không?

Ma trận $T = [t_{ij}]_{n+1,n+1}$ được gọi là ma trận tam giác trên ngặt nếu như: $t_{ij} = 0, \forall i \geq j$:

$$T = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

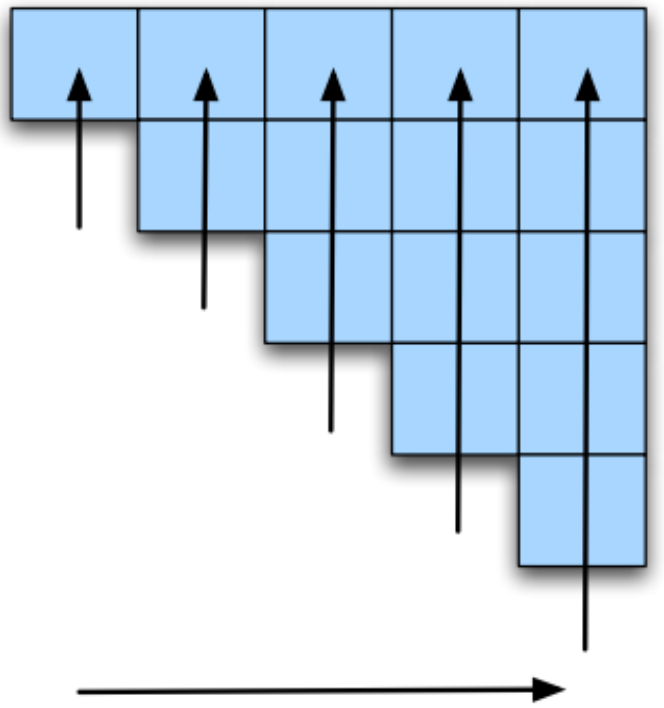
Thuật toán CYK được xác định như hình 4.2. $w \in L(G)$ khi và chỉ khi $S \in t_{0,n}$

```
function CKY-PARSE(words, grammar) returns table

  for j ← from 1 to LENGTH(words) do
      for all {A | A → words[j] ∈ grammar}
              table[j − 1, j] ← table[j − 1, j] ∪ A
      for i ← from j − 2 downto 0 do
          for k ← i + 1 to j − 1 do
              for all {A | A → BC ∈ grammar and B ∈ table[i,k] and C ∈ table[k, j]}
                      table[i,j] ← table[i,j] ∪ A
```

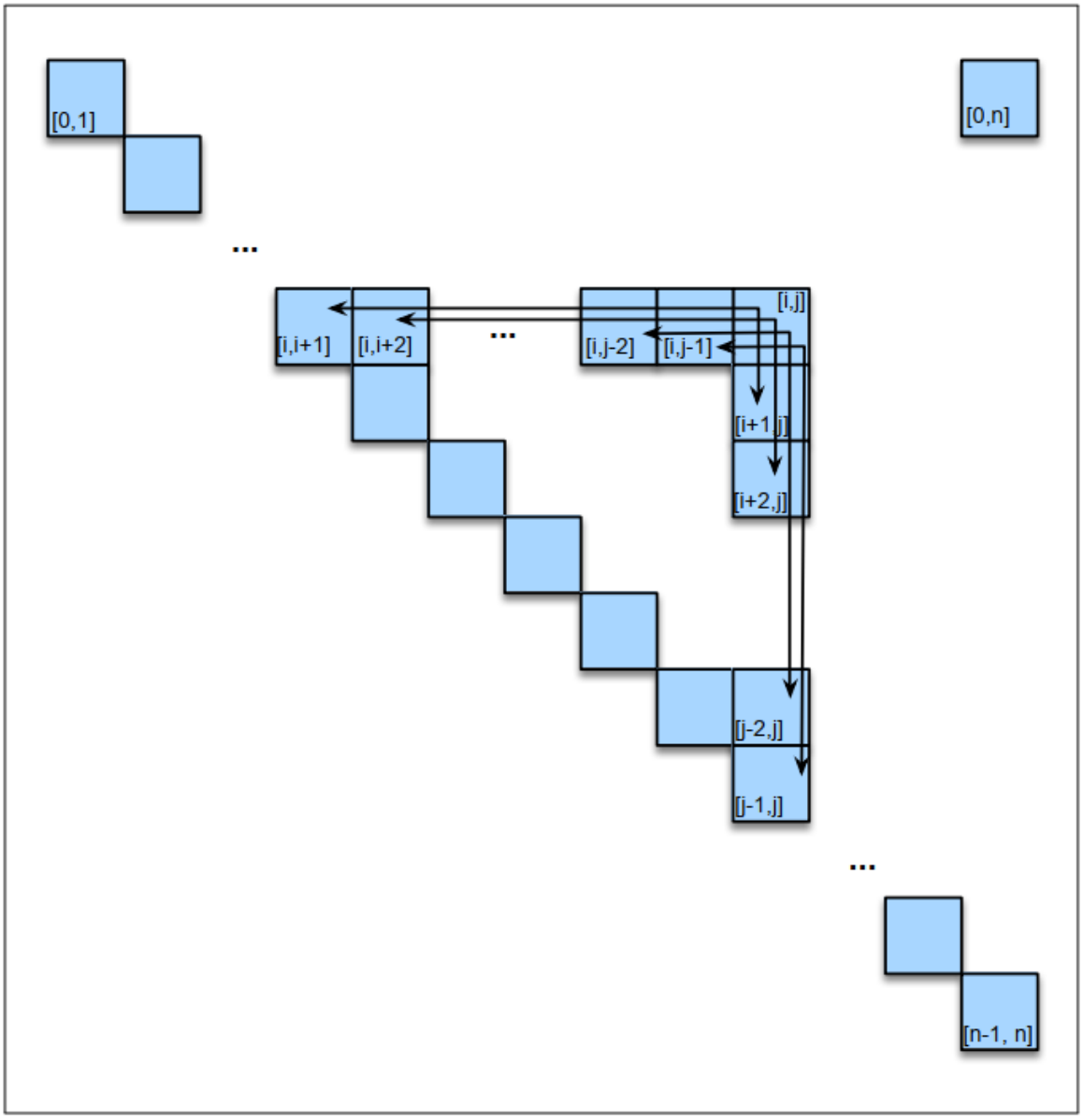
Hình 4.2: Thuật toán CYK

Từ thuật toán, ta có thể thấy rằng, thứ tự điền vào bảng của thuật toán CYK sẽ như hình 4.3.



Hình 4.3: Thứ tự điền vào bảng của thuật toán CYK

Tại mỗi vị trí (i,j), thì sẽ có những cách để điền vào ô đó như hình 4.4.



Hình 4.4: Cách điền ô (i,j) của bảng của thuật toán CYK

Ví dụ 20.

Cho văn phạm chính quy ở dạng chuẩn Chomsky G có tập quy tắc như sau:

$$S \to AB|BC$$
$$A \to BA|a$$
$$B \to CC|b$$
$$C \to AB|a$$

xâu $w = baaba$ có thuộc ngôn ngữ sinh bởi G hay không ($w \in L(G)$)?

Lời giải

$t_{0,1} = \{A \mid A \Rightarrow a_1 = b$ là suy dẫn trong P$\} = \{B\}$

$t_{1,2} = \{A, C\}$

$t_{2,3} = \{A, C\}$

$t_{3,4} = \{B\}$

$t_{4,5} = \{A, C\}$

$t_{0,2} = t_{0,1} * t_{1,2} = \{A \mid A \Rightarrow \{B\} * \{A, C\}\} = \{A \mid A \Rightarrow BA, BC\} = \{A, S\}$

$t_{1,3} = \{B\}$

$t_{2,4} = \{S, C\}$

$t_{3,5} = \{S, A\}$

$t_{0,3} = t_{0,1} * t_{1,3} \cup t_{0,2} * t_{2,3} = \{A \mid \Rightarrow \{B\} * \{B\} \cup \{A, S\} * \{A, C\}\} = \emptyset$

$t_{1,4} = \{B\}$

$t_{2,5} = \{B\}$

$t_{0,4} = t_{0,1} * t_{1,4} \cup t_{0,2} * t_{2,4} \cup t_{0,3} * t_{3,4} = \{A \mid A \Rightarrow \{B\} * \{B\} \cup \{A, S\} * \{S, C\} \cup \emptyset * \{B\}\} = \emptyset$

$t_{1,5} = \{S, A, C\}$

Cuối cùng :

$t_{0,5} = t_{0,1} * t_{1,5} \cup t_{0,2} * t_{2,5} \cup t_{0,3} * t_{3,5} \cup t_{0,4} * t_{4,5} = \{S, A, C\}$.

Do $S \in t_{0,5}$. Theo **thuật toán CYK** suy ra $w \in L(G)$

Id	0	1	2	3	4	5
0		{B}	{A,S}	Ø	Ø	{S,A,C}
1			{A,C}	{B}	{B}	{S,A,C}
2				{A,C}	{S,C}	{B}
3					{B}	{S,A}
4						{A,C}
5						

4.3 Phân tích cú pháp một phần (Partial Parsing)

Nhiều nhiệm vụ xử lý không yêu cầu phức tạp, chúng ta không cần phải phân tích cú pháp hoàn chỉnh cho tất cả đầu vào. Mà đôi khi chỉ cần **phân tích cú pháp một phần (partial parsing)** hoặc **phân tích nông (shallow parse)**. Ví dụ như hệ thống trích xuất thông tin thường không trích xuất tất cả thông tin có thể từ một văn bản, mà chỉ đơn giản là xác định và phân loại các phân đoạn của một văn bản có khả năng chứa thông tin có giá trị.

Có nhiều cách khác nhau để tiếp cận phân tích cú pháp một phần, một trong số những kiểu phân tích cú pháp một phần là **chunking**. Chunking là quá trình xác định và phân loại các phân đoạn không chồng chéo của một câu thành các cụm từ (không lồng nhau) tương ứng với các thành phần của POS. Tập các nhãn POS có thể là cụm danh từ (NP), cụm động từ (VP), cụm tính từ (AP), cụm giới từ (PP), ... Các văn bản đã được chunked không có cấu trúc phân cấp, vì vậy ta có thể biểu diễn bằng các dấu ngoặc vuông ví dụ như sau:

$$[_{NP} \text{ The morning flight}] [_{PP} \text{ from}] [_{NP} \text{ Denver}] [_{VP} \text{ has arrived.}]$$

Ký hiệu dấu ngoặc này làm rõ hai nhiệm vụ cơ bản có liên quan đến phân đoạn: tìm phạm vi không chồng chéo của các khối và gán nhãn chính xác cho các khối được phát hiện.

Lưu ý là nhiều từ, cụm từ có thể không nằm trong phân đoạn nào, ví dụ như bài toán tìm cụm danh từ trong câu ta có:

$$[_{NP} \text{ The morning flight}] \text{ from } [_{NP} \text{ Denver}] \text{ has arrived}$$

4.3.1 Tiếp cận dựa trên học máy cho Chunking

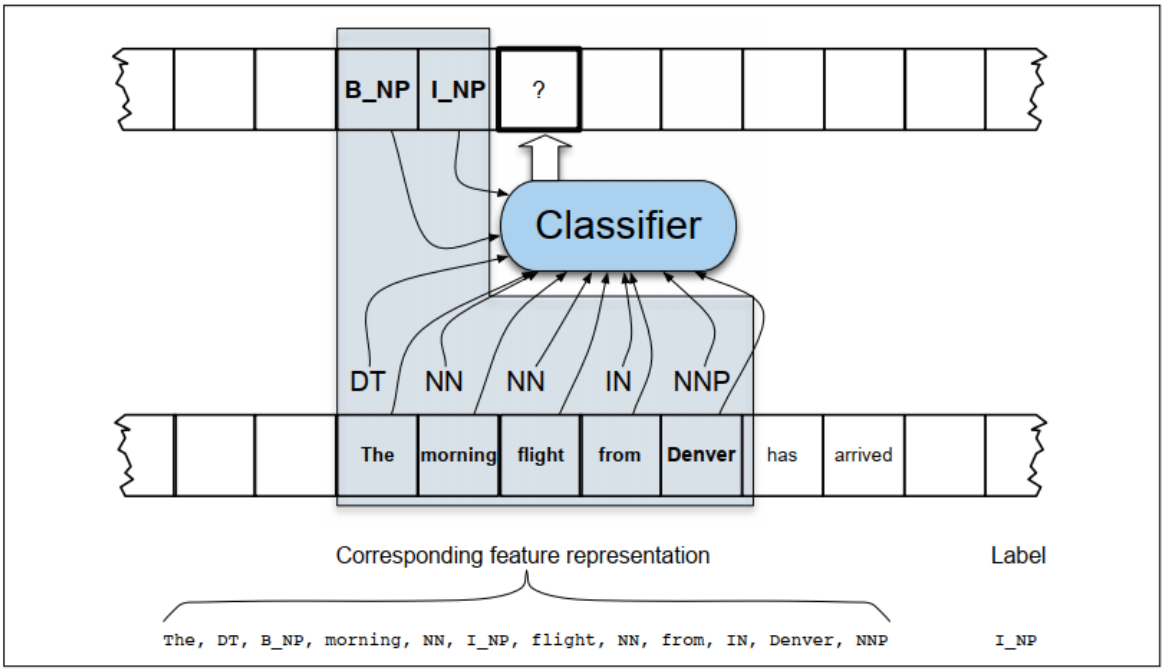
Ở chương POS tagging, chúng ta gán thẻ cho từng từ của chuỗi đầu và. Trong phần này, một cách tiếp cận đặc biệt hiệu quả hơn cho chunking là sử dụng tiêu chuẩn **IOB tagging**, theo đó, phát hiện các cụm phân đoạn trong câu bằng các thẻ phần đầu (B), phần giữa là (I), và ngoài bất kỳ đoạn nào là (O). Theo tiêu chuẩn này, kích thước bộ thẻ (nhãn) sẽ là $2 \times n + 1$ trong đó n là số danh mục ta cần phân loại. Ví dụ dưới đây cho thấy cách đánh nhãn theo tiêu chuẩn trên:

The morning	flight from	Denver has	arrived
B_NP I_NP	I_NP B_PP	B_NP B_VP	I_VP

tương tự như vậy, với bài toán chỉ phân loại cụm danh từ (NP), ta có:

The morning	flight from	Denver has	arrived
B_NP I_NP	I_NP O	B_NP O	O

Lưu ý rằng chúng ta không đánh dấu kết thúc của một cụm, chúng ta phân biệt các cụm bằng cách nhận thấy sự chuyển đổi từ thẻ B hoặc I sang thẻ B hoặc O. Để đào tạo mô hình theo tiêu chuẩn IOB như trên, chúng ta cần các dữ liệu được gán nhãn theo tiêu chuẩn trên. Cách tiếp cận trực tiếp là xây dựng một bộ dữ liệu gán nhãn theo quy tắc tiêu chuẩn này. Tuy nhiên, cách này có thể tốn kém thời gian, vì vậy, chúng ta có thể tận dụng sẵn các treebank hiện có để tạo các bộ dữ liệu có nhãn, ví dụ trong tiếng Anh là Penn TreeBank. TreeBank cung cấp phân tích cú pháp hoàn chỉnh, chúng ta phải chuyển nó sang hình thức hữu ích cho việc huấn luyện chunking của chúng ta.



Hình 4.5: Mô phỏng việc huấn luyện mô hình

4.3.2 Đánh giá hệ thống Chunking

Không giống như POS tagging ở các chương trước, các hệ thống Chunking, việc đánh giá bằng độ đo **accuracy** để xem số từ dự đoán đúng trên tổng số từ là không phù hợp. Accuracy chỉ phù hợp với các bài toán mà kích thước các lớp dữ liệu là tương đối như nhau. Thay vào đó, sử dụng các độ đo **precision**, **recall**, và **F-measure**.

Precision

Precision đo tỷ lệ phần trăm đúng của hệ thống. Đúng ở đây có nghĩa là cả ranh giới các cụm và nhãn trong các cụm chính xác. Đó đó precision được định nghĩa là:

$$\textbf{precision} = \frac{\text{Tổng số chunk đúng của hệ thống}}{\text{Tổng số chunk của hệ thống}}$$

Recall

Recall đo tỷ lệ phần trăm của chunk thực sự hiện diện trong đầu vào mà đã được xác định một cách chính xác bởi hệ thống:

$$\textbf{recall} = \frac{\text{Tổng số chunk đúng của hệ thống}}{\text{Tổng số chunk thực tế trong văn bản}}$$

F-measure

F-measure là tổ hợp của hai độ đo trên và được định nghĩa như sau:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

Tham số β đánh giá trọng số cho tầm quan trọng của recall và precision. Giá trị $\beta > 1$ nâng tầm quan trọng của recall, trong khi $\beta < 1$ nâng tầm quan trọng của precision. Khi $\beta = 1$, precision và recall là cân bằng nhau; gọi là $F_{\beta=1}$ hoặc độ đo F_1:

$$F_1 = \frac{2PR}{P + R}$$

F-measure xuất phát từ trung bình điều hoà (Harmonic Mean) cho recall và precision.

$$\text{HarmonicMean}(a_1, a_2, \ldots, a_n) = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \ldots + \frac{1}{a_n}}$$

Và áp dụng cho F-measure thành:

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha)R}$$

với $\beta^2 = \frac{1-\alpha}{\alpha}$. Từ đó thu được công thức cuối cùng của F-measure:

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

Chương 5

Statistical Parsing

Trong xử lý ngôn ngữ tự nhiên, phân tích cú pháp có thể được chia làm nhiều cấp độ: cấp độ thấp nhất là dừng ở mức tách từ và gán nhãn từ loại cho câu, cho đến cấp độ cao nhất là cây ngữ pháp hoàn chỉnh thể hiện sự phụ thuộc giữa các thành phần trong câu. Bài toán phân tích cú pháp ở mức cao nhất là một bài toán khó, do có quá nhiều cây cú pháp hoàn chỉnh tương ứng với một câu đầu vào, tạo nên sự nhập nhằng khiến độ chính xác của hệ thống phân tích cú pháp có thể rất thấp. Trong chương này sẽ trình bày phương pháp phân tích cú pháp xác suất để giải quyết sự nhập nhằng này.

5.1 Văn phạm phi ngữ cảnh xác suất

Mở rộng đơn giản nhất của văn phạm phi ngữ cảnh là **văn phạm phi ngữ cảnh xác suất (Probabilistic context-free language) (PCFG)**, hay còn được gọi là **văn phạm phi ngữ cảnh ngẫu nhiên (Stochatic context-free language) (SCFG)**, lần đầu được giới thiệu bởi Booth (1969). Nhắc lại, văn phạm G được định nghĩa bởi 4 tham số (Σ, Δ, S, P). Một văn phạm phi ngữ cảnh xác suất cũng được định nghĩa bởi 4 tham số, với một mở rộng nhỏ cho mỗi quy tắc trong P:

- Σ: Tập các ký hiệu kết thúc (terminal).
- Δ: Tập các ký hiệu không kết thúc (non-terminal).
- S: Ký hiệu bắt đầu.
- P: Tập các quy tắc (rules), mỗi quy tắc có dạng $A \to \beta[p]$ với A là một ký hiệu không kết thúc, $\beta \in (\Sigma \cup \Delta)^*$ và p là một số nằm trong khoảng $(0, 1)$ biểu diễn $P(\beta|A)$.

Văn phạm phi ngữ cảnh xác suất khác với văn phạm phi ngữ cảnh thông thường bởi mỗi quy tắc mở rộng ra bằng cách thêm xác suất có điều kiện:

$$A \to \beta[p]$$

Tại đây, p biểu thị xác suất mà cho ký hiệu không kết thúc A sẽ mở rộng đến chuỗi β, p là xác xuất có điều kiện $P(\beta|A)$. Chúng ta có thể biểu diễn như sau:

$$P(A \rightarrow \beta)$$

hoặc:

$$P(A \rightarrow \beta|A)$$

và ta có:

$$\sum_{\beta} P(A \rightarrow \beta) = 1$$

Hình 5.1 biểu diễn một văn phạm phi ngữ cảnh xác suất.

Grammar		Lexicon
S → *NP VP*	[.80]	*Det* → *that* [.10] | *a* [.30] | *the* [.60]
S → *Aux NP VP*	[.15]	*Noun* → *book* [.10] | *flight* [.30]
S → *VP*	[.05]	| *meal* [.015] | *money* [.05]
NP → *Pronoun*	[.35]	| *flight* [.40] | *dinner* [.10]
NP → *Proper-Noun*	[.30]	*Verb* → *book* [.30] | *include* [.30]
NP → *Det Nominal*	[.20]	| *prefer* [.40]
NP → *Nominal*	[.15]	*Pronoun* → *I* [.40] | *she* [.05]
Nominal → *Noun*	[.75]	| *me* [.15] | *you* [.40]
Nominal → *Nominal Noun*	[.20]	*Proper-Noun* → *Houston* [.60]
Nominal → *Nominal PP*	[.05]	| *NWA* [.40]
VP → *Verb*	[.35]	*Aux* → *does* [.60] | *can* [40]
VP → *Verb NP*	[.20]	*Preposition* → *from* [.30] | *to* [.30]
VP → *Verb NP PP*	[.10]	| *on* [.20] | *near* [.15]
VP → *Verb PP*	[.15]	| *through* [.05]
VP → *Verb NP NP*	[.05]	
VP → *VP PP*	[.15]	
PP → *Preposition NP*	[1.0]	

Hình 5.1: Ví dụ về văn phạm phi ngữ cảnh xác suất

Một PCFG được gọi là **thích hợp (consistent)** nếu tổng các xác suất của tất cả các câu trong văn phạm bằng 1. Một số văn phạm chứa quy tắc đệ quy gây ra văn phạm **không thích hợp (inconsisten)**, vì có thể lặp vô hạn dẫn xuất cho một vài câu nào đó. Ví dụ như văn phạm chứa quy tắc $S \rightarrow S$ với xác suất 1 sẽ là một văn phạm không thích hợp.

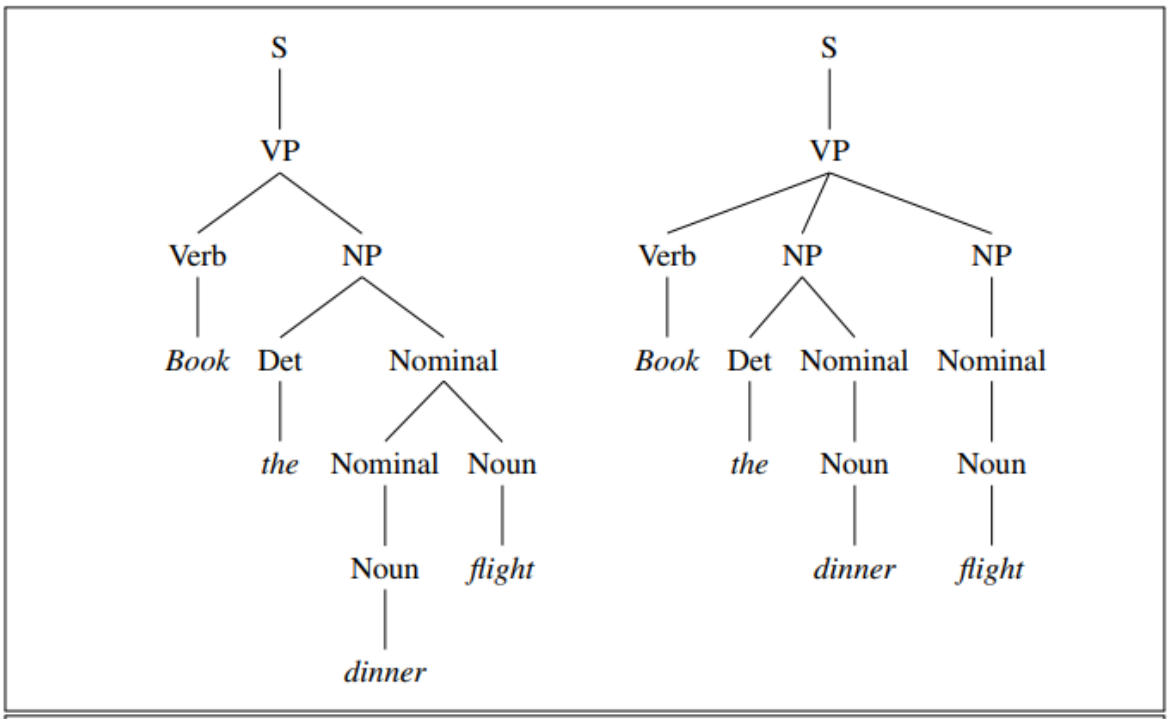
Văn phạm phi ngữ cảnh xác suất có thể được sử dụng để ước lượng xác suất cho một câu (hữu ích cho language modeling) hoặc một cây phân tích (hữu ích cho disambiguation). Trong các phần tiếp theo chúng ta sẽ tìm hiểu tác dụng, cách sử dụng của văn phạm phi ngữ cảnh xác suất.

5.1.1 PCFG for Disambiguation

Một PCFG chỉ định một xác suất cho mỗi cây phân tích cú pháp T. Đặc tính này hữu ích cho xử lý nhập nhằng. Ví dụ xem xét hai cây phân tích cú pháp của câu:

Book the dinner flight

được cho bởi hình 5.2. Phân tích hợp lý ở bên trái nghĩa là *Book a flight that serves dinner*, cây phân tích không hợp lý là cây ở bên phải.



	Rules		P		Rules		P
S	→	VP	.05	S	→	VP	.05
VP	→	Verb NP	.20	VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20	NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20	NP	→	Nominal	.15
Nominal	→	Noun	.75	Nominal	→	Noun	.75
				Nominal	→	Noun	.75
Verb	→	book	.30	Verb	→	book	.30
Det	→	the	.60	Det	→	the	.60
Noun	→	dinner	.10	Noun	→	dinner	.10
Noun	→	flight	.40	Noun	→	flight	.40

Hình 5.2: Hai cây phân tích cú pháp cho một câu nhập nhằng

Các xác suất của một cây phân tích cú pháp cụ thể T được định nghĩa là tích của các xác suất của tất cả n quy tắc được sử dụng để tạo nên cây đó, tại đây mỗi quy tắc

i có dạng $LHS_i \rightarrow RHS_i$ (left-hand-side $\rightarrow$ right-hand-side):

$$P(T, S) = \prod_{i=1}^{n} P(RHS_i|LHS_i)$$

Xác suất kết quả $P(T, S)$:

$$P(T, S) = P(T)P(S|T)$$

Nhưng một cây phân tích cú pháp bao gồm tất cả các từ của một câu, do đó $P(S|T) = 1$:

$$P(T, S) = P(T)P(S|T) = P(T)$$

Chúng ta có thể tính xác suất của mỗi cây trong hình 5.2 bằng cách nhân các xác suất của mỗi quy tắc được sử dụng trong một dẫn xuất với nhau. Ví dụ, xác xuất của cây bên trái của hình 5.2 (gọi là T_{left}) và cây bên phải (T_{right}) có thể được tính như sau:

$$P(T_{left}) = 0.05 * 0.20 * 0.20 * 0.20 * 0.75 * 0.30 * 0.60 * 0.10 * 0.40 = 2.2 \times 10^{-6}$$
$$P(T_{right}) = 0.05 * 0.10 * 0.20 * 0.15 * 0.75 * 0.75 * 0.20 * 0.60 * 0.10 * 0.40 = 6.1 \times 10^{-7}$$

Chúng ta có thể thấy rằng cây trái trong hình 5.2 có xác suất cao hơn so với cây phải. Do đó, phân tích bên trái sẽ được lựa chọn.

Xem xét tất cả các cây phân tích cú pháp có khả năng cho ra từ S. Chuỗi các từ S được gọi là **sinh bởi (yeild)** của cây phân tích cú pháp bất kỳ trên S. Thuật toán chọn cây phân tích mà có xác suất cao nhất cho S:

$$\begin{aligned}
\hat{T} &= \underset{\text{S=yield(T)}}{\text{argmax}}\ P(T|S) \\
&= \underset{S=\text{yield(T)}}{\text{argmax}}\ \frac{P(T, S)}{P(S)} \\
&= \underset{S=\text{yield(T)}}{\text{argmax}}\ P(T, S) \\
&= \underset{S=\text{yield(T)}}{\text{argmax}}\ P(T)
\end{aligned}$$

5.1.2 PCFG for Language Modeling

Đặc tính thứ hai của một PCFG là nó chỉ định một xác suất cho một chuỗi các từ cấu thành một câu. Điều này là quan trọng trong **mô hình ngôn ngữ (language modeling)**, dùng cho nhận dạng giọng nói, dịch máy, sửa chính tả, ... Xác xuất của một câu nhập nhằng là tổng của các xác suất của tất cả các cây phân tích cú pháp cho câu đó:

$$\begin{aligned}
P(S) &= \sum_{S=\text{yield(T)}} P(T, S) \\
&= \sum_{S=\text{yield(T)}} P(T)
\end{aligned}$$

Một đặc trưng bổ sung của PCFG là khả năng gán một xác suất cho chuỗi con của một câu. Ví dụ, giả sử chúng ta muốn biết xác suất của từ tiếp theo w_i trong câu cho bởi các từ $w_1, \ldots, w_{i-1}$. Công thức chung sẽ có dạng:

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) = \frac{P(w_1, w_2, \ldots, w_i)}{P(w_1, w_2, \ldots, w_{i-1})}$$

5.2 Probabilistic CKY Parsing of PCFG

Bài toán phân tích cú pháp của PCFG là chọn cây có khả năng nhất $\hat{T}$ cho một câu S:

$$\hat{T(S)} = \underset{S=\text{yield(T)}}{\text{argmax}}\ P(T)$$

Hầu hết phân tích xác suất hiện đại dựa trên thuật toán **probabilistic CYK**, được giới thiệu lần đầu bởi Ney (1991). Với thuật toán **probabilistic CYK**, chúng ta thêm một chiều thứ 3 nữa, ma trận sẽ có kích cỡ $(n+1) \times (n+1) \times V$. Thuật toán propabilistic CYK được biểu diễn như hình 5.3:

function PROBABILISTIC-CKY(*words*,*grammar*) **returns** most probable parse and its probability

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{ A \mid A \rightarrow words[j] \in grammar \}$
    $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{ A \mid A \rightarrow BC \in grammar,$
        and $table[i,k,B] > 0$ **and** $table[k,j,C] > 0 \}$
        if $(table[i,j,A] < P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C])$ **then**
          $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
          $back[i,j,A] \leftarrow \{k, B, C\}$
  return BUILD_TREE($back$[1, LENGTH(*words*), *S*]), $table$[1, LENGTH(*words*), *S*]

Hình 5.3: Thuật toán probabilistic CYK

Lưu ý rằng là cũng giống như thuật toán CYK, thuật toán probabilistic CYK cũng yêu cầu văn phạm đầu vào ở dạng chuẩn Chomsky, chúng ta cần đưa văn phạm về dạng chuẩn Chomsky trước khi thực hiện thuật toán.

Câu hỏi đặt ra là các xác suất cho các quy tắc đến từ đâu? Có 2 cách để học xác suất cho các quy tắc của văn phạm. Cách đơn giản nhất là sử dụng một TreeBank, một corpus đã phân tích cú pháp câu, với mỗi luật $\alpha \rightarrow \beta$ có thể tính như sau:

$$P(\alpha \rightarrow \beta|\alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_\gamma Count(\alpha \rightarrow \gamma)} = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

trong đó:

- $Count(\alpha \to \beta)$: Số lần quy tắc $\alpha \to \beta$ xuất hiện trong TreeBank.
- $Count(\alpha)$: Số lần xuất hiện của các quy tắc có vế trái là α trong TreeBank.

Nếu chúng ta không có một TreeBank, chúng ta có thể sử dụng một biến thể của thuật toán EM là **thuật toán inside-outside**, được giới thiệu bởi Baker (1979) để ước lượng trọng số các xác suất cho các quy tắc của văn phạm.

Ví dụ 21.

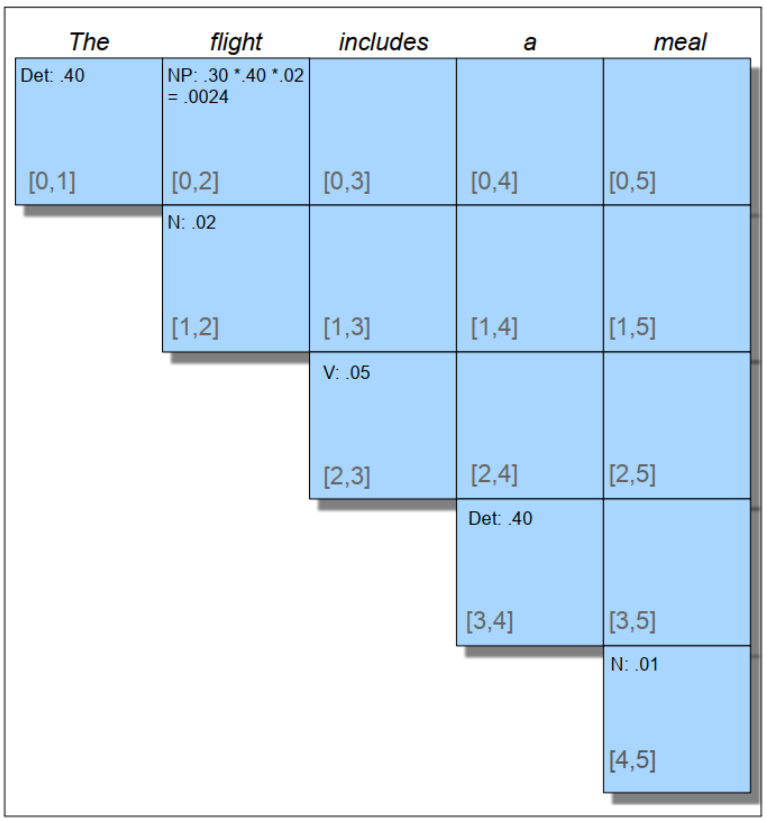
Thực hiện thuật toán probabilistic CYK với vặn phạm ở dạng chuẩn Chomsky như hình 5.4 cho câu:

The flight includes a meal

S	$\to$	$NP\ VP$	.80	Det	$\to$	*the*	.40
NP	$\to$	$Det\ N$	.30	Det	$\to$	*a*	.40
VP	$\to$	$V\ NP$	.20	N	$\to$	*meal*	.01
V	$\to$	*includes*	.05	N	$\to$	*flight*	.02

Hình 5.4: Văn phạm ví dụ để thực hiện thuật toán probabilistic CYK

Bảng CYK của thuật toán những bước đầu tiên được minh hoạ như hình 5.5:



Hình 5.5: Bảng cho ví dụ minh hoạ thuật toán probabilistic CYK

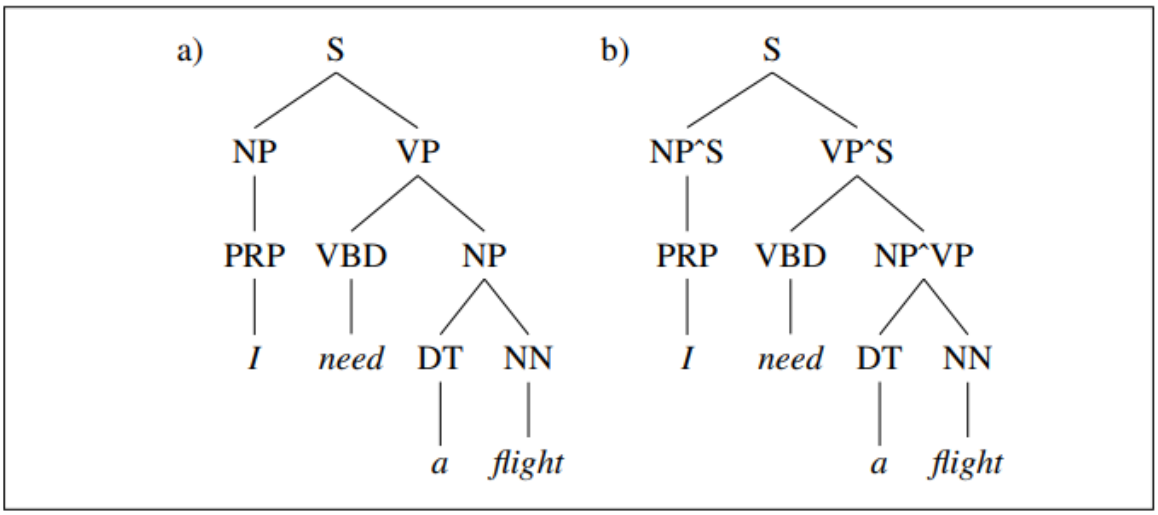
5.3 Các vấn đề với PCFG

PCFG đang có những nhược điểm sau:

- **Độc lập vị trí:** Xác suất một cây con không phụ thuộc vào vị trí của các từ của cây con đó ở trong câu.
- **Độc lập ngữ cảnh:** Xác suất một cây con không phụ thuộc vào các từ ngoài cây con đó.
- **Độc lập tổ tiên:** Xác suất một cây con không phụ thuộc vào các nút ngoài cây con đó.

Văn phạm phi ngữ cảnh xác suất từ vựng thiếu sự nhạy bén đối với các thông tin từ vựng. Tuy nhiên, trong ngôn ngữ, ý nghĩa và cấu trúc của câu lại phụ thuộc nhiều vào ngữ cảnh của câu đó. Văn phạm phi ngữ cảnh xác suất từ vựng cũng thiếu sự nhạy bén trong việc xác định cấu trúc ngữ pháp. Ví dụ một luật $NP \rightarrow$ N N, ta nhận thấy trong ngôn ngữ có thể có những từ không thể đi cùng với nhau, nhưng trong quá trình tính toán cấu trúc có sự phi lý đó lại có xác suất lớn, vì vậy xảy ra hiện tượng nhập nhằng. Một trong số những cải tiến là sử dụng đánh dấu **parent annotation**. Trong đó, đánh dấu nút cha của một nút như hình 5.6. Ví dụ như nút NP có nút cha là S thì sẽ được đánh dấu là NP^ S.

Khi đó, chúng ta có thể tăng tính phụ thuộc của mô hình hơn. Giúp cải thiện mô hình. Kỹ thuật này cũng làm tăng cấu trúc của văn phạm, các mô hình hiện đại sau này tự động tìm kiếm các phương pháp tác (split) tối ưu. Thuật toán **split and merge** của Petrov et al. (2006) là một ví dụ. Tính đến thời điểm viết bài này, hiệu suất của Petrov et al. (2006) thuật toán là thuật toán tốt nhất trong số các thuật toán phân tích cú pháp đã biết trên Penn Treebank. .



Hình 5.6: Kỹ thuật parent anotation

5.4 Văn phạm phi ngữ cảnh xác suất từ vựng (Probabilistic Lexicalized CFGs)

Phần trước cho thấy rằng thuật toán probabilistic CYK đơn giản để phân tích PCFG thô có thể đạt được độ chính xác phân tích cú pháp rất cao nếu các ký hiệu quy tắc ngữ pháp được thiết kế lại bằng cách **split and merge** tự động.

Trong phần này, chúng ta thảo luận về một nhóm các mô hình thay thế trong đó thay vì sửa đổi các quy tắc ngữ pháp, chúng ta sửa đổi mô hình xác suất của trình phân tích cú pháp. Họ các trình phân tích cú pháp này phổ biến gồm **phân tích cú pháp Collins** (Collins, 1999) và **phân tích cú pháp Charniak** (Charniak, 1997). Cả hai đều được sử dụng rộng rãi trong quá trình xử lý ngôn ngữ tự nhiên.

5.4.1 Phân tích cú pháp Collins

Ở phần trước, chúng ta đã thảo luận một số nhược điểm của mô hình phân tích cú pháp xác suất. Collins đã cải tiến bằng cách đưa ra mô hình xác xuất dựa vào từ vựng. Trong mô hình, Collins đưa vào văn phạm phi ngữ cảnh xác suất cấu trúc head.

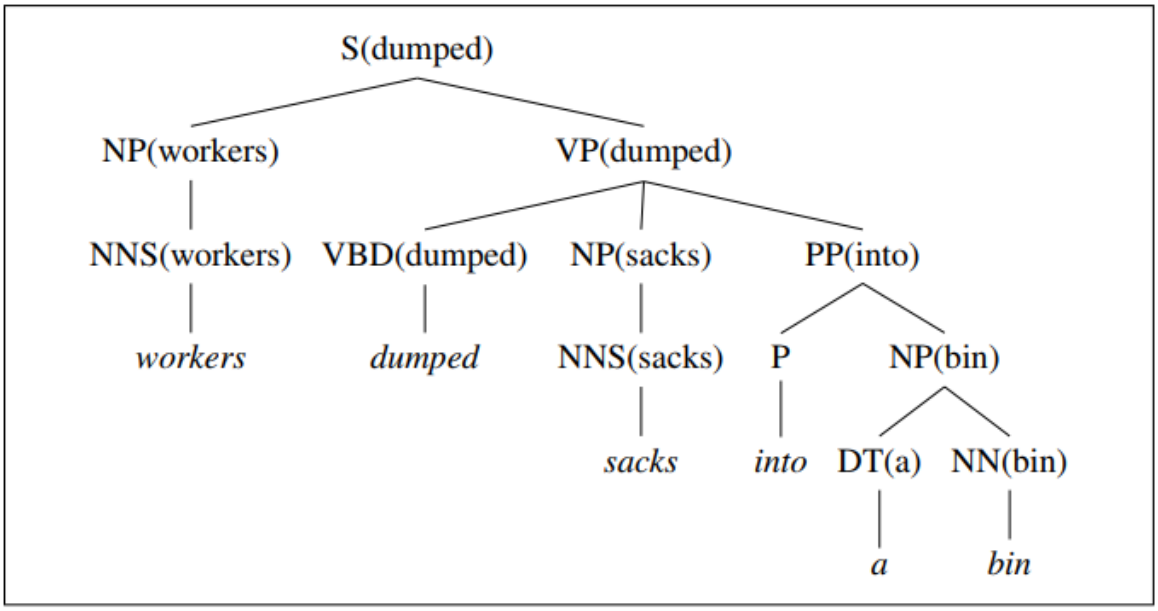
Cấu trúc head

Giả sử chúng ta có cụm từ như sau: "*Một mái tóc đẹp*". Đối với cụm này, ta có thể phân chia thành các thành phần như sau:

- *Một*: thành phần phụ trước.
- *mái*: thành phần trung tâm.
- *tóc*: thành phần phụ sau.
- *đẹp*: thành phần phụ sau.

Từ *mái* ở đây là thành phần trung tâm của cụm từ, ta thử bỏ thành phần trung tâm này đi thì cụm từ sẽ là *một tóc đẹp*. Đối với cụm mới tạo ra này, ta thấy rõ ràng cụm này vô nghĩa, bởi *tóc* là danh từ không đếm được không thể đi với từ chỉ số lượng *một*. Tuy nhiên, chú ý rằng ta có thể bỏ từ *một* nhưng cụm vẫn có nghĩa (*mái tóc đẹp*). Như vậy từ *mái* là thành phần trung tâm của cụm từ hay câu (head).

Như vậy trong văn phạm này, mỗi ký tự không kết thúc được gán thêm thành phần trung tâm của cụm từ hay câu đó. Việc xác định head của một ký tự không kết thúc, ta phải dựa vào head của các ký tự con.



Hình 5.7: Cây phân tích cú pháp của xâu *workers dumped sacks into a bin* có thêm thông tin từ vựng

Cấu trúc Head tag

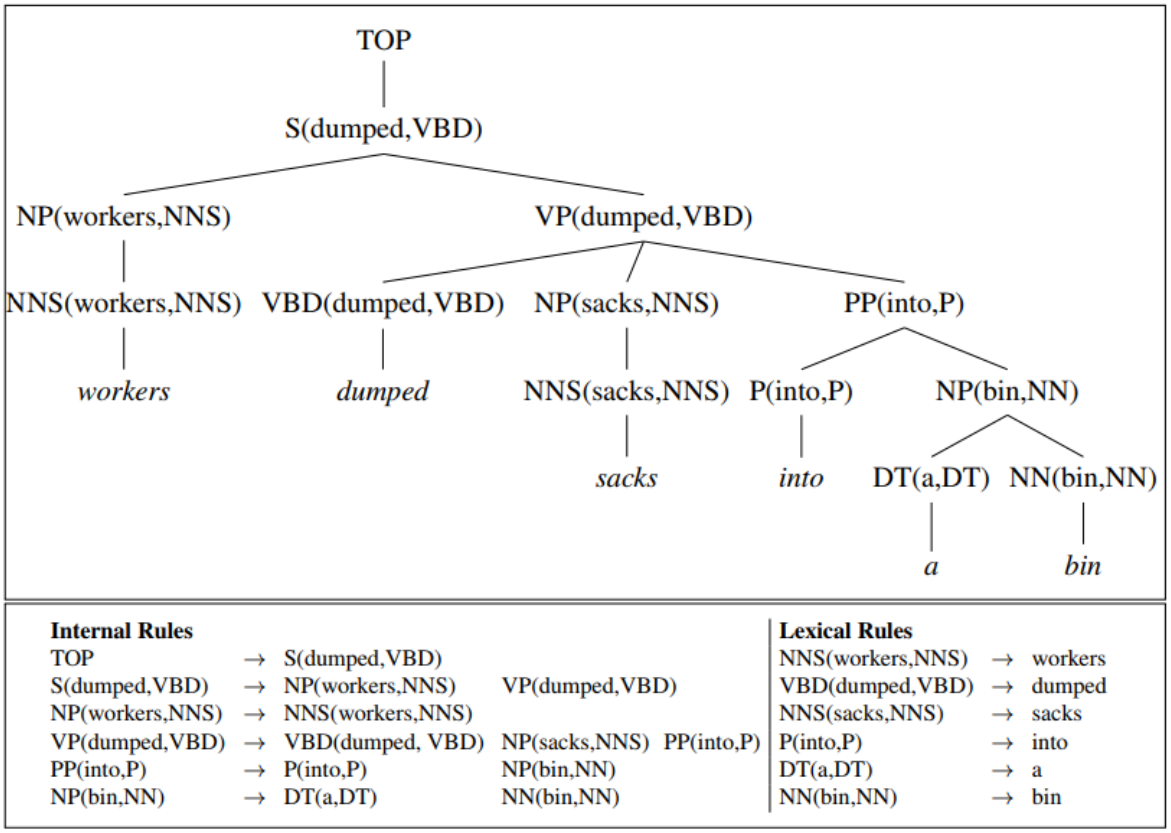
Chúng ta có thể mở rộng thêm cấu trúc head thành cấu trúc **head tag**, trong đó thêm Part-of-speech tags của headwords bằng các ký hiệu không kết thúc. Do đó mỗi quy tắc của văn phạm sẽ bao gồm cả headword và head tag của mỗi thành phần con :

$$VP(dumped, VBD) \rightarrow VBD(dumped, VBD)NP(sacks, NNS)PP(into, P)$$

Chúng ta cũng có thêm 2 khái niệm sau đây:

- **lexicon rules**: quy tắc của một ký hiệu không kết thúc tới một từ.
- **internal rules**: các quy tắc khác.

Hình 5.8 bên trên minh hoạ cấu trúc cây phân tích cú pháp dựa trên head tag cho một câu ví dụ, còn bên dưới biểu diễn lexcicon rules và internal rules.



Internal Rules				Lexical Rules		
TOP	→	S(dumped,VBD)		NNS(workers,NNS)	→	workers
S(dumped,VBD)	→	NP(workers,NNS)	VP(dumped,VBD)	VBD(dumped,VBD)	→	dumped
NP(workers,NNS)	→	NNS(workers,NNS)		NNS(sacks,NNS)	→	sacks
VP(dumped,VBD)	→	VBD(dumped, VBD)	NP(sacks,NNS) PP(into,P)	P(into,P)	→	into
PP(into,P)	→	P(into,P)	NP(bin,NN)	DT(a,DT)	→	a
NP(bin,NN)	→	DT(a,DT)	NN(bin,NN)	NN(bin,NN)	→	bin

Hình 5.8: Cây phân tích cú pháp của xâu *workers dumped sacks into a bin* có thêm head tag

Mô hình cơ sở

Đây là mô hình cơ bản nhất trong các mô hình Collins đưa ra. Ta có một quy tắc giống như sau:

$$LSH \rightarrow L_n L_{n-1} \dots L_1 H R_1 \dots R_{n-1} R_n$$

trong đó:

- H: là ký tự đánh dấu cho thành phần head.
- $L_1, \dots, L_n$: bên trái của thành phần head.
- $R_1, \dots, R_n$: bên phải của thành phần head.

Ngoài ra ở vế trái và vế phải của chuỗi ta thêm 2 ký tự STOP nhằm đánh dấu kết thúc luật. Ví dụ:

$$P(VP(dumped, VBD) \rightarrow \text{STOP } VBD(dumped, VBD) NP(sacks, NNS) PP(into, P) \text{ STOP})$$

Chúng ta tận dụng 3 loại xác suất: P_H cho tạo head, P_L cho phụ thuộc trái và P_R cho phụ thuộc phải. Nếu H là head với head word hw và head tà ht, lw/lt và rw/rt biểu diễn tương ứng biểu diễn word/tag của bên trái và phải. P là gốc, thì xác suất toàn bộ quy tắc có thể diễn tả như sau:

1. Sinh nhãn head của cây $H(hw, ht)$ với xác suất:

$$P_H(H(hw, ht)|P, hw, ht)$$

2. Tính toán xác suất với vế trái của head:

$$\prod_{i=1}^{n+1} P_L(L_i(lw_i, lt_i)|P, H, hw, ht)$$

trong đó $L_{n+1}(lw_{n+1}, lt_{n+1}) = STOP$, ký tự STOP được thêm vào bảng ký hiệu không kết thúc, và mô hình dừng việc sinh kế tiếp xác suất khi gặp ký tự STOP.

3. Tính toán xác suất với vế phải của head:

$$\prod_{i=1}^{n+1} P_R(R_i(rw_i, rt_i)|P, H, hw, ht)$$

trong đó $R_{n+1}(rw_{n+1}, rt_{n+1}) = STOP$, ký tự STOP được thêm vào bảng ký hiệu không kết thúc, và mô hình dừng việc sinh kế tiếp xác suất khi gặp ký tự STOP.

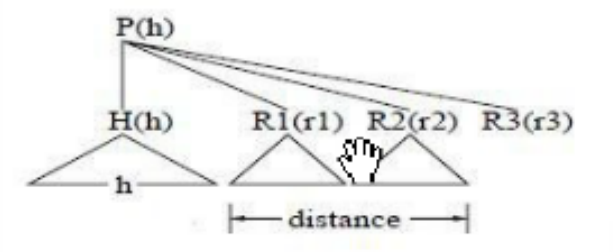
5.4.2 Nâng cao: Các bổ sung của trình phân tích Collins

Mô hình của Collins thực tế phức tạp hơn phần bên trên, ở mở rộng đầu tiên, Collins thêm một đặc trưng **khoảng cách (distance)**:

$$P_L(L_i(lw_i, lt_i)|P, H, hw, ht, distance_L(i-1))$$

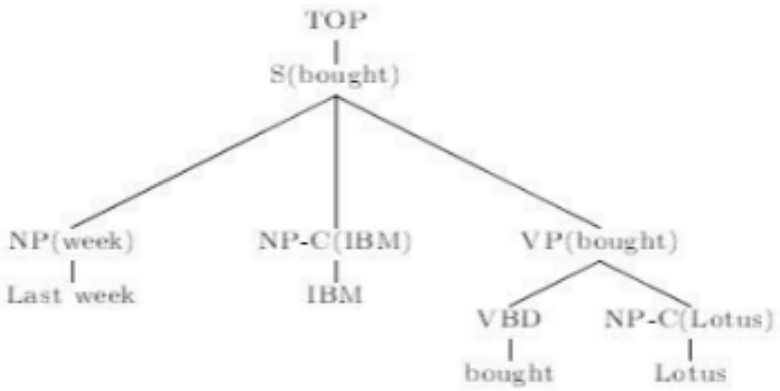
$$P_R(R_i(rw_i, rt_i)|P, H, hw, ht, distance_R(i-1))$$

Khoảng cách giữa các từ bổ nghĩa cho thành phần trung tâm cũng rất quan trọng. Trong một số trường hợp đặc biệt, đó là dấu hiệu để nhận biết các cấu trúc phân nhánh (độ phụ thuộc của các từ liền kề nhau). Thông tin về khoảng cách được đưa vào mô hình nhằm nâng cao sự phụ thuộc giữa các từ bổ trợ.



Hình 5.9: Miêu tả độ đo khoảng cách trong câu

Trong mở rộng thứ 2, Collins phân biệt định ngữ, bổ ngữ, subcategorization. Ví dụ, ta thử lấy một câu tiếng Anh như hình 5.10.



Hình 5.10: Ví dụ về mở rộng thứ 2 của Collins

Trong mô hình này, để đánh dấu đâu là bổ ngữ thì ta thêm hậu tố -C vào mỗi ký tự không kết thúc. Tại sao việc xác định bổ ngữ lại quan trọng? Trong ví dụ trên, "IBM" được xác định là chủ ngữ của câu, còn "Last week" là thành phần bổ nghĩa, mặc dù cả hai cùng được gán nhãn NP, ngoài ra "Lotus" cũng là một danh từ được gán nhãn NP, ở đây "Last week" là một thành phần bổ nghĩa cho sự kiện, còn "Lotus" lại bổ nghĩa cho động từ "bought". Vì vậy "Lotus" là bổ ngữ (complement), "Last week" là định ngữ (adjunction). Sự khác biệt về chức năng của hai nhãn NP "Last week" và "IBM" là không phù hợp trong cây cú pháp, vì hai nhãn NP đặt cùng vị trí. Ngoài ra, chúng ta chỉ có thể đưa những thông tin này vào trong quá trình phân tích. Việc đưa thông tin này vào làm tăng khả năng xác định cây cú pháp đúng, giảm sự phập nhằng của văn phạm.

Việc xác định bổ ngữ rất phức tập đối với việc sử dụng xác suất. Thông tin về từ vựng là cần thiết. Ngoài ra, độ ưu tiên của các subcategoziation cũng cần được để ý đến. Trong quá trình phân tích cú pháp, việc phân biệt bổ ngữ và định ngữ cũng làm tăng độ chính xác.

Mô hình trên có thể được huấn luyện với tập dữ liệu nâng cao bao gồm các ký tự không kết thúc và quá trình học các thông tin từ vựng có thể chỉ ra được sự phân biệt bổ ngữ và định ngữ. Tuy nhiên, nó vẫn còn gặp phải những ước lượng độc lập tồi. Để giải quyết vấn đề này, quá trình xử lý mới cải tiến bằng cách thêm xác suất phụ thuộc subcategorization frame trái và phải. Trong phạm vi báo cáo này, chỉ mang tính chất giới thiệu mà không đi sâu vào cách thức hoạt động của thuật toán.

5.5 Đánh giá độ chính xác

Độ chính xác của parser được đo qua việc tính xem có bao nhiêu thành phần ngữ pháp trong cây giống với cây chuẩn, gọi là **gold-standard reference parses**. Chúng ta đánh giá qua các tiêu chuẩn như **độ chính xác (precision)** và **độ phủ (recall)**

và **F-measure**

$$\text{Precision} = \frac{\text{Trường hợp hệ gán đúng}}{\text{Tổng số trường hợp hệ gán}}$$
$$\text{Recall} = \frac{\text{Số trường hợp hệ gán đúng}}{\text{tổng số trường hợp đúng}}$$
$$\text{F-measure} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

Các công thức này tương tự như đã giới thiệu ở các phần trước.

Chương 6

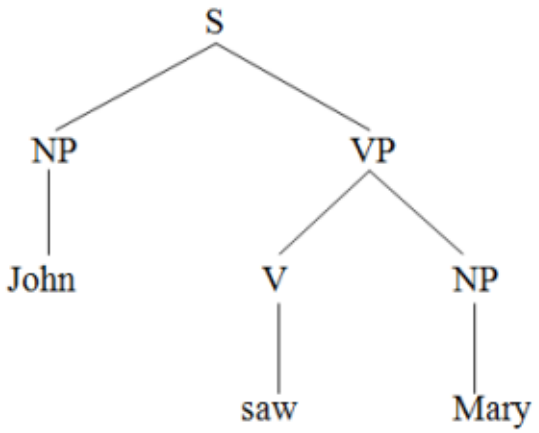
Dependency Parsing

6.1 Cú pháp phụ thuộc

6.1.1 Định nghĩa cú pháp phụ thuộc

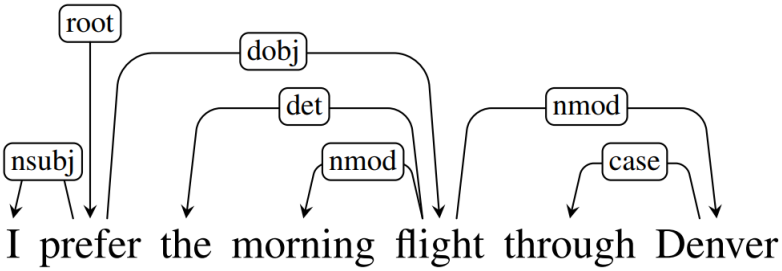
Ở trong các chương trước, chúng ta đã tìm hiểu về **cú pháp thành phần**. Trong chương này chúng ta sẽ tìm hiểu về **cú pháp phụ thuộc** và các phương pháp phân tích cú pháp phụ thuộc.

Cú pháp thành phần là cấu trúc câu theo thứ tự bậc các thành phần của câu, sử dụng cấu trúc cụm từ. Ví dụ như hình 6.1.



Hình 6.1: Cú pháp thành phần

Cú pháp phụ thuộc là cấu trúc biểu diễn quan hệ giữa các từ trong câu dựa trên ngữ nghĩa. Ví dụ như hình 6.2.



Hình 6.2: Cú pháp phụ thuộc

Như hình 6.2, ta có thể thấy quan hệ phụ thuộc giữa hai từ vựng là quan hệ nhị phân không đối xứng. Các quan hệ phụ thuộc này được đặt tên để làm rõ (ví dụ case, nmode, . . .). Bảng 6.3 cung cấp một số quan hệ phụ thuộc.

Clausal Argument Relations	**Description**
NSUBJ	Nominal subject
DOBJ	Direct object
IOBJ	Indirect object
CCOMP	Clausal complement
XCOMP	Open clausal complement
Nominal Modifier Relations	**Description**
NMOD	Nominal modifier
AMOD	Adjectival modifier
NUMMOD	Numeric modifier
APPOS	Appositional modifier
DET	Determiner
CASE	Prepositions, postpositions and other case markers
Other Notable Relations	**Description**
CONJ	Conjunct
CC	Coordinating conjunction

Hình 6.3: Tập các quan hệ phụ thuộc

Chúng cũng có thể định nghĩa một cách hình thức như sau: Cú pháp phụ thuộc của một câu cho trước là một đồ thị có hướng với gốc *root* là một *đỉnh giả*, thường được chèn vào phía bên trái câu, các đỉnh còn lại là các từ của câu. Cấu trúc phụ thuộc được xác định bởi mối quan hệ một từ trung tâm (head) và từ phụ thuộc (dependent) của nó. Theo quy ước phổ biến, thì từ nằm ở gốc của của mũi tên là từ trung tâm, từ nằm ở đầu mũi tên là từ phụ thuộc. Cấu trúc phụ thuộc thường đơn giản hơn cấu trúc thành phần, dễ dàng hơn cho cả người và máy khi học một cấu trúc ngữ pháp. Hơn nữa, cấu trúc phụ thuộc thích hợp hơn với các ngôn ngữ có trật tự từ tự do, như tiếng Séc hay Thổ Nhĩ Kì. Tuy nhiên, không phải vì thế mà các ngôn ngữ có trật tự từ tự do thì luôn dùng cú pháp phụ thuộc và ngược lại.

6.1.2 Bài toán phân tích cú pháp phụ thuộc

Phân tích cú pháp phụ thuộc đưa ra mô tả về quan hệ và vai trò ngữ pháp của các từ trong câu, đồng thời đưa ra hình thái của câu. Bài toán phân tích cú pháp phụ thuộc là tìm đồ thị phụ thuộc cho một câu. Đầu vào của bài toán là câu đã được tách từ và gán nhãn từ loại, trong đó mỗi từ có đặc điểm hình thái xác định. Mục tiêu của bài toán là tìm ra phương pháp sinh đồ thị phụ thuộc chính xác nhất cho một câu đầu vào, nghĩa là làm cực đại số cung chính xác trong đồ thị và số nhãn gán đúng cho các cung. Ta có:

- Đầu vào:
  - Câu $x = w_1, w_2, \ldots, w_n$ đã được tiền xử lý, tách từ và gán nhãn từ loại.
  - Kho ngữ liệu gồm các câu đã được gán nhãn phụ thuộc (phục vụ cho quá trình huấn luyện trong các thuật toán).
- Đầu ra: Đồ thị phụ thuộc của câu x

6.1.3 Biểu diễn cú pháp phụ thuộc

Cho một câu x gồm n từ $w_1, w_2, \ldots, w_n$, khi đó ta sẽ ký hiệu x như sau:

$$X = (w_1, w_2, \ldots, w_n)$$

Trong phân tích cú pháp phụ thuộc, cú pháp phụ thuộc của một câu được biểu diễn bởi một đồ thị có hướng, các đỉnh trong đồ thị tương ứng với các từ của một câu, các cung trong đồ thị được gán nhãn, các nhãn của các cung tương ứng với loại phụ thuộc giữa hai từ.

Định nghĩa 14 (Đồ thị phụ thuộc)**.**

Cho một tập $L = \{r_1, \ldots, r_{|L|}\}$ các loại phụ thuộc (các nhãn cung), đồ thị phụ thuộc của một câu $x = (w_1, \ldots, w_n)$ là một đồ thị có hướng được gán nhãn $G = (V, E, R)$, trong đó:

1. $V = Z_{n+1}$.
2. $E \subseteq V \times V$.
3. R là một hàm xác định nhãn cung.

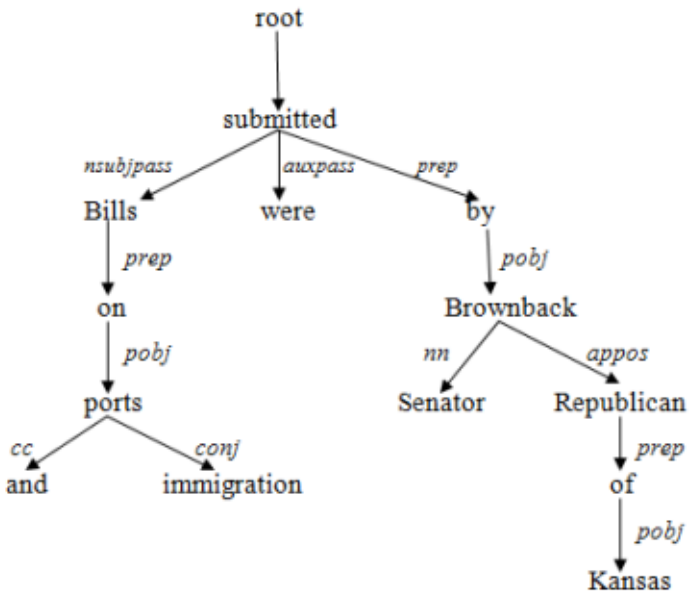
Tập đỉnh V là một tập $Z_{n+1} = \{0, 1, \ldots, n\}, n \in Z^+$, là tập số nguyên không âm tăng dần tới n. Điều này có nghĩa là tất cả các từ trong câu là một đỉnh $(1 \leq i \leq n)$ và có một đỉnh đặc biệt 0, không tương ứng với bất kỳ từ nào của câu và luôn là gốc của đồ thị phụ thuộc. Sử dụng V^+ là một tập tất cả các đỉnh tương ứng với các từ của câu cụ thể $x = (w_1, \ldots, w_n)$. Thoả mãn $|V^+| = n$ và $|V| = n + 1$.

Tập hợp các cung E là một cặp (i, j), trong đó i, j là các đỉnh, kí hiệu $i \to j$ có nghĩa là một cung nối giữa đỉnh i và đỉnh j, khi đó ta có $(i, j) \in E$. Kí hiệu $i \to^* j$ khi và chỉ khi $i = j$ hoặc có một cung nối từ đỉnh i đến đến j.

Hàm R chỉ một loại phụ thuộc $r \in L$ tới mỗi cung $e \in E$. Kí hiệu $i \to^r j$ có nghĩa là có một cung có nhãn r kết nối đỉnh i tới đỉnh j (ví dụ $i \to j$ và $R((i, j)) = r$).

Từ w_0 là từ được thêm vào ngay đầu của câu và không bổ nghĩa cho bất cứ từ nào trong câu, một phụ tố, tiền tố hoặc bất cứ hình vị nào trong câu. Quy ước 0 (tương ứng với w_0) luôn là gốc của đồ thị phụ thuộc của câu cần phân tích.

Ví dụ: Đồ thị phụ thuộc của câu *Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas* như trong hình 6.4.



Hình 6.4: Đồ thị phụ thuộc cho một câu ví dụ

Trong ví dụ trên, tập $L = \{nsubjpass; auxpass; prep; pobj; nn; cc; conj; appos\}$ là các quan hệ phụ thuộc của các từ trong câu, và cũng là các nhãn cung của đồ thị phụ thuộc. Các từ ở gốc mũi tên là các từ trung tâm, các từ ở đầu mũi tên là các từ phụ thuộc. Với một cung *submitted* $\to$ *Bills*, thì *submitted* là từ trung tâm, *Bills* là từ phụ thuộc và quan hệ phụ thuộc giữa hai từ này được biểu thị bằng nhãn phụ thuộc *nsubjpass*. Ngoài ra, chúng ta cũng có thể minh hoạ đồ thị phụ thuộc như hình 6.2.

Định nghĩa 15 (Đồ thị phụ thuộc xây dựng đúng)**.**

Một đồ thị phụ thuộc G xây dựng đúng nếu và chỉ nếu:

1. Đỉnh 0 là gốc *(ROOT)*.

2. G liên thông yếu *(CONECTEDNESS)*.

3. Mọi đỉnh đều có nhiều nhất một từ trung tâm, tức là nếu $i \rightarrow j$ thì với một từ bất kì khác trong câu, không tồn tại k thoả mãn $k \neq i$ và $k \rightarrow j$ *(SINGLE-HEAD)*.

4. Các đồ thị G là không có chu trình, tức là có $i \rightarrow j$ thì không tồn tại $j \rightarrow^* i$ *(ACYLICITY)*.

Ngoài các tính chất trên của một đồ thị phụ thuộc, hầu hết các đồ thị còn thoả mãn điều kiện xạ ảnh. Các đồ thị là xạ ảnh, nếu như có $i \rightarrow j$ thì $i \rightarrow^* k, \forall k$ thoả mãn $i \leq k \leq j$ hoặc $j \leq k \leq i$ *(PROJECTIVETY)*. Tuy nhiên, không phải tất cả các câu đều thoả mãn điều kiện này nên một số thuật toán được phát triển để giải quyết vấn đề không xạ ảnh trong phân tích cú pháp phụ thuộc.

Nhờ cách mô hình hoá như trên, cú pháp phụ thuộc biểu diễn được những ngôn ngữ có trật tự từ tự do, đây là điều mà cú pháp cấu trúc cụm (vốn phù hợp với những ngôn ngữ có nhiều quy tắc chặt chẽ trong cấu thành câu) không làm được. Tuy vậy, không có nghĩa là phân tích ngôn ngữ có trật từ từ xác định thì chỉ dùng cấu trúc cụm hay phân tích ngôn ngữ có trật từ từ tự do thì chỉ dùng cấu trúc phụ thuộc.

6.2 Các thuật toán phân tích cú pháp phụ thuộc

Có hai phương pháp phân tích cú pháp phụ thuộc cơ bản:

- **Phân tích cú pháp phụ thuộc dựa vào bước chuyển**: Phân tích cú pháp phụ thuộc thông qua các bước chuyển từ trạng thái phân tích này tới trạng thái phân tích khác. Các tham số trong mô hình thường được huấn luyện sử dụng kỹ thuật phân lớp chuẩn để dự đoán bước chuyển tiếp theo từ một tập hợp các bước chuyển trước đó. Sử dụng suy luận cục bộ, hệ thống bắt đầu từ một trạng thái ban đầu cố định và xây dựng các đồ thị bằng hàm điểm chuyển đổi cao nhất tại mỗi trạng thái cho đến khi một điều kiện được đáp ứng.

- **Phân tích cú pháp phụ thuộc dựa vào đồ thị**: Phân tích cú pháp phụ thuộc thông qua tham số hoá mô hình phụ thuộc dựa vào các đồ thị con và huấn luyện các tham số trên toàn bộ đồ thị. Sử dụng ngay suy luận toàn cục trong hệ thống để tìm những đồ thị có trọng số cao nhất trong số các cách thiết lập tất cả các đồ thị.

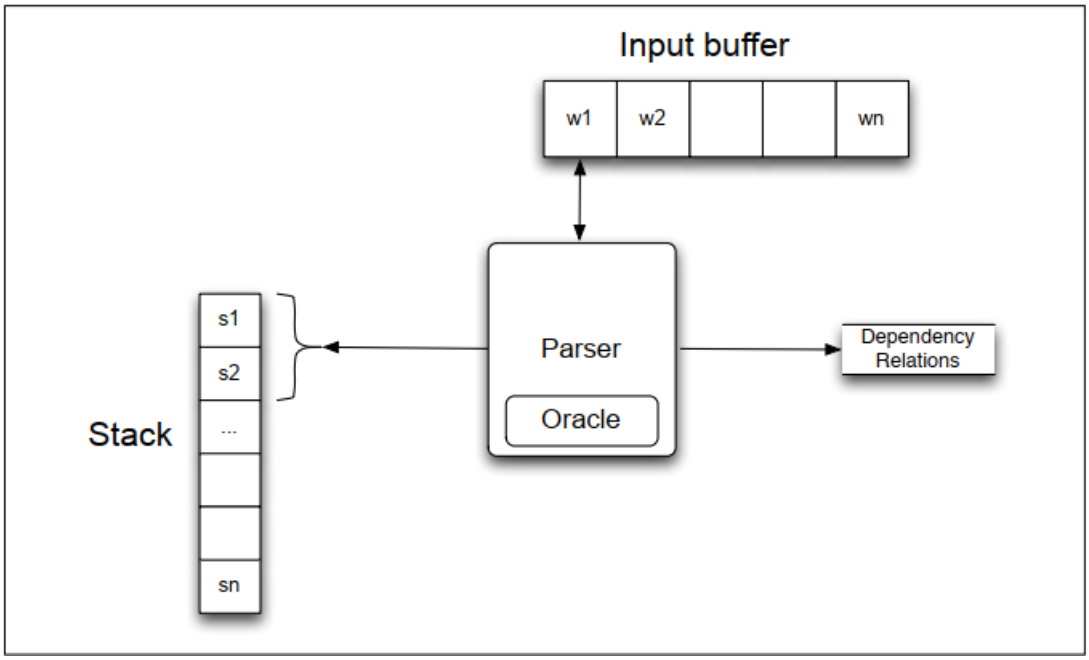
Cả hai phương pháp đều đưa ra kết quả phân tích với độ chính xác tương đương nhau, như ví dụ trong hình 6.5. Ngoài ra thì hiện nay cũng có nhiều phương pháp mới cải tiến hơn, tuy nhiên, trong phần này, chúng ta tập trung vào 2 phương pháp trên.

Ngôn ngữ	Graph-based (McDonald cùng cộng sự)	Transition-based (Nivre cùng cộng sự)	Số câu tập huấn luyện	Số nhãn phụ thuộc
Arabic	66.91%	66.71%	1500	27
Bulgarian	87.57%	87.41%	14400	18
Chinese	85.90%	86.92%	57000	82
Czech	80.18%	78.42%	72700	78
Danish	84.79%	84.77%	5200	52
Dutch	79.19%	78.59%	13300	26
German	87.34%	85.82%	39200	46
Japanese	90.71%	91.65%	17000	7
Portuguese	86.82%	87.60%	9100	55
Slovene	73.44%	70.30%	1500	25
Spanish	82.25%	81.29%	3300	21
Swedish	82.55%	84.58%	11000	56
Turkish	63.19%	65.68%	5000	25

Hình 6.5: Kết quả phân tích cú pháp phụ thuộc của 2 mô hình cho hệ thống CoNNL-X (Buchholz và Marsi 2006)

6.2.1 Phân tích cú pháp phụ thuộc dựa trên bước chuyển

Thuật toán Shift-Reduce (Phân tích cú pháp phụ thuộc dựa vào các bước chuyển) là một thuật toán cơ bản và có hiệu quả cao với rất nhiều các ngôn ngữ khác nhau. Thuật toán này phân tích câu đầu vào từ bên trái sang bên phải sử dụng hai cấu trúc dữ liệu chính: một vùng đệm lưu trữ những dữ liệu đầu vào còn lại và một ngăn xếp lưu trữ những dữ liệu đã được xử lý một phần.



Hình 6.6: Trình phân tích cú pháp phụ thuộc dựa trên bước chuyển

Một yếu tố quan trọng dựa trên phân tích cú pháp phụ thuộc dựa vào bước chuyển là khái niệm **cấu hình (configuration)**, bao gồm ngăn xếp, bộ đệm đầu vào của từ và một tập các quan hệ biểu thị cây phụ thuộc. Với khung này, quá trình phân tích cú pháp bao gồm một chuỗi các chuyển đổi thông qua không gian các cấu hình có thể. Mục tiêu của quá trình này là tìm ra một cấu hình cuối cùng trong đó tất cả các từ đã được tính toán và một cây phụ thuộc thích hợp đã được tổng hợp.

Một cách hình thức, chúng ta có thể mô tả như sau: Cho tập $L = (r_0, \ldots, r_m)$ là tập các nhãn phụ thuộc và một câu $x = (w_1, \ldots, w_n)$, một cấu hình phân tích cú pháp phụ thuộc là một bộ ba: $c = (\sigma, \beta, A)$. Trong đó, c chứa một ngăn xếp σ, một vùng đệm β và một tập các cung phụ thuộc A.
Cấu hình ban đầu của một câu $s = (w_1, \ldots, w_n)$ là:

- $\sigma = ROOT$.
- $\beta = (w_1, \ldots, w_n)$.
- $A = \emptyset$.

Về cơ bản, chúng ta có mô hình chung cho phân tích cú pháp phụ thuộc dựa trên bước chuyển như hình 6.7. Đây được gọi là **tiêu chuẩn arc (arc standard)**.

```
function DEPENDENCYPARSE(words) returns dependency tree

  state ← {[root], [words], [] }  ; initial configuration
  while state not final
    t ← ORACLE(state)         ; choose a transition operator to apply
    state ← APPLY(t, state)   ; apply it, creating a new state
  return state
```

Hình 6.7: Thuật toán tổng quát phân tích cú pháp phụ thuộc dựa trên bước chuyển

Ở thuật toán trên, chúng ta thấy có bước chọn toán tử chuyển để áp dụng. Thuật toán **arc-eager** định nghĩa ra bốn loại hàm chuyển: LEFT-ARC, RIGHT-ARC, SHIFT, REDUCE.

Sử dụng ký hiệu $v|\beta$ để chỉ ra rằng phần tử đầu tiên của vùng đệm là từ v, ký hiệu $\sigma|u$ để chỉ ra rằng phần tử trên cùng của ngắn xếp là từ u, và $A = (x, y)$, trong đó x, y là các từ của câu cần được phân tích để chỉ ra tập cung phụ thuộc của một cấu hình c. Bốn loại hàm chuyển được định nghĩa như sau:

1. LEFT-ARC(r):

$$(\sigma|u, v|\beta, A) \rightarrow (\sigma, v|\beta, A \cup (v, u)) \text{ với điều kiện } \nexists k : (k, u) \in A$$

2. RIGHT-ARC(r):

$$(\sigma|u, v|\beta, A) \rightarrow (\sigma|u|v, \beta, A \cup (u, v)) \text{ với điều kiện } \nexists k : (k, v) \in A$$

3. REDUCE:

$$(\sigma|u, \beta, A) \rightarrow (\sigma, \beta, A) \text{ với điều kiện } \exists v : (v, u) \in A$$

4. SHIFT:

$$(\sigma, v|\beta, A) \rightarrow (\sigma, \beta, A)$$

Bốn hàm chuyển trên có thể được giải thích một cách rõ ràng như sau:

- Bước chuyển LEFT-ARC(r): $u \leftarrow v$ nếu không tồn tại bất kỳ cung nào đi đến u hay nói cách khác u không phải là phụ thuộc của bất cứ từ nào thì phân tích của u sẽ được thực hiện, có một cung đi từ v đến u với nhãn r. Khi đó u sẽ được lấy ra khỏi ngắn xếp.
- Bước chuyển RIGHT-ARC(r): $u \rightarrow v$ là nếu không tồn tại bất kỳ cung nào đến v thì v được đưa vào trong ngăn xếp để xét các từ tiếp theo. Chú ý rằng có thể có nhiều cung đi ra từ u.
- Bước chuyển REDUCE: Là bước lấy một từ u ra khỏi ngăn xếp nếu như có một quan hệ phụ thuộc giữa từ u và từ v trong bước chuyển RIGHT-ARC trước đó.
- Bước chuyển SHIFT: Là bước lấy phần tử đầu tiên của vùng đệm và đẩy nó vào trong ngăn xếp. Quá trình chuyển này không đòi hỏi bất cứ điều kiện tiên quyết nào.

Ví dụ minh hoa với câu *Book me the morning flight*. Chúng ta có như hình 6.8. Ở đây, để đơn giản, thì ta chưa xét đến nhãn r ở các hàm chuyển.

Step	Stack	Word List	Action	Relation Added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book → me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning ← flight)
7	[root, book, the, flight]	[]	LEFTARC	(the ← flight)
8	[root, book, flight]	[]	RIGHTARC	(book → flight)
9	[root, book]	[]	RIGHTARC	(root → book)
10	[root]	[]	Done	

Hình 6.8: Ví dụ minh hoạ thuật toán phân tích cú pháp phụ thuộc dựa trên bước chuyển

Hệ thống bước chuyển được xác định là không đơn định, vì thế thường có nhiều hơn một bước chuyển đối với một cấu hình nhất định. Để thực hiện phân túch cú pháp đơn định, hệ thống các bước chuyển cần phải bổ sung một kỹ thuật để dựa đoán bước chuyển tiếp theo ở mỗi lựa chọn không đơn định, cũng như lựa chọn một loại phụ thuộc r cho quá trình chuyển đổi LEFT-ARC(r) và RIGHT-ARC(r). Nếu trạng thái phân tích cú pháp chưa phải trạng thái kết, thì hệ thống sẽ tiếp tục thực hiện các trạng thái tiếp theo, nếu ngăn xếp rỗng thì sẽ thực hiện bước chuyển SHIFT, ngược lại sẽ thực hiện một hàm chức năng để đưa ra bước chuyển kế tiếp, hàm này được dự đoán bằng các thuật toán huấn luyện dựa vào các đặc trưng của mô hình. Khi thực hiện đến cấu hình kết, thì ta thu được đồ thị phụ thuoc của câu đầu vào. Đồ thị phụ thuộc được đưa ra cuối cùng đảm bảo không có chu trình và không xạ ảnh.

Các mô hình đặc trưng cho phân tích cú pháp phụ thuộc dựa vào bước chuyển thường kết hợp các đặc trưng từ loại, từ vựng với các đặc trưng phụ thuộc như nhãn phụ thuộc hay từ trung tâm trong quan hệ phụ thuộc của các từ trong ngăn xếp hay trong bộ đếm. Mô hình đặc trưng chuẩn là mô hình kết hợp các đặc trưng từ loại, từ vựng và loại phụ thuộc, theo hình 6.9.

$p(\sigma_1)$	$w(h(\sigma_0))$	$d(l(\sigma_0))$
$p(\sigma_0)$	$w(\sigma_0)$	$d(\sigma_0)$
$p(\tau_0)$	$w(\tau_0)$	$d(r(\sigma_0))$
$p(\tau_1)$	$w(\tau_1)$	$d(l(\tau_0))$
$p(\tau_2)$		
$p(\tau_3)$		

Hình 6.9: Các đặc trưng dùng trong MaltParser

Mô hình này chứa 6 đặc trưng từ loại, là từ loại của hai từ trên cùng của ngăn xếp là $(p(\sigma_0), p(\sigma_1))$ và 4 từ đầu tiên của đầu vào là $p(\tau_0), p(\tau_1), p(\tau_2), p(\tau_3)$. Các đặc tính loại phụ thuộc bao gồm từ trên đầu của ngăn xếp $d(\sigma_0)$, và con trái nhất, con phải nhất của nó là $(d(r(\sigma_0), d(l(\sigma_0)))$ và con trái nhất của từ tiếp theo của đầu vào là $d(l(r_0))$. Cuối cùng, mô hình chuẩn chứa 4 đặc tính từ vựng, là dạng từ của từ đầu tiên trong ngắn xếp $w(\sigma_0)$, đầu của từ đầu tiên trong ngắn xếp $w(h(\sigma_0))$, và hai từ tiếp theo ở đầu vào là $(w(\tau_0), w(\tau_1))$.

Khi dùng các đặc trưng này, các từ trong câu thường được mã hoá và biểu diễn bằng one-hot vector hay cũng gọi là vector chỉ số với các giá trị trong vecto là 0 hoặc 1. Đây là cách biểu diễn này khá đơn giản và dễ hiểu, được áp dụng trong rất nhiều những hệ thống của xử lý ngôn ngữ tự nhiên. Tuy nhiên, biểu diễn theo dạng này gặp phải hai vấn đề lớn. Một là, dữ liệu thưa, các thông số tương ứng với các từ hiếm hoặc các từ không xác định thường ước tính kém. Hai là, nó không có khả năng nắm bắt sự giống nhau về ngữ nghĩa giữa các từ có liên quan chặt chẽ đến nhau. Sự hạn chế này đã thúc đẩy các phương pháp giám sát để tạo ra một biểu diễn từ tốt hơn. Gần đây,

biễn diễn phân tán từ được chứng minh là đã đạt được nhiều kết quả tốt trong các bài toán xử lý ngôn ngữ tự nhiên. Biểu diễn phân tán (hay còn được gọi là nhúng từ - *Word embedding*) có thể được sử dụng cho các đơn vị khác nhau của ngôn ngữ như từ, cụm từ, câu và các tài liệu. Sử dụng biểu diễn phân tán, các dơn vị ngôn ngữ được nhúng trong một khong gian ít chiều và liên tục. Mỗi chiều của biểu diễn phân tán đại diện cho một số tính năng tiềm ẩn của từ và hy vọng có thể nắm bắt được các đặc tính về cú pháp và tương đồng ngữ nghĩa. Thông thường, các biểu diễn phân tán thường được tạo ra bằng cách sử dụng mô hình mạng nơ-ron, trong đó các mạng nơ-ron được sử đụng để dự đoán. Một số mô hình đã được phát triển để tạo ra biểu diễn phân tán từ như: mô hình *skip-gram* và mô hình *CBOW*. Phương pháp này đã và đang được sử dụng trong nhiều vấn đề liên quan đến phân tích cú pháp phụ thuộc. Nó được chứng minh rằng đã đạt được hiệu quả cao và có thể áp dụng cho nhiều ngôn ngữ khác nhau. Ngoài ra, biểu diễn phân tán còn được sử dụng để phân tích cú pháp phụ thuộc đa ngôn ngữ. Phương pháp này cũng đã được nhóm các tác giả Lê Hồng Phương và cộng sự sử dụng và đem lại kết quả khá khả quan đối với tiếng Việt.

Dựa vào các đặc trưng, vấn đề huấn luyện được chuyển thành vấn đề phân loại, trong đó đầu vào là các vector đặc trưng và các lớp đầu ra là những quyết định trong phân tích cú pháp. Huấn luyện mô hình phân tích cú pháp phụ thuộc là bước quan trọng để có một kết quả tốt.

Ví dụ phân tích cú pháp phụ thuộc dựa vào các bước chuyển đổi với câu " *He had good control.*" ở hình 6.10.

Trasition	**Stack**	**Buffer**	**A**
	[ROOT]	[He has good control .]	$\emptyset$
SHIFT	[ROOT He]	[has good control .]	
SHIFT	[ROOT He has]	[good control .]	
LEFT-ARC(nsubj)	[ROOT has]	[good control .]	$A \cup nsubj(has, He)$
SHIFT	[ROOT has good]	[control .]	
SHIFT	[ROOT has good control]	[.]	
LEFT-ARC(amod)	[ROOT has control]	[.]	$A \cup amod(control, good)$
RIGHT-ARC(dobj)	[ROOT has]	[.]	$A \cup dobj(has, control)$
...	...	...	...
RIGHT-ARC(root)	[ROOT]	[]	$A \cup root(ROOT, has)$

Hình 6.10: Ví dụ phân tích cú pháp dựa vào các bước chuyển

6.2.2 Phân tích cú pháp phụ thuộc dựa trên đồ thị

Phương pháp tiếp cận dựa trên đồ thị tìm kiếm cây phân tích cú pháp có điểm số lớn nhất trong không gian có thể của cây. Các phương pháp này mã hoá không gian tìm kiếm là các đồ thị có hướng và sử dụng các thuật toán trong lý thuyết đồ thị để tìm kiếm cây tối ưu. Cho một câu đầu vào $x = w_0, w_1, \ldots, w_n$ có tập đỉnh là V_x, ta

định nghĩa lại tập cung E_x của đồ thị phụ thuộc câu x như sau:

$$E_x = \{(i, j, r) | i, j \in V_x; r \in L\}$$

G_x là những đồ thị phụ thuộc đúng của câu x. $D(G_x)$ là những đồ thị con của G_x. Vì G_x chứa tất cả những cung được gán nhãn, tập $D(G_x)$ phải chứa tất cả những đồ thị phụ thuộc của x.

Giả sử tồn tại một hàm tính trọng số của cạnh phụ thuộc $s : V \times V \times L \to R$. Định nghĩa trọng số của một đồ thị là tổng các trọng số của cạnh trong đồ thị đó:

$$s(G_x = (V_x, E_x)) = \sum_{(i,j,r) \in R_x} s(i, j, r)$$

Trọng số của một cạnh $s(i, j, r)$ biểu diễn khả năng tạo ra quan hệ phụ thuộc r giữa từ trung tâm w_i và với từ phụ thuộc w_j trong đồ thị phụ thuộc. Trọng số của cạnh được định nghĩa là tích của các vectơ đặc trưng f với các vectơ tham số w:

$$s(i, j, r) = w.f(i.j.r)$$

Các đặc trưng đại diện $f(i, j)$ được trình bày trong hình 6.11 cho một cung không được gán nhãn (i, j).

(a) Đặc trưng Uni-gram	**(b) Đặc trưng Bi-gram**	**(c) Đặc trưng từ loại**
$x_i - word, x_i - pos$	$x_i - word, x_i - pos, x_j - pos, x_j - word$	$x_i - pos, x_b - pos, x_j - pos$
$x_i - word$	$x_i - pos, x_j - pos, x_j - word$	$x_i - pos, x_{i+1} - pos, x_{j-1} - pos, x_j - pos$
$x_i - pos$	$x_i - word, x_j - pos, x_j - word$	$x_{i-1} - pos, x_i - pos, x_{j-1} - pos, x_j - pos$
$x_j - word, x_j - pos$	$x_i - word, x_i - pos, x_j - pos$	$x_i - pos, x_{i+1} - pos, x_j - pos, x_{j+1} - pos$
$x_j - word$	$x_i - word, x_i - pos, x_j - word$	$x_{i-1} - pos, x_i - pos, x_j - pos, x_{j+1} - pos$
$x_j - pos$	$x_i - word, x_j - word$	$x_{i-1} - pos, x_i - pos, x_j - pos, x_{j+1} - pos$
	$x_i - pos, x_j - pos$	

Hình 6.11: Các đặc trưng dùng trong MSTParser

Những đặc trưng này đại diện cho các thông tin liên quan đến từ trung tâm trong quan hệ phụ thuộc, nhãn phụ thuộc. Ngoài ra còn có cả những đặc trưng về nhãn từ loại của các từ kế tiếp (bao gồm cả nhãn thô và nhãn mịn). Cụ thể với một cung (i, j), ta có:

- Nhóm đặc trưng (a) và (b): Xét cho từ loại và từ vựng của cung (i, j) trong ngữ cảnh Uni-gram và Bi-gram.
- Nếu i và j có nhiều hơn 5 ký tự thì xét thêm đặc trưng 5-gram phía trước từ đó.
- Nhóm (c): bổ sung cho bối cảnh đồ thị phụ thuộc (nhóm (a) và (b)), ta xét các từ trong bối cảnh câu, cụ thể là thông qua từ loại của các từ nằm giữa i và j, cùng với từ loại của các từ nằm bên trái và bên phải từ i và từ j.

Các tác giả đã thử thêm bớt hoặc thay đổi nhiều lần các đặc trưng và chứng minh bằng thực nghiệm rằng các đặc trưng này là hiệu quả nhất đối với phân tích cú pháp phụ thuộc cho tiếng Anh.

Vectơ w là một vectơ trọng số được đưa ra cho mỗi câu bằng phương pháp học máy (MIRA - *Margin Infused Relaxed Algorithm*). Phương pháp học máy MIRA được lựa chọn vì nó có nhiều những đặc tính phù hợp với bài toán phân tích cú pháp phụ thuộc.

Khi hàm trọng số của cạnh đã có, thì việc phân tích cú pháp có thể được biểu diễn:

$$G^* = argmax_{G \in F(G_x)} s(G) = argmax_{G \in D(G_x)} \sum_{(i,j,r) \in E_x} s(i,j,r)$$

McDonald cùng cộng sự (2005) chỉ ra vấn đề này tương đương với việc tìm cây bao trùm cực đại có hướng của đồ thị G_x ban đầu. **Thuật toán Chu-Lin-Edmonds** được sử dụng để tìm ra cây bao trùm lớn nhất trong đồ thị có hướng với trường hợp không xạ ảnh (Hình 6.12). **Thuật toán Eisner** cũng được sử dụng để tìm ra cây bao trùm lớn nhất trong đồ thị có hướng với trường hợp xạ ảnh.

function MAXSPANNINGTREE(*G=(V,E), root, score*) **returns** *spanning tree*

F ← []
T' ← []
score' ← []
for each *v* ∈ *V* **do**
  bestInEdge ← argmax$_{e=(u,v) \in E}$ *score[e]*
  F ← *F* ∪ *bestInEdge*
  for each *e=(u,v)* ∈ *E* **do**
    score'[e] ← *score[e]* − *score[bestInEdge]*

if *T=(V,F)* is a spanning tree **then return** it
else
  C ← a cycle in *F*
  G' ← CONTRACT(*G, C*)
  T' ← MAXSPANNINGTREE(*G', root, score'*)
  T ← EXPAND(*T', C*)
  return *T*

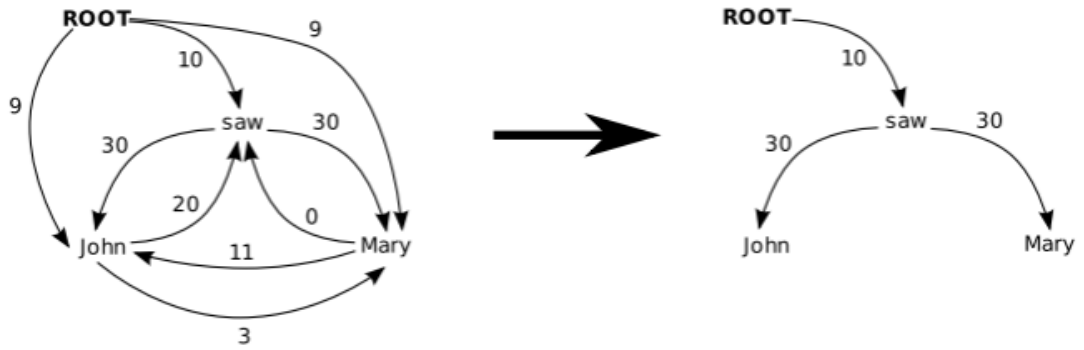
function CONTRACT(*G, C*) **returns** *contracted graph*

function EXPAND(*T, C*) **returns** *expanded graph*

Hình 6.12: Thuật toán Chu-Lin-Edmonds

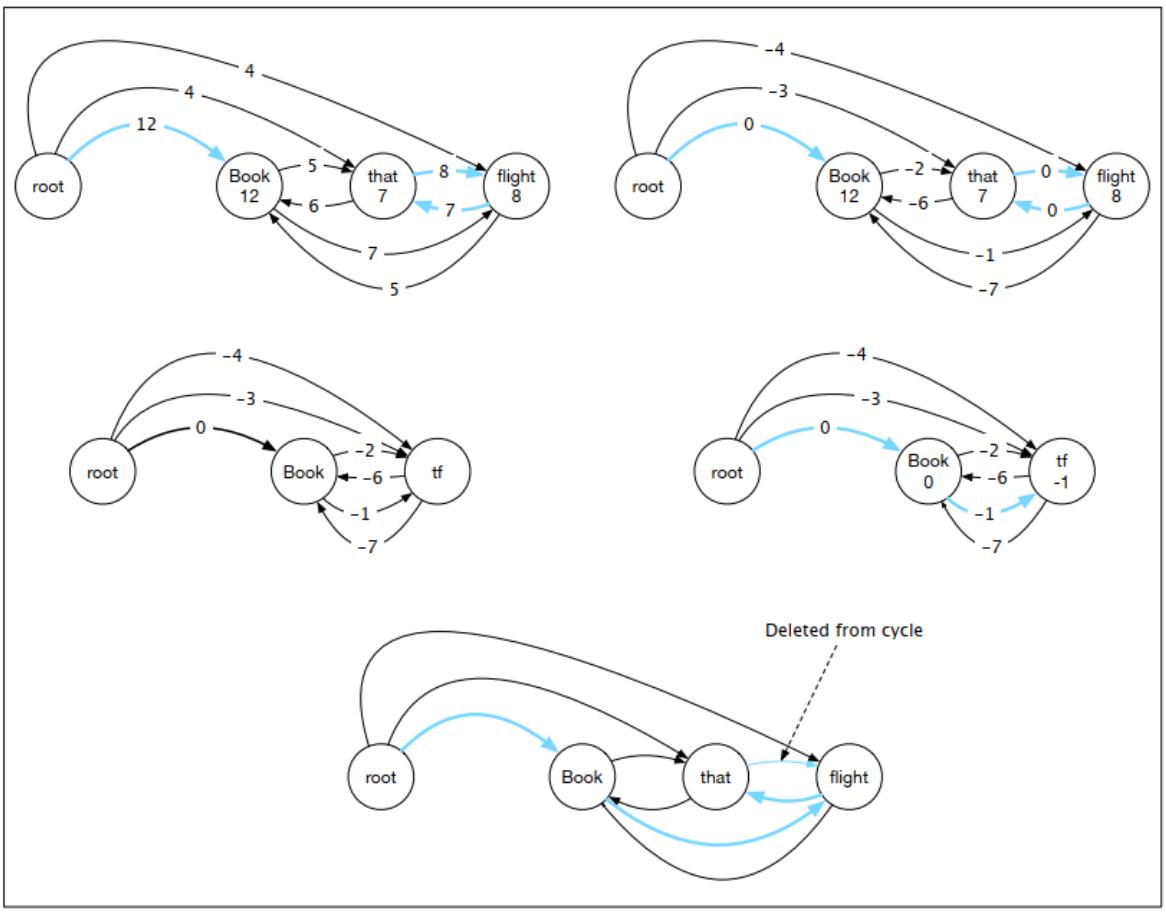
Một ví dụ của đồ thị đầy đủ G_x và đồ thị phụ thuộc có hàm trọng số cao nhất được đưa ra như trong hình 6.13 cho câu *"John saw Mary"*. Hình 6.13 gồm đồ thị đầy đủ G_x

chứa trọng số trên các cạnh, sau đó dựa vào thuật toán phân tích cú pháp phụ thuộc trên đồ thị để chuyển thành đồ thị phụ thuộc chính xác của câu.



Hình 6.13: Ví dụ về phân tích cú pháp dựa trên đồ thị

Ta cũng có ví dụ khác về thuật toán Chu-Lin-Edmonds cho câu *"Book that flight"* như hình 6.14.



Hình 6.14: Ví dụ về thuật toán Chu-Lin-Edmonds