ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
UIT-HCM

GÁN NHÃN TỪ LOẠI

Thành viên:

Nguyễn Ngọc Thái Nguyên	19521917
Nguyễn Đức Phương Nguyên	19521913
Lê Quang Minh	19521836

Nội dung

I. Giới thiệu đề tài
II. Thu thập ngữ liệu
III. Phương pháp tách từ
IV. Mô hình Hidden Markov
V. Thực nghiệm và đánh giá

I. Giới thiệu đề tài

- Bài toán gán nhãn từ loại có thể xem là bài toán cơ bản nhất khi ta nhập môn xử lý ngôn ngữ tự nhiên
- Đây là bước cơ bản khi phân tích cú pháp hay các vấn đề xử lý phức tạp khác
- Mục tiêu khi giải quyết bài toán này

Con người chỉ đơn giản là những cỗ máy vô cùng phức tạp

→

con_người/N chỉ/R đơn_giản/A là/V những/D cỗ_máy/N vô_cùng/R phức_tạp/A

I. Giới thiệu đề tài

1. Bài toán tách từ

- 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
- Mục tiêu là xác định ranh giới của các từ trong câu

Ví dụ:

Input: Con người chỉ đơn giản là những cỗ máy vô cùng phức tạp

Output: Con_người chỉ đơn_giản là những cỗ_máy vô_cùng phức_tạp

➡ Bước quan trọng chúng ta cần thực hiện trước khi đưa dữ liệu vào các bước tiếp theo

I. Giới thiệu đề tài

2. Gán nhãn từ loại tiếng việt

- Phân biệt các bộ phận 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

Con_người chỉ đơn_giản là những cỗ_máy vô_cùng phức_tạp

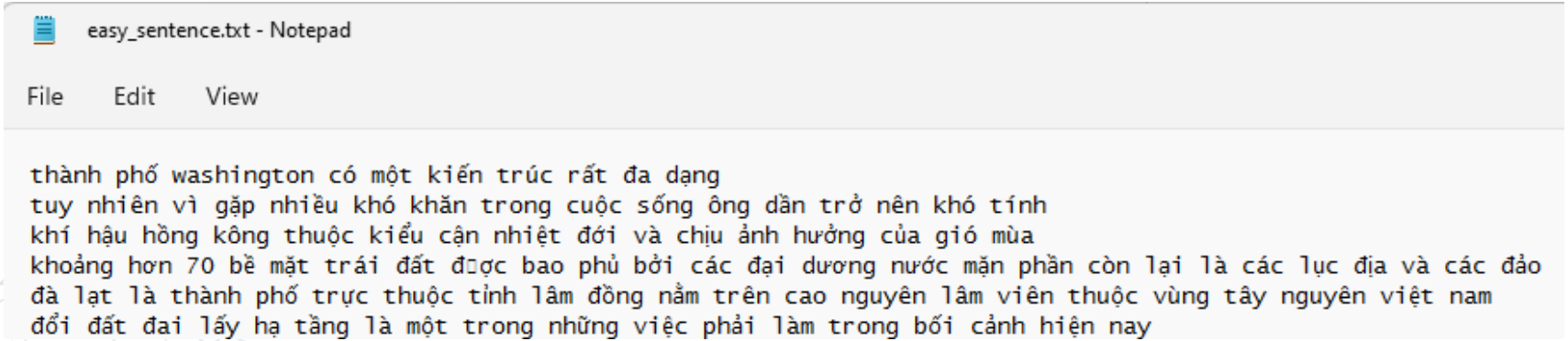
→

con_người/N chỉ/R đơn_giản/A là/V những/D cỗ_máy/N vô_cùng/R phức_tạp/A

II. Thu thập ngữ liệu

1. Thông tin về bộ ngữ liệu

- Nguồn thu thập là các trang báo điện tử như: https://tuoitre.vn/ , https://tinhte.vn/, https://kenh14.vn/, https://www.wikipedia.org/ và các câu tự tạo
- Bao gồm 62 câu
- Ngoài ra nhóm đã tự xây dựng một bộ Vocab để phục vụ cho thuật toán tách từ bằng cách crawl thêm nội dung 1000 bài báo ở https://kenh14.vn/



II. Thu thập ngữ liệu

- Câu dài nhất: 32 tiếng

Biến đổi khí hậu đe dọa đến con người khi nó gây bất an lương thực, khan hiếm nước, lũ lụt, nắng nóng cực đoan, thiệt hại kinh tế và di cư.

- Câu ngắn nhất: 5 tiếng

Đó là một thói quen

III. Phương pháp tách từ

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**.
- Tuy nhiên, độ chính xác của phương pháp này phụ thuộc rất lớn vào kích thước của bộ **vocab.**
- Do không cần training nên thời gian xử lý tương đối nhanh, đơn giản và dễ hiểu. Do sự phụ thuộc nên phương pháp này khó có thể xử lý được các từ mới không xuất hiện trong **vocab**

III. Phương pháp tách từ

1. Thuật toán Longest Matching

- Xây dựng vocab
- 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:

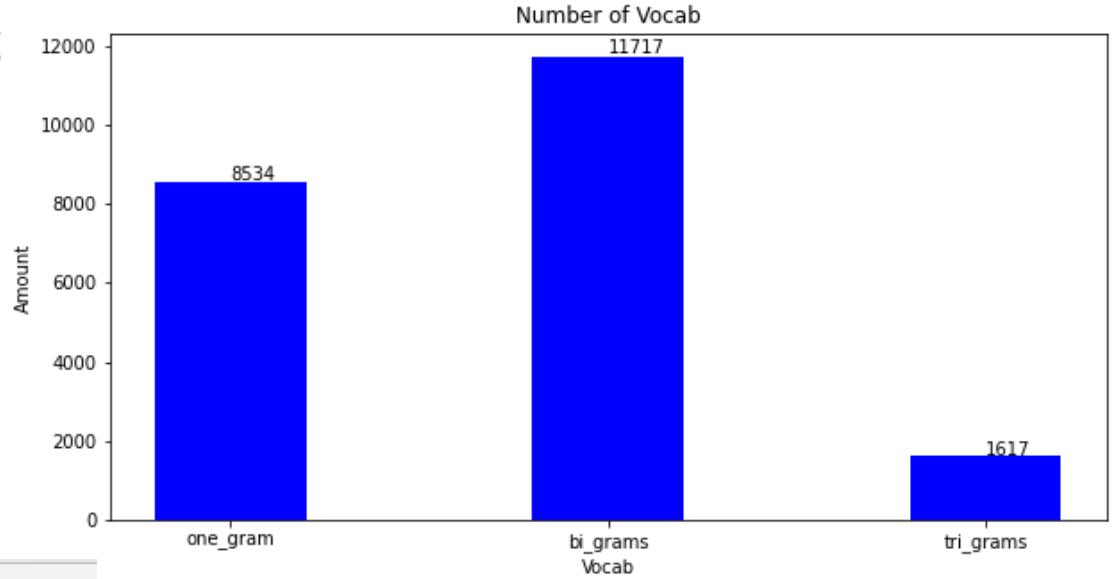
- Vocab one gram: Bộ vocab chỉ chứa những từ đơn
- Vocab bi grams: Bộ vocab chứa từ 2 âm
- Vocab tri grams: Bộ vocab chứa từ 3 âm

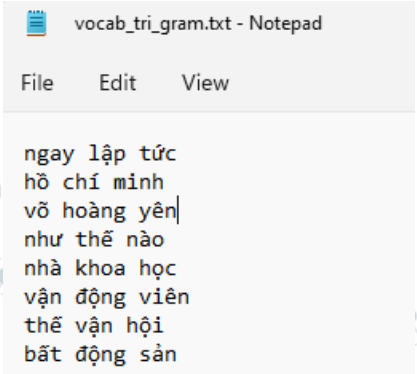
III. Phương pháp tách từ

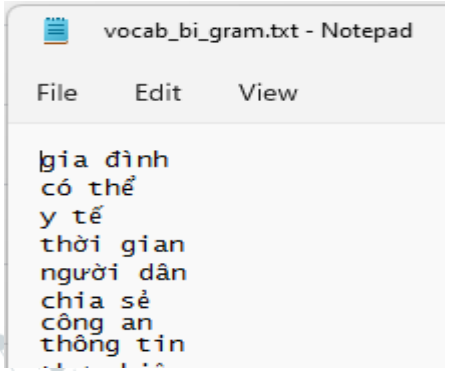
1. Thuật toán Longest Matching

- Chi tiết về Vocab gồm

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



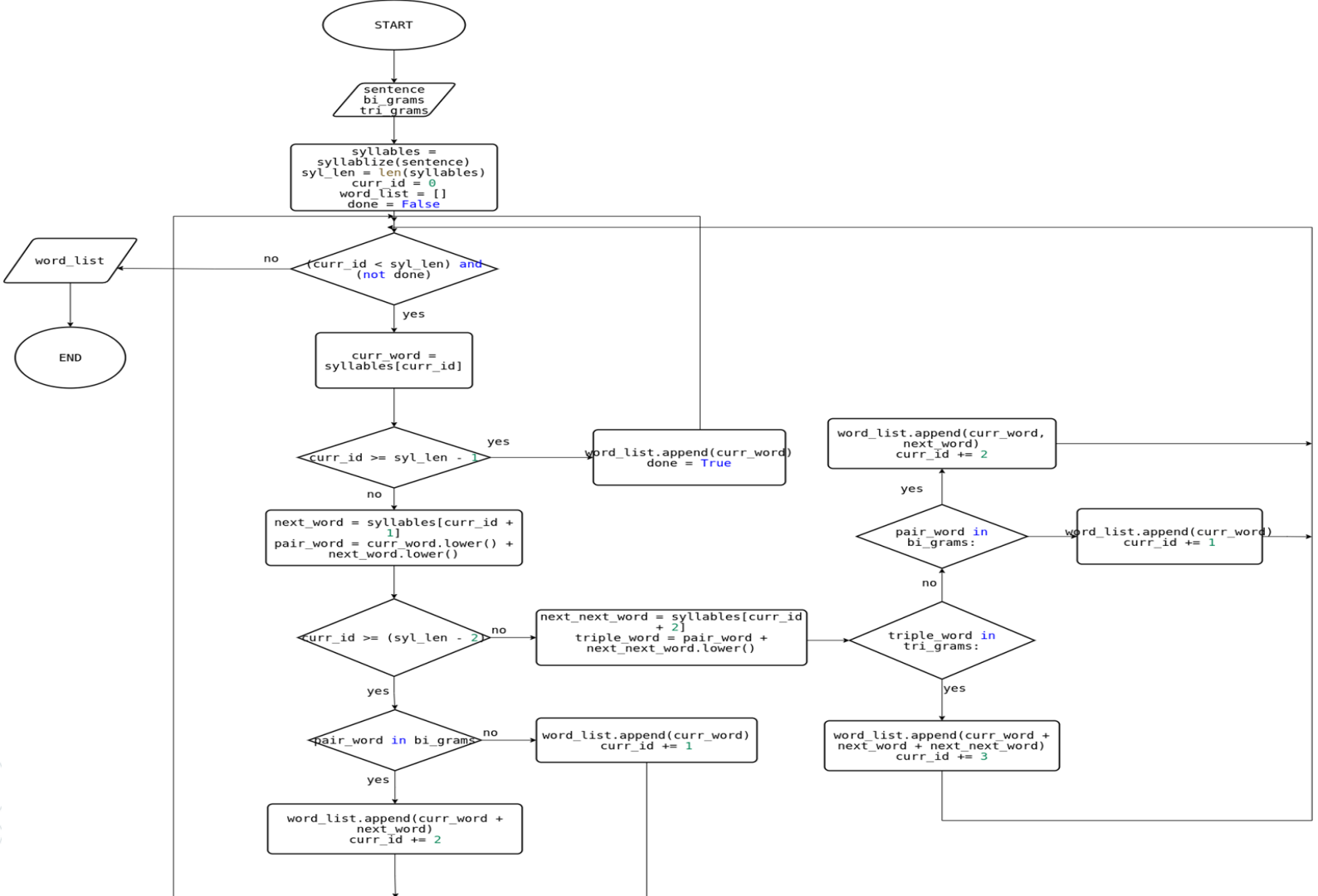




III. Phương pháp tách từ

1. Thuật toán Longest Matching

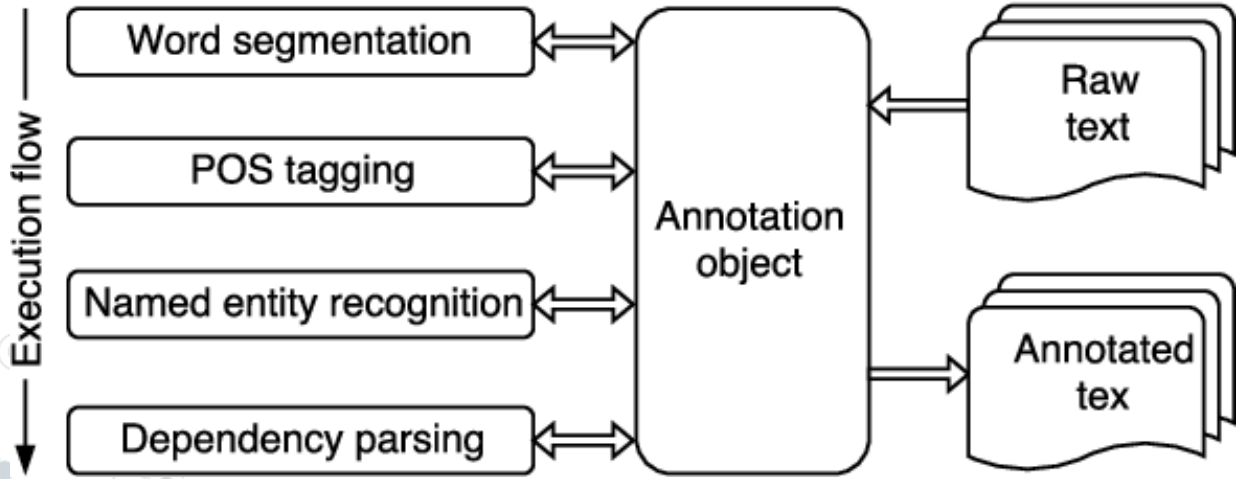
```
current_word _id = 0 # Đặt từ hiện tại ở đầu câu
while (curret_word_id+1) != len(text) {
  # xét từ hiện tại với 2 từ sau nó trong vocab tri_gram
  check if {“word, word+1, word+2”} is in tri_gram
  if true:
    take {word, word1, word2; current_word_id += 3} # 3 tiếng đó tạo thành từ ghép
  # xét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


START
sentence
bi_grams
tri_grams
syllables = syllablize(sentence)
syl_len = len(syllables)
curr_id = 0
word_list = []
done = False
(curr_id < syl_len) and (not done)
no
word_list
END
yes
curr_word = syllables[curr_id]
curr_id >= syl_len - 1
yes
word_list.append(curr_word)
done = True
no
next_word = syllables[curr_id + 1]
pair_word = curr_word.lower() + next_word.lower()
curr_id >= (syl_len - 2)
no
next_next_word = syllables[curr_id + 2]
triple_word = pair_word + next_next_word.lower()
yes
pair_word in bi_grams
no
word_list.append(curr_word)
curr_id += 1
yes
word_list.append(curr_word + next_word)
curr_id += 2
triple_word in tri_grams:
yes
word_list.append(curr_word + next_word + next_next_word)
curr_id += 3
no
pair_word in bi_grams:
yes
word_list.append(curr_word)
curr_id += 1
word_list.append(curr_word, next_word)
curr_id += 2

III.Phương pháp tách từ

2. Tách từ sử dụng 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



3. Đánh giá kết quả tách từ của VnCoreNLP và Longest Matchings

- Về bộ ngữ liệu dành cho tách từ: nhóm 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
- 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

Tạo bộ ngữ liệu

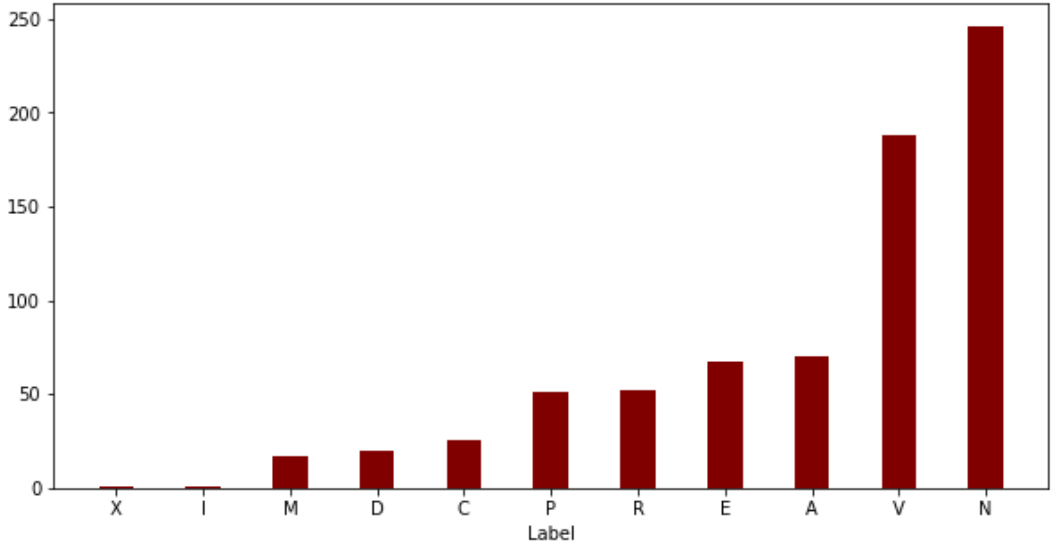
- Gán nhãn thủ công theo quy tắc
- Chia bộ dữ liệu thành 49 câu cho tập train và 13 câu cho tập test

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

https://vlsp.hpda.vn/demo/vcl/PoSTag.htm

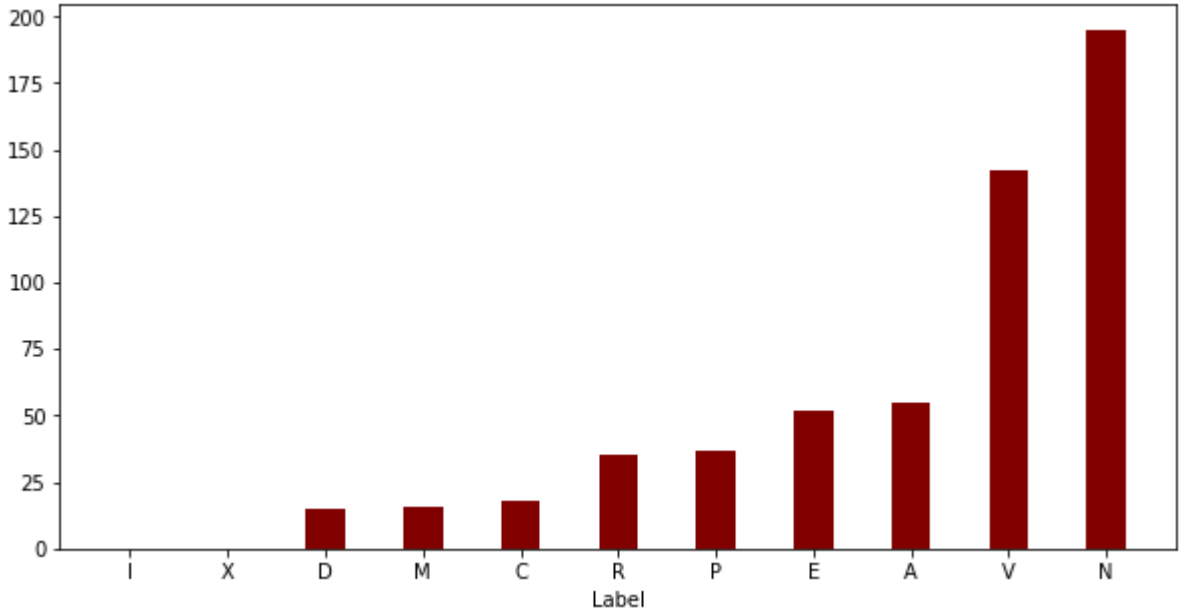
Tạo bộ ngữ liệu

- Ngữ liệu bao gồm 62 câu
- {'X': 1, 'I': 1, 'M': 17, 'D': 20, 'C': 25, 'P': 51, 'R': 52, 'E': 67, 'A': 70, 'V': 188, 'N': 246}
- Câu nhiều nhãn nhất: 22 nhãn
- Câu ít nhãn nhất: 4 nhãn
- Trung bình số lượng nhãn trong mỗi câu: 12 nhãn



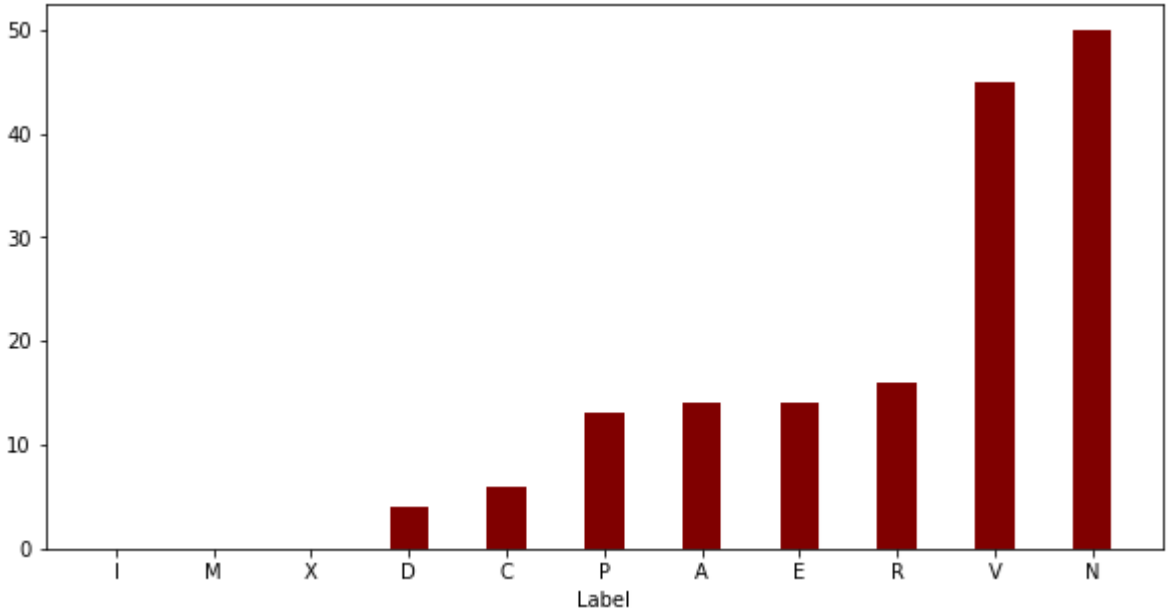
Tạo bộ ngữ liệu

- Bộ train
- {'I': 0, 'X': 0, 'D': 15, 'M': 16, 'C': 18, 'R': 35, 'P': 37, 'E': 52, 'A': 55, 'V': 142, 'N': 195}



Tạo bộ ngữ liệu

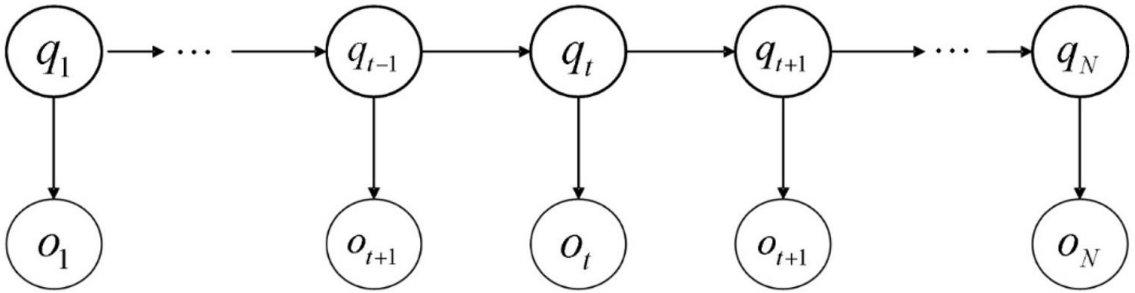
- Bộ test
- {'I': 0, 'M': 0, 'X': 0, 'D': 4, 'C': 6, 'P': 13, 'A': 14, 'E': 14, 'R': 16, 'V': 45, 'N': 50}



III. Phương pháp giải quyết bài toán

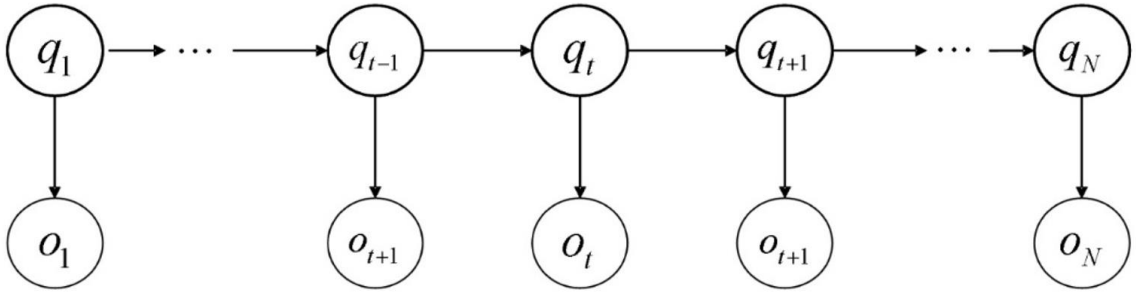
1. Mô hình Hidden Markov (HMM)

- 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.



- Ở đây q_i là các trạng thái ẩn, o_i là các trạng thái quan sát được.

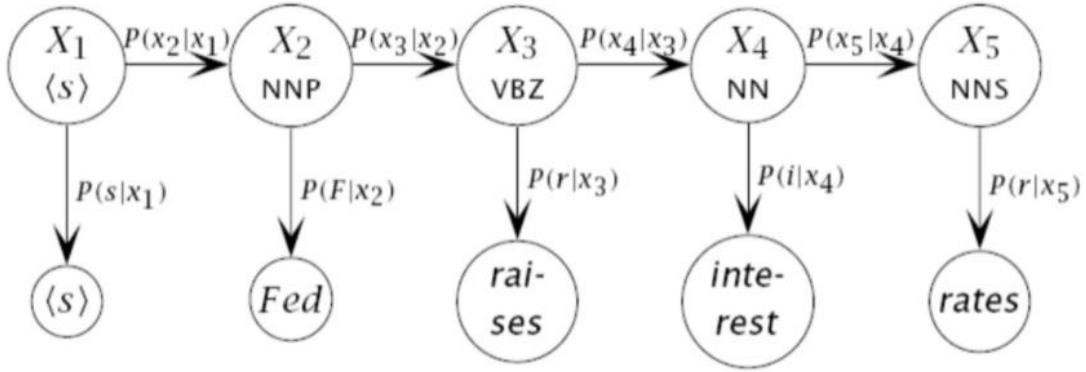
IV. Mô hình Markov



- Trong bài toán tách từ, trạng thái ẩn là các nhãn từ loại, trạng thái quan sát được là các từ trong ngữ liệu.
- Hidden Markov tổng quát bậc n, trạng thái q_i phụ thuộc vào n trạng thái đứng trước nó. Trong bài làm nhóm chỉ xét bậc 1, tức trạng thái q_i phụ thuộc vào trạng thái q_{i-1} và độc lập với các trạng thái khác:
  - $P(q_i|q_{i-1},q_{i-2},\ldots) = P(q_i|q_{i-1})$

IV. Mô hình Hidden Markov

- Biểu diễn mô hình Hidden Markov trong bài toán tách từ



IV. Mô hình Hidden Markov

- Tổng quát:
  - Một chuỗi các từ đã được tách S = {w_1, w_2 … w_n} là các giá trị quan sát
  - Tập hữu hạn các nhãn từ loại T = {t_1, t_2 … 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 … t_n để P(T* | S) lớn nhất:
  - T* = argmax P(T* | S)

  ⇔ T* = argmax P(w_1, w_2 … w_n | t_1, t_2 … t_n) P(t_1, t_2 … t_n) (1)
- Với Mô hình Hidden Markov bậc 1:
  - (1) ⇔ $T^* = \arg\max \prod_{i=1}^{n} P(w_i \mid t_i) \prod_{i=1}^{n} P(t_i \mid t_{i-1})$

IV. Mô hình Hidden Markov

- Để thuận tiện cho việc tính T*, HMM sử dụng 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 trạng 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 thể hiện, trong đó b_{ij} là xác suất trạng thái ẩn nhãn t_i thể hiện bằng giá trị quan sát w_j

IV. Mô hình Hidden Markov

- Ma trận Transition A
- Ma trận trận được smoothing với công thức:

	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

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

- N: tổng số nhãn
- $C(ti{-}1\ ,ti)$: số lượng giá trị cặp (prev_tag, tag)
- $C(ti{-}1)$: số lượng của nhãn prev_tag.
- α: tham số smoothing.

IV. Mô hình Hidden Markov

- Ma trận Emission B
- Ma trận trận được smoothing với công thức:

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

	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

- N: số lượng từ trong từ điển.
- $C(ti\ ,\ wi)$: số lượng của từ wi đi với nhãn ti
- $C(ti)$: số lần xuất hiện nhãn ti
- α: tham số smoothing.

IV. Mô hình Hidden Markov

2. 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 $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

IV. Mô hình Hidden Markov

Thuật toán Viterbi

- Thuật toán Viterbi nhóm thực hiện qua 3 bước chính:
  - Khởi tạo: khởi tạo 2 ma trận best_probs và best_paths
  - 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

IV. Mô hình Hidden Markov

Thuật toán Viterbi

- Khởi tạo:
  - Khởi tạo 2 ma trận có cùng chiều: (N x T) **“kiến_thức đó rất phức_tạp”**
    - 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>):
  - với nhãn ở vị trí i:
    - best_probs[0,i] = A[0,i] x B[i, corpus[0]]
    - best_probs[0,i] = log(A[0,i]) + log(B[i, corpus[0]])

	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

	kiến_thức
A	0.000036
C	0.000041
D	0.000041
E	0.000036
I	0.000044
M	0.000042
N	0.004797
P	0.000038
R	0.000037
V	0.000026

	kiến_thức	đó	rất	phức_tạp
A	-13.432310	0.0	0.0	0.0
C	-12.615773	0.0	0.0	0.0
D	-12.607610	0.0	0.0	0.0
E	-13.028505	0.0	0.0	0.0
I	-18.533749	0.0	0.0	0.0
M	-13.281737	0.0	0.0	0.0
N	-6.237567	0.0	0.0	0.0
P	-11.880628	0.0	0.0	0.0
R	-12.995715	0.0	0.0	0.0
V	-13.738634	0.0	0.0	0.0

IV. Mô hình Hidden Markov

Thuật toán Viterbi

- Forward để tính best_probs, best_path ở các từ tiếp theo:

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

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

	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

	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

IV. Mô hình Hidden Markov

Thuật toán Viterbi

- Backward:
  - Với mỗi hàng(nhãn) trên cột cuối cùng của best_probs, tìm ra giá trị lớn nhất.
  - Bắt đầu tại cột cuối cùng của best_paths, đối chiếu với best_probs 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)

	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

	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

```
['N', 'P', 'R', 'A']
```

```
kiến_thức/N đó/P rất/R phức_tạp/A
```

V. Thực nghiệm và đánh giá

1. HMM + Viterbi

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

V. Thực nghiệm và đánh giá

2. VnCoreNLP

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

V. Thực nghiệm và đánh giá

3. Tổng hợp

Phương pháp	Accuracy
HMM + Viterbi trên bộ train	78%
HMM + Viterbi trên bộ test	57%
VnCoreNLP trên bộ train	88%
VnCoreNLP trên bộ test	93%

V. Thực nghiệm và đánh giá

4. Đánh giá

- Dữ liệu quá ít không bao quát hết các trường hợp
- 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.

Thank you!

Any questions?