ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO ĐỒ ÁN

XỬ LÝ NGÔN NGỮ TỰ NHIÊN - CS212

GÁN NHÃN TỪ LOẠI TIẾNG VIỆT SỬ DỤNG MÔ HÌNH HIDDEN MARKOV VÀ THUẬT TOÁN VITERBI

Giáo viên hướng dẫn:

Th.S Nguyễn Trọng Chỉnh

GV Đặng Văn Thìn

Sinh viên thực hiện:

Ngọ Viết Dũng - 17520375

Nguyễn Văn Thiện Tâm - 18521369

Đặng Phước Sang - 21521377

TP. Hồ Chí Minh, tháng 06 năm 2023

LỜI CẢM ƠN

Để hoàn thành đồ án môn học này, nhóm em xin chân thành cảm ơn đến:

Giảng viên lý thuyết - Th.S Nguyễn Trọng Chỉnh đã giảng dạy tận tình, chi tiết để chúng em có đủ kiến thức và vận dụng chúng vào đồ án này.

Giảng viên thực hành - Anh Đặng Văn Thìn đã hướng dẫn, góp ý và cung cấp các bộ dữ liệu cần thiết giúp chúng em hoàn thành đồ án này.

Do chưa có nhiều kinh nghiệm và hạn chế về kiến thức, lý luận của bản thân, kính mong nhận được những nhận xét, ý kiến đóng góp của các Thầy để đồ án môn học của chúng em được hoàn thiện hơn.

Nhóm em xin trân trọng cảm ơn!

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

CHƯƠNG I: GIỚI THIỆU

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, dựa theo định nghĩa và bối cảnh văn phạm của từ đó. Đây là một bài toán cơ bản trong Xử lý ngôn ngữ tự nhiên.

Phân biệt các từ loại trong câu giúp ta hiểu rõ hơn về ý nghĩa, là bước tiền xử lý quan trọng cho các bài toán khác như phân tích cú pháp, tìm kiếm văn bản, nhận dạng thực thể, …

Các bài toán gãn nhãn từ loại luôn đi kèm với những thách thức khó khăn. Khó khăn đầu tiên là sự nhập nhằng (ambiguity), thể hiện qua việc một từ có thể được gán nhiều từ loại phụ thuộc vào ngữ cảnh của văn bản. Ví dụ, trong câu "Con ngựa đá con ngựa đá.", ta thấy từ "đá" đầu tiên là động từ, trong khi từ "đá" còn lại là tính từ.

Ngoài ra, trong thực tế có nhiều từ không xuất hiện trên ngữ liệu huấn luyện (training corpus) gây ra nhiều khó khăn khi xây dựng mô hình gán nhãn.

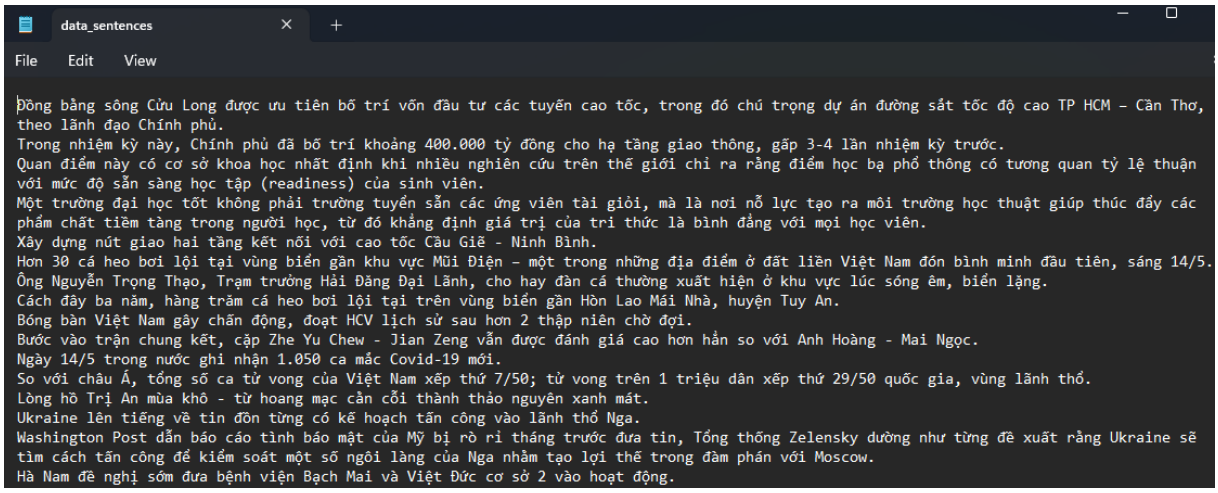
Trong đề tài này, nhóm xin được trình bày phương pháp gán nhãn từ loại sử dụng mô hình Hidden Markov kết hợp với thuật toán Viterbi để gán nhãn từ loại tiếng Việt, đồng thời so sánh với một số thư viện gán nhãn thông dụng hiện nay.

CHƯƠNG II: THU THẬP DỮ LIỆU

2.1 Nguồn dữ liệu thu thập

Dữ liệu được thu thập là các câu ngẫu nhiên thuộc nhiều chủ đề, lĩnh vực khác nhau.

Nguồn dữ liệu thu thập từ các trang báo uy tín ở Việt Nam như https://dantri.com.vn/, https://thanhnien.vn/, https://vnexpress.net/. Các câu được thu thập có thể bao gồm tiêu đề và nội dung bên trong.



Hình 2.1 Một số câu trong bộ dữ liệu gốc

2.2 Nhận xét dữ liệu

Bộ dữ liệu được lưu trữ trong file data_sentences.txt:

- Mỗi dòng là một câu, cuối câu kết thúc bằng dấu chấm "." hoặc dấu ba chấm "…" hoặc dấu chấm hỏi "?"

- Các từ được phân cách với nhau bởi dấu khoảng cách " "

- Số lượng câu: 90

- Chứa một số từ viết tắt cho tiếng Việt và tiếng Anh (VD: HĐND, HCV, IELTS, TP.HCM, …), từ tiếng Anh (VD: readness, …), tên riêng nước ngoài (VD: IELST, Zelensky, Joe Biden, …)

- Chứa các dạng số nguyên được ngăn cách bởi dấu chấm (VD: 1.000.0000), các dạng ngày tháng (VD: 30-4, 1/5, …), số thập phân.

CHƯƠNG III: TÁCH TỪ

3.1 Giới thiệu

Tách từ (Word Segmentation) là một bài toán cơ bản trong Xử lý ngôn ngữ tự nhiên với tiếng Việt, đồng thời là bước tiền xử lý quan trọng cho các bài toán khác. Tách từ là một quá trình xử lý nhằm mục đích xác định ranh giới của các từ trong câu văn, cũng có thể hiểu đơn giản rằng tách từ là quá trình xác định các từ đơn, từ ghép… có trong câu.

Khác với tiếng Anh, một số từ tiếng Việt có thể được tạo ra bởi nhiều âm hay tiếng (syllable). Xét ví dụ từ "cá thể" được tạo ra bởi hai âm là "cá" và "thể", các từ đơn "cá" và "thể" lại có thể mang ý nghĩa khác so với từ "cá thể". Vì vậy ta cần dùng dấu gạch dưới "_" để liên kết hai âm của từ này thành "cá_thể".

Ngoài việc xác định các từ có nhiều âm tiết, chúng ta cũng cần tách các dấu câu riêng khỏi từ. Ví dụ câu "Hôm nay, tôi đi học." ta cần tách dấu "," khỏi từ "nay" và dấu chấm "." khỏi từ "học". Đây là quy ước chung cho tất cả các ngôn ngữ của bài toán tách từ trong xử lý ngôn ngữ tự nhiên. Việc quy ước như vậy là để tạo thành chuẩn chung và để dễ xử lý hơn trong lập trình.

Phát biểu bài toán: Đầu vào là một câu hay văn bản tiếng Việt, đầu ra là câu hay văn bản đã được tách từ.

Ví dụ của tách từ:

- Input: "U22 Indonesia vô địch SEA Games sau trận đấu có 7 bàn thắng, 7 thẻ đỏ."

- Output: "U22 Indonesia vô_địch SEA_Games sau trận đấu có 7 bàn thắng , 7 thẻ_đỏ ."

3.2 Thuật toán Longest Matching

3.2.1 Giới thiệu

Longest Matching là thuật toán dựa trên chiến lược tham lam (Greedy). Ý tưởng của thuật toán là xét các tiếng từ trái sang phải, các tiếng đầu tiên dài nhất xuất hiện trong từ điển sẽ được tách thành một từ. Thuật toán sẽ dừng khi xét hết các tiếng trong câu.

Nhóm sử dụng bộ dữ liệu từ điển gồm 31158, lưu trong file Dictionary.txt. Nhóm sẽ chia bộ dữ liệu thành các file bi_grams.txt chỉ chứa các từ có 2 tiếng, file tri_grams.txt chỉ chứa các từ có 3 tiếng, file quadri_grams.txt chỉ chứa các từ có 4 tiếng và file penta_grams.txt chi chứa các từ có 5 tiếng. Sau đó tiến hành cài đặt thuật toán.

3.2.2 Mã giả

```
W = []
V = sentence #Mảng tập hợp các tiếng trong câu đang xét
#word là tiếng đang xét, word + 1 là tiếng tiếp theo
while V ≠ Ø
        five_word = (word, word + 1, word + 2, word + 3, word + 4)
        four_word = (word, word + 1, word + 2, word + 3)
        three_word = (word, word + 1, word + 2)
        two_word = (word, word + 1)
        if five_word  in penta_grams:
                W.append(five_word)
                word = word + 5
        else if four_word in quadri_grams:
                W.append(four_word)
                word = word + 4
        else if three_word in tri_grams:
                W.append(three_word)
                word = word + 3
        else if two_word in two_grams:
                W.append(two_word)
                word = word + 2
        else:
                W.append(word)
                word = word + 1
return W
```

3.2.3 Ưu và nhược điểm

Ưu điểm:

- Thời gian xử lý tương đối nhanh.

- Cài đặt đơn giản.

- Độ chính xác tương đối cao.

Nhược điểm:

- Độ chính xác phụ thuộc hoàn toàn vào tính đầy đủ và tính chính xác của từ điển.

- Khó có thể xử lý được các tình huống nhập nhằng.

- Không thể nhận ra các từ ghép ngoài từ điển.

3.3 Triển khai tách từ

Các bước thực nghiệm tách từ tiếng Việt của nhóm:

- Tạo dữ liệu tách từ thủ công: nhóm tiến hành phân công thực hiện tách từ thủ công, dựa vào từ điển VLSP (https://vlsp.hpda.vn/demo/?page=vcl) tạo thành dữ liệu tách từ chuẩn, lưu lại trong file manual_tokens.txt.

- Cài đặt thuật toán Longest Matching, tiến hành tách từ cho tất cả 90 câu đã thu thập, lưu kết quả tách từ vào file longest_matching_tokens.txt.

- Sử dụng thư viện VnCoreNLP tách từ cho tất cả 90 câu, lưu kết quả vào file vncore_tokens.txt.

- Sử dụng thư viện pyvi tách từ cho tất cả 90 câu, lưu kết quả vào file pyvi_tokens.txt.

- So sánh phương pháp Longest Matching với các thư viện, đưa ra nhận xét.

Khi thực hiện tách từ thủ công, nhóm thông nhất một số quy ước như sau:

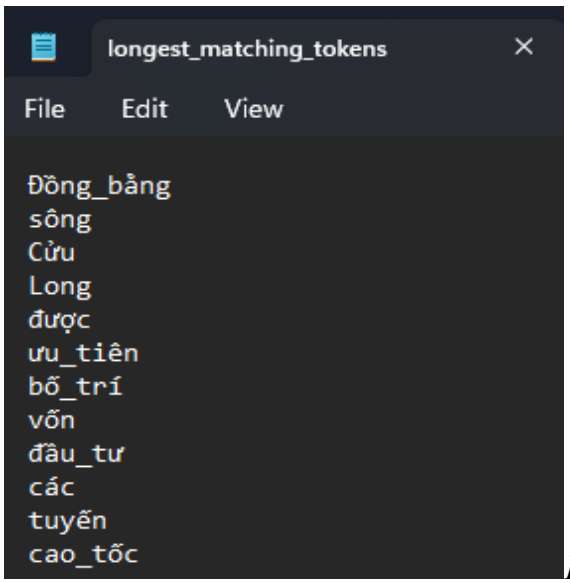
- Giữ nguyên các từ viết tắt tiếng Việt và tiếng Anh

- Giữ dấu của số thập phân, phân số, dấu định dạng ngày tháng thay vì tách ra giống như với các từ. (Ví dụ: “3.000.000” được giữ nguyên thay vì tách thành “3000000”, “.” và “.”)

- Đối với từ có các âm được nối bởi dấu gạch nối “-“ gắn liền (Ví dụ: “Covid-19”) sẽ được giữ nguyên.

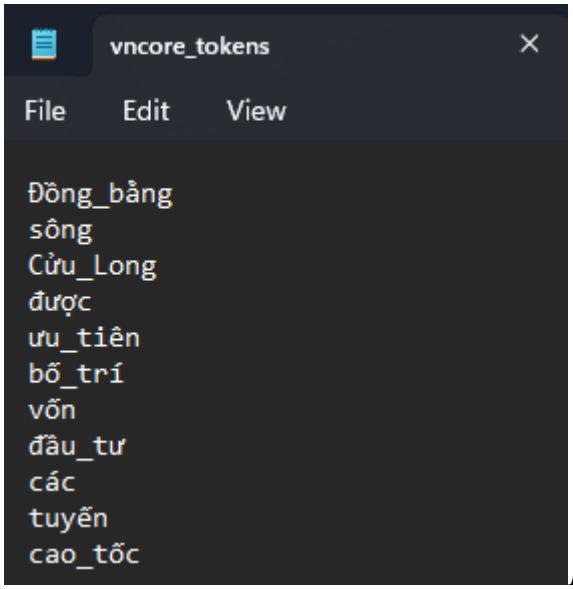
3.4 Đánh giá kết quả và so sánh

- Với thuật toán Longest Matching: số lượng từ ghép được là 570



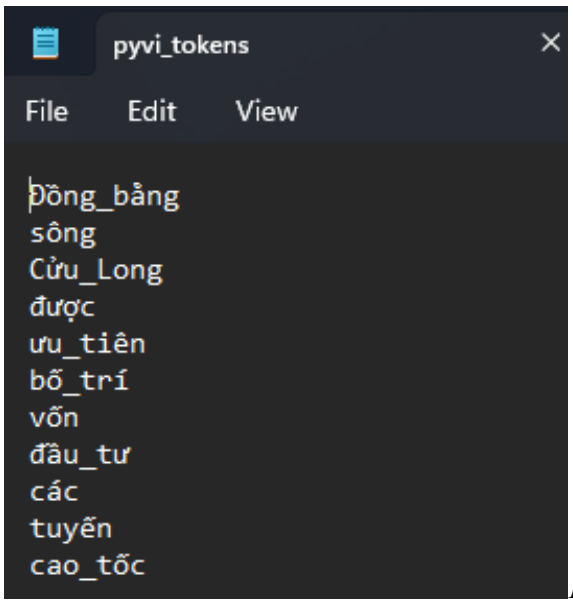
Hình 3.1 Kết quả tách từ của thuật toán Longest Matching

- Với thư viện VnCoreNLP, số lượng từ ghép được là 622



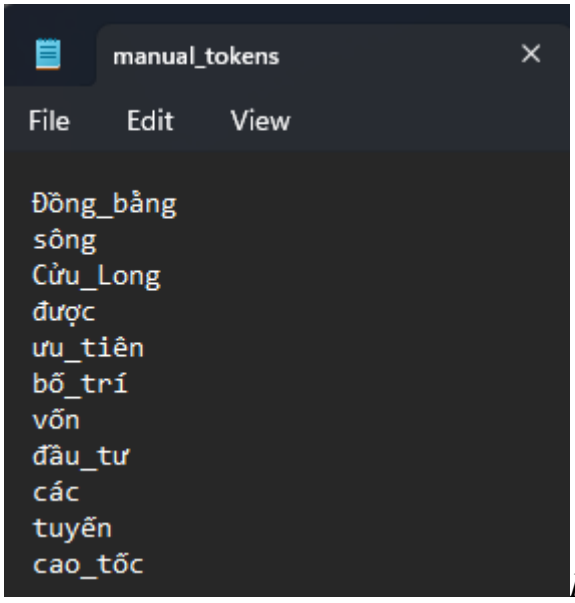
Hình 3.2 Kết quả tách từ của thư viện VnCoreNLP

- Với thư viện pyvi, số lượng từ ghép được là 627



Hình 3.3 Kết quả tách từ của thư viện pyvi

- Với phương pháp thủ công, số lượng từ tách được là 641



Hình 3.4 Kết quả tách từ thủ công

	Longest Matching	pyvi	VnCoreNLP
Accuracy	0.90591	0.914894	0.964539
Precision	0.910211	0.888000	0.967742
Recall	0.806552	0.865835	0.936037
True Positive	517	555	600
False Positive	51	70	20
Total True	1916	1935	2040
Total Errors	304	221	91

Bảng 3.1 Kết quả đánh giá các phương pháp tách từ so với phương pháp thủ công

Qua bảng trên, ta thấy được thư viện VnCoreNLP hoạt động hiểu quả nhất, tốt hơn so với pyvi. Thuật toán Longest Matching hoạt động kém hiệu quả nhất.

Độ phủ (Recall) của thuật toán Longest Matching tương đối thấp so với Accuracy và Precision, vì vậy kết quả tách từ này chưa tốt. Ta có thể thấy thuật toán này không thể phân biệt tốt các tên riêng (VD: sông Cửu_Long ở hình 3.1), do chúng không xuất hiện trong dữ liệu từ điển.

Nhóm sẽ sử dụng kết quả tách từ thủ công (manual_tokens.txt) để tiến hành gán nhãn từ loại ở chương tiếp theo.

CHƯƠNG IV: GÁN NHÃN TỪ LOẠI

4.1 Tạo ngữ liệu

4.1.1 Gán nhãn thủ công

Sau khi thực hiện tách từ thủ công, nhóm sẽ dùng bộ dữ liệu này để tiến hành phân công gán nhãn từ loại thủ công, dựa trên từ điển VLSP và danh sách nhãn các từ loại như hình 4.1.

Phát biểu bài toán: Đầu vào là một câu hay đoạn văn bản tiếng Việt đã tách từ, đầu ra là tập nhãn phù hợp nhất cho các từ trong câu hoặc đoạn văn bản đó.

STT	Nhãn	Tên nhãn	Ví dụ
1	N	Danh từ	đồng bằng, đường sắt, khoa học, …
2	Np	Danh từ riêng	Việt Nam, Đức, Trị An, …
3	Nc	Danh từ phân loại	con, cái, ngôi, đứa, tấm, …
4	Nu	Danh từ đơn vị	mét, cân, xu, đồng, USD
5	Ny	Danh từ viết tắt	HCV, THCS, THPT, IELTS, …
6	V	Động từ	nghiên cứu, có, tuyển, viết, đọc, bơi lội, …
7	A	Tính từ	tốt, xấu, đẹp, cao, thấp, …
8	P	Đại từ	tôi, tớ, ta, chúng tôi, chúng ta, …
9	R	Phó từ	đã, cũng, sẽ, chẳng, chưa, …
10	M	Số từ	0, 1, 2.000, một, trăm, triệu, …
11	L	Định từ	những, mỗi, từng, mấy, …
12	E	Giới từ	trên, dưới, trong, ngoài, trừ, …
13	C	Liên từ	vì vậy, tuy nhiên, ngược lại, …
14	Cc	Liên từ đẳng lập	và, hoặc, cùng, …
15	I	Thán từ	ôi, chao, vâng, dạ
16	T	Trợ từ	ngay, chính, thì, là, chỉ, …
17	X	Không xác định	
18	Z	Yếu tố cấu tạo từ	bất, vô, phi
19	CH	Nhãn cho các loại dấu	.!?,(){}/\:";[]

Bảng 4.1 Danh sách nhãn từ loại tiếng Việt

Nhóm cũng thống nhất một số quy ước gán nhãn cho các trường hợp sau:

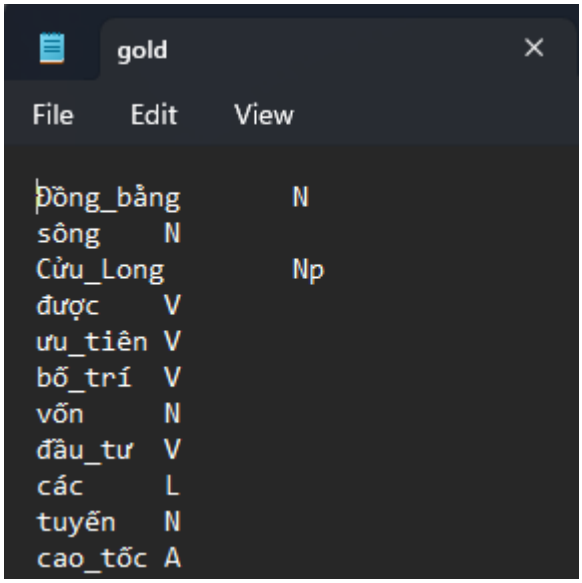
- Đối với các từ viết tắt tiếng Việt và tiếng Anh (Ví dụ: HCV, HĐND, TP HCM, IELTS, IDP, …), nhóm sử dụng nhãn Ny (Danh từ ký hiệu).

- Các từ tiếng Anh khác (không viết tắt, ví dụ: website, readiness, …), nhóm sẽ tra cứu từ điển tiếng Anh để xác định từ loại phù hợp.

- Các tên riêng tiếng Anh (không viết tắt, ví dụ: Joe Biden, Zelensky, …) nhóm sử dụng nhãn Np (Danh từ riêng).

- Các dạng ngày tháng (30/4, 1-5) hay phân số đều gán nhãn M (Số từ)

Kết quả gán nhãn sẽ được lưu vào file gold.txt.



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

Thông tin bộ dữ liệu:

- Các âm tiết của từ ghép được phân cách bởi dấu gạch dưới "_"

- Nhãn được phân tách với từ bởi dấu tab "\t"

- Mỗi dòng là 1 từ kèm nhãn của nó

- Mỗi câu được ngăn cách bởi một dòng trống

- Số lượng câu: 90

- Số lượng từ kèm nhãn: 2211

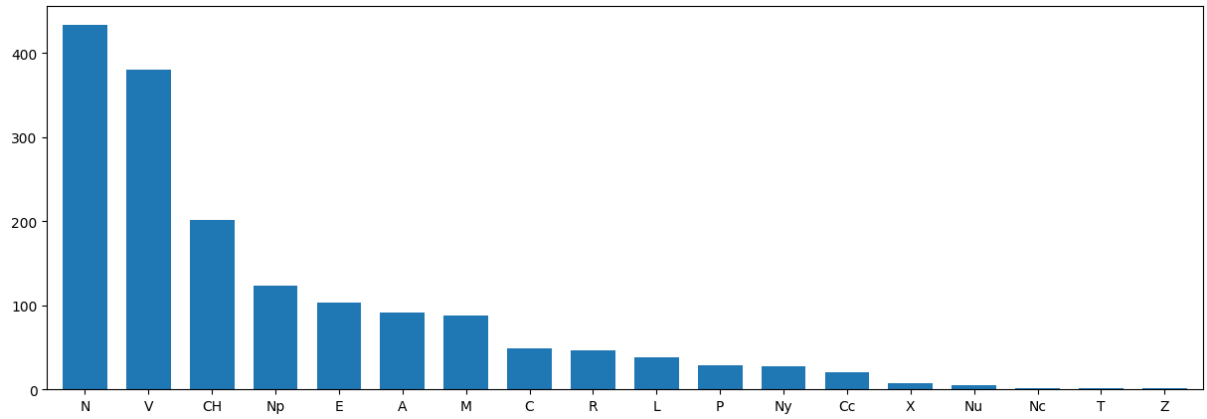
4.1.2 Phân chia tập dữ liệu train và tập dữ liệu test

Nhóm phân chia tập dữ liệu gold.txt thành hai tập dữ liệu train và test:

- Tập train: gồm 2 file train_gold.txt và train_words.txt, mỗi file chứa dữ liệu 60 câu đầu tiên, số lượng từ mỗi file: 1709

+ train_gold: chứa các từ kèm nhãn, phục vụ cho việc huấn luyện.

+ train_words: chỉ chứa các từ, dùng để kiểm thử trên tập train.

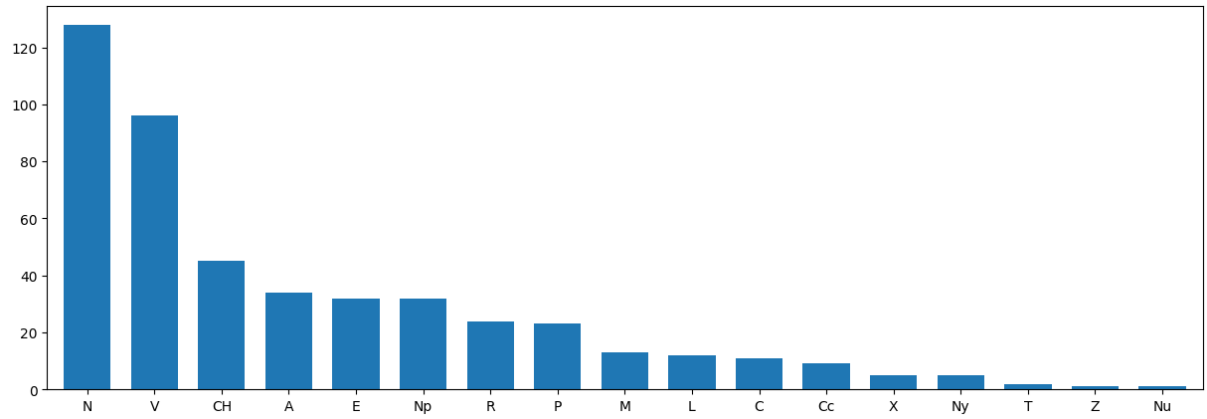


Hình 4.2 Các nhãn trong tập train

- Tập test: gồm 2 file test_gold.txt và test_words.txt, mỗi file chứa dữ liệu 30 câu còn lại, số lượng từ mỗi file là 502

+ test_gold: chứa các từ kèm nhãn, phục vụ cho đánh giá kết quả dự đoán.

+ test_word: chỉ chứa các từ, phục vụ cho việc dự đoán.



Hình 4.3 Các nhãn trong tập test

Vì trong lúc dự đoán mô hình gán nhãn có thể sẽ gặp những từ không có trong dataset của nó. Những từ này sẽ được thay thế bằng một mã không xác định.

Những từ vựng trong tập train đã được xử lý để nằm trong bộ từ vựng. Bộ từ vựng được lưu với tên vocabs.txt gồm 55008 từ, lưu trữ trong file vocabs.txt.

Những từ vựng trong tập test không thuộc bộ từ vựng sẽ được thay thế bằng mã '--unk--'. Quá trình tiền xử lý cũng sẽ xác định kết thúc của một câu, giá trị đó sẽ được đặt nhãn là '—n—'. Mã "—s—" đại diện cho nhãn bắt đầu.

Có 23 từ trong tập test không nằm trong bộ từ vựng: Covid, Mẫu, Viêm, Giáng_sinh, Tân_Sơn_Nhất, TPHCM, Bắt, Kinh_doanh, Công_ty, TNHH, Bảo_hiểm_nhân_thọ, Dai-ichi, Học_phí, PV, Thanh_Niên, Pháp, trải_nghiệm, Đài, Fox_News, Tara_Reade, Thượng_viện, Khi, Bắc.

4.1.3 Thử nghiệm gán nhãn đơn giản

Hướng tiếp cận đơn giản khi gán nhãn cho một từ là gán nhãn thường gặp nhất của từ đó trong tập train. Nhóm sẽ bắt đầu từ hướng tiếp cận này, và sau đó xây dựng các mô hình phức tạp hơn.

Các bước triển khai:

- Tạo một từ điển (dictionary) đếm số lần mỗi nhãn xuất hiện bên cạnh một nhãn khác trong tập train, với keys là các (prev_tag, tag), value là số lần xuất hiện của chúng theo thứ tự đó. Từ điển này được đặt tên là transition_counts, được sử dụng để tính xác suất $P(t_i|t_{i-1})$ là xác suất của một nhãn ở vị trí i được cho bởi nhãn ở vị trí i - 1.

```
Transition examples:
(('--s--', 'N'), 14)
(('N', 'N'), 75)
(('N', 'Np'), 42)
```

Hình 4.4 Một số giá trị trong từ điển transition_counts

- Tạo một từ điển đếm số lần của một từ được cho bởi nhãn của nó trong tập train, với keys là các (tag, word), value là số lần xuất hiện của chúng. Từ điển này được đặt tên là emission_counts, được sử dụng để tính xác suất $P(w_i|t_i)$ là xác suất một từ ở vị trí i được cho bởi nhãn của nó.

```
Emission examples:
(('N', 'Đồng_bằng'), 1)
(('N', 'sông'), 1)
(('Np', 'Cửu_Long'), 1)
```

Hình 4.5 Một số giá trị trong từ điển emission_counts

- Tạo một từ điển đếm số lần xuất hiện của một nhãn trong tập train. Từ điển này đặt tên là tag_counts, với keys là các nhãn, value là số lần xuất hiện của chúng.

```
Số nhãn: 19
['--s--', 'A', 'C', 'CH', 'Cc', 'E', 'L', 'M', 'N', 'Nc', 'Np', 'Nu', 'Ny', 'P', 'R', 'T', 'V', 'X', 'Z']
```

Hình 4.6 Các nhãn trong từ điển tag_counts

- Theo hướng tiếp cận đơn giản, nhóm sẽ sử dụng từ điển emission_counts, dự đoán nhãn của một từ dựa trên nhãn xuất hiện thường xuyên nhất với từ đã cho, sau

đó kiểm tra lại với nhãn thực. Tính độ chính xác bằng số dự đoán đúng chia cho tổng số từ mà đã dự đoán nhãn

```
accuracy = predict_pos(train_words, train_gold, emission_counts, vocabs_dict, states)
print('Độ chính xác trên tập train:', accuracy)
```

```
Độ chính xác trên tập train: 0.9607957870099474
```

```
accuracy = predict_pos(test_words, test_gold, emission_counts, vocabs_dict, states)
print('Độ chính xác trên tập test:', accuracy)
```

```
Độ chính xác trên tập test: 0.48804780876494025
```

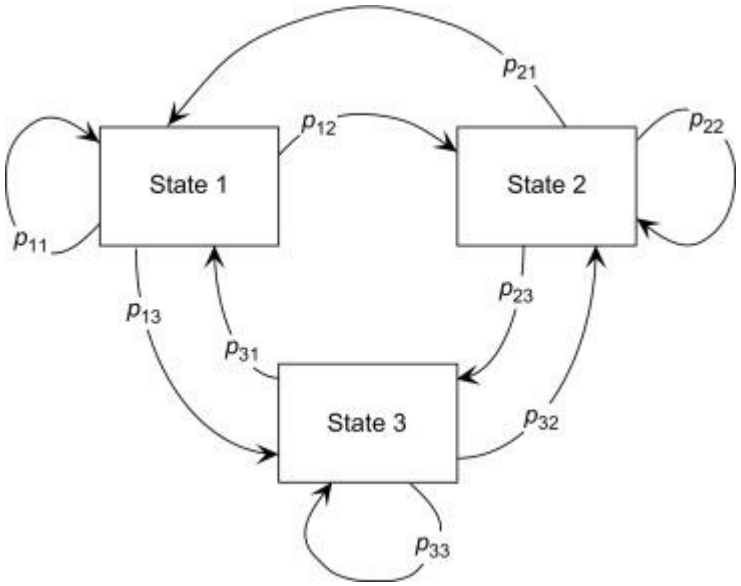
Hình 4.7 Kết quả thử nghiệm trên tập train và tập test

Qua hình 4.7 ta thấy độ chính xác trên tập test rất thấp. Vì vậy, sử dụng nhãn thường xuyên nhất của một từ đến gán cho từ đó không phải là hướng tiếp cận hiểu quả, do một từ có thể có nhiều nhãn và phụ thuộc vào ngữ cảnh của câu.

4.2 Mô hình Hidden Markov (HMM)

4.2.1 Markov Chain

Markov Chain (Xích Markov), hay Visible Markov Model là một dạng mô hình máy trạng thái hữu hạn (Finite State Automata - FSA) được dùng để mô hình hóa xác suất của các biến ngẫu nhiên có quan hệ với nhau theo dạng chuỗi.



Hình 4.8 Markov Chain

Một Markov Chain đưa ra một giả định rằng nếu dự đoán tương lai của chuỗi thì những trạng thái hiện tại là điều quan trọng nhất để dự đoán. Tương lai thì không bị ảnh hưởng bởi tất cả trạng thái trước trạng thái hiện tại mà nó thông qua trạng thái hiện tại.

Một cách tổng quát, Markov Chain được xác định bởi các thành phần sau:

- $Q = q_1, q_2, \ldots q_n$ là tập chuỗi n biến trạng thái quan sát được. Ta có giả định Markov: Xác suất chuyển sang trạng thái tiếp theo chỉ phụ thuộc vào trạng thái hiện tại chứ không phụ thuộc vào những trạng thái trước đó.

$$P(q_{i+1} \mid q_1, q_2, \ldots, q_i) = P(q_{i+1} \mid q_i)$$

- $A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}$ là một ma trận chuyển đổi trạng thái,

với mỗi phần tử $a_{ij} = P(q_n = j \mid q_{n-1} = i)$ thể hiện xác suất chuyển từ trạng thái i đến trạng thái j với ràng buộc $\sum_{j=1}^{n} a_{ij} = 1$ với mọi i và $a_{ij} \geq 0$ với mọi i, j.

- $B = \begin{pmatrix} b_{11} & b_{12} & \ldots & b_{1N} \\ b_{21} & b_{22} & \ldots & b_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ b_{n1} & b_{n2} & \ldots & b_{nN} \end{pmatrix}$ là ma trận thể hiện, với mỗi phần tử $b_{ij} = P(w_i|q_n = j)$ thể hiện xác suất quan sát w_i thuộc W là tập các quan sát có N phần tử từ một trạng thái j.

- $\pi = \pi_1, \pi_2, \ldots, \pi_n$: là một phân phối xác suất ban đầu trên mỗi trạng thái. π_i là xác suất mà Markov chain sẽ bắt đầu ở trạng thái i, một vài trạng thái j có thể có $\pi_j = 0$, có nghĩa là chúng không được khởi tạo phân phối xác suất ban đầu, nó cũng có một ràng buộc là $\sum_{i=1}^{n} \pi_i = 1$.

4.2.2 Mô hình Hidden Markov

HMM (Hidden Markov Models) là một trong những thuật toán được sử dụng phổ biến nhất trong Xử lý ngôn ngữ tự nhiên và là nền tảng cho nhiều kỹ thuật học sâu. Ngoài gán nhãn từ loại, HMM còn được dùng để nhận dạng giọng nói, tổng hợp giọng nói, …

Mô hình Markov sử dụng ma trận chuyển trạng thái A (Transition Matrix). Mô hình Markov ẩn thêm một ma trận thể hiện B (Emission Matrix) mô tả xác suất của một quan sát có thể nhìn thấy khi ta ở một trạng thái cụ thể. Trong trường hợp này, các quan sát là các từ. Trạng thái, thứ được xem là ẩn (Hidden) chính là nhãn của từ đó.

Nhóm sẽ triển khai xây dựng Transition Matrix A và Emission Matrix B, dựa vào các từ điển transition_counts, emission_counts và tag_counts đã xây dựng trước đó

- Transition Matrix A: với mỗi phần tử a_{ij} thể hiện xác suất chuyển từ nhãn i đến nhãn j với ràng buộc $\sum_{j=1}^{n} a_{ij} = 1$ (Tổng của mỗi hàng = 1) với mọi i và $a_{ij} \geq 0$ với mọi i, j. Xác suất này được tính theo công thức như hình 4.9:

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

Hình 4.9 Công thức xác suất chuyển đổi trạng thái

+ N: tổng số nhãn

+ $C(t_{i-1}, t_i)$: số lượng bộ (prev_tag, tag) trong transition_counts

+ $C(t_{i-1})$: số lượng nhãn prev_tag trong tag_counts

+ α: tham số làm mịn (smoothing), ở đây lấy α = 0.1

	A	C	CH	Cc	E
--s--	0.033386	0.065183	0.017488	0.001590	0.049285
A	0.096912	0.043663	0.192758	0.001065	0.075612
C	0.060904	0.021611	0.041257	0.001965	0.021611
CH	0.015204	0.029917	0.015204	0.005395	0.044630
Cc	0.050228	0.004566	0.004566	0.004566	0.050228

Hình 4.10 Transition Matrix A

- Emission Matrix B: với mỗi phần tử b_{ij} thể hiện xác suất một từ ở vị trí i được gán nhãn j. Xác suất này được tính theo công thức như hình 4.11:

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

Hình 4.11 Công thức tính xác suất thể hiện

+ N: số lượng từ trong từ điển

+ $C(t_i, w_i)$: số lượng cặp (tag, word) thứ i trong emission_counts

+ $C(t_i)$: số lượng nhãn thứ i trong tag_counts

+ α: tham số làm mịn (smoothing), ở đây lấy α = 0.1

	sản_xuất	kinh_tế	trí_tuệ_nhân_tạo	cường_quốc	vị_trí
N	0.000017	0.000017	0.000185	0.000017	0.000185
V	0.000187	0.000017	0.000017	0.000017	0.000017
CH	0.000018	0.000018	0.000018	0.000018	0.000018
Cc	0.000018	0.000018	0.000018	0.000018	0.000018
A	0.000018	0.000018	0.000018	0.000018	0.000018

Hình 4.12 Emission Matrix B

Mục đích của tham số làm mịn là để tránh số đếm được là 0 và tăng hiệu năng của mô hình với các dữ liệu không xuất hiện trong mẫu.

4.3 Thuật toán Viterbi

Thuật toán Viterbi là phương pháp để ước lượng xác suất chuỗi trạng thái cực đại xác suất của mô hình đưa ra dãy các quan sát.

Nhóm sẽ tiến hành sử dụng thuật toán Viterbi kết hợp với hai ma trận A và B của mô hình Hidden Markov để tiến hành xác định chuỗi trạng thái ẩn, sử dụng chiến lược quy hoạch động. Quy trình được chia thành 3 bước: Khởi tạo, Forward, Backward.

4.3.1 Khởi tạo

Khởi tạo hai ma trận cùng chiều:

+ best_probs: mỗi phần tử chứa xác suất đi từ một nhãn sang một từ.

+ best_paths: ma trận giúp tìm đường đi tốt nhất.

Cả hai ma trận đều được khởi tạo bằng 0, trừ cột đầu tiên của best_probs sẽ được khởi tạo với giả định rằng từ đầu tiên của ngữ liệu được đặt trước bởi một ký tự bắt đầu (‘--s--’):

if A[s_idx, i - 1] != 0:

best_probs[i, 0] = ln(A[s_idx, i]) + ln(B[i, index]) (*)

else:

best_probs[i, 0] = float(“-inf”)

Ở đây s_idx là chỉ số của nhãn ‘--s--’ trong tập trạng thái (s_idx = 0 theo hình 4.6), index là chỉ số của từ đầu tiên trong corpus xuất hiện trong từ điển. Corpus được sử dụng là tập train_words và test_words.

Phép tính ở (*) thực chất là phép nhân xác suất:

*best_probs[i, 0] = A[s_idx, i] * B[i, index]*

Các giá trị xác suất trong 2 ma trận A và B có giá trị rất nhỏ, khi nhân lại sẽ càng nhỏ. Để tránh điều này, ta dùng logarit tự nhiên đưa chúng thành tổng 2 log.

```
best_probs_train[0, 0]: -17.359546940100284
best_paths_train[0, 0]: 0
```

```
best_probs_test[0, 0]: -17.359546940100284
best_paths_test[0, 0]: 0
```

Hình 4.13 Kết quả khởi tạo trên tập train và tập test

4.3.2 Forward

Tiến hành điền vào hai ma trận best_probs và best_paths đã khởi tạo.

Triển khai:

+ Duyệt tất cả từ trong corpus (dùng biến i), trừ từ đầu tiên.

+ Với mỗi từ, duyệt tất cả nhãn có thể của từ đó (dùng biến j)

+ Duyệt tất cả nhãn có thể của từ đứng trước nó (dùng biến k)

+ Với mỗi k, tính xác suất để từ hiện tại có nhãn j và từ trước đó có nhãn k:

prob = best_probs[k, i - 1] + log(A[k, j]) + log(B[j, index])

- best_probs[k, i -1]: Xác suất lớn nhất để từ trước nó có nhãn k

- A[k, j]: Xác suất để nhãn j xuất hiện sau nhãn k

- B[j, index]: Xác suất từ đang xét được gán nhãn j

+ Cập nhật best_probs[j, i] và best_paths[j, i]

+ Trả về hai ma trận best_probs và best_paths

Mã giả cho hàm viterbi_forward:

(Lưu ý: Do ma trận A có kích thước 19 x 18 theo hình 4.10, j trong công thức được thay là j - 1)

```
def viterbi_forward(A, B, corpus, best_probs, best_paths, vocabs_dict):
    num_tags = best_probs.shape[0] #Hoặc = len(tag_counts)
    for i in range(1, len(corpus)):
        for j in range(num_tags):
```

```
        best_prob_i = float('-inf') #Âm vô cùng
        best_path_i = Null
        for k in range(num_tags):
            index = vocabs[corpus[i]] #Ví trị từ thứ i của corpus trong vocabs
            prob = best_probs[k, i - 1] + log(A[k, j - 1]) + 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
```

```
best_probs_train[0, 1]: -29.22273512484214
best_paths_train[0, 4]: 16
```

```
best_probs_test[0, 1]: -26.72965883454188
best_paths_test[0, 4]: 6
```

Hình 4.14 Kết quả bước forward

4.3.3 Backward

Sử dụng best_probs và best_paths trả về danh sách các nhãn được dự đoán cho mỗi từ trong corpus.

Triển khai:

+ Duyệt tất cả nhãn của từ cuối cùng trong best_probs, tìm nhãn có giá trị lớn nhất.

+ Dùng best_paths từ vị trí từ cuối cùng, tìm nhãn có giá trị cao nhất cho từ trước nó.

+ Xuất ra mảng kết quả dự đoán nhãn.

Mã giả cho hàm viterbi_backward:

```
def viterbi_backward(best_probs, best_paths, corpus, states):
    m = best_paths.shape[1] #Hoặc = len(corpus)
    z, pred = [None] * m
    best_prob_for_last_word = float('-inf')
    num_tags = best_probs.shape[0] #Hoặc = len(tag_counts)
```

```
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
```

```
Dự đoán cho test_pred[-7:501]:
['nào', 'miền', '--unk--', 'chấm_dứt', 'nắng_nóng', '?']
['N', 'V', 'N', 'V', 'N', 'V']
Dự đoán cho test_pred[0:7]:
[',', 'các', 'hãng', 'smartphone', 'sẽ', 'cho', 'người']
['V', 'L', 'N', 'Np', 'R', 'E', 'N']
```

Hình 4.15 Một số kết quả của bước backward trên tập test

4.4 Đánh giá kết quả và so sánh

4.4.1 Kết quả của mô hình Hidden Markov kết hợp Viterbi

```
Nữ/V công_nhân/N gom/V những/L xe/N rác/V cao/A quá/V đầu/N người/N ./CH
Những/V ngày/N đẹp_đẽ/V ấy/N ,/CH bố/N luôn/V dành/V thời_gian/N để/E vui_vẻ/N với/C anh_em/V tôi/N ./CH
Trong/E khi/N chờ/V nước/N nóng/Np ,/CH tôi/N pha/V rượu/N táo/V nóng/N ./CH
Hàng/N trăm/M người/N đến/V dùng/V tiệc/N tự/V chọn/N ở/E chỗ/N mẹ/V những/L ngày/N --unk--/Np ./CH
Năm/Np nào/Np cũng/R vậy/V hoặc/Cc có_vẻ/N như_vậy/Np ./CH
Tôi/V chưa/N bao_giờ/V thực_sự/N cảm_thấy/V mệt_mỏi/N vì/V nó/N ./CH
Tôi/Np đã/R có_thể/R chống/V lại/V cảm_giác/N đó/P trong/E nhiều/N năm/N qua/Np ./CH
Những/V năm/N gần/A đây/P tôi/R qua_lại/V sân_bay/N --unk--/V thường_xuyên/N ./CH
Trường/N tôi/N từng/P nghiêm_cấm/R sử_dụng/V điện_thoại/N trong/E lớp/N ./CH
Học_sinh/N ngày_nay/Np có_thể/R dễ_dàng/V tiếp_cận/N bài_học/Np và/Cc phương_pháp/N giải/V bài_tập/N ./CH
Điện_thoại/V thông_minh/N là/V công_cụ/N không_thể/R đảo_ngược/V trong/E cuộc_sống/N ./CH
--unk--/V tổ_chức/V đấu_giá/N lô/V xe/N vi_phạm/V với/C giá/N khởi_điểm/V gần/A 4/M tỷ/M đồng/Nu ./CH
--unk--/V cựu/V Trưởng/N phòng/V --unk--/N --unk--/V --unk--/N --unk--/V --unk--/N Việt_Nam/Np ./CH
--unk--/V trường/N quốc_tế/N Australia/V tại/E TP/Ny HCM/Ny bao_nhiêu/E ?/N
```

Hình 4.16 Kết quả gán nhãn của HMM kết hợp Viterbi trên 1 số câu trong tập test

Các từ không thuộc bộ từ vựng sẽ có giá trị '--unk--', kết quả dự đoán trên các từ này không có độ chính xác cao.

Kết quả của mô hình Hidden Markov kết hợp thuật toán Viterbi trên tập train:

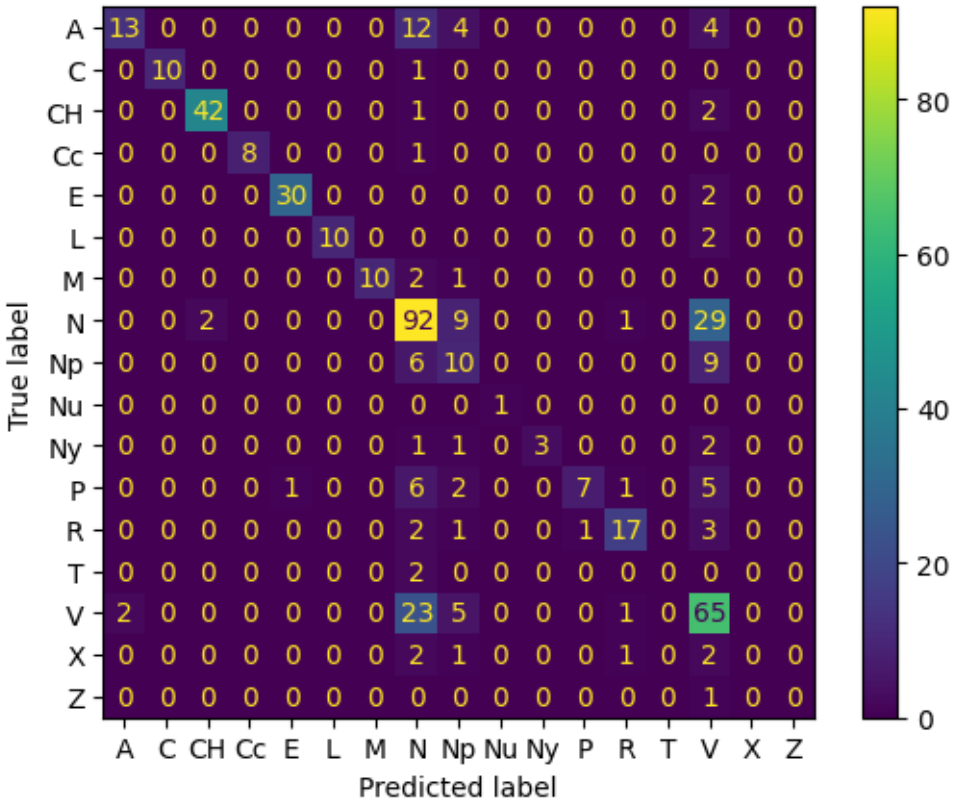
	precision	recall	f1-score	support
A	1.00	0.96	0.98	92
C	1.00	0.92	0.96	49
CH	1.00	1.00	1.00	202
Cc	1.00	1.00	1.00	20
E	1.00	0.99	1.00	103
L	1.00	0.97	0.99	38
M	1.00	0.96	0.98	89
N	0.95	0.99	0.97	447
Nc	0.00	0.00	0.00	1
Np	0.96	0.96	0.96	107
Nu	0.57	0.80	0.67	5
Ny	1.00	0.90	0.95	30
P	1.00	0.86	0.93	29
R	0.98	0.96	0.97	47
T	0.00	0.00	0.00	1
V	0.97	0.99	0.98	380
X	1.00	0.14	0.25	7
Z	1.00	1.00	1.00	1
accuracy			0.98	1648
macro avg	0.86	0.80	0.81	1648
weighted avg	0.98	0.98	0.97	1648

Hình 4.17 Kết quả của mô hình trên tập train

Kết quả của mô hình Hidden Markov kết hợp thuật toán Viterbi trên tập test:

	precision	recall	f1-score	support
A	0.87	0.39	0.54	33
C	1.00	0.91	0.95	11
CH	0.95	0.93	0.94	45
Cc	1.00	0.89	0.94	9
E	0.97	0.94	0.95	32
L	1.00	0.83	0.91	12
M	1.00	0.77	0.87	13
N	0.61	0.69	0.65	133
Np	0.29	0.40	0.34	25
Nu	1.00	1.00	1.00	1
Ny	1.00	0.43	0.60	7
P	0.88	0.32	0.47	22
R	0.81	0.71	0.76	24
T	0.00	0.00	0.00	2
V	0.52	0.68	0.59	96
X	0.00	0.00	0.00	6
Z	0.00	0.00	0.00	1
accuracy			0.67	472
macro avg	0.70	0.58	0.62	472
weighted avg	0.70	0.67	0.67	472

Hình 4.18 Kết quả của mô hình trên tập test



Hình 4.19 Kết quả Confusion Matrix

So với hướng tiếp cận đơn giản ban đầu, độ chính xác của mô hình Hidden Markov kết hợp thuật toán Viterbi đã cải thiện hơn.

Ta thấy kết quả của mô hình này trên tập train rất tốt, nhưng trên tập test lại không tốt, do đó có thể kết luận mô hình đã bị overfitting.

Ta có thể cải thiện mô hình bằng cách tăng cường dữ liệu train, hoặc sử dụng bộ dữ liệu từ vựng đầy đủ hơn, khi đó sẽ hạn chế xuất hiện các từ vựng mới trong tập test.

4.4.2 Thử nghiệm tách từ tiếng Việt trên một số thư viện

	Accuracy	Precision	Recall	F1-score
HMM + Viterbi	0.67	0.7	0.58	0.62
VnCoreNLP	0.96	0.91	0.87	0.88
pyvi	0.78	0.60	0.52	0.49

Bảng 4.2 Kết quả gán nhãn trên tập test của mô hình và các thư viện khác

Ta thấy thư viện VnCoreNLP thực hiện tốt hơn so với mô hình HMM đã xây dựng. Đối với thư viện pyvi, bộ nhãn quy ước của thư viện này có một ít khác biệt với bảng 4.1, vì vậy dù accuracy cao hơn, nhưng precision và recall của pyvi lại thấp hơn so với HMM.

CHƯƠNG V: KẾT LUẬN

Trong đề tài này, nhóm đã trình bày về hai bài toán tách từ tiếng Việt và gán nhãn từ loại tiếng Việt.

Đối với bài toán tách từ tiếng Việt, đây là bước tiền xử lý quan trọng cho bài toán gán nhãn từ loại. Nhóm đã tìm hiểu hướng tiếp cận theo từ điển bằng thuật toán Longest Matching, triển khai trên bộ dữ liệu đã thu thập và so sánh kết quả với hai thư viện là VnCoreNLP và pyvi. Từ đó nhóm đưa ra kết luận: thuật toán Longest Matching tuy dễ cài đặt và có thời gian thực thi nhanh, nhưng không thể tách các từ không nằm trong bộ từ điển (đặc biệt là với các tên riêng), đồng thời không giải quyết được vấn đề nhập nhằng.

Đối với bài toán gán nhãn từ loại tiếng Việt, nhóm đã đi từ hướng tiếp cận đơn giản bằng cách gán nhãn theo số lượng nhãn thường xuyên của mỗi từ. Sau đó nhóm tiến hành tìm hiểu về mô hình Hidden Markov cũng như thuật toán Viterbi, từ đó xây dựng một mô hình cải thiện hơn, tuy nhiên mô hình vẫn còn phụ thuộc nhiều vào bộ dữ liệu từ vựng và tập dữ liệu train. Vì vậy nhóm đã kết luận mô hình bị overfitting và đưa ra một số hướng cải thiện bằng cách tăng cường dữ liệu train và hoàn thiện đầy đủ hơn bộ từ vựng.

Cuối cùng, nhóm tiến hành thử nghiệm gán nhãn từ loại với hai thư viện VnCoreNLP và pyvi để so sánh với mô hình HMM. Nhóm kết luận rằng hai thư viện trên cho kết quả gán nhãn từ loại tốt hơn.

Sau khi tìm hiểu đề tài, nhóm đã nắm được kiến thức về bài toán tách từ và gán nhãn từ loại, hiểu được cách hoạt động của mô hình Hidden Markov và thuật toán Viterbi. Từ nền tảng trên, nhóm có thể tiếp tục tìm hiểu các hướng tiếp cận khác cho bài toán này, cũng như tiếp cận các bài toán nâng cao hơn.

TÀI LIỆU THAM KHẢO

[1] W.Li, C.Zhang, "Markov Chain Analysis" in International Encyclopedia of Human Geography, ScienceDirect, https://www.sciencedirect.com/topics/earth-and-planetary-sciences/markov-chain , 2009, pp. 455-460.

[2] "Speech and Language Processing." https://web.stanford.edu/~jurafsky/slp3/ (accessed June. 1, 2023).

[3] "VnCoreNLP: A Vietnamese natural language processing toolkit" GitHub. https://github.com/vncorenlp/VnCoreNLP (accessed June. 1, 2023).

[4] "Python Vietnamese Toolkit" PyPI. https://pypi.org/project/pyvi/ (accessed June. 1, 2023).

[5] "Gán nhãn từ loại Tiếng Việt" GitHub. https://github.com/ds4v/vietnamese-pos-tagging (accessed June. 1, 2023).

[6] "Hidden Markov Model" Github. https://mmz33.github.io/Hidden-Markov-Model/ (accessed June. 1, 2023).