# SỬ DỤNG KỸ THUẬT PRUNING VÀO BÀI TOÁN XÁC ĐỊNH TỪ LOẠI

**Nguyễn Chí Hiếu[1], Phan Thị Tươi[2], Nguyễn Xuân Dũng[3], Nguyễn Quang Châu[1]**
(1) Trường Đại học Công Nghiệp thành phố Hồ Chí Minh
(2) Trường Đại học Bách Khoa, Đại học Quốc gia Hồ Chí Minh
(3) Phân viện Công nghệ Thông tin thành phố Hồ Chí Minh

*(Bài nhận ngày 14 tháng 9 năm 2005, hoàn chỉnh sửa chữa ngày 18 tháng 11 năm 2005)*

***TÓM TẮT***: Xác định tự động chính xác từ loại cho các từ trong văn bản của ngôn ngữ tự nhiên là một vấn đề quan trọng. Việc xác định chính xác từ loại sẽ hỗ trợ cho việc phân tích cú pháp các văn bản, góp phần giải quyết tính đa nghĩa của từ, trợ giúp truy xuất thông tin hướng đến ngữ nghĩa, v.v.... Vì vậy thời gian xác định từ loại cho các từ trong văn bản (nhất là những văn bản có số lượng từ lớn) luôn được quan tâm. Bài báo này trình bày kỹ thuật pruning để xác định chuỗi từ loại thích hợp cho các từ trong văn bản của ngôn ngữ tự nhiên – giải thuật Viterbi [4]. Giải thuật Viterbi sử dụng kỹ thuật pruning được thử nghiệm trên tập ngữ liệu 1,000,000 từ được rút trích từ kho ngữ liệu "Phố Uôn" [8].

## 1. GIỚI THIỆU

### 1.1. Mở đầu

"Tagging" là thuật ngữ tiếng Anh để biểu thị cho việc xác định (hay gọi là gắn) từ loại (mã từ loại) thích hợp cho các từ trong văn bản của ngôn ngữ tự nhiên. Xác định từ loại thích hợp cho các từ là một công việc quan trọng trong xử lý ngôn ngữ tự nhiên. Hiện nay đã tồn tại một số phương pháp gắn từ loại [6]:

- Hệ thống luật sinh (hệ thống TAGGIT).
- Phương pháp xác suất dựa trên mô hình Markov ẩn.
- Phương pháp kết hợp giữa tập luật sinh và sinh tự động từ loại để gắn mã từ loại chính xác hơn với sự ràng buộc giữa luật sinh và thống kê [9].

Các phương pháp này đều có độ chính xác khá cao từ 95% đến 97% [4].

Tuy nhiên để thực hiện gắn từ loại cho các từ trong các văn bản lớn thì thời gian là yếu tố rất quan trọng. Vì vậy các tác giả bài báo thử nghiệm sử dụng kỹ thuật *pruning* để tăng tốc độ xử lý cho quá trình gắn từ loại các từ trong giải thuật Viterbi [4]. Giải thuật Viterbi nguyên thủy có độ phức tạp là $O(K^2 * T)$; trong đó K là số nút trạng thái của mô hình và T là chiều dài của chuỗi nhập. Khi K có kích thước lớn thì giải thuật trở nên chậm. Thí dụ, xét mô hình Markov để gắn nhãn mã từ loại Trigram, với số mã từ loại là 40, nên chúng ta có $40^2$ ( bằng 1.600 trạng thái và 41 trạng thái khởi tạo). Do đó để tính đầy đủ cho một trạng thái kế tiếp giải thuật cần tính $40^3$ (bằng 64.000) khả năng. Vì vậy người ta đã tìm cách giảm thiểu số lần tính toán này trong các cài đặt thực tế - các kỹ thuật giảm thiểu này còn được gọi là các kỹ thuật *pruning*.

### 1.2. Các Kỹ thuật Pruning hiện tại [11][10][1]

Thuật ngữ "*Pruning*" thực chất là phương pháp nhánh cận, nó cắt bỏ một phần không gian tìm kiếm, để giúp quá trình tìm kiếm được nhanh hơn, phần bị loại bỏ này được xem là không quan trọng và không chứa giải pháp tối ưu mà chúng ta đang tìm kiếm. Trước khi khảo sát các kỹ thuật Pruning hiện tại chúng ta xem lại sơ lược về mô hình Markov ẩn.

#### *1.2.1. Mô hình Markov ẩn*

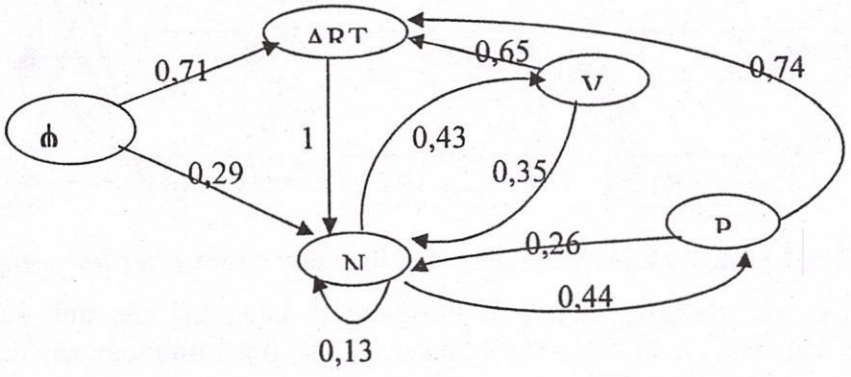
Đầu thế kỷ 20 Andrei Andreyevich Markov đã đưa ra một mô hình toán học, để mô tả sự thay đổi từ nguyên âm sang phụ âm và ngược lại của 20.000 từ trong một vở kịch của Puskin. Về sau mô hình này phát triển và được ứng dụng trong nhiều lĩnh vực như: Viễn thông, tin học, vật lý, sinh học, y học, kinh tế,... và được mang tên là quá trình Markov.

Chuỗi Markov là trường hợp riêng của quá trình Markov (khi ta có thể đánh số được các trạng thái).

Để xác định *tính Markov* của một hệ, người ta nghiên cứu sự tiến triển theo thời gian của một hệ nào đó. Ký hiệu X(t) là vị trí của hệ tại thời điểm t trong không gian trạng thái S. Tập hợp các vị trí có thể có của hệ được gọi là không gian trạng thái. Giả sử trước thời điểm s hệ ở trạng thái nào đó, còn ở thời điểm s hệ ở trạng thái i. Ta cần biết tại thời điểm t trong tương lai(t>s) hệ ở trạng thái j với xác suất là bao nhiêu? Nếu xác suất này chỉ phụ thuộc vào s,t,i,j thì điều này có nghĩa là : *Sự tiến triển của hệ trong tương lai chỉ phụ thuộc vào hiện tại và độc lập với quá khứ* [5]. Đó là *tính Markov*, hệ có tính chất này được gọi là quá trình Markov (hay mô hình Markov). Về phương diện toán học, tính Markov có thể định nghĩa như sau: Giả sử $S = \{S_0, S_1, \ldots, S_n\}$ là các trạng thái của hệ ở các thời điểm 0,1,2..,n. Ta nói rằng $S_t$ có tính Markov nếu xác suất: $P(S_{t-1} \mid S_t) = P(S_0 S_1 \ldots S_{t-1} \mid S_t)$.

Mô hình Markov ẩn *(HMM – Hidden Markov Models)* được định nghĩa là một cặp quá trình xử lý (S, O). Quá trình S theo tính chất của chuỗi Markov (tương lai chỉ phụ thuộc hiện tại), và không quan sát trực tiếp được, trong khi quá trình O là một chuỗi biến ngẫu nhiên lấy giá trị trong không gian trạng thái hoặc quan sát. Hai giả thuyết chính của HMM được áp dụng trong POS là *tương lai chỉ phụ thuộc hiện tại* và giả thuyết *độc lập* cho các trạng thái. Trong bài toán gắn nhãn từ loại thì S là nhãn cần gắn và O là từ. Với giả thiết rằng chúng ta chỉ có thể quan sát được chuỗi từ và không quan sát được chuỗi nhãn (*tag*). Nên trong mô hình Markov ẩn (HMM) các biến nhãn là ẩn; kết quả đạt được nó không quan sát được. [6]

Ví dụ: Mô hình Markov ẩn với xác suất bigram ở H.1




**H 1: Mô hình Markov ẩn [4, tr.200]**

Từ "ẩn" ở đây có nghĩa là với một chuỗi từ cho trước không thể biết rằng mô hình Markov tương ứng đang ở trạng thái thái nào (*sẽ được gắn những nhãn nào*). Ví dụ chuỗi từ trong tiếng Anh: *Flies like a flower*, từ *flies* nếu ở trạng thái là danh từ (N) thì có xác suất là 0,25, nhưng nếu ở trạng thái là động từ (V) thì có xác suất là 0,76 [4, tr.199]. Như vậy xác suất của chuỗi: *flies like the flower* với các từ loại khác nhau sẽ khác nhau.

*1.2.2. Beam Search pruning*

Ý tưởng đơn giản nhất là phân loại theo xác suất các nút trước, và chỉ mở rộng không gian tìm kiếm trên những nút *có khả năng nhất*. Kỹ thuật này gọi là *Beam search pruning*. Quá trình pruning được làm tại mỗi nút (khi xác định xác suất của nó) cho tới nút cuối cùng. Khi áp dụng kỹ thuật này vào giải thuật Viterbi nguyên thủy người ta gọi nó là *Viterbi-Beam Search*. Trong giải thuật Viterbi nguyên thủy mỗi một từ ($W_t$) trong chuỗi nhập đều phải tính lặp đi lặp lại với $K^2$ phép toán để xác định xác suất của nó, cụ thể là xác suất $\delta_t(i)$ được tính:

$$\delta_t(i) = \text{Max}_{j=1,N}(\delta_{t-1}(j) * P(C_i \mid C_j)) * P(W_t \mid C_i) \quad (1)$$

Trong đó :

- $W_t$: từ thứ *t* trong chuỗi nhập
- $C_i$ : mã từ loại thứ i của từ
- $\delta_t(i)$: xác suất của từ loại thứ *t* có mã từ loại là $C_i$

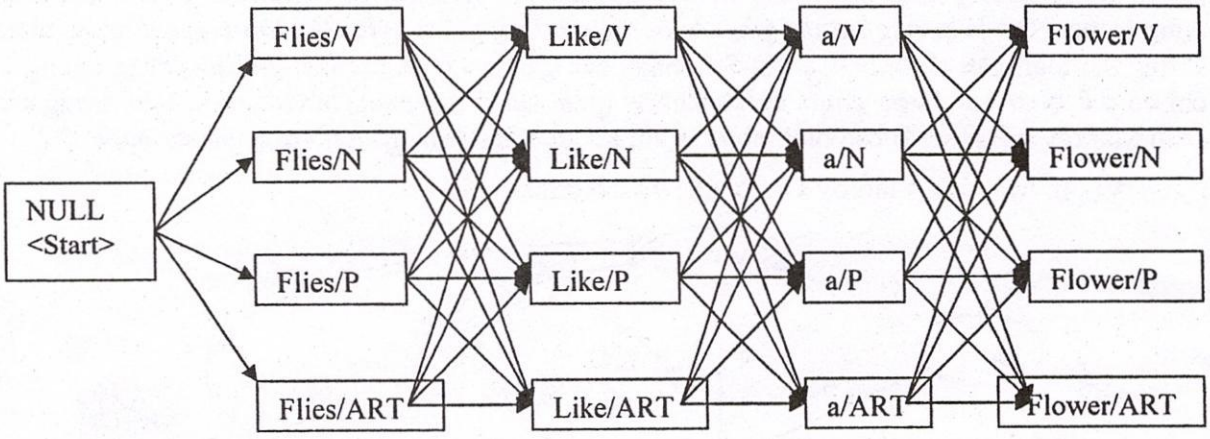
- $P(C_i \mid C_j)$: xác suất của mã từ loại $C_i$ trong ngữ cảnh mã từ loại $C_j$
- $P(W_t \mid C_i)$: xác suất từ $W_t$ trong ngữ cảnh mã từ loại $C_i$.

*Beam Search* cắt bớt những trạng thái không có nhiều triển vọng tại mỗi bước, chỉ để lại những nút được cho là tốt nhất để sử dụng tính toán cho các trạng thái kế tiếp. Thí dụ, tại nút B. Luân phiên, chúng ta lựa chọn một vài hệ số *h*, và xóa bỏ tất cả các trạng thái có xác suất nhỏ hơn *M*h*. Ở đây *M* là xác suất lớn nhất của các trạng thái ở bước trước bước đang xét. Như vậy biểu thức (1) được viết lại là:

$$\delta_t(i) = \text{Max}_{\,j \in \text{Beam}(t-1)} (\delta_{t-1}(j) * P(C_i \mid C_j)) * P(W_t \mid C_i) \qquad (2)$$

Khi đó chúng ta có thể tính Beam(t) bằng cách tìm giá trị lớn nhất $M_t$ tại thời điểm t, và xóa bỏ các trạng thái nhỏ hơn ngưỡng $M_t * h$. Với chiến lược này chúng ta thu giảm số phép toán phải thực hiện trong vòng lặp xuống còn B*K, trong đó B là số trung bình các trạng thái trong ngưỡng từ $M_t$ tới $M_t * h$.

Thí dụ, thiết lập các khả năng có thể có của chuỗi *Flies like a flower,* với mỗi từ có nhiều nhất là 4 khả năng từ loại là V, N, P, ART. Theo giải thuật vét cạn chúng ta có $T^K = 4^4 = 256$ khả năng kết hợp để tính xác suất khi gắn nhãn.




**H 2: Mã hóa 256 chuỗi khả năng kết hợp cho chuỗi *Flies like a flower***

Với giả thuyết Markov, Viterbi đã thu giảm số khả năng cần tính toán xuống còn $K^2 * T = 64$ khả năng cho chuỗi *Flies like a flower*, và giải thuật tính toán này được gọi là giải thuật Viterbi.

| | ART | N | V | P |
|---|---|---|---|---|
| **Flies** | $7{,}1\ e^{-5}$ | **$7{,}22\ e^{-3}$** | $7{,}59\ e^{-6}$ | $1{,}00\ e^{-8}$ |
| **like** | $4{,}93\ e^{-10}$ | $1{,}11\ e^{-5}$ | **$3{,}10\ e^{-4}$** | **$2{,}16\ e^{-4}$** |
| **A** | **$7{,}25\ e^{-5}$** | $9{,}76\ e^{-8}$ | $4{,}48\ e^{-10}$ | $4{,}95\ e^{-10}$ |
| **flower** | $7{,}25\ e^{-13}$ | $4{,}56\ e^{-6}$ | $2{,}10\ e^{-9}$ | $3{,}27\ e^{-9}$ |

**H 3: Search Viterbi đầy đủ cho chuỗi *Flies like a flower***

| | ART | N | V | P |
|---|---|---|---|---|
| **Flies** | **$7{,}1\ e^{-5}$** | **$7{,}22\ e^{-3}$** | | |
| **like** | | | **$3{,}10\ e^{-4}$** | **$2{,}16\ e^{-4}$** |
| **a** | **$7{,}25\ e^{-5}$** | | | |
| **flower** | | | | |

**H 4: Viterbi Beam Search sử dụng hệ số h = 0,1 cho chuỗi *Flies like a flower***

Cách cắt bỏ đơn giản nhất là tại mỗi bước người ta cố định D trạng thái lớn được giữ lại.

| | ART | N | V | P |
|---|---|---|---|---|
| **Flies** | **$7{,}1\ e^{-5}$** | **$7{,}22\ e^{-3}$** | **$7{,}59\ e^{-6}$** | **$1{,}00\ e^{-8}$** |
| **like** | | **$1{,}11\ e^{-5}$** | **$3{,}10\ e^{-4}$** | **$2{,}16\ e^{-4}$** |
| **A** | **$7{,}25\ e^{-5}$** | | | |
| **flower** | | | | |

**H5. Viterbi Beam Search cố định D = 2 cho chuỗi *Flies like a flower***

Ngày nay Viterbi beam search với sự trợ giúp của các kỹ thuật *pruning* hiệu quả là phương pháp thích hợp cho tất cả các tác vụ (task) của nhận dạng tiếng nói (SR: Speech Recognition). Kỹ thuật *Pruning* không những được áp dụng trong những bài toán về nhận dạng tiếng nói, mà còn được áp dụng trong dịch máy thống kê (SMT: Statisticaḷ Machine Translation).

*1.2.3. Pruning trong dịch máy thống kê [1]*

Trong nghiên cứu của mình Tillmann và Ney đưa ra 4 loại ngưỡng để pruning :

- Ngưỡng pruning mức độ bao phủ $t_C$.
- Ngưỡng biểu đồ mức độ bao phủ $n_C$.
- Ngưỡng pruning tập hợp $t_c$.
- Ngưỡng biểu đồ tập hợp $n_c$.

Các tác giả tổng hợp lại và đưa ra ba loại pruning là:

1) Pruning dịch dựa theo xác suất dịch:

$Q_{1e'}(e,C,j) < t_C * Q_1(C)$

Và : $Q_{1e'}(e,C,j) < t_c * Q_1(c)$

***Công thức tính xác suất từng phần tốt nhất:***

$$Q_{1e'}(e,C,j) = Q_{e'}(e,C,j) * \prod P_1(f)\ ,\ j \in C_1\ ; \quad (3)$$

Trong đó:

- $C_1 = \{1,\ldots,J\}/\ C$ (loại trừ các phần tử thuộc C).
- Cận bao trên $P_1(f)$ lấy tích xác suất của mô hình ngôn ngữ và mô hình dịch:

$$P_1(f) = \max \{p(e \mid e', e'') * p(f \mid e)\} \quad (4)$$

***Công thức xác suất từng phần $Q_{e'}(e,C,j)$ được tính như sau:***

$$Q_{e'}(e,C,j) = P(f_j \mid e) * \max \{P(j \mid j', J) * P(e \mid e', e'') * Q_{e''}(e', C/\{j\}, j')\} \quad (5)$$

$C = \{ b_k \mid k = 1,\ldots,i\}$, $b_i = j$, $e_i = e$, và $e_{i-1} = e^j$

- $b_i$ : Đối sánh ngược $i \rightarrow j$
- e, f : tương ứng là các câu tiếng Anh và tiếng Pháp
- $Q_1(C) = \max Q_{1e'}(e,C,j)$ : *(Coverage pruning)*
- $Q_1(c) = \max Q_1(C)$ : *(Cardinality pruning)*

2) Pruning biểu đồ:

- $N(C) > n_C$ ; $N(c) > n_c$
- N(C) : Toàn bộ số phần tử của tập phủ C
- N(c) : Toàn bộ số phần tử của tập con c, không vượt quá số đã cho

3) Pruning quan sát: $P(f \mid e) * P_{uni}(e)$

- $P(f \mid e)$: là xác suất dịch từ ngôn ngữ nguồn f sang ngôn ngữ đích e
- $P_{uni}(e)$ : là xác suất unigram của từ ngôn ngữ đích e.

## 2. ĐỀ XUẤT GIẢI PHÁP PRUNING

Việc loại bỏ các đường dẫn trên những tiêu chuẩn đơn giản (Hệ số *h*, cố định số D) có thể dẫn đến làm mất lời giải, và đó là nhược điểm của các phương pháp tìm kiếm sử dụng *heuristic* (thực tế ít khi xảy ra), nhưng số trạng thái thăm dò khi tìm kiếm được giảm đáng kể.

Hơn nữa bằng cách thay đổi các mức ngưỡng để tìm giải pháp thỏa hiệp giữa độ chính xác cần đạt được và tốc độ tính toán nhanh có thể tìm được một hệ số phù hợp. Tiêu chuẩn pruning dựa trên cơ sở so sánh giữa xác suất của trạng thái hiện đang tính với xác suất của trạng thái tốt nhất, Beam Search trở nên kém hiệu quả khi hầu hết các trạng thái có xác suất gần bằng với xác suất của trạng thái tốt nhất. Làm thế nào tìm được *beam* tốt là một vấn đề - hãy xem một số ưu khuyết điểm của các phương pháp pruning trong SR. [10]

| Phương pháp Pruning | Ưu điểm | Nhược điểm |
|---|---|---|
| Beam Search | Không cần sắp xếp<br>Chiều rộng beam tự động | Fuzzy scores ⇒ chiều rộng beam<br>Chiều rộng Beam = ? |
| Giới hạn số trạng thái D | Cố định chiều rộng beam độc lập với dữ liệu | Cần sắp xếp |

**H 6: Bảng so sánh ưu nhược điểm của 2 phương pháp *pruning***

Trong biểu thức (2) các kỹ thuật *pruning beam search* chỉ tập trung vào thành phần $\delta_{t-1}(j)$ và xác định mức ngưỡng theo thực nghiệm. Đóng góp quan trọng của giải pháp này là *xây dựng công thức tính mức ngưỡng cho tích* $\delta_{t-1}(j) * P(C_i \mid C_j)$ trong bài toán xác định từ loại. *Các mức ngưỡng sẽ được tính chính xác trong các bước thực hiện của giải thuật cho chuỗi nhập bất kỳ*. Bài báo đã sử dụng mô hình Markov ẩn trong quá trình tính toán của mình.

### 2.1. Bài toán xác định từ loại

*2.1.1. Từ và từ loại*

Câu trong ngôn ngữ tự nhiên được tạo nên từ các từ. Các từ trong câu được sắp xếp ở các vị trí theo một cấu trúc hợp lệ, được gọi là cấu trúc cú pháp. Các từ trong cấu trúc câu sẽ có những vai trò như danh từ, động từ, tính từ, trạng từ, hình thái từ. Vai trò của từ trong câu được gọi là từ loại của từ.

Ví dụ 1: - Từ "cưa" có từ loại là danh từ (ký hiệu là N) trong câu: *Cái cưa này cũ quá.*
- Từ "cưa"có từ loại là động từ (V) trong câu: *Tôi cưa miếng gỗ ra làm nhiều mảnh.*

Ví dụ 2: Từ "can" trong tiếng Anh có thể có ba từ loại: động từ (V), danh từ (N), trợ động từ (AUX) trong câu sau: *The large can can hold the warter.*

Khi xử lý ngôn ngữ tự nhiên máy tính thường xử lý các chuỗi từ dựa trên từ loại của chúng. Việc xác định từ loại nào của từ trong câu là thích hợp là một bài toán đã có nhiều giải thuật. Một giải thuật hiệu quả là giải thuật Viterbi, dựa trên mô hình Markov ẩn (Hidden Markov Model [4], tr.200).

*2.1.2. Xác định từ loại cho từ trong câu [4]*

- Gọi $O \in \Sigma^*$ là chuỗi đã biết của tập ký hiệu quan sát được, $S \in \Omega^*$ là chuỗi chưa biết của các trạng thái trong O, được sinh ra.
- Chuỗi S có xác suất lớn nhất, ký hiệu max Pr (S | O) (6)

Theo công thức Bayes: Pr (S | O) = Pr (O | S) * Pr (S) / Pr (O) (7)

- Chuỗi S có xác suất lớn nhất khi: Pr (O | S) * Pr (S) lớn nhất, vì Pr (O) là xác suất của các tín hiệu quan sát được (đã biết).
- Xác định từ loại (Part_ of_ speech: POS) được thực hiện như sau:

Tập các ký hiệu quan sát là các từ của câu. Trạng thái là các từ loại của các từ trong câu. Nhiệm vụ xác định từ loại là tìm xác suất lớn nhất của chuỗi từ trong câu ở các trạng thái (từ loại) tương ứng.

Áp dụng công thức (7) vào việc xác định từ loại:

Pr (S | O) = Pr (O | S) * Pr (S) / Pr (O)

Suy từ (7): max Pr (S | O) = max (Pr (O | S) * Pr (S)) (8)

Với O là câu có T từ; $W_1, W_2, \ldots W_T$ là các từ của O,

S là chuỗi từ loại $C_1, \ldots, C_T$ tương ứng với $W_1, \ldots, W_T$.

Thay (8) bằng $\max P = \max \Pr(C_1C_2 \ldots C_T \mid W_1W_2 \ldots W_T)$

$$= \max (\Pr(W_1 \ldots W_T \mid C_1 \ldots C_T) * \Pr(C_1 \ldots C_T) \quad (9)$$

Xác định từ loại là tìm xác suất P ở (9) lớn nhất.

Không thể tính chính xác biểu thức (9), bởi vì không có không gian mẫu đủ lớn (tất cả các câu trong quá khứ, hiện tại và tương lai), tuy nhiên có thể tính gần đúng *bằng giả thiết độc lập và phụ thuộc xác suất*. Các giả thiết này tuy chưa thực sự hợp lý, nhưng vẫn cho kết quả khá tốt trong thực tế (95 % - 97%) [4].

- Với thành phần thứ hai trong biểu thức (9): $\Pr(C_1, \ldots, C_T)$, có thể tính gần đúng bởi giả thiết rằng, *xác suất của một từ loại chỉ phụ thuộc vào một hay hai từ loại đứng trước nó*. Xác suất *Bigram* là xác suất có điều kiện khi sự xuất hiện từ loại $C_i$ chỉ phụ thuộc vào từ loại $C_{i-1}$, được ký hiệu là $Pr(C_i \mid C_{i-1})$. Tương tự, xác suất *Trigram* được ký hiệu là $Pr(C_i \mid C_{i-2}\, C_{i-1})$ là xác suất có điều kiện khi sự xuất hiện từ loại $C_i$ phụ thuộc vào sự xuất hiện của 2 từ loại đứng trước nó. *Để đơn giản, các tác giả bài báo chỉ sử dụng xác suất Bigram* (Bigram cho độ chính xác 96,28% so với 96,58% của Trigram trên dữ liệu Penn Treebank [2]).

Với giả thiết trên, có công thức:

$$Pr(C_1, \ldots, C_T) \cong \prod_{i=1}^{T} \Pr(Ci \mid Ci-1)$$

Ký hiệu: ART, V, N, P là các từ loại. Nếu chúng ta có chuỗi từ loại ART N V N tương ứng với $C_1\, C_2\, C_3\, C_4$ của một câu có các từ $W_1\, W_2\, W_3\, W_4$ nào đó, thì:

Pr (ART N V N) ≅ Pr (ART | <Start>) * Pr(N | ART)*Pr (V | N)*(N | V), với giả thiết từ loại đứng trước của $C_1$ là <Start> (được gọi là từ loại rỗng).

- Với thành phần thứ nhất trong biểu thức (9), $Pr(W_1, \ldots, W_T \mid C_1, \ldots, C_T)$, có thể tính gần đúng bởi giả thiết độc lập (*Sự xuất hiện của từ có từ loại là độc lập với từ đứng trước hoặc sau nó*). Nên công thức:

$$Pr(W_1, \ldots, W_T \mid C_1, \ldots, C_T) \cong \prod_{i=1}^{T} \Pr(Wi \mid Ci)$$

Cuối cùng công thức xấp xỉ là:

$$Pr(C_1, \ldots, C_T) * Pr(W_1, \ldots, W_T \mid C_1, \ldots, C_T) \cong \prod_{i=1}^{T} \Pr(Ci \mid Ci-1) * \Pr(Wi \mid Ci) \quad (10)$$

Như vậy xác định từ loại cho các từ trong câu là đi tìm Max của (10):

$$\text{Max} \prod_{i=1}^{T} \Pr(Ci \mid Ci-1) * \Pr(Wi \mid Ci) \quad (11)$$

*2.1.3. Giải thuật Viterbi [4]*

Cho một chuỗi các từ $W_1, \ldots, W_T$, từ loại $C_1, \ldots, C_N$ , xác suất $\Pr(W_i \mid C_i)$ và xác suất Bigram $\Pr(C_i \mid C_j)$, tìm chuỗi từ loại $C_1, \ldots, C_T$ phù hợp nhất cho chuỗi từ $W_1, \ldots, W_T$.

*Bước khởi tạo:*

for i = 1 to K do /*K là số lượng từ loại; <Start>: từ loại rỗng */

SeqScore(i,1) = $\Pr(C_1 \mid$ <Start>)* $\Pr(W_1 \mid C_i)$

BACKPTR(i,1) = 0;

*Bước lặp:*

for t = 2 to T do /* T là số lượng từ trong câu cho trước */

for i = 1 to K do

SeqScore (i,t) = Max (SeqScore (j,t -1)* $\Pr(C_i \mid C_j)$)* $\Pr(W_t \mid C_i)$, với j = 1,..K

BACKPTR(i,t) = Chỉ số j cho giá trị Max ở trên.

*Bước xác định chuỗi từ loại:*

C(T) = i là Max của SeqScore(i,t)

for i = T-1 to 1 do

C(i) = BACKPTR(C(i+1),i+1)

### 2.2. Áp dụng kỹ thuật pruning vào giải thuật Viterbi

Thời gian chạy chương trình phụ thuộc vào nhiều yếu tố như: chất lượng mã của chương trình dịch, trạng thái và tốc độ các lệnh máy, dữ liệu cho chương trình, độ phức tạp thời gian của giải thuật. Trong mục này, bài báo sẽ trình bày phân tích *dữ liệu và giải thuật* để tìm ra cách *pruning* phù hợp nhằm cải thiện thời gian thực thi của giải thuật Viterbi.

*2.2.1. Phân tích dữ liệu*

Bài báo sử dụng một phần bộ sưu tập dữ liệu khoảng 1.000.000 từ đã gắn từ loại của dự án Penn TreeBank. Tập từ loại chứa 46 từ loại khác nhau. Với 36,5% số từ trong sưu tập là có nhiều hơn một từ loại và trung bình là 2,44 từ loại/từ. Tính tổng thể là 1,47 từ loại/từ cho 1.000.000 từ [3]. Với dữ liệu như trên thì đa số các từ có một từ loại (chiếm tới 63,5%).

1.000.000 lượt từ trong không gian xác suất là không nhiều. Để khắc phục tình trạng dữ liệu thưa trong thực nghiệm, có thể tính xác suất theo kỹ thuật ELE (Expected Likelihood Estimator) [4, tr.195] nghĩa là cộng thêm 0,5 cho những từ có từ loại bằng 0 trước khi tính xác suất $Pr(W_t \mid C_j)$, và cho $Pr(C_i \mid C_j) = 10^{-6}$ với những xác suất $Pr(C_i \mid C_j) = 0$.

*2.2.2. Phân tích giải thuật*

Qua xem xét giải thuật Viterbi nguyên thủy, nhận thấy rằng ở mỗi bước t (t = 2,...T), giải thuật cần tính xác suất của mỗi từ loại để lấy ra tích số lớn nhất.

$SeqScore(i,t) = Max(SeqScore(j,t-1) * Pr(C_i \mid C_j)) * Pr(W_t \mid C_i)$, *với* $j = 1..K$.

Ký hiệu: $P = Max(SeqScore(j,t-1) * Pr(C_i \mid C_j))$, *với* $j = 1..K$.

Kết quả của biểu thức P sẽ phụ thuộc vào hai thành phần SeqScore(j,t -1) và $Pr(C_i \mid C_j)$, có nghĩa là cần *tìm kiếm hai số SeqScore(j,t -1) và* $Pr(C_i \mid C_j)$ *để tích SeqScore(j,t -1)** $Pr(C_i \mid C_j)$ là lớn nhất. Giải thuật Breadth–First–Search [7] với *heuristic* hợp lý sẽ cho kết quả tìm kiếm nhanh nhất cho thành phần SeqScore(j,t -1). Nhưng vấn đề ở đây là thành phần $Pr(C_i \mid C_j)$ chưa hẳn đã đủ lớn để tạo nên kết quả *SeqScore(j,t -1)** $Pr(C_i \mid C_j)$ là lớn nhất.

*2.2.3. Hệ số cầm canh*

Tuy không biết trước được kết quả *SeqScore(j,t -1)** $Pr(C_i \mid C_j)$ khi nào là lớn nhất, nhưng dựa trên đặc tính của dữ liệu và quá trình xử lý có thể biết được khi nào có thể dừng quá trình tính giá trị P. Thông số để kiểm soát quá trình này được gọi là *hệ số cầm canh*. Giả thiết rằng: tại mỗi bước tính toán *t* luôn tạo ra được kết quả:

*SeqScore(j,t -1) >= SeqScore(j+1,t -1). (Bằng cách sắp xếp)*

Gọi $Pr_{max}(C_i \mid C_j)$ là giá trị lớn nhất trong các $Pr(C_i \mid C_j)$, *với* $j,i = 1,..K$.

Xét biểu thức: $P = Max\ (SeqScore(j,t-1) * Pr(C_i \mid C_j))$, *với* $j = 1,..K$. Nhận thấy: nếu ở bước thứ *j* nào đó mà biết được $SeqScore(j,t-1) * Pr(C_i \mid C_j) >= SeqScore(j+1,t-1) * Pr_{max}(C_i \mid C_j)$, thì có thể kết thúc việc tính P vì $SeqScore(r,t-1) < SeqScore(j+1,t-1)$, với r = j+2...K (*theo giả thiết*). Để giảm số phép tính nhân trong giải thuật, có thể biến đổi bất đẳng thức:

$SeqScore(j,t-1) * Pr(C_i \mid C_j) >= SeqScore(j+1,t-1) * Pr_{max}(C_i \mid C_j)$ (12)

thành: $SeqScore(j,t-1) * Pr(C_i \mid C_j) / Pr_{max}(C_i \mid C_j) >= SeqScore(j+1,t-1)$ (13)

Đặt $dBeam = Pr(C_i \mid C_j) / Pr_{max}(C_i \mid C_j)$, (14)

*Beam* có thể tính trước khi chạy chương trình và hệ số này chúng tôi gọi là *hệ số cầm canh*. Ngoài ra chúng tôi thêm một biến để xếp hạng cho các cặp $Pr(C_i \mid C_j)$ gọi là *iBack*.

*2.2.4.Giải thuật Viterbi sử dụng kỹ thuật pruning*

1) Giai đoạn 1: Chuẩn bị dữ liệu

– Sắp xếp bảng xác suất cặp $Pr(W_t \mid C_i)$ theo thứ tự giá trị giảm dần, công việc này giúp cho việc tạo *heap* nhanh hơn, vì không cần thêm thao tác đổi chỗ (theo chiến lược tối ưu hóa từng phần của giải thuật [7]).

– *Tính hạng* và *hệ số cầm canh* cho các thành phần trong bảng $Pr(C_i \mid C_j)$.

2) Giai đoạn 2: Phần sử dụng kỹ thuật pruning trong *giai đoạn lặp* của giải thuật Viterbi.

```
(i)    for t = 2 to T do
(ii)       for i = 1 to K do
                    begin
           (1)        Chuyển SeqScore(t –1,K-i) thành một heap; //O(logK)
           (2)      PMax = SeqScore(i,t -1)* Pr(Ci | C1);
           (3)      max=1;
           (4)      if ((hạng củaPr(Ci | C1)>1) and
                    (SeqScore (0,t -1)* Pr(Ci | C0).dBeam < SeqScore(1,t-1)))
                       then
(iii)      for j=2 to K do
                    begin
           (5)      if (SeqScore(j,t -1)* Pr(Ci | Cj).dBeam >= SeqScore (j+1,t -1))
                    then
                    thoát khỏi vòng for có  biến j;
           (6)      A = SeqScore(j,t -1)* Pr(Ci | Cj);
           (7)           if (A>PMax) then
                            begin
           (8)              PMax=A;
           (9)              max = j;
                            end;
                    end;
           (10)          SeqScore(i,t) = PMax* Pr(Wt | Ci);
           (11)          BackPtr(i,t) = max;
                    end;
```

3) Nhận xét: Thời gian chạy chương trình chỉ có thể khác nhau trong giai đoạn lặp của giải thuật. Giai đoạn lặp của cả hai giải thuật (Viterbi nguyên thủy và Viterbi sử dụng kỹ thuật *pruning*) khác nhau ở một số lần thực hiện câu lệnh *If* hoặc không có *If*. Ngoài ra trong giải thuật sử dụng kỹ thuật *pruning* có kết hợp với giải thuật tạo *Heap* để lấy phần tử lớn nhất trong dãy *SeqScore(t,K)* sao cho nhanh nhất.

Dùng phương pháp thống kê để so sánh thời gian thực thi (thông qua việc đếm số câu lệnh) giữa giải thuật Viterbi nguyên thủy và giải thuật Viterbi có sử dụng kỹ thuật pruning. Trong giải thuật sử dụng kỹ thuật *pruning*, đếm tất cả các câu lệnh từ 1 đến 11 khi thực thi. Ở giải thuật Viterbi nguyên thủy loại bỏ các câu lệnh 1,4,5. Kết quả được thống kê ở mục "Đánh giá kết quả".

## 3. ĐÁNH GIÁ KẾT QUẢ

Lấy ngẫu nhiên 3000 câu có chiều dài khác nhau trong bộ sưu tập tạp chí "Phố Uôn" và gắn các biến để đếm số lần thực hiện các *câu lệnh gán* và *câu lệnh if* trong giải thuật khi kết hợp các tổ hợp hạng (*iBack*) và hệ số cầm canh (*dBeam*). Giả thiết rằng thời gian thực thi một phát biểu gán, phát biểu if và tạo heap là như nhau.

Đặt trường hợp: A: Có iBack và dBeam,
B: Không có iBack, có dBeam,
C: Có iBack, không có dBeam,
D: Không có iBack và không có dBeam,
E: Số lần thực hiện tạo Heap. ( *T*K*logK:* )

Kết quả so sánh các trường hợp thể hiện ở H7.

| Stt | Số từ trong câu | Số lượng câu | T*K*logK (E) (Heap) | Kết quả | | | | Tỷ số D/(A+E) |
|---|---|---|---|---|---|---|---|---|
| | | | | A | B | C | D | |
| 1 | 3 | 250 | 189750 | 285241 | 1140456 | 398843 | 2071326 | 4,36 |
| 2 | 5 | 250 | 316250 | 676012 | 2159305 | 1153451 | 4168350 | 4,20 |
| 3 | 8 | 250 | 506000 | 1063105 | 3534163 | 1993250 | 7263122 | 4,63 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | 10 | 250 | 632500 | 2743247 | 5042016 | 5077515 | 10468054 | 3,10 |
| 5 | 16 | 250 | 1012000 | 2704276 | 7120150 | 5124199 | 15570318 | 4,19 |
| 6 | 20 | 350 | 1771007 | 4545821 | 12835032 | 80905650 | 27589426 | 4,37 |
| 7 | 24 | 250 | 1518000 | 4287091 | 11584866 | 7862927 | 24988512 | 4,30 |
| 8 | 32 | 250 | 2024000 | 5352826 | 14153251 | 13044250 | 32290542 | 4,39 |
| 9 | 38 | 250 | 2403500 | 11679096 | 19414311 | 21777512 | 37841566 | 2,68 |
| 10 | 40 | 250 | 2530000 | 8336102 | 19731752 | 16142521 | 40692570 | 3,75 |
| 11 | 53 | 250 | 3352250 | 9784750 | 25789183 | 18259751 | 57221084 | 4,36 |
| 12 | 62 | 150 | 2352975 | 7537491 | 17847771 | 14254069 | 38719195 | 3,90 |
| ***Tổng số:*** | | **3000** | **18608232** | **58995058** | | | **298884065** | ***3,85*** |

**H 7: Bảng thống kê số câu lệnh cần thực thi**

Với giải thuật Viterbi nguyên thủy thì độ phức tạp của giải thuật là $T_{origin} = O(T*K*K)$.

Ở đây:

$Tg_{origin}$ : thời gian chạy chương trình của giải thuật Viterbi nguyên thủy,

$Tg_{pruning}$: thời gian thực thi chương trình theo giải thuật Viterbi với kỹ thuật *pruning*

T: là số từ trong câu, K: số từ loại.

Dữ liệu của Penn TreeBank: K = 46.

$Tg_{origin} / Tg_{pruning} = \underline{3,85}$

Đóng góp ban đầu của bài báo này là đã đưa ra được công thức tính ngưỡng *beam*. Ngưỡng này được tính dựa trên cả hai thành phần xác suất $\delta_{t-1}(j) * P(C_i \mid C_j)$ thay vì phải chọn các mức ngưỡng từ thực nghiệm thông qua $\delta_{t-1}(j)$. Tuy nhiên để có những khảo sát đầy đủ về hiệu quả và tốc độ trong tính toán, bài báo cần thực nghiệm tiếp, khi lựa chọn được các ngân hàng dữ liệu về gắn nhãn từ loại (*TreeBank*) khác thích hợp trong tương lai (ngoài ngôn ngữ tiếng Anh).

## THE USE OF PRUNING TECHNIQUES FOR THE PART-OF-SPEECH TAGGING PROBLEM

**Nguyen Chi Hieu[(1)], Phan Thi Tuoi[(2)], Nguyen Xuan Dung[(3)], Nguyen Quang Chau[(1)]**

(1) Ho Chi Minh University of Industry

(2) University of Technology, Vietnam National University of HCM City

(3) Ho Chi Minh City Institute of Information Technology

***ABSTRACT:*** The exact and automatic tagging the words in the texts of natural language is a very important problem. It will support a parsing for the texts, contribute to solve the polysemantic of the word, and help to access a semantic information, etc. Therefore, time tagging words in a text is always an essential problem, especially in the large document. This paper will present the pruning techniques for tagging the texts of a natural language – the Viterbi algorithm [4]. The Viterbi pruning algorithm was tested with 1.000.000 words on the tag of the Wall Street Journal corpus [8].

## TÀI LIỆU THAM KHẢO

[1]. Christoph Tillmann, Hermann Ney, *Word Reordering and a Dynamic Programming Beam Search Algorithm for Statistical Machine Translation,* IBM T. J. Watson Research Center, 2003.

[2]. Ferran Pla And Antonio Molina, *Part-of-Speech Tagging with Lexicalized HMM*,{fpla,amolina}@dsic.upv.es, 2001.

[3]. Ferran Pla, Antonio Molina And Natividad Prieto, *Tagging and Chunking with Bigram*, {fpla,amolina,nprieto}@dsic.upv.es, 2000.

[4]. James Allen, *Natural Language Understanding*, The Benjamin/Cummings Publishing Company, Inc, năm 1995.

[5]. Nguyễn Duy Tiến, Vũ Việt Yên, *Lý thuyết xác suất,* NXB Giáo dục, 2003.

[6]. Ruslan Mitkov, *Computational Linguistics*, The Oxford University Press, First Published, 2003.

[7]. Thomas H. Cormen, Charles E. Leiserson & Ronald L. Rivest, *Introduction to Algorithms*, The MIT Press, năm 1990.

[8]. The Penn TreeBank Project - ftp.cis.upenn.edu/pub/treebank/doc/update.cd2

[9]. Padro, *A hybrid environment for syntax-semantic tagging*. Ph.D. thesis, Departament de Llenguatges i Sistemes Informatics, Universitat Politecnica de Catalunya, Barcelona, 1997.

[10]. http://www.cs.toronto.edu/~gpenn/csc2518/lecture8.pdf

[11]. www.cs.rochester.edu/u/james/CSC248/Lec9.pdf