ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO CUỐI KÌ

MÔN HỌC: XỬ LÝ NGÔN NGỮ TỰ NHIÊN

GIẢNG VIÊN: THẦY NGUYỄN TRỌNG CHỈNH

LỚP CS221.N11

SINH VIÊN THỰC HIỆN:

BÙI KHÁNH DUY - 19521418

ĐINH HOÀNG LỘC - 19520148

Tp. Hồ Chí Minh, 01/2023

MỤC LỤC

LỜI CẢM ƠN

Sau quá trình học tập và rèn luyện tại Trường Đại học Công Nghệ Thông Tin, chúng em đã được trang bị các kiến thức cơ bản, các kỹ năng thực tế để có thể hoàn thành đồ án môn học của mình.

Chúng em xin gửi lời cảm ơn chân thành đến thầy Th.S. Nguyễn Trọng Chỉnh – Giảng viên phụ trách lớp CS221.M11 – Xử lý ngôn ngữ tự nhiên đã tận tâm hướng dẫn, truyền đạt những kiến thức cũng như kinh nghiệm cho chúng em trong suốt thời gian học tập.

Trong quá trình làm đồ án môn học, mặc dù nhóm chúng em đã cố gắng nhưng chắc chắn sẽ không tránh được những sai sót không đáng có. Mong nhận được sự góp ý cũng như kinh nghiệm quý báu của các các thầy và các bạn sinh viên để được hoàn thiện hơn và rút kinh nghiệm cho những môn học sau. Chúng em xin chân thành cảm ơn.

TP. Hồ Chí Minh, tháng 1 năm 2023

LÝ DO CHỌN ĐỀ TÀI

Trong những năm gần đây, cùng với sự phát triển vượt bậc của công nghệ thông tin đã thúc đẩy sự phát triển của nhiều lĩnh vực xã hội khác như: y học, giáo dục, giải trí, kinh tế, … Lĩnh vực xử lý ngôn ngữ tự nhiên đã ra đời và thâm nhập mạnh mẽ vào đời sống con người.

Một trong các vấn đề nền tảng của ngôn ngữ tự nhiên là việc phân loại các từ thành các lớp từ loại dựa theo thực tiễn hoạt động ngôn ngữ. Mỗi từ loại tương ứng với một lớp từ giữ một vai trò ngữ pháp nhất định. Tóm lại, mỗi từ trong một ngôn ngữ có thể gắn với nhiều từ loại, và việc tự động "hiểu" đúng nghĩa một từ phụ thuộc vào việc nó được xác định đúng từ loại hay không. Công việc gán nhãn từ loại cho một văn bản là xác định từ loại của mỗi từ trong phạm vi văn bản đó. Xác định từ loại chính xác cho các từ trong văn bản là vấn đề rất quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên. Công cụ gán nhãn từ loại có thể được ứng dụng rộng rãi trong các hệ thống tìm kiếm thông tin, trong các ứng dụng tổng hợp tiếng nói, các hệ thống nhận dạng tiếng nói cũng như trong các hệ thống dịch máy. Công cụ này cũng hỗ trợ cho việc phân tích cú pháp các văn bản, góp phần giải quyết tính đa nghĩa của từ, và trợ giúp các hệ thống rút trích thông tin hướng đến ngữ nghĩa, v.v…

Xuất phát trong hoàn cảnh đó, "Gán nhãn từ loại tiếng Việt" đã được nhóm em chọn làm đề tài để nghiên cứu báo cáo. Trong báo cáo này sẽ trình bày 2 bài toán quan trọng trong xử lý ngôn ngữ tự nhiên đó là bài toán tách từ và gán nhãn từ loại tiếng Việt.

PHẦN 1. GIỚI THIỆU BÀI TOÁN

Gán nhãn từ loại (POS tagging) hay phân tích cú pháp, là việc xác định từ loại trong câu văn. Đây là một trong những phương pháp quan trọng của xử lý ngôn ngữ tự nhiên, cũng là bước cơ bản trước khi phân tích sâu văn phạm hay các vấn đề xử lý ngôn ngữ phức tạp khác. 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… Thông thường, một từ có thể có nhiều chức năng ngữ pháp, ví dụ: trong câu "Bà đang la con la", cùng một từ "la" nhưng từ thứ nhất giữ chức năng ngữ pháp là động từ còn từ la thứ hai lại là danh từ trong câu.

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 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 Underthesea.

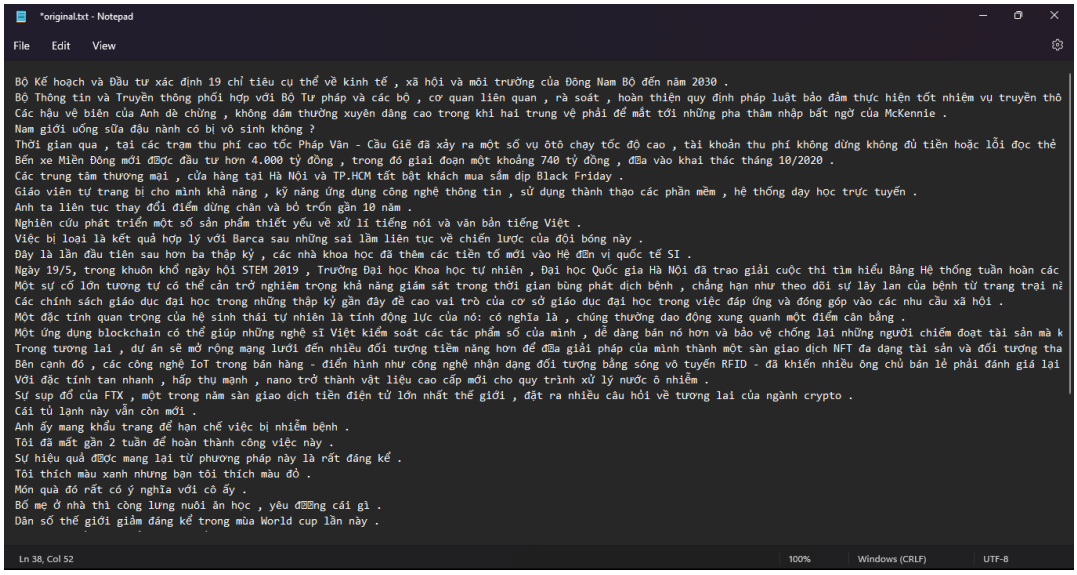
PHẦN 2. THU THẬP DỮ LIỆU

1. Nguồn thu thập

Gồm các câu bất kỳ thuộc nhiều chủ đề được thu thập từ các mục Chính trị xã hội, đời sống, văn hóa, thể thao, sức khỏe trên các bài báo, văn bản và một số câu do nhóm em tự chỉnh sửa tạo thành.

2. Thông tin cơ bản

- Bộ dữ liệu được nhóm đặt tên là *original.txt*:
- Số lượng: 40 câu
- Mỗi dòng là 1 câu
- Các từ phân cách nhau bởi dấu cách
- Kết thúc câu bằng 1 dấu chấm



Hình 2.1. Một số câu có trong bộ dữ liệu gốc

PHẦN 3. TÁCH TỪ

1. Tổng quan về tách từ

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.

Đối với văn bản tiếng Việt, dấu cách được phân biệt giữa các âm tiết chứ không phải giữa các từ. Một từ tiếng Việt có thể được tạo bởi nhiều hơn một âm tiết nên có nhiều cách phân chia các âm tiết thành các từ, gây ra hiện tượng nhập nhằng. Việc giải quyết vấn đề nhập nhằng này gọi là bài toán tách từ. Ví dụ như từ: "chăm sóc" (mang nghĩa chăm lo, bảo bọc) được tạo lên bởi 2 âm tiết là "chăm" và "sóc". Trong khi hai từ đơn "săn" và từ đơn "sóc" lại có thể mang ý nghĩa rất khác so với từ "chăm sóc". 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.

2. Quy trình thực hiện

Nhóm sẽ thực hiện tách từ bán thủ công: sử dụng thư viện Underthesea để tách từ trước sau đó sẽ kiểm tra thủ công kỹ lại kết quả tách từ nghi ngờ là thư viện tách có sự sai sót ở trang VLSP và sửa lại các từ bị sai để thu được 1 bộ dữ liệu mới tốt nhất để so sánh.

Bộ dữ liệu mới được lưu với tên original_tachtu.txt sẽ chứa kết quả tách từ đúng nhất cho 40 câu nhóm mà đã thu thập:

- Các âm tiết của từ ghép sẽ được nối bằng dấu gạch dưới '_'
- Mỗi dòng là 1 từ, dấu câu
- Mỗi câu sẽ được ngăn cách bằng 1 dòng trống
- Số lượng câu: 40
- Số lượng từ: 731
- Số lượng từ ghép: 233

Sau đó, nhóm sẽ sử dụng bộ dữ liệu này làm dữ liệu chuẩn để đánh giá kết quả tách từ của 2 phương pháp: sử dụng Thuật toán Longest Matching và sử dụng thư viện Underthesea.

2.1. Thuật toán Longest Matching

2.1.1. Cách triển khai:

Longest Matching (so khớp từ dài nhất) là thuật toán dựa trên giải thuật tham lam. Với độ phức tạp O (n.V). Với mỗi câu, nó duyệt từ trái qua phải hoặc từ phải qua trái các âm tiết trong câu, rồi kiểm tra xem có nhóm các âm tiết có tồn tại từ trong từ điển hay không. Chuỗi dài nhất các âm tiết được xác định xuất hiện trong từ điển là từ sẽ được tách ra làm một từ. Tiếp tục thực hiện việc so khớp cho đến hết câu.

Ở đồ án này nhóm chúng em tự tạo ra 2 bộ từ điển với bộ từ ghép 2 âm tiết và bộ từ ghép 3 âm tiết đặt tên là tudien2amtiet.txt và tudien3amtiet.txt.

Với vị trí âm tiết hiện tại sẽ kiểm tra xem từ đó và 2 âm tiết tiếp theo có thể ghép thành 1 từ có nghĩa hay không bằng cách kiểm tra trong từ điển tudien3amtiet.txt. 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 tiết tiếp theo có thể ghép được thành một từ có nghĩa hay không bằng cách kiểm tra trong từ điển tudien2amtiet.txt. Thuật toán sẽ dừng khi chúng ta xét hết các tiếng trong câu.

Ví dụ:

Bộ từ điển: Nam_giới, sữa_đậu_nành, vô_sinh

Câu: Nam giới uống sữa đậu nành có bị vô sinh không

Cách tách:

Nam giới uống => nam giới → uống sữa đậu nành có bị vô sinh không

Uống sữa đậu => uống → sữa đậu nành có bị vô sinh không

Sữa đậu nành => sữa đậu nành → có bị vô sinh không

Có bị vô => có → bị vô sinh không

Bị vô sinh => bị → vô sinh không

Vô sinh => vô sinh → không

Kết quả tách:

Nam_giới uống sữa_đậu_nành có bị vô_sinh không

2.1.2. Ưu điểm và nhược điểm:

Ưu điểm:

- Cài đặt đơn giản.
- Độ phức tạp không quá cao.
- Không yêu cầu dữ liệu huấn luyện.

Nhược điểm:

- Không xử lý được các tình huống nhập nhằng nhất định.
- Phụ thuộc nhiều vào bộ từ điển. Các từ không có trong từ điển thì phương pháp này sẽ không tách được.
- Không phân biệt được các trường hợp ghi tắt có dấu "." và các tên riêng.

2.2. Thư viện Underthesea

2.2.1. Giới thiệu chung:

Thư viện Underthesea là một mã nguồn mở bằng Python bao gồm các bộ dữ liệu (data sets) được tạo ra nhằm mục đích hỗ trợ nghiên cứu và phát triển về xử lý ngôn ngữ tự nhiên tiếng Việt bằng cách cung cấp chú thích ngôn ngữ API phong phú, dễ dàng áp dụng các mô hình pretrained NLP cho văn bản tiếng Việt, chẳng hạn như phân đoạn từ, gán thẻ một phần giọng nói (PoS), nhận dạng thực thể có tên (NER) và phân tích cú pháp hay phân loại ngữ liệu.

2.2.2. Ưu điểm và nhược điểm:

Ưu điểm:

- Cài đặt đơn giản, sử dụng thư viện sẵn có.
- Nhận diện được hầu hết các tên riêng có 2 đến 3 âm tiết kể cả tên tiếng anh.
- Phân biệt được các trường hợp ghi tắt có dấu "."

Nhược điểm:

- Không xử lý được các tình huống nhập nhằng nhất định
- Có một số từ còn tách bị sai.

3. Đánh giá kết quả

Với kết quả tách từ của mỗi phương pháp lần lượt được lưu trong các file có tên là longest_matching.txt chứa 774 từ với 222 từ ghép và underthesea.txt chứa 746 từ với 235 từ ghép.

Bảng tổng hợp kết quả tách từ

Phương pháp	**Số từ tách**	**Accuracy**	**Precision**	**Recall**
Longest Matching	774	0.864454	0.882353	0.83691
Underthesea	746	0.984786	0.952991	0.957082

Qua bảng tổng hợp kết quả tách từ, có thể thấy rằng các số liệu đánh giá của việc tách từ khi sử dụng phương pháp sử dụng thuật toán Longest Matching có độ chính xác cao và gần bằng với phương pháp sử dụng thư viện Underthesea vì từ những lời góp ý nhận xét của thầy và các bạn trong buổi thuyết trình về đồ án môn học, nhóm em đã nhận ra được nhiều sai sót trong từ điển và bộ ngữ liệu của mình. Nên nhóm em đã cố gắng rà soát và kiểm tra và chuẩn bị lại bộ từ điển thủ công khá tốt, các trường hợp như tên riêng, từ ghép có 4 âm tiết và các trường hợp ghi tắt có dấu "." thì nhóm không thêm vào trong từ điển. Và phương pháp này chưa giải quyết được các tình huống nhập nhằng trong câu. Nên độ phủ (Recall) của thuật toán này thấp hơn khác nhiều so với Accuracy và Precision.

Phương pháp sử dụng thư viện Underthesea cho ra kết quả rất tốt vì thư viện này giải quyết được hầu hết việc có thể nhận diện được các tên riêng, kể cả tên tiếng anh (nhưng các tên riêng có 4 âm tiết và 1 số tên riêng có 2 âm tiết thư viện vẫn chưa xử lý triệt để được), và phân biệt được các trường hợp ghi tắt có dấu "." Ví dụ: TP.HCM.

4. Các ví dụ phương pháp tách sai điển hình trong bộ ngữ liệu của nhóm

Ví dụ 1:

Tách thủ công	Longest matching	Underthesea
Thời_gian	Thời_gian	Thời_gian
qua	qua	qua
,	,	,
tại	tại	tại
các	các	các
trạm_thu_phí	trạm_thu_phí	trạm
cao_tốc	cao_tốc	thu
Pháp_Vân	Pháp	phí
-	Vân	cao_tốc
Cầu_Giẽ	-	Pháp_Vân
đã	Cầu	-
xảy	Giẽ	Cầu_Giẽ
ra	đã	đã
một_số	xảy_ra	xảy
vụ	một_số	ra
ôtô	vụ	một_số
chạy	ôtô	vụ
tốc_độ	chạy	ôtô
cao	tốc	chạy
,	độ	tốc_độ
tài_khoản	cao	cao
thu	,	,
phí	tài_khoản	tài_khoản
không	thu	thu
dừng	phí	phí
không	không	không
đủ	dừng	dừng
tiền	không	không
hoặc	đủ	đủ
lỗi	tiền	tiền
đọc	hoặc	hoặc
thẻ	lỗi	lỗi
khiến	đọc	đọc
barie	thẻ	thẻ
hạ	khiến	khiến
xuống	barie	barie
gây	hạ	hạ
va_chạm	xuống	xuống
,	gây	gây
vỡ	va_chạm	va_chạm
kính	,	,
xe	vỡ	vỡ
.	kính	kính
	xe	xe
	.	.

Với câu *"Thời gian qua, tại các trạm thu phí cao tốc Pháp Vân - Cầu Giẽ đã xảy ra một số vụ ôtô chạy tốc độ cao, tài khoản thu phí không dừng không đủ tiền hoặc lỗi đọc thẻ khiến barie hạ xuống gây va chạm, vỡ kính xe."* Ta thấy rằng phương pháp Longest Matching không tách được các từ tên riêng như Pháp_Vân, Cầu_Giẻ vì trong từ điển không có các tên riêng địa danh nà, còn Underthesea thì không tách được từ có ba âm tiết là trạm_thu_phí.

Ví dụ 2:

Tách thủ công	Longest matching	Underthesea
Bộ_Kế_hoạch_và_Đầu_tư xác_định 19 chỉ_tiêu cụ_thể về kinh_tế , xã_hội và môi_trường của Đông_Nam_Bộ đến năm 2030 .	Bộ Kế hoạch và Đầu_tư xác_định 19 chỉ_tiêu cụ_thể về kinh_tế , xã_hội và môi_trường của Đông Nam Bộ đến năm 2030 .	Bộ Kế_hoạch và Đầu_tư xác_định 19 chỉ_tiêu cụ_thể về kinh_tế , xã_hội và môi_trường của Đông_Nam_Bộ đến năm 2030 .

Với câu *"Bộ Kế hoạch và Đầu tư xác định 19 chỉ tiêu cụ thể về kinh tế, xã hội và môi trường của Đông Nam Bộ đến năm 2030."* Ta thấy cả hai phương pháp đều không thể tách được cụm từ bộ_kế_hoạch_và_đầu_tư.

Ví dụ 3:

Tách thủ công	Longest matching	Underthesea
Các trung_tâm thương_mại , cửa_hàng tại Hà_Nội và TP.HCM tất_bật khách mua_sắm dịp Black_Friday .	Các trung_tâm thương_mại , cửa hàng tại Hà Nội và TP . HCM tất_bật khách mua_sắm dịp Black Friday .	Các trung_tâm thương_mại , cửa_hàng tại Hà_Nội và TP.HCM tất_bật khách mua_sắm dịp Black_Friday .

Với câu *"Các trung tâm thương mại, cửa hàng tại Hà Nội và TP.HCM tất bật khách mua sắm dịp Black Friday ."* Ta thấy rằng Underthesea có thể nhận biết được các từ đặc biệt của tiếng Anh và có thể tách được từ viết tắt với dấu "." như TP.HCM.

PHẦN 4. TẠO NGỮ LIỆU VÀ GÁN NHÃN

1. Tạo ngữ liệu

1.1. Tập train

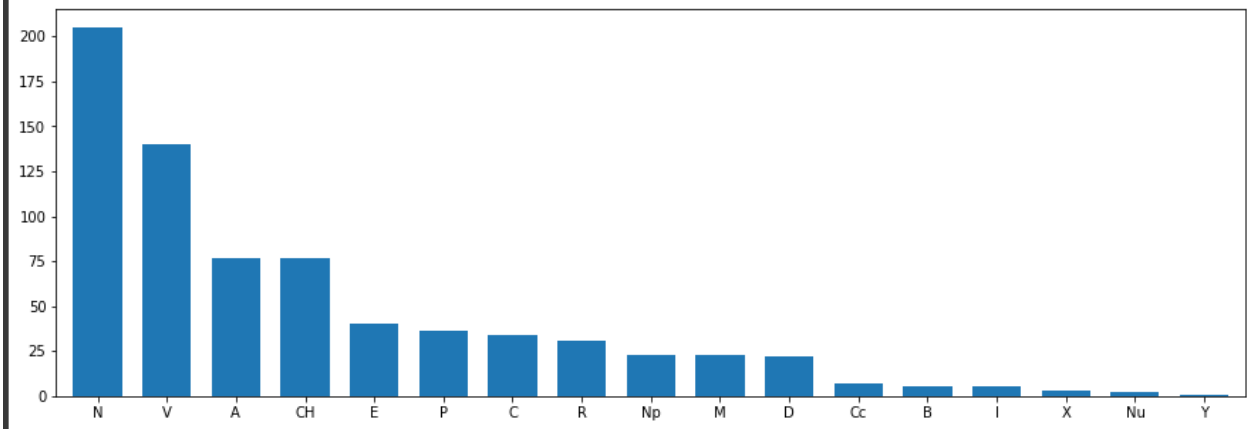
Sau khi có kết quả tách từ, nhóm lại tiếp tục tiến hành gán nhãn thủ công trên bộ dữ liệu tách từ đã chọn.

Chi tiết bộ dữ liệu train được lưu với tên gannhan_th ucong.txt:

- Các âm tiết của từ ghép sẽ được nối bằng dấu gạch dưới '_'.
- Từ được phân cách với nhãn từ loại bằng dấu tab.
- Mỗi dòng là một từ cùng với nhãn của nó. (dấu câu cũng được xem là một từ)
- Các câu được ngăn cách bằng một dòng trống.
- Số lượng câu: 40
- Số lượng từ: 731 từ (tính cả dấu câu)
- Số lượng nhãn xuất hiện: 17 (không tính nhãn bắt đầu câu --s-- mà nhóm thêm vào)

Thống kê các nhãn có trong tập train:

	N	V	A	CH	E	P	C	R	Np	M	D	Cc	B	I	X	Nu	Y	Total
0	205	140	77	77	40	36	34	31	23	23	22	7	5	5	3	2	1	731



Hình 4.1. Các nhãn có trong tập train

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
5	Ni	Danh từ ký hiệu	A1, A4, 60A, 60B, 20a, 20b, ABC
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, đại
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, ngoài
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	. ! ? , ; :

Hình 4.2. Bảng các nhãn từ loại

Nhóm sẽ sử dụng 40 câu này để làm tập Train (tập train đã nêu chi tiết ở trên) cho việc gán nhãn từ loại và tạo thêm tập Test 10 câu.

Nhóm tạo một từ điển gồm tập các từ có trong các câu của tập Train và sử dụng nó để tạo các ma trận. Có thêm một từ là “unk” để chỉ chung các từ không nằm trong từ điển.

Các từ được sử dụng trong tập Test hầu hết sẽ nằm trong tập từ điển, chỉ có vài từ nằm ngoài để kiểm tra khả năng gán nhãn cho những từ nằm ngoài tập từ điển. Những từ này sẽ được quy ước chung là “unk”.

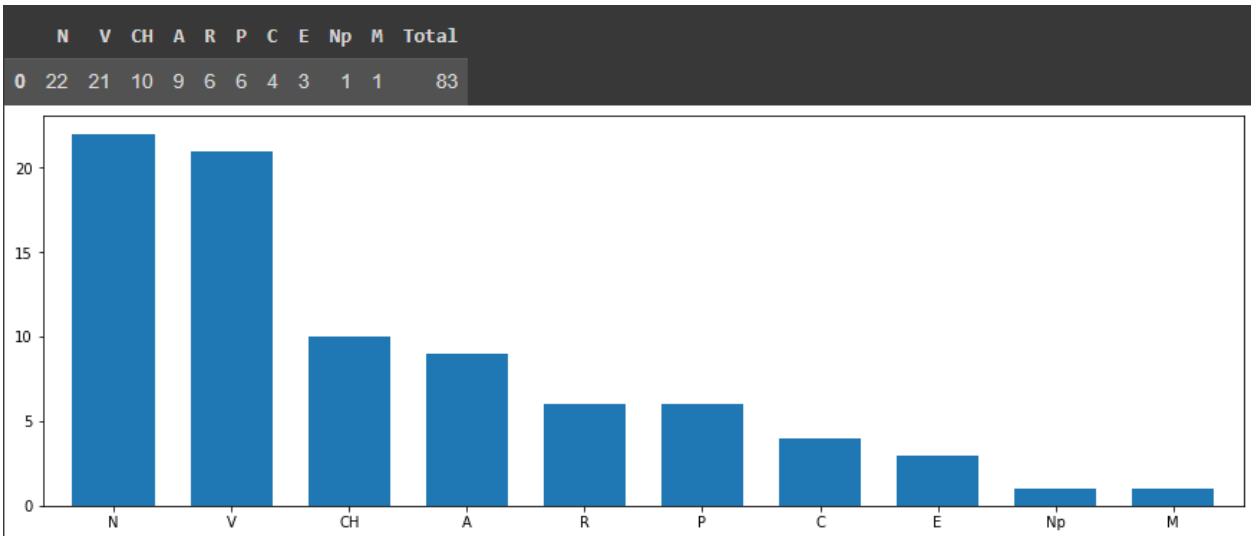
1.2. Tập test

Tập Test gồm các file test_khongunk_gannhan.txt và test_khongunk.txt:

- test_khongunk_gannhan.txt: chứa các từ đã được gán nhãn thủ công để thực hiện cho việc kiểm tra độ chính xác khi gán nhãn

- test_khongunk.txt: chỉ chứa từ của các câu để thực hiện việc gán nhãn.
- Số lượng câu của tập Test: 10
- Số lượng từ của tập Test: 83 (kể cả dấu câu)

Thống kê các nhãn có trong tập test:



Hình 4.3. Các nhãn có trong tập test (file test_khongunk_gannhan)

2. Gán nhãn từ loại

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, thêm nhãn "--s--" để chỉ ra phần bắt đầu của mỗi câu.

Từ điển Transition (Transition[(i,j)]): tính số lần nhãn i chuyển sang nhãn j. Từ điển này giúp tạo ma trận chuyển tiếp.

```
[ ]  Transition

     defaultdict(int,
                 {('--s--', 'Np'): 2,
                  ('Np', 'V'): 6,
                  ('V', 'M'): 3,
                  ('M', 'N'): 14,
                  ('N', 'A'): 27,
                  ('A', 'E'): 8,
                  ('E', 'N'): 20,
                  ('N', 'CH'): 25,
                  ('CH', 'N'): 9,
                  ('N', 'Cc'): 3,
                  ('Cc', 'N'): 2,
                  ('N', 'E'): 16,
                  ('E', 'Np'): 5,
                  ('V', 'N'): 45,
                  ('N', 'M'): 6,
                  ('M', 'CH'): 2,
                  ('V', 'C'): 5,
```

Hình 4.4. Một phần từ điển Transition

Từ điển emis (emis[(i,j)]): tính số lần từ i được gán nhãn j. Từ điển này giúp tạo ma trận phát xạ.

```
[ ]  emis

     defaultdict(int,
                 {('bộ_kế_hoạch_và_đầu_tư', 'Np'): 1,
                  ('xác_định', 'V'): 1,
                  ('19', 'M'): 1,
                  ('chỉ_tiêu', 'N'): 1,
                  ('cụ_thể', 'A'): 1,
                  ('về', 'E'): 4,
                  ('kinh_tế', 'N'): 1,
                  (',', 'CH'): 33,
                  ('xã_hội', 'N'): 2,
                  ('và', 'Cc'): 7,
                  ('môi_trường', 'N'): 1,
                  ('của', 'E'): 15,
                  ('đông_nam_bộ', 'Np'): 1,
                  ('đến', 'V'): 1,
                  ('năm', 'N'): 2,
                  ('2030', 'M'): 1,
```

Hình 4.5. Một phần từ điển emis

Từ điển Tag counts (tagd[i]): đếm số lần mỗi nhãn xuất hiện.

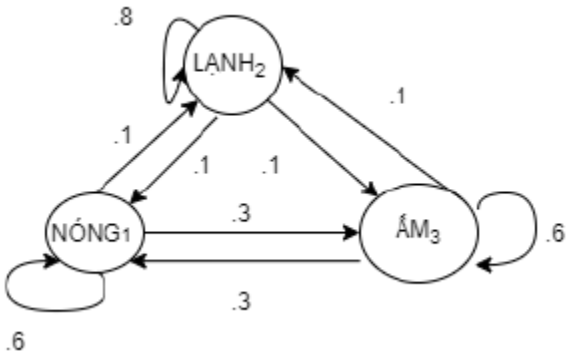
```
[ ]  tagd

     defaultdict(int,
                 {'--s--': 40,
                  'Np': 23,
                  'V': 140,
                  'M': 23,
                  'N': 205,
                  'A': 77,
                  'E': 40,
                  'CH': 77,
                  'Cc': 7,
                  'C': 34,
                  'D': 22,
                  'R': 31,
                  'B': 5,
                  'Nu': 2,
                  'P': 36,
                  'X': 3,
                  'Y': 1,
                  'I': 5})
```

Hình 4.6. Từ điển Tag counts

3. Mô hình Hidden Markov (HMM)

3.1. Markov Chain

Markov Chain (Xích Markov), hay Visible Markov Model, là một dạng FSA được dùng để mô hình hóa xác suất của các biến ngẫu nhiên có quan hệ với nhau theo dạng chuỗi.



Hình 4.7. Ví dụ về 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.

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

- $Q = q1, q2, \ldots qn$: là tập các trạng thái.
- δ là ma trận xác suất chuyển trạng thái từ qi sang qj. Xác suất này được tính theo mô hình bậc 1 (theo giả thiết p (xn|x1x2…xn-1) = p(xn|xn-1) như sau:

$$\delta_{q_i q_j} = \frac{count(q_i, q_j)}{count(q_i)}$$

Trong đó count(qi,qj) là số lần chuyển từ qi sang qj và count(qi) là số lần xuất hiện của qi

- q0 là một ký hiệu khởi đầu của tất cả các chuỗi.

3.1.1. Tổng quan về HMM

HMM (Hidden Markov Models) là một trong những thuật toán được sử dụng phổ biến nhất trong Xử lý ngôn ngữ tự nhiên dựa trên Markov chain 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,...

Các thành phần:

- S= {s1, s2, …, sn} là tập các trạng thái ẩn
- S0 là trạng thái bắt đầu
- K={k1,k2,…,kn} là tập các giá trị quan sát (chính là các từ trong câu)
- Ma trận A là ma trận chuyển trạng thái chứa xác suất chuyển từ trạng thái này sang trạng thái khác (chính là các nhãn từ loại)
- Ma trận B là ma trận phát xạ chứa xác suất trạng thái ẩn được thể hiện bởi giá trị quan sát (xác suất một từ có thể được gán nhãn nào đó)

Dựa vào các từ điển Transition, emis and Tag_counts đã được tạo như đã nói ở trên, nhóm sẽ triển khai mô hình Hidden Markov. Ta bắt đầu xây dựng các ma trận chuyển tiếp (Transition Matrix) và ma trận phát xạ (Emission Matrix) với tham số smoothing.

3.1.2. Ma trận chuyển tiếp MatrixA

	A	B	C	CH	Cc	D	E	I	M	N	Np	Nu	P
--s--	0.017544	0.017544	0.052632	0.017544	0.017544	0.087719	0.052632	0.017544	0.087719	0.333333	0.052632	0.017544	0.140351
A	0.074468	0.010638	0.095745	0.191489	0.021277	0.021277	0.095745	0.021277	0.053191	0.180851	0.010638	0.010638	0.021277
B	0.090909	0.045455	0.045455	0.136364	0.045455	0.045455	0.045455	0.045455	0.045455	0.045455	0.045455	0.045455	0.045455
C	0.098039	0.019608	0.019608	0.039216	0.019608	0.058824	0.019608	0.019608	0.039216	0.235294	0.058824	0.019608	0.078431
CH	0.053191	0.010638	0.021277	0.010638	0.010638	0.031915	0.042553	0.010638	0.021277	0.106383	0.042553	0.010638	0.031915
Cc	0.041667	0.041667	0.041667	0.041667	0.041667	0.083333	0.041667	0.041667	0.041667	0.125000	0.083333	0.041667	0.041667
D	0.025641	0.025641	0.025641	0.025641	0.025641	0.025641	0.025641	0.025641	0.025641	0.589744	0.025641	0.025641	0.025641
E	0.035088	0.017544	0.035088	0.017544	0.017544	0.087719	0.017544	0.017544	0.017544	0.368421	0.105263	0.017544	0.140351
I	0.045455	0.045455	0.045455	0.181818	0.045455	0.045455	0.045455	0.090909	0.045455	0.045455	0.045455	0.045455	0.045455
M	0.025000	0.025000	0.050000	0.075000	0.025000	0.025000	0.025000	0.025000	0.100000	0.375000	0.025000	0.075000	0.025000
N	0.126126	0.018018	0.049550	0.117117	0.018018	0.009009	0.076577	0.013514	0.031532	0.189189	0.045045	0.004505	0.054054
Np	0.075000	0.025000	0.050000	0.200000	0.075000	0.025000	0.050000	0.025000	0.025000	0.075000	0.025000	0.025000	0.025000
Nu	0.052632	0.052632	0.052632	0.157895	0.052632	0.052632	0.052632	0.052632	0.052632	0.052632	0.052632	0.052632	0.052632

Hình 4.8. Một phần của ma trận chuyển tiếp (Transition Matrix A)

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.

Thực hiện smoothing theo công thức như sau:

$$A_{ij} = \frac{count(s_i, s_j) + \alpha}{count(s_i) + \alpha * N}$$

- N: Tổng số nhãn không kể nhãn bắt đầu
- $count(si,sj)$: số lượng bộ giá trị (prev_tag, tag) trong từ điển Transition.
- c$ount(si)$: số lượng của nhãn trước trong từ điển tag_d (như trên)
- α: là tham số smoothing.

Đồng thời khi ma trận trên nhóm em cũng tạo từ điển transsub (transsub[(i,j)]) để tính xác suất từ nhãn i chuyển sang nhãn j. Từ điển này được tạo ra để thuận tiện trong việc áp dụng thuật toán Viterbi.

```
[ ]  transsub

     defaultdict(int,
                 {('--s--', 'A'): 0.017543859649122806,
                  ('--s--', 'B'): 0.017543859649122806,
                  ('--s--', 'C'): 0.05263157894736842,
                  ('--s--', 'CH'): 0.017543859649122806,
                  ('--s--', 'Cc'): 0.017543859649122806,
                  ('--s--', 'D'): 0.08771929824561403,
                  ('--s--', 'E'): 0.05263157894736842,
                  ('--s--', 'I'): 0.017543859649122806,
                  ('--s--', 'M'): 0.08771929824561403,
                  ('--s--', 'N'): 0.3333333333333333,
                  ('--s--', 'Np'): 0.05263157894736842,
                  ('--s--', 'Nu'): 0.017543859649122806,
                  ('--s--', 'P'): 0.14035087719298245,
                  ('--s--', 'R'): 0.017543859649122806,
                  ('--s--', 'V'): 0.03508771929824561,
                  ('--s--', 'X'): 0.017543859649122806,
                  ('--s--', 'Y'): 0.017543859649122806,
                  ('A', 'A'): 0.07446808510638298,
```

Hình 4.9. Một phần từ điển transsub

3.1.3. Từ điển phát xạ MatrixB

	,	-	.	10	10/2020	19	19/5	2	2019	2030	...	đó	đối_tượng	đồng	đội
A	0.001992	0.001992	0.001992	0.001992	0.001992	0.001992	0.001992	0.001992	0.001992	0.001992	...	0.003984	0.001992	0.001992	0.001992
B	0.002326	0.002326	0.002326	0.002326	0.002326	0.002326	0.002326	0.002326	0.002326	0.002326	...	0.002326	0.004651	0.002326	0.002326
C	0.002179	0.002179	0.002179	0.002179	0.002179	0.002179	0.002179	0.002179	0.002179	0.002179	...	0.002179	0.002179	0.002179	0.002179
CH	0.067729	0.007968	0.079681	0.001992	0.001992	0.001992	0.001992	0.001992	0.001992	0.001992	...	0.001992	0.001992	0.001992	0.001992
Cc	0.002315	0.002315	0.002315	0.002315	0.002315	0.002315	0.002315	0.002315	0.002315	0.002315	...	0.002315	0.002315	0.002315	0.002315
D	0.002237	0.002237	0.002237	0.002237	0.002237	0.002237	0.002237	0.002237	0.002237	0.002237	...	0.002237	0.002237	0.002237	0.002237
E	0.002151	0.002151	0.002151	0.002151	0.002151	0.002151	0.002151	0.002151	0.002151	0.002151	...	0.002151	0.002151	0.002151	0.002151
I	0.002326	0.002326	0.002326	0.002326	0.002326	0.002326	0.002326	0.002326	0.002326	0.002326	...	0.002326	0.002326	0.002326	0.002326
M	0.002232	0.002232	0.002232	0.004464	0.002232	0.004464	0.002232	0.002232	0.004464	0.004464	...	0.002232	0.002232	0.002232	0.002232
N	0.001587	0.001587	0.001587	0.001587	0.001587	0.001587	0.001587	0.003175	0.001587	0.001587	...	0.001587	0.004762	0.001587	0.003175
Np	0.002232	0.002232	0.002232	0.002232	0.002232	0.002232	0.002232	0.002232	0.002232	0.002232	...	0.002232	0.002232	0.002232	0.002232
Nu	0.002342	0.002342	0.002342	0.002342	0.002342	0.002342	0.002342	0.002342	0.002342	0.002342	...	0.002342	0.002342	0.007026	0.002342

Hình 4.10. Một phần ma trận của ma trận phát xạ (Emission Matrix B)

Ma trận B được tính toán smoothing theo công thức:

$$B_{ij} = \frac{count(s_i, k_j) + \alpha}{count(s_i) + \alpha * N}$$

- N: số từ ở trong tập từ điển (tập từ điển gồm các từ của các câu trong tập train có thêm từ ‘unk’)
- $count(si,kj)$: số lần nhãn si được thể hiện bởi từ kj
- $count(si)$: tổng số nhãn si

- α: là tham số smoothing.

Đồng thời khi tạo ma trận trên, nhóm em cũng tạo từ điển emissub (emissub[(i,j)]: tính xác suất để từ i được gán nhãn j. Từ điển này được tạo để thuận tiện trong việc sử dụng trong thuật toán Viterbi.

3.2. Thuật toán Viterbi

Nhóm tiến hành kết hợp mô hình Hidden Markov đã có với thuật toán Viterbi. 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:

1. Bước khởi tạo: khởi tạo cột đầu của hai ma trận C và D (nhóm em sử dụng từ điển để dễ tính toán C[(i,j)], D[(i,j)]). Ma trận C để lưu xác suất các đường đi tốt nhất và ma trận D để lưu các giá trị dẫn tới các đường đi tốt nhất đó:
    a. Cột đầu tiên của ma trận C lưu xác suất đi từ từng nhãn i tới từ đầu tiên trong câu
    b. Cột đầu tiên của ma trận C lưu xác suất đi từ từng nhãn i tới từ đầu tiên trong câu
2. Bước forward: Ở mỗi bước, tính toán xác suất xảy ra ở các đường đi và các giá trị dẫn tới đường đi tốt nhất tới điểm đó
    a. Mỗi ô trong ma trận C được tính bởi công thức:

$$C_{i,j} = max(c_{k,j-1} * a_{k,i} * b_{i,cindex(w_j)})$$

- Với $C_{i,j}$ là xác suất của đường đi trước đã đi qua
- $a_{k,i}$ là xác suất chuyển từ thẻ k sang thẻ i (Lấy được thông qua từ điển transsub tạo ở trên)
- $b_{i,cindex(w_j)}$ là xác suất để từ j mang nhãn i

    b. Mỗi ô trong ma trận C được tính bởi công thức:

$$d_{i,j} = argmax(c_{k,j-1} * a_{k,i} * b_{i,cindex(w_j)})$$

Tức là lưu chỉ số k giúp $C_{i,j}$ đạt cực đại

3. Bước backward: Tìm ra đường đi tốt nhất với xác suất cao nhất
    a. Tìm ra giá trị xác suất cao nhất ở cột cuối cùng trong ma trận C
    b. Truy vết ngược từ xác suất cao nhất của đường đi đến từ cuối cùng để tìm ra đường đi tốt nhất mà ta có được ở trên, từ đây ta truy vết thông qua D.

Ví dụ:

$C =$

	w_1	w_2	w_3	w_4	w_5
t_1	0.25	0.125	0.025	0.0125	0.01
t_2	0.1	0.025	0.05	0.01	0.003
t_3	0.3	0.05	0.025	0.02	0.0000
t_4	0.2	0.1	0.000	0.0025	0.0003

$D =$

	w_1	w_2	w_3	w_4	w_5
t_1	0	1	3	2	3
t_2	0	2	4	1	3
t_3	0	2	4	1	4
t_4	0	4	4	3	1

Ta thấy C[1,5] có xác suất cao nhất và trong ma trận D thì D[1,5] lưu chỉ số 3. Tức là để đạt được C[1,5] thì phải đi từ C[3,4]. Truy vết ngược dần ta được kết quả gán nhãn như sau:

W1/t2 W2/t3 W3/t1 W4/t3 W5/t1

Với W là từ và t là nhãn

4. 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 kết quả khi sử dụng thư viện Underthesea.

```
[['việt_nam'], ['đang'], ['thực_hiện'], ['nhiệm_vụ'], ['hoàn_thiện'], ['việc'], ['thay_đổi'], ['chính_sách'], ['.']]
P R V N V N V N CH
[['mở_rộng'], ['mạng_lưới'], ['giao_dịch'], ['khẩu_trang'], ['là'], ['rất'], ['tiềm_năng'], ['.']]
D N V N C R A CH
[['phương_pháp'], ['ấy'], ['mang'], ['lại'], ['hiệu_quả'], ['đáng_kể'], ['với'], ['thế_giới'], ['.']]
N P V V N A C N CH
[['cổ_động_viên'], ['từ_chối'], ['nói_chuyện'], ['với'], ['nhà_khoa_học'], ['.']]
N V A C N CH
[['hiện_thực'], ['dần'], ['được'], ['hoàn_thành'], ['qua'], ['cái'], ['tủ_lạnh'], ['.']]
N V V V V N N CH
[['màu'], ['xanh'], ['và'], ['màu0'], ['đỏ'], ['rất'], ['đẹp'], ['.']]
N A C N A R A CH
[['đối_tượng'], ['nhiễm_bệnh'], ['đưa'], ['ra'], ['giải_pháp'], ['cho'], ['mình'], ['.']]
N V V V N V N CH
[['tương_lai'], ['bán_lẻ'], ['vật_liệu'], ['cao_cấp'], ['đã'], ['được'], ['hiện_thực'], ['.']]
N V N A R V N CH
[['câu_đố'], ['unk'], ['này'], ['làm'], ['tôi'], ['unk0'], ['unk1'], ['2'], ['năm'], ['unk2'], ['.']]
N E P V V N V N N A CH
```

Hình 4.11. Kết quả gán nhãn một số câu trong tập test sử dụng Hidden Markov và Viterbi

Model	Độ chính xác
HMM (train)	0.841313
HMM (test)	0.710843
Underthesea	0.783132

Theo như ta thấy thì khi gán nhãn bằng mô hình Hidden Markov kết hợp thuật toán Viterbi có độ chính xác tạm chấp nhận được, điều này xảy ra một phần vì tập train của bọn em không đủ độ lớn và một phần vì các từ trong tập test của bọn em phần lớn nằm trong tập train.

Khi trong câu nếu như có những từ không có trong từ điển tạo từ tập train thì tỉ lệ gán nhãn sai sẽ tăng lên.

Những trường hợp sai cũng có thể đến từ sự chênh lệch giữa số lượng các nhãn quá lớn, điều này sẽ khiến các ma trận có những giá trị không tối ưu và lúc gán sẽ bị sai.

Độ chính xác khi sử dụng Viterbi ở tập train là 0.841313. Nó khá cao do hiện tượng overfitting.

PHẦN 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết luận

Mô hình HMM cho độ chính xác kém nhất trên cả tập test và tập train, điều này có thể là do giả thuyết Markov khi trạng thái hiện tại chỉ phụ thuộc duy nhất vào trạng thái trước đó, về thực tế thì điều này chưa chắc đúng do có nhiều cụm danh từ, động từ đi liền nhau.

Để cải thiện độ chính xác thì cần phải mở rộng bộ từ điển để tránh trường hợp những từ không có trong từ điển và điều chỉnh sự chênh lệch giữa các nhãn từ loại.

2. Hướng phát triển:

Tìm hiểu về một số phương pháp gán nhãn khác và các thuật toán hỗ trợ.

Áp dụng các từ đã được gán nhãn vào mô hình machine learning để thử nghiệm.

Xây dựng một app tách từ và gán nhãn từ loại trong tương lai.

Xây dựng bộ dữ liệu lớn và đa dạng hơn nhằm tránh hiện tượng overfitting và cải thiện độ chính xác.

BẢNG PHÂN CÔNG CÔNG VIỆC

	Tên thành viên/ Mức độ hoàn thành	
Nhiệm vụ	**Bùi Khánh Duy**	**Đinh Hoàng Lộc**
Tạo bộ ngữ liệu	100%	100%
Tìm hiểu đề tài bài toán tách từ	90%	100%
Thực hiện bài toán tách từ	95%	100%
Tìm hiểu đề tài bài toán gán nhãn từ loại	100%	100%
Thực hiện bài toán gán nhãn từ loại	100%	90%
Thực hiện code	100%	95%
Làm slide và báo cáo	90%	100%
Viết báo cáo đồ án	100%	100%

TÀI LIỆU THAM KHẢO

1. Slide bài giảng xử lý ngôn ngữ tự nhiên (Trường Đại học Công nghệ thông tin)
2. https://github.com/ds4v/vietnamese-pos-tagging
3. http://www.adeveloperdiary.com/data-science/machine-learning/implement-viterbi-algorithm-in-hidden-markov-model-using-python-and-r/
4. Viterbi Hidden Markov Models book by Angela B. Shiflet *University "Magna Græcia" of Catanzaro, Catanzaro, Italy*