**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

🙤🙧🟍🙥🙦



**GÁN NHÃN TỪ LOẠI TIẾNG VIỆT**

**VỚI MÔ HÌNH HIDDEN MARKORV**

|  |  |  |
| --- | --- | --- |
| Giảng viên hướng dẫn: | | |
| Th.S Nguyễn Trọng Chỉnh | | |
| Danh sách thành viên: | | |
| STT | Họ tên | MSSV |
| 1 | Đặng Hoàng Quân | 18520339 |
| 2 | Phạm Phú Phước | 18520131 |
| 3 | Dương Quốc Lộc | 18521006 |

**TP. HỒ CHÍ MINH – 01/2021**

# MỤC LỤC

[MỤC LỤC 2](#_Toc62775828)

[TÓM TẮT 4](#_Toc62775829)

[CHƯƠNG 1: GIỚI THIỆU 5](#_Toc62775830)

[CHƯƠNG 2: THU THẬP DỮ LIỆU 6](#_Toc62775831)

[2.1. Nguồn thu thập 6](#_Toc62775832)

[2.2. Thông tin cơ bản 6](#_Toc62775833)

[CHƯƠNG 3: TÁCH TỪ 7](#_Toc62775834)

[3.1. Thuật toán Longest Matching 7](#_Toc62775835)

[3.1.1. Giới thiệu 7](#_Toc62775836)

[3.1.2. Mã giả 7](#_Toc62775837)

[3.1.3. Ưu và nhược điểm 8](#_Toc62775838)

[3.2. Cách triển khai 8](#_Toc62775839)

[3.3. Đánh giá kết quả 9](#_Toc62775840)

[CHƯƠNG 4: TẠO NGỮ LIỆU VÀ GÁN NHÃN 10](#_Toc62775841)

[4.1 Tạo ngữ liệu 10](#_Toc62775842)

[4.2. Gán nhãn từ loại 14](#_Toc62775843)

[4.2.1 Training 14](#_Toc62775844)

[4.2.2 Testing 14](#_Toc62775845)

[4.3. Mô hình Hidden Markov 14](#_Toc62775846)

[4.3.1 Ma trận chuyển tiếp 14](#_Toc62775847)

[4.3.2 Ma trận phát xạ 14](#_Toc62775848)

[4.4. Thuật toán Viterbi 14](#_Toc62775849)

[4.4.1 Khởi tạo 14](#_Toc62775850)

[4.4.2 Viterbi Forward 15](#_Toc62775851)

[4.4.3 Viterbi Backward 15](#_Toc62775852)

[CHƯƠNG 5: KẾT QUẢ DỰ ĐOÁN VÀ ĐÁNH GIÁ 16](#_Toc62775853)

[5.1. Các độ đo được sử dụng 16](#_Toc62775854)

[5.1.1. Độ đo Accuracy 16](#_Toc62775855)

[5.1.2. Độ đo Precision 16](#_Toc62775856)

[5.1.3. Độ đo Recall 16](#_Toc62775857)

[5.1.4. Độ đo F1-score 16](#_Toc62775858)

[5.2. Kết quả 17](#_Toc62775859)

[5.3. Nhận xét 17](#_Toc62775860)

[CHƯƠNG 6: KẾT LUẬN 18](#_Toc62775861)

# TÓM TẮT

Tìm hiểu cách gán nhãn từ loại Tiếng Việt

Tính toán ma trận chuyển tiếp A trong Mô hình Markov ẩn

Tính toán ma trận chuyển tiếp B trong Mô hình Markov ẩn

Tính toán thuật toán Viterbi

Tính toán độ chính xác của mô hình

# CHƯƠNG 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 tử 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 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 dung 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

# CHƯƠNG 2: THU THẬP DỮ LIỆU

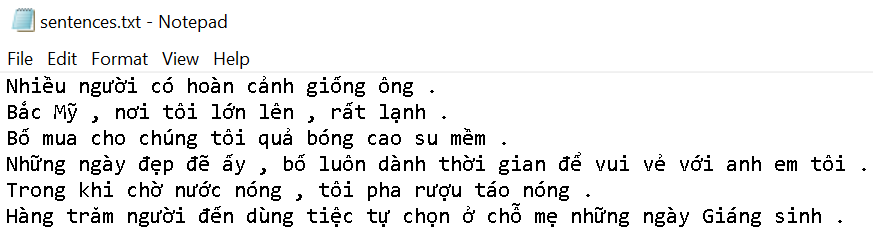
## Nguồn thu thập

Gồm các câu bất kỳ thuộc nhiều chủ đề được thu thập từ 2 trang <https://vnexpress.net/> và <https://dantri.com.vn/>. Do đây là 2 nguồn báo uy tín ở Việt Nam nên nhóm sẽ lựa chọn để sử dụng cho việc thu thập dữ liệu.

## 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 cách ‘ ’.
* Cuối câu là 1 dấu chấm ‘.’.
* Số lượng câu: 60.
* Số từ nhiều nhất trong một câu: 46.
* Số từ ít nhất trong một câu: 8.



Hình 2: Một số câu có trong bộ dữ liệu gốc.

# CHƯƠNG 3: 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 (word) “cá\_nhân” được tạo lên bởi 2 âm (syllable) là “cá” và “nhân”. Trong khi hai từ đơn “cá” và từ đơn “nhân” 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.

## Thuật toán Longest Matching

### Giới thiệu

Longest Matching là thuật toán dựa trên tử tưởng tham lam. Nó xét các tiếng từ trái qua phải, các tiếng đầu tiên dài nhất có thể mà xuất hiện trong từ điển sẽ được tách ra làm một từ. Với vị trí âm tiết hiện tại sẽ kiểm tra xem từ đó và 2 âm tiếp theo có thể ghép thành 1 từ có nghĩa hay không bằng cách kiểm tra trong từ điển tri\_gram. Nếu không thể tạo ra được từ có nghĩa từ 3 âm tiết thì ta tiếp tục kiểm tra xem âm tiết hiện tại và âm tiếp theo có thể ghép được thành một từ có nghĩa hay không bằng cách kiểm tra trong từ điển bi\_gram. Thuật toán sẽ dừng khi xét hết các tiếng trong câu.

### Mã giả

current\_word \_id = 0 (Đặt từ hiện tại ở đầu câu):

While (curret\_word\_id+1) != len(text) {

xét từ hiện tại với 2 từ sau nó trong từ điển tri\_gram

check if {“word, word+1, word+2”} is in tri\_gram

if true: take {word, word1, word2; current\_word\_id += 3}🡪 3 từ đó tạo thành từ ghép

xét từ hiện tại với tiếng sau trong từ điển trong từ điển bi\_gram

else check if: take {“word, word+1”} is in bi\_gram

if true: take {word, word1; current\_word\_id += 2}🡪 2 tiếng đó tạo thành từ ghép

else => {take word; current\_word\_id += 1} 🡪 tiếng đó là từ đơn

}

### Ưu và nhược điểm

Ưu điểm thuật toán này chính là cài đặt đơn giản, độ phức tạp tính toán hợp lý, không yêu cầu dữ liệu huấn luyện. Tuy nhiên, phương pháp này sẽ khó có thể xử lý được các tình huống nhập nhằng nhất định như việc lặp từ trong câu cũng như xử lý tình huống xuất hiện từ mới không tồn tại trong từ điển

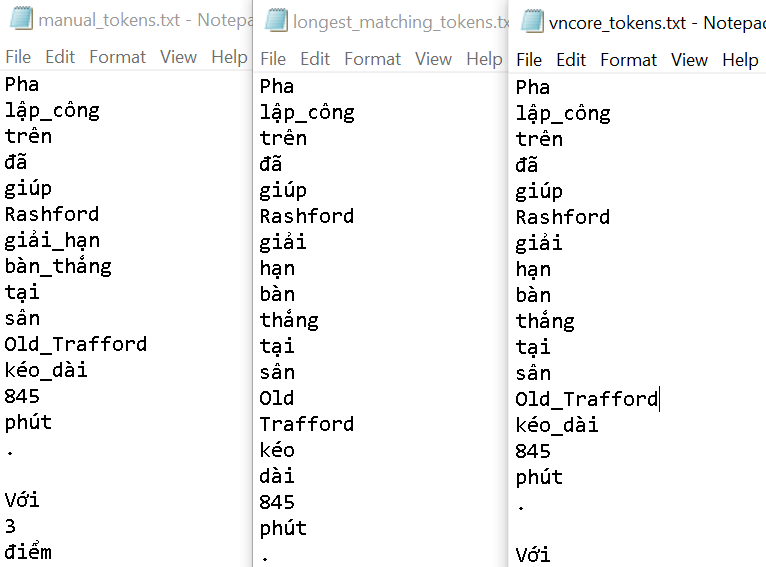
## Cách triển khai

Nhóm sẽ 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 1 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 60 câu nhóm mà 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: 970.
* Số lượng từ ghép: 309.

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 Longest 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 longest\_matching\_tokens.txt chứa 1049 từ với 257 từ ghép và vncore\_tokens.txt chứa 1006 từ với 282 từ ghép.



Hình 3: Kết quả tách từ theo từng phương pháp khác nhau của câu đầu tiên

## Đánh giá kết quả

Bảng 0.1: Tổng hợp kết quả tách từ.

|  |  |  |
| --- | --- | --- |
|  | Longest Matching | VnCoreNLP |
| Accuracy | 0.856044 | 0.931868 |
| Precision | 0.891051 | 0.950355 |
| Recall | 0.7411 | 0.867314 |
| True Positive | 229 | 268 |
| False Positive | 28 | 14 |
| Total True | 779 | 848 |
| Total Errors | 210 | 99 |

Qua bảng 1, có thể thấy rằng các số liệu đánh giá của việc tách từ khi sử dụng thư viện VnCoreNLP đều rất tốt và tốt hơn nhiều so với khi dùng thuật toán Longest Matching đồng thời nhóm nhận thấy Longest Matching không phân biệt tốt được các tên riêng như Old\_Trafford, … có thể thấy ở hình 3. Đây có thể do bộ dữ liệu bi-grams và tri-grams được dùng cho thuật toán này chưa đủ 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.

# CHƯƠNG 4: TẠO NGỮ LIỆU VÀ GÁN NHÃN

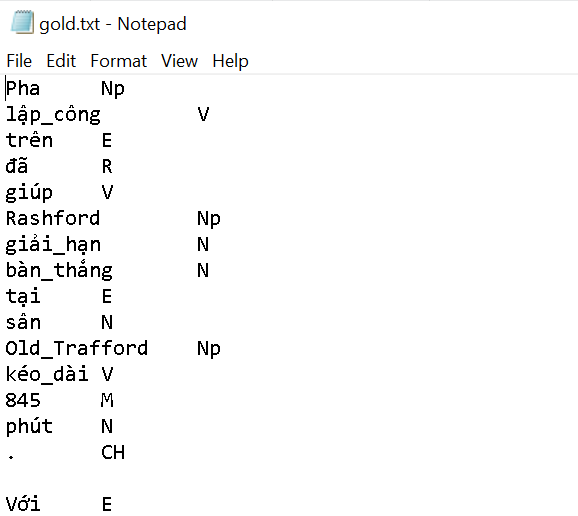
## 4.1 Tạo ngữ liệu

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

Nhóm sẽ 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 60 câu đã thu thập:

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: 60.
* Số lượng từ: 970.
* Số lượng nhãn: 909.



Hình 4: Một số từ kèm nhãn trong bộ dữ liệu GOLD

*Bảng 2: Danh sách nhãn từ loại*

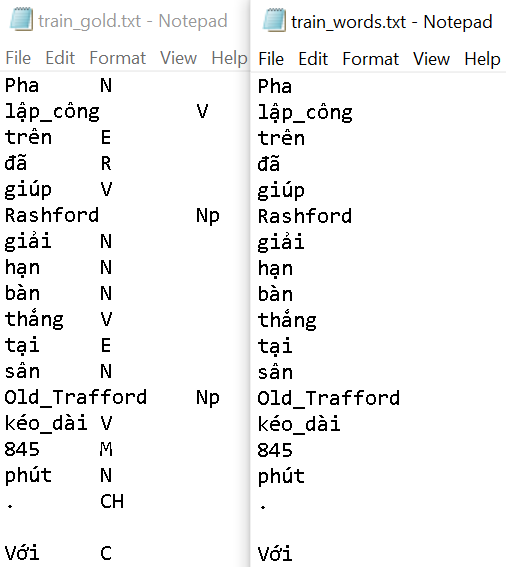
*(Nguồn:* [*https://github.com/vncorenlp/VnCoreNLP/blob/master/VLSP2013\_POS\_tagset.pdf*](https://github.com/vncorenlp/VnCoreNLP/blob/master/VLSP2013_POS_tagset.pdf)*)*

|  |  |  |  |
| --- | --- | --- | --- |
| **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 Hà Nội, Mộc tinh, Hoả 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, đồng |
| 5 | Ni | Danh từ ký hiệu | A1, A4, 60A, 60B, 20a, 20b, ABC, ABCD |
| 6 | V | Động từ | ngủ, ngồi, cười; đọc, viết, đá, đặt; thích, yêu |
| 7 | A | Tính từ | tốt, xấu, đẹp; cao, thấp, rộng |
| 8 | P | Đại từ | tôi, chúng tôi, hắn, nó, y, đại nhân, đại ca, |
| 9 | L | Định từ | mỗi, từng, mọi, cái; các, những, mấy |
| 10 | M | Số từ | một, mười, mười ba; dăm, vài, mươi, nửa |
| 11 | R | Phó từ | đã, sẽ, đang, vừa, mới, từng, xong, rồi, rất |
| 12 | E | Giới từ | trên, dưới, trong, ngoài; của, trừ, ngoài, khỏi |
| 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 | . ! ? , ; : |

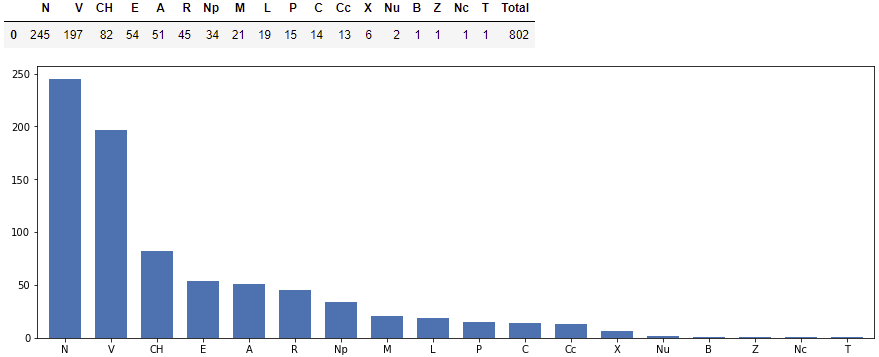
Tiếp theo nhóm sẽ chia 60 câu đã được tách từ và gán nhãn trong bộ dữ liệu Gold này thành 2 tập dữ liệu: 50 câu cho tập Train và 10 câu cho tập Test. Vì trong lúc dự đoán mô hình gán nhãn có thể sẽ gặp những từ không có trong dataset của nó. Những từ này sẽ được thay thế bằng một mã không xác định. Ở 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 ‘—unk—’. Bộ từ vựng được lưu với tên vocabs.txt gồm 54818 từ. Đồng thời, quá trình tiền xử lý cũng sẽ xác định kết thúc của một câu, giá trị đó sẽ được đặt là ‘—n—’.

Tập Train gồm các file train\_gold.txt và train\_words.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\_words.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: 50.
* Số lượng từ của tập Train: 852.
* Số lượng nhãn của tập Train: 802.



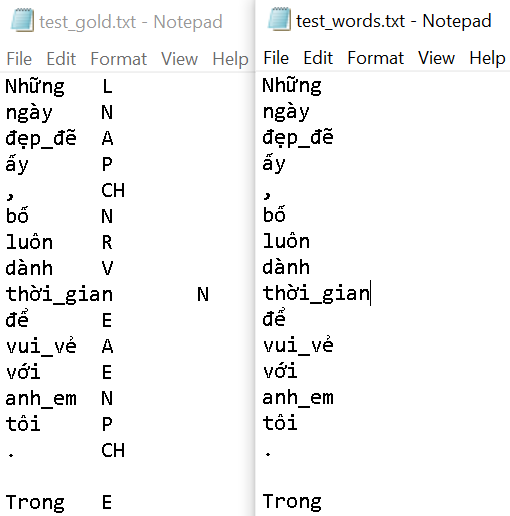
Hình 5: Một số từ trong tập Train



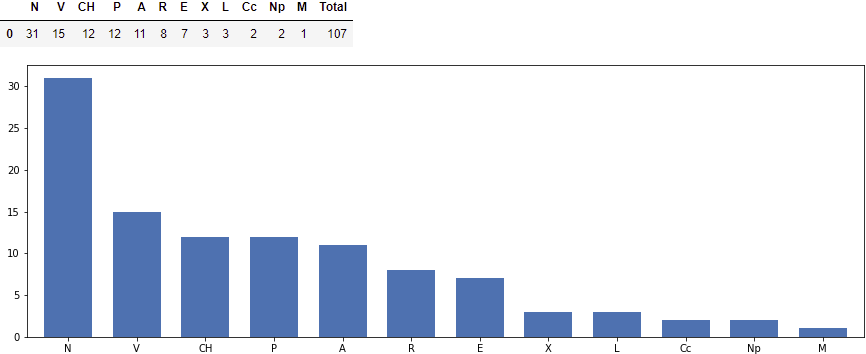
Hình 6: Các nhãn trong tập Train

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

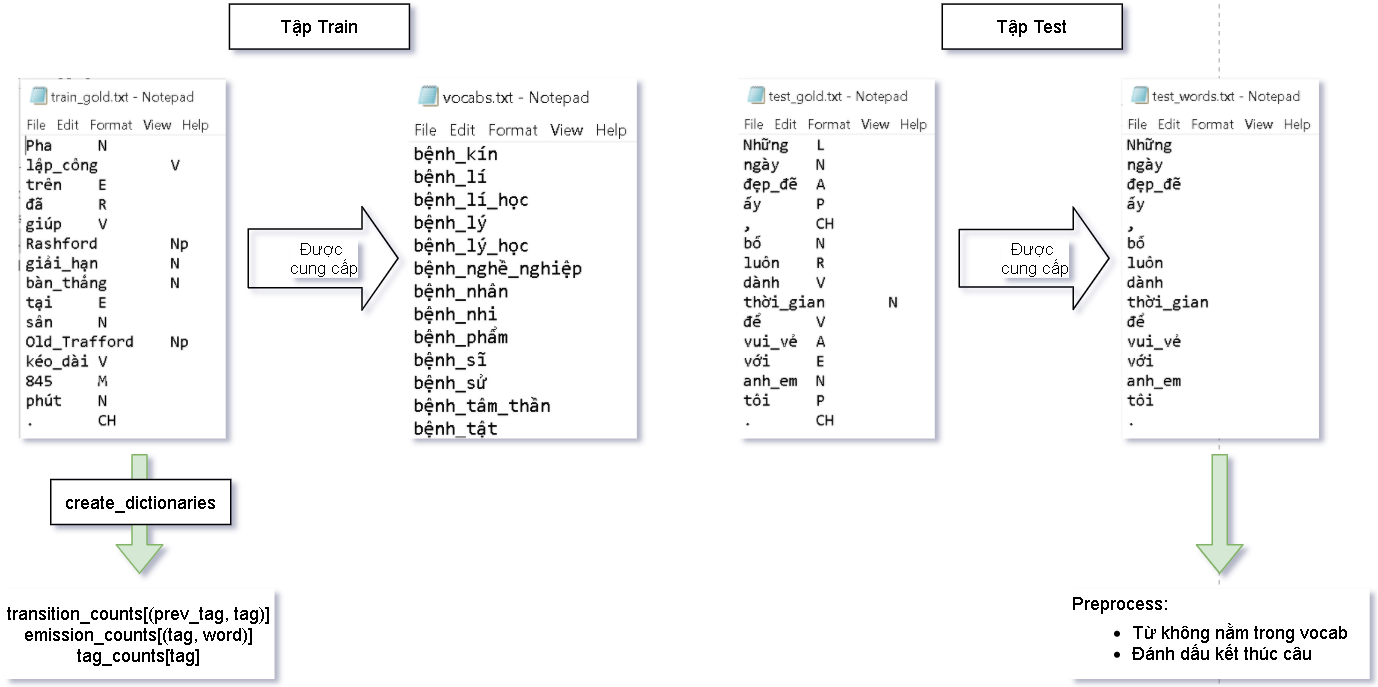
* test\_gold.txt: chứa các từ kèm nhãn để thực hiện cho việc đánh giá kết quả gán nhãn
* test\_words.txt: chỉ chứa từ của các câu để thực hiện cho việc dự đoán
* Số lượng câu của tập Test: 10.
* Số lượng từ của tập Test: 117.
* Số lượng nhãn của tập Train: 107.
* Các từ không nằm trong vocab: Giáng\_sinh, năm\_qua, gần\_đây, Tân\_Sơn\_Nhất.



Hình 7: Một số từ trong tập Test



Hình 8: Các nhãn trong tập Test



Hình 8: Quá trình đọc dữ liệu 2 tập Train và Test

## 4.2. Gán nhãn từ loại

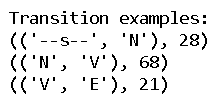
Nhóm sẽ bắt đầu với mô hình gán nhãn từ loại đơn giản nhất có thể và sau đó sẽ xây dựng lên mô hình phức tạp hơn.

### 4.2.1 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:

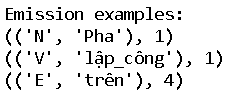
* Tính số lần mỗi nhãn xảy ra bên cạnh một nhãn khác
* Từ điển sẽ tính giá trị P(ti|ti-1). Đây là xác suất của nhãn ở vị trí i được cho bởi nhãn ở vị trí i-1. Để tính toán giá trị này, nhóm sẽ tạo một từ điển tên là transition\_counts, trong đó: keys là (prev\_tag, tag), value là số lần 2 nhãn đó xuất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



Hình 8: 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.



Hình 8: 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



Hình 8: Các nhãn trong từ điển Tag Counts

### 4.2.2 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 gán 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 mà đã dự đoán nhãn:

* Độ chính xác trên tập Train: 0.9330985915492958
* Độ chính xác trên tập Test: 0.4188034188034188

## 4.3. Mô hình Hidden Markov

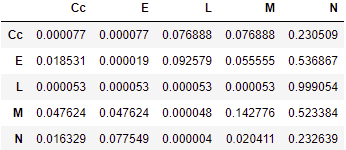
### 4.3.1 Giới thiệu

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 việc gán nhãn từ loại, HMM còn được sử dụng trong nhận dạng giọng nói, tổng hợp giọng nói, v.v.

Mô hình Markov chứa một số 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 ần thêm một ma 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 ẩn (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 làm mịn (smoothing) khi tính toán các ma trận này.

### 4.3.2 Transition Matrix A

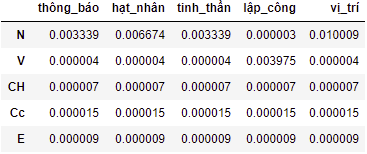


Hình 8: Ma trận chuyển tiếp (Transition Matrix)

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:

* : tổng số nhãn.
* : số lượng của bộ giá trị (prev\_tag, tag) trong từ điển transition\_counts.
* : số lượng của nhãn trước trong từ điển tag\_counts
* : là tham số smoothing.

### 4.3.3 Emission Matrix B



Hình 8: 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:

* : là số lượng của từ thứ đi với nhãn thứ ) trong tập Train (được lưu trong từ điển emission\_counts)
* : là số lần nhãn thứ 𝑖 ) trong tập Train (được lưu trong từ điển tag\_counts)
* 𝑁: số lượng từ trong từ điển.
* 𝛼: tham số smoothing.

## 4.4. Thuật toán Viterbi

Nhóm tiến hành kết hợp mô hình Hidden Markov đã có với thuật toán Viterbi sử dụng quy hoạch động. Cụ thể, nhóm 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:

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

### 4.4.1 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 (‘—s—’):

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

Việc này trông như sau:

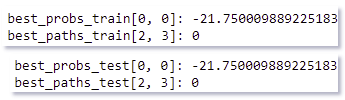
best\_probs

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

best\_probs

Tóm lại việc triển khai khởi tạo best\_probs như sau:

* best\_probs
* best\_probs



Hình 8: Khởi tạo thuật toán Viterbi cho 2 tập Train và Test

### 4.4.2 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 , từ trước đó  *–* *1* trong ngữ liệu, nhãn hiện tại và nhãn trước đó là:

prob best\_prob*k, i-1* +

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

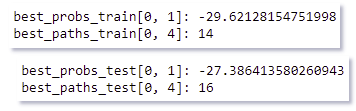
Triển khai hàm viterbi\_forward, lưu trữ best\_path và best\_prob của mọi nhãn có thể có cho mỗi từ trong ma trận best\_probs và best\_tags bằng cách sử dụng mã giả bên dưới:

for mỗi từ kho ngữ liệu

for mỗi loại nhãn mà từ này có thể là

for loại nhãn mà từ trước đó có thể là

* Tính xác suất để từ trước đó có nhãn nhất định, từ hiện tại có nhãn nhất định và nhãn sẽ cho ra từ hiện tại này.
* Giữ lại xác suất cao nhất được tính cho từ hiện tại
* Lưu xác suất cao nhất này vào best\_probs
* Lưu giá trị , đại diện cho nhãn của từ trước đó mà tạo ra xác suất cao nhất

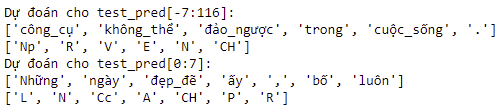


Hình 8: Viterbi Forward cho 2 tập Train và Test

### 4.4.3 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:

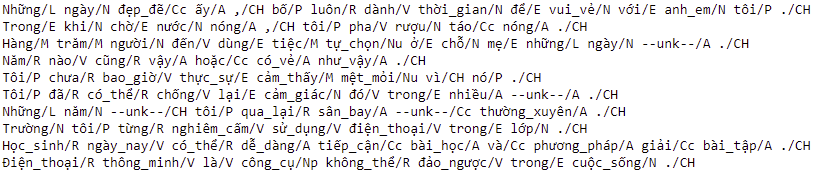
* Lặp qua tất cả các hàng (nhãn) tại cột cuối cùng của best\_probs và tìm hàng (nhãn) có giá trị lớn nhất cho từ cuối cùng.
* Bắt đầu tại cột cuối cùng của best\_paths, sử dụng best\_probs để tìm nhãn có nhiều khả năng nhất cho từ cuối cùng trong ngữ liệu. Sau đó, sử dụng best\_paths để tìm nhãn có nhiều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

****

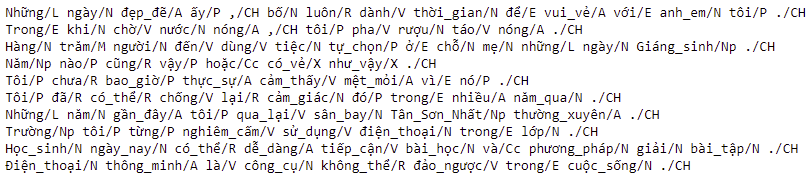
Hình 8: Một số kết quả dữ đoán sau khi Viterbi Backward trên tập Test

# CHƯƠNG 5: KẾT QUẢ DỰ ĐOÁN VÀ ĐÁNH GIÁ

Kết quả của mô hình Hidden Markov kết hợp thuật toán Viterbi

****

Hình 8: Kết quả gán nhãn 10 câu trong tập Test sử dụng Hidden Markov và Viterbi



Hình 8: Kết quả gán nhãn 10 câu trong tập Test sử dụng Hidden Markov và Viterbi

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | precision | | recall | | f1-score | |
|  | HMM+Viterbi | VnCoreNLP | HMM+Viterbi | VnCoreNLP | HMM+Viterbi | VnCoreNLP |
| A | 1 | 0.94 | 1 | 0.98 | 1 | 0.96 |
| B | 1 | 0 | 1 | 0 | 1 | 0 |
| C | 0.93 | 0.71 | 0.93 | 0.91 | 0.93 | 0.8 |
| CH | 1 | 1 | 1 | 1 | 1 | 1 |
| Cc | 1 | 1 | 1 | 1 | 1 | 1 |
| E | 0.91 | 1 | 0.96 | 0.86 | 0.93 | 0.92 |
| L | 1 | 0.74 | 1 | 0.93 | 1 | 0.82 |
| M | 1 | 1 | 1 | 0.95 | 1 | 0.98 |
| N | 1 | 0.89 | 0.99 | 0.97 | 1 | 0.93 |
| Nc | 1 | 1 | 1 | 1 | 1 | 1 |
| Np | 1 | 0.88 | 1 | 0.58 | 1 | 0.7 |
| Nu | 1 | 1 | 1 | 1 | 1 | 1 |
| P | 1 | 1 | 1 | 1 | 1 | 1 |
| R | 1 | 0.98 | 0.98 | 0.98 | 0.99 | 0.98 |
| T | 1 | 1 | 1 | 1 | 1 | 1 |
| V | 0.99 | 0.97 | 0.99 | 0.96 | 0.99 | 0.96 |
| X | 0.83 | 0.83 | 1 | 1 | 0.91 | 0.91 |
| Z | 1 | 1 | 1 | 1 | 1 | 1 |

Bảng 0.1

## 5.1. Các độ đo được sử dụng

### 5.1.1. Độ đo Accuracy

Độ đo accuracy là tỉ lệ giữa số điểm dữ liệu được dự đoán đúng và tổng số điểm dữ liệu trong tập dữ liệu. Độ chính xác giúp ta đánh giá hiệu quả dự báo của mô hình trên một bộ dữ liệu. Độ chính xác càng cao thì mô hình của chúng ta càng chuẩn xác. Công thức chung: Accuracy = . Hệ số accuracy cho bảng kết quả trên là (55+900)/1000 = 90.5%.

*Bảng 4: Ví dụ về các độ đo.*

|  |  |  |  |
| --- | --- | --- | --- |
| Dự đoán/Thực tế | | Thực tế | |
| BAD (Positive) | GOOD (Negative) |
| Dự đoán | BAD(Positive) | 55 (True Positive - TP) | 50 (False Positive - FP) |
| GOOD (Negative) | 45 (False Negative - FN) | 850 (True Negative - TN) |
| Tổng cộng |  | 100 | 900 |

### 5.1.2. Độ đo Precision

Độ đo Precision cho biết có bao nhiêu phần trăm trong các trường hợp được dự báo là positive là đúng. Công thức cho độ đo này là Precision = . Như vậy, độ đo precision cho bảng kết quả trên là 55/(50+55) = 52.4%.

### 5.1.3. Độ đo Recall

Recall đo lường tỷ lệ dự báo chính xác các trường hợp positive trên toàn bộ các mẫu thuộc nhóm positive. Công thức cho độ đo này là Recall = .

Như vậy, độ đo recall cho bảng kết quả trên là 55/(55+45) = 55%.

### 5.1.4. Độ đo F1-score

F1 Score là trung bình điều hòa giữa precision và recall. Do đó nó đại diện hơn trong việc đánh g**iá** độ chính xác trên đồng thời precision và recall.

Công thức của F1-score là:F1 = .

Như vậy, F1 của bảng kết quả trên là 2\*

## 5.2. Kết quả

Sau khi chạy thực nghiệm, nhóm em đã được kết quả sau trên bộ dữ liệu gold.

*Bảng 5: Kết quả của 3 thuật toán*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Thuật toán** | **Accuracy** | **Precision** | **Recall** | **F1** |
| 1 | TBL | 0.8125 | 0.8095 | 0.7857 | 0.7809 |
| 2 | HMM | 0.9375 | 0.9523 | 0.9761 | 0.9584 |
| 3 | Perceptron | 0.9617 | 0.9242 | 0.9586 | 0.9245 |

## 5.3. Nhận xét

Thuật toán Transformation Based Learning cho kết quả thấp nhất giữa 3 thuật toán. Các độ đo chỉ ở mức 0.80 với Accuracy đạt 81.25%. Độ đo F1 đạt 78.09%. Đây là một kết quả đạt ở mức chấp nhận được nhưng kết quả này có thể cải thiện hơn nữa trong 2 thuật toán tiếp theo.

Ở thuật toán Hidden Markov Model, tỉ lệ chính xác của mô hình này đạt 93.75%, một con số khá cao. Nhưng 3 độ đo còn lại trên thuật toán này còn đạt cao hơn với Precision, Recall, F1 lần lượt là 95.23%, 97.61% và 95.84%. Như vậy, F1 của HMM là cao nhất giữa 3 thuật toán.

Mặc dù thuật toán Perceptron cho kết quả Accuracy cao nhất (96.17%) nhưng 3 độ đo còn lại trên thuật toán này lại không đạt tốt bằng thuật toán Hidden Markov Model. Độ đo recall đạt 95.86%, cao nhất giữa 3 độ đo nên F1 đạt 92.45%.

Một điều đặc biệt giữa các độ đo là Recall cho kết quả cao hơn Precision trên 2 thuật toán Hidden Markov Model và Perceptron. Điều đó cho thấy số lượng dự đoán đúng trên tổng số lượng thực tế là đúng là tốt hơn so với tỉ lệ trên độ đo Precision.

# CHƯƠNG 6: KẾT LUẬN

Qua đồ án môn học Xử lí Ngôn ngữ Tự nhiên này, nhóm em đã làm được nhiều phần trong bài toán gán nhãn từ loại cho Tiếng Việt. Từ bộ dữ liệu thu thập trên một bài báo ở trang báo điện tử Zingnews. Nhóm em đã thực hiện các bước xử lí như tách từ bằng WFST, gán nhãn thủ công qua công cụ VLSP.

Sau khi có được bộ dữ liệu hoàn chỉnh, nhóm em tiến hành thực nghiệm trên ba mô hình là Transformation Based Learning, Hidden Markov Model và Perceptron. Mô hình TBL cho kết quả thấp nhất giữa ba mô hình với Accuracy đạt 81.25%. Mô hình Hidden Markov Model dù cho độ đo Accuracy thấp hơn mô hình Perceptron nhưng cho F1, Recall và Perceptrong với F1 đạt 95.84%. Độ đo Accuracy cao nhất ở thuật toán Perceptron với kết quả 96.17%.

Hạn chế của đồ án này là bộ dữ liệu không thật sự lớn. Dữ liệu là một phần rất quan trọng, không thể thiếu trong các dự án về máy học. Chất lượng dữ liệu ảnh hưởng lớn đến chất lượng của mô hình nhưng vì trong phạm vi đồ án này nên nhóm em chỉ có thể xây dựng một bộ dữ liệu nhỏ để phục vụ cho bài toán. Khi phải giải quyết với dữ liệu chưa có trong bộ dữ liệu thì độ chính xác của mô hình sẽ không cao như bảng kết quả trên. Vì vậy, hướng phát triển của bài toán này có thể là mở rộng dữ liệu hơn nữa, xây dựng một bộ dữ liệu lớn hơn, bao quát đầy đủ trường hợp ngữ cảnh trong tiếng Việt.