**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**ĐỒ ÁN MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

**--- LỚP CS221.N11 ---**

# ĐỀ TÀI: GÁN NHÃN TỪ LOẠI TIẾNG VIỆT SỬ DỤNG MÔ HÌNH HIDDEN MARKORV VÀ THUẬT TOÁN VITERBI

**GIẢNG VIÊN HƯỚNG DẪN**

**Th.S NGUYỄN TRỌNG CHỈNH**

**DANH SÁCH THÀNH VIÊN**

| | |
|---|---|
| 1. Nguyễn Đức Phương Nguyên | MSSV: 19521913 |
| 2. Nguyễn Ngọc Thái Nguyên | MSSV: 19521917 |
| 3. Lê Quang Minh | MSSV: 19521836 |

**TP. HỒ CHÍ MINH, NĂM 2023**

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

*……, ngày … tháng 2 năm 2023*

**Người nhận xét**

*(Ký tên và ghi rõ họ tên)*

# MỤC LỤC

Chương 1. **GIỚI THIỆU** ....4
Chương 2. **THU THẬP DỮ LIỆU** ....5
Chương 3. **PHƯƠNG PHÁP TÁCH TỪ** ....5
3.1. Thuật toán Longest Matching ....5
3.2. Thư viện VnCoreNLP ....6
3.3. Triển khai....6
3.4. Đánh giá kết quả tách từ....8
Chương 4. **TẠO NGỮ LIỆU VÀ GÁN NHÃN** ....9
4.1. Tạo ngữ liệu....9
4.2. Gán nhãn từ loại ....10
Chương 5. **HIDDEN MARKOV VÀ VITERBI** ....11
5.1. Mô hình Hidden Markov....11
5.2. Thuật toán Viterbi ....13
Chương 6. **KẾT QUẢ VÀ ĐÁNH GIÁ** ....17
6.1. Kết quả thực nghiệm ....17
6.2. Nhận xét và hướng phát triển ....18
Chương 7. **KẾT LUẬN** ....18
7.1. Tỉ lệ đóng góp....19
**TÀI LIỆU THAM KHẢO** ....20

# Chương 1. **GIỚI THIỆU**

Gán nhãn từ loại (POS) có lẽ là bài toán cơ bản và được mọi người biết đến nhiên nhất khi nhập môn Xử lý Ngôn ngữ Tự nhiên. Trong báo cáo này, nhóm sẽ tìm hiểu về bài toán gán nhãn từ loại thông qua các hướng tiếp cận cơ bản để giải quyết bài toán này.

Gán nhãn từ loại là gì? Là việc xác định các chức năng ngữ pháp của từ ngữ trong câu. Đây là bước cơ bản khi phân tích sâu văn phạm để hiểu rõ ý nghĩa của câu và để giải quyết các vấn đề xử lí ngôn ngữ phức tạp khác như truy vẫn tìm kiếm, nhận dạng giọng nói,...

Bài toán gán nhãn từ loại có thể chia ra làm 2 bài toán nhỏ đó chính là: bài toán tách từ và bài toán gán nhãn. Bài toán tách từ là một bài toán quan trọng đối với tiếng Việt. Khác với tiếng Anh, một từ tiếng Việt có thể được tạo bởi nhiều hơn một âm và mục tiêu của bài toán là xác định ranh giới của các từ trong câu. Ví dụ: từ câu "Con người chỉ đơn giản là những cỗ máy vô cùng phức tạp" sau khi thực hiện tách từ ta sẽ có được: "Con_người chỉ đơn_giản là những cỗ_máy vô_cùng phức_tạp" với mỗi từ cách nhau bởi dấu cách duy nhất. Đây là bước quan trọng chúng ta cần thực hiện trước khi đưa dữ liệu vào các bước tiếp theo. Bài toán gán nhãn phân biệt các từ loại của từ trong câu để giúp ta hiểu rõ hơn ý nghĩa của câu. Việc xác định danh từ riêng, tổ chức, ký hiệu cổ phiếu hoặc bất cứ thứ gì tương tự sẽ cải thiện đáng kể mọi thứ, từ nhận dạng giọng nói đến tìm kiếm. Ví dụ với câu đã được tách từ ở bước trước sau khi gán nhãn ta có được: "Con_người/N chỉ/R đơn_giản/A là/V những/D cỗ_máy/N vô_cùng/R phức_tạp/A".

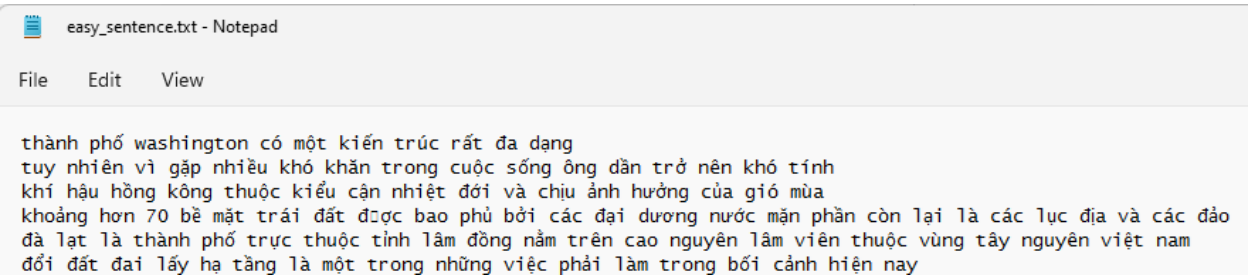
Trong báo cáo lần này nhóm sẽ sử dụng thuật toán Longest Matching để tách từ và dùng HMM kết hợp Viterbi để gán nhãn từ loại tiếng Việt và sau đó sẽ tiến hành so sánh độ chính xác với thư viện VnCoreNLP

# Chương 2. **THU THẬP DỮ LIỆU**

Dữ liệu được nhóm thu thập nhiều nguồn từ https://tuoitre.vn, https://tinhte.vn, https://kenh14.vn, https://www.wikipedia.org và các câu tự tạo, bao gồm các câu thuộc đa dạng chủ đề khác nhau.

Bao gồm 62 câu:

- Mỗi dòng là 1 câu.
- Các từ được phân cách nhau bằng khoảng trắng " ".
- Số từ nhiều nhất trong 1 câu là 32 từ.
- Số từ ít nhất trong 1 câu là 5 từ.




*Hình 1. Một số câu được thu thập trong bộ ngữ liệu*

# Chương 3. **PHƯƠNG PHÁP TÁCH TỪ**

## 3.1. Thuật toán Longest Matching

Ý tưởng chính của phương pháp này là dựa trên một bộ vocab (từ điển) sẵn có, xét các tiếng trong văn bản từ trái qua phải và thực hiện so khớp với các từ có trong vocab.

*Điểm yếu của phương pháp:* độ chính xác của phương pháp này phụ thuộc rất lớn vào kích thước của bộ vocab. Do sự phụ thuộc nên phương pháp này khó có thể xử lý được các từ mới không xuất hiện trong vocab

*Điểm mạnh của phương pháp:* Do không cần training nên thời gian xử lý tương đối nhanh, đơn giản và dễ hiểu.

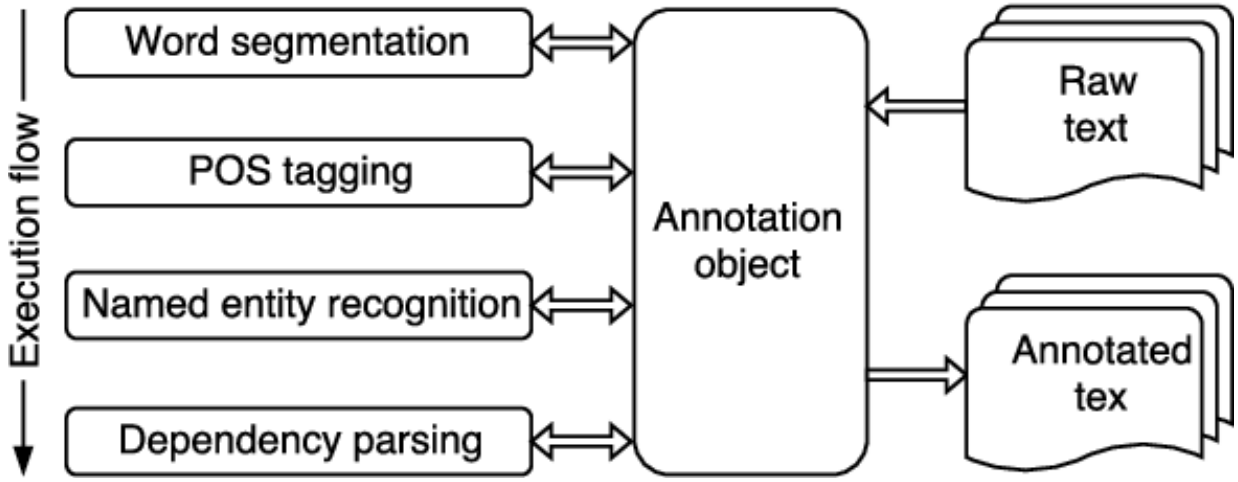
```
current_word _id = 0 # Đặt từ hiện tại ở đầu câu
while (curret_word_id+1) != len(text) {
  # xét từ hiện tại với 2 từ sau nó trong vocab tri_gram
  check if {“word, word+1, word+2”} is in tri_gram
  if true:
    take {word, word1, word2; current_word_id += 3} # 3 tiếng đó tạo thành từ ghép
  # xét từ hiện tại với tiếng sau trong vocab trong vocab bi_gram
  else check if: take {“word, word+1”} is in bi_gram
  if true:
    take {word, word1; current_word_id += 2} # 2 tiếng đó tạo thành từ ghép
  else => {take word; current_word_id += 1} # tiếng đó là từ đơn
}
```

*Hình 2. Mã giả thuật toán Longest Matching*

### 3.2. Thư viện VnCoreNLP

Vietnamese Natural Language Processing Toolkit (VnCoreNLP) là một tool dùng để hỗ trợ xử lý ngôn ngữ tự nhiên nhanh và chính xác dành cho tiếng Việt.

Có các nhiệm vụ chính như word segmentation, POS tagging, named entity recognition (NER), dependency parsing




*Hình 3 Mô hình hoạt động của VnCoreNLP*

### 3.3. Triển khai

Nhóm sẽ thực hiện tách từ bán thủ công: sử dụng thư viện VnCoreNLP để tách từ trước sau đó sẽ phân công các thành viên kiểm tra thủ công lại kết quả tách từ và sửa lại các từ bị sai để thu được 1 bộ dữ liệu mới chặt chẽ hơn.

Bộ Vocab được xây dựng từ nội dung của 1000 bài báo ở https://kenh14.vn và sử dụng thư viện VnCoreNLP để tách từ sau đó sẽ được chia thành 3 bộ Vocab:

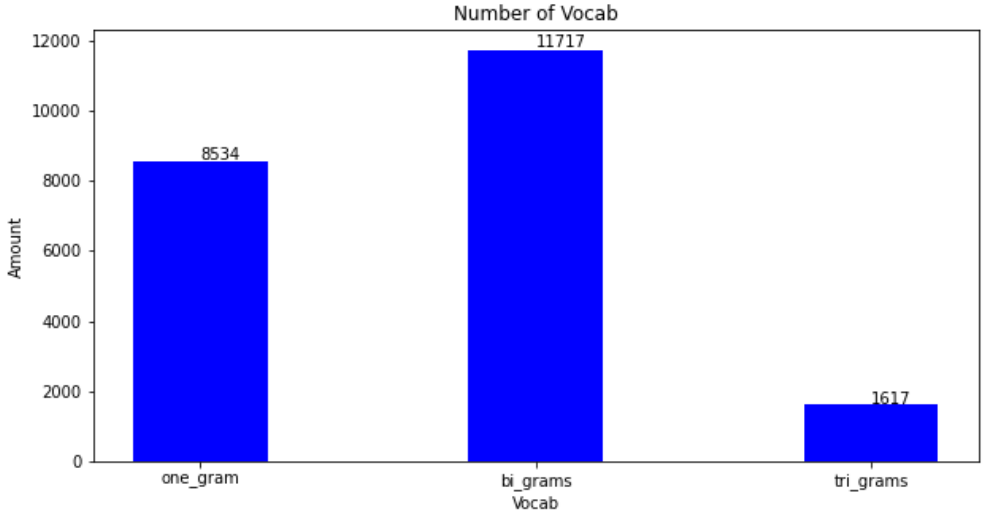
Vocab one gram: Bộ vocab chỉ chứa những từ đơn

Vocab bi grams: Bộ vocab chứa từ 2 âm

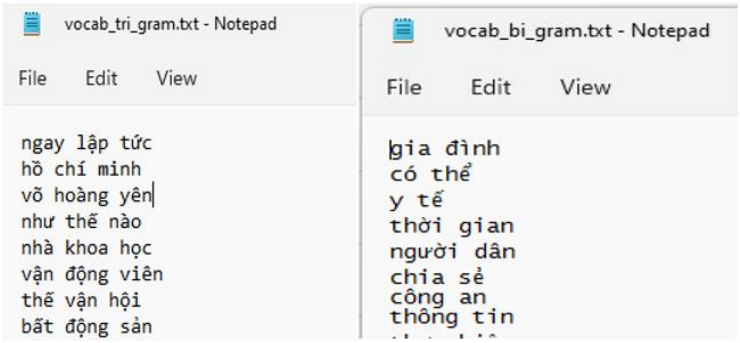
Vocab tri grams: Bộ vocab chứa từ 3 âm

Chi tiết về Vocab gồm:

- 8534 từ đơn (one_gram)
- 11717 từ kép (bi_grams)
- Và 1617 từ 3 âm (tri_grams)




*Hình 4. Phân bố n-gram*




*Hình 5. Một số mẫu tri_gram và bi_gram trong bộ dữ liệu vocab*

Nhóm sử dụng 3 bộ vocab one gram, bi grams, tri grams để thực hiện tách từ bằng thuật toán LongestMatching.

### 3.4. Đánh giá kết quả tách từ

Về bộ ngữ liệu dành cho việc đánh giá tách từ: nhóm sử dụng 62 câu đã thu thập và chia nhau tách từ thủ công và sau đó từng thành viên nhóm sẽ review chéo lại cho nhau để đảm bảo độ chính xác. Sau đó, nhóm sẽ sử dụng bộ dữ liệu này làm chuẩn và đánh giá kết quả tách từ của 2 phương pháp: sử dụng thuật toán Longest Matching và sử dụng thư viện VnCoreNLP.

- Số lượng từ ghép khi tách thủ công : 255
- Số lượng từ ghép khi tách từ bằng thuật toán Longest Matching: 224
- Số lượng từ ghép khi tách từ bằng thư viện VnCoreNLP: 231

| | Longest Matching | VnCoreNLP |
|---|---|---|
| **Accuracy** | 0.904762 | 0.914286 |
| **Precision** | 0.950893 | 0.943723 |
| **Recall** | 0.832031 | 0.851562 |
| **F1** | 0.8875 | 0.895277 |
| **True Positive** | 213 | 218 |
| **False Positive** | 11 | 13 |
| **Total True** | 665 | 672 |
| **Total Errors** | 99 | 81 |

*Hình 6. Kết quả đánh giá phương pháp tách từ Longest Matching và VnCoreNLP*

Ta có thể thấy rằng số liệu đánh giá của việc tách từ khi sử dụng thư viện VnCoreNLP tốt hơn với khi dùng thuật toán Longest Matching. Recall của thuật toán Longest Matching thấp hơn.

Kết quả này có được là do ở đây nhóm sử dụng bộ vocab với kích thước khá lớn và có thể đã bao trùm được những trường hợp trên tập test dẫn đến thuật toán Longest Matching cũng có độ chính xác không thua kém nhiều hơn so với VnCoreNLP.

# Chương 4. **TẠO NGỮ LIỆU VÀ GÁN NHÃN**

## 4.1. Tạo ngữ liệu

Để tạo ngữ liệu nhóm quyết định sử dụng bộ ngữ liệu đã được tách từ thủ công để giúp kết quả cho ra là chính xác nhất. Sau đó phân công từng thành viên gán nhãn thủ công theo bộ quy tắc của VLSP và đối chiếu lại với nhau để đảm bảo tránh sai sót.

| 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 |

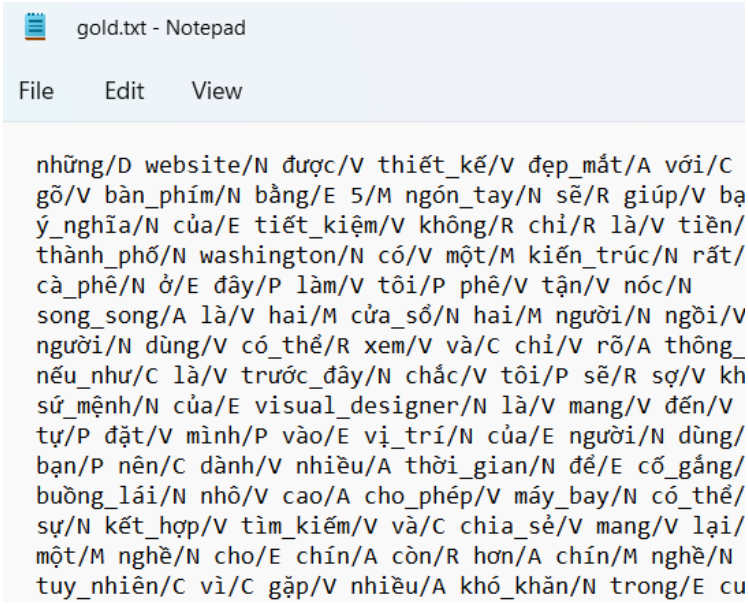
*Bảng 1. Danh sách các từ loại (Nguồn:https://vlsp.hpda.vn/demo/vcl/PoSTag.htm)*

Nhóm có được bộ dữ liệu Gold gồm 62 câu:

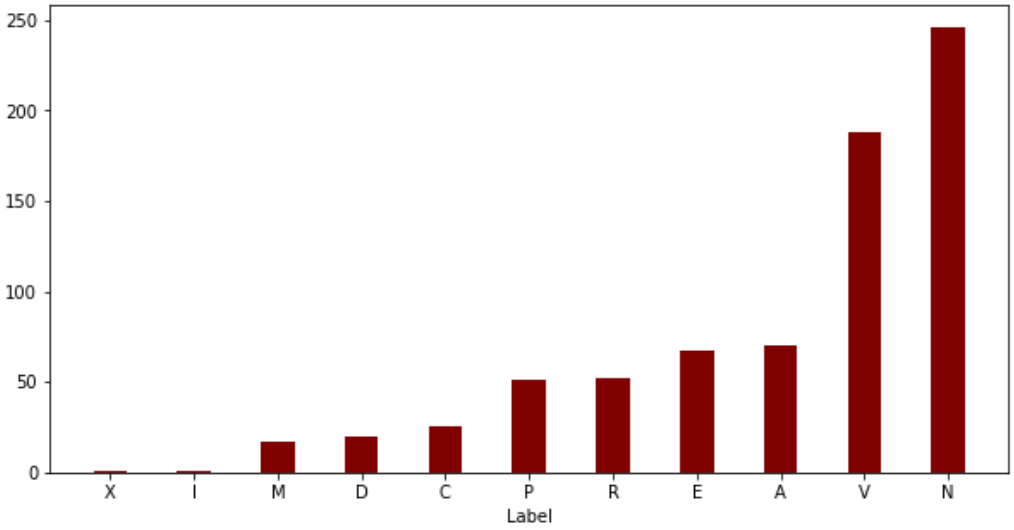
- Tổng số lượng nhãn là: 749 nhãn.
- Trung bình khoảng 12 nhãn mỗi câu.
- Câu nhiều nhãn nhất: 22 nhãn.
- Câu ít nhãn nhất: 4 nhãn.

Nhóm tiếp tục chia tập Gold thành 2 tập data là tập train và tập test:

- Tập train: 49 câu.
- Tập test: 13 câu.
- Số lượng nhãn tập train: 607 nhãn.
- Số lượng nhãn tập test: 142 nhãn.




*Hình 7. Bộ dữ liệu 62 câu sau khi gán nhãn thủ công.*




*Hình 8. Phân bố các nhãn trong bộ dữ liệu Gold*

### 4.2. Gán nhãn từ loại

Để xây dựng mô hình HMM gán nhãn từ loại, từ bộ ngữ liệu đã có ta còn phải thực hiện một số bước tiền xử lý và tính toán trước khi đưa vào quá trình training.

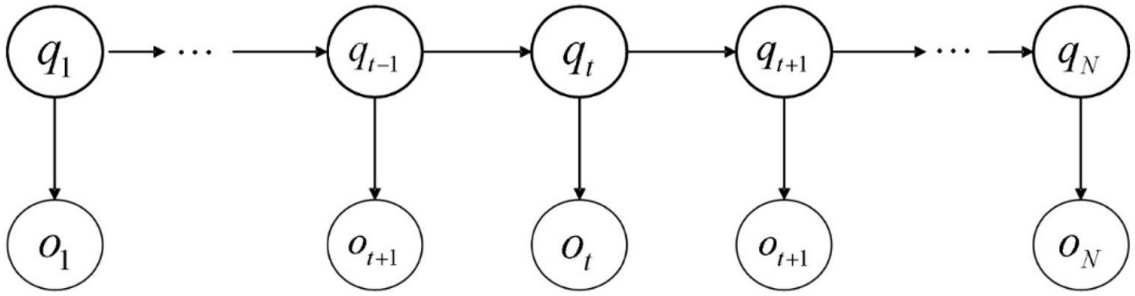
Nhóm thực hiện thêm ký tự bắt đầu (<s>) vào trước mỗi câu để đánh dấu vị trí bắt đầu một câu mới giúp khi thực hiện training chính xác hơn.Ngoài ra nhóm thực hiện tính toán tổng hợp số lượng của từng nhãn, số lượng cặp nhãn liền kề nhau để thuận tiện cho các bước sau.

# Chương 5. **HIDDEN MARKOV VÀ VITERBI**

## 5.1. Mô hình Hidden Markov

HMM (Hidden Markov Model) là một trong những thuật toán được sử dụng phổ biến nhất trong Xử lý ngôn ngữ tự nhiên và là nền tảng cho nhiều kỹ thuật học sâu. Ngoài gán nhãn từ loại, HMM còn được dùng để nhận dạng giọng nói , tổng hợp giọng nói,...

HMM là một mô hình thống kê có chứa các tham số quan sát được và các tham số ẩn chưa biết dựa trên Markov chain. HMM có thể được biểu diễn dưới dạng đồ thị chuyển trạng thái.




Hình 9. Markov Chain với $q_i$ là các trạng thái ẩn, $o_i$ là các trạng thái quan sát được

HMM tổng quát bậc n, giả định trạng thái $q_t$ phụ thuộc vào n trạng thái đứng trước nó. Trong bài làm nhóm chỉ xét bậc 1, tức trạng thái $q_t$ phụ thuộc vào trạng thái $q_{t-1}$ và độc lập với các trạng thái khác:

$$P(q_t|q_{t-1}, q_{t-2}, \dots) = P(q_t|q_{t-1})$$

$$P(q_t, q_{t-1}, q_{t-2}, \dots) = P(q_t|q_{t-1})P(q_{t-1}|q_{t-2}) \dots$$

Tổng quát bài toán gán nhãn từ loại với mô hình HMM:

- Một chuỗi các từ đã được tách $S = \{w_1, w_2, \dots w_n\}$ là các giá trị quan sát
- Tập hữu hạn các nhãn từ loại $T = \{T_1, T_2, \dots T_m\}$ là các trạng thái ẩn
- Ta cần xác định nhãn từ loại $t_i$ tương ứng với mỗi từ $w_i$ ($t_i \in T$) để thu được $T' = \{t_1, t_2, \dots t_n\}$ để $P(T'|S)$ lớn nhất

$$T' = argmax\ P(T|S)$$

$$T' = argmax\ P(w_1, w_2, \dots w_n|t_1, t_2, \dots t_n)P(t_1, t_2, \dots t_n) \qquad (1)$$

- Với Mô hình Hidden Markov bậc 1:

$$(1) \Leftrightarrow T' = argmax \prod_{i=1}^{n} P(w_i|t_i) \prod_{i=1}^{n} P(t_i|t_{i-1})$$

- Để thuận tiện cho việc tính $T'$, HMM sử dụng một ma trận chuyển tiếp A (Transition Matrix) và một ma trận phát xạ B (Emission Matrix) đại diện cho 2 xác suất ở công thức trên:
  - $A = \{a_{ij}\} (i, j = 1..m)$ là ma trận chuyển trạn thái, trong đó $a_{ij}$ là xác suất chuyển từ trạng thái nhãn $t_i$ sang trạng thái nhãn $t_j$.
  - $B = \{b_{ij}\} (i = 1..m, j = 1..n)$ là ma trận phát xạ, trong đó $b_{ij}$ là xác suất trạng thái ẩn nhãn $t_i$ thể hiện bằng giá trị quan sát $w_j$

### 5.1.1. Ma trận chuyển trạng thái A (Transition Matrix A)

Ma trận trận được smoothing với công thức:

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

Trong đó:

- $N$: tổng số lượng nhãn.
- $C(t_{i-1}, t_i)$: tổng số lượng cặp nhãn $(t_{i-1}, t_i)$
- $C(t_{i-1})$: tổng số lần xuất hiện nhãn $t_{i-1}$
- $\alpha$: tham số làm mịn (smoothing)

| | A | C | D | E | I | M | N | P | R | V |
|---|---|---|---|---|---|---|---|---|---|---|
| <s> | 0.040929 | 0.081653 | 0.081653 | 0.061291 | 0.000204 | 0.040929 | 0.407453 | 0.183466 | 0.061291 | 0.040929 |
| A | 0.056675 | 0.113161 | 0.037846 | 0.094333 | 0.000188 | 0.037846 | 0.150819 | 0.056675 | 0.056675 | 0.169648 |
| C | 0.095216 | 0.047845 | 0.047845 | 0.000474 | 0.047845 | 0.047845 | 0.189957 | 0.142586 | 0.047845 | 0.332070 |
| D | 0.000552 | 0.000552 | 0.000552 | 0.000552 | 0.000552 | 0.000552 | 0.994478 | 0.000552 | 0.000552 | 0.000552 |
| E | 0.038572 | 0.019382 | 0.076953 | 0.000192 | 0.000192 | 0.019382 | 0.556707 | 0.134523 | 0.000192 | 0.153713 |
| I | 0.004739 | 0.004739 | 0.004739 | 0.004739 | 0.004739 | 0.004739 | 0.004739 | 0.478673 | 0.004739 | 0.478673 |
| M | 0.000709 | 0.000709 | 0.000709 | 0.071580 | 0.000709 | 0.000709 | 0.922041 | 0.000709 | 0.000709 | 0.000709 |
| N | 0.103620 | 0.020765 | 0.005230 | 0.129512 | 0.000052 | 0.010409 | 0.258972 | 0.041479 | 0.062193 | 0.238258 |
| P | 0.025825 | 0.025825 | 0.000256 | 0.025825 | 0.000256 | 0.000256 | 0.025825 | 0.000256 | 0.307083 | 0.486065 |
| R | 0.204262 | 0.000227 | 0.000227 | 0.022897 | 0.000227 | 0.022897 | 0.045568 | 0.000227 | 0.068238 | 0.635003 |
| V | 0.091503 | 0.026190 | 0.039253 | 0.104565 | 0.006597 | 0.032722 | 0.313565 | 0.052315 | 0.065378 | 0.215597 |

*Hình 10. Ma trận chuyển trạng thái A sau khi tính toán và smoothing*

### 5.1.2. Ma trận phát xạ B (Emission Matrix B)

Ma trận trận được smoothing với công thức:

$$P(w_i|t_i) = \frac{C(t_i, w_i) + \alpha}{C(t_i) + \alpha * N}$$

Trong đó:

- $N$: tổng số lượng từ trong bộ từ điển.
- $C(w_i, t_i)$: số lượng từ $w_i$ đi với nhãn $t_i$
- $C(t_i)$: tổng số lần xuất hiện nhãn $t_i$
- $\alpha$: tham số làm mịn (smoothing)

| | bạn | nên | dành | nhiều | thời_gian | để | cố_gắng | phát_triển | kiến_thức |
|---|---|---|---|---|---|---|---|---|---|
| A | 0.000036 | 0.000036 | 0.000036 | 0.014373 | 0.000036 | 0.000036 | 0.000036 | 0.000036 | 0.000036 |
| C | 0.000040 | 0.004089 | 0.000040 | 0.000040 | 0.000040 | 0.000040 | 0.000040 | 0.000040 | 0.000040 |
| D | 0.000041 | 0.000041 | 0.000041 | 0.000041 | 0.000041 | 0.000041 | 0.000041 | 0.000041 | 0.000041 |
| E | 0.000036 | 0.000036 | 0.000036 | 0.000036 | 0.000036 | 0.028813 | 0.000036 | 0.000036 | 0.000036 |
| I | 0.000044 | 0.000044 | 0.000044 | 0.000044 | 0.000044 | 0.000044 | 0.000044 | 0.000044 | 0.000044 |
| M | 0.000042 | 0.000042 | 0.000042 | 0.000042 | 0.000042 | 0.000042 | 0.000042 | 0.000042 | 0.000042 |
| N | 0.002411 | 0.000024 | 0.000024 | 0.000024 | 0.004797 | 0.000024 | 0.000024 | 0.000024 | 0.004797 |
| P | 0.011358 | 0.000038 | 0.000038 | 0.000038 | 0.000038 | 0.000038 | 0.000038 | 0.000038 | 0.000038 |
| R | 0.000037 | 0.000037 | 0.000037 | 0.000037 | 0.000037 | 0.000037 | 0.000037 | 0.000037 | 0.000037 |
| V | 0.000026 | 0.000026 | 0.005303 | 0.000026 | 0.000026 | 0.000026 | 0.010580 | 0.005303 | 0.000026 |

*Hình 11. Ma trận phát xạ B sau khi tính toán và smoothing*

### 5.2. Thuật toán Viterbi

Thuật toán Viterbi nhóm thực hiện qua 3 bước chính:

- Khởi tạo: khởi tạo 2 ma trận best_prob và best_path.
- Forward: Ở mỗi bước, tính toán xác suất xảy ra ở các đường đi và đường đi tốt nhất tới điểm đó.
- Backward: Tìm ra đường đi tốt nhất với xác suất cao nhất.

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

create a path probability matrix *viterbi[N,T]*
**for** each state *s* **from** 1 **to** *N* **do** ; initialization step
*viterbi*[*s*,1] $\leftarrow \pi_s * b_s(o_1)$
*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] $\leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
*bestpathprob* $\leftarrow \max_{s=1}^{N} viterbi[s,T]$ ; termination step
*bestpathpointer* $\leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$ ; termination step
*bestpath* $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
**return** *bestpath*, *bestpathprob*

*Hình 12. Mã giả thuật toán Viterbi*

### 5.2.1. Khởi tạo (Initialization step)

Giai đoạn khởi tạo bao gồm các bước:

- Khởi tạo 2 ma trận có cùng chiều: (NxT):

  best_probs: Mỗi ô chứa xác suất đi từ một nhãn sang một từ.

  best_paths: Ma trận giúp tìm đường đi tốt nhất.

- Cả 2 ma trận sẽ được khởi tạo bằng 0.
- Cột 0 của best_prob sau đó sẽ được tính toán với giả định rằng từ đầu tiên của ngữ liệu luôn được đặt trước bởi một ký tự bắt đầu (<s>):
  - Với nhãn ở vị trí i:

$$best\_prob[i,0] = A[0,i] \times B[i, corpus[0]]$$

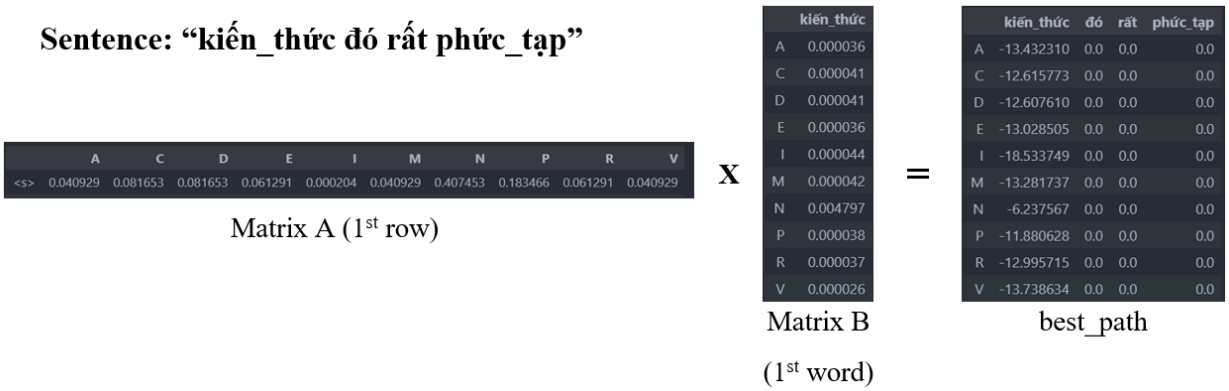
Trong đó:

$corpus$: ngữ liệu đầu vào để gán nhãn từ loại

$corpus[0]$: từ đầu tiên của ngữ liệu

- Tuy nhiên để tránh việc mất mát dữ liệu vì nhân 2 số quá nhỏ, nhóm thực hiện lấy log toàn bộ tích và biến đổi thành tổng 2 log:

$$best\_prob[i, 0] = log(A[0, i]) + log(B[i, corpus[0]])$$




*Hình 13. Thực hiện bước khởi tạo và hoàn thành xong cột đầu tiên của best_path*

### 5.2.2. Forward (Recursion step)

Ở giai đoạn Forward để tính best_prob, best_path ở các từ tiếp theo ta thực hiện theo 2 công thức đệ quy như sau:

best_prob: $\sigma_{i+1}(t_j) = \max_{1 \le k \le T}[\sigma_i(t_k) \times P(w_{i+1}|t_j) \times P(t_j|t_k)]$

best_path: $\psi_{i+1}(t_j) = \operatorname{argmax}_{1 \le k \le T}[\sigma_i(t_k) \times P(w_{i+1}|t_j) \times P(t_j|t_k)]$

Để hiểu rõ hơn ta có thể chia thành các bước:

- Với các từ tiếp theo $w_{i+1}$ $(corpus[i + 1])$ ta lặp qua tất cả các nhãn $t_j$ để tính xác suất:
  - Với mỗi nhãn trước đó $t_k$, tính xác suất:

    $$prob = best\_prob[k, i] \times A[k, i + 1] \times B[j, corpus[i + 1]]$$
  - Giá trị xác suất lớn nhất sẽ được giữ lại và giá trị này sẽ được đặt cho $best\_prob[j, i + 1]$
  - Giá trị k của xác suất lớn nhất sẽ được đặt cho $best\_path[j, i + 1]$

| | kiến_thức | đó | rất | phức_tạp |
|---|---|---|---|---|
| A | -13.432310 | -23.726003 | -30.470820 | -34.232251 |
| C | -12.615773 | -23.600122 | -31.952381 | -38.721491 |
| D | -12.607610 | -23.591959 | -33.323058 | -43.078430 |
| E | -13.028505 | -19.110882 | -29.835089 | -38.847371 |
| I | -18.533749 | -23.524136 | -34.248343 | -43.260626 |
| M | -13.281737 | -23.575430 | -32.618345 | -43.311919 |
| N | -6.237567 | -16.961773 | -27.685980 | -38.410186 |
| P | -11.880628 | -17.967410 | -31.334880 | -42.351448 |
| R | -12.995715 | -23.693213 | -24.954561 | -36.338792 |
| V | -13.738634 | -23.857735 | -29.944518 | -36.218681 |

best_prob

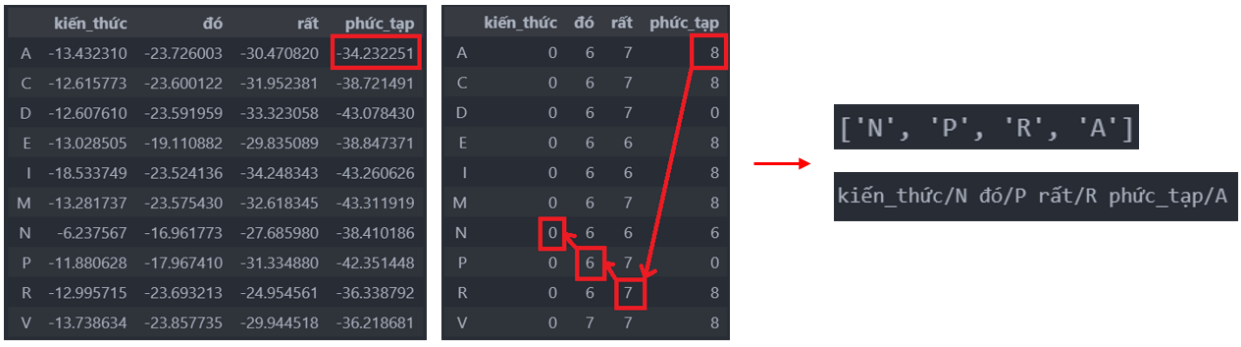
| | kiến_thức | đó | rất | phức_tạp |
|---|---|---|---|---|
| A | 0 | 6 | 7 | 8 |
| C | 0 | 6 | 7 | 8 |
| D | 0 | 6 | 7 | 0 |
| E | 0 | 6 | 6 | 8 |
| I | 0 | 6 | 6 | 8 |
| M | 0 | 6 | 7 | 8 |
| N | 0 | 6 | 6 | 6 |
| P | 0 | 6 | 7 | 0 |
| R | 0 | 6 | 7 | 8 |
| V | 0 | 7 | 7 | 8 |

best_path

*Hình 14. Hai ma trận best_prob và best_path sau bước Forward*

### 5.2.3. Backward (Termination step)

Ở bước Backward, ta bắt đầu tại cột cuối cùng của best_prob, chọn ra giá trị xác suất cao nhất và đối chiếu với best_path để tìm nhãn có nhiều khả năng nhất cho từ trước đó và cập nhật lại nhãn cho mỗi từ. Tìm các nhãn tốt nhất bằng cách đi lùi qua best_path từ từ cuối cùng đến từ đầu tiên trong ngữ liệu.




*Hình 15. Quá trình Backtrack và cho ra kết quả*

# Chương 6. **KẾT QUẢ VÀ ĐÁNH GIÁ**

## 6.1. Kết quả thực nghiệm

Nhóm đánh giá cả 2 phương pháp trên 2 tập dữ liệu xây dựng trước đó là tập train và tập test cho ra kết quả như sau:

Kết quả của HMM + Viterbi trên tập train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| A | 0.94 | 0.83 | 0.88 | 58 |
| C | 0.67 | 0.91 | 0.77 | 11 |
| D | 0.50 | 0.90 | 0.64 | 10 |
| E | 0.86 | 0.94 | 0.90 | 52 |
| I | 0.00 | 0.00 | 0.00 | 0 |
| M | 0.42 | 0.71 | 0.53 | 7 |
| N | 0.82 | 0.72 | 0.77 | 213 |
| P | 0.77 | 0.91 | 0.83 | 33 |
| R | 0.86 | 0.97 | 0.91 | 38 |
| V | 0.70 | 0.69 | 0.70 | 140 |
| accuracy | | | 0.78 | 562 |
| macro avg | 0.65 | 0.76 | 0.69 | 562 |
| weighted avg | 0.79 | 0.78 | 0.78 | 562 |

Kết quả của HMM + Viterbi trên tập test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| A | 0.47 | 0.50 | 0.48 | 16 |
| C | 0.56 | 1.00 | 0.71 | 5 |
| D | 1.00 | 0.75 | 0.86 | 4 |
| E | 0.67 | 1.00 | 0.80 | 8 |
| M | 0.40 | 0.40 | 0.40 | 5 |
| N | 0.43 | 0.57 | 0.49 | 40 |
| P | 0.62 | 0.73 | 0.67 | 11 |
| R | 0.77 | 0.83 | 0.80 | 12 |
| V | 0.67 | 0.44 | 0.53 | 73 |
| X | 0.00 | 0.00 | 0.00 | 0 |
| accuracy | | | 0.57 | 174 |
| macro avg | 0.56 | 0.62 | 0.57 | 174 |
| weighted avg | 0.60 | 0.57 | 0.57 | 174 |

*Hình 16. Kết quả của phương pháp HMM kết hợp Viterbi*

Kết quả của VnCoreNLP trên tập train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| A | 0.86 | 0.79 | 0.82 | 56 |
| C | 0.87 | 0.76 | 0.81 | 17 |
| D | 0.94 | 1.00 | 0.97 | 17 |
| E | 0.95 | 0.90 | 0.92 | 60 |
| I | 0.00 | 0.00 | 0.00 | 0 |
| M | 0.92 | 0.92 | 0.92 | 12 |
| N | 0.88 | 0.92 | 0.90 | 179 |
| P | 0.87 | 0.94 | 0.91 | 36 |
| R | 0.84 | 0.90 | 0.87 | 40 |
| V | 0.86 | 0.83 | 0.85 | 143 |
| X | 0.00 | 0.00 | 0.00 | 2 |
| accuracy | | | 0.88 | 562 |
| macro avg | 0.73 | 0.72 | 0.72 | 562 |
| weighted avg | 0.88 | 0.88 | 0.87 | 562 |

Kết quả của VnCoreNLP trên tập test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| A | 0.94 | 1.00 | 0.97 | 16 |
| C | 0.89 | 1.00 | 0.94 | 8 |
| D | 1.00 | 0.75 | 0.86 | 4 |
| E | 1.00 | 1.00 | 1.00 | 12 |
| M | 1.00 | 1.00 | 1.00 | 5 |
| N | 0.91 | 0.91 | 0.91 | 53 |
| P | 0.77 | 1.00 | 0.87 | 10 |
| R | 1.00 | 1.00 | 1.00 | 13 |
| V | 0.98 | 0.90 | 0.94 | 52 |
| X | 0.00 | 0.00 | 0.00 | 1 |
| accuracy | | | 0.93 | 174 |
| macro avg | 0.85 | 0.86 | 0.85 | 174 |
| weighted avg | 0.94 | 0.93 | 0.93 | 174 |

*Hình 17. Kết quả của VnCoreNLP*

Tổng hợp lại kết quả thực nghiệm và so sánh kết quả của 2 phương pháp HMM kết hợp với Viterbi và sử dụng VnCoreNLP với bảng dữ liệu như sau:

| | ***Accuracy*** |
|---|---|
| *HMM + Viterbi trên bộ train* | 78% |
| *HMM + Viterbi trên bộ test* | 57% |
| *VnCoreNLP trên bộ train* | 88% |
| *VnCoreNLP trên bộ test* | 93% |

*Bảng 2. Kết quả thực nghiệm*

### 6.2. Nhận xét và hướng phát triển

Trên cả 2 tập dữ liệu thì phương pháp sử dụng VnCoreNLP đều cho ra kết quả tốt hơn so với phương pháp HMM kết hợp Viterbi. Với phương pháp HMM kết hợp Viterbi đã được huấn luyện trên tập train nên độ chính xác cho ra là cao hơn khá nhiều so với trên tập test. Điều này có thể là do dữ liệu train là quá ít dẫn đến hiện tượng overfitting và độ chính xác trên tập test là rất thấp. Với phương pháp VnCoreNLP dù không được huấn luyện trên tập train nhưng vẫn cho ra kết quả vô cùng tốt. Kích thước chênh lệch giữa hai tập dữ liệu với tập train có 49 mẫu và tập test chỉ có 12 mẫu cũng khiến cho độ chính xác của VnCoreNLP trên tập train là thấp hơn trên tập test.

Hướng phát triển:

- Xây dựng bộ dữ liệu với câu và nhãn đa dạng hơn.
- Cải thiện khả năng gán nhãn.
- Thực hiện thử nghiệm với Hidden Markov bậc 2.

## Chương 7. **KẾT LUẬN**

Trong báo cáo này, nhóm áp dụng các kiến thức cơ bản của môn học Xử lý Ngôn ngữ Tự nhiên để thực hiện bài toán gán nhãn từ loại qua 2 bài toán nhỏ là bài toán tách từ bằng thuật toán Longest Matching và gán nhãn tự động với mô hình Hidden Markov kết hợp với thuật toán Viterbi. Kết quả đạt được là thành công thực hiện tương đối chính xác việc gán nhãn từ loại Tiếng Việt qua mô hình và so sánh với công cụ VnCoreNLP.

Cụ thể với bài toán tách từ, nhờ việc xây dựng bộ ngữ liệu và Vocab hợp lý, độ chính xác của thuật toán Longest Matching đã đạt độ chính xác hơn 90% và gần bằng với độ chính xác của công cụ VnCoreNLP. Điểm mạnh của phương pháp này là đơn giản, dễ cài đặt tuy nhiên bộ Vocab bi grams và tri grams chắc chắn sẽ không thể bao quát được toàn

bộ các từ vựng Tiếng Việt và sẽ có nhiều từ phương pháp sẽ không thể sử lý được nhất là danh từ riêng, tên riêng,…

Với bài toán gán nhãn từ loại nhãn từ loại, nhóm cũng đã thành công xây dựng được mô hình Hidden Markov kết hợp với thuật toán Viterbi. Tuy kết quả dự đoán có độ chính xác chưa được cao (khoảng 78% trên tập train và 57% trên tập test) nhưng kết quả này là có thể hiểu được bởi sự hạn chế về mặt dữ liệu. Chỉ với bộ dữ liệu tập train 49 câu, tập test 13 câu và số lượng các nhãn cũng chưa thật sự cân bằng, sẽ có rất nhiều trường hợp khó mà mô hình chưa được học, hay các từ không hề có trong tập train khiến mô hình rất dễ bị overfiting. Nhóm đưa ra hướng phát triển là tổng hợp thêm nhiều dữ liệu bao quát nhiều ngữ cảnh và nhiều từ vựng Tiếng Việt sẽ giúp kết quả dự đoán tăng lên rất nhiều.

Cuối cùng nhóm cũng dự định thử nghiệm mô hình Hidden Markov bậc cao hơn để cải thiện độ chính xác. Qua báo cáo đồ án lần này nhóm đã học được nhiều kiến thức về ngôn ngữ và xử lý ngôn ngữ tự nhiên, từ đó xây dựng được nền tảng để tìm hiểu các mô hình xử lý ngôn ngữ tự nhiên cao hơn như BERT, RoBERTa, GPT, …  và nhiều ứng dụng bổ ích của Xử lý Ngôn ngữ Tự nhiên.

### 7.1. Tỉ lệ đóng góp

Tỉ lệ đóng góp của các thành viên trong nhóm được ghi nhận như sau:

| | |
|---|---|
| **Nguyễn Đức Phương Nguyên (19521913)** | **40%** |
| **Nguyễn Ngọc Thái Nguyên (19521917)** | **30%** |
| **Lê Quang Minh (19521836)** | **30%** |

# TÀI LIỆU THAM KHẢO

vncorenlp. “GitHub - Vncorenlp/VnCoreNLP: A Vietnamese Natural Language Processing Toolkit (NAACL 2018).” *GitHub*, 11 Feb. 2023, github.com/vncorenlp/VnCoreNLP. Accessed 1 Mar. 2023.

Ong, Hong. “Gán Nhãn Từ Loại (Part-of-Speech Tagging POS).” *Ông Xuân Hồng*, Ông Xuân Hồng, 2 Sept. 2016, ongxuanhong.wordpress.com/2016/09/02/gan-nhan-tu-loai-part-of-speech-tagging-pos/. Accessed 1 Mar. 2023.

VCCorp.vn. “Kênh Tin Tức Giải Trí - Xã Hội - Kenh14.Vn.” *Kenh14.Vn*, 2023, kenh14.vn/. Accessed 1 Mar. 2023.

“[Bài Dịch] - Mô Hình Markov Ẩn và Thuật Toán Viterbi.” *Donga.edu.vn*, 2014, dien.donga.edu.vn/chi-tiet-bai-viet/[bai-dich-chuyen-nganh]-mo-hinh-markov-an-va-thuat-toan-viterbi-5446. Accessed 1 Mar. 2023.

“Tách Từ Tiếng Việt Sử Dụng Longest Matching Và CONDITIONAL RANDOM FIELDS.” *123docz.net*, 2021, 123docz.net/document/7824054-tach-tu-tie-ng-vie-t-su-du-ng-longest-matching-va-conditional-random-fields.htm. Accessed 1 Mar. 2023.

“Trường ĐH CNTT - Website Môn Học: Đăng Nhập Vào Trang.” *Courses.uit.edu.vn*, courses.uit.edu.vn/pluginfile.php/341907/mod_resource/content/1/Chuong-4.pdf. Accessed 1 Mar. 2023.