ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

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

BÁO CÁO ĐỒ ÁN
MÔN HỌC: 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 Chí Thắng 19522205
2. Đinh Quang Đông 20521189
3. Nguyễn Ngọc Lương 21522311

Thành phố Hồ Chí Minh, tháng 7 năm 2023

MỤC LỤC

I. Giới thiệu đề tài

- 1. Tổng quan đề tài

Gán nhãn từ loại (POS tagging), còn gọi là phân tích từ pháp, là việc xác định từ loại của một từ trong một câu xác định, bao gồm danh từ, động từ, tính từ, trạng từ và nhiều loại từ khác.

Gán nhãn từ loại được thực hiện để làm rõ nghĩa của từ, đồng thời cho thấy khả năng kết hợp của từ trong câu, từ đó góp phần tăng hiệu quả cho việc phân tích cú pháp.

Tuy nhiên, việc gán nhãn từ loại tiếng Việt còn đối mặt với một số thách thức, bao gồm sự đa nghĩa của từ, sự khác biệt trong cách sử dụng từ ngữ giữa các địa phương,...

Có 2 bước trong việc gán nhãn từ loại tiếng Việt:

- Tách từ: xác định ranh giới giữa các từ trong câu.
- Gán nhãn từ loại: xác định từ loại tương ứng cho các từ trong câu dựa vào định nghĩa của từ và ngữ cảnh của câu.

- 2. Mô tả bài toán

Input: Một câu tiếng Việt.

Output: Câu input kèm nhãn của từng từ trong câu.

Ví dụ: Input: Tai nạn xuất phát từ việc tài xế xe bán tải dừng lại để nhường cho một con rùa sang đường.

Output: Tai_nạn/**N** xuất_phát/**V** từ/**E** việc/**N** tài_xế/**N** xe/**N** bán/**Z** tải/**N** dừng/**V** lại/**V** để/**C** nhường/**V** cho/**E** một/**M** con/**N** rùa/**N** sang/**V** đường/**N** ./**CH**

Trong đồ án này, nhóm chúng em đã sử dụng hướng tiếp cận sau:

- Tách từ: Longest Matching.
- Gán nhãn từ loại: Hidden Markov Model.

II. Chuẩn bị ngữ liệu

- Ngữ liệu được thu thập từ trang báo online VnExpress, một trong những trang báo online tiếng Việt được nhiều người đọc nhất.

- Thống kê:
    + Số lượng: 62 câu.
    + Số từ nhiều nhất trong một câu là 40, số từ ít nhất trong một câu là 6.
    - Câu dài nhất: “Các chuyên gia cho rằng tuổi thọ trung bình người Việt là 74, gần 40% người vẫn tiếp tục làm việc sau tuổi 60, nên nghỉ hưu lúc nào còn tùy điều kiện sức khỏe, vật chất, khả năng lao động.”
    - Câu ngắn nhất: “Má tôi là mẹ đơn thân.”

III. Tách từ

- 1. Tách từ thủ công bằng từ điển VLSP

Nhóm chúng em thực hiện việc tách từ thủ công bằng từ điển online VLSP, link của từ điển tại đây. Nhóm thực hiện việc tra những từ có nhiều tiếng nhất có thể mà chúng em cho là có trong từ điển, nếu từ đó có trong từ điển thì thực hiện tách từ và chuyển sang từ kế tiếp, nếu không thì giảm số tiếng xuống và tiếp tục tra từ điển.

Bộ ngữ liệu tách từ thủ công được chúng em lưu tại file *hinhthai_full.txt*, bao gồm:

- Mỗi dòng là một từ
- Các tiếng trong một từ được nối với nhau bằng dấu “_”
- Mỗi câu được ngăn cách với nhau bởi một dòng trống
- Gồm có 959 dòng, bao gồm 62 dòng trống.

- 2. Tách từ bằng thuật toán Longest Matching

2.1 Giới thiệu

Thuật toán Longest Matching là một thuật toán so khớp tham lam, nó chạy từ trái sang phải của một câu, tìm từ dài nhất có thể có và so khớp với các từ trong từ điển, nếu từ đó xuất hiện trong từ điển thì sẽ được tách ra thành một từ.

2.2 Chi tiết thuật toán

Đầu tiên, xây dựng cho mình một bộ từ điển, nhóm chúng em sử dụng bộ từ điển Underthesea, được lưu trong file *Viet74k.txt*, từ điển này sẽ được chia ra thành 2 từ điển nhỏ hơn:

- bi_grams: chứa những từ gồm 2 tiếng tạo thành, gồm 49564 từ.

- tri_grams: chứa những từ gồm 3 tiếng tạo thành, gồm 5746 từ.

Các bước tiến hành thuật toán:

- Bước 1: Gọi V là danh sách các tiếng chưa xét của câu, nếu V=Ø thì dừng thuật toán, ngược lại, tiếp tục bước 2.
- Bước 2: Vị trí hiện tại là vị trí của từ đầu tiên trong V, xét tiếng tại vị trí hiện tại và hai tiếng liền kề sau nó, nếu ba tiếng đó tạo thành một từ xuất hiện trong từ điển tri_grams, ta tách ba tiếng đó ra khỏi V và xem ba tiếng đó là một từ; nếu không, thực hiện bước 3.
- Bước 3: Xét tiếng tại vị trí hiện tại và tiếng liền kề sau nó, nếu hai tiếng đó tạo thành một từ xuất hiện trong từ điển bi_grams, ta tách hai tiếng đó ra khỏi V và xem hai tiếng đó là một từ; nếu không, thực hiện bước 4.
- Bước 4: Tiếng hiện tại được tách ra khỏi V và xem đó là một từ.
- Bước 5: Quay lại bước 1.

Ví dụ: Tách từ cho câu: "Tôi cũng tủi thân mỗi khi nhìn vào gian nhà lụp xụp của mình."

Bước	Từ dài nhất có thể	Các tiếng còn lại (V)
1	Tôi	cũng tủi thân mỗi khi nhìn vào gian nhà lụp xụp của mình
2	cũng	tủi thân mỗi khi nhìn vào gian nhà lụp xụp của mình
3	tủi thân	mỗi khi nhìn vào gian nhà lụp xụp của mình
4	mỗi	khi nhìn vào gian nhà lụp xụp của mình
5	khi	nhìn vào gian nhà lụp xụp của mình

6	nhìn	vào gian nhà lụp xụp của mình
7	vào	gian nhà lụp xụp của mình
8	gian	nhà lụp xụp của mình
9	nhà	lụp xụp của mình
10	lụp xụp	của mình
11	của	mình
12	mình	

2.3 Nhận xét

- Ưu điểm:
  + Thuật toán đơn giản, dễ cài đặt và sử dụng.
  + Không yêu cầu bước huấn luyện nên thời gian xử lý nhanh.
  + Độ chính xác tương đối cao nếu xây dựng bộ từ điển đủ tốt.
- Nhược điểm:
  + Phụ thuộc hoàn toàn vào tính đầy đủ và chính xác của từ điển.
  + Nếu xảy ra những vấn đề đặt dấu khác nhau (oà, uý) thì từ điển sẽ bị so khớp sai lệch.

    Ví dụ: Từ "**hoà** thuận" đáng lẽ phải được coi như một từ có hai tiếng, tuy nhiên trong từ điển viết là "**hòa** thuận" nên thuật toán sẽ tách từ đó thành 2 từ là "hòa" và "thuận".

  + Vì chỉ chú trọng vào việc so khớp với từ điển, do đó từ điển sẽ có thể bị tách từ nhầm lẫn nghĩa của các câu.

    Ví dụ: Tách từ cho câu: "Ông ấy là một ông quan tài giỏi."

    - Output theo Longest Matching: Ông ấy là một ông quan_tài giỏi.

- Output đúng mong muốn: Ông ấy là một ông quan tài_giỏi.

- 3. Tách từ sử dụng thư viện VnCoreNLP

Nhóm sử dụng thư viện VnCoreNLP để thực hiện tách từ và so sánh kết quả.

Đầu tiên, cần cài đặt thư viện viện VnCoreNLP bằng câu lệnh:

```
!pip install py_vncorenlp
```

Download model VnCoreNLP và khởi tạo nó để bắt đầu thực hiện việc tách từ.

```
import py_vncorenlp
py_vncorenlp.download_model(save_dir='/content/drive/MyDrive/NLP_Data/VnCoreNLP')
```

```
#Khởi tạo model của VnCoreNLP
model = py_vncorenlp.VnCoreNLP(save_dir='/content/drive/MyDrive/NLP_Data/VnCoreNLP')
```

Duyệt qua từng câu của bộ ngữ liệu và bắt đầu tách từ, lưu kết quả vào mảng *vncorenlp_sentences*, sau đó lưu vào file *vncorenlp_token.txt*.

```
with open('/content/drive/MyDrive/NLP_Data/vncorenlp_token.txt', 'w', encoding='utf-8') as f:
    vncorenlp_sentences = []
    for sentence in sentences:
        vncore_sent = model.annotate_text(sentence)
        sent = vncore_sent[0]

        word_list = []
        for a in sent:
            word_list.append(a['wordForm'])

        vncorenlp_sentences.append(' '.join(word_list))
        for word in word_list:
            f.write(word + '\n')
        if sentence != sentences[-1]:
            f.write('\n')
    f.write('\n')
```

- 4. So sánh kết quả

Dưới đây là kết quả tách từ của thuật toán Longest Matching và thư viện VnCoreNLP khi so sánh với bộ ngữ liệu tách từ thủ công của nhóm.

	Longest Matching	VnCoreNLP

Accuracy	0.731	0.970
Precision	0.66	0.935
Recall	0.766	0.944
True Positive	164	202
False Positive	82	14
Total True	648	859
Total Errors	212	24

Nhìn vào bảng trên, ta thấy rằng thuật toán Longest Matching hoạt động chưa thực sự tốt với bộ ngữ liệu của nhóm, lý do là bởi bộ từ điển của Underthesea chưa thực sự đủ tốt với bộ ngữ liệu của nhóm em.

Số từ ghép mà các phương pháp có được:

- Thủ công: 214 từ
- Longest Matching: 250 từ
- Thư viện VnCoreNLP: 220 từ

Ví dụ về một câu cụ thể mà 3 phương pháp tách được:

Phương pháp	Câu được tách từ
Thủ công	Ông Tâm cho_hay dù ngư_dân kêu to, tàu hàng không dừng lại.
Longest Matching	Ông Tâm cho_hay dù ngư_dân kêu to, tàu hàng_không dừng lại.
VnCoreNLP	Ông Tâm cho_hay dù ngư_dân kêu to, tàu hàng không dừng lại.

Trong ví dụ trên, thuật toán Longest Matching đã so khớp thấy từ "hàng không" có trong bộ từ điển, nên nó đã tách thành một từ riêng, tuy nhiên, trong ngữ cảnh của câu này, từ "hàng" và "không" phải là hai từ riêng biệt.

IV. Tạo ngữ liệu gán nhãn

Với ngữ liệu đã được tách từ trong file hinhthai_full.txt, nhóm chúng em thực hiện gán nhãn thủ công với một số quy tắc:

- Dấu câu (dấu chấm, dấu phẩy, dấu hai chấm,...) được gán nhãn CH
- Các số (4, 5,...) được gán nhãn là M
- Nhãn được gán theo bộ nhãn của từ điển VLSP

POS

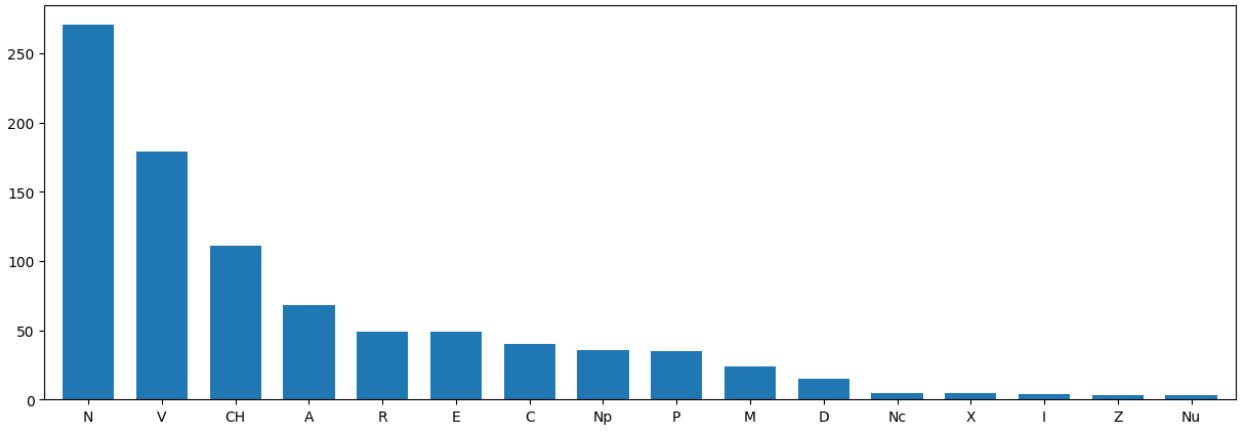
Stt	idPOS	vnPOS	enPOS
1	N	danh từ	noun
2	V	động từ	verb
3	A	tính từ	adjective
4	P	đại từ	pronoun
5	M	số từ	numeral
6	D	định từ (những, các, vài...)	determiner
7	R	phụ từ	adverb
8	E	giới từ	preposition
9	C	liên từ	conjunction
10	I	trợ từ	auxiliary word
11	O	cảm từ	emotivity word
12	Z	yếu tố cấu tạo từ (bất, vô…)	component stem
13	X	không (hoặc chưa) xác định	undetermined

Bộ nhãn của từ điển VLSP

Sau khi đã gán nhãn xong, dữ liệu gán nhãn được lưu trong file tuloai_full.txt với các thông tin sau:

- Các tiếng trong từ được nối với nhau bằng dấu "_"
- Một từ được phân cách với nhãn của nó bằng một dấu tab
- Mỗi dòng là một từ và nhãn của nó
- Giữa các câu với nhau được ngăn cách bằng một dòng trống
- Số lượng câu: 62
- Số dòng: 959 (bao gồm 62 dòng trống)
- Số nhãn: 897

	N	V	CH	A	R	E	C	Np	P	M	D	Nc	X	I	Z	Nu	Total
0	271	179	111	68	49	49	40	36	35	24	15	5	5	4	3	3	897



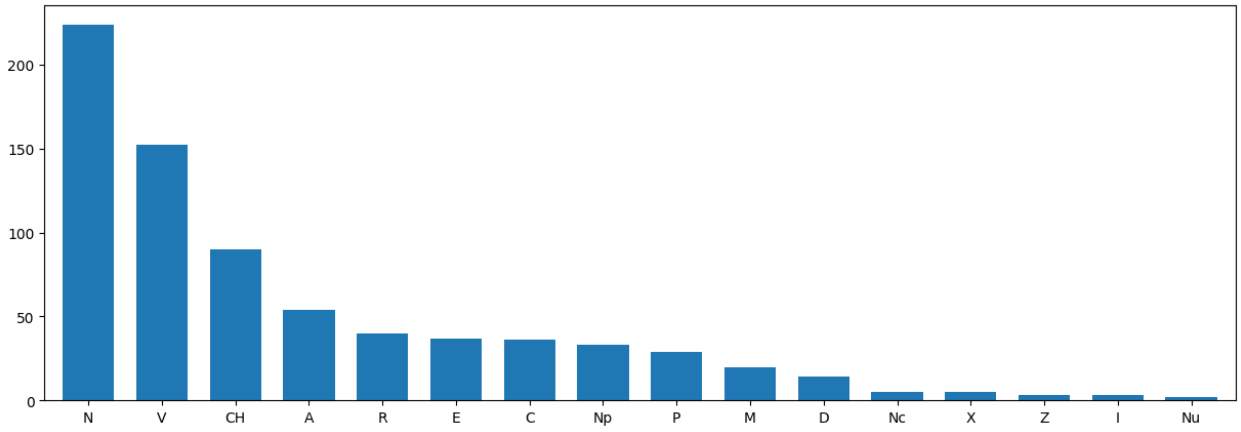
Thống kê các nhãn có trong bộ ngữ liệu

Sau đó, nhóm thực hiện việc chia dữ liệu ra thành 2 tập: train và test. Tập train gồm có 50 câu đầu, tập test gồm có 12 câu sau.

Tập train gồm có các file train_gold.txt và file train_words.txt:

- train_gold.txt: chứa các từ kèm nhãn của tập train
- train_words.txt: chỉ chứa các từ của tập train
- Số câu: 50
- Số dòng: 797 (bao gồm 50 dòng trống)
- Số nhãn: 747

	N	V	CH	A	R	E	C	Np	P	M	D	Nc	X	Z	I	Nu	Total
0	224	152	90	54	40	37	36	33	29	20	14	5	5	3	3	2	747

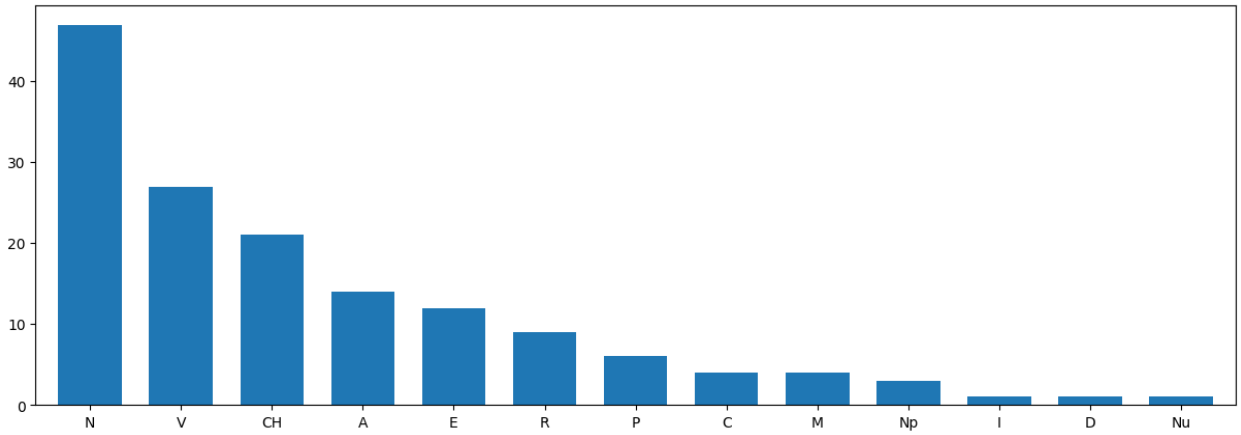


Thống kê các nhãn trong tập train

Tập test gồm có các file test_gold.txt và file test_words.txt:

- test_gold.txt: chứa các từ kèm nhãn của tập test
- test_words.txt: chỉ chứa các từ của tập test
- Số câu: 12
- Số dòng: 162 (bao gồm 12 dòng trống)
- Số nhãn: 150

	N	V	CH	A	E	R	P	C	M	Np	I	D	Nu	Total
0	47	27	21	14	12	9	6	4	4	3	1	1	1	150



Thống kê các nhãn trong tập test

V. Gán nhãn từ loại

- 1. Chuẩn bị gán nhãn

Để phục vụ cho việc gán nhãn, nhóm tạo ra một số từ điển (dictionary) sẽ giúp tạo ra các bảng. Ngoài ra, nhóm thêm nhãn "--s–" để chỉ ra phần bắt đầu của mỗi câu.

- Từ điển Transition Counts:
  + Tính số lần xuất hiện của các cặp nhãn (số lần xuất hiện của một nhãn bên cạnh một nhãn khác).
  + Key của từ điển là cặp nhãn (prev_tag, tag), value là số lần xuất hiện của cặp nhãn đó.
- Từ điển Emission Counts:
  + Tính số lần xuất hiện của một từ với một nhãn nào đó.
  + Key của từ điển là cặp (tag, word), value là số lần xuất hiện của cặp đó trong tập train.
- Từ điển Tag Counts:
  + Đếm số lần xuất hiện của từng nhãn.
  + Key của từ điển là nhãn, value là số lần nhãn đó xuất hiện.

- 2. Hidden Markov Model

2.1 Markov Chain

Là một dạng FSA được dùng để mô hình hoá xác suất của các biến ngẫu nhiên có quan hệ với nhau theo dạng chuỗi. Markov Chain đưa ra 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 quá khứ mà chỉ bị ảnh hưởng bởi trạng thái hiện tại. Ví dụ, để dự đoán thời tiết ngày mai, có thể dựa vào thời tiết hôm nay nhưng không thể dựa vào thời tiết hôm qua. Tổng quát, xét một chuỗi các biến ngẫu nhiên $q1, q2, \dots, qi$. 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:

$P(qi = a|q1 \dots qi-1) = P(qi = a|qi-1)$

Một Markov Chain là một bộ <Q, δ, q_0>, trong đó:

- Q là tập các trạng thái
- δ là hàm chuyển đổi trạng thái có trọng số. Trọng số trong δ là xác suất chuyển trạng thái tương ứng.

- q_0 là trạng thái bắt đầu.

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.

Hidden Markov Model gồm:

- S = $\{s_1, s_2,..., s_n\}$ là tập các trạng thái ẩn
- Trạng thái đặc biệt s_0 là trạng thái bắt đầu.
- K = $\{k_1, k_2,..., k_m\}$ là tập các giá trị quan sát.
- A = $\{a_{ij}\}$, (i, j = 1…n) là ma trận chuyển trạng thái, trong đó a_{ij}là xác suất chuyển từ trạng thái s_i sang trạng thái s_j.
- B = $\{b_{ij}\}$ (i=1…n, j=1…m) là ma trận emission (thể hiện), trong đó b_{ij} là xác suất trạng thái ẩn s_i thể hiện bằng giá trị quan sát k_j.

Một số ứng dụng của Hidden Markov Model:

- Nhận dạng giọng nói
- Xử lý ngôn ngữ tự nhiên: phân tích ngữ pháp, phân tích cú pháp, dịch máy.
- Phân tích chuỗi DNA

2.3 Ma trận chuyển trạng thái A

	--S--	A	C	CH	D	E	I	M
--S--	0.000203	0.020541	0.061216	0.000203	0.061216	0.000203	0.000203	0.040879
A	0.000185	0.055566	0.074026	0.314011	0.000185	0.110947	0.000185	0.037105
C	0.000276	0.027924	0.000276	0.055571	0.055571	0.000276	0.000276	0.055571
CH	0.543529	0.033381	0.044472	0.011201	0.022291	0.011201	0.000111	0.011201
D	0.000706	0.000706	0.000706	0.000706	0.000706	0.000706	0.000706	0.000706

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:

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

Trong đó:

– fij: tần số xuất hiện của cặp nhãn thứ ij

– fi: 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

	"	,	-	.	0	12	16	20
--S--	0.000188	0.000188	0.000188	0.000188	0.000188	0.000188	0.000188	0.000188
A	0.000171	0.000171	0.000171	0.000171	0.000171	0.000171	0.000171	0.000171
C	0.000248	0.000248	0.000248	0.000248	0.000248	0.000248	0.000248	0.000248
CH	0.042510	0.371144	0.010707	0.530160	0.000106	0.000106	0.000106	0.000106
D	0.000546	0.000546	0.000546	0.000546	0.000546	0.000546	0.000546	0.000546
E	0.000242	0.000242	0.000242	0.000242	0.000242	0.000242	0.000242	0.000242
I	0.001364	0.001364	0.001364	0.001364	0.001364	0.001364	0.001364	0.001364
M	0.000411	0.000411	0.000411	0.000411	0.041513	0.041513	0.041513	0.041513
N	0.000044	0.000044	0.000044	0.000044	0.000044	0.000044	0.000044	0.000044

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:

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

Trong đó:

– fij tần số xuất hiện của cặp nhãn thứ i và từ thứ j

– fi 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 hai ma trận có cùng kích thước:

- Ma trận best_prob chứa xác suất của một nhãn đến một 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:

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

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

Các bước thực hiện:

- Bước 1: Lần lượt duyệt qua từng từ, từng nhãn của từ đang xét và nhãn của từ trước đó.
- Bước 2: 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.
- Bước 3: Tính toán đường đi đến vị trí đó rồi điền vào bảng best_path.

Bảng best_prob có các giá trị được tính bằng:

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

Trong đó:

- $words_i$: từ thứ i trong câu đang xét
- $vocabs[words_i]$: 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 hai chứ vị trí của nhãn i-1 có best_prob cao nhất.

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.

Cách thực hiện:

- Duyệt qua các dòng cuối cùng của bảng best_prob: Tìm hàng có giá trị cao nhất và trả về hàng đó.
- Duyệt qua lần lượt từng cột của bảng best_path từ cột cuối lên cột đầu tiên:
  + Từ vị trí của hàng có giá trị cao nhất ở bước trên, so khớp với cột cuối của bảng best_path để trả về vị trí của nhãn của từ trước đó.
  + Lần lượt cập nhật nhãn của các từ.

4

VI. Nhận xét

- 1. Tách từ

Thuật toán Longest Matching có một số ưu điểm đáng kể. Đầu tiên, nó dễ dàng để cài đặt và sử dụng. Người dùng chỉ cần có một từ điển để dựa vào và thuật toán sẽ tự động thực hiện các phép so sánh và tính toán để đưa ra kết quả. Điều này giúp cho việc triển khai thuật toán trở nên nhanh chóng và thuận tiện.

Tuy nhiên, thuật toán này cũng có một số hạn chế. Đầu tiên, độ chính xác của thuật toán hoàn toàn phụ thuộc vào độ chính xác của từ điển. Nếu từ điển không đầy đủ hoặc cung cấp định nghĩa không chính xác cho các từ, kết quả trả về có thể bị sai lệch hoặc không chính xác.

Một khó khăn khác đối với thuật toán này là việc xử lý các cụm từ theo thứ tự từ trái qua phải. Điều này có thể dẫn đến một số trường hợp kết quả không chính

xác. Bởi vì thuật toán không lường trước các kết hợp từ phía sau có thể tạo thành từ mới hoặc cụm từ có ý nghĩa phù hợp với ngữ cảnh.

Để cải thiện thuật toán, có thể xem xét việc mở rộng từ điển hoặc áp dụng các kỹ thuật xử lý ngôn ngữ tự nhiên nâng cao. Các phương pháp như xử lý ngữ cảnh, xác suất ngôn ngữ, hoặc mô hình học sâu có thể giúp cải thiện độ chính xác và khả năng nhận diện các cụm từ có ý nghĩa hơn trong văn bản.

- 2. Gán nhãn từ loại

Thư viện Underthesea thường cho kết quả gán nhãn từ vựng tốt hơn so với Hidden Markov Model. Có một số nguyên nhân giải thích điều này:

Kích thước tập dữ liệu huấn luyện: Thông thường, kích thước của tập dữ liệu huấn luyện đối với Hidden Markov Model (HMM) là khá nhỏ. Với một tập dữ liệu nhỏ, HMM có khả năng kháng nhiễu và khả năng tổng quát hóa thấp. Điều này làm cho việc xác định nhãn trở nên khó khăn và có thể dẫn đến kết quả không chính xác. Trong khi đó, Underthesea được huấn luyện trên các tập dữ liệu lớn hơn và đa dạng hơn, giúp cải thiện khả năng xử lý và đưa ra nhãn chính xác hơn.

Mẫu kiểm tra khó khăn: Trong một số trường hợp, mẫu kiểm tra có thể quá khó cho Hidden Markov Model xử lý. Có thể xảy ra phân bố không đồng đều về số lượng từ vựng trong mỗi nhãn giữa tập huấn luyện và tập kiểm tra. Điều này có nghĩa là có một số nhãn có ít dữ liệu huấn luyện, không đủ để phân biệt chính xác, trong khi tập kiểm tra lại có nhiều từ thuộc các nhãn này. Điều này gây khó khăn cho HMM trong việc đưa ra các nhãn chính xác. Trong khi đó, Underthesea có thể xử lý tốt hơn các mẫu kiểm tra khó khăn này nhờ sử dụng các phương pháp xử lý ngôn ngữ tự nhiên tiên tiến và được huấn luyện trên dữ liệu phong phú.

Số lượng từ vựng: Hidden Markov Model thường cần có một số lượng từ vựng đủ lớn để xây dựng mô hình phù hợp. Nếu tập từ vựng đưa vào không đủ lớn, HMM có thể gặp khó khăn trong việc đưa ra các nhãn chính xác. Underthesea, với tập từ vựng lớn hơn, có thể cung cấp kết quả tốt hơn trong việc gán nhãn từ vựng.

Tóm lại, Underthesea thường cho kết quả gán nhãn từ vựng tốt hơn so với Hidden Markov Model do kích thước tập dữ liệu huấn luyện lớn hơn, khả năng xử lý mẫu kiểm tra khó và sử dụng một tập từ vựng đa dạng hơn.

VII. Phân công công việc

	Nguyễn Chí Thắng	Đinh Quang Đông	Nguyễn Ngọc Lương
Thu thập ngữ liệu	Có tham gia	Có tham gia	Có tham gia
Tách từ và gán nhãn thủ công	Có tham gia	Có tham gia	Có tham gia
Hiện thực hóa bằng python	Có tham gia	Có tham gia	Có tham gia
Đánh giá và nhận xét	Có tham gia	Có tham gia	Có tham gia
Slide	Có tham gia	Có tham gia	Có tham gia
Thuyết trình	Có tham gia	Có tham gia	Có tham gia
Báo cáo	Có tham gia	Có tham gia	Có tham gia

Đánh giá tiến độ hoàn thành	Hoàn thành 100% công việc được phân công	Hoàn thành 100% công việc được phân công	Hoàn thành 100% công việc được phân công

VIII. Tài liệu tham khảo

1. Longest Matching
2. Hidden Markov & Viterbi