**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**A picture containing logo, graphics, clipart, symbol

Description automatically generated**

**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**

LỜI CẢM ƠN……………………………………………………………………2

DANH MỤC HÌNH ẢNH……………………………………………………….4

DANH MỤC BẢNG..............................................................................................5

CHƯƠNG I: GIỚI THIỆU....................................................................................6

CHƯƠNG II: THU THẬP DỮ LIỆU....................................................................7

CHƯƠNG III: TÁCH TỪ......................................................................................8

CHƯƠNG IV: GÁN NHÃN TỪ LOẠI................................................................14

CHƯƠNG V: KẾT LUẬN...................................................................................28

TÀI LIỆU THAM KHẢO....................................................................................29

**DANH MỤC HÌNH ẢNH**

Hình 2.1 Một số câu trong bộ dữ liệu gốc……………………………………………...7

Hình 3.1 Kết quả tách từ của thuật toán Longest Matching…………………………..11

Hình 3.2 Kết quả tách từ của thư viện VnCoreNLP…………………………………..11

Hình 3.3 Kết quả tách từ của thư viện pyvi…………………………………………...12

Hình 3.4 Kết quả tách từ thủ công…………………………………………………….12

Hình 4.1 Một số từ kèm nhãn trong bộ dữ liệu gold.txt………………………………15

Hình 4.2 Các nhãn trong tập train…………………………………………………….16

Hình 4.3 Các nhãn trong tập test……………………………………………………...16

Hình 4.4 Một số giá trị trong từ điển transition\_counts................................................17

Hình 4.5 Một số giá trị trong từ điển emission\_counts.................................................17

Hình 4.6 Các nhãn trong từ điển tag\_counts.................................................................17

Hình 4.7 Kết quả thử nghiệm trên tập train và tập test..................................................18

Hình 4.8 Markov Chain.................................................................................................18

Hình 4.9 Công thức xác suất chuyển đổi trạng thái.......................................................20

Hình 4.10 Transition Matrix A......................................................................................20

Hình 4.11 Công thức tính xác suất thể hiện...................................................................20

Hình 4.12 Emission Matrix B........................................................................................21

Hình 4.13 Kết quả khởi tạo trên tập train và tập test......................................................22

Hình 4.14 Kết quả bước forward....................................................................................23

Hình 4.15 Một số kết quả của bước backward trên tập test...........................................24

Hình 4.16 Kết quả gán nhãn của HMM kết hợp Viterbi trên 1 số câu trong tập test.....24

Hình 4.17 Kết quả của mô hình trên tập train.................................................................25

Hình 4.18 Kết quả của mô hình trên tập test..................................................................26

Hình 4.19 Kết quả Confusion Matrix.............................................................................26

**DANH MỤC BẢNG**

Bảng 3.1 Kết quả đánh giá các phương pháp tách từ so với phương pháp thủ công…12

Bảng 4.1 Danh sách nhãn từ loại tiếng Việt…………………………………………..14

Bảng 4.2 Kết quả gán nhãn trên tập test của mô hình và các thư viện khác.................27

**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.

A screen shot of a computer

Description automatically generated with medium confidence

*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 ≠ ∅*

*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

A screenshot of a computer program

Description automatically generated with medium confidence*Hình 3.1 Kết quả tách từ của thuật toán Longest Matching*

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

A screenshot of a computer

Description automatically generated*Hình 3.2 Kết quả tách từ của thư viện VnCoreNLP*

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

A screenshot of a computer

Description automatically generated*Hình 3.3 Kết quả tách từ của thư viện pyvi*

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

A screenshot of a computer program

Description automatically generated with medium confidence*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.

A screenshot of a computer

Description automatically generated

*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.

A picture containing screenshot, plot, line, rectangle

Description automatically generated

*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.

A picture containing text, screenshot, plot, line

Description automatically generated

*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(ti|ti-1) là xác suất của một nhãn ở vị trí i được cho bởi nhãn ở vị trí i - 1.

A picture containing text, font, screenshot, typography

Description automatically generated

*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(wi|ti) là xác suất một từ ở vị trí i được cho bởi nhãn của nó.

A picture containing text, font, screenshot, typography

Description automatically generated

*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.



*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

A screenshot of a computer program

Description automatically generated with low confidence

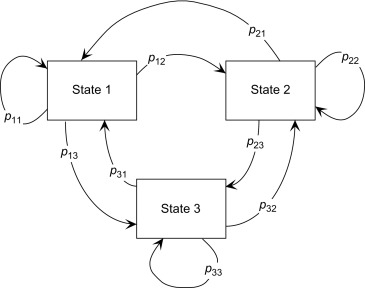
*Hình 4.7 Kết quả thử nghiệm trên tập train và tập test*

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 = q1, q2, … qn 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(qi+1 | q1, q2, …, qi) = P(qi+1 | qi)

- A = là một ma trận chuyển đổi trạng thái,

với mỗi phần tử aij = P(qn = j | qn-1 = i ) thể hiện xác suất chuyển từ trạng thái i đến trạng thái j với ràng buộc = 1 với mọi i và aij ≥ 0 với mọi i, j.

- B = là ma trận thể hiện, với mỗi phần tử bij = P(wi|qn = j) thể hiện xác suất quan sát wi thuộc W là tập các quan sát có N phần tử từ một trạng thái j.

- 𝜋 = 𝜋1, 𝜋2, … , 𝜋n: là một phân phối xác suất ban đầu trên mỗi trạng thái. 𝜋𝑖 là xác suất mà Markov chain sẽ bắt đầu ở trạng thái i, một vài trạng thái j có thể có 𝜋𝑗 = 0, có nghĩa là chúng không được khởi tạo phân phối xác suất ban đầu, nó cũng có một ràng buộc là = 1.

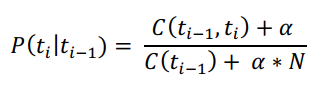
**4.2.2 Mô hình Hidden Markov**

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 ần 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 aij thể hiện xác suất chuyển từ nhãn i đến nhãn j với ràng buộc = 1 (Tổng của mỗi hàng = 1) với mọi i và aij ≥ 0 với mọi i, j. Xác suất này được tính theo công thức như hình 4.9:



*Hình 4.9 Công thức xác suất chuyển đổi trạng thái*

+ N: tổng số nhãn

+ C(ti-1, ti): số lượng bộ (prev\_tag, tag) trong transition\_counts

+ C(ti-1): số lượng nhãn prev\_tag trong tag\_counts

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

A screenshot of a graph

Description automatically generated with low confidence

*Hình 4.10 Transition Matrix A*

- Emission Matrix B: với mỗi phần tử bij 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:

A picture containing font, text, line, typography

Description automatically generated

*Hình 4.11 Công thức tính xác suất thể hiện*

+ N: số lượng từ trong từ điển

+ C(ti, wi): số lượng cặp (tag, word) thứ i trong emission\_counts

+ C(ti): số lượng nhãn thứ i trong tag\_counts

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

A screenshot of a computer

Description automatically generated with low confidence

*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.

A picture containing text, font, screenshot, typography

Description automatically generated

A picture containing text, font, screenshot, typography

Description automatically generated

*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*

A picture containing text, font, white, typography

Description automatically generated

A picture containing text, font, typography

Description automatically generated

*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*

A picture containing text, font, screenshot, black

Description automatically generated

*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**

A black screen with white text

Description automatically generated with low confidence

*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.

A screenshot of a computer screen

Description automatically generated with medium confidence

*Hình 4.17 Kết quả của mô hình trên tập train*

A screenshot of a computer screen

Description automatically generated with medium confidence

*Hình 4.18 Kết quả của mô hình trên tập test*

A picture containing screenshot, text, colorfulness, rectangle

Description automatically generated

*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).