**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

🙤🙧🟍🙥🙦



**GIÁN NHÃN TỪ LOẠI TIẾNG VIỆT SỬ DỤNG MÔ HÌNH HIDDEN MARKOV KẾT HỢP THUẬT TOÁN VITERBI**

**Giảng viên hướng dẫn: Th.S Nguyễn Trọng Chỉnh**

**Danh sách thành viên:**

**19522351 – Trần Trung Tín**

**19522486 – Trương Văn Tuấn**

**TP. HỒ CHÍ MINH – 12/2022**

1. **Giới thiệu**

Part of speech (POS) tagging là một trong những phương pháp quan trọng của xử lý ngôn ngữ tự nhiên, cũng như trong việc hiểu nội dung câu hoặc văn bản. POS là thuật ngữ truyền thống để chỉ các loại từ được phân biệt về mặt ngữ pháp trong một ngôn ngữ. Trong quá trình phát triển chúng ta quen với việc xác định từ loại trong văn bản. Đọc một câu chúng ta có thể xác định rõ từ loại như là danh từ, động từ hoặc tính từ….

Để xác định từ rõ từ loại trong câu thường phức tạp hơn nhiều trong việc ánh xạ các từ qua từ điển. Đó là bởi vì một từ có thể được gán rất nhiều từ loại dựa vào ngữ cảnh của văn bản. Đây gọi là sự nhập nhằng, thật khó để ta xác định một từ đó thuộc từ loại nào dựa vào một ngữ liệu nhất định vì tất cả ngữ cảnh mới và từ mới mỗi ngày liên tục xuất hiện đó cũng là vấn đề cho việc gán từ loại thủ công.

Phân biệt các bộ phận của từ trong câu sẽ giúp ta hiểu rõ hơn về ý nghĩa của câu. Điều này cực kỳ quan trọng trong các truy vấn tìm kiếm. Việc này xác định danh từ riêng, tổ chức, ký hiệu cổ phiếu hoặc bất kỳ thứ gì tương tự sẽ cải thiện đáng kể mọi thứ, từ nhận dạng giọng nói đến tìm kiếm.

Trong đề tài này nhóm sẽ sử dụng mô hình Hidden Markov kết hợp thuật toán Viterbi để gán nhãn từ loại tiếng Việt và so sánh độ chính xác với thư viện VNCoreNLP.

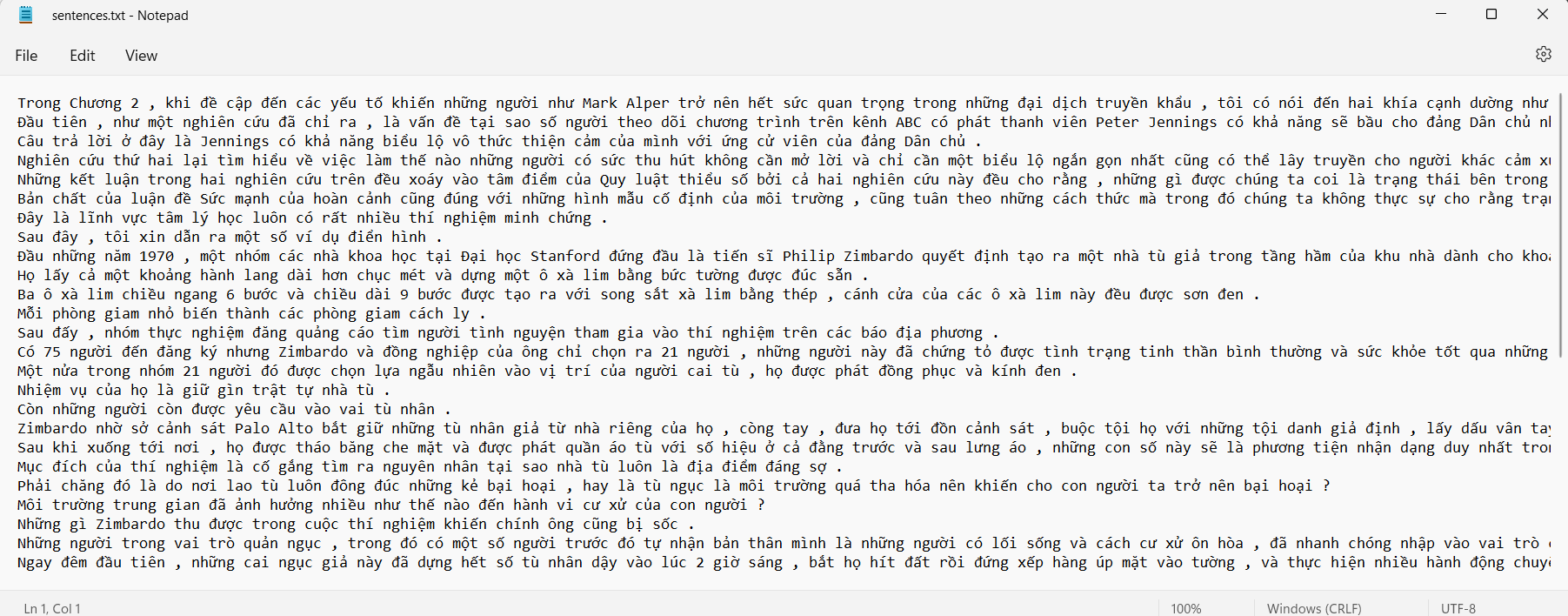
1. **Thu thập dữ liệu**
2. **Nguồn thu nhập**

Gồm các câu bất kỳ thuộc nhiều chủ đề được thu thập trên các trang mạng.

1. **Thông tin cơ bản**

Bộ dữ liệu gốc được lưu với tên sentences.txt

* Mỗi dòng là 1 câu.
* Các từ được phân cách với nhau bằng dấu các ‘ ’.
* Cuối câu là 1 dấu chấm ‘.’.
* Số lượng câu: 45.
* Số từ nhiều nhất trong một câu: 117.
* Số từ ít nhất trong một câu: 4.



*Một số câu có trong bộ dữ liệu gốc*

1. **Tách từ**

Bài toán tách từ là 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. Ví dụ từ “du\_ngoạn” được tạo lên bởi 2 âm là “du” và “ngoạn”. Trong khi hai từ đơn “du” và từ đơn “ngoạn” lại có thể mang ý nghĩa khác. Do vậy, tách từ tiếng Việt là bước quan trọng chúng ta cần thực hiện trước khi đưa dữ liệu vào các bước tiếp theo, ví dụ như word embedding.

1. **Thuật toán Maximum Matching**
2. **Giới thiệu**

Maximum matching là một thuật toán đơn giản, hoạt động dựa trên việc so các từ trong câu với một thư viện có sẵn để kiểm tra xem các từ đó và từ tiếp theo là từ đơn hay tạo thành từ ghép và tách chúng ra thành các thành phần riêng. Để sử dụng việc gán nhãn sau này.

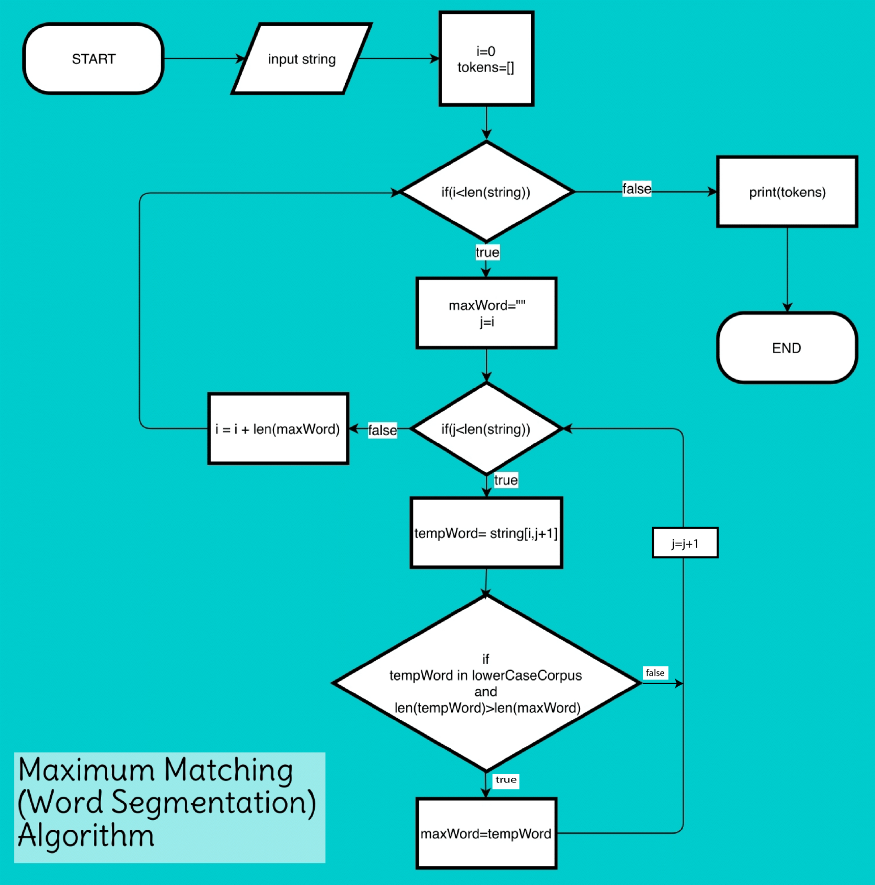
Trong bài này nhóm thực hiện matching từ trái sang phải để phù hợp với văn phạm tiếng Việt là đọc từ trái sang phải.



*Ví dụ về tách từ*

1. **Các bước thực hiện**

* Bước 1: Bắt đầu với ký tự đầu tiên của chuỗi đã cho.
* Bước 2: Tìm kiếm từ dài nhất trong danh sách bắt đầu bằng ký tự này.
* Bước 3: **Nếu** tìm thấy khớp, ranh giới được đánh dấu. **Nếu khác**, ký tự được coi là từ.



*Sơ đồ khối*

**Python Code:**

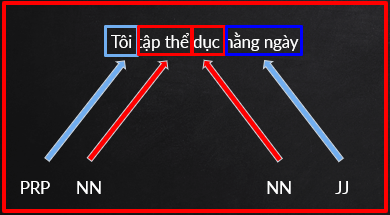
**Ảnh có chứa văn bản

Mô tả được tạo tự động**

1. **Ưu và nhược điểm**

Ưu điểm thuật toán này chính là cài đặt đơn giản, độ phức tạp tính toán hợp lý, không cầu dữ liệu huấn luyện.

Tuy nhiên, vẫn dẫn đến lỗi với các từ đã biết khi văn bản có thể phân đoạn chính xác ưu tiên nhiều từ hơn thay vì ít từ hơn. Cách tách từ trái sang phải đôi lúc cũng có những lỗi về mặt ngữ nghĩa.



*Tách từ từ trái sang phải*

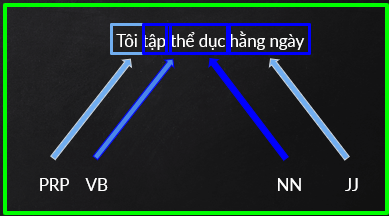
Phương pháp này phụ thuộc vào từ điển mà chúng ta dựng sẵn thế nên sẽ có những trường hợp từ ghép ngoài từ điển sẽ không tách được. Lỗi tách sai từ ghép nhập nhằng.



*Ví dụ về những trường hợp không tách được*

1. **Cách tối ưu**

Cách đơn giản nhất để giải quyết vấn đề này là chúng ta tiến hành chạy thêm tách từ phải sang trái để so sánh kết quả, giúp tăng độ chính xác.



*Tách từ từ phải sang trái*

Bổ sung thêm từ loại vào từ điển.

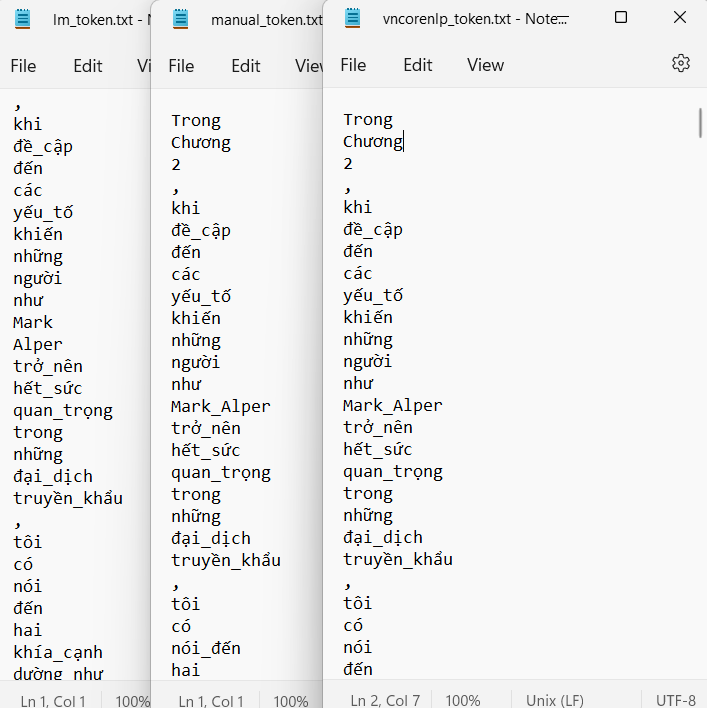
1. **Cách triển khai**

Thực hiện tách từ bán thủ công: sử dụng thư viện VnCoreNLP để tách từ trước sau đó sẽ phân công các thành viên kiểm tra thủ công lại kết quả tách từ và sửa lại các từ bị sai để thu được một bộ dữ liệu mới chặt chẽ hơn.

Bộ dữ liệu mới được lưu với tên manual\_tokens.txt sẽ chứa kết quả tách từ đúng nhất cho 45 câu nhóm đã thu thập:

* Các âm tiết của từ ghép sẽ được nối bằng dấu gạch dưới ‘\_’.
* Mỗi dòng là 1 từ.
* Mỗi câu sẽ được ngăn cách bằng 1 dòng trống.
* Số lượng câu: 60.
* Số lượng từ:
* Số lượng từ ghép:

Sau đó, nhóm sẽ sử dụng bộ dữ liệu này làm chuẩn và đánh giá kết quả tách từ của 2 phương pháp: sử dụng thuật toán Maximum Matching và sử dụng thư viện VnCoreNLP, với kết quả tách từ của mỗi phương pháp lần lượt được lưu trong các file có tên là lm\_token.txt chứa …. từ với … từ ghép và vncorenlp\_token.txt chứa … từ với … từ ghép.



*Kết quả tách từ theo từng phương pháp khác nhau*

1. **Đánh giá kết quả**

|  |  |  |
| --- | --- | --- |
|  | Longest Matching | VnCoreNLP |
| Accuracy | 0.88 | 0.890233 |
| Precision | 0.917563 | 0.913495 |
| Recall | 0.711111 | 0.733333 |
| True Positive | 256 | 264 |
| False Positive | 23 | 25 |
| Total True | 946 | 957 |
| Total Errors | 220 | 193 |

*Tổng hợp kết quả tách từ*

Qua bảng trên, có thể thấy rằng các số liệu đánh giá của việc tách từ khi sử dụng thư viện VnCoreNLP so với khi dùng thuật toán Longest Matching chênh lệch nhau không đáng kể. Độ phủ (Recall) của thuật toán này thấp hơn so với Accuracy và Precision chứng tỏ Longest Matching tách từ chưa được tốt.

Vì quá trình tách từ bằng thuật toán và thư viện đều có thể tồn tại những sai sót nên nhóm quyết định sẽ sử dụng bộ dữ liệu có được sau khi thực hiện tách từ bán thủ công (ở đây là file manual\_tokens.txt) để bắt đầu tiến hành gán nhãn từ loại.

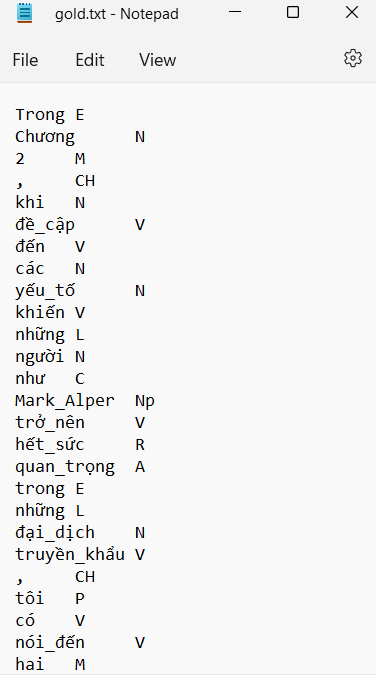
1. **Tạo ngữ liệu và gán nhãn**
2. **Tạo ngữ liệu**

Sau khi có kết quả tách từ, sẽ tiếp tục tiến hành gán nhãn bán thủ công trên bộ dữ liệu tách từ đã chọn.

Sử dụng thư viện VnCoreNLP để gán nhãn trước, sau đó sẽ phân công các thành viên kiểm tra thủ công lại kết quả gán nhãn và sửa lại các nhãn bị sai để thu được tập dữ liệu Gold cho 45 câu đã thu nhập.

Bộ dữ liệu Gold được lưu với tên gold.txt:

* Các âm tiết của từ ghép sẽ được nối bằng dấu gạch dưới ‘\_’.
* Từ được phân tách với nhãn bằng dấu tab.
* Mỗi dòng là một từ cùng với nhãn của nó.
* Mỗi câu sẽ được ngăn cách bằng một dòng trống.
* Số lượng câu: 45.
* Số lượng từ:
* Số lượng nhãn:



*Một số từ kèm nhãn trong bộ dữ liệu GOLD*

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Nhãn** | **Tên** | **Ví dụ** |
| 1 | N | Danh từ | Tiếng, nước, thủ đô, nhân dân, đồ đạc |
| 2 | Np | Danh từ riêng | Nguyễn Du, Việt Nam, Hải Phòng, Trường Đại học Bách khoa, Mộc tinh, Thủy tinh, Phật, Đạo Phật |
| 3 | Nc | Danh từ chỉ loại | con, cái, đứa, bức |
| 4 | Nu | Danh từ đơn vị | mét, cân, giờ, nắm, nhúm, hào, xu |
| 5 | Ni | Danh từ ký hiệu | A1, A4, 60A, 70B, 20c, ABCD |
| 6 | V | Động từ | ngủ, ngồi, cười, đọc, viết, đá, đặt, thích |
| 7 | A | Tính từ | tốt, xấu, đẹp, cao, thấp, rộng |
| 8 | P | Đại từ | tôi, chúng tôi, hắn, nó, y, đại nhân |
| 9 | L | Định từ | mỗi, từng, mọi, cái, các, những, mấy |
| 10 | M | Số từ | một, mười, mười ba, dăm, vài, mươi |
| 11 | R | Phó từ | đã, sẽ, đang, vừa, mới, từng, xong, rồi |
| 12 | E | Giới từ | trên, dưới, trong, ngoài, của, trừ |
| 13 | C | Liên từ | vì vậy, tuy nhiên, ngược lại |
| 14 | Cc | Liên từ đẳng lập | và, hoặc, với, cùng |
| 15 | I | Thán từ | ôi, chao, a ha |
| 16 | T | Trợ từ | à, a, á, ạ, ấy, chắc, chăng, cho, chứ |
| 17 | B | Từ vay mượn | internet, email, video, chat |
| 18 | Y | Từ viết tắt | OPEC, WTO, HIV |
| 19 | X | Các từ không thể phân loại |  |
| 20 | Z | Yếu tố cấu tạo từ | bất, vô, phi |
| 21 | CH | Nhãn dành cho các loại dấu | . ! ? , : : ‘’ “” |

*Danh sách nhãn từ loại*

Tiếp theo nhóm sẽ chia 45 câu đã được tách từ và gán nhãn trong bộ dữ liệu Gold này thành 2 tập dữ liệu: ….câu cho tập Train và …câu cho 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. Ở đây những từ vựng trong tập Train đã được xử lý để nằm trong bộ từ vựng. Những từ vựng trong tập Test không thuộc bộ từ vựng sẽ được thay thế bằng “…..”. Bộ từ vựng được lưu với tên vocabs.txt gồm ….từ. Quá trình tiền xử lý cũng sẽ xác định kết thúc của một câu, giá trị đó sẽ được đặt là “…..”.

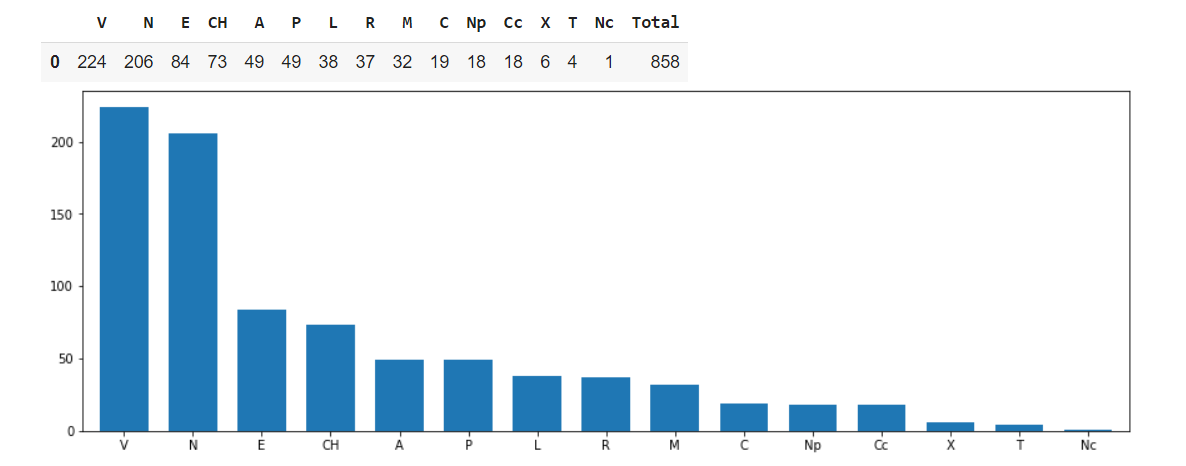
Tập Train gồm các file train\_gold.txt và train.txt:

* train\_gold.txt: chứa các từ kèm nhãn để thực hiện cho việc huấn luyện (tạo ra các ma trận transition\_counts, emission\_counts, tag\_counts)
* train.txt: chỉ chứa từ của các câu để thực hiện cho việc kiểm thử trên tập train (kiểm tra overfitting).
* Số lượng câu của tập Train:
* Số lượng từ của tập Train:
* Số lượng nhãn của tập Train:

Ảnh có chứa văn bản

Mô tả được tạo tự động

*Một số từ trong tập Train*

**

*Các nhãn trong tập Train*

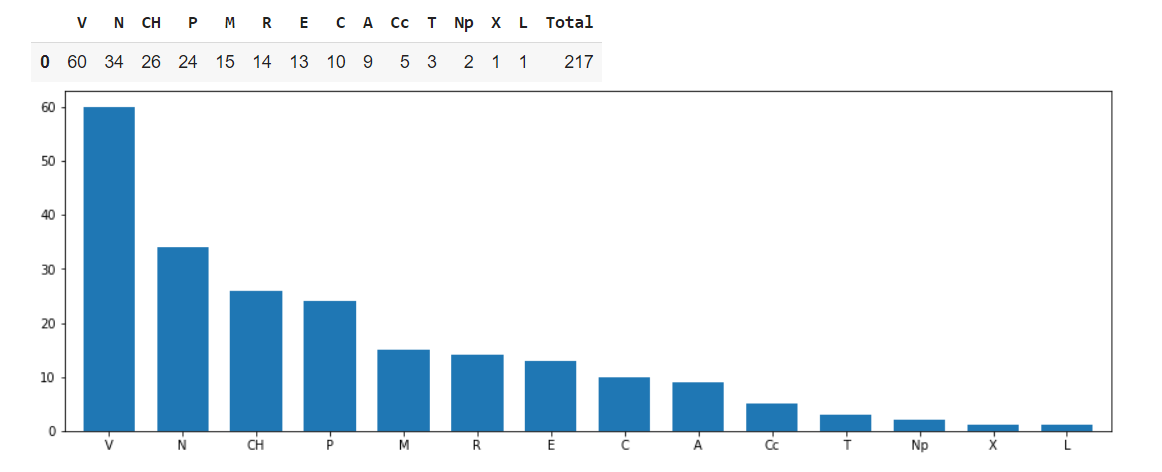
Tập Test gồm các file test\_gold.txt và test.txt:

* test\_gold.txt: chứa các từ kèm nhãn, thực hiện cho việc đánh giá kết quả.
* test.txt: chỉ chứa từ của các câu để thực hiện cho việc dự đoán.
* Số lượng câu của tập Test:
* Số lượng từ của tập Test:
* Số lượng nhãn của tập Test:

*Ảnh có chứa văn bản

Mô tả được tạo tự động*

*Một số từ trong tập Test*

**

*Các nhãn trong tập Test*

1. **Gán nhãn từ loại**

Bắt đầu với mô hình gán nhãn từ loại đơn giản nhất có thể và sau đó sẽ xây dựng lên mô hình phức tạp hơn.

1. **Training**

Trước khi bắt đầu dự đoán nhãn của mỗi từ, nhóm tính toán một số từ điển (dictionary) sẽ giúp tạo ra các bảng. Ngoài ra, nhóm bổ sung thêm nhãn ‘−s−’ để chỉ ra phần bắt đầu của mỗi câu.

Từ điển Transition Counts:

* Tính số lần mỗi nhãn xảy ra bên cạnh một nhãn khác.
* Từ điển sẽ tính giá trị P(ti|ti-1). Đây là xác suất của nhãn ở vị trí I được cho bởi nhãn ở vị trí i-1. Để tính toán giá trị này, nhóm sẽ tạo một từ điển tên là transition\_counts, trong đó: keys là (prev\_tag, tag), value là số lần 2 nhãn đó xuất hiện theo thứ tự đó.

*Ảnh có chứa văn bản

Mô tả được tạo tự động*

*Một số giá trị trong từ điển Transition Counts*

Từ điển Emission Counts:

* Tính xác suất của một từ được cho bởi nhãn của nó.
* Từ điển sẽ tính giá trị P(wi|ti). Để tính toán giá trị này, nhóm sẽ tạo một từ điển tên là emission\_counts, trong đó: keys là (tag, word), value là số lần cặp giá trị đó xuất hiện trong tập Train

Ảnh có chứa văn bản

Mô tả được tạo tự động

*Một số giá trị trong từ điển Emission Counts*

Từ điển Tag Counts:

* Được đặt với tên tag\_counts.
* key là nhãn, value là số lần nhãn đó xuất hiện.

**

*Các nhãn trong từ điển Tag Counts*

1. **Testing**

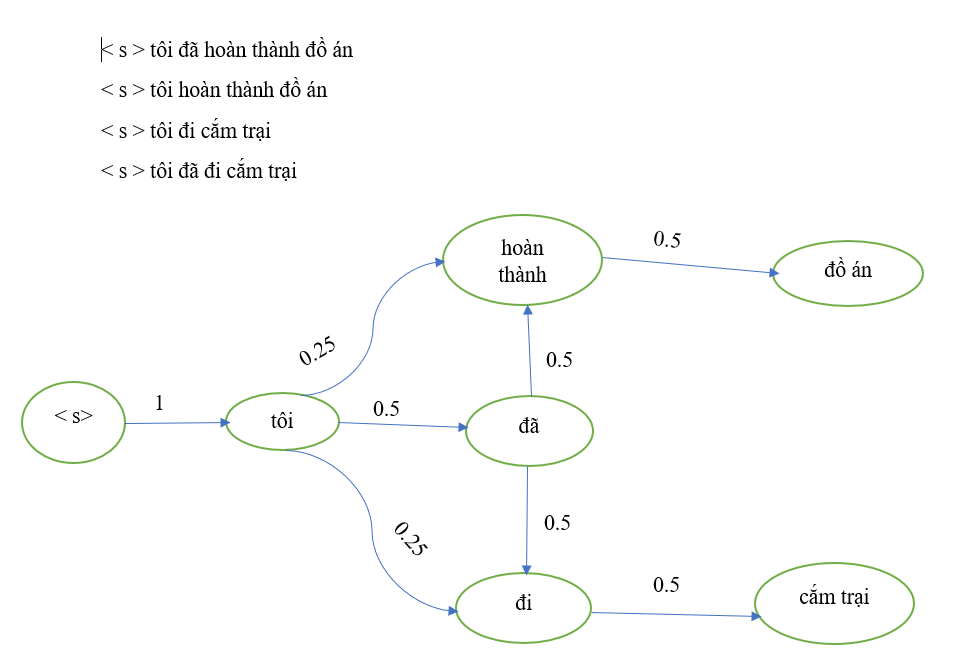
Sau khi tạo ra các từ điển, nhóm bắt đầu kiểm tra độ chính xác của mô hình gắn thẻ đơn giản này bằng cách sử dụng từ điển Emission Counts.

Để gắn nhãn cho một từ, nhóm chỉ định nhãn thường gặp nhất cho từ đó trong tập Train. Sau đó đánh giá xem cách tiếp cận này hoạt động có tốt không. Mỗi lần dự đoán sẽ dựa trên nhãn thường xuyên nhất cho từ đã cho rồi sẽ kiểm tra xem có giống với nhãn thực của từ không. Nếu có thì dự đoán đã đúng. Tính độ chính xác bằng số dự đoán đúng chia cho tổng số từ đã dự đoán nhãn:

* Độ chính xác trên tập Train: 0.8712206047032475
* Độ chính xác trên tập Test: 0.6637168141592921

1. **Mô hình Hidden Markov (HMM)**
2. **Markov Chain**

Mô hình Hidden Markov (HMM) dựa trên Markov Chain. Một Markov Chain là một mô hình cho chúng ta biết về xác suất của một chuỗi các biến ngẫu nhiên hay còn được gọi là các trạng thái, mỗi trạng thái có thể nhận những giá trị từ một tập nào đó. Các tập này có thể là tập các từ, các nhãn hoặc những biểu tưởng trình bày một điều gì đó. Một Markov Chain thì 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.



*Markov Chain*

Một cách tổng quát hóa, xét chuỗi các biến trạng thái q1, q2, …., qi. Mô hình Markov thể hiện giả định Markov về các xác suất của chuỗi này đó là khi dự đoán tương lai không quan trọng quá khứ mà chỉ cần hiện tại.

Giả định Markov: P(qi = a|q1, …, qi-1) = P(qi = a|qi-1)

Một Markov Chain được xác định bởi các thành phần sau:

* Q = q1, q2, …, qN là một tập hợp của N trạng thái.
* A = a11, a12, …aN1,….aNN: là một ma trận chuyển trạng thái A, mỗi phần tử aij thể hiện xác suất chuyển từ trạng thái i đến trạng thái j với ràng buộc là = 1, ∀i.
* 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. **Giới thiệu HMM**

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 chứa một trạng thái và xác suất chuyển đổi giữa các trạng thái đó. Trong trường hợp này, các trạng thái là nhãn từ loại. Mô hình Markov sử dụng ma trận chuyển tiếp A (Transition Matrix). Mô hình Markov ẩn thêm một trận phát xạ 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, emission là các từ trong ngữ liệu. Trạng thái, thứ được xem là Hidden chính là nhãn của từ đó.

Dựa vào các từ điển transition\_counts, emission\_counts and tag\_counts đã thu được. Nhóm sẽ bắt đầu triển khai mô hình Hidden Markov. Điều này cho phép xây dựng Transition Matrix A và Emission Matrix B. Bên cạnh đó nhóm cũng sẽ sử dụng một tham số smoothing khi tính toán các ma trận này.

1. **Transition Matrix A**

Ảnh có chứa văn bản

Mô tả được tạo tự động

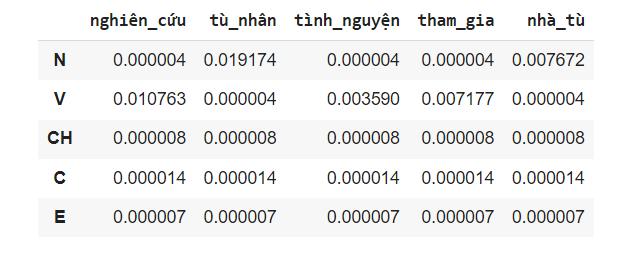
*Ma trận chuyển tiếp (Transition Matrix)*

Ma trận ở trên đã được tính toán với tham số smoothing. Mỗi ô là xác suất để đi từ 1 nhãn tới nhãn khác. Nói cách khác, có 0.077549 cơ hội để chuyển từ nhãn N tới nhãn E. Tổng của mỗi hàng phải bằng 1, vì giả định rằng nhãn tiếp theo phải là một trong các cột có sẵn trong bảng. Việc làm mịn được thực hiện như sau

P(ti|ti-1) =

* N: Tổng số nhãn.
* C(ti-1, ti): Số lượng bộ giá trị (prev\_tag, tag) trong từ điển transition\_counts.
* C(ti-1): Số lượng của nhãn trước trong từ điển tag\_counts.
* α: Là tham số smoothing.

1. **Emission Matrix B**



*Ma trận phát xạ (Emission Matrix)*

Ma trận này có chiều là (num\_tags, N) với num\_tags là số nhãn có thể có và N là số từ trong bộ từ vựng. Xác suất ma trận B được tính theo công thức:

P(wi|ti) =

* C(ti,wi): Là số lượng của từ thứ i (wi) đi với nhãn thứ i (ti) trong tập Train (được lưu trong từ điển emission\_counts).
* C(ti): Là số lần nhãn thứ i (ti) trong tập Train ( được lưu trong từ điển tag\_counts).
* N: Số lượng từ trong từ điển.
* α: Tham số smoothing.

1. **Thuật toán Viterbi**
2. **Giới thiệu**

Vào năm 1967, Viterbi là người đầu tiên đưa ra thuật toán Viterbi. Thuật toán Viterbi là thuật toán phân tích dãy trạng thái ẩn gọi là đường Viterbi. Mô hình này thường được sử dụng trong ngữ cảnh của nguồn Markov hoặc nguồn mà các biến ngẫu nhiên đưa ra các ẩn số quan trọng.

Về cơ bản, thông qua các phương tiện logic, thuật toán Viterbi xem xét một tập hợp các đối tượng theo các thuộc tính nhất định và cố gắng chứng minh các thuộc tính đó có thể ảnh hưởng đến những thuật tính khác như thế nào. Điều này thường được gọi là chuỗi Markov và có thể được chứng minh bằng sơ đồ. Các thuật toán Viterbi rất hữu ích trong các công nghệ như phần mềm nhận dạng giọng nói, chương trình phát hiện từ khóa và trong một số loại hệ thống phần mềm tin sinh học.

Tiến hành kết hợp mô hình Hidden Markov đã có với thuật toán Viterbi sử dụng quy hoạch động. Cụ thể, sẽ sử dụng 2 ma trận của A, B để tính toán thuật toán Viterbi. Quy trình này sẽ được chia thành 3 bước chính:

* Khơi tạo: Khởi tạo 2 ma trận best\_paths và best\_probabilities sẽ được dùng cho hàm feed\_forward.
* 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.

1. **Khởi tạo**

Khởi tạo 2 ma trận có cùng chiều:

* 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 (‘….’):

* Xác suất của đường đi tốt nhất từ vị trí bắt đầu đến một nhãn nhất định có vị trí i được ký hiệu là best\_probs[sidx,i]. Đây là xác suất mà nhãn bắt đầu đi sang nhãn được biểu thị bằng chỉ số i.
* A[sidx, i] và nhãn được biểu thị bằng chỉ số I cho ra từ đầu tiên của ngữ liệu là B[i, vocabs[corpus[0]]], trong đó vocabs là từ điển mà trả về 1 số nguyên duy nhất tương ứng với 1 từ cụ thể và corpus[0] là từ đầu tiên của ngữ liệu.

Việc này được thực hiện như sau:

best\_probs[sidx, i] = ln(A[sidx, i]) + ln(B[i, corpus[0]])

Để tránh việc nhân và lưu các giá trị nhỏ, nhóm sẽ lấy ln của tích trên để chúng trở thành tổng của 2 log:

best\_probs[sidx, i] = ln(A[sidx, i]) + ln(B[I, corpus[0]])

1. **Viterbi Forward**

Điền thông tin vào ma trận best\_probs và best\_paths đã khởi tạo bằng hàm viterbi\_forward:

* Lặp qua ngữ liệu.
* Với mỗi từ, tính xác suất cho mỗi nhãn có thể có.
* Tính toán sẽ bao gồm cả đường đi đến tổ hợp (từ, thẻ) đó.

Công thức để tính xác suất và đường đi cho từ thứ I, từ trước đó i – 1 trong ngữ liệu, nhãn j hiện tại và nhãn k trước đó là:

prob = best\_probk, i-1 + log(Ak,i) + log(Bj,vocabs(corpus))

* corpusi: từ ở vị trí thứ I trong ngữ liệu.
* vocabs: từ điển trả về những số nguyên duy nhất đại diện cho từ nhất định.
* k: số nguyên đại diện nhãn trước đó.

Triển khai hàm viterbi\_forward, lưu trữ best\_path và best\_prob của mọi nhãn có thể có cho mỗi từ trong ma trận best\_probs và best\_tags bằng cách dùng mã giả bên dưới:

**Function** viterbi\_forward (A, B, corpus, best\_probs, best\_paths, vocabs\_dict):

**for** i ← 1: count(corpus) **do**

**for** j ← 0: count(số nhãn) **do**

best\_prob\_i ← âm vô cùng

best\_path\_i ← null

**for** k ← 1: count(số nhãn) **do**

index ← vị trí từ thứ i trong corpus

prob ← best\_probs[k][i-1]+log(A[k][j])+log(B[j][index])

**if** prob>best\_prob\_i **then**

best\_prob\_i ← prob

best\_path\_i ← k

**end if**

**end for**

best\_probs[j][i] ← best\_prob\_i

best\_path[j][i] ← best\_path\_i

**end for**

**Return** best\_probs, best\_paths

1. **Viterbi Backward**

Thuật toán Viterbi Backward bằng cách sử dụng các ma trận best\_paths và best\_probs sẽ trả về danh sách các nhãn được dự đoán cho mỗi từ trong ngữ liệu:

* Lặp qua tất cả các hàng (nhãn) tại cột cuối cùng của best\_probs và tìm hàng (nhãn) có giá trị lớn nhất cho từ cuối cùng.
* Bắt đầu tại cột cuối cùng của best\_paths, sử dụng best\_probs để tìm nhãn có nhiều khả năng nhât cho từ cuối cùng trong ngữ liệu. Sau đó, sử dụng best\_paths để tìm nhãn có nhiều khả năng nhất cho từ trước đó và cập nhập lại nhãn cho mỗi từ (Tìm các nhãn tốt nhất bằng cách đi lùi qua best\_paths từ từ cuồi cùng đến từ thứ 0 trong ngữ liệu).

**Mã giả hàm viterbi\_backward:**

**function** viterbi\_backward(best\_probs, best\_paths, corpus, states):

m ← count(corpus)

khởi tạo mảng z với m phần tử

khởi tạo mảng pred với m phần tử

best\_prob\_for\_last\_word ← âm vô cùng

**for** k ← 1: count(số nhãn) **do**

**if** best\_probs[k][m-1] > best\_prob\_for\_last\_word **then**

best\_prob\_for\_last\_word ← best\_probs[k][m-1]

z[m] ← k

**end if**

**end for**

pred[m] ← states[z[m-1]]

**for** i ← m-1: -1 **do**

z[i-1] ← best\_paths[z[i], i]

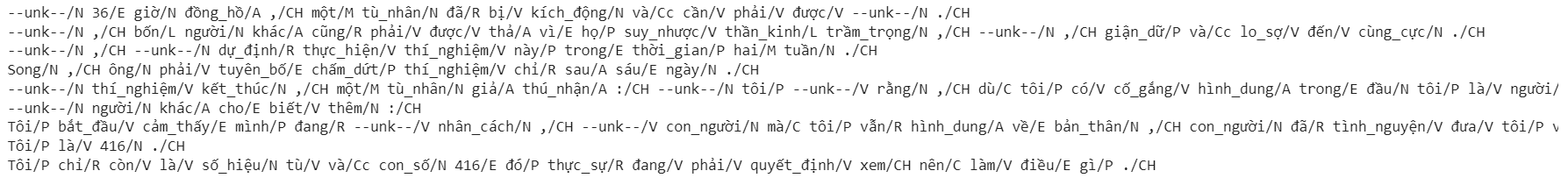
pred[i-1] ← states[z[i-1]]

**end for**

**Return** pred

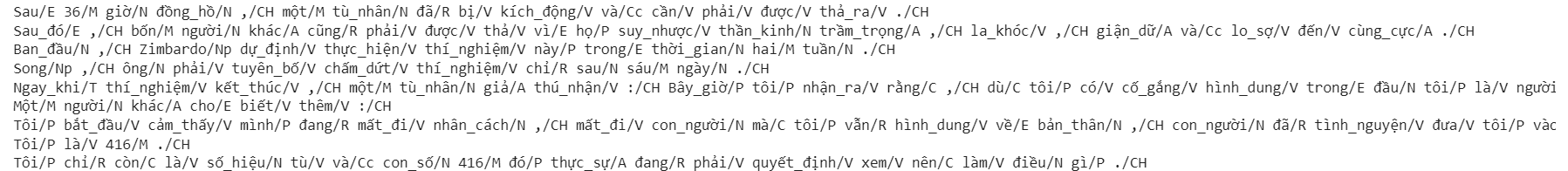
1. **Kết quả và đánh giá**

Sau khi xây dựng xong mô hình Hidden Markov kết hợp thuật toán Viterbi. Nhóm tiến hành dự đoán 10 câu đã được chia cho tập Test, sau đó sẽ so sánh kết quả này với kêt quả sử dụng thư viện VnCoreNLP.



*Kết quả gán nhãn 10 câu trong tập Test sử dụng Hidden Markov và Viterbi*

Như đã phân tích những từ không thuộc bộ từ vựng sẽ có giá trị là −unk−, với VnCoreNLP đều có thể nhận biết được hết những từ này.



*Kết quả gán nhãn 10 câu trong tập Test sử dụng thư viện VnCoreNLP*

Kết quả đánh giá sau khi so sánh với thu viện VnCoreNLP trên tập Test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | F1-score | Precision | Recall |
| HMM + Viterbi | 0.73 | 0.65 | 0.68 | 0.71 |
| VnCoreNLP | 1.00 | 1.00 | 1.00 | 1.00 |

*Đánh giá gán nhãn trên tập Test*

Có thể thấy VnCoreNLP cho kết quả khá tốt so với khi sử dụng mô hình HMM kết hợp với thuật toán Viterbi. Đây có thể do dữ liệu dùng để train cho mô hình này quá ít, chỉ có 35 câu.

Sau khi xem xét kết quả trên tập Train của mô hình này thì thu dược:

* F1-score: 0.84
* Precision: 0.81
* Recall: 0.90

Kết quả trên tập Train rất tốt nhưng trên tập Test lại khá tệ chứng tỏ mô hình đã bị overfiting.

Tiến hành xem kết quả đánh giá chi tiết của từng nhãn. Các nhãn f1-score khá lần lượt là nhãn “CH: 0.9”, “Cc: 1.00”, “M: 0.75”, “P: 0.86”, “R: 0.76”, “V: 0.77”, “X: 1.00”. Các nhãn f1-score dưới mức trung bình lần lượt là “A: 0.29”, “L: 0.33”, “Np: 00”.

Ảnh có chứa bàn

Mô tả được tạo tự động

*Kết quả đánh giá chi tiết trên tập Test sử dụng Hidden Markov và Viterbi*

1. **Kết luận**

Trong đề tài này, nhóm đã áp dụng các kiến thức về xử lý ngôn ngữ tự nhiên để xây dựng bộ tách từ bằng thuật toán Longest Matching, sau đó thực hiện gán nhãn bán thủ công để có thể tạo ra ngữ liệu sẽ được sử dụng cho mô hình Hidden Markov để thực hiện gán nhãn từ loại Tiếng Việt.

Với việc tách từ thì việc áp dụng thuật toán Longest Matching khá đơn giản. Tuy nhiên nhược điểm của phương pháp này là không tách được các từ không có trong 2 bộ dữ liệu bi-grams và tri-grams, nhất là các từ riêng.

Bắt đầu gán nhãn từ loại với mô hình đơn giản nhất có thể, sử dụng các từ điển điếm số lượng đã được xây dựng trước. Kết quả thu được lúc này trên tập Test chỉ có …Sau khi xây dựng xong mô hình Hidden Markov và sử dụng kết hợp với thuật toán Viterbi để thực hiện dự đoán trên tập Test thì kết quả trên đã tăng lên đáng kể, từ …. Lên …..

Kết luận mô hình đã bị overfitting do dữ liệu để train chỉ có …. câu. Hướng giải phát triển của nhóm có thể là sẽ thu nhập càng nhiều dữ liệu hơn từ nhiều nguồn khác nhau, bao quát nhiều ngữ cảnh hơn trong tiếng Việt.

Bên cạnh đó nhóm sẽ tìm hiểu thêm những cách triển khai khác sử dụng gán nhãn 2 chiều (Bidiretional POS tagging). Gán nhãn 2 chiều yêu cầu biết được từ trước đó và từ tiếp theo trong ngữ liệu khi dự đoán nhãn của từ hiện tại. Gán nhãn 2 chiều sẽ cho ta biết thêm về nhãn thay vì chỉ biết từ trước đó. Vì đã học được cách triển khai phương pháp tiếp cận đơn hướng qua đề tài này, nhóm đã có nền tảng để triển khai các trình gán nhãn khác được sử dụng trong thực tế.

**TÀI LIỆU THAM KHẢO**

<https://www.techopedia.com/definition/9950/viterbi-algorithm>

<https://filegi.com/tech-term/viterbi-algorithm-3489/>