

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

ĐH \* ĐHT



**GÁN NHÃN TỪ LOẠI TIẾNG VIỆT**  
**SỬ DỤNG MÔ HÌNH HIDDEN MARKOV**  
**KẾT HỢP THUẬT TOÁN VITERBI**

Giảng viên hướng dẫn:		
Th.S Nguyễn Trọng Chính		
Danh sách thành viên:		
STT	Họ tên	MSSV
1	Đặng Hoàng Quân	18520339
2	Phạm Phú Phước	18520131
3	Dương Quốc Lộc	18521006

**TP. HỒ CHÍ MINH – 01/2021**

# MỤC LỤC

<b>DANH MỤC HÌNH ẢNH .....</b>	<b>4</b>
<b>DANH MỤC BẢNG .....</b>	<b>5</b>
<b>TÓM TẮT .....</b>	<b>6</b>
<b>Chương 1: Giới thiệu .....</b>	<b>7</b>
<b>Chương 2: Thu thập dữ liệu.....</b>	<b>8</b>
2.1. Nguồn thu thập .....	8
2.2. Thông tin cơ bản.....	8
<b>Chương 3: Tách từ .....</b>	<b>9</b>
3.1. Thuật toán Longest Matching.....	9
3.1.1. Giới thiệu.....	9
3.1.2. Mã giả.....	9
3.1.3. Ưu và nhược điểm .....	10
3.2. Cách triển khai.....	10
3.3. Đánh giá kết quả.....	11
<b>Chương 4: Tạo ngữ liệu và gán nhãn .....</b>	<b>12</b>
4.1. Tạo ngữ liệu.....	12
4.2. Gán nhãn từ loại .....	16
4.2.1. Training .....	16
4.2.2. Testing .....	17
4.3. Mô hình Hidden Markov (HMM) .....	18
4.3.1. Markov Chain.....	18
4.3.2. Giới thiệu HMM.....	19
4.3.3. Transition Matrix A.....	19
4.3.4. Emission Matrix B.....	20

4.4. Thuật toán Viterbi .....	21
4.4.1. Khởi tạo .....	21
4.4.2. Viterbi Forward .....	22
4.4.3. Viterbi Backward .....	23
<b>Chương 5: Kết quả và đánh giá .....</b>	<b>25</b>
<b>Chương 6: Kết luận.....</b>	<b>27</b>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>28</b>

## DANH MỤC HÌNH ẢNH

Hình 2.1. Một số câu có trong bộ dữ liệu gốc. ....	8
Hình 3.1. Kết quả tách từ theo từng phương pháp khác nhau của câu đầu tiên .....	11
Hình 4.1. Một số từ kèm nhãn trong bộ dữ liệu GOLD .....	12
Hình 4.2. Một số từ trong tập Train.....	14
Hình 4.3. Các nhãn trong tập Train .....	14
Hình 4.4. Một số từ trong tập Test .....	15
Hình 4.5. Quá trình đọc dữ liệu 2 tập Train và Test.....	16
Hình 4.6. Một số giá trị trong từ điển Transition Counts .....	17
Hình 4.7. Một số giá trị trong từ điển Emission Counts.....	17
Hình 4.8. Các nhãn trong từ điển Tag Counts .....	17
Hình 4.9. Markov Chain .....	18
Hình 4.10. Ma trận phát xạ (Emission Matrix) .....	20
Hình 4.11. Khởi tạo thuật toán Viterbi cho 2 tập Train và Test.....	22
Hình 4.12. Viterbi Forward cho 2 tập Train và Test .....	23
Hình 4.13. Một số kết quả dự đoán sau khi Viterbi Backward trên tập Test .....	24
Hình 5.1. Kết quả gán nhãn 10 câu trong tập Test sử dụng Hidden Markov và Viterbi...	25
Hình 5.2. Kết quả gán nhãn 10 câu trong tập Test sử dụng thư viện VnCoreNLP .....	25
Hình 5.3. Kết quả đánh giá chi tiết trên tập Test sử dụng Hidden Markov và Viterbi .....	26

## **DANH MỤC BẢNG**

Bảng 3.1. Tổng hợp kết quả tách từ.....	11
Bảng 4.1. Danh sách nhãn từ loại.....	13
Bảng 5.1: Đánh giá gán nhãn trên tập Test .....	25

## TÓM TẮT

Trong báo cáo này sẽ trình bày 2 bài toán quan trọng trong xử lý ngôn ngữ tự nhiên đó là bài toán tách từ và gán nhãn từ loại tiếng Việt. Báo cáo gồm 6 phần. Trong đó 2 chương đầu tiên sẽ giới thiệu đề tài và bộ dữ liệu nhóm đã thu thập. Chương 3 sẽ nói về cách xử lý bài toán tách từ tiếng Việt có nhiều âm tiết phức tạp. Chương 4 là chương chính, trình bày cách giải quyết bài toán gán nhãn từ loại với phương pháp chính Hidden Markov kết hợp cùng thuật toán Viterbi. 2 chương cuối sẽ kết luận và đánh giá các phương pháp đã thực hiện và cũng như định hướng phát triển bài toán sau này.

## Chương 1: Giới thiệu

Part of speech (POS) tagging là một trong những phương pháp quan trọng của xử lý ngôn ngữ tự nhiên, cũng như trong việc hiểu nội dung câu hoặc văn bản. POS là thuật ngữ truyền thống để chỉ các loại từ được phân biệt về mặt ngữ pháp trong một ngôn ngữ. Trong quá trình phát triển chúng ta quen với việc xác định từ loại trong văn bản. Đọc một câu chúng ta có thể xác định rõ từ loại như là danh từ, động từ hoặc tính từ...

Để xác định từ rõ từ loại trong câu thường phức tạp hơn nhiều trong việc ánh xạ các từ qua từ điển. Đó là bởi vì một từ có thể được gán rất nhiều từ loại dựa vào ngữ cảnh của văn bản. Đây gọi là sự nhập nhằng. Thật khó để ta xác định một từ đó thuộc từ loại nào dựa vào một ngữ liệu nhất định vì tất cả ngữ cảnh mới và từ mới mỗi ngày liên tục xuất hiện đó cũng là vấn đề cho việc gán từ loại thủ công.

Phân biệt các bộ phận của từ trong câu sẽ giúp ta hiểu rõ hơn về ý nghĩa của câu. Điều này cực kỳ quan trọng trong các truy vấn tìm kiếm. Việc xác định danh từ riêng, tổ chức, ký hiệu cổ phiếu hoặc bất kỳ thứ gì tương tự sẽ cải thiện đáng kể mọi thứ, từ nhận dạng giọng nói đến tìm kiếm.

Trong đề tài này nhóm sẽ sử dụng mô hình Hidden Markov kết hợp thuật toán Viterbi để gán nhãn từ loại tiếng Việt và so sánh độ chính xác với thư viện VnCoreNLP

## Chương 2: Thu thập dữ liệu


### 2.1. Nguồn thu thập

Gồm các câu bất kỳ thuộc nhiều chủ đề được thu thập từ 2 trang <https://vnexpress.net/> và <https://dantri.com.vn/>. Do đây là 2 nguồn báo uy tín ở Việt Nam nên nhóm sẽ lựa chọn để sử dụng cho việc thu thập dữ liệu.

### 2.2. Thông tin cơ bản

Bộ dữ liệu gốc được lưu với tên sentences.txt:

- Mỗi dòng là 1 câu.
- Các từ được phân cách với nhau bằng dấu cách ‘ ’.
- Cuối câu là 1 dấu chấm ‘.’.
- Số lượng câu: 60.
- Số từ nhiều nhất trong một câu: 46.
- Số từ ít nhất trong một câu: 8.

 sentences.txt - Notepad

File Edit Format View Help

```
Nhiều người có hoàn cảnh giống ông .  
Bắc Mỹ , nơi tôi lớn lên , rất lạnh .  
Bố mua cho chúng tôi quả bóng cao su mềm .  
Những ngày đẹp đẽ ấy , bố luôn dành thời gian để vui vẻ với anh em tôi .  
Trong khi chờ nước nóng , tôi pha rượu táo nóng .  
Hàng trăm người đến dùng tiệc tự chọn ở chỗ mẹ những ngày Giáng sinh .
```

*Hình 2.1. Một số câu có trong bộ dữ liệu gốc.*



## Chương 3: Tách từ

Bài toán tách từ là 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. Ví dụ từ (word) “cá\_nhân” được tạo lên bởi 2 âm (syllable) là “cá” và “nhân”. Trong khi hai từ đơn “cá” và từ đơn “nhân” lại có thể mang ý nghĩa khác. Do vậy, tách từ tiếng Việt là bước quan trọng chúng ta cần thực hiện trước khi đưa dữ liệu vào các bước tiếp theo, ví dụ như word embedding.

### 3.1. Thuật toán Longest Matching

#### 3.1.1. Giới thiệu

Longest Matching là thuật toán dựa trên tử tưởng tham lam. Nó xét các tiếng từ trái qua phải, các tiếng đầu tiên dài nhất có thể mà xuất hiện trong từ điển sẽ được tách ra làm một từ. Với vị trí âm tiết hiện tại sẽ kiểm tra xem từ đó và 2 âm tiếp theo có thể ghép thành 1 từ có nghĩa hay không bằng cách kiểm tra trong từ điển tri\_gram. Nếu không thể tạo ra được từ có nghĩa từ 3 âm tiết thì ta tiếp tục kiểm tra xem âm tiết hiện tại và âm tiếp theo có thể ghép được thành một từ có nghĩa hay không bằng cách kiểm tra trong từ điển bi\_gram. Thuật toán sẽ dừng khi xét hết các tiếng trong câu.

#### 3.1.2. Mã giả

current\_word\_id = 0 (Đặt từ hiện tại ở đầu câu):

```
while (current_word_id+1) != len(text) {  
    xét từ hiện tại với 2 từ sau nó trong từ điển tri_gram  
    check if {"word, word+1, word+2"} is in tri_gram  
    if true: take {word, word1, word2; current_word_id += 3} → 3 từ đó tạo  
    thành từ ghép  
    xét từ hiện tại với tiếng sau trong từ điển trong từ điển 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  
}
```

### 3.1.3. Ưu và nhược điểm

Ưu điểm thuật toán này chính là cài đặt đơn giản, độ phức tạp tính toán hợp lý, không yêu cầu dữ liệu huấn luyện. Tuy nhiên, phương pháp này sẽ khó có thể xử lý được các tình huống nhập nhằng nhất định như việc lặp từ trong câu cũng như xử lý tình huống xuất hiện từ mới không tồn tại trong từ điển

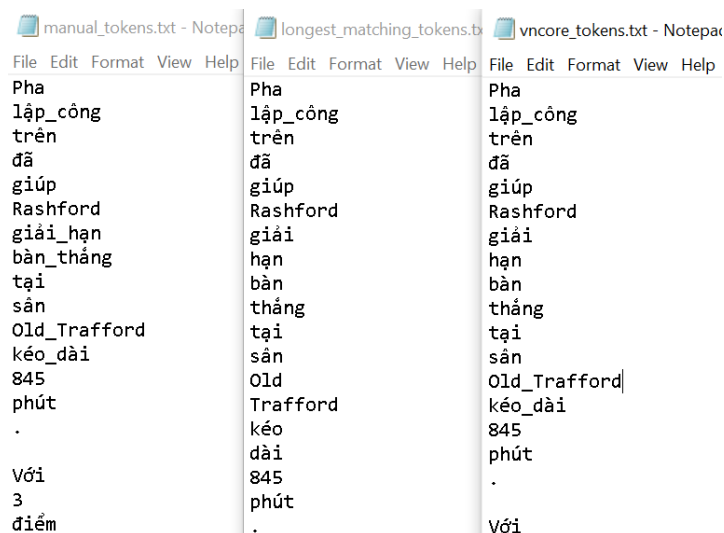
### 3.2. Cách triển khai

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 dữ liệu mới được lưu với tên manual\_tokens.txt sẽ chứa kết quả tách từ đúng nhất cho 60 câu nhóm mà đã thu thập:

- Các âm tiết của từ ghép sẽ được nối bằng dấu gạch dưới ‘\_’.
- Mỗi dòng là 1 từ.
- Mỗi câu sẽ được ngăn cách bằng 1 dòng trống.
- Số lượng câu: 60.
- Số lượng từ: 970.
- Số lượng từ ghép: 309.

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, với kết quả tách từ của mỗi phương pháp lần lượt được lưu trong các file có tên là longest\_matching\_tokens.txt chứa 1049 từ với 257 từ ghép và vncore\_tokens.txt chứa 1006 từ với 282 từ ghép.



Hình 3.1. Kết quả tách từ theo từng phương pháp khác nhau của câu đầu tiên

### 3.3. Đánh giá kết quả

Bảng 3.1. Tổng hợp kết quả tách từ.

	Longest Matching	VnCoreNLP
Accuracy	0.856044	0.931868
Precision	0.891051	0.950355
Recall	0.7411	0.867314
True Positive	229	268
False Positive	28	14
Total True	779	848
Total Errors	210	99

Qua bảng 1, có thể thấy rằng các số liệu đánh giá của việc tách từ khi sử dụng thư viện VnCoreNLP đều rất tốt và tốt hơn nhiều so với khi dùng thuật toán Longest Matching, đồng thời nhóm nhận thấy Longest Matching không phân biệt tốt được các tên riêng như Old\_Trafford, ... có thể thấy ở hình 3. Đây có thể do bộ dữ liệu bi-grams và tri-grams được dùng cho thuật toán này chưa đủ tốt. Độ phủ (Recall) của thuật toán này thấp hơn khá nhiều so với Accuracy và Precision chứng tỏ Longest Matching tách từ chưa được tốt.

Vì quá trình tách từ bằng thuật toán và thư viện đều có thể tồn tại những sai sót nên nhóm quyết định sẽ sử dụng bộ dữ liệu có được sau khi thực hiện tách từ bán thủ công (ở đây là file manual\_tokens.txt) để bắt đầu tiến hành gán nhãn từ loại.

## Chương 4: Tạo ngữ liệu và gán nhãn

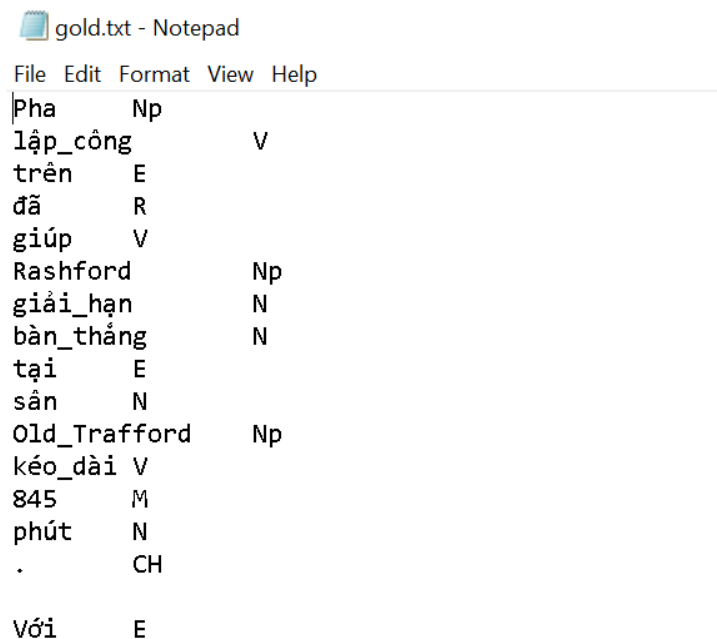
### 4.1. Tạo ngữ liệu

Sau khi có kết quả tách từ, nhóm lại tiếp tục tiến hành gán nhãn bán thủ công trên bộ dữ liệu tách từ đã chọn.

Nhóm sẽ sử dụng thư viện VnCoreNLP để gán nhãn trước, sau đó sẽ phân công các thành viên kiểm tra thủ công lại kết quả gán nhãn và sửa lại các nhãn bị sai để thu được tập dữ liệu Gold cho 60 câu đã thu thập:

Bộ dữ liệu Gold được lưu với tên gold.txt:

- Các âm tiết của từ ghép sẽ được nối bằng dấu gạch dưới ‘\_’.
- Từ được phân tách với nhãn bằng dấu tab.
- Mỗi dòng là một từ cùng với nhãn của nó.
- Mỗi câu sẽ được ngăn cách bằng một dòng trống.
- Số lượng câu: 60.
- Số lượng từ: 970.
- Số lượng nhãn: 909.



```
gold.txt - Notepad
File Edit Format View Help
Pha      Np
lập_công      V
trên      E
đã      R
giúp      V
Rashford      Np
giải_hạn      N
bàn_thắng      N
tại      E
sân      N
Old_Trafford      Np
kéo_dài V
845      M
phút      N
.      CH

Với      E
```

Hình 4.1. Một số từ kèm nhãn trong bộ dữ liệu GOLD

Bảng 4.1. Danh sách nhãn từ loại

(Nguồn: [https://github.com/vncorenlp/VnCoreNLP/blob/master/VLSP2013\\_POS\\_tagset.pdf](https://github.com/vncorenlp/VnCoreNLP/blob/master/VLSP2013_POS_tagset.pdf))

STT	Nhãn	Tên	Ví dụ
1	N	Danh từ	tiếng, nước, thủ đô, nhân dân, đồ đạc
2	Np	Danh từ riêng	Nguyễn Du, Việt Nam, Hải Phòng, Trường Đại học Bách khoa Hà Nội, Mộc tinh, Hoả tinh, Phật, Đạo Phật
3	Nc	Danh từ chỉ loại	con, cái, đứa, bức
4	Nu	Danh từ đơn vị	mét, cân, giờ, năm, nhóm, hào, xu
5	Ni	Danh từ ký hiệu	A1, A4, 60A, 60B, 20a, 20b, ABC
6	V	Động từ	ngủ, ngồi, cười; đọc, viết, đá, đặt, thích
7	A	Tính từ	tốt, xấu, đẹp; cao, thấp, rộng
8	P	Đại từ	tôi, chúng tôi, hắn, nó, y, đại nhân, đại
9	L	Định từ	mỗi, từng, mọi, cái; các, những, mấy
10	M	Số từ	một, mười, mười ba; dặm, vài, mười
11	R	Phó từ	đã, sẽ, đang, vừa, mới, từng, xong, rồi
12	E	Giới từ	trên, dưới, trong, ngoài; của, trừ, ngoài
13	C	Liên từ	vì vậy, tuy nhiên, ngược lại
14	Cc	Liên từ đẳng lập	và, hoặc, với, cùng
15	I	Thán từ	ôi, chao, a ha
16	T	Trợ từ	à, a, á, ạ, ấy, chắc, chẳng, cho, chứ
17	B	từ vay mượn	Internet, email, video, chat
18	Y	Từ viết tắt	OPEC, WTO, HIV
19	X	Các từ không thể phân loại	
20	Z	Yếu tố cấu tạo từ	bất, vô, phi
21	CH	Nhãn dành cho các loại dấu	. ! ? , ; :

Tiếp theo nhóm sẽ chia 60 câu đã được tách từ và gán nhãn trong bộ dữ liệu Gold này thành 2 tập dữ liệu: 50 câu cho tập Train và 10 câu cho tập Test. Vì trong lúc dự đoán mô hình gán nhãn có thể sẽ gặp những từ không có trong dataset của nó. Những từ này sẽ được thay thế bằng một mã không xác định. Ở đây những từ vựng trong tập train đã được xử lý để nằm trong bộ từ vựng. Những từ vựng trong tập Test không thuộc bộ từ vựng sẽ được thay thế bằng ‘—unk—’. Bộ từ vựng được lưu với tên vocabs.txt gồm 54818 từ. Quá trình tiền xử lý cũng sẽ xác định kết thúc của một câu, giá trị đó sẽ được đặt là ‘—n—’.

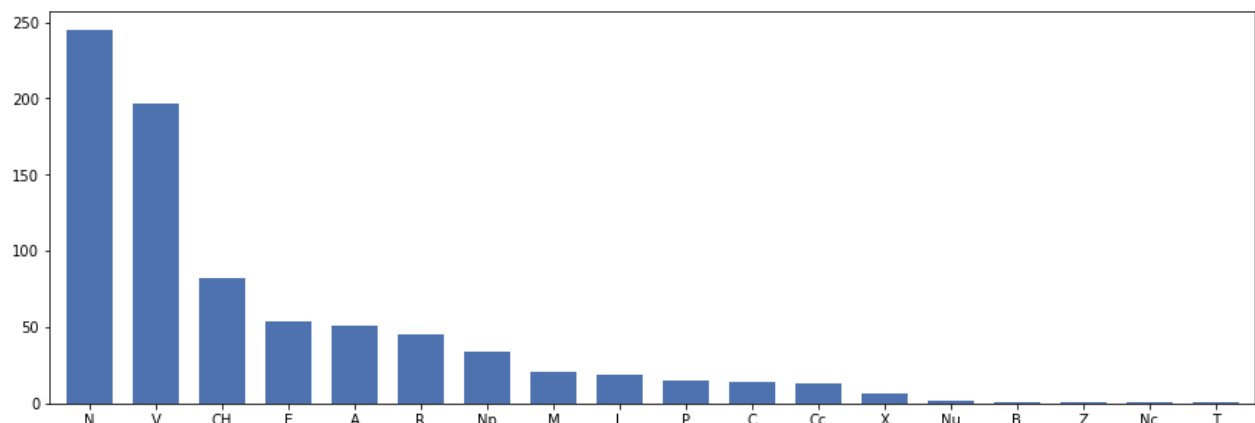
Tập Train gồm các file train\_gold.txt và train\_words.txt:

- train\_gold.txt: chứa các từ kèm nhãn để thực hiện cho việc huấn luyện (tạo ra các ma trận transition\_counts, emission\_counts, tag\_counts)
- train\_words.txt: chỉ chứa từ của các câu để thực hiện cho việc kiểm thử trên tập train (kiểm tra overfitting).
- Số lượng câu của tập Train: 50.
- Số lượng từ của tập Train: 852.
- Số lượng nhãn của tập Train: 802.

train_gold.txt - Notepad		train_words.txt - Notepad	
File	Edit Format View Help	File	Edit Format View Help
Pha	N	Pha	
lập_công	V	lập_công	
trên	E	trên	
đã	R	đã	
giúp	V	giúp	
Rashford	Np	Rashford	
giải	N	giải	
hạn	N	hạn	
bàn	N	bàn	
thắng	V	thắng	
tại	E	tại	
sân	N	sân	
Old_Trafford	Np	Old_Trafford	
kéo_dài	V	kéo_dài	
845	M	845	
phút	N	phút	
.	CH	.	
Với	C	Với	

Hình 4.2. Một số từ trong tập Train

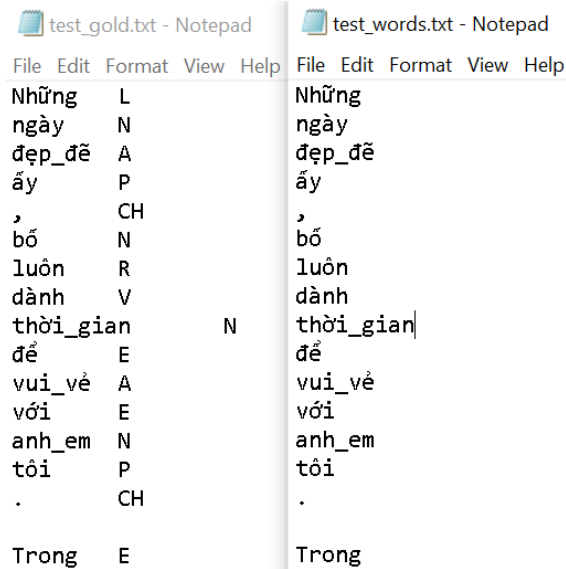
	N	V	CH	E	A	R	Np	M	L	P	C	Cc	X	Nu	B	Z	Nc	T	Total
0	245	197	82	54	51	45	34	21	19	15	14	13	6	2	1	1	1	1	802



Hình 4.3. Các nhãn trong tập Train

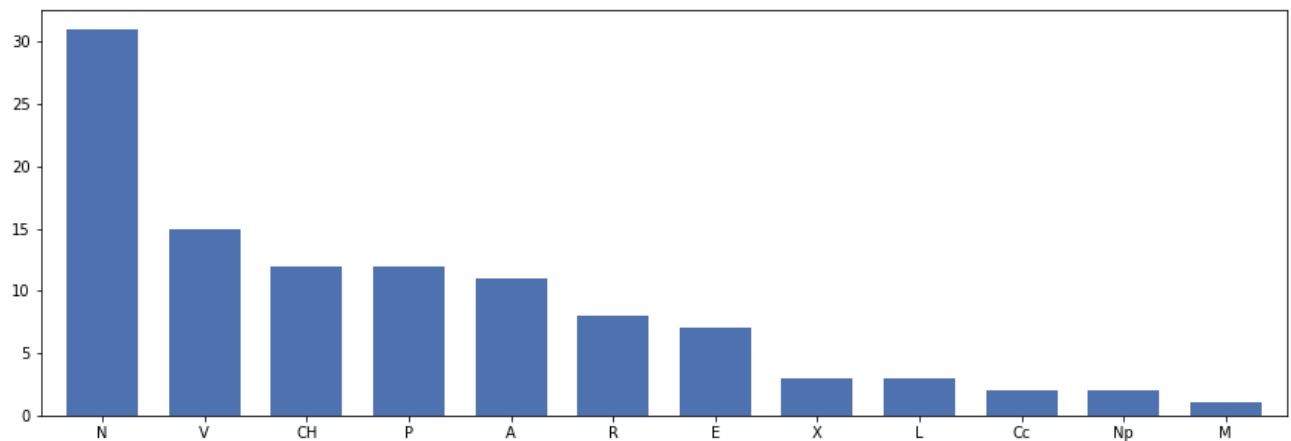
Tập Test gồm các file test\_gold.txt và test\_words.txt:

- test\_gold.txt: chứa các từ kèm nhãn, thực hiện cho việc đánh giá kết quả.
- test\_words.txt: chỉ chứa từ của các câu để thực hiện cho việc dự đoán.
- Số lượng câu của tập Test: 10.
- Số lượng từ của tập Test: 117.
- Số lượng nhãn của tập Train: 107.
- Các từ không nằm trong vocab: Giảng\_sinh, năm\_qua, gần\_đây, Tân\_Son\_Nhất.

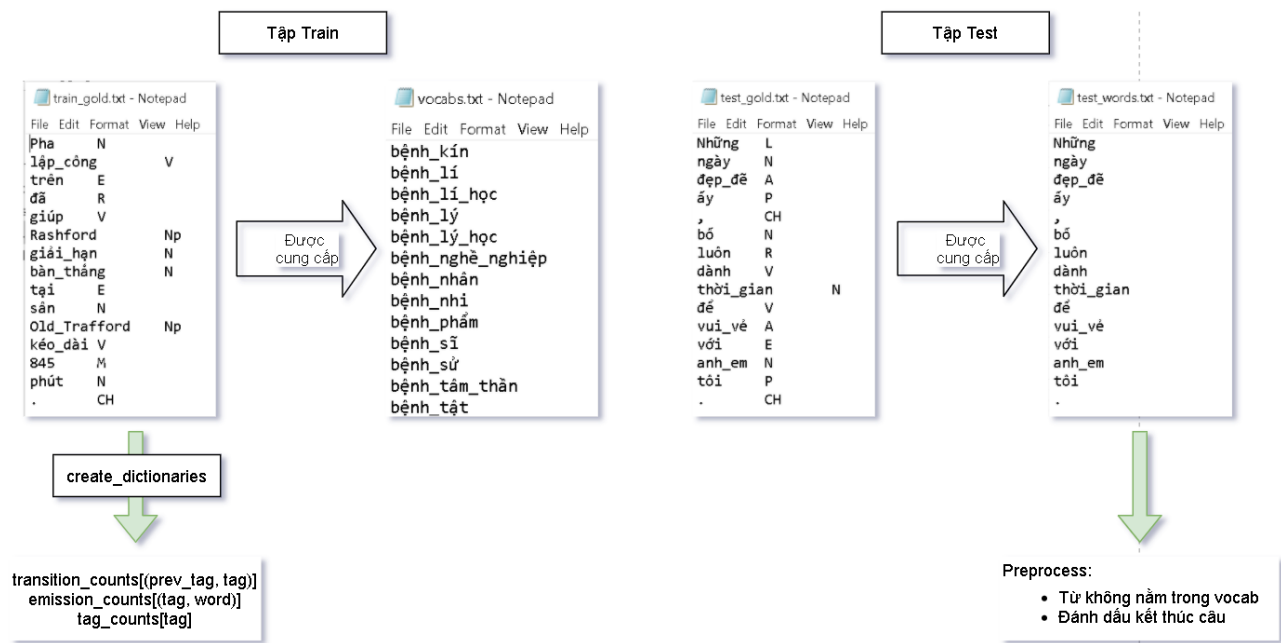


Hình 4.4. Một số từ trong tập Test

	N	V	CH	P	A	R	E	X	L	Cc	Np	M	Total
0	31	15	12	12	11	8	7	3	3	2	2	1	107



Hình 8: Các nhãn trong tập Test



Hình 4.5. Quá trình đọc dữ liệu 2 tập Train và Test

## 4.2. Gán nhãn từ loại

Nhóm sẽ bắt đầu với mô hình gán nhãn từ loại đơn giản nhất có thể và sau đó sẽ xây dựng lên mô hình phức tạp hơn.

### 4.2.1. Training

Trước khi bắt đầu dự đoán nhãn của mỗi từ, nhóm tính toán một số từ điển (dictionary) sẽ giúp tạo ra các bảng. Ngoài ra, nhóm bổ sung thêm nhãn ‘—s—’ để chỉ ra phần bắt đầu của mỗi câu.

Từ điển Transition Counts:

- Tính số lần mỗi nhãn xảy ra bên cạnh một nhãn khác
- Từ điển sẽ tính giá trị  $P(t_i|t_{i-1})$ . Đây là xác suất của nhãn ở vị trí  $i$  được cho bởi nhãn ở vị trí  $i - 1$ . Để tính toán giá trị này, nhóm sẽ tạo một từ điển tên là transition\_counts, trong đó: keys là (prev\_tag, tag), value là số lần 2 nhãn đó xuất hiện theo thứ tự đó.



Transition examples:  
(( '--s--', 'N'), 28)  
(('N', 'V'), 68)  
(('V', 'E'), 21)

Hình 4.6. Một số giá trị trong từ điển Transition Counts

Từ điển Emission Counts:

- Tính xác suất của một từ được cho bởi nhãn của nó
- Từ điển sẽ tính giá trị  $P(w_i|t_i)$ . Để tính toán giá trị này, nhóm sẽ tạo một từ điển tên là emission\_counts, trong đó: keys là (tag, word), value là số lần cặp giá trị đó xuất hiện trong tập Train.

Emission examples:  
(('N', 'Pha'), 1)  
(('V', 'lập\_công'), 1)  
(('E', 'trên'), 4)

Hình 4.7. Một số giá trị trong từ điển Emission Counts

Từ điển Tag Counts:

- Được đặt với tên tag\_counts
- key là nhãn, value là số lần nhãn đó xuất hiện

Số nhãn: 19

['--s--', 'A', 'B', 'C', 'CH', 'Cc', 'E', 'L', 'M', 'N', 'Nc', 'Np', 'Nu', 'P', 'R', 'T', 'V', 'X', 'Z']

Hình 4.8. Các nhãn trong từ điển Tag Counts

#### 4.2.2. Testing

Sau khi tạo ra các từ điển, nhóm bắt đầu kiểm tra độ chính xác của mô hình gán thẻ đơn giản này bằng cách sử dụng từ điển Emission Counts.

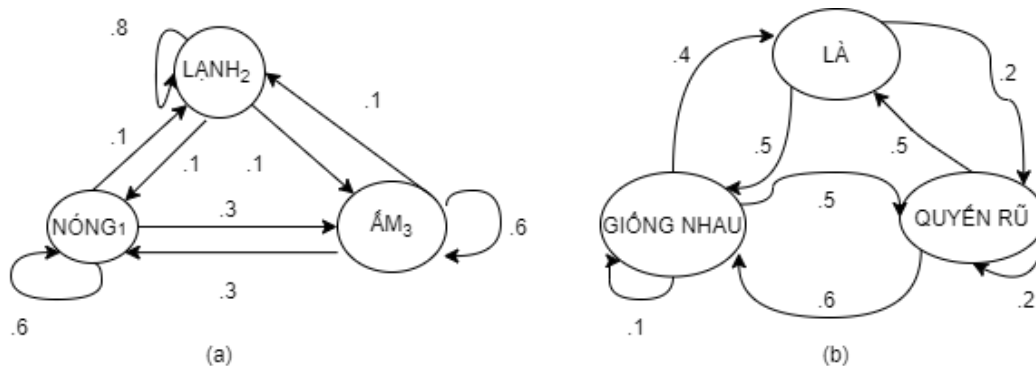
Để gán nhãn cho một từ, nhóm chỉ định nhãn thường gặp nhất cho từ đó trong tập Train. Sau đó đánh giá xem cách tiếp cận này hoạt động có tốt không. Mỗi lần dự đoán sẽ dựa trên nhãn thường xuyên nhất cho từ đã cho rồi sẽ kiểm tra xem có giống với nhãn thực của từ không. Nếu có thì dự đoán đã đúng. Tính độ chính xác bằng số dự đoán đúng chia cho tổng số từ mà đã dự đoán nhãn:

- Độ chính xác trên tập Train: 0.9330985915492958
- Độ chính xác trên tập Test: 0.4188034188034188

### 4.3. Mô hình Hidden Markov (HMM)

#### 4.3.1. Markov Chain

Mô hình Hidden Markov (HMM) dựa trên Markov Chain (xích Markov). Một Markov Chain là một mô hình cho chúng ta biết về xác suất của một chuỗi các biến ngẫu nhiên hay còn được gọi là các trạng thái, mỗi trạng thái có thể nhận những giá trị từ một tập nào đó. Các tập này có thể là tập các từ, các nhãn hoặc những biểu tượng trình bày một điều gì đó. Một Markov Chain thì đưa ra một giả định rằng nếu dự đoán tương lai của chuỗi thì những trạng thái hiện tại là điều quan trọng nhất để dự đoán. Tương lai thì không bị ảnh hưởng bởi tất cả trạng thái trước trạng thái hiện tại mà nó thông qua trạng thái hiện tại. Ví dụ nếu bạn muốn dự báo thời tiết của ngày mai, bạn có thể dựa vào thời tiết của hôm nay nhưng không được phép xem thời tiết của hôm qua.



Hình 4.9. Markov Chain

Một cách tổng quát hóa, xét chuỗi các biến trạng thái  $q_1, q_2, \dots, q_i$ . Mô hình Markov thể hiện giả định Markov về các xác suất của chuỗi này đó là khi dự đoán tương lai không quan trọng quá khứ mà chỉ cần hiện tại.

Giả định Markov:  $P(q_i = a | q_1, \dots, q_{i-1}) = P(q_i = a | q_{i-1})$

Một Markov chain được xác định bởi các thành phần sau:

- $Q = q_1, q_2, \dots, q_N$ : là một tập của N trạng thái.

- $A = a_{11}, a_{12}, \dots, a_{N1}, \dots, a_{NN}$ : là một ma trận chuyển trạng thái A, mỗi phần tử  $a_{ij}$  thể hiện xác suất chuyển từ trạng thái  $i$  đến trạng thái  $j$  với ràng buộc là  $\sum_{j=1}^n a_{ij} = 1, \forall i$ .
- $\pi = \pi_1, \pi_2, \dots, \pi_N$ : là một phân phối xác suất ban đầu trên mỗi trạng thái.  $\pi_i$  là xác suất mà Markov chain sẽ bắt đầu ở trạng thái  $i$ , một vài trạng thái  $j$  có thể có  $\pi_j = 0$ , có nghĩa là chúng không được khởi tạo phân phối xác suất ban đầu, nó cũng có một ràng buộc là  $\sum_{i=1}^n \pi_i = 1$ .

#### 4.3.2. Giới thiệu HMM

HMM (Hidden Markov Models) 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, ...

Mô hình Markov chứa một số trạng thái và xác suất chuyển đổi giữa các trạng thái đó. Trong trường hợp này, các trạng thái là nhãn từ loại. Mô hình Markov sử dụng ma trận chuyển tiếp A (Transition Matrix). Mô hình Markov ần thêm một ma trận phát xạ B (Emission Matrix) mô tả xác suất của một quan sát có thể nhìn thấy khi ta ở một trạng thái cụ thể. Trong trường hợp này, emission là các từ trong ngữ liệu. Trạng thái, thứ được xem là ẩn (Hidden) chính là nhãn của từ đó.

Dựa vào các từ điển transition\_counts, emission\_counts and tag\_counts đã thu được, Nhóm sẽ bắt đầu triển khai mô hình Hidden Markov. Điều này cho phép xây dựng Transition Matrix A và Emission Matrix B. Bên cạnh đó nhóm cũng sẽ sử dụng một tham số làm mịn (smoothing) khi tính toán các ma trận này.

#### 4.3.3. Transition Matrix A

	Cc	E	L	M	N
Cc	0.000077	0.000077	0.076888	0.076888	0.230509
E	0.018531	0.000019	0.092579	0.055555	0.536867
L	0.000053	0.000053	0.000053	0.000053	0.999054
M	0.047624	0.047624	0.000048	0.142776	0.523384
N	0.016329	0.077549	0.000004	0.020411	0.232639

Hình 8: Ma trận chuyển tiếp (Transition Matrix)

Ma trận ở trên đã được tính toán với tham số smoothing. Mỗi ô là xác suất để đi từ 1 nhãn tới nhãn khác. Nói cách khác, có 0.077549 cơ hội để chuyển từ nhãn N tới nhãn E. Tổng của mỗi hàng phải bằng 1, vì giả định rằng nhãn tiếp theo phải là một trong các cột có sẵn trong bảng. Việc làm mịn được thực hiện như sau:

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

- $N$ : tổng số nhãn.
- $C(t_{i-1}, t_i)$ : số lượng bộ giá trị (prev\_tag, tag) trong từ điển transition\_counts.
- $C(t_{i-1})$ : số lượng của nhãn trước trong từ điển tag\_counts
- $\alpha$ : là tham số smoothing.

#### 4.3.4. Emission Matrix B

	thông_báo	hạt_nhân	tinh_thần	lập_công	vị_trí
N	0.003339	0.006674	0.003339	0.000003	0.010009
V	0.000004	0.000004	0.000004	0.003975	0.000004
CH	0.000007	0.000007	0.000007	0.000007	0.000007
Cc	0.000015	0.000015	0.000015	0.000015	0.000015
E	0.000009	0.000009	0.000009	0.000009	0.000009

Hình 4.10. Ma trận phát xạ (Emission Matrix)

Ma trận này có chiều là (num\_tags, N) với num\_tags là số nhãn có thể có và N là số từ trong bộ từ vựng. Xác suất ma trận B được tính theo công thức:

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

- $C(t_i, w_i)$ : là số lượng của từ thứ  $i$  ( $w_i$ ) đi với nhãn thứ  $i$  ( $t_i$ ) trong tập Train (được lưu trong từ điển emission\_counts)
- $C(t_i)$ : là số lần nhãn thứ  $i$  ( $t_i$ ) trong tập Train (được lưu trong từ điển tag\_counts)
- $N$ : số lượng từ trong từ điển.
- $\alpha$ : tham số smoothing.

#### 4.4. Thuật toán Viterbi

Nhóm tiến hành kết hợp mô hình Hidden Markov đã có với thuật toán Viterbi sử dụng quy hoạch động. Cụ thể, nhóm sẽ sử dụng 2 ma trận của A, B để tính toán thuật toán Viterbi. Quy trình này sẽ được chia thành 3 bước chính:

- Khởi tạo: khởi tạo 2 ma trận `best_paths` và `best_probabilities` sẽ được dùng cho hàm `feed_forward` ở bước Forward
- 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.

##### 4.4.1. Khởi tạo

Khởi tạo 2 ma trận có cùng chiều:

- `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 ngoại trừ cột 0 của `best_probs`. Cột 0 của `best_probs` được khởi tạo với giả định rằng từ đầu tiên của ngữ liệu được đặt trước bởi một ký tự bắt đầu ('—s—'):

- Xác suất của đường đi tốt nhất từ vị trí bắt đầu đến một nhãn nhất định có vị trí  $i$  được ký hiệu là `best_probs[s_idx, i]`. Đây là xác suất mà nhãn bắt đầu đi sang nhãn được biểu thị bằng chỉ số  $i$
- $A[s\_idx, i]$  và nhãn được biểu thị bằng chỉ số  $i$  cho ra từ đầu tiên của ngữ liệu là  $B[i, \text{vocabs}[\text{corpus}[0]]]$ , trong đó `vocabs` là từ điển mà trả về 1 số nguyên duy nhất tương ứng với 1 từ cụ thể và `corpus[0]` là từ đầu tiên của ngữ liệu.

Việc này trông như sau:

$$\text{best\_probs}[s\_idx, i] = A[s\_idx, i] \times B[i, \text{corpus}[0]]$$

Để tránh việc nhân và lưu các giá trị nhỏ, nhóm sẽ lấy ln của tích trên để chúng trở thành tổng của 2 log:

$$\text{best\_probs}[s\_idx, i] = \ln(A[s\_idx, i]) + \ln(B[i, \text{corpus}[0]])$$

Tóm lại việc triển khai khởi tạo best\_probs như sau:

- *if*  $A[s_{idx}, i] \neq 0$ :  $\text{best\_probs}[i, 0] = \ln(A[s_{idx}, i]) + \ln(B[i, \text{vocab}[\text{corpus}[0]]])$
- *if*  $A[s_{idx}, i] == 0$ :  $\text{best\_probs}[i, 0] = \text{float}(' - inf')$

```
best_probs_train[0, 0]: -21.750009889225183
best_paths_train[2, 3]: 0

best_probs_test[0, 0]: -21.750009889225183
best_paths_test[2, 3]: 0
```

Hình 4.11. Khởi tạo thuật toán Viterbi cho 2 tập Train và Test

#### 4.4.2. Viterbi Forward

Điền thông tin vào ma trận best\_probs và best\_paths đã khởi tạo bằng hàm viterbi\_forward:

- Lặp qua ngữ liệu.
- Với mỗi từ, tính xác suất cho mỗi nhãn có thể có.
- Tính toán sẽ bao gồm cả đường đi đến tổ hợp (từ, thẻ) đó.

Công thức để tính xác suất và đường đi cho từ thứ  $i$ , từ trước đó  $i - 1$  trong ngữ liệu, nhãn  $j$  hiện tại và nhãn  $k$  trước đó là:

$$\text{prob} = \text{best\_prob}_{k, i-1} + \log(A_{k,i}) + \log(B_{j, \text{vocabs}(\text{corpus}_i)})$$

- $\text{corpus}_i$ : từ ở vị trí thứ  $i$  trong ngữ liệu
- $\text{vocabs}$ : từ điển trả về những số nguyên duy nhất đại diện cho từ nhất định
- $k$ : số nguyên đại diện nhãn trước đó.

Triển khai hàm viterbi\_forward, lưu trữ best\_path và best\_prob của mọi nhãn có thể có cho mỗi từ trong ma trận best\_probs và best\_tags bằng cách dùng mã giả bên dưới:

for mỗi từ trong ngữ liệu

for mỗi loại nhãn mà từ này có thể là

for loại nhãn mà từ trước đó có thể là

- Tính xác suất để từ trước đó có nhãn nhất định, từ hiện tại có nhãn nhất định và nhãn sẽ cho ra từ hiện tại này.

- Giữ lại xác suất cao nhất được tính cho từ hiện tại
- Lưu xác suất cao nhất này vào best\_probs
- Lưu giá trị  $k$  vào best\_paths, đại diện cho nhãn của từ trước đó mà tạo ra xác suất cao nhất

```
best_probs_train[0, 1]: -29.62128154751998
best_paths_train[0, 4]: 14

best_probs_test[0, 1]: -27.386413580260943
best_paths_test[0, 4]: 16
```

Hình 4.12. Viterbi Forward cho 2 tập Train và Test

#### Mã giả hàm viterbi\_forward:

**Function** viterbi\_forward (A, B, corpus, best\_probs, best\_paths, vocabs\_dict):

```
for i ← 1: count(corpus) do
  for j ← 0: count(số nhãn) do
    best_prob_i ← âm vô cùng
    best_path_i ← Null
    for k ← 1: count(số nhãn) do
      index ← vị trí từ thứ i trong corpus
      prob ← best_probs[k][I - 1] + log(A[k][j]) + log(B[j][index])
      if prob > best_prob_i then
        best_prob_i ← prob
        best_path_i ← k
      end if
    end for
    best_probs[j][i] ← best_prob_i
    best_paths[j][i] ← best_path_i
  end for
end for
Return best_probs, best_paths
```

#### 4.4.3. Viterbi Backward

Thuật toán Viterbi Backward bằng cách sử dụng các ma trận best\_paths và best\_probs sẽ trả về danh sách các nhãn được dự đoán cho mỗi từ trong ngữ liệu:

- Lặp qua tất cả các hàng (nhãn) tại cột cuối cùng của `best_probs` và tìm hàng (nhãn) có giá trị lớn nhất cho từ cuối cùng.
- Bắt đầu tại cột cuối cùng của `best_paths`, sử dụng `best_probs` để tìm nhãn có nhiều khả năng nhất cho từ cuối cùng trong ngữ liệu. Sau đó, sử dụng `best_paths` 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_paths` từ từ cuối cùng đến từ thứ 0 trong ngữ liệu)

```
Dự đoán cho test_pred[-7:116]:
['công_cụ', 'không_thể', 'đào_ngược', 'trong', 'cuộc_sống', '.']
['Np', 'R', 'V', 'E', 'N', 'CH']
Dự đoán cho test_pred[0:7]:
['Những', 'ngày', 'đẹp_đẽ', 'ấy', ',', 'bố', 'luôn']
['L', 'N', 'Cc', 'A', 'CH', 'P', 'R']
```

Hình 4.13. Một số kết quả dự đoán sau khi Viterbi Backward trên tập Test

#### Mã giả hàm `viterbi_backward`:

**Function** `viterbi_backward` (`best_probs`, `best_paths`, `corpus`, `states`):

`m ← count (corpus)`

Khởi tạo mảng `z` với `m` phần tử

Khởi tạo mảng `pred` với `m` phần tử

`best_prob_for_last_word ← âm vô cùng`

**for** `k ← 1: count(số nhãn)` **do**

**if** `best_probs[k][m - 1] > best_prob_for_last_word` **then**

`best_prob_for_last_word ← best_probs[k][m - 1]`

`z[m] ← k`

**end if**

**end for**

`pred[m] ← states[z[m - 1]]`

**for** `i ← m - 1: -1` **do**

`z[i - 1] ← best_paths[z[i], i]`

`pred[i - 1] ← states[z[i - 1]]`

**end for**

**Return** `pred`



## Chương 5: Kết quả và đánh giá

Sau khi xây dựng xong mô hình Hidden Markov kết hợp thuật toán Viterbi. Nhóm tiến hành dự đoán 10 câu đã được chia cho tập Test, sau đó sẽ so sánh kết quả này với kết quả khi sử dụng thư viện VnCore NLP.

Những/L ngày/N đẹp\_đẽ/Cc ấy/A ,/CH bố/P luôn/R dành/V thời\_gian/N để/E vui\_vẻ/N với/E anh\_em/N tôi/P ./CH  
Trong/E khi/N chờ/E nước/N nóng/A ,/CH tôi/P pha/V rượu/N táo/Cc nóng/A ./CH  
Hàng/M trăm/M người/N đến/V dùng/E tiệc/M tự\_chọn/Nu ở/E chỗ/N mẹ/E những/L ngày/N --unk--/A ./CH  
Năm/R nào/V cũng/R vậy/A hoặc/Cc có\_vẻ/A như\_vậy/A ./CH  
Tôi/P chưa/R bao\_giờ/V thực\_sự/E cảm\_thấy/M mệt\_mỏi/Nu vì/CH nó/P ./CH  
Tôi/P đã/R có\_thể/R chống/V lại/E cảm\_giác/N đó/V trong/E nhiều/A --unk--/A ./CH  
Những/L năm/N --unk--/CH tôi/P qua\_lại/R sân\_bay/A --unk--/Cc thường\_xuyên/A ./CH  
Trường/N tôi/P từng/R nghiêm\_căm/V sử\_dụng/V điện\_thoại/V trong/E lớp/N ./CH  
Học\_sinh/R ngày\_nay/V có\_thể/R dễ\_dàng/A tiếp\_cận/Cc bài\_học/A và/Cc phương\_pháp/A giải/Cc bài\_tập/A ./CH  
Điện\_thoại/R thông\_minh/V là/V công\_cụ/Np không\_thể/R đảo\_ngược/V trong/E cuộc\_sống/N ./CH

Hình 5.1. Kết quả gán nhãn 10 câu trong tập Test sử dụng Hidden Markov và Viterbi

Như đã phân tích những từ không thuộc bộ từ vựng sẽ có giá trị là –unk–, với VnCoreNLP đều có thể nhận biết được hết những từ này.

Những/L ngày/N đẹp\_đẽ/A ấy/P ,/CH bố/N luôn/R dành/V thời\_gian/N để/E vui\_vẻ/A với/E anh\_em/N tôi/P ./CH  
Trong/E khi/N chờ/V nước/N nóng/A ,/CH tôi/P pha/V rượu/N táo/V nóng/A ./CH  
Hàng/N trăm/M người/N đến/V dùng/V tiệc/N tự\_chọn/P ở/E chỗ/N mẹ/N những/L ngày/N Giáng\_sinh/Np ./CH  
Năm/Np nào/P cũng/R vậy/P hoặc/Cc có\_vẻ/X như\_vậy/X ./CH  
Tôi/P chưa/R bao\_giờ/P thực\_sự/A cảm\_thấy/V mệt\_mỏi/A vì/E nó/P ./CH  
Tôi/P đã/R có\_thể/R chống/V lại/R cảm\_giác/N đó/P trong/E nhiều/A năm\_qua/N ./CH  
Những/L năm/N gần\_đây/A tôi/P qua\_lại/V sân\_bay/N Tân\_Sơn\_Nhất/Np thường\_xuyên/A ./CH  
Trường/Np tôi/P từng/P nghiêm\_căm/V sử\_dụng/V điện\_thoại/N trong/E lớp/N ./CH  
Học\_sinh/N ngày\_nay/N có\_thể/R dễ\_dàng/A tiếp\_cận/V bài\_học/N và/Cc phương\_pháp/N giải/N bài\_tập/N ./CH  
Điện\_thoại/N thông\_minh/A là/V công\_cụ/N không\_thể/R đảo\_ngược/V trong/E cuộc\_sống/N ./CH

Hình 5.2. Kết quả gán nhãn 10 câu trong tập Test sử dụng thư viện VnCoreNLP

Kết quả đánh giá sau khi so sánh với thư viện VnCoreNLP trên tập Test:

Bảng 5.1: Đánh giá gán nhãn trên tập Test

	Accuracy	F1-score	Precision	Recall
HMM + Viterbi	0.61	0.49	0.6	0.48
VnCoreNLP	0.92	0.91	0.93	0.91

Có thể thấy VnCoreNLP cho kết quả khá tốt so với khi sử dụng mô hình HMM kết hợp với thuật toán Viterbi. Đây có thể do dữ liệu dùng để train cho mô hình này quá ít, chỉ có 50 câu. Nhóm xem xét kết quả trên tập Train của mô hình này thì thu được:

- F1-score: 0.99
- Precision: 0.98

- Recall: 0.99

Kết quả trên tập Train rất tốt nhưng tập Test lại khá tệ chứng tỏ mô hình đã bị overfitting. Nhóm tiến hành xem thêm kết quả đánh giá chi tiết của từng nhãn:

	precision	recall	f1-score	support
A	0.45	0.33	0.38	15
CH	1.00	0.86	0.92	14
Cc	1.00	0.29	0.44	7
E	0.86	0.50	0.63	12
L	1.00	1.00	1.00	3
M	1.00	0.25	0.40	4
N	0.45	0.93	0.61	15
Np	0.00	0.00	0.00	1
Nu	0.00	0.00	0.00	2
P	0.58	0.88	0.70	8
R	0.88	0.58	0.70	12
V	0.53	0.57	0.55	14
X	0.00	0.00	0.00	0

*Hình 5.3. Kết quả đánh giá chi tiết trên tập Test sử dụng Hidden Markov và Viterbi*

Các nhãn f1\_score khá lần lượt là nhãn “CH”:0.92, “P”:0.7, “R”:0.7. Các nhãn f1\_score dưới mức trung bình lần lượt là “A”: 0.45, “Np”:0 và “X”:0, “M”:0.4, “Cc”:0.44

## Chương 6: Kết luận

Trong đề tài này, nhóm đã áp dụng các kiến thức về xử lý ngôn ngữ tự nhiên để xây dựng bộ tách từ bằng thuật toán Longest Matching, sau đó thực hiện gán nhãn bán thủ công để có thể tạo ra ngữ liệu sẽ được sử dụng cho mô hình Hidden Markov để thực hiện gán nhãn từ loại Tiếng Việt.

Với việc tách từ thì việc áp dụng thuật toán Longest Matching khá đơn giản. Tuy nhiên nhược điểm của phương pháp này là không tách được các từ không có trong 2 bộ dữ liệu bi-grams và tri-grams, nhất là các từ tên riêng.

Nhóm bắt đầu gán nhãn từ loại với mô hình đơn giản nhất có thể, sử dụng các từ điển đếm số lượng đã được xây dựng trước. Kết quả thu được lúc này trên tập Test chỉ có 0.42. Sau khi xây dựng xong mô hình Hidden Markov và sử dụng kết hợp với thuật toán Viterbi để thực hiện dự đoán trên tập Test thì kết quả trên đã tăng lên đáng kể, từ 0.42 lên 0.61.

Nhóm kết luận mô hình đã bị overfitting do dữ liệu để train chỉ có 50 câu. Hướng phát triển của nhóm có thể là sẽ thu thập nhiều dữ liệu hơn từ nhiều nguồn khác nhau, bao quát nhiều ngữ cảnh hơn trong Tiếng Việt.

Bên cạnh đó nhóm sẽ tìm hiểu thêm những cách triển khai khác sử dụng gán nhãn 2 chiều (Bidirectional POS tagging). Gán nhãn 2 chiều yêu cầu biết được từ trước đó và từ tiếp theo trong ngữ liệu khi dự đoán nhãn của từ hiện tại. Gán nhãn 2 chiều sẽ cho ta biết thêm về nhãn thay vì chỉ biết từ trước đó. Vì đã học được cách triển khai phương pháp tiếp cận đơn hướng qua đề tài này, nhóm đã có nền tảng để triển khai các trình gán nhãn khác được sử dụng trong thực tế

## TÀI LIỆU THAM KHẢO

- [1] A. Taylor, M. Marcus, and B. Santorini, “The Penn Treebank: An Overview,” in *Treebanks: Building and Using Parsed Corpora*, A. Abeillé, Ed. Dordrecht: Springer Netherlands, 2003, pp. 5–22.
- [2] “VnCoreNLP: A Vietnamese natural language processing toolkit” *GitHub*. <https://github.com/vncorenlp/VnCoreNLP> (accessed Jan. 30, 2021).
- [3] “Speech and Language Processing.” <https://web.stanford.edu/~jurafsky/slp3/> (accessed Jan. 30, 2021).