TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công Nghệ Thông Tin và Truyền Thông

BÁO CÁO BÀI TẬP LỚN

TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI : NHẬN DẠNG THỰC THỂ CÓ TÊN SỬ DỤNG TRƯỜNG NGẪU NHIÊN CÓ ĐIỀU KIỆN

Giảng viên hướng dẫn: *TS. Nguyễn Nhật Quang*

Sinh viên tham gia: *Vũ Tiến Đạt*

Trần Hữu Trí

Lê Tuấn Thành

Nguyễn Thị Hoài

Hà Nội, 2018

STT	*Họ và tên*	*MSSV*	*Email*
1	*Vũ Tiến Đạt*	*20160975*	*vudat1710@gmail.com*
2	*Lê Tuấn Thành*	*20163705*	*thanhlt998@gmail.com*
3	*Trần Hữu Trí*	*20164306*	*tritranhuu123@gmail.com*
4	*Nguyễn Thị Hoài*	*20161626*	*hoaint210903@gmail.com*

Các công việc chính:

Nội dung	**Người thực hiện**
Đề xuất ý tưởng đề tài	Vũ Tiến Đạt
Nghiên cứu các tài liệu, kiến thức liên quan đến đề tài: NER, HMM, CRF, các công cụ.	Vũ Tiến Đạt, Trần Hữu Trí, Lê Tuấn Thành, Nguyễn Thị Hoài
Viết báo cáo về HMM, learning weight, gradient descent	Vũ Tiến Đạt
Viết báo cáo về CRF, viterbi	Lê Tuấn Thành
Viết báo cáo về công cụ, tổng hợp kết quả	Trần Hữu Trí
Tổng hợp báo cáo, làm slide	Nguyễn Thị Hoài
Viết code, hiệu chỉnh code	Vũ Tiến Đạt, Trần Hữu Trí, Lê Tuấn Thành, Nguyễn Thị Hoài

MỤC LỤC

I. Giới thiệu bài toán

1. Giới thiệu bài toán nhận dạng thực thể có tên (NER)

Trong lĩnh vực Trí tuệ nhân tạo cũng như Học máy, các nghiên cứu về xử lí ngôn ngữ tự nhiên (NLP – Natual Language Processing) luôn đem lại nhiều ứng dụng thiết thực trong cuộc sống. Chúng ta có thể sử dụng hệ thống nhận diện từ vào trong các mô hình thương mại điện tử để có thể tự động tư vấn cho khách hàng, hay sử dụng nó vào nền công nghiệp sản xuất robot.

Bài toán nhận dạng thực thể có tên (NER- Named Entity Recognition) mà một trong những đề tài nghiên cứu nổi bật trong lĩnh vực xử lý ngôn ngữ tự nhiên nói riêng cũng như học máy nói chung. NER có công việc tìm và phân loại các từ đặc biệt trong một văn bản vào những thể loại tương ứng với nó như person, location, organization, … Trong thực tế, NER được ứng dụng vô cùng rộng rãi và phổ biến. Ta có thể kể đến như ứng dụng vào hệ thống hỗ trợ khách hàng. NER sẽ nhận dạng những yếu tố trong phản hồi hay phàn nàn của khách hàng như thông số của sản phẩm, thông tin nhà cung cấp, công ty,.. từ đó mà phản hồi có thể được gửi tới nhà cung cấp có trách nhiệm cho sản phẩm một cách nhanh chóng, tự động. Hay ứng dụng NER cho việc tóm tắt hồ sơ các bệnh nhân. Với thời kỳ phát triển mạnh mẽ của Công nghệ thông tin như bây giờ, mô hình việc làm online không phải là hiếm, và có hàng ngàn hồ sơ xin việc gửi tới các nhà tuyển dụng. NER sẽ giúp họ trong việc trích xuất thông tin cần thiết của ứng cử viên.

Nhiều tập đoàn lớn đang theo đuổi việc xây dựng hệ thống NER cho họ. Có rất nhiều thuật toán cũng như các công cụ hỗ trợ cho việc xây dựng một hệ thống NER cho các ứng dụng khác nhau.

Do tính ứng dụng, tính cần thiết cùng sự phổ biến của NER. Chúng em quyết định chọn nó làm đề tài nghiên cứu cho bài tập lớn môn Trí tuệ nhân tạo.

2. Chi tiết bài toán

NER là phân loại các thực thể có tên trong văn bản. Ta sẽ phải chuyển đổi từ thực thể thành chuỗi các nhãn, các nhãn này có tác dụng phân loại thực thể đó là người hay địa điểm hay bất cứ thứ gì. Trong thực tế, tên thực thể có thể viết dưới dạng một cụm từ, và một nhãn chỉ có thế gán cho một từ. Để giải quyết vấn đề này, chúng em sử dụng các nhãn B-, I- và O. Nhãn B- sẽ đại diện cho từ đầu tiên trong cụm tên thực thể, nhãn I- sẽ cho các từ còn lại, và nhãn O sẽ được gán cho những từ không thuộc tên thực thể. Bài toán NER chúng em giải quyết sẽ phân loại cho 4

thực thể : người (PER), tổ chức (ORG), địa điểm (LOC), và các loại khác (MISC). Như vậy sẽ có 9 nhãn : O, B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG, B-MISC, I-MISC. Ví dụ với câu "Hồng Quang, phóng viên VTV tại Bỉ", gán nhãn NER sẽ trả về kết quả sau:

Hồng Quang	phóng viên	VTV	tại	Bỉ
B-PER	O	B-ORG	O	B-LOC

Tuy nhiên việc từ một câu trên đi đến nhãn NER phải trải qua nhiều quá trình: tách từ, gán nhãn PoS (part of speech), … để đưa một văn bản thuần về định dạng dữ liệu khác.

Trong đồ án này, khi xây dựng hệ thống học máy, chúng em sẽ sử dụng train data và test data có định dạng như sau: văn bản ban đầu sẽ được phân tích thành nhiều dòng, các dòng sẽ gồm 4 phần: nội dung từ, nhãn PoS, nhãn chunk và nhãn NER (các nhãn PoS và chunk được nêu ra ở mục IV).

Và để xây dựng mô hình ứng dụng, với đầu vào là một văn bản trơn, chúng em sẽ dùng các công cụ có sẵn cho bước tách từ và sử dụng hệ thống gán nhãn NER đã xây dựng để gán nhãn NER và sinh ra output là văn bản đầu vào đã được gán nhãn cho thực thể (công cụ chúng em sử dụng được nêu ở mục IV).

3. Các phương pháp tiếp cận

Các hệ thống NER hiện tại sử dụng các kĩ thuật dựa vào ngữ pháp trong ngôn ngữ học cũng như các mô hình xác suất như học máy. Việc nhận dạng trực tiếp bằng năng lực của các chuyên gia ngôn ngữ có kinh nghiệm thường cho độ chính xác cao hơn thế nhưng lại yêu cầu rất nhiều thời gian, công sức, tiền của của các tổ chức cũng như doanh nghiệp. Các hệ thống NER thống kê thường yêu cầu 1 lượng lớn dữ liệu huấn luyện được chú thích bằng tay (thường được thực hiện bởi các cộng tác viên hoặc các chuyên gia thống kê ngôn ngữ). Hiện nay có rất nhiều loại hệ thống phân loại mô phỏng và thực hiện công việc nhận dạng thực thể có tên, và trường điều kiện ngẫu nhiên (CRF) là một trong những phương pháp phổ biến nhất dùng để tiếp cận bài toán này.

Tài liệu này sẽ đưa ra những khái niệm cơ bản về NER, trường điều kiện ngẫu nhiên (CRFs - Conditional Random Fields), và ứng dụng kiến thức thu được để xây dựng một mô hình nhận dạng thực thể có tên sử dụng thư viện mạnh mẽ hỗ trợ trường điều kiện ngẫu nhiên là CRFsuite.

II. Mô hình Markov ẩn (HMM)

Trước khi giới thiệu chi tiết về trường điều kiện ngẫu nhiên, nhóm chúng em xin trình bày 1 mô hình cũng rất phổ biến trong bài toán nhận dạng thực thể nói riêng hay các bài toán nhận dạng từ nói chung (Ví dụ: Part of Speech Tagging) đó là mô hình Markov ẩn (HMM- Hidden Markov Model). Lý thuyết về mô hình Markov ẩn là vô cùng quan trọng giúp chung ta có cảm quan đầu tiên về bài toán nhận dạng từ trong xử lý ngôn ngữ tự nhiên, qua đó dễ dàng tiếp cận hơn tới phương pháp sử dụng trường điều kiện ngẫu nhiên.

1. Giới thiệu về HMM

Một trong những phương pháp tiếp cận phổ biến nhất trong bài toán gán nhãn từ trong văn bản là phương pháp sử dụng mô hình Markov ẩn (hay gọi tắt là HMM). HMM là một mô hình chuỗi có nhiệm vụ gán một nhãn hoặc một lớp cho mỗi phần tử của chuỗi, sau đó thực hiện gán một chuỗi các nhãn cho một chuỗi quan sát. Một HMM là một mô hình chuỗi xác suất; cho trước một chuỗi các phần tử (từ, kí tự, chữ cái, câu, …), mô hình sẽ tính toán một phân bố xác suất trên tất cả các chuỗi nhãn có thể và lựa chọn ra chuỗi nhãn tốt nhất để gán cho chuỗi quan sát.

Một mô hình Markov ẩn cho phép ta xem xét cả sự kiện quan sát được (ví dụ các từ ta sử dụng là chuỗi đầu vào) và cả các sự kiện ẩn (ví dụ những nhãn ta cần gán cho chuỗi đầu vào). Một HMM bao gồm các thành phần sau:

$Q = q_1 q_2 \dots q_N$	Một tập gồm N trạng thái
$A = a_{11} \dots a_{ij} \dots a_{NN}$	Ma trận chuyển trạng thái A: a_{ij} thể hiện xác suất khi chuyển từ trạng thái i sang trạng thái j
$O = o_1 o_2 \dots o_T$	Chuỗi quan sát gồm T phần tử
$B = \{ b_j(k) \}$ $b_j(k) = p(v_k = O_t \mid q_t = S_j)$, $1 \le j \le N, 1 \le k \le M$	Ma trận xác suất quan sát, mỗi phần tử biểu diễn xác suất của một phần tử O_t được sinh ra từ trạng thái i
$\pi = \{\pi_i\}$	Xác suất khởi đầu của mỗi trạng thái với $\pi_i = p(q_1 = S_i), 1 \le i \le N$ thỏa mãn ràng buộc $\sum_{j=1}^{N} \pi_i = 1$

HMM bậc nhất được thực hiện dựa trên 2 giả sử sau:

- Xác suất của một trạng thái cụ thể chỉ phụ thuộc vào trạng thái trước đó

Markov Assumption: $P(q_i | q_1 \dots q_{i-1}) = P(q_i | q_{i-1})$

- Xác suất đầu ra của một dữ liệu quan sát o_i trong chỉ phụ thuộc vào trạng thái q_i và không phụ thuộc vào bất kì trạng thái hay giá trị quan sát nào khác.

Output Independence: $P(o_i|q_1 \dots, q_i, \dots, q_T, o_1, \dots, o_i, \dots, o_T) = P(o_i|q_i)$

2. Phần tử của mô hình Markov ẩn

Một HMM gồm 2 thành phần cơ bản là 2 ma trận A và B. A là ma trận xác suất chuyển trạng thái $P(t_i|t_{i-1})$ thể hiện xác suất xảy ra một nhãn nếu biết nhãn trước đó. Trong phần này, chúng em xin phép lấy ví dụ về bài toán gán nhãn từ loại (PoS tagging) trong tiếng Anh để ví dụ có thể dễ hiểu nhất, một động từ dạng nguyên thể (Verb base form - VB) thường xuất hiện ngay sau một động từ khuyết thiếu (Model Verb - MD). Do đó, xác suất chuyển trạng thái mong đợi trong tình huống này thường là cao. Ta sẽ tính toán xác suất này bằng phương pháp đếm trong tập dữ liệu đã gán nhãn và công thức:

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

Qua thực nghiệm, giả sử MD xuất hiện 100000 lần và ngay sau chúng có 90000 lần VB xuất hiện, qua đó xác suất thu được là 9/10 hay 0.9.

Như chúng em đã đề cập tới ở trên, trong một mô hình Markov ẩn, các xác suất cần tính được ước lượng bằng cách đếm trong tập dữ liệu đã gán nhãn (giả sử tập dữ liệu sử dụng là WSJ corpus).

Ma trận xác suất quan sát B: $P(w_i|t_i)$ là ma trận thể hiện xác suất mà tại đó, nếu ta đã biết trước một tag (giả sử là MD), thì nó có thể gán được cho từ cho trước nào (giả sử will). Xác suất này được xác định bởi công thức:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

Ví dụ: $P(will|MD)$ = 5000/10000 = 0.5. Giá trị này không trả lời cho câu hỏi đâu là tag khả dĩ nhất dành cho từ will. Câu hỏi này được giải đáp qua biểu thức $P(MD|will)$. Thay vào đó, $P(will|MD)$ có thể trả lời cho một câu hỏi khác: "Nếu ta sinh ra một động từ khuyết thiếu, xác suất động từ khuyết thiếu đó là từ will là bao nhiêu?"

3. Decoding

Decoding là quá trình tìm ra chuỗi các giá trị ẩn tương ứng với chuỗi quan sát: cho đầu vào của HMM là λ = (A, B), và một chuỗi quan sát O (đã định nghĩa ở phần trước), tìm chuỗi các trạng thái Q khả dĩ nhất cho chuỗi quan sát trên. Ví dụ trong

bài toán PoS, mục tiêu của mô hình Markov là lựa chọn ra một chuỗi tag t_1^n phù hợp nhất với chuỗi quan sát gồm n từ cho trước w_1^n:

$$\hat{t}_1^n = \underset{t_1^n}{argmax} P(t_1^n | w_1^n)$$

Cách thức để thực hiện là sử dụng công thức Bayes để tính toán:

$$\hat{t}_1^n = \underset{t_1^n}{argmax} \frac{P(w_1^n | t_1^n) P(t_1^n)}{P(w_1^n)}$$

Ta đơn giản hóa công thức bằng cách lược bỏ đi mẫu số của biểu thức. Từ đó ta được công thức:

$$\hat{t}_1^n = \underset{t_1^n}{argmax} P(w_1^n | t_1^n) P(t_1^n)$$

Như đã đề cập tới ở phần trước, HMM bao gồm 2 giả sử. Thứ nhất, xác suất xuất hiện của một từ chỉ phụ thuộc vào tag của nó và không phụ thuộc vào bất kì từ hay tag nào khác trong câu:

$$P(w_1^n | t_1^n) \approx \prod_{i=1}^{n} P(w_i | t_i)$$

Giả thiết thứ 2 cho biết, một tag chỉ phụ thuộc vào tag trước đó chứ không phụ thuộc vào cả chuỗi tag:

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1})$$

Từ các công thức rút ra từ 2 giả thiết trên, ta có công thúc sau:

$$\hat{t}_1^n = \underset{t_1^n}{argmax} P(t_1^n | w_1^n) \approx \underset{t_1^n}{argmax} \prod_{i=1}^{n} \overbrace{P(w_i | t_i)}^{emission} \overbrace{P(t_i | t_{i-1})}^{emission}$$

*Note: Phương pháp decoding phổ biến trong HMM là thuật toán quy hoạch động Viterbi (được nhóm em trình bày ở phần tiếp theo của báo cáo).

III. Trường điều kiện ngẫu nhiên (CRF)

1. *Giới thiệu về CRF (Conditional Random Fields)*

CRF là một mô hình xác suất phân biệt, trực tiếp mô hình các xác suất có điều kiện của một chuỗi tag gán cho một chuỗi từ cho trước. Với CRF xác suất có điều kiện của chuỗi tag $y = (y_1, y_2, \dots, y_m)$ của một chuỗi từ cho trước $x = (x_1, x_2, \dots, x_m)$ được định nghĩa như sau:

$$P(y|x) = \frac{\exp(wF(y,x))}{\sum_{y' \in Y} \exp(wF(y',x))}$$

Trong đó w là tham số vector được ước lượng từ dữ liệu huấn luyện; $F(y,x) \in R^d$ là một hàm đặc trưng được định nghĩa cho toàn bộ chuỗi đầu vào và toàn bộ chuỗi tag, Y là tập tất cả các chuỗi tag khả năng được gán. Hàm đặc trưng $F(y,x)$ được tính bởi tổng các hàm đặc trưng nội bộ:

$$F_j(y,x) = \sum_{i=1}^{n} f_j(y_{i-1}, y_i, x, i)$$

Tham số của CRF được ước lượng bởi hàm cực đại log-likelihood, tham số được ước lượng bằng cách sử dụng thuật toán iterative-scaling hoặc phương pháp gradient descent.

2. *Hàm đặc trưng*

a. Hàm đặc trưng trong CRF

- Đầu vào (input):
  - Một câu s
  - Vị trí i của 1 từ trong câu
  - Nhãn l_i của từ hiện tại
  - Nhãn l_{i-1} của từ trước đó
- Đầu ra (output): Một giá trị số thực nằm trong khoảng [0, 1]. Bằng cách giới hạn các đặc trưng chỉ phụ thuộc vào nhãn hiện tại và nhãn trước đó thay vì phụ thuộc một cách tùy ý vào các nhãn trong câu, ta xây dựng một trường hợp đặc biệt của CRF là CRF chuỗi tuyến tính (Linear-chain CRF).
- Ví dụ: Một hàm đặc trưng có thể đo được khả năng của từ hiện tại được gán nhãn là một tính từ nếu biết từ trước nó là “very”.

b. Từ đặc trưng tới xác suất

- Ta gán mỗi hàm đặc trưng f_i một trọng số λ_i (trọng số được học từ dữ liệu huấn luyện). Cho một câu s, ta có thể tính điểm cho nhãn l trong câu s bởi tính tổng trọng số các hàm đặc trưng trên tất cả các từ của câu:

$$score(l|s) = \sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_i f_i(s, l, l_i, l_{i-1})$$

Với m là số hàm đặc trưng, n là số vị trí trong câu.

- Ta có thể chuyển hóa những điểm tích lũy đó về một xác suất có điều kiện $p(l|s)$ bằng cách chuẩn hóa và lũy thừa:

$$p(l|s) = \frac{\exp[score(l|s)]}{\sum_{l'} \exp[score(l'|s)]} = \frac{\exp[\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_i f_i(s, l, l_i, l_{i-1})]}{\sum_{l'} \exp[\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_i f_i(s, l, l'_i, l'_{i-1})]}$$

Với l' là các nhãn có thể xuất hiện trong câu s.

c. Ví dụ về hàm đặc trưng

Ở đây, để dễ hình dung, nhóm em xin trình bày 1 ví dụ trong bài toán nhận dạng từ loại với tiếng Anh (PoS tagging):

$-f_1(s, l, l_i, l_{i-1}) = 1$ nếu $l_i = ADVERB$ và từ thứ i kết thúc bằng "ly", trái lại hàm có giá trị bằng 0.

Nếu trọng số λ_1 của hàm đặc trưng này là số dương và lớn thì đặc trưng này cho biết rằng ta sẽ gán nhãn vị trí từ kết thúc bởi "ly" là trạng từ.

$-f_2(s, l, l_i, l_{i-1}) = 1$ nếu $i = 1$, $l_i = VERB$ và câu kết thúc bởi dấu "?", trái lại hàm có giá trị bằng 0.

Nếu trọng số λ_2 của hàm đặc trưng này là số dương và lớn thì đặc trưng này cho biết rằng ta sẽ gán nhãn từ có vị trí đầu tiên là động từ nếu câu kết thúc bởi dấu "?".

$-f_3(s, l, l_i, l_{i-1}) = 1$ nếu $l_{i-1} = ADJECTIVE$ và $l_i = NOUN$, trái lại hàm có giá trị bằng 0.

Nếu trọng số λ_3 của hàm đặc trưng này là số dương và lớn thì đặc trưng này cho biết rằng một tính từ có xu hướng được theo sau bởi một danh từ.

$-f_4(s, l, l_i, l_{i-1}) = 1$ nếu $l_{i-1} = PREPOSITION$ và $l_i = PREPOSITION$, trái lại hàm có giá trị bằng 0.

Nếu trọng số λ_4 của hàm đặc trưng là số âm thì đặc trưng này cho ta biết rằng ta sẽ không gán nhãn PREPOSITION cho từ hiện tại nếu từ trước đó đã được gán nhãn PREPOSITION.

3. So sánh CRFs với Logistic Regression

CRF cơ bản là phiên bản tuần tự của Logistic Regression. Trong khi Logistic Regression là một hô hình tuyến tính logarit (log-linear) cho việc phân loại, CRF là mô hình log-linear cho chuỗi nhãn tuần tự.

4. So sánh CRFs và HMMs

a. So sánh

- HMM là một mô hình khác cho PoS Tagging. Trong khi CRFs gộp nhiều hàm đặc trưng lại với nhau để tính toán điểm tích lũy của nhãn, HMM sử dụng phương pháp sinh dựa trên xác suất kết hợp để gán nhãn:

$$p(l|s) = p(l_1)\prod_i p(l_i|l_{i-1})p(w_i|l_i)$$

Với:

- $p(l_i|l_{i-1})$ là xác suất chuyển tiếp của nhãn l_i theo sau bởi nhãn l_{i-1}
- $p(w_i|l_i)$ là mật độ xác suất để từ w_i được gán nhãn l_i.

- CRF mạnh hơn so với HMM bởi CRF có thể mô hình hóa mọi việc mà HMM có thể làm được và nhiều hơn thế nữa.

Hàm logarit xác suất của mô hình HMM:

$$logp(l|s) = logp(l_0) + \sum_i logp(l_i|l_{i-1}) + \sum_i logp(w_i|l_i)$$

Đây cũng chính là dạng log-linear của CRF nếu ta coi logarit của các xác suất là các trọng số của các chuyển hóa nhị phân và các đặc trưng quan sát.

Do đó, ta có thể xây dựng mô hình CRF tương đương với bất kỳ mô hình HMM nào.

- HMMs là các mô hình sinh mẫu sinh (generative models) mô hình hóa bằng cách sử dụng xác suất kết hợp$P(x, y)$, do đó các HMMs mô hình hóa các dữ liệu phân tán $P(x)$ lần lượt áp đặt những đặc trưng hoàn toàn độc lập với nhau.

Những đặc trưng độc lập này nhiều khi không mong muốn gây khó khăn trong việc mô hình hóa và tính toán.

CRFs là mô hình phân biệt ("discriminative models)" mô hình hóa bằng cách sử dụng xác suất có điều kiện $P(y|x)$, do đó CRF không yêu cầu các mẫu $P(x)$ rõ ràng và phụ thuộc vào bài toán. Từ đó, hiệu suất được nâng cao do chúng cần ít tham số để học, phù hợp để sử dụng các đặc trưng phức tạp và chồng lên nhau.

b. Chuyển hóa giữa CRFs và HMMs

Ta có thể xây dựng mô hình CRF tương đương với bất kỳ mô hình HMM nào bằng cách:

- Với mỗi xác suất chuyển hóa HMM $p(l_i = y|l_{i-1} = x)$, ta định nghĩa một tập các đặc trưng chuyển hóa CRF (transition features CRF) với dạng $f_{x,y}(s, l, l_i, l_{i-1}) = 1$ nếu $l_i = y, l_{i-1} = x$
- Tương tự với mỗi xác xuất quan sát $p(w_i = z|l_i = x)$, ta định nghĩa một tập các đặc trưng quan sát (emission features) với dạng $g_{x,y}(s, i, l_i, l_{i-1}) = 1$ nếu $w_i = z, l_i = x$. Mỗi đặc trưng ta gán một trọng số $w_{x,z} = logp(w_i = z|l_i = x)$.

c. Ưu điểm của CRFs so với HMMs

- CRF định nghĩa một tập các đặc trưng lớn hơn nhiều HMM:

- HMM bị ràng buộc bởi chuyển tiếp nhị phân và các đặc trưng quan sát, điều này bắt buộc mỗi từ chỉ phụ thuộc vào nhãn hiện tại và mỗi nhãn chỉ phụ thuộc vào nhãn trước đó.
- CRF sử dụng các đặc trưng tổng thể.

  Ví dụ: Một trong các đặc trưng PoS Tagger ở trên làm tăng xác suất gán nhãn từ đầu tiên của một câu là động từ nếu câu kết thúc bởi dấu "?"

- CRF có thể có các trọng số của các hàm đặc trưng tùy ý theo đánh giá lựa chọn trong khi HMM phải đảm bảo thỏa mãn các ràng buộc nhất định như $0 \leq p(w_i|l_i) \leq 1$ và $\sum_w p(w_i = w|l_1 = 1)$. Trọng số của CRF không bị giới hạn bởi $logp(w_i|l_i)$ có thể là bất cứ giá trị nào.

5. Thuật toán gán nhãn tối ưu

Sau khi huấn luyện xong một mô hình CRF, ta có một câu mới cần gán nhãn. Ta cần thực hiện gán nhãn cho câu:

a. Phương pháp ngây thơ

Cách đơn giản nhất là tính $p(l|s)$ với tất cả các nhãn l có thể, sau đó chọn nhãn mà có xác xuất lớn nhất trong các nhãn.

Tuy nhiên, một câu có độ dài m có thể có k^m khả năng gán nhãn với một tập gồm k nhãn khác nhau. Cách tiếp cận này cần phải kiểm tra cấp lũy thừa xác trường hợp xảy ra, điều đó là không khả thi.

Do đó ta cần sử dụng một thuật toán quy hoạch động để tìm nhãn tối ưu (tương tự thuật toán Viterbi trong HMM).

b. Thuật toán Viterbi

Viterbi thuật toán giải mã cho mô hình HMM, là thuật toán tìm chuỗi nhãn tối ưu. Thuật toán tả về đường dẫn trạng thái thông qua HMM được gán chuỗi trạng thái có xác suất xảy ra lớn nhất khi cho trước một chuỗi quan sát và một mẫu HMM $\lambda = (A, B)$.

- Thuật toán Viterbi đầu tiên cài đặt một ma trận xác suất với mỗi cột là một quan sát o_t và mỗi hàng cho mỗi trạng thái trong đồ thị trạng thái. Mỗi hàng có một ô cho mỗi trạng thái q_i trong chuỗi kết hợp đơn tự động.

Mỗi ô của ma trận $v_t(j)$ biểu diễn xác suất HMM ở trạng thái j sau khi quan sát chuỗi t quan sát (observations) đầu tiên và truyền qua chuỗi trạng thái có khả năng xảy ra cao nhất $q_1, q_2, \dots, q_{t-1}$ được cho bởi HMM λ. Giá trị của mỗi ô $v_t(j)$ được tính bởi quy hoạch động qua đường dẫn có khả năng xảy ra lớn nhất có thể dẫn tới ô $v_t(j)$ này. Xác suất mỗi ô được tính bởi công thức:

$$v_t(j) = \max_{q_1,\dots,q_{t-1}} p(q_0, q_1, \dots, q_{t-1}, o_1, o_2, \dots, o_t, q_t = j|\lambda)$$

Đường dẫn có khả năng cao nhất được lấy thông qua cực đại của tất cả các chuỗi trạng thái có thể trước đó $\max_{q_0,\dots,q_{t-1}}$. Như tất cả các thuật toán quy hoạch động khác, Viterbi hoàn thiện các ô bởi đệ quy. Sau khi tính được xác suất của mỗi trạng thái ở thời điểm t – 1, xác suất Viterbi được tính bởi đường dẫn có khả năng xảy ra

cao nhất trong tất cả các đường dẫn mở rộng tới ô hiện tại. Khi đã biết trạng thái q_j tại thời điểm t, giá trị $v_t(j)$ được tính bởi công thức:

$$v_t(j) = \max_{i=1...N} v_{t-1}(i) a_{ij} b_j(o_t)$$

Ba thừa số được nhân với nhau để mở rộng đường dẫn trước đó dùng để tính xác suất Viterbi tại thời điểm t là

- $v_{t-1}(i)$: Đường dẫn xác suất Viterbi trước đó.
- a_{ij}: xác suất chuyển tiếp trạng thái
- $b_j(o_t)$: khả năng quan sát trạng thái của quan sát o_t được cho bởi trạng thái j hiện tại.

Mã giả thuật toán Viterbi:

function $VITERBI(observationsoflenT, state-graphoflenN)$ **returns** *best-path*, *path-prob*
create a path probability matrix $viterbi[N,T]$
for each state *s* **from** 1 **to** N **do** ; bước khởi tạo
    $viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$
    $backpointer[s,1] \leftarrow 0$
for each time step *t* **from** 2 **to** T **do** ; bước đệ quy
    for each state *s* **from** 1 **to** N **do**
        $viterbi[s,t] \leftarrow \max_{s'=1\rightarrow N} viterbi[s',t] * a_{s',s} * b_s(o_t)$
        $backpointer[s,t] \leftarrow \underset{s'=1\rightarrow N}{argmax} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1\rightarrow N} viterbi[s,T]$; bước kết thúc
$bestpathpointer \leftarrow \underset{s=1\rightarrow N}{argmax} viterbi[s,T]$; bước kết thúc
$bestpath \leftarrow$ the path starting at state $bestpathpointer$, that follows $backpointer[]$ to states back in time
return $bestpath, bestpathprob$

6. Huấn luyện tập trọng số (Learning weights)

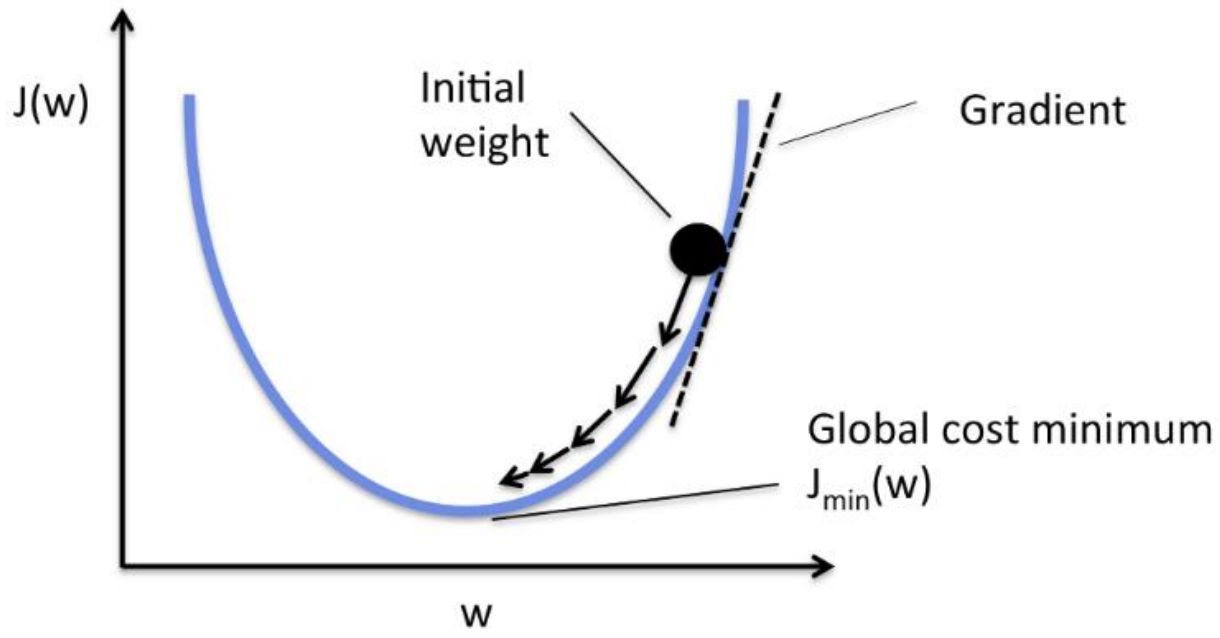
Như đã đề cập tới ở trên, ta gán mỗi hàm đặc trưng f_i một trọng số λ_i (trọng số được học từ dữ liệu huấn luyện). Phương pháp huấn luyện được sử dụng trong trường điều kiện ngẫu nhiên là thuật toán Gradient Descent.

a. Gradient Descent

Một trong những thuật toán phổ biến và được sử dụng nhiều nhất trong lĩnh vực học máy là Gradient Descent. Vai trò của thuật toán này trong bài toán gán nhãn thực thể là vô cùng quan trọng. Nhưng trước khi đến với vai trò đó, nhóm em xin trình bày sơ lược và ngắn gọn nhất về thuật toán Gradient Descent.

Các mô hình học máy nói chung thường có các tham số như các trọng số (weights) và hàm chi phí để đánh giá mức độ phù hợp của một tập các tham số trong bài toán cần giải quyết. Rất nhiều bài toán trong lĩnh vực học máy được gói gọn bằng việc tìm ra một tập hợp các trọng số cho mô hình nhằm tối thiểu hóa hàm chi phí.

Đây là thuật toán sử dụng vòng lặp để cải thiện chất lượng của tham số. Ta thường khởi tạo các giá trị ngẫu nhiên cho tập tham số (các trọng số), sau đó cải thiện dần dần qua các vòng lặp.



Hình 1: Mô phỏng giải thuật

Để cải thiện một tập các trọng số cho trước, ta cần xem xét các giá trị gradient của hàm chi phí với tập các trọng số hiện tại của mô hình. Tiếp đó, ta sẽ thực hiện cập nhật giá trị cho các trọng số nhằm mục đích làm giảm giá trị của hàm chi phí. Việc lặp lại bước trên với số lượng lớn (từ vài trăm đến vài nghìn vòng lặp), ta có thể tối thiểu hóa hàm chi phí của mô hình.

b. Các bước cài đặt thuật toán Gradient Descent:

Gradient Descent được sử dụng để tối thiểu hóa hàm chi phí $J(W)$ được tham số bởi các tham số W của mô hình.

Các gradient (hoặc đạo hàm nhiều biến) chỉ ra độ dốc của đồ thị hàm chi phí. Từ đó để thực hiện tối thiểu hóa hàm chi phí, ta dịch chuyển theo hướng gradient.

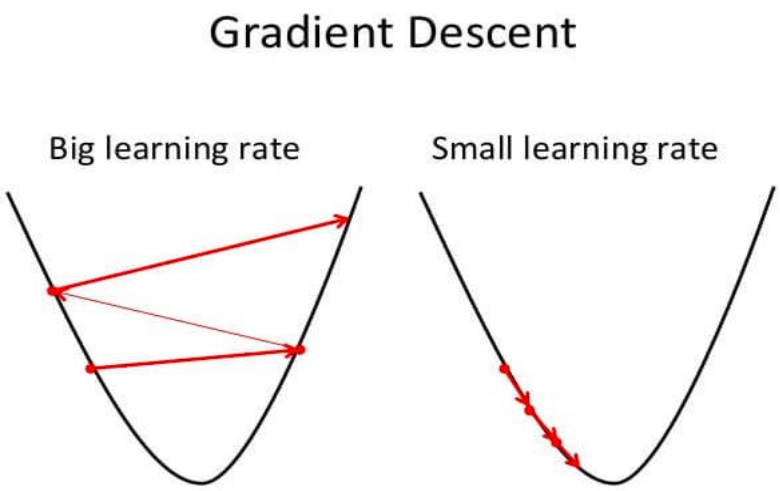
Khởi tạo ngẫu nhiên các giá trị trọng số weights W.

Tính toán các gradient G của hàm chi phí bằng cách đạo hàm từng phần $G = \partial J(W)/\partial W$. Giá trị của các gradient thì phụ thuộc vào dữ liệu đầu vào, giá trị của các tham số hiện tại trong mô hình và hàm chi phí.

Cập nhật các giá trị weights: $W = W - \alpha G$ với α là learning rate lựa chọn.

Lặp lại các bước trên với số lần lặp nhất định cho đến khi hàm chi phí không thể giảm nữa hoặc bắt gặp 1 điều kiện dừng nào đó.

Việc lựa chọn learning rate α ảnh hưởng rất lớn đến kết quả bài toán. Nếu α quá lớn, hàm chi phí sẽ không hội tụ về cực trị địa phương (hoặc cực trị toàn cục). Mặt khác nếu α quá bé, giải thuật sẽ cải thiện vô cùng chậm



Hình 2: Ảnh hưởng của độ học η

c. Huấn luyện trọng số trong CRFs

Như nhóm em đã đề cập ở phần trước, các đặc trưng của CRF được gán với các trọng số. Các tham số này sẽ được "học" qua quá trình huấn luyện mô hình trong bài toán gán nhãn thực thể. Việc thực hiện quá trình training này được thực hiện bằng phương pháp gradient descent.

Giả sử rằng ra có một tập các dữ liệu huấn luyện. Theo phương pháp trên, ta khởi tạo các giá trị weights ngẫu nhiên cho mô hình CRF. Với mỗi dữ liệu huấn luyện:

- Chạy qua tất cả các hàm đặc trưng f_i, tính toán giá trị gradient của xác suất hàm log của dữ liệu huấn luyện đó theo w_i:

$$\frac{\partial}{\partial w_j} logp(l|s) = \sum_{j=1}^{m} f_i(s, j, l_j, l_{j-1}) - \sum_{l'} p(l'|s) \sum_{j=1}^{m} f_i(s, j, l'_j, l'_{j-1})$$

Thành phần đầu tiên của vế phải đại diện cho phân bố của hàm đặc trưng f_i dưới nhãn đúng, thành phần thứ 2 của gradient là phân bố kì vọng của hàm đặc trưng f_i dưới nhãn đang xét.

- Dịch chuyển w_i theo hướng gradient:

$$w_i = w_i + \alpha \left[\sum_{j=1}^{m} f_i(s, j, l_j, l_{j-1}) - \sum_{l'} p(l'|s) \sum_{j=1}^{m} f_i(s, j, l'_j, l'_{j-1}) \right]$$

- Lặp lại các bước trên cho đến khi đạt trạng thái dừng

d. Thuật toán tối ưu L-BFGS (Limited Memory Broyden-Fletcher-Goldfarb-Shanno)

Đây là thuật toán hướng tới việc tìm ra cực tiểu (địa phương) của một hàm mục tiêu lợi dụng các giá trị của hàm mục tiêu cũng như gradient của hàm.

Phương pháp bậc nhất là phương pháp sử dụng đạo hàm bậc nhất chứ không sử dụng ma trận Hesse (đạo hàm của đạo hàm).

Phương pháp bậc 2 được sử dụng, có nghĩa là cả gradients (đạo hàm riêng của hàm) cũng như ma trận Hesse đều được sử dụng. Về mặt trực quan, gradient cho biết mức độ biến thiên của hàm số, còn ma trận Hesse cho biết thông tin về mức độ biến thiên của gradient. Do đó nếu sử dụng phương pháp bậc 2, ta sẽ có nhiều thông tin

hơn về độ cong của hàm dẫn đến khả năng hội tụ có thể tốt hơn. Ví dụ, để tìm cực tiểu của một hàm bậc 2, ta cần chạy nhiều hơn một vòng lặp nếu sử dụng phương pháp bậc nhất. Nhưng khi sử dụng phương pháp bậc 2, chỉ cần một bước ta đã có thể tìm được cực tiểu của hàm số. Với các bài toán thực tế, việc chỉ dùng một vòng lặp để tìm ra cực trị là khó có thể xảy ra, thế nhưng về cơ bản, phương pháp bậc 2 sẽ tăng độ chính xác và tốc độ đạt tới cực trị nhanh hơn so với phương pháp bậc một chỉ sử dụng đơn thuần gradients. Một trong những phương pháp bậc 2 được sử dụng rộng rãi là phương pháp Quasi-Newton (BFGS và L-BFGS). BFGS là phương pháp tính toán và lưu trữ toàn bộ ma trận Hesse sau mỗi vòng lặp. Việc tính toán này là khá tốn kém với độ phức tạp O(n2) với n là số chiều ta cần tối ưu hóa. L-BFGS thì khác, phương pháp này tính toán và lưu trữ xấp xỉ Hesse dựa trên m giá trị gradients gần nhất với m thường có giá trị từ 10 đến 20. Mỗi vòng lặp của L-BFGS cố gắng để dự đoán bước tương đương của BFGS sẽ làm gì với mức độ phức tạp thấp hơn rất nhiều so với BFGS (O(m.n)). Do đó ta có thể thấy L-BFGS hội tụ về điểm cực trị nhanh hơn nhiều so với BFGS bởi nó có thể thực hiện được nhiều vòng lặp hơn trong cùng một đơn vị thời gian.

Việc sử dụng L-BFGS trong thuật toán Gradient Descent giúp thời gian thực hiện huấn luyện nhanh hơn nhiều so với các thuật toán huấn luyện khác. Do đó thuật toán được bọn em lựa chon trong huấn luyện mô hình CRF là gradient descent sử dụng thuật toán tối ưu L-BFGS.

Một điểm đặc biệt khi sử dụng thuật toán tối ưu L-BFGS trong thuật toán Gradient Descent đó là việc ta không cần lựa chọn learning rate α, và thuật toán này được cài đặt sẵn trong rất nhiều thư viện phổ biến hiện nay, trong đó có CRFsuite.

IV. Quy trình thực nghiệm

1. CRFsuite và eli5

a. CRFsuite

CRFsuite là một ứng dụng triển khai mô hình CRFs cho việc gán nhãn dữ liệu tuần tự. So với các ứng dụng triển khai CRFs khác, CRFsuite có các tính năng:

- **Training và tagging nhanh**
  CRFsuite hướng đến mục tiêu là train và sử dụng mô hình CRF một cách nhanh nhất có thể
- **Định dạng dữ liệu đơn giản cho training và tagging**
  Định dạng dữ liệu CRFsuite sử dụng giống với các công cụ học máy khác. Gồm các dòng chứa nhãn và thuộc tính (đặc trưng) của một item, dòng trống đánh dấu sự kết thúc chuỗi item tuần tự. Vậy người dùng có thể thiết kế một số lượng bất kì các đặc trưng cho mỗi item
- **Phương pháp training hiện đại**
  CRFsuite có thể thực thi các thuật toán: Limited-memory BFGS (L-BFGS), Orthant-Wise Limited-memory Quasi-Newton (OWL-QN), Stochastic Gradient Descent (SGD), Averaged Perceptron, Passive Aggressive, Adaptive Regularization Of Weight Vector (AROW).
- **Thuật toán tiến/lùi (forward/backward) sử dụng phương pháp thang đo**
  Phương pháp thang đo có vẻ nhanh hơn việc tính toán điểm forward/backward trong miền logarit
- **Sử dụng Linear-chain CRF**
- **Đánh giá quá trình training:**
  CRFsuite có thể đưa ra điểm precision, recall, F1 cho một tập kết quả dựa trên test data
- **Định dạng file hiệu quả cho việc lưu trữ và truy cập mô hình CRF**
- **Hỗ trợ API**

b. Tại sao lựa chọn CRFsuite

Ngoài CRFsuite, còn có rất nhiều phần mềm khác phục vụ cho việc triển khai CRFs để gán nhãn thực thể có thể kể đến như Wapiti, CRF++, sgd, … Nhưng với

những ưu điểm của mình và khi so với các phần mềm khác, CRFsuite có kết quả vượt trội, độ chính xác cao và thời gian thực thi nhanh.

Trong cùng một điều kiện, sử dụng giải thuật L-BFGS để tiến hành training và thử nghiệm trên cùng một test data thu được kết quả:

	CRFsuite	**Wapiti**	**CRF++**	**MALLET**
Số các đặc trưng	7385312	7448122	7448606	592965
Thời gian train (s)	636.8	1292.5	4985.8	11858.8
Số vòng lặp	159	226	131	105
Độ chính xác (%)	96.010	95.753	96.070	95.762

Sklearn-crfsuite

Sklearn-crfsuite là một thư viện trong python giúp cho việc phân loại chuỗi bằng CRF. Sklearn-crfsuite bao hàm thư viện python-crfsuite (đây là thư viện python hỗ trợ sử dụng CRFsuite) và nó cung cấp các API cao cấp hơn python-crfsuite.

Lớp sklearn_crfsuite.CRF

Các tham số khởi tạo:

- **algorithm**: str (mặc định = ‘lbfgs’)
  Giải thuật cho training. Hỗ trợ các thuật toán: ‘lbfgs’, ‘l2sgd’, ‘ap’, ‘pa’, ‘arow’.
- **min_freg**: float (mặc định = 0)
- **all_possible_states**: bool (mặc định = False)
  Chỉ định CRFsuite có sinh ra đặc trưng trạng thái mà không xuất hiện trong training data.
  Nếu True, CRFsuite sẽ sinh ra tất cả các đặc trưng trạng thái cho các nhãn và từ xuất hiện trong training data. Tức là nếu có A từ và B nhãn, CRFsuite sẽ sinh ra (A*B) đặc trưng trạng thái
- **all_possible_transitions**: bool (mặc định = False)
  Chỉ định CRFsuite có sinh ra đặc trưng chuyển trạng thái mà không xuất hiện trong training data.
  Nếu True, CRFsuite sẽ sinh ra tất cả cá đặc trung chuyển trạng thái cho tất cả các nhãn. Tức nếu có A nhãn, CRFsuite sẽ sinh ra (A*A) đặc trưng chuyển trạng thái
- **c1**: float (mặc đinh = 0)

Hệ số cho chính quy hóa L1

- **c2**: float (mặc định = 1.0)
  Hệ số cho chính quy hóa L2
- **max_iterations**: int
  Vòng lặp tối đa cho các thuật toán tối ưu, giá trị mặc định phụ thuộc vào thuật toán sử dụng (với lbfgs sẽ là không giới hạn)
- **num_memories** : int (mặc định = 6)
  Bộ nhớ giới hạn để ước lượng ma trận Hesse đảo
- **epsilon**: float (mặc định = 1e-5)
  Tham số epsilon xác định điều kiện hội tụ
- **period**: int (mặc định = 10)
  Thời gian vòng lặp kiểm tra điều kiện dừng
- **delta**: float (mặc định = 1e-5)
  Ngưỡng dừng vòng lặp
- **linesearch**: str (mặc định = 'MoreThuente')
  Thuật toán line search sử dụng trong cập nhật L-BFGS
- **max_linesearch**: int (mặc đinh = 20)
  Số phép thử tối đa cho thuật toán line search

Và các tham số khác nhưng không hỗ trợ cho thuật toán lbfgs nên sẽ không nêu ra ở đây.

Các phương thức chính:

- fit(X,y,X_dev=None,y_dev=None)
  Thực hiện train
  Tham số: X: tập các đặc trưng ; **y**: tập các nhãn ; **X_dev**: tập các đặc trưng cho testing ; **y_dev**: tập các nhãn tương ứng với X_dev
- predict(X)
  Đưa ra dự đoán
  Tham số: X: tập các đặc trưng
  Giá trị trả về: **Y**: tập các nhãn dự đoán

c. Eli5

Eli5 là một thư viện python cho phép hiển thị và debug các mô hình học máy khác nhau sử dụng API thông nhất. Hỗ trợ tích hợp cho các framework về ML như scikit-learning, XGBoost, LightGBM, lightning,… và sklearn-crfsuite.

Eli5 hỗ trợ hàm eli5.explain_weights() cho đối tượng lớp sklearn_crfsuite.CRF. Hàm này sẽ đưa ra giải nghĩa về tham số ước lượng (weights), trả về dạng có thể đọc được qua eli5.formatters.

2. Quy trình thực nghiệm:

Dữ liệu huấn luyện và kiểm tra: VLSP 2016- underthesea (Chi tiết github tại mục tài liệu tham khảo).

Chúng em lựa chọn ngôn ngữ Python và thư viện sklearn-crfsuite, thuật toán để training là L-BFGS với hệ số c1=0.06 và c2=0.1.

Định nghĩa các feature:

Mô hình NER này sẽ bao gồm các đặc trưng của từ, đặc trưng wordshape, đặc trưng nhãn PoS và chunk.

Chúng em sử dụng mô hình ngôn ngữ N-gram với unigram và bigram (với window size = 5). Nghĩa là với một đặc trưng F cho từ hiện tại, ta sẽ có các đặc trưng unigram và bigram sau:

- **unigrams**: F[-2],F[-1],F[0],F[1],F[2]
- **bigrams**: F[-2]F[-1], F[-1]F[0], F[0]F[1], F[1]F[2]

Đặc trưng từ: các đặc trưng nội dung của từ được chiết suất vào unigram và bigram với window size là 5. Sử dụng đặc trưng dạng viết thường của từ. Sử dụng đặc trưng về tiền tố và hậu tố lấy tới 4 kí tự của từ đang xét.

Đặc trưng wordshape: wordshape được sử dụng để tăng độ chính xác cho chương trình, dành cho những từ hiếm và giảm thiểu về vấn đề bộ nhớ. Các đặc trưng chúng em sử dụng trong chương trình:

Đặc trưng	**Ý nghĩa**	**Ví dụ**
wordShape	Hình thái của từ	“Việt” – “ULLL”
isLower	Từ có được viết thường hết không	“trường”
isUpper	Từ có được viết hoa hết không	“GPA”
isNumber	Từ đang xét có phải là số	“1234”
isTitle	Kí tự đầu của từ có viết hoa không	“Mỹ”
isCapWithPeriod	Từ bắt đầu bằng kí tự viết hoa và kết thúc bởi dấu chấm	“Abc.”
endsInDigit	Kết thúc bởi 1 chữ số	“B5”
containHyphen	Chứa dấu ‘-‘	“New-York”
isName	Có định dang tên riêng	“Việt_Nam”

isMixCase	Có kí tự đầu viết thường và chứa kí tự viết hoa	“iPhone”
d&comma	Dạng số với dấu phẩy	“40,000”
d&period	Dạng số với dấu chấm	“40.000”

Đặc trưng nhãn PoS và chunking : Sử dụng unigram và bigram với window size là 5 cho nhãn PoS và chunking.

Các nhãn PoS sử dụng:

Nhãn	Giải thích	Ý nghĩa
Np	Proper noun	Danh từ riêng
Nc	Classifier	Danh từ chỉ loại
Nu	Unit noun	Danh từ đơn vị
Ny	Noun shorten	Danh từ dạng viết tắt
N	Common noun	Danh từ thường
V	Verb	Động từ
Vy	Verb shorten	Động từ dạnh viết tắt
A	Adjective	Tính từ
P	Pronoun	Đại từ
R	Adverb	Trạng từ
L	Determiner	Từ hạn định
M	Numeral	Lượng từ
E	Preposition	Giới từ
C	Conjunction	Liên từ
CH	punctuation	Dấu câu
I	Interjection	Thán từ
T	Auxiliary	Trợ động từ
Y	Abbreviation	Từ viết tắt
Z	Bound morphemes	Yếu tố cấu tạo từ
X	Unkown	Không xác định

Các nhãn chunking sử dụng:

Nhãn	Ý nghĩa
O	Các từ không thuộc các cụm danh, động, tính, giới từ nào
B-PP	Thành phần đầu trong cụm giới từ
I-VP	Thành phần không phải đầu trong cụm động từ
B-AP	Thành phần đầu trong cụm tính từ

B-NP	Thành phần đầu trong cụm danh từ
I-NP	Thành phần không phải đầu trong cụm danh dừ
I-AP	Thành phần không phải đầu trong cụm tính từ
B-VP	Thành phần đầu trong cụm động từ

Sử dụng thuật toán training là L-BFGS với các hệ số c1 = 0.06 và c2 = 0.1

Tham số ước lượng sẽ được tính và thống kê lại nhờ thư viện eli5, đưa ra kết quả cho các đặc trưng chuyển trạng thái (transition feature) và top 30 cho các đặc trưng trạng thái (state feature).

3. Đánh giá kết quả

Chúng em đưa ra danh sách kết quả của mình theo chunk-base và token-base, cho 2 trường hợp là không sử dụng đặc trưng nhãn PoS và chunk và có sử dụng. Công cụ đánh giá sử dụng là conlleval.

Kết quả khi train không có đặc trưng nhãn PoS và chunk:

- Token-base:

	Precision	**Recall**	**FB1**
B-LOC	85.86%	87.35%	86.60%
B-MISC	92.11%	71.43%	80.46%
B-ORG	73.99%	46.72%	57.27%
B-PER	92.22%	85.24%	88.59%
I-LOC	79.16%	85.31%	82.12%
I-MISC	92.11%	71.43%	80.46%
I-ORG	80.54%	67.76%	73.60%
I-PER	95.62%	93.29%	94.44%
Overall	88.07%	83.60%	85.78%

- Chunk-base:

	Precision	**Recall**	**FB1**
LOC	85.14%	86.50%	85.82%
MISC	92.11%	71.43%	80.46%
ORG	68.21%	43.07%	52.80%
PER	91.81%	84.85%	88.19%
Overall	87.03%	81.57%	84.21%

Kết quả khi train sử dụng đặc trưng nhãn PoS và chunk:

- Token-base:

	Precision	Recall	FB1
B-LOC	89.94%	94.19%	92.01%
B-MISC	100%	87.76%	93.48%
B-ORG	86.89%	65.33%	74.58%
B-PER	94.04%	93.82%	93.93%
I-LOC	89.19%	92.66%	90.89%
I-MISC	100%	87.76%	93.48%
I-ORG	93.18%	89.42%	91.26%
I-PER	97.56%	97.46%	97.51%
Overall	92.79%	92.47%	92.63%

- Chunk-base:

	Precision	Recall	FB1
LOC	90.01%	94.12%	92.02%
MISC	100%	87.76%	93.48%
ORG	86.89%	65.33%	74.58%
PER	93.96%	93.74%	93.85%
Overall	91.65%	91.22%	91.43%

Như vậy việc sử dụng đặc trưng nhãn PoS và chunk đã cải thiện độ chính xác 1 cách đáng kể cho chương trình (khoảng 7%). Chúng em sẽ lấy kết quả theo chunk base cho hệ thống sử dụng đặc trưng nhãn PoS và chunk làm kết quả cuối cùng.

Ngoài ra, còn 1 kết quả có thể nhận ra 1 cách rõ ràng đó là việc nhận dạng B-ORG cho kết quả khá thấp. Sau khi xem xét đến dữ liệu đầu ra, có thể nhận thấy rằng mô hình thường nhận dạng từ bắt đầu của tổ chức sai khi nó đứng riêng 1 mình và được bắt đầu bằng 1 chữ cái in hoa. Dấu hiệu trên thường bị nhầm lẫn với dấu hiệu nhận biết B-LOC. Đây cũng là vấn đề mà nhóm em vẫn chưa giải quyết được trong đề tài này. Các kết quả nhận dạng khác tuy chưa phải quá cao nhưng cũng đạt mức độ chính xác chấp nhận được.

Thống kê về tham số ước lượng (weight):

- *Tham số ước lượng cho đặc trưng chuyển trạng thái (transition feature):*

From \ To	O	B-LOC	I-LOC	B-MISC	I-MISC	B-ORG	I-ORG	B-PER	I-PER
O	2.874	1.143	-3.25	0.01	-2.117	0.284	-3.011	0.497	-2.043
B-LOC	-0.519	0.315	5.984	-0.079	-0.324	-0.149	-2.057	-0.903	-1.882
I-LOC	-0.173	-0.015	4.592	0.0	0.0	-0.487	-2.33	-0.372	-2.45
B-MISC	-1.292	-0.038	-0.676	0.0	3.987	0.0	-0.781	-0.09	-0.316
I-MISC	-0.41	-0.0	-0.19	0.0	1.264	0.008	-0.438	0.0	0.0
B-ORG	-2.567	-0.525	-1.007	0.0	-0.15	0.229	3.501	-0.666	-2.111
I-ORG	-0.257	-0.896	-2.623	0.0	-0.324	0.039	3.985	0.364	-1.731
B-PER	-0.328	-0.952	-1.829	0.0	-0.279	-0.244	-2.143	-1.944	3.5
I-PER	-0.418	-0.307	-2.071	0.0	-0.06	0.0	-1.925	-0.47	3.82

Có thể thấy kết quả phán đoán tính ra được cũng khá đúng với thực tế, các tham số ước lượng cho đặc trưng chuyển từ B- hay I- sang I- với cùng loại luôn đạt kết quả cao. Và chuyển từ I- đến I- khác loại luôn có kết quả thấp.

- *Tham số ước lượng cho đặc trưng trạng thái (state feature):*
Chúng em thống kê được hơn 28000 đặc trưng trạng thái đã được sinh ra, trong đó, dưới đây là một số đặc trưng có tham số ước lượng cao nhất và thấp nhất:

	Positive	Negative
y=O	**chunk(0):O**: +4.162 **BOS** : +4.108	**pos(0):Np** : -4.626 **chunk(0):I-NP** : -2.255
y=B-LOC	**wordShape: ULLL ULLLL**: +2.678 **wordShape: ULL ULL** : +2.426	**wordShape:ULLL** : -3.098 **isUpper** : -2.829

y=I-LOC	**w(-1).lower:đường** : +1.208 **w(-2):Bệnh_viện** : +1.152	**chunk(0):B-NP** : -2.674 **pos(0):CH** : -1.072
y=B-MISC	**w(-2)+w(-1):mìn_của** : +1.595 **W(-2):mìn** : +1.585	**isTitle(-1)** : -0.552 **isName** : -0.550
y=I-MISC	**wordShape(-1):LLLLL** : +1.977 **w(-1).lower:người** : +1.343	**chunk(0):B-NP** : -1.469 **isTitle** : -0.577
y=B-ORG	**w(-1).lower:xe** : +1.651 **w(0)[-4:]:đoàn** : +1.512	**chunk(0):I-NP** : -1.585 **wordShape:ULL** : -1.253
y=I-ORG	**isTitle(-1)** : +1.495 **w(-1).lower:báo** : +1.175	**chunk(0):B-NP** : -2.703 **wordShape:ULLL**: -1.117
y=B-PER	**wordShape:ULL** : + 3.556 **wordShape:UL** : +3.295	**chunk(0):I-NP** : -2.553 **pos(-1)+pos(0):Np_Np** : -1.778
y=I-PER	**pos(-1)+pos(0):Np_Np** : +1.297 **chunk(0):I-NP**: +0.822	**chunk(0):B-NP** : -2.081 **w(0)[:2]:Tr** : -1.200

(Danh sách đầy đủ chúng em đã đính kèm cùng với mã nguồn)

Ngoài ra các đặc trưng cho nhãn O, B-PER và B-LOC là nhiều nhất là trọng số cũng khá cao. Điều này là hợp lí vì trong thực tế, các thực thể về người và địa danh, và các từ không thuộc tên riêng xuất hiện là nhiều nhất.

Với những đặc trưng cho giá trị trọng số cao, giá trị trọng số cho biết mô hình sau huấn luyện đã ghi nhớ 1 số đặc trưng trong tập huấn luyện. Ví dụ như B-LOC: **wordShape: ULLL ULLLL**: +2.678 cho biết mô hình có khả năng cao sẽ gán nhãn B-LOC nếu nhận thấy từ ghép bao gồm 1 từ 4 chữ cái, 1 từ 5 chữ cái và cả 2 từ cấu thành nên từ ghép này đều bắt đầu bằng chữ in hoa. Ngược lại với các giá trị trọng số thấp, ví dụ B-ORG: **wordShape:ULL** : -1.253, mô hình có khả năng sẽ không gán tag B-ORG cho những từ gồm 3 chữ cái bắt đầu bằng chữ in hoa.

4. Đánh giá chung

Với kết quả có được, chúng em so sánh với các kết quả có từ các mô hình NER khác trong hội nghị VLSP 2016:

Mô hình	**Precision**	**Recall**	**F1**
Vitk	89.56	89.75	89.66
vie-ner-lstm	91.09	93.03	92.05
NNVLP	92.76	93.0	92.91
Mô hình này	91.65	91.22	91.43

Với điểm F1 đạt 91.43%, tuy không quá ấn tượng nhưng cũng đã thể hiện rõ những ưu điểm của mô hình CRF trong bài toán nhận dạng thực thể có tên, đồng thời cũng phần nào chứng minh được lý do tại sao CRF lại là 1 trong những mô hình được sử dụng rộng rãi nhất trong bài toán này.

5. Xây dựng chương trình ứng dụng

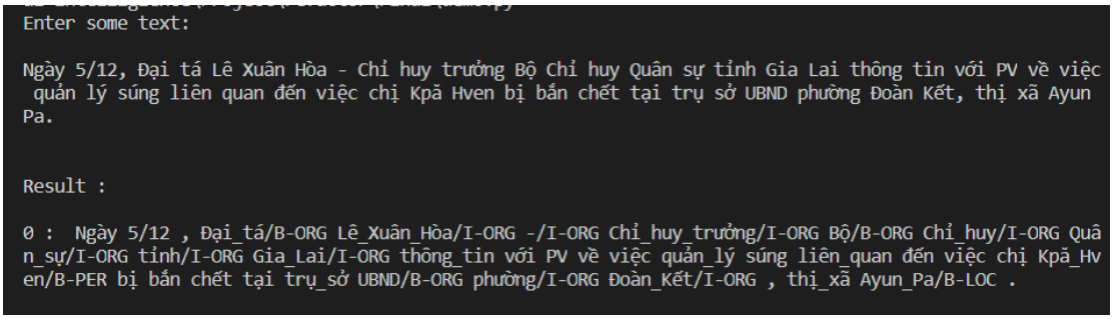
Với nghiên cứu của mình, chúng em sẽ xây dựng một chương trình gán nhãn NER dựa trên mô hình gán nhãn NER nhóm đã nghiên cứu và xây dựng.

Sản phẩm của chúng em sẽ sử dụng model gán nhãn NER mà không có đặc trưng về PoS và chunk. Chúng em sử dụng công cụ ViTokenizer từ thư viện pyvi để tách từ cho văn bản.

Tuy nhiên, khi tách từ, ViTokenizer sẽ tách tên riêng người thành một từ chứ không phải một cụm từ như trong training data. Do đó, để tăng độ chính xác, chúng em đã có bước tiền xử lý training data và test data, ghép lại các cụm từ cho tên người trở về thành 1 từ với nhãn là B-PER và thực hiện train lại ra một model mới dùng cho ứng dụng chúng em xây dựng.

Ứng dụng của chúng em sẽ nhận đầu vào là 1 đoạn văn bản bất kì và đầu ra là văn bản đó với tên thực thể đã được gán nhãn.

Ví dụ Demo:



Đây chỉ là file demo mà bọn em xây dựng nhằm tạo ra cái nhìn trực quan nhất cho mô hình. Thực tế, để tạo ra 1 công cụ gán nhãn hoàn chỉnh vẫn cần rất nhiều thời gian để hoàn thiện và xử lý dấu câu cũng như tách từ để đạt độ chính xác cao hơn.

V. Khó khăn gặp phải trong Project

Trong quá trình xây dựng chương trình ứng dụng, bước đầu, các nhãn gán cho địa danh, tổ chức đều chính xác nhưng nhãn gán cho người có tỉ lệ đúng rất thấp. Cả nhóm đã mất khá nhiều thời gian để tìm ra nguyên nhân. Và khi tìm hiểu kĩ hơn về công cụ ViTokenizer và về data, nhóm mới xác định được vấn đề.

Bài toán nhận dạng thực thể có tên là một lĩnh vực điển hình trong Học máy. Đối với mức độ nhập môn như nhóm em, việc tiếp cận với những phương pháp học máy kinh điển như Gradient Descent hay các lý thuyết sâu về thuật toán tối ưu là khá khó khăn. Sau một khoảng thời gian tìm hiểu, thực sự nhóm em vẫn chưa thể nào hiểu hết lý thuyết về những vấn đề này, đặc biệt là thuật toán tối ưu L-BFGS nên trong báo cáo và tìm hiểu vẫn còn nhiều sai sót.

Quá trình làm việc: việc coding sử dụng thư viện crfsuite khá dễ hiểu và dễ dùng nhưng tìm hiểu lý thuyết cho bài toán này là một khó khăn lớn với chúng em. Nhưng nhóm em đã cố gắng hết sức hoàn thành cơ bản mô hình gán nhãn thực thể. Tuy mô hình còn sơ khai, nhưng em hy vọng sau project này, các thành viên của nhóm đã bổ sung cho bản thân những kiến thức mới, sâu hơn và vô cùng hữu dụng sau này nếu các bạn có ý định theo đuổi lĩnh vực học máy. Với công cụ demo, như nhóm em đã trình bày ở phần trước, đó chỉ là công cụ bọn em sử dụng để tạo ra góc nhìn trực quan nhất. Đây vẫn là công cụ hết sức sơ khai, vẫn còn những vấn đề phân tách dấu câu cần phải xử lý để mô hình thực sự trở thành công cụ có thể sử dụng được trong các bài toán thực tiễn.

VI. Tài liệu tham khảo

- A Feature-Rich Vietnamese Named-Entity Recognition Model – Pham Quang Nhat Minh, 12 March 2018, VLSP 2016: *https://arxiv.org/pdf/1803.04375.pdf*
- Introduction to Conditional Random Fields – Edwin Chen's Blog: *http://blog.echen.me/2012/01/03/introduction-to-conditional-random-fields/?fbclid=IwAR0qvPJ0fogDE8HkgUdUM6OyfDok3VhP0ljja_NcZfZDuY4zPwiamvT8xSk*
- Speech and Language Processing – Dan Jurafsky and James H. Martin – Stanford University, Sep 23, 2018: *https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf*
- Forum machine learning cơ bản – Gradient Descent, First-order and Second order method
- Sklearn-crfsuite documentations – Install instructions, tutorial and API reference: *https://sklearn-crfsuite.readthedocs.io/en/latest/tutorial.html*
  http://www.chokkan.org/software/crfsuite/manual.html
  http://www.chokkan.org/software/crfsuite/
  https://media.readthedocs.org/pdf/sklearn-crfsuite/latest/sklearn-crfsuite.pdf
- Intuitive difference between hidden Markov models and conditional random fields: *https://stats.stackexchange.com/questions/58221/intuitive-difference-between-hidden-markov-models-and-conditional-random-fields?fbclid=IwAR0nTmCFnhutmGacT4SXwCpm76NjmtdwYYcHSNQHjo9CQ_5AISJ9WFIBrIw*
- Eli 5 Named Entity Recognition documentations: *https://eli5.readthedocs.io/en/latest/tutorials/sklearn_crfsuite.html?fbclid=IwAR2KabHzIpcMRiGYH96jl1m6V7VmstJAjzppQGpiqAFzIQq_rTdIFxHf6F0*
- [ML] Tối ưu hàm lỗi với Gradient Descent – Do Minh Hai' Blog *https://dominhhai.github.io/vi/2017/12/ml-gd/?fbclid=IwAR2730sL_CLH-HCnZQkMxNU3VgFKcgHNAX6JrlKeTpOOiFGxc25FTxDO8aY*
- Limited Memory BFGS for nonsmooth optimization – Anders Skajaa, January 2010, Courant Institute of Mathematical Science New York University: *https://cs.nyu.edu/overton/mstheses/skajaa/msthesis.pdf?fbclid=IwAR3tM5cl7jbHNdHawaIlYTXcNHUg-IX_eSpXbPnhfz4CPCSpr6pnCoNJ2i0*
- Feature Selection in Conditional Random Fields for Activity Recognition – Douglas L. Vail, John D. Lafferty, Manuela M. Veloso – Computer Science Department Carnegie Mellon University Pittsburg, PA, USA: *http://www.cs.cmu.edu/~dvail2/pubs/vail07feature_DRAFT.pdf*
- Nhãn từ loại sử dụng trong từ điển hội nghị VLSP (PoS, SubPoS): *https://vlsp.hpda.vn/demo/vcl/PoSTag.htm?fbclid=IwAR3aVb7dXTF5fdPS1_h4zvDuyVWjCaQMlWCbPB3bCFtc1_W1jGCWeRqBpNk*

- Thư viện Pyvi (Python Vietnames toolkit)- Author Viet-Trung Tran, MIT License: *https://pypi.org/project/pyvi/*

- Dữ liệu từ VLSP 2016- underthesea:
https://github.com/undertheseanlp/classification/tree/master/data/corpus

- Conlleval: *https://github.com/pth1993/NNVLP/blob/master/conlleval.pl*