ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO MÔN HỌC

XỬ LÝ NGÔN NGỮ TỰ NHIÊN

CS221.M12.KHCL

GIẢNG VIÊN HƯỚNG DẪN: **Th.S Nguyễn Trọng Chỉnh**

SINH VIÊN THỰC HIỆN:	HOÀN THÀNH CÔNG VIỆC
Nguyễn Đỗ Quang – 20520720 (Nhóm trưởng)	100%
Âu Thiên Phước - 19522050	100%
Phan Thanh Phong - 19522012	90%

TP. HỒ CHÍ MINH, 12/2021

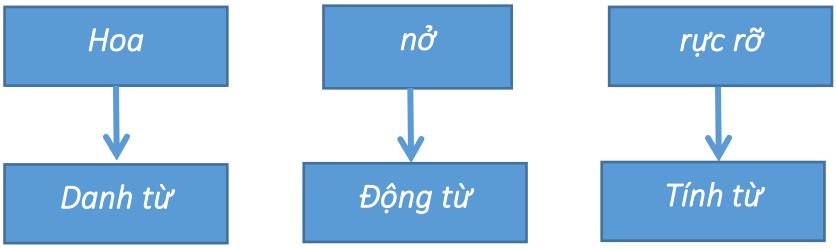
Tên đề tài: TÁCH TỪ LOẠI - GÁN NHÃN TỪ LOẠI SỬ DỤNG MÔ HÌNH HIDDEN MARKOV

Github project file: https://github.com/quangcrazymen/CS221.M12

Mục lục:

Phần I: Giới thiệu:

Trong xử lý ngôn ngữ tự nhiên, **gán nhãn từ loại** (**part-of-speech tagging**, hay **POS tagging**) là quá trình đánh dấu một từ trong văn bản (ngữ liệu) tương ứng với một **từ loại** nào đó (động từ, danh từ, tính từ…), dựa theo định nghĩa và bối cảnh văn phạm của từ đó.



Quá trình gán nhãn thường được thực hiện thủ công bằng tay, nay với sự phát triển của công nghệ, ta đã có thể tự động hóa quá trình này bằng cách sử dụng máy tính.

Tuy nhiên khi làm việc với bài toán này, ta sẽ phải đối mặt với hai vấn đề chính:

- Nhập nhằng (ambiguity): bất kì ngôn ngữ tự nhiên nào cũng có thể xuất hiện tính nhập nhằng, một từ có thể mang nhiều nghĩa. Ví dụ:

  "Con ruồi ***đậu*** mâm xôi ***đậu***"

  Từ "đậu" có lúc là **động từ** (hành động đậu lên một vật thể) hoặc có lúc là **danh từ** (tên của một loài thực vật).

- Trong thực tế, có nhiều từ không xuất hiện trong ngữ liệu huấn luyện (training corpus) nên khi xây dựng mô hình gán nhãn sẽ gặp nhiều khó khăn.

Độ chính xác của mô hình gán nhãn phụ thuộc vào hai yếu tố:

- Bản thân từ đó sẽ có xu hướng (xác suất lớn) về từ loại nào. Ví dụ: từ *"đậu"* có xu hướng là động từ nhiều hơn là danh từ (phụ thuộc vào ngữ liệu đang xét).
- Ngữ cảnh trong câu. Ví dụ "con **ruồi đậu** mâm **xôi đậu**". Từ *"đậu"* có xu hướng là động từ khi theo sau từ ***"ruồi"*** và từ *"đậu"* có xu hướng là danh từ khi theo sau từ ***"xôi"***.

Để gán nhãn từ loại, ta sử dụng phương pháp học có giám sát (supervised learning), cụ thể là xác suất liên hợp thường gọi là mô hình sinh mẫu (Generative model). Hidden Markov Model là một trong những mô hình thuộc phân nhóm này.

Vì thế trong đồ án lần này nhóm em sẽ thực hiện gán nhãn từ loại giữa trên mô hình Hidden Markov, được tính toán bằng thuật toán Viterbi, cũng như so sánh kết quả với thư viện VNcoreNLP, và đề xuất hướng cải thiện cho mô hình máy học này.

Phần II: Tách từ tiếng việt:

1. Phương pháp tách từ:

Trước khi đến với phần gán nhãn từ loại, ta cần thực hiện một bước vô cùng quan trọng, đó là tách từ loại. Khác với Tiếng Anh, khi mà hầu hết các từ được phân tách nhau rõ ràng bằng các khoảng trắng, một từ Tiếng Việt có thể được cấu tạo bởi **hai từ đơn trở lên** (Từ ghép ví dụ như: xác nhận, lâu đài, công chúa…). Bởi lẽ đó, muốn xử lí văn phạm ở mức sâu hơn về sau, ta sẽ phải tách được từ loại.

Có rất nhiều kĩ thuật và thuật toán có thể được sử dụng trong việc tách từ. Nhóm em quyết định sẽ sử dụng thuật toán đơn giản, cũng như dễ hiểu nhất: Longest matching.

Longest matching là thuật toán mà ta sẽ đi hết chiều dài văn bản và lần lượt so khớp từ, xem từ có tồn tại trong bộ từ điển dựng sẵn hay không.

2. Ví dụ:

Mã giả: (giả định trong trường hợp các từ ghép trong bộ ngữ liệu đều chỉ có tối đa 2 âm tiết)

Function Longest-Matching(*sentence, dict*):

Trong đó: **sentence** là câu đầu vào

dict là bộ từ điển

```
    result=[ ]
    curr_word = 0 (vị trí của từ đầu tiên i=0)
    while(curr_word + 1 != length (sentence)):
        if(word[curr_word]+word[curr_word+1] in dict)
            result.append(word[curr_word]+word[curr_word+1])
            curr_word+=2
        else:
            result.append(word[curr_word])
            curr_word+=1
    return result
```

Với câu sau:

"Chúng ta không chấp nhận bội chi"

Chạy thuật toán thử công:

Lượt so sánh	Từ cần so khớp	Có trong từ điển (in dict)	Kết quả (Result)
1	Chúng ta	có	['Chúng_ta']
2	không chấp	Không	['Chúng_ta', 'không']
3	chấp nhận	có	['Chúng_ta', 'không','chấp nhận']
4	bội chi	có	['Chúng_ta', 'không','chấp nhận','bội chi']

Vậy câu cần tách cho ra kết quả là: **Chúng_ta không chấp_nhận bội_chi**

Các trường hợp có thể thất bại là những câu có các từ mang nghĩa **nhập nhằng**, ta có thể sử dụng kĩ thuật matching từ theo chiều từ **phải sang trái** thay vì từ trái sang phải.

Như các ví dụ sau sẽ so sánh hai phương pháp so khớp, cho các câu sau:

A) Tay đua về ở vị trí thứ hai

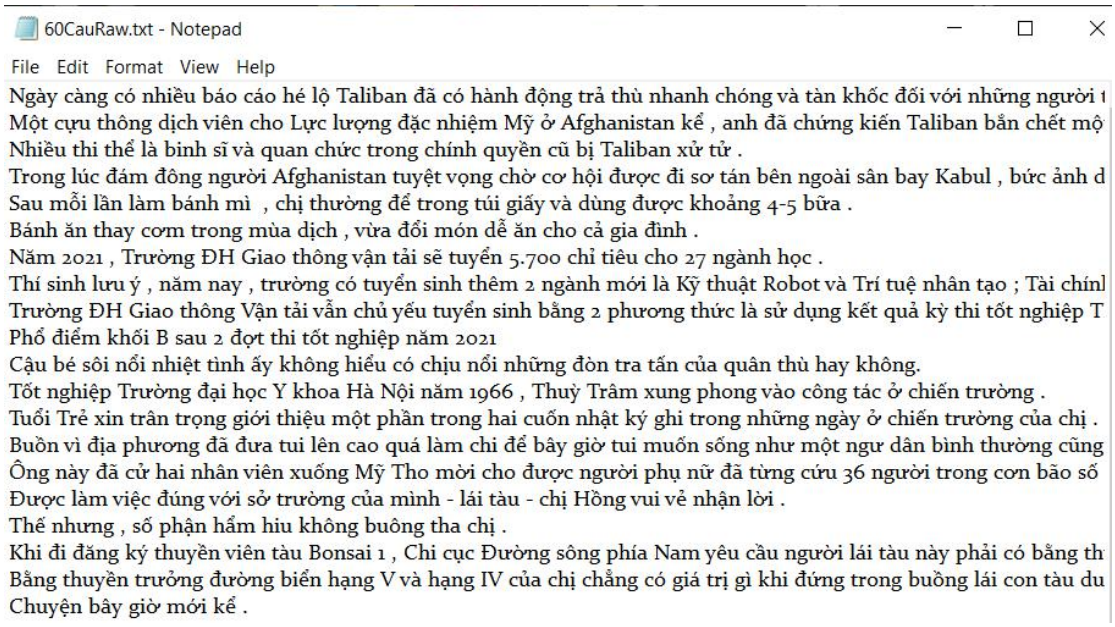
B) Tốc độ truyền thông tin cao

C) Sự thực hiện vẫn còn chưa thực sự phù hợp

So khớp từ trái qua phải	So khớp từ phải qua trái	Nhận xét
Tay_đua về ở vị_trí **thứ_hai**	Tay_đua về ở vị_trí **thứ_hai**	Cả hai trường hợp đều cho kết quả chưa chính xác, thứ hai ở đây là vị trí thứ hai chứ không phải ngày thứ hai =>***thứ hai ở đây không phải từ ghép***
Tốc_độ **truyền_thông** tin cao	Tốc_độ truyền thông_tin cao	Do matching theo chiều trái -> phải thuật toán sẽ gán cho truyền thông (là từ ghép) trước (do truyền thông có trong từ điển), ngược lại matching phải -> trái sẽ giải quyết được trường hợp này.
Sự_thực hiện vẫn còn chưa thực_sự phù_hợp	Sự thực_hiện vẫn còn chưa thực_sự phù_hợp	Tương tự như trường hợp số 2

3. Ngữ liệu thu thập:

- Ngữ liệu thu thập gồm 60 câu được trích ở nhiều nguồn đa dạng: Sách, báo điện tử, tiểu thuyết, lời bài hát, truyện cổ tích… được lưu trong 60CauRaw.txt
- Từ điển từ ghép 2 tiếng được tham khảo từ github[1] và lưu trong file TuGhep.txt

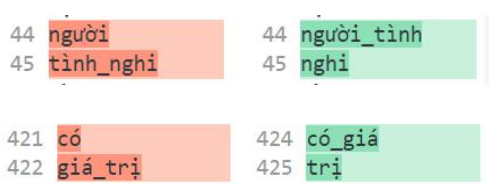

60CauRaw.txt - Notepad

File Edit Format View Help

Ngày càng có nhiều báo cáo hé lộ Taliban đã có hành động trả thù nhanh chóng và tàn khốc đối với những người t
Một cựu thông dịch viên cho Lực lượng đặc nhiệm Mỹ ở Afghanistan kể , anh đã chứng kiến Taliban bắn chết mộ
Nhiều thi thể là binh sĩ và quan chức trong chính quyền cũ bị Taliban xử tử .
Trong lúc đám đông người Afghanistan tuyệt vọng chờ cơ hội được đi sơ tán bên ngoài sân bay Kabul , bức ảnh d
Sau mỗi lần làm bánh mì , chị thường để trong túi giấy và dùng được khoảng 4-5 bữa .
Bánh ăn thay cơm trong mùa dịch , vừa đổi món dễ ăn cho cả gia đình .
Năm 2021 , Trường ĐH Giao thông vận tải sẽ tuyển 5.700 chỉ tiêu cho 27 ngành học .
Thí sinh lưu ý , năm nay , trường có tuyển sinh thêm 2 ngành mới là Kỹ thuật Robot và Trí tuệ nhân tạo ; Tài chính
Trường ĐH Giao thông Vận tải vẫn chủ yếu tuyển sinh bằng 2 phương thức là sử dụng kết quả kỳ thi tốt nghiệp T
Phổ điểm khối B sau 2 đợt thi tốt nghiệp năm 2021
Cậu bé sôi nổi nhiệt tình ấy không hiểu có chịu nổi những đòn tra tấn của quân thù hay không.
Tốt nghiệp Trường đại học Y khoa Hà Nội năm 1966 , Thuỳ Trâm xung phong vào công tác ở chiến trường .
Tuổi Trẻ xin trân trọng giới thiệu một phần trong hai cuốn nhật ký ghi trong những ngày ở chiến trường của chị .
Buồn vì địa phương đã đưa tui lên cao quá làm chi để bây giờ tui muốn sống như một ngư dân bình thường cũng
Ông này đã cử hai nhân viên xuống Mỹ Tho mời cho được người phụ nữ đã từng cứu 36 người trong cơn bão số
Được làm việc đúng với sở trường của mình - lái tàu - chị Hồng vui vẻ nhận lời .
Thế nhưng , số phận hẩm hiu không buông tha chị .
Khi đi đăng ký thuyền viên tàu Bonsai 1 , Chi cục Đường sông phía Nam yêu cầu người lái tàu này phải có bằng th
Bằng thuyền trưởng đường biển hạng V và hạng IV của chị chẳng có giá trị gì khi đứng trong buồng lái con tàu du
Chuyện bây giờ mới kể .

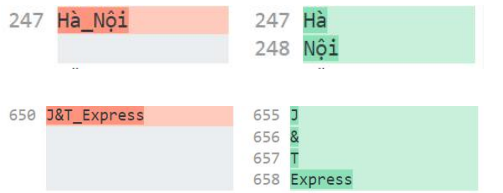
4. Đánh giá trên ngữ liệu thu thập được:

Các trường hợp mà thuật toán cho ra kết quả sai là những trường hợp nhập nhằng:



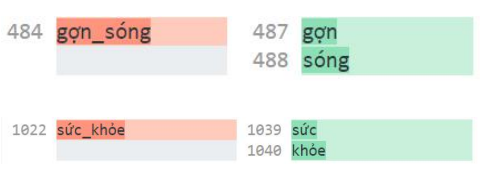
Có thể được giải quyết bằng cách so khớp từ **phải sang trái** (thay vì trái sang phải)

Những trường hợp tên địa danh cũng như tên riêng: Hà Nội, Kiên Giang, Hà Nam…



Có thể được giải quyết bằng cách sử sụng **Regular Expression** (do các tên riêng luôn có chữ cái đầu tiên được viết hoa)

Những trường hợp từ không có trong từ điển dựng sẵn: quen quen, giải hạn…



Có thể được giải quyết bằng cách **thêm** những từ này vào bộ từ điển tự dựng

Longest-Matching	vnCoreNLP
Với thuật toán Longest-Matching ta có được độ chính xác khả quan **98.12%** Link: https://www.diffchecker.com/TdrqglCx	So sánh với thư viện vnCoreNLP với độ chính xác là: **99.06%** Link: https://www.diffchecker.com/IZ1NtwL3

Thư viện **VnCoreNLP** đã giải quyết được một số trường hợp nhập nhằng, cũng như nhận diện được hầu hết các tên riêng.

5.Kết luận:

Ưu và nhược điểm của thuật toán Longest Matching:

Ưu điểm	Nhược điểm
Đơn giản, dễ cài đặt, ý tưởng dễ hiểu, độ chính xác tương đối cao.	Chưa xử lí được các trường hợp nhập nhằng như đã đề cập ở trên. Không xử lí được những từ không tồn tại trong từ điển dựng sẵn.

=> Muốn thuật toán chạy tốt thì cần bộ từ điển lớn bao quát được hết các từ trong bộ ngữ liệu cần tách từ.

III. Gán Nhãn từ loại tiếng việt:

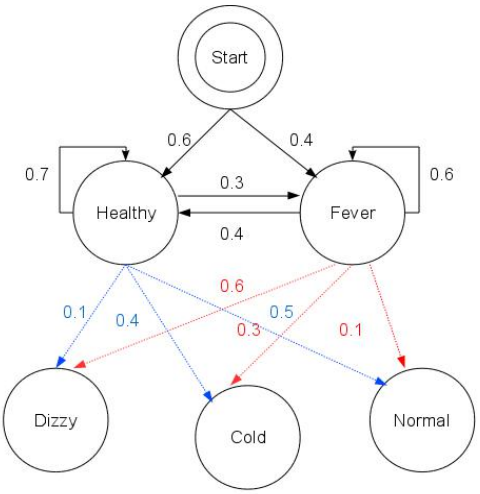
1. Phương pháp gán nhãn

1. 1 Mô hình Markov ẩn (Hidden markov):

Mô hình Hidden Markov là một mô hình máy học được sử dụng để tính toán xác suất của các chuỗi.

Mô hình Hidden Markov được áp dụng cho rất nhiều lĩnh vực có tính toán chuỗi xác suất, điển hình như các lình vực khoa học tự nhiên: sinh học trong việc giải mã chuỗi gene (tin sinh học), nhận dạng pattern: điển hình được sử dụng trong xử dụng tiếng nói và ngôn ngữ tự nhiên.

Đồ án lần này nhóm em sẽ áp dụng mô hình Hidden Markov để áp dụng cho 1 phần quan trọng trong sử lý ngôn ngữ tự nhiên đó là **gán nhãn từ loại (Pos-tagging)**.



Ảnh minh họa mô hình Hidden Markov

Chuỗi Markov vô cùng hữu dụng khi sử dụng để tính toán xác suất của một chuỗi các sự kiện quan sát được. Trong nhiều trường hợp ta sẽ quan tâm đến trạng thái ẩn(hidden). Ví dụ như trong bài toán gán nhãn tất nhiên ta sẽ không thể thấy được ngay nhãn ở đoạn văn bản đó, thay vào đó mà ta sẽ xét từng từ và tùy thuộc vào ngữ cảnh mà ta suy luận và gán nhãn cho chúng.

Các thành phần:

$Q=q_1q_2...q_N$	Là tập N các trạng thái
$A=a_{11}....a_{ij}...a_{NN}$	Ma trận chuyển trạng thái A(transition probability matrix)
$O=o_1o_2...o_T$	Là chuỗi T tất cả các quan sát
$B=b_i(o_t)$	Ma trận phát xạ B(emission probability matrix)
$\pi = \pi_1,\pi_2,...,\pi_N$	Là xác suất trạng thái ban đầu

Mô hình Hidden Markov đặt ra hai giả định:

Thứ nhất là: xác suất của một trạng thái nhất định chỉ phụ thuộc vào xác suất của trạng thái trước đó.

Giả định Markov(Markov assumption):

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

Thứ hai là: xác suất đầu ra của của quan sát o_1 chỉ phục thuộc vào trạng thái mà ở đó sản sinh ra q_i chứ không phụ thuộc vào bất kì trạng thái hay quan sát nào khác.

Đầu ra độc lập(Output Independence):

$$P(o_i|q_1,....,q_i,...,q_T,+o_1,....,o_i,....,o_T) = P(o_i|q_i)$$

1.2. <u>Các thành phần của mô hình hidden Markov và công thức sử dụng mô hình để gán nhãn</u>

Theo đó một mô hình Hidden Markov có 2 thành phần chính, xác suất A và B

Xác suất chuyển tiếp A(transition probabilities): là ma trận ở đó mỗi giá trị(MLE) của mỗi nhãn được tính toán khi đã biết xác suất của nhãn trước đó

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

Ví dụ như số lần danh từ chuyển sang động từ là 80 và số lượng danh từ là 100

Ta có hợp lí cực đại(Maximum likelihood estimation)

$$P(\text{động từ}|\text{danh từ}) = \frac{C(\text{danh từ},\text{động từ})}{C(\text{danh từ})} = 0.8$$

Xác suất phát xạ B(emission probabilities): là ma trận mà ở đó mỗi giá trị(MLE) xác suất mỗi nhãn của mỗi từ(như là đậu) ứng với mỗi nhãn

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

Ví dụ như nhãn động từ xuất hiện 8 lần trong đó từ đậu có 6 lần xuất hiện dưới nhãn động từ

Ta có hợp lí cực đại(Maximum likelihood estimation)

$$P(\text{“đậu”}|\text{động từ}) = \frac{C(\text{động từ,"đậu"})}{C(\text{động từ})} = 0.75$$

Như vậy theo như hai giả định bên trên kết hợp với 2 xác suất A và B ta có được công thức

$$\hat{t}_{1:n} = \underbrace{argmax}_{t_1 \dots t_n} P(t_1 \dots t_n | w_1 \dots w_n) = \underbrace{argmax}_{t_1 \dots t_n} \prod_{i=1}^{n} \overbrace{P(w_i|t_i)}^{emission} \overbrace{P(w_i|t_i)}^{transition}$$

Chi tiết tham khảo sách Speech and Language Processing(156-158)[2]

2. Ví dụ và áp dụng thuật toán viterbi vào để tính toán

Thuật toán viterbi là một thuật toán quy hoạch động, thuật toán thực hiện việc tính toán để tìm ra xác xuất lớn nhất mà một chuỗi các trạng thái ẩn có thể đạt được (Viterbi path), đây là kết quả dựa trên các trạng thái quan sát được trong chuỗi các sự kiện quan sát được, đặc biệt thuật toán được sử dụng chủ yếu để tính toán trong mô hình Hidden Markov.

Ví dụ với bộ ngữ liệu được cho như sau:

a) Em/**N** tập/**V** đánh/**V** đàn/**N**

b) Đàn/**N** chó/**N** chạy/**V** ngoài/**E** sân/**N**

c) Sân/**N** trường/**N** rợp/**A** bóng/**N** cây/**N**

d) Cây/**N** cao/**A** bóng/**N** cả/**A**

Gán nhãn cho câu sau:

Đàn chó đánh đàn giữa sân trường

Ma trận A: (Transition matrix):

	A	E	N	V
<S>	0	0	4	0
A	0	0	3	0
E	0	0	0	1
N	2	1	3	1
V	0	0	2	1

Lapplace smoothing

	A	E	N	V
<S>	1	1	5	1
A	1	1	4	1
E	1	1	1	2
N	3	2	4	2
V	1	1	3	2

Máy tính tính xác suất

	A	E	N	V
--s--	0.125000	0.125000	0.625000	0.125000
A	0.125000	0.125000	0.375000	0.125000
E	0.166667	0.166667	0.333333	0.166667
N	0.250000	0.062500	0.250000	0.187500
V	0.125000	0.250000	0.250000	0.250000

Ma trận B: (Emission matrix):

	em	tập	đánh	đàn	chó	chạy	ngoài	sân	trường	rợp	bóng	cây	cao	cả
A										1			1	1
E							1							
N	1			2	1			2	1		2	2		
V		1	1			1								

Laplace smoothing

	em	tập	đánh	đàn	chó	chạy	ngoài	sân	trường	rợp	bóng	cây	cao	cả
A	1	1	1	1	1	1	1	1	1	2	1	1	2	2
E	1	1	1	1	1	1	2	1	1	1	1	1	1	1
N	2	1	1	3	2	1	1	3	2	1	3	3	1	1
V	1	2	2	1	1	2	1	1	1	1	1	1	1	1

Máy tính tính toán xác suất

	em	tập	đánh	đàn	chó	chạy	ngoài	sân	trường	rợp	bóng	cây	cao	cả
N	0.074074	0.037037	0.037037	0.111111	0.074074	0.037037	0.037037	0.111111	0.074074	0.037037	0.111111	0.111111	0.037037	0.037037
V	0.052632	0.105263	0.105263	0.052632	0.052632	0.105263	0.052632	0.052632	0.052632	0.052632	0.052632	0.052632	0.052632	0.052632
E	0.058824	0.058824	0.058824	0.058824	0.058824	0.058824	0.117647	0.058824	0.058824	0.058824	0.058824	0.058824	0.058824	0.058824
A	0.052632	0.052632	0.052632	0.052632	0.052632	0.052632	0.052632	0.052632	0.052632	0.105263	0.052632	0.052632	0.105263	0.105263

Với 2 ma trận A và B ta áp dụng thuật toán Viterbi

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 ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Mã giả tham khảo sách Speech and Language Processing(trang 159)[2]

Ta có thể chi ra thành 2 pha để tính như sau (thay vì đệ quy) như sau:

```python
def viterbi_forward(A, B, corpus, best_probs, best_paths, vocabs_dict):
    num_tags = best_probs.shape[0]

    for i in range(1, len(corpus)):

        for j in range(num_tags):
            best_prob_i = float('-inf')
            best_path_i = None

            for k in range(num_tags):
                index = vocabs_dict[corpus[i]]
                prob = best_probs[k, i - 1] + math.log(A[k, j - 1]) + math.log(B[j - 1, index])

                if prob > best_prob_i:
                    best_prob_i = prob
                    best_path_i = k

            best_probs[j, i] = best_prob_i
            best_paths[j, i] = best_path_i

    return best_probs, best_paths
```

```python
def viterbi_backward(best_probs, best_paths, states):
    m = best_paths.shape[1]
    z = [None] * m
    pred = [None] * m

    best_prob_for_last_word = float('-inf')
    num_tags = best_probs.shape[0]

    for k in range(num_tags):
        if best_probs[k, m - 1] > best_prob_for_last_word:
            best_prob_for_last_word = best_probs[k, m - 1]
            z[m - 1] = k

    pred[m - 1] = states[z[m - 1]]
    for i in range(m - 1, -1, -1):
        z[i - 1] = best_paths[z[i], i]
        pred[i - 1] = states[z[i - 1]]
    return pred
```

Mã nguồn được tham khảo từ github[1]

Với thuật toán như trên ta tính toán được hai ma trận là ma trận xác suất tốt nhất (best_probs) cũng như ma trận đường đi tốt nhất(best_path)

Xác suất âm do đã được chuyển đổi thành *xác suất log để tận dụng khả năng tính toán của máy tính [3]*

<s>	đàn	chó	đánh	đàn	<unk> *	sân	trường
A	-2.913	-3.152	-4.156	-8.156	-13.155	-16.164	-19.165
E	-1.512	-2.165	-7.895	-10.366	-15.156	-18.265	-20.355
N	-10.545	-11.135	-12.125	-13.156	-14.156	-17.156	-18.623
V	-5.131	-6.456	-8.26	-12.156	-14.216	-16.156	-23.456

Xác suất lớn nhất (Best_probability)

Ma trận xác suất

Bắt đầu backtrack từ vị trí này

<s>	đàn	chó	đánh	đàn	<unk> *	sân	trường
A	(-)	2	3	1	2	2	0
E	(-)	2	1	1	2	0	3
N	(+)	2	2	3	1	1	2
V	(-)	2	2	0	2	1	0

Ma trận đường đi

Ghi chú: *Các dòng của ma trận tính từ 0*
→ *Là đường đi sau khi đã hoàn thành backtrack*

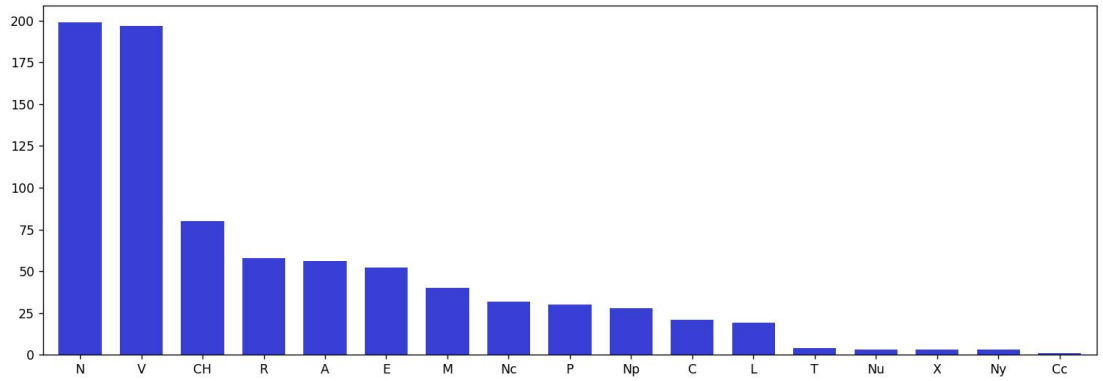
Câu cần gán nhãn là: **đàn/N chó/N đánh/V đàn/N giữa/E sân/N trường/N**

3. Ngữ liệu thu thập

- Ngữ liệu gồm 60 câu đã tách từ thủ công ở bên trên và đem đi gán nhãn,
- 50 câu sẽ được lấy để cho vào tập train (50CauGanNhanTrain_Gold.txt)
- 10 câu còn lại sẽ được cho và tập test_gold(10CauGanNhanTest_Gold.txt)
- 10 câu cần dự đoán được chứa trong 10CauTest.txt
- 5000 câu đã được gán nhãn, được nhóm tổng hợp từ dự án VLSP2016 (được lưu trong LargeVLSPData.txt)
- Từ điển được tham khảo từ ***github[1]*** (được lưu trong file vocabs.txt)

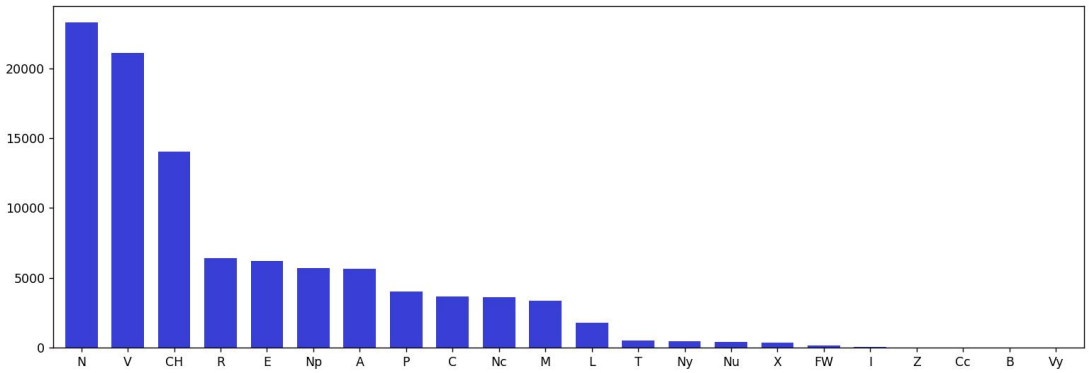
Bộ 50 câu đã gán nhãn: (826 từ)

	N	V	CH	R	A	E	M	Nc	P	Np	C	L	T	Nu	X	Ny	Cc	Total
0	199	197	80	58	56	52	40	32	30	28	21	19	4	3	3	3	1	826



Bộ 5000 câu đã gán nhãn:(100957 từ)

	N	V	CH	R	E	Np	A	P	C	Nc	M	L	T	Ny	Nu	X	FW	I	Z	Cc	B	Vy	Total
0	23286	21101	14028	6438	6204	5726	5668	4039	3675	3637	3363	1774	502	489	406	344	172	58	32	13	1	1	100957



Sau buổi thuyết trình thì nhóm em có xem xét lại bộ dữ liệu, ngoài việc bộ train không đồng nhất (chỗ thì là khoảng trắng chỗ thì là '_'), thì có vẻ là nhóm đã load nhầm data nên kết quả cho ra khá bất thường. Nhóm đã dành thời gian để chỉnh sửa, và chăm chút hơn cho bộ dữ liệu 50 câu train.

Bộ nhãn được sử dụng VLSP2013 (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, nắm, nhúm, 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; dăm, vài, mươi
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, chăng, 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

4. Kết quả đánh giá trên ngữ liệu thu thập được:

	precision	recall	f1-score	support
--s--	0.00	0.00	0.00	1
C	0.00	0.00	0.00	0
CH	0.83	1.00	0.90	19
E	0.78	1.00	0.88	7
L	0.00	0.00	0.00	0
M	0.33	1.00	0.50	3
N	0.50	0.46	0.48	48
Nc	0.40	1.00	0.57	2
Np	0.00	0.00	0.00	0
Ny	0.00	0.00	0.00	0
P	0.00	0.00	0.00	0
R	0.27	1.00	0.43	3
V	0.97	0.40	0.57	98
accuracy			0.52	181
macro avg	0.29	0.42	0.31	181
weighted avg	0.79	0.52	0.58	181

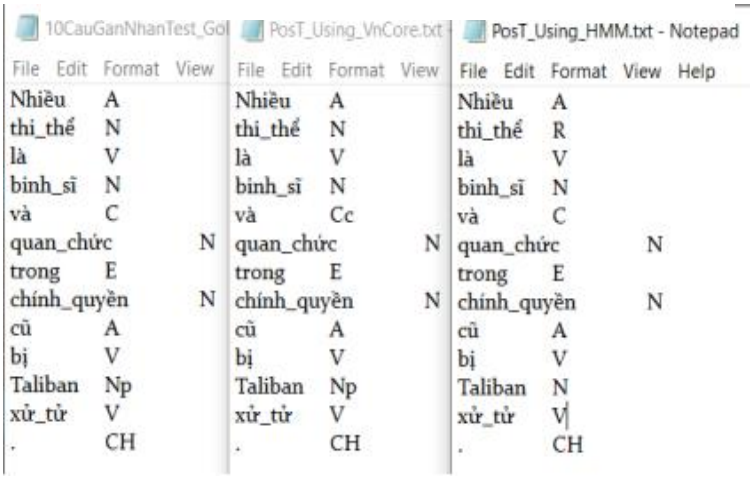
Đánh giá trên ngữ liệu thu thập được gồm 50 câu gán nhãn cho ra f1-score chỉ khoảng 58%

	precision	recall	f1-score	support
--s--	0.00	0.00	0.00	1
A	0.70	1.00	0.82	7
C	1.00	1.00	1.00	7
CH	0.96	0.92	0.94	24
E	0.89	0.89	0.89	9
L	1.00	1.00	1.00	1
M	0.67	1.00	0.80	6
N	0.80	0.71	0.75	49
Nc	0.80	1.00	0.89	4
Np	0.50	0.54	0.52	13
Ny	0.00	0.00	0.00	0
P	1.00	1.00	1.00	5
R	0.82	0.82	0.82	11
V	0.95	0.86	0.90	44
accuracy			0.82	181
macro avg	0.72	0.77	0.74	181
weighted avg	0.84	0.82	0.83	181

Đánh giá trên ngữ liệu thu thập được gồm 5000 câu gán nhãn cho ra f1-score lên đến 83%

	precision	recall	f1-score	support
A	0.80	1.00	0.89	8
C	0.43	0.75	0.55	4
CH	1.00	1.00	1.00	23
Cc	0.00	0.00	0.00	4
E	1.00	0.82	0.90	11
L	1.00	1.00	1.00	1
M	0.89	0.89	0.89	9
N	0.82	0.84	0.83	43
Nc	0.20	0.50	0.29	2
Np	0.71	0.62	0.67	16
Ny	1.00	0.75	0.86	4
P	1.00	1.00	1.00	5
R	0.64	1.00	0.78	7
V	1.00	0.91	0.95	44
accuracy			0.85	181
macro avg	0.75	0.79	0.76	181
weighted avg	0.86	0.85	0.85	181

Kết quả sử dụng vnCoreNLP với độ chính xác 85%

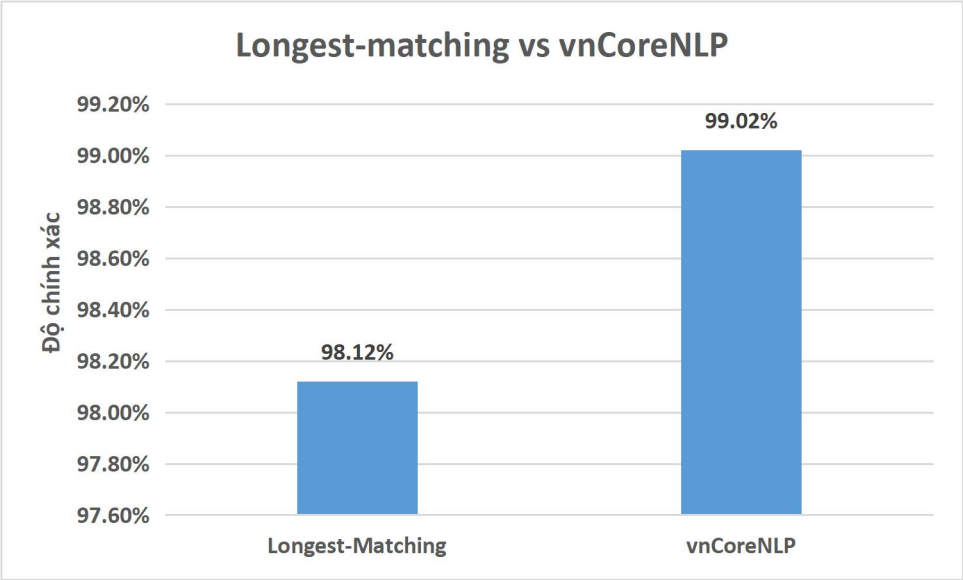


Kết quả được lưu vào các file text

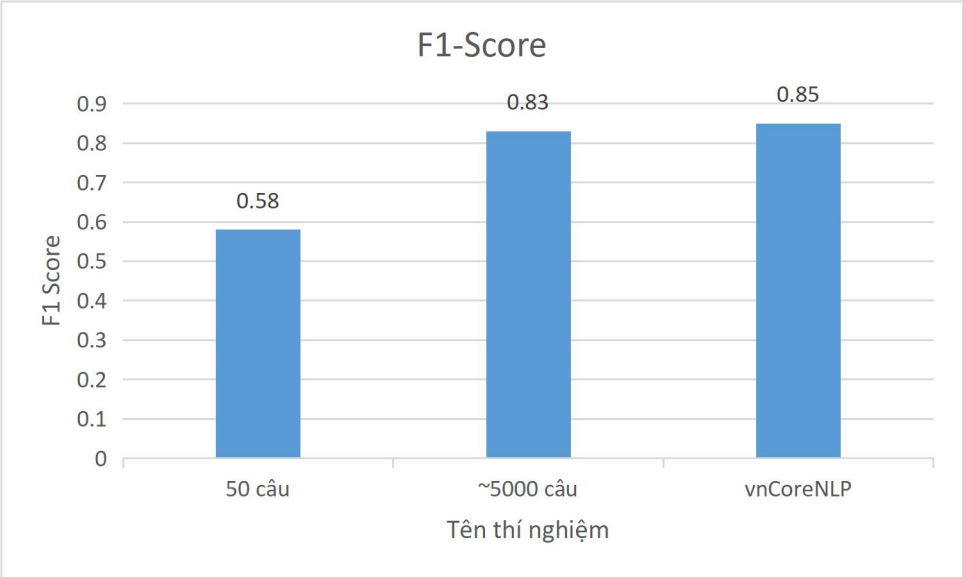
Phần IV: Kết luận và đánh giá kết quả tổng quát

1. **Về phần tách từ** loại sử dụng thuật toán longest-matching cũng như thư viện vnCoreNLP cho kết quả với độ chính xác xấp xỉ bằng nhau.

Longest-Matching	vnCoreNLP
98.12%	99.02%



2. **Về phần gán nhãn** từ ta có thể thấy khi kích thước, và độ đa dạng của bộ dữ liệu train tăng lên ta thấy có sự cải thiện rõ rệt khi, dự đoán thực sự trên tập test. (độ chính xác tăng thêm 27%).



Phần V: Rút kinh nghiệm và cải thiện trong tương lai

1. Rút kinh nghiệm:

Làm bộ dữ liệu và load dữ liệu vào model cẩn thận hơn.

Khi trình bày thuyết trình cần lấy ví dụ mà nhóm tự xây dựng.

2. Cải thiện:

Thu thập bộ dữ liệu lớn hơn, tìm ra những phương pháp mới, áp dụng kĩ thuật tích hợp luật để có thể cải thện và nâng cao chất lượng của ứng dụng gán nhãn Tiếng Việt.

Tư liệu tham khảo:

[1] https://github.com/18520339/vietnamese-pos-tagging

[2] https://web.stanford.edu/~jurafsky/slp3/

[3] https://math.stackexchange.com/questions/892832/why-we-consider-log-likelihood-instead-of-likelihood-in-gaussian-distribution

Hidden Markov Model: https://en.wikipedia.org/wiki/Hidden_Markov_model

Viterbi Algorithm: https://en.wikipedia.org/wiki/Viterbi_algorithm