ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

----------o0o----------

BÁO CÁO ĐỒ ÁN
MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Đề tài: Gán nhãn từ loại tiếng Việt

GVHD: Nguyễn Trọng Chỉnh

Nhóm sinh viên thực hiện:

1. Nguyễn Hoàng Gia 20520478
2. Lê Ngọc Mỹ Trang 20520817

Thành phố Hồ Chí Minh, tháng 2 năm 2023

MỤC LỤC

I.Phân công

	20520478	20520817
Thu thập dữ liệu	Không tham gia trực tiếp	Có tham gia
Tách từ và gán nhãn thủ công	Có tham gia	Có tham gia
Hiện thực hóa bằng python	Có tham gia	Không tham gia trực tiếp
Đánh giá và nhận xét	Có tham gia	Có tham gia
Slide	Không tham gia trực tiếp	Có tham gia
Thuyết trình	Có tham gia	Có tham gia
Báo cáo	Có tham gia	Có tham gia
Đánh giá tiến độ hoàn thành	50% (Hoàn thành 100% công việc được phân công)	50% (Hoàn thành 100% công việc được phân công)

II.Giới thiệu đề tài

1. Tổng quan đề tài

Gán nhãn từ loại tuy không phải là giải pháp cho toàn bộ các vấn đề của xử lý ngôn ngữ tự nhiên, nhưng cũng là một trong những bước quan trọng, tiên quyết, có vai trò nền tảng của xử lý ngôn ngữ tự nhiên, giúp đơn giản hóa bài toán trước khi phân tích sâu vào các phần phức tạp khác.

Có 2 bước quan trọng trong đề tài của nhóm về gán nhãn từ loại tiếng Việt:

- Tách từ là một quá trình xử lý nhằm mục đích xác định ranh giới của các từ trong câu.
- Gán nhãn từ loại là quá trình xác định từ loại tương ứng của một từ trong văn bản (ngữ liệu) dựa theo định nghĩa và bối cảnh văn phạm của từ đó.

Gán nhãn từ loại giúp làm rõ nghĩa của từ, cho thấy khả năng kết hợp của các từ trong câu giúp tăng hiệu quả cho việc phân tích cú pháp.

2. Khái quát bài toán

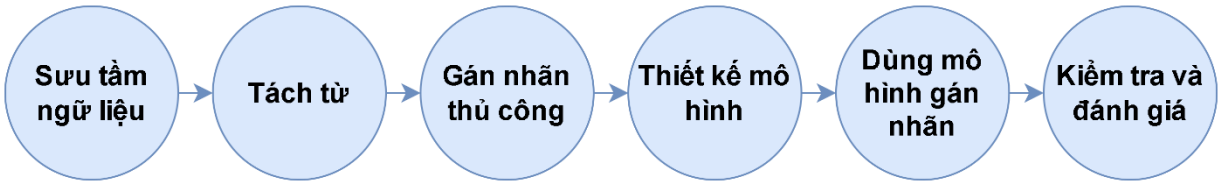
	Input Một câu tiếng Việt	**Output** Câu input kèm nhãn của từng từ hoặc cụm từ trong câu
Ví dụ	Cho em hỏi còn môn lập trình hướng đối tượng có nên đăng ký không, vì em còn code yếu?	Cho/**V** em/**N** hỏi/**V** còn/**C** môn/**N** lập_trình/**V** hướng/**N** đối_tượng/**N** có/**V** nên/**C** đăng_ký/**V** không/**R** ,/**CH** vì/**E** em/**N** còn/**C** code/**X** yếu/**A** ?/**CH**

- Hướng tiếp cận: trong bài này, nhóm đã sử dụng các hướng tiếp cận sau

  + Tách từ: Longest Matching

  + Gán nhãn từ loại: Hidden Markov kết hợp với thuật toán Viterbi

- Quy trình thực hiện



III.Ngữ liệu thu thập

- Ngữ liệu được thu thập từ forum UIT, mục giải đáp thắc mắc cho các bạn sinh viên

Figure 1 - Forum UIT

- Thống kê:
  + Số lượng có 60 câu gồm câu hỏi của sinh viên và câu trả lời đến từ nhà trường hoặc các sinh viên khác.
  + Số từ nhiều nhất trong một câu là 66, số từ ít nhất trong một câu là 11.
- Ví dụ:

Bất kỳ xử phạt nào cũng có lý do và đều được thông báo riêng với account vi phạm hoặc thông qua bảng phong thần - trừ việc xóa bài là lý do bị ẩn đi.
BHTCNPM với các bài training luôn luôn đặt chất lượng và kiến thức của sinh viên lên hàng đầu, đồng thời với nội dung kiến thức dàn trải từ cơ bản đến nâng cao, cùng với các trainer dồi dào kinh nghiệm chuyên môn sẽ giúp cho tất cả các bạn đều có thể ôn tập và thi thật tốt.
Do trong quá trình nâng cấp gặp một số lỗi kỹ thuật nên hiện tại BQT đang xử lý, diễn đàn vẫn đang hoạt động ở phiên bản cũ, dự kiến trong tuần này sẽ khắc phục các lỗi và nâng cấp phiên bản mới.

- Thách thức: khác với câu trả lời thường được đảm bảo ngữ pháp văn viết từ phía admin, dữ liệu phần lớn ở dạng văn nói, có nhiều từ không đúng ngữ pháp cũng như các từ chuyên ngành và tên riêng cả tiếng Việt lẫn tiếng Anh, còn một số từ viết tắt, viết sai chính tả,… ở phần lớn các câu hỏi và câu trả lời của sinh viên khiến cho việc gán nhãn thủ công không đảm bảo được tính chính xác, gây khó khăn cho quá trình huấn luyện của máy.

“**BHTCNPM** với các bài **training** luôn luôn đặt chất lượng và kiến thức của sinh viên lên hàng đầu, đồng thời với nội dung kiến thức dàn trải từ cơ bản đến nâng cao, cùng với các **trainer** dồi dào kinh nghiệm chuyên môn sẽ giúp cho tất cả các bạn đều có thể ôn tập và thi thật tốt.”

IV.Tách từ

1. Longest Matching

1.1. Giới thiệu

Thuật toán Longest Matching là một thuật toán so khớp tham lam, bằng cách so khớp các tiếng trong câu từ trái qua phải với các tiếng có trong từ điển, các tiếng đầu tiên dài nhất có thể xuất hiện trong từ điển sẽ được tách ra thành một từ.

1.2. Các bước tiến hành

Yêu cầu đầu tiên để thực hiện thuật toán Longest Matching là xây dựng cho mình 1 bộ từ điển.

Nguồn từ điển: Thư viện từ điển Underthesea

Tạo từ điển: từ điển sao khi được tải xuống sẽ được tách thành 2 file từ điển nhỏ hơn :

- bi_grams: chứa những từ có 2 tiếng tạo thành: 27938 từ.
- tri_grams: chứa những từ có 3 tiếng tạo thành: 3402 từ.

Các bước tiến hành thuật toán:

- B1: Tại vị trí hiện tại thuật toán tiến hành kiểm tra tiếng hiện tại và 2 tiếng liền kề với nó kết hợp có tạo thành một chữ có trong từ điển tri_grams hay không. Nếu có, thực hiện tách 3 tiếng đó ra khỏi câu và xem 3 tiếng đó là 1 từ và tăng vị trí hiện tại lên 3.
- B2: Nếu không, thực hiện kiểm tra tiếng ở vị trí hiện tại và tiếng liền kề với nó kết hợp có tạo thành một chữ có trong từ điển bi_grams hay không. Nếu có, thực hiện tách 2 tiếng đó ra khỏi câu và xem 2 tiếng đó là 1 từ tăng vị trí hiện tại lên 2.
- B3: Nếu không, tiếng hiện tại được tách ra khỏi câu và xem đó là 1 từ và tăng vị trí hiện tại lên 1.
- B4: Thuật toán tiến hành lặp lại các bước trên cho đến khi đã xét hết tất cả các tiếng trong câu.

1.3. Đánh giá thuật toán

Ưu điểm: Longest Matching là một thuật toán đơn giản, độ phức tạp hợp lý, không yêu cầu dữ liệu huấn luyện trước.

Nhược điểm: Thuật toán quá phụ thuộc vào từ điển được cung cấp. Không xử lý được những vấn đề nhập nhằng của ngôn ngữ vì chỉ chú trọng vào việc so khớp với từ điển.

VD: Con ngựa đá con ngựa đá

- Output theo Longest Matching: Con, ngựa_đá, con, ngựa_đá.
- Output đúng theo mong muốn: Con, ngựa, đá, con, ngựa_đá.

2. Các bước tiến hành tách từ

Nhóm thực hiện tách từ thủ công dựa trên thư viện online VLSP: Với mỗi tiếng trong câu, nhóm thực hiện tra từ điển từ mà chúng em cho là có trong từ điển. Nếu từ đó có trong từ điển thì thực hiện tách từ và chuyển sang tiếng kế tiếp, còn không thì giảm số tiếng xuống và tiếp tục tra từ điển tiếng đó.

Nhóm kiểm tra lại bằng cách sử dụng thuật toán Longest matching và Vncore tokenize để tạo ra 2 file tách từ khác nhau, sau đó dùng excel để lọc những điểm khác giữa chúng và

file tách từ thủ công. Cuối cùng, tại những vị trí đó, nhóm tra lại các cụm từ để chỉnh sửa lần cuối.

Bộ dữ liệu mới được lưu với tên manual_tokens.txt chứa kết quả tách từ chuẩn nhất cho 60 câu đã thu thập:

- Các âm tiết trong từ ghép sẽ được nối bằng dấu _ .
- Mỗi dòng là một từ.
- Mỗi câu được ngăn cách với nhau bởi 1 dòng trống.
- Số lượng câu: 60 câu.
- Số lượng từ ghép: 339 từ.

3. Đánh giá kết quả tách từ

	Longest Matching	VnCoreNLP	Underthesea
Accuracy	**0.491239**	**0.902115**	**0.883384**
Precision	**0.453552**	**0.904478**	**0.876081**
Recall	**0.244838**	**0.893805**	**0.896755**
True Positive	**83**	**303**	**304**
False Positive	**100**	**32**	**43**
Total True	**813**	**1493**	**1462**
Total Errors	**1005**	**136**	**156**

Bảng 1. Bảng kết quả tách từ

Bảng đánh giá trên có được nhờ việc sử dụng kết quả tách từ thủ công từ bước trên làm chuẩn và thực hiện việc so sánh kết quả đó với các kết quả tách từ từ các thuật toán khác.

Nhìn vào bảng trên, có thể thấy thuật toán Longest Matching cho kết quả thấp hơn thuật toán của 2 thư viện còn lại. Đây phần lớn là do 2 bộ từ điển bi_grams và tri_grams cung cấp cho thuật toán là chưa đủ tốt với dữ liệu của chúng em, một phần là do dữ liệu chúng em được lấy từ forum của trường nên phần lớn các câu đưa vào ở dạng văn nói, và mang tính chủ quan của người nói. Độ phủ (Recall) của thuật toán này thấp hơn nhiều so với Accuracy và Precision chứng tỏ thuật toán này hoạt động chưa được tốt.

Kết quả tách từ bằng các phương pháp khác nhau:

- Tách từ thủ công: 339 từ ghép.
- Tách từ bằng thuật toán Longest Matching: 187 từ ghép.

- Tách từ bằng thư viện VnCoreNLP: 342 từ ghép.
- Tách từ bằng thư viện Underthesea: 357 từ ghép.

Một trường hợp tách từ cụ thể sử dụng cả 4 phương pháp:

	Ví dụ
Tách thủ công	Bất_kỳ xử_phạt nào cũng có lý_do
Longest Matching	Bất_kỳ xử_phạt nào cũng có_lý do
VNCoreNLP	Bất_kỳ xử_phạt nào cũng có lý_do
Underthesea	Bất_kỳ xử_phạt nào cũng có lý_do

Có thể thấy ở trường hợp trên, tại vị trí tiếng "có" thuật toán Longest Matching đã xem từ "có_lý" là một từ hợp lệ dựa vào bộ từ điển được cung cấp. Nhưng trong trường hợp này, từ "lý_do" mới là từ thích hợp về nghĩa ở đây.

V.Tạo ngữ liệu

Với dữ liệu tách từ có trong file manual_tokens.txt nhóm chúng em thực hiện việc gán nhãn thủ công với quy tắc:

- Các từ tiếng Anh, các từ viết tắt, hay các từ đặt biệt (@gmail.com, ...) nhóm đã gán nhãn X (không có nhãn).
- Nhãn của các từ được gán theo việc tra cứu từ điển VLSP.

Sau đó, chúng em lại thực hiện gán nhãn cho file manual_tokens.txt bằng thư viện VnCoreNLP.

Cuối cùng, so sánh đối chiếu kết quả của 2 file trên. Dùng excel để lọc những vị trí khác nhau về nhãn của 2 file và thực hiện tra cứu lại trên từ điển VLSP để có được kết quả cuối cùng, được lưu trong file gold.txt.

Chi tiết về dữ liệu trong file gold.txt:

- Các tiếng trong từ được nối với nhau bằng dấu '_'.
- Từ được phân tách với nhãn của nó bằng dấu tab.
- Mỗi dòng là một từ và nhãn của nó.
- Mỗi câu được ngăn cách bằng một dòng trống.
- Số lượng câu: 60 câu.
- Số dòng: 1714 dòng (bao gồm 60 dòng trống).
- Số nhãn: 1654 nhãn.

	N	V	R	CH	C	E	X	A	P	D	M	I	Cc	Np	Ni	Z	Nu	Nc	O	Total
0	453	396	141	132	105	92	73	71	49	49	26	24	23	10	5	2	1	1	1	1654

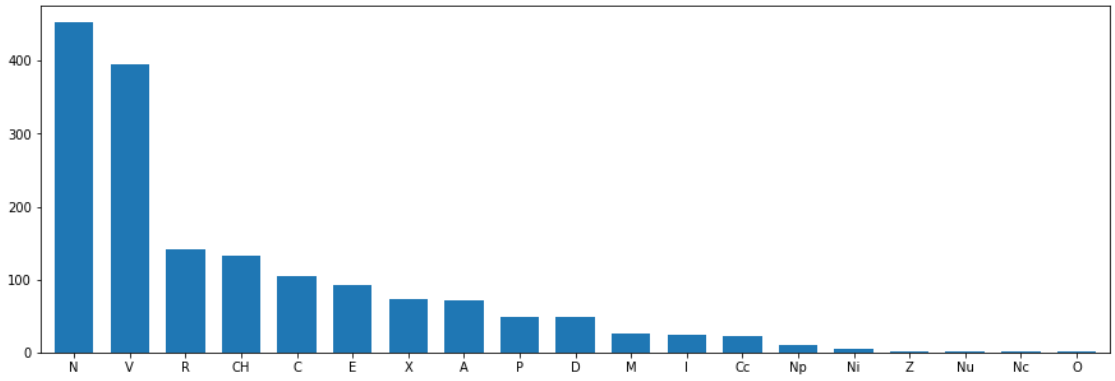


Figure 2 - Thống kê các nhãn trong file gold.txt

Kế tiếp nhóm sẽ tiến hành chia dữ liệu đã được tách từ và gán nhãn trong file gold.txt thành 2 tập dữ liệu: tập Train có 50 câu, tập Test có 10 câu. Những từ trong tập test không có trong bộ từ vựng thì sẽ được gán nhãn là '–unk–'. Bộ từ vựng được sử dụng ở đây được lưu với tên vocabs.txt gồm 54818 từ. Khi quá trình tiền xử lý kết thúc một câu, giá trị tại đó 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 quá trình huấn luyện.
- train_words.txt: chỉ chứa các từ để thực hiện việc kiểm thử trên tập train (kiểm tra overfitting).
- Số lượng câu của tập Train: 50 câu.
- Số lượng từ của tập Train: 1490 từ.
- Số lượng nhãn của tập Train: 1441 nhãn.

	N	V	R	CH	C	E	X	A	P	D	M	I	Cc	Np	Z	Nu	Ni	Nc	O	Total
0	391	343	129	104	102	79	68	51	47	40	26	24	21	10	2	1	1	1	1	1441

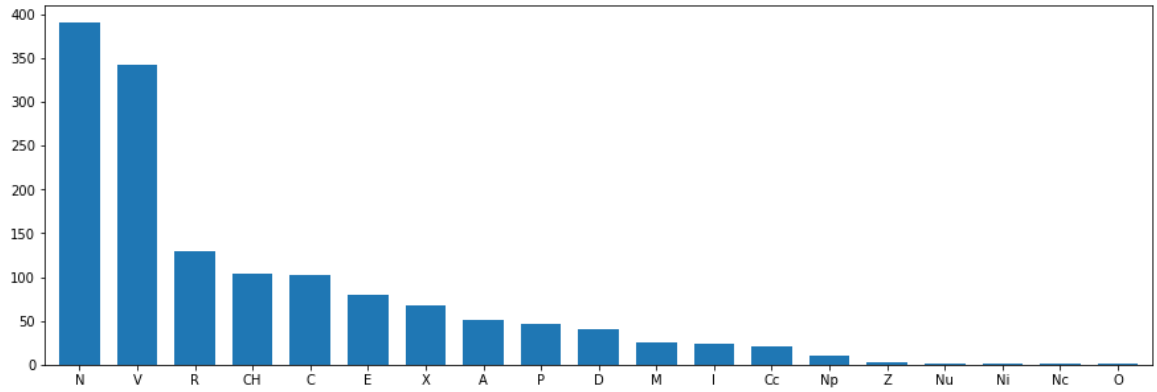


Figure 3 - Thống kê các nhãn trong tập train.

1	Để	C
2	đủ	A
3	điều_kiện	N
4	ra	V
5	trường	N
6	thì	C
7	em	N
8	cần	V
9	đạt	V
10	bằng	N
11	Toeic	Np
12	ở	E
13	mức	N
14	bao_nhiêu	P
15	điểm	N
16	ạ	I
17	?	CH
18		
19	Nếu	C

Figure 4 - Một câu điển hình trong file gold.txt

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 việc dự đoán.
- Số lượng câu của tập test: 10
- Số lượng từ của tập test: 221
- Số lượng nhãn của tập test: 212

	N	V	CH	A	E	R	D	X	Ni	C	Cc	P	Total
0	61	53	28	20	13	12	9	5	4	3	2	2	212

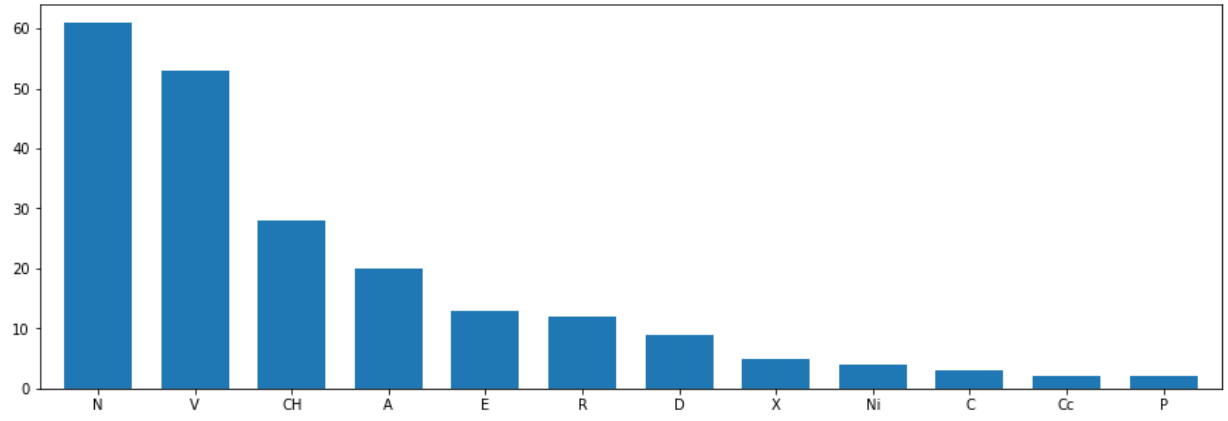


Figure 5 - Thống kê nhãn tập test.

VI.Gán nhãn từ loại

1. Chuẩn bị gán nhãn

1.1. 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:
(('N', 'C'), 30)
(('C', 'N'), 39)
(('V', 'V'), 72)
(('N', 'Np'), 7)
(('Np', 'E'), 2)
```

Figure 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:
(('C', '--unk--'), 5)
(('A', 'đủ'), 2)
(('N', 'điều_kiện'), 2)
(('V', 'ra'), 7)
(('N', 'trường'), 26)
(('C', 'thì'), 20)
(('N', 'em'), 43)
```

Figure 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: 20
['--s--', 'A', 'C', 'CH', 'Cc', 'D', 'E', 'I', 'M', 'N', 'Nc', 'Ni', 'Np', 'Nu', 'O', 'P', 'R', 'V', 'X', 'Z']
```

Figure 8 -C ác nhãn trong từ điển Tag Counts

1.2. Testing

Sau khi tạo ra các từ điển, nhóm bắt đầu kiểm tra độ chính xác của mô hình gắn 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: 0.886058981233244
- Độ chính xác trên tập Test: 0.6153846153846154

2. Mô hình Markov ẩn (Hidden Markov Model - HMM)

2.1. Markov Chain (xích Markov)

HMM hoạt động dựa trên Markov Chain – một mô hình cho biết về xác suất của một chuỗi các trạng thái được biểu hiện bằng các biến ngẫu nhiên, mỗi trạng thái có thể nhận các giá trị từ một tập hợp nào đó, có thể là tập các từ, các nhãn thể hiện một điều bất kỳ. Một Markov Chain đưa ra một giả định rằng chúng ta có thể dự đoán trạng thái trong tương lai bằng trạng thái hiện tại, tương lai không bị ảnh hưởng bởi các trạng thái trước hiện tại mà thông qua trạng thái hiện tại. Ví dụ để dự

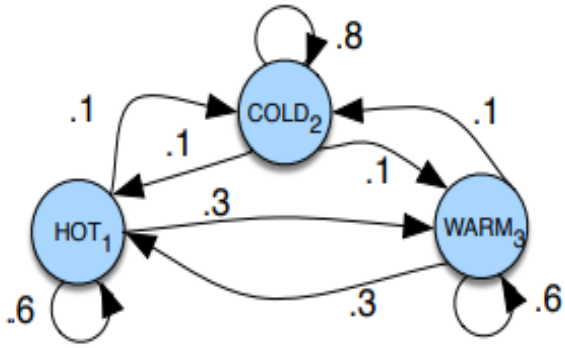


Figure 9 - Markov chain thời tiết

đoán thời tiết ngày mai, có thể dựa vào thời tiết hôm nay những không thể dựa vào thời tiết hôm qua.

Tổng quát, xét một chuỗi các biến ngẫu nhiên $q_1, q_2, \dots, q_i$. Một mô hình Markov thể hiện **giả định Markov** về xác suất của chuỗi trên rằng khi dự đoán tương lai chỉ dựa vào hiện tại mà không phải quá khứ: $P(q_i = a|q_1 \dots q_{i-1}) = P(q_i = a|q_{i-1})$

Một Markov Chain gồm các thành phần sau:

- $Q = q_1 q_2 \dots q_N$ một tập gồm N trạng thái, gồm chuỗi các biến ngẫu nhiên
- $A = a_{11} a_{12} \dots a_{N1} \dots a_{NN}$ ma trận chuyển trạng thái A, mỗi phần tử a_{ij} đại diện cho xác suất chuyển từ trạng thái i sang trạng thái j biết $\sum_{j=1}^{N} a_{ij} = 1, \forall i$
- $\pi = \pi_1, \pi_2, \dots, \pi_N$ phân phối xác xuất ban đầu trên các trạng thái, π_i là xác suất để Markov Chain bắt đầu từ trạng thái i. Nếu trạng thái nào có giá trị xác suất bằng 0 nghĩa là không được khởi tạo phân phối xác suất ban đầu. Biết $\sum_{i=1}^{N} \pi_i = 1$

2.2. Hidden Markov Model

Có thể sử dụng Markov Chain để tính xác suất cho một chuỗi các trạng thái có thể quan sát được. Tuy nhiên trong nhiều trường hợp, các trạng thái không được thể hiện mà bị ẩn đi khiến ta không thể quan sát chúng một cách trực tiếp. Mô hình Markov ẩn cho phép chúng ta làm việc với các trạng thái có thể quan sát được và cả các trạng thái ẩn được nêu trên.

Một số ứng dụng của HMM:

- Nhận dạng giọng nói hay ký tự trong xử lý ngôn ngữ tự nhiên
- Các bài toán mô phỏng cấu trúc sinh học liên quan đến gen và protein

Mô hình Markov ẩn (HMM) gồm:

- $Q = q_1 q_2 \dots q_N$ một tập gồm N trạng thái, gồm chuỗi các biến ngẫu nhiên
- $A = a_{11} a_{12} \dots a_{n1} \dots a_{NN}$ ma trận chuyển trạng thái A, mỗi phần tử a_{ij} đại diện cho xác suất chuyển từ trạng thái i sang trạng thái j biết $\sum_{j=1}^{N} a_{ij} = 1, \forall i$

- $O = o_1 o_2 \dots o_T$ là tập các giá trị quan sát
- $B = b_i(o_t)$ ma trận thể hiện hay ma trận phát xạ, mỗi phần tử đại diện cho xác suất của một quan sát o_t được tạo ra từ trạng thái i
- $\pi = \pi_1, \pi_2, \dots, \pi_N$ phân phối xác xuất ban đầu trên các trạng thái, π_i là xác suất để Markov Chain bắt đầu từ trạng thái i. Nếu trạng thái nào có giá trị xác suất bằng 0 nghĩa là không được khởi tạo phân phối xác suất ban đầu. Biết $\sum_{i=1}^{N} \pi_i = 1$

2.3. Ma trận chuyển trạng thái A

	--s--	A	C	CH	Cc	D	E	I	M	N
--s--	0.008197	0.024590	0.122951	0.008197	0.008197	0.040984	0.057377	0.008197	0.008197	0.352459
A	0.008197	0.040984	0.057377	0.188525	0.040984	0.008197	0.155738	0.024590	0.008197	0.139344
C	0.004464	0.013393	0.031250	0.013393	0.004464	0.066964	0.022321	0.004464	0.013393	0.352679
CH	0.451754	0.013158	0.083333	0.013158	0.013158	0.013158	0.013158	0.004386	0.048246	0.118421
Cc	0.016129	0.016129	0.048387	0.016129	0.016129	0.048387	0.048387	0.016129	0.048387	0.209677

Figure 10 - Ma trận chuyển trạng thái A

Ma trận chuyển trạng thái A có dạng $n * n$ với n là tổng số lượng nhãn, mỗi ô là xác suất chuyển tiếp từ 1 nhãn tới 1 nhãn khác có trong dữ liệu.

Công thức:

$$A_{ij} = \frac{f_{ij} + \alpha}{f_i + \alpha n}$$

Trong đó:

- f_{ij} tần số xuất hiện của cặp nhãn thứ ij
- f_i tần số xuất hiện của nhãn thứ i
- n tổng số lượng nhãn
- α hệ số smoothing điều chỉnh tần số

2.4. *Ma trận thể hiện B*

	em	và	cho	là	nhà_vệ_sinh	ngủ
R	0.000018	0.000018	0.000018	0.000018	0.000018	0.000018
V	0.000018	0.000018	0.000306	0.000090	0.000018	0.000018
N	0.001565	0.000018	0.000018	0.000018	0.000018	0.000018
A	0.000018	0.000018	0.000018	0.000018	0.000018	0.000018
D	0.000018	0.000018	0.000018	0.000018	0.000018	0.000018
E	0.000018	0.000018	0.000418	0.000018	0.000018	0.000018
C	0.000018	0.000018	0.000018	0.000345	0.000018	0.000018
Cc	0.000018	0.000674	0.000018	0.000018	0.000018	0.000018
CH	0.000018	0.000018	0.000018	0.000018	0.000018	0.000018

Figure 11 - Ma trận thể hiện B

Ma trận thể hiện B có dạng $n * len(words)$ với n là tổng số lượng nhãn, $words$ là tập từ vựng, mỗi ô là xác suất mang 1 nhãn của từ đó.

Công thức:

$$B_{ij} = \frac{f_{ij} + \alpha}{f_i + \alpha * len(words)}$$

Trong đó:

- f_{ij} tần số xuất hiện của cặp nhãn thứ i và từ thứ j
- f_i tần số xuất hiện của nhãn thứ i
- $len(words)$ tổng số lượng từ
- α hệ số smoothing điều chỉnh tần số

3. Thuật toán Viterbi

3.1 . Khởi tạo

Khởi tạo 2 ma trận có cùng kích thước:

- Ma trận *best_prob* chứa xác suất của 1 nhãn đến 1 từ
- Ma trận *best_path* chứa đường đi tốt nhất giữa các nhãn của mỗi từ trong câu

Các giá trị trong 2 ma trận đều được khởi tại bằng 0, tuy nhiên ở cột đầu tiên của ma trận *best_prob*, các giá trị sẽ được tính theo công thức:

$$probs_{i0} = A_{start_{idx},i} * B_{i,start}$$

Trong đó:

- $A_{start_{idx},i}$ Xác suất chuyển từ nhãn bắt đầu tới nhãn thứ i
- $B_{i,start}$ Xác suất mang nhãn thứ i của từ đầu tiên

3.2 . Forward

Thực hiện theo thứ tự sau:

- B1: Lần lượt duyệt qua từng từ, từng nhãn của từ đang xét và nhãn của từ trước đó
- B2: Tính xác suất bằng tích của xác suất tốt nhất trước đó, xác suất chuyển trạng thái và xác suất thể hiện. Nếu xác suất này là xác suất cao nhất ở thời điểm hiện tại thì giữ lại nó và nhãn của từ trước đó, điền thông tin vào bảng *best_prob*
- B3: Tính toán đường đi đến vị trí đó rồi điền vào bảng *best_path*

Bảng $best\_prob$ có các giá trị được tính bằng

$$probs = best\_prob_{k,i-1} * A_{ki} * B_{i,vocabs[word_i]}$$

Trong đó:

- $word_i$ Từ thứ i trong câu đang xét
- $vocabs[word_i]$ Vị trí của từ thứ i trong từ điển
- k Chỉ số của nhãn trước từ đang xét

Bảng $best\_path$ có các giá trị mỗi ô từ cột thứ 2 chứa vị trí của nhãn $i - 1$ có $best\_prob$ cao nhất

Mô tả thuật toán:

Duyệt qua các từ trong câu input:

Duyệt qua danh sách nhãn có thể có của từ:

Duyệt qua từng loại nhãn của từ trước đó:

- Tìm vị trí của từ ở cột thứ *i* trong tập ngữ liệu
- Tính xác xuất là tích của xác xuất mang nhãn gì đó của từ trước đó, xác suất chuyển từ nhãn của từ trước đó sang nhãn đang xét và xác suất mang nhãn đó của từ đang xét.
- So sánh với các xác suất cùng bậc đang xét và giữ lại xác suất cao nhất
- Lưu giá trị vào vị trí *ij* tương ứng trong bảng *best_prob*
- Lưu chỉ số *k* của nhãn của từ trước đó ứng với xác xuất cao nhất được điền vào ở bước trên.

Kết quả:

	--unk--	đủ	điều_kiện	ra	trường	thì	em	cần	đạt	bằng
0: --s--	-15.715868	-25.016843	-37.186964	-48.312317	-60.745085	-72.337751	-82.823066	-89.394443	-101.521035	-112.638683
1: A	-13.520429	-21.964109	-33.118474	-46.099755	-58.532523	-68.625964	-79.780330	-91.005667	-97.809249	-109.474440
2: C	-10.613564	-21.568662	-33.967627	-45.704309	-58.103274	-68.230518	-79.020046	-90.756727	-96.315190	-109.812768
3: CH	-13.011532	-21.747783	-34.407067	-45.702562	-58.316198	-68.409639	-79.459485	-90.935848	-97.592924	-110.252207
4: Cc	-15.716561	-22.968537	-35.449875	-47.104184	-59.536952	-69.630393	-81.214321	-91.004574	-98.813678	-111.295016
5: D	-13.009203	-23.279385	-34.404738	-46.837506	-58.540385	-69.941241	-75.486864	-89.906654	-99.124525	-109.932018
6: E	-12.674151	-21.708408	-33.966791	-45.844055	-58.102438	-68.370264	-79.019209	-90.896473	-97.553548	-109.811932
7: I	-15.716670	-23.278802	-34.406792	-47.246518	-58.542439	-69.940658	-81.068648	-91.004683	-99.123942	-110.251932
8: M	-14.107305	-22.457894	-35.379593	-46.593541	-59.026308	-69.119750	-80.878032	-91.004756	-98.303034	-111.224734
9: N	-8.076939	-20.998638	-32.212586	-44.645354	-54.738795	-67.660494	-77.265004	-83.922080	-96.843779	-108.057726
10: Nc	-15.715831	-25.016806	-36.526216	-48.834306	-60.661863	-72.337714	-82.823030	-91.003845	-101.520999	-112.371357

Figure 12 - Viterbi – Forward - bảng best_prob

	--unk--	đủ	điều_kiện	ra	trường	thì	em	cần	đạt	bằng
0: --s--	0	18	5	5	5	9	17	5	9	2
1: A	0	9	17	9	9	9	17	5	9	17
2: C	0	9	17	9	17	9	17	9	9	17
3: CH	0	9	17	1	9	9	17	9	9	17
4: Cc	0	9	8	9	9	9	17	5	9	8
5: D	0	9	17	17	17	9	17	5	9	2
6: E	0	9	17	9	17	9	17	9	9	17
7: I	0	9	16	16	16	9	16	5	9	16
8: M	0	9	9	9	9	9	17	5	9	9
9: N	0	9	17	17	17	9	17	5	9	17

Figure 13 - Viterbi - Forward - Bảng best_path

3.3 . Backward

Ý tưởng: Quay lui từ cột cuối cùng của ma trận xác suất (*best_prob*) và lần lượt tìm hàng có giá trị lớn nhất rồi đối chiếu qua ma trận đường đi (*best_path*) để tìm nhãn cho từ tương ứng.

Mã giả:

Duyệt qua các dòng ở cột cuối cùng của bảng *best_prob:*

- Tìm hàng có giá trị cao nhất (max)
- Trả về vị trí của hàng đó

Duyệt qua lần lượt từng cột của bảng *best_path* từ cột cuối lên cột đầu tiên:

- Từ vị trí của max đã có được ở bước trên, so khớp với cột cuối của bảng *best_path* để trả về vị trí của nhãn từ trước đó
- Lần lượt cập nhật nhãn của các từ

Kết quả:

	--unk--	đủ	điều_kiện	ra	trường	thì	em	cần	đạt	bằng
0: --s--	-15.715868	-25.016843	-37.186964	-48.312317	-60.745085	-72.337751	-82.823066	-89.394443	-101.521035	-112.638683
1: A	-13.520429	-21.964109	-33.118474	-46.099755	-58.532523	-68.625964	-79.780330	-91.005667	-97.809249	-109.474440
2: C	-10.613564	-21.568662	-33.967627	-45.704309	-58.103274	-68.230518	-79.020046	-90.756727	-96.315190	-109.812768
3: CH	-13.011532	-21.747783	-34.407067	-45.702562	-58.316198	-68.409639	-79.459485	-90.935848	-97.592924	-110.252207
4: Cc	-15.716561	-22.968537	-35.449875	-47.104184	-59.536952	-69.630393	-81.214321	-91.004574	-98.813678	-111.295016
5: D	-13.009203	-23.279385	-34.404738	-46.837506	-58.540385	-69.941241	-75.486864	-89.906654	-99.124525	-109.932018
6: E	-12.674151	-21.708408	-33.966791	-45.844055	-58.102438	-68.370264	-79.019209	-90.896473	-97.553548	-109.811932
7: I	-15.716670	-23.278802	-34.406792	-47.246518	-58.542439	-69.940658	-81.068648	-91.004683	-99.123942	-110.251932
8: M	-14.107305	-22.457894	-35.379593	-46.593541	-59.026308	-69.119750	-80.878032	-91.004756	-98.303034	-111.224734
9: N	-8.076939	-20.998638	-32.212586	-44.645354	-54.738795	-67.660494	-77.265004	-83.922080	-96.843779	-108.057726
10: Nc	-15.715831	-25.016806	-36.526216	-48.834306	-60.661863	-72.337714	-82.823030	-91.003845	-101.520999	-112.371357

Figure 14 - Viterbi - Backward - bảng best_prob

Như trong hình trên thì giá trị cần tìm có nhãn "N", đối chiếu với bảng *best_path* tại vị trí tương ứng có giá trị là 17, dựa vào đó đánh dấu giá trị ở hàng thứ 17 của cột tiếp theo và cứ tiếp tục cho tới cột đầu tiên.

	--unk--	đủ	điều_kiện	ra	trường	thì	em	cần	đạt	bằng
0: --s--	0	18	1	9	17	9	2	9	17	17
1: A	0	9	1	9	17	9	2	9	17	17
2: C	0	9	17	9	17	9	2	9	17	17
3: CH	0	9	1	9	17	9	2	9	17	17
4: Cc	0	9	1	9	17	9	2	9	17	17
5: D	0	9	17	9	17	9	2	9	17	17
6: E	0	9	1	9	17	9	2	9	17	17
7: I	0	9	16	9	17	9	2	9	16	17
8: M	0	9	9	9	17	9	2	9	17	17
9: N	0	9	17	9	17	9	2	9	17	17
10: Nc	0	18	1	9	17	9	2	9	17	17
11: Ni	0	9	1	9	17	9	2	9	17	17
12: Np	0	9	9	9	17	9	2	9	17	17
13: Nu	0	18	1	3	17	9	2	9	17	17
14: O	0	18	1	9	17	9	2	9	17	17
15: P	0	9	17	9	17	9	2	9	17	17
16: R	0	9	1	9	17	9	2	9	17	17
17: V	0	9	16	9	17	9	2	9	17	17
18: X	0	9	17	9	17	9	2	9	17	17
19: Z	0	18	1	9	17	9	2	9	17	17

Figure 15 - Viterbi - Backward - bảng best_path

Sau khi hoàn thành xong bảng trên, ta có kết quả cuối cùng:

	--unk--	đủ	điều_kiện	ra	trường	thì	em	cần	đạt	bằng
0	N	V	N	V	N	C	N	V	V	N

Figure 16 - Kết quả cuối cùng

VII.Đánh giá

1. Kết quả

Một số kết quả thực tế sau khi thực hiện gán nhãn từ loại

	HMM + Viterbi	**VnCoreNLP**	**Underthesea**
1	ĐTĐH/N linh_động/V hơn/N tạo/V điều_kiện/N giúp/V các/D sinh_viên/N lỡ/V chậm/N tiến_độ/V sớm/N theo/V kịp/V chương_trình/N đào_tạo/V ./CH	ĐTĐH/Np linh_động/V hơn/A tạo/V điều_kiện/N giúp/V các/L sinh_viên/N lỡ/V chậm/A tiến_độ/N sớm/A theo/V kịp/A chương_trình/N đào_tạo/V ./CH	ĐTĐH/Np linh_động/M hơn/A tạo/V điều_kiện/N giúp/V các/L sinh_viên/N lỡ/M chậm/A tiến_độ/N sớm/A theo/V kịp/A chương_trình/N đào_tạo/V ./CH
2	Đề_nghị/V nhà_trường/N thiết_kế/V thời_gian/N học/V hợp_lí/N cho/V sinh_viên/N CTTT/X ,/CH cụ_thể/N tăng/V thời_gian/N học/V hè/N lên/V để/C đảm_bảo/V chất_lượng/N dạy/V và/N học/V ./CH	Đề_nghị/V nhà_trường/N thiết_kế/V thời_gian/N học/V hợp_lí/A cho/E sinh_viên/N CTTT/Np ,/CH cụ_thể/A tăng/V thời_gian/N học/V hè/N lên/V để/E đảm_bảo/V chất_lượng/N dạy/V và/Cc học/V ./CH	Đề_nghị/V nhà_trường/N thiết_kế/V thời_gian/N học/V hợp_lí/M cho/E sinh_viên/N CTTT/Np ,/CH cụ_thể/A tăng/V thời_gian/N học/V hè/N lên/V để/E đảm_bảo/V chất_lượng/N dạy/V và/C học/V ./CH

2. Đánh giá

Độ đo: Nhóm đã sử dụng 4 độ đo để tiến hành đánh giá gồm Accuracy, Precision, Recall và F1-score:

	Predict label	
	Negative	Positive

True label	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

- Accuracy là độ chính xác thể hiện tỷ lệ số nhãn được dự đoán đúng trên toàn bộ tập dữ liệu.

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

- Precision là độ chính xác thể hiện tỷ lệ số nhãn được tự đoán là TP trên tổng số nhãn Positive dự đoán.

$$Precision = \frac{TP}{TP + FP}$$

- Recall là độ phủ thể hiện tỷ lệ số nhãn được dự đoán là TP trên tổng số nhãn Positive thực.

$$Recall = \frac{TP}{TP + FN}$$

- F1-score là trung bình điều hòa giữa Precision và Recall

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Kết quả:

- Kết quả của HMM:

A	0.37	1.00	0.54	19
C	0.80	0.87	0.84	94
CH	0.89	0.98	0.93	95
Cc	0.76	1.00	0.86	16
D	0.90	1.00	0.95	36
E	0.68	0.93	0.79	58
I	1.00	0.92	0.96	26
M	0.42	1.00	0.59	11
N	0.95	0.79	0.87	470
Nc	0.00	0.00	0.00	0
Ni	0.00	0.00	0.00	0
Np	0.00	0.00	0.00	0
Nu	0.00	0.00	0.00	0
O	0.00	0.00	0.00	0
P	0.62	1.00	0.76	29
R	0.83	0.87	0.85	123
V	0.95	0.74	0.83	441
X	0.29	0.87	0.44	23
Z	0.00	0.00	0.00	0
accuracy			0.83	1441
macro avg	0.50	0.63	0.54	1441
weighted avg	0.89	0.83	0.84	1441

Figure 17 - Kết quả của mô hình Hidden Markov kết hợp thuật toán Viterbi trên tập train

A	0.10	1.00	0.18	2
C	0.67	0.25	0.36	8
CH	0.75	1.00	0.86	21
Cc	0.50	1.00	0.67	1
D	1.00	1.00	1.00	9
E	0.62	1.00	0.76	8
I	0.00	0.00	0.00	4
M	0.00	0.00	0.00	2
N	0.75	0.65	0.70	71
Ni	0.00	0.00	0.00	0
P	0.50	1.00	0.67	1
R	0.92	0.73	0.81	15
V	0.81	0.63	0.71	68
X	0.20	0.50	0.29	2
accuracy			0.68	212
macro avg	0.49	0.63	0.50	212
weighted avg	0.75	0.68	0.70	212

Figure 18 - Kết quả của mô hình Hidden Markov kết hợp thuật toán Viterbi trên tập test

– Kết quả của Underthesea:

	precision	recall	f1-score	support
A	0.75	0.94	0.83	16
C	1.00	0.50	0.67	6
CH	1.00	1.00	1.00	28
Cc	0.00	0.00	0.00	0
D	0.00	0.00	0.00	0
E	1.00	1.00	1.00	13
L	0.00	0.00	0.00	9
M	0.00	0.00	0.00	27
N	0.75	1.00	0.86	46
Nc	0.00	0.00	0.00	1
Ni	0.00	0.00	0.00	0
Np	0.00	0.00	0.00	6
P	1.00	1.00	1.00	2
R	1.00	0.92	0.96	13
T	0.00	0.00	0.00	1
V	0.81	0.98	0.89	44
X	0.00	0.00	0.00	0
accuracy			0.76	212
macro avg	0.43	0.43	0.42	212
weighted avg	0.68	0.76	0.71	212

Figure 19 - Kết quả khi sử dụng thư viện Underthesea trên tập test

– Kết quả của VNCoreNLP:

	precision	recall	f1-score	support
A	0.90	1.00	0.95	18
C	1.00	1.00	1.00	3
CH	1.00	1.00	1.00	28
Cc	1.00	1.00	1.00	2
D	0.00	0.00	0.00	0
E	1.00	1.00	1.00	13
L	0.00	0.00	0.00	9
M	0.00	0.00	0.00	1
N	0.87	1.00	0.93	53
Ni	1.00	1.00	1.00	4
Np	0.00	0.00	0.00	9
P	1.00	1.00	1.00	2
R	1.00	1.00	1.00	12
T	0.00	0.00	0.00	1
V	1.00	0.93	0.96	57
X	0.00	0.00	0.00	0
accuracy			0.89	212
macro avg	0.61	0.62	0.62	212
weighted avg	0.86	0.89	0.87	212

Figure 20 - Kết quả khi sử dụng thư viện VnCoreNLP trên tập test:

- So sánh kết quả giữa HMM, Underthesea, VNCoreNLP:

	Accuracy	Precision	Recall	F1-score
HMM+Viterbi	0.68	0.75	0.68	0.70
Underthesea	0.76	0.68	0.76	0.71
VnCoreNLP	0.89	0.86	0.89	0.87

3. Nhận xét

3.1. Tách từ - Longest Matching

- Thuật toán dễ cài đặt và sử dụng, chỉ cần dựa vào từ điển và cho ra độ chính xác tương đối cao.
- Tuy nhiên, do dựa vào từ điển nên độ chính xác haofn toàn phụ thuộc vào độ chính xác của từ điển.
- Nhiều trường hợp cho kết quả sai vì xét cụm từ theo thứ tự từ trái qua phải nên không lường được trường hợp các tiếng phía sau cũng có thể tổ hợp lại thành từ có nghĩa phù hợp với ngữ cảnh.

3.2. Gán nhãn từ loại

Thư viện VnCoreNLP và Underthesea cho kết quả gán nhãn từ vựng tốt hơn so với Hidden Markov Model. Nguyên nhân là do:

- Phần lớn là do tập dữ liệu đưa vào train chưa đủ lớn (50 câu - 1493 từ), chưa bao quát đủ các trương hợp xảy ra, dẫn đến khó khăn trogn việc xác định nhãn.
- Mẫu test có thể quá khó với thuật toán: Phân bố không đồng đều số lượng từ ở mỗi nhãn của tập train và test vì một số nhãn có quá ít ngữ liệu, không đủ để chia ra dẫn đến việc tập train không có nhưng tập test lại có, tập test có nhiều nhưng tập train ít gặp
- Số lượng từ vựng đưa vào chưa nhiều.

Tài liệu tham khảo

Ngữ liệu	https://forum.uit.edu.vn/node/23
Từ điển VLSP	https://vlsp.hpda.vn/demo/?page=vcl
Longest Matching	https://123docz.net/document/7824054-tach-tu-tie-ng-vie-t-su-du-ng-longest-matching-va-conditional-random-fields.htm
Hidden Markov & Viterbi	https://web.stanford.edu/~jurafsky/slp3/A.pdf
2 bộ từ điền bi_grams và tri_grams	https://github.com/undertheseanlp/underthesea/blob/main/underthesea/corpus/data/Viet74K.txt