GÁN NHÃN TỪ LOẠI TIẾNG VIỆT TRÊN BỘ DỮ LIỆU VLSP2013

Trần Nhật Nam
19521872

Nguyễn Thành Phúc
19522040

Nguyễn Võ Thiên Ân
19521186

Abstract

Mục tiêu chính của đồ án này là tìm hiểu hai phương pháp gán nhãn phổ biến hiện nay của bài toán Gán nhãn từ loại (POS tagging) trên bộ dữ liệu tiếng Việt. Sau đó chúng tôi tiến hành thực nghiệm một vài kiến trúc mô hình tiêu biểu của hai phương pháp trên và đưa ra kết quả so sánh hiệu suất giữa chúng với mong muốn đưa ra một cái nhìn tổng quát về cách thức hoạt động cũng như hướng phát triển sau này của các phương pháp gán nhãn phục vụ bài toán gán nhãn từ loại tại Việt Nam.

1 Giới thiệu

Đối với việc nghiên cứu xử lý ngôn ngữ tự nhiên (Natural Language Processing), đặc biệt là đối với tiếng Việt thì việc phân đoạn từ (Word Segmentation) và gán nhãn từ loại (POS Tagging) là một tác vụ cần thiết, cơ bản và rất quan trọng trong quá trình xử lý từ loại. Gán nhãn từ loại có lẽ là bài toán sớm nhất được nghiên cứu khi con người tiếp cận với xử lý ngôn ngữ tự nhiên vì mục đích cho ra mô hình để các nhà khoa học (đặc biệt là các nhà khoa học dữ liệu) có được một cách tiếp cận và sử dụng nhanh hơn trong các tác vụ khác của NLP nói riêng cũng như các mô hình học sâu (Deep Learning) khác nói chung.

2 Bài toán gán nhãn từ loại

Trong ngữ pháp, từ loại chính là các thành phần nhỏ của một câu văn hoặc câu nói gồm danh từ (noun), động từ (verb), tính từ (adjective), giới từ (preposition)... Ngoài ra, trong các ngôn ngữ khác nhau có những hình thức từ loại khác nhau. Ví dụ tiếng Nhật có đến ba loại tính từ, tiếng Anh thì chỉ có một; tiếng Trung, tiếng Việt còn có thêm các lượng từ trong khi các ngôn ngữ ở châu Âu lại không có.

Để quá trình xử lý và tiếp nhận ngôn ngữ trong đời sống được chính xác, việc phân biệt rõ các từ loại là điều tối cần thiết cho một ngôn ngữ, và các ngôn ngữ khác nhau nên có các cách phân loại từ khác nhau. Đây chính là Part of Speech (POS) tagging - phân loại các từ loại trong một câu hay nói ngắn gọn là gán nhãn từ loại (Jurafsky, 2000).

Việc phân loại từ như thế này sẽ góp phần giúp cho các chương trình xử lý ngôn ngữ tự nhiên nắm được thêm ý nghĩa của câu thay vì chỉ xem nó như là tập hợp của các ký tự.

3 Phương pháp

Hiện nay có nhiều phương pháp gán nhãn POS khác nhau, ở đề tài này chúng tôi nghiên cứu hai phương pháp tiêu biểu là Dự đoán theo xác suất (Probabilistic Methods) và phương pháp gán nhãn bằng cách sử dụng mạng nơ ron.

3.1 Probabilistic Methods

Phương pháp dự theo xác suất gắn nhãn POS dựa trên xác xuất xảy ra của một chuỗi nhãn cụ thể. Conditional Random Fields (CRFs) và Hidden Markov Models (HMMs) là hai phương pháp phổ biến nhất. Ở đây chúng tôi tìm hiểu phương pháp HMM kết hợp với thuật toán Viterbi để xây dựng mô hình gán nhãn.

3.1.1 Mô hình Markov ẩn

Mô hình Markov ẩn được phát triển từ lý thuyết Xích Markov. Xích Markov là một chuỗi biến đổi trạng thái và trạng thái n chỉ phụ thuộc vào trạng thái n-1, tức trạng thái xảy ra trước đó mà không xét toàn bộ dãy chuyển đổi xảy ra trong quá khứ hay tương lai. Chuỗi Markov được áp dụng cho rất nhiều bài toán thực tế, tính toán được tương lai khi đã biết quá khứ, dự đoán những điều chưa biết dựa trên điều đã biết

Mô hình Markov ẩn là một công cụ thống kê rất mạnh trong việc mô hình hóa các chuỗi có thể sinh ra, hay nói cách khác là các chuỗi mà có thể đặc trưng bởi các chuỗi trạng thái sinh ra các chuỗi

quan sát khác nhau. Mô hình Markov ẩn đã được ứng dụng trong rất nhiều lĩnh vực của xử lý tín hiệu nói chung và xử lý tiếng nói nói riêng, là mô hình tiêu biểu của bài toán POS tagging.

Ta giả sử mô hình Markov ẩn thỏa mãn 2 điều kiện sau:

1. Là mô hình Markov first-order, trạng thái hiện tại chỉ phụ thuộc vào trạng thái liền trước nó, đặc trưng cho tính nhớ của mô hình
2. Quan sát được ở thời điểm t, chỉ phụ thuộc vào trạng thái hiện tại, độc lập với các trạng thái và quan sát được trong quá khứ.

Ta có công thức cho mô hình Markov ẩn như sau:

$$\lambda = (A, B, \pi)$$

S là tập hợp các trạng thái:

$$S = (s_1, s_2, ..., s_N)$$

V là tập hợp tất cả các quan sát được:

$$V = (v_1, v_2, ..., s_M)$$

Q là chuỗi các trạng thái có thể xảy ra, có chiều dài T:

$$Q = q_1, q_2, ..., q_T$$

Và tương ứng với nó là chuỗi O các quan sát có thể quan sát được:

$$O = o_1, o_2, ..., o_T$$

A là bảng chuyển đổi, hay còn gọi là Transition Matrix, chứa những giá trị xác suất chuyển đổi từ trạng thái i sang trạng thái j:

$$A = [a_ij], a_ij = P(q_t = s_j | q_t - 1 = s_i)$$

B là bảng xác suất quan sát, hay còn gọi là Emmission Matrix, chứa những giá trị xác suất của quan sát k từ trạng thái i:

$$B = [b_j(k)], b_i(k) = P(x_t = v_k | q_t = s_i)$$

Π là bảng xác suất đầu tiên:

$$\pi = [\pi_i], \pi_i = P(q_1 = s_i)$$

Cho một mô hình Markov ẩn và một chuỗi quan sát được O, ta có thể tính được $P(O|)$, là xác xuất xuất hiện của chuỗi quan sát đó cho bởi mô hình Markov ẩn. Từ đó ta có thể đánh giá chất lượng mô hình khi dự đoán về chuỗi O cho trước và chọn được mô hình thích hợp nhất (Blunsom, 2004). Tuy nhiên, nếu tính trực tiếp xác suất chuỗi O, lượng phép tính cần dùng là rất lớn. Vì vậy ta có thể sử dụng phương pháp tốt hơn là nhận biết lượng tính toán dư, sau đó đệm chúng để đạt được mục tiêu giảm độ phức tạp trong tính toán. Ta thực hiện đệm bằng biểu đồ mắt cáo cho mỗi bước tính, ta tính giá trị đệm (ký hiệu α) tại mỗi trạng thái bằng cách tổng tất cả trạng thái trước nó. α lúc này là xác suất của chuỗi O tại mức trạng thái S_i tại thời điểm t. Công thức tính xác suất chuỗi O sau khi tối ưu như sau:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

3.1.2 Thuật toán Viterbi

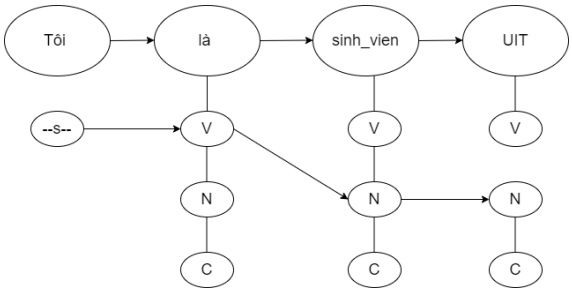
Mục đích của thuật toán Viterbi là xác định được chuỗi trạng thái mà có khả năng đưa ra được chuỗi quan sát cho trước nhiều nhất. Ví dụ trong bài toán POS tagging, sau khi tính được chuỗi các nhãn (chuỗi quan sát) cho một câu trong đoạn văn bản, ta áp dụng thuật toán Viterbi để tìm dãy trạng thái tối ưu. Với bài toán gán nhãn từ loại, thuật toán này dựa trên công thức truy hồi dưới đây:

$$\sigma_t(j) = \max_{1 \le i \le N} [\sigma_{t-1}(i) a_i j] b_j(o_t)$$

$$với\ 2 \le t \le T, 1 \le i \le N$$

$$\psi_t(j) = arg \max_{1 \le i \le N} [\sigma_{t-1}(i) a_i j]$$

$$với\ 2 \le t \le T, 1 \le i \le N$$



Hình 1: Mô tả thuật toán Viterbi trong POS tagging

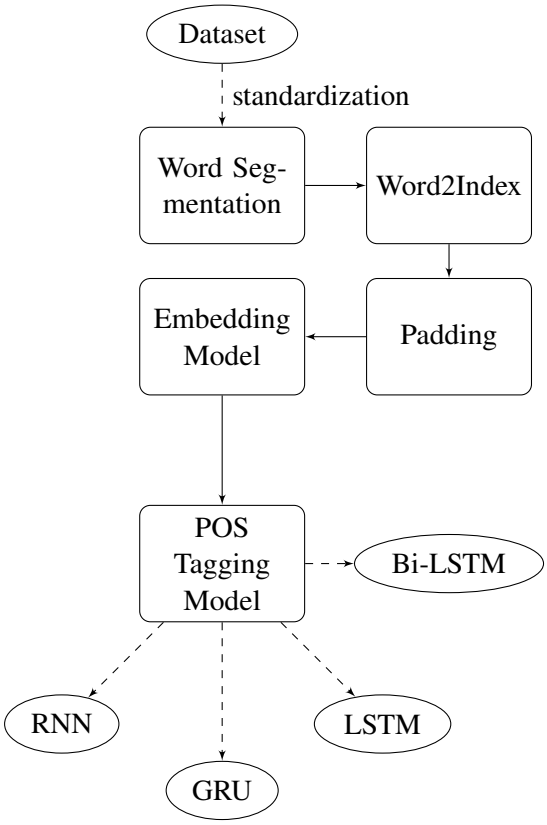
Hình trên đã trực quan hóa cách thuật toán Viterb hoạt động. Kết hợp với mô hình Markov ẩn, thuật toán này có tác dụng tìm ra các đường đi có xác suất cao nhất gồm các nhãn của một câu và trả về đường đi có xác suất cao nhất.

	–s–	A	B	C	CH
–s–	3.588085e-08	0.033369	0.000431	0.072695	0.057158
A	1.909393e-03	0.061838	0.000229	0.046945	0.262474
B	1.382555e-03	0.073204	0.037293	0.031768	0.349438
C	6.462026e-08	0.046204	0.000323	0.016995	0.054863
CH	2.999014e-01	0.030837	0.001063	0.031025	0.091438

Bảng 1: Ví dụ của một Transition Matrix - Xác suất từ một nhãn từ loại sang một nhãn từ loại

	Hải_tặc	tre	xe_đạp	nam	vị_trí
–s–	3.394720e-08	3.394720e-08	3.394720e-08	3.394720e-08	3.394720e-08
V	7.935405e-09	4.762037e-05	7.935405e-09	3.809074e-04	5.396155e-04
CH	1.088133e-08	1.088133e-08	1.088133e-08	1.088133e-08	1.088133e-08
Cc	9.122831e-08	9.122831e-08	9.122831e-08	9.122831e-08	9.122831e-08
E	2.209480e-08	2.209480e-08	2.209480e-08	2.209480e-08	2.209480e-08

Bảng 2: Ví dụ của một Emmision Matrix - Xác suất từ một nhãn từ loại sang một từ



Hình 2: Sơ đồ thể hiện quá trình POS Tagging

3.2 Deep Learning Methods

3.2.1 Embedding Model

Mô hình Embedding hay Word Embedding trong NLP là một không gian vector được dùng để miêu tả các mối quan hệ hay các tương đồng về mặt ngữ cảnh hoặc ngữ nghĩa của dữ liệu. Điểm đặc biệt của Word Embedding là có thể có nhiều chiều dữ liệu và các từ có ý nghĩa gần giống với nhau sẽ có vị trí gần nhau trong không gian này.

Có một số phương pháp Word Embedding thường được sử dụng trong NLP, tiêu biểu như Word2Vec (Church, 2017) hoặc FastText (Athiwaratkun et al., 2018) vì độ chính xác của các phương pháp này được đánh giá cao và tốc độ xử lý nhanh, có thể dễ dàng kết hợp một câu văn bản mới hay thêm vào từ vựng. Trong bài nghiên cứu này chúng tôi sẽ sử dụng cả 2 phương pháp đã giới thiệu và so sánh với phương pháp tự xây dựng mô hình Embedding riêng.

3.2.2 Các kiến trúc Deep Learning

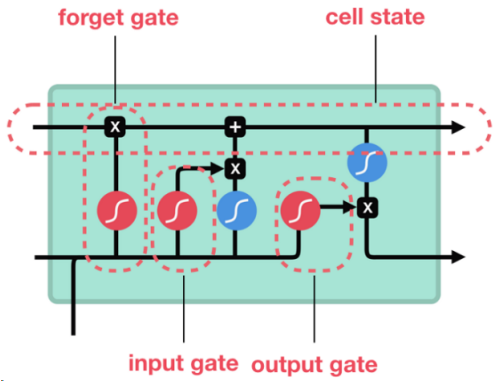
Ngoài cách phổ biến để gán nhãn POS như sử dụng mô hình Markov ẩn, ta có thể làm điểu tương tự bằng cách sử dụng mạng nơ-ron (Recurrent neural network). Bản thân mạng nơ-ron cũng có nhiều biến thể, chúng tôi sử dụng 4 kiến trúc mạng nơ-ron, bao gồm Vanilla RNN, LSTM, GRU và Bi-LSTM. Chúng ta sẽ huấn luyện ma trận words embedding song song với trọng số của mô hình, ta gọi chúng là trọng số words embedding. Trong khi huấn luyện, các trọng số words embedding sẽ được coi là các trọng số bình thường của mạng nơ-ron được cập nhật sau mỗi lần lặp.

- **Vanilla RNN**: Mô hình Recurrent Neural Network thông thường hay còn gọi là Vanilla RNN, vanilla ở đây mang nghĩa là 'basic', 'simple' hoặc 'regular'. RNN coi dữ liệu đầu vào là một chuỗi (sequence) liên tục, nối tiếp nhau theo thứ tự thời gian. Điểm hạn chế của RNN thông thường là nó không thể lưu thông tin của chuỗi dữ liệu có độ dài lớn (short-term memory).

  Để khắc phục nhược điểm này, có hai biến thể của RNN là LSTM (Long Short-Term Memory) và GRU (Gated Recurrent Units) với việc sử dụng cơ chế 'Gates' nhằm bổ sung thông tin mới và loại bỏ thông tin không cần thiết từ 'memory', giúp tăng khả năng lưu trữ thông tin quan trọng của RNN.

- **Long, Short Term Memory (LSTM)**: LSTM và GRU đều có nguyên tắc hoạt động giống như Vanilla RNN, tuy nhiên điểm khác nhau cơ bản giữa chúng là về cấu trúc của các Cell.Trong Vanilla RNN, chúng ta chỉ sử dụng tanh function với dữ liệu đầu vào là Current input (xt) và thông tin lưu trữ từ timestep trước (Hidden state ht-1). Tuy nhiên trong LSTM và GRU, ta sử dụng kết hợp tanh và sigmoid function cùng với các thuật toán để quyết định thông tin nào nên được lưu trữ và thông tin nào nên được loại bỏ.

  Cấu trúc Cell của LSTM được hình thành từ 3 gates: Forget gate, Input gate và Output gate như sau:

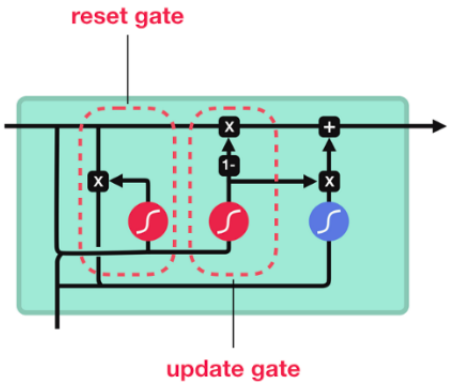


Hình 3: Cấu trúc mạng LSTM

Trong LSTM có 3 input là các gate và 2 output là Cell state và Hidden state. Ngoài việc sử dụng hai dữ liệu đầu vào là Current input (xt) và Hidden state ht-1 như mạng RNN thông thường, LSTM còn sử dụng thêm một đầu vào nữa là Cell state. Có thể xem Cell state là 'memory' của mạng LSTM và nó có thể lưu trữ thông tin của các timestep đầu tiên, do đó hạn chế ảnh hưởng của short-term memory như trong Vanilla RNN.

Cách thức hoạt động của 3 cổng đầu vào như sau: forget gate có nhiệm vụ quyết định liệu thông tin của Cell state ở timestep t-1 có cần được lưu trữ hay không; input gate có nhiệm vụ cập nhật thông tin vào Cell state; output gate có nhiệm vụ tính giá trị của Hidden state cho timestep tiếp theo. Ở đây giá trị của Hidden state mới này cũng chính là giá trị Prediction.

- **Gated Recurrent Unit (GRU)**: Có thể coi GRU là một phiên bản của LSTM với nguyên tắc hoạt động tương tự như LSTM nhưng có cấu tạo đơn giản hơn. GRU kết hợp Cell state và Hidden state thành 1, do đó nó chỉ có 2 input và 1 output. Ngoài ra, trong GRU sử dụng 2 gate là Reset gate và Update gate để quyết định việc lưu trữ và loại bỏ thông tin.



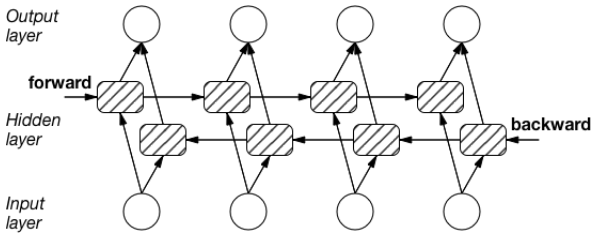
Hình 4: Cấu trúc mạng GRU

Thông thường LSTM có thể lưu trữ thông tin với dữ liệu dài hơn so với GRU. Tuy nhiên do cấu tạo đơn giản của mình, GRU thường xử lý nhanh hơn LSTM và có thể dễ dàng sử dụng để xây dựng các mạng có cấu trúc phức tạp. Cả hai đều có ưu nhược điểm khác nhau, vì thế chung tôi sử dụng cả hai mô hình và xem xét độ thích hợp với bài toán.

Một đặc điểm chung của 3 loại RNN (Vanilla RNN, LSTM, GRU) là chúng đều hoạt động theo một chiều nhất định (forward direction). Hay nói một cách khác, các mạng này chỉ mang thông tin

tính tới thời điểm hiện tại. Tuy nhiên, trong bài toán POS tagging thì việc biết thông tin của các timesteps tiếp theo giúp cải thiện rất nhiều kết quả output. Trong trường hợp này chúng ta có thể sử dụng Bi-directional RNN với việc xử lý thông tin theo cả hai chiều (forward và backward) (Zhou and Wu, 2018). Ở đây chúng tôi sử dụng mô hình Bi-directional LSTM.

- **Bi-directional LSTM (Bi-LSTM)**: Việc nhận dạng chính xác loại từ trong một đoạn văn bản trong nhiều trường hợp phụ thuộc không chỉ vào các thông tin phía trước của từ đang xét mà còn cả các thông tin phía sau. Tuy nhiên, một kiến trúc LSTM truyền thống với một lớp duy nhất chỉ có thể dự đoán nhãn của từ hiện tại dựa trên thông tin có được từ các từ nằm trước đó. Bi-directional LSTM (BiLSTM) đã được tạo ra để khắc phục điểm yếu trên. Một kiến trúc BiLSTM thường chứa 2 mạng LSTM đơn được sử dụng đồng thời và độc lập để mô hình hoá chuỗi đầu vào theo 2 hướng: từ trái sang phải (forward LSTM) và từ phải sang trái (backward LSTM).



Hình 5: Cấu trúc mạng LSTM hai chiều

4 Bộ dữ liệu

```
Việt_Nam          Np
tiếp_tục          V
hội_nhập          V
sâu_rộng          A
vào       E
nền       N
kinh_tế   N
```

Hình 6: Ví dụ một vài từ trong bộ dữ liệu

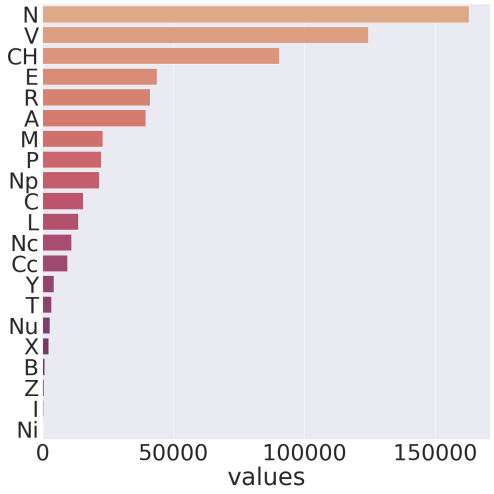
Bộ dữ liệu sử dụng ở bài nghiên cứu này được CLB Xử lý Ngôn ngữ và Tiếng nói tiếng Việt (VLSP) cung cấp. Đây là bộ dữ liệu từ VLSP 2013 - WordSeg & POSTag Task gồm 2 tác vụ chính đó là: Word segmentation và POS tagging. Tuy nhiên ở bài nghiên cứu này sẽ tập trung chính vào xây dựng mô hình gán nhãn từ loại cho tiếng Việt.

Nguồn thu thập: từ các bài báo online trong tiếng Việt.

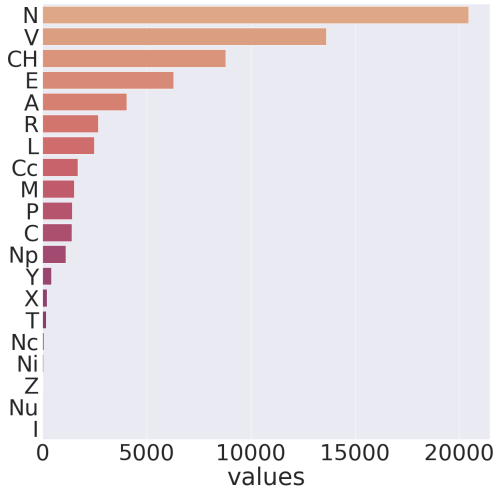
Link: https://vlsp.org.vn/vi/node/80

Nhãn	Tên	Train	Test
N	Danh từ	162847	20456
Np	Danh từ riêng	21525	1117
Nc	Danh từ chỉ loại	11048	47
Nu	Danh từ đơn vị	2769	2
Ni	Danh từ kí hiệu	68	1
V	Động từ	124430	13627
A	Tính từ	39280	4036
P	Đại từ	22270	1426
L	Định từ	13518	2469
M	Số từ	22995	1509
R	Phó từ	41003	2668
E	Giới từ	43672	6278
C	Liên từ	15475	1401
Cc	Liên từ đẳng lập	9374	1696
I	Thán từ	213	1
T	Trợ từ	3324	164
B	Từ vay mượn	724	0
Y	Từ viết tắt	4306	418
X	Các từ không thể phân loại	2275	204
Z	Yếu tố cấu tạo từ	354	13
CH	Nhãn dành cho các loại dấu	90313	8788

Bảng 3: Danh sách từ loại và số lượng điểm nhãn



Hình 7: Biểu đồ biểu diễn phân bố số lượng từ trên các nhãn trong tập Train



Hình 8: Biểu đồ biểu diễn phân bố số lượng từ trên các nhãn trong tập Test

5 Mô hình thực nghiệm

5.1 Xử lý dữ liệu

Trước khi tiến hành chạy thử nghiệm, chúng tôi thực hiện các bước xử lý bộ dữ liệu để có thể đưa vào mô hình. Các bước xử lý gồm chuẩn hóa, phân tách từ, padding đầu vào và chuẩn hóa dữ liệu đầu ra.

Theo như mô tả, bộ dữ liệu VLSP 2013 có 21 nhãn từ loại. Tuy nhiên trong tập test của bộ dữ liệu này lại lên đến 32 nhãn từ loại. Nguyên nhân là do những phần nhãn 'Y' có thể được chia ra thành các nhãn nhỏ khác nhau, vì vậy chúng ta sẽ tiến hành định dạng lại về nhãn Y.

Ngoài ra, do môi trường thực nghiệm (Google Colab) khi đọc dữ liệu đã hiểu nhầm từ 'nan' trong tiếng Việt (nan hoa, nan tre...) là giá trị null, tương đồng với 'np.nan' trong thư viện Numpy, do đó cần gán lại cho một số điểm dữ liệu này bằng chuỗi 'nan' để phù hợp với kiểu dữ liệu.

5.2 Các mô hình Deep Learning

Chúng tôi sẽ sử dụng 4 phiên bản của mô hình RNN là Vanilla RNN, LSTM, GRU và Bi-LSTM đã mô tả ở trên kết hợp cùng 3 bộ words bedding gồm tự thu thập, FastText và Word2vec để tiến hình xây dựng mô hình, các mô hình sẽ có 3 lớp:

- Lớp đầu tiên: 1 lớp Embedding nhúng các câu trong tập train thành các vector.
- Lớp thuật toán: Sử dụng các thuật toán ở trên.
- Lớp ouput: Sử dụng lớp TimeDistributed để output ra kết quả.

Cùng với mô hình chúng tôi compile chung với các thông số sau.

- Optimizer: sử dụng thuật toán phương pháp tối ưu Adam.
- Learning rate: khởi tạo bằng 0.01
- Loss: Sử dụng hàm CategoricalCrossentropy()
- Metric: Trong bài toán sử dụng độ đo F1 để đánh giá chính xác nhất

Sau khi kết hợp các bước trên ta thu được 12 mô hình ứng với 4 kiến trúc Deep Learning và 3 bộ words embedding.

5.3 Mô hình HMM

Quá trình thực nghiệm mô hình Markov ẩn có 2 bước:

- **Bước 1:** Xây dựng Transition Matrix A và Emission Matrix B
- **Bước 2:** Thuật toán Viterbi:
  - Khởi tạo: khởi tạo 2 ma trận $bestpaths$ và $bestprobabilities$ sẽ được dùng cho hàm $feedforward$ 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

6 Kết quả thực nghiệm

	Manual	**FastText**	**Word2Vec**
RNN	0.7452	0.8299	0.7934
LSTM	0.7303	0.764	0.7877
GRU	0.66	0.6222	0.7797
Bi-LSTM	0.7926	0.7866	0.8527
HMM	0.1459	0.1324	0.3754

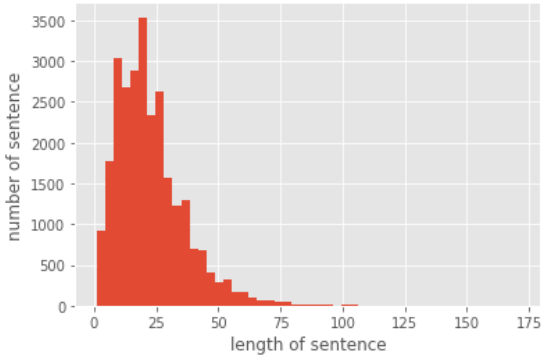
Bảng 4: Độ đo f1 của các mô hình

Dựa vào kết quả của các độ đo f1, dù là một thuật toán cơ bản và phổ biến trong POS tagging nhưng ma trận HMM cùng thuật toán Viterbi cho kết quả khá thấp. Khi áp dụng những kiến trúc Deep Learning thì kết quả hoàn toàn khả quan hơn rất nhiều, cuối cùng thu được mô hình với kiến trúc Bi-LSTM kết hợp với bộ Word2Vec có kết quả tốt nhất.

Đầu tiên, trên bộ Embedding chúng tôi tự xây dựng, kết quả ở các mô hình Deep Learning xấp

xỉ 0.7 đến gần 0.8. Đây có thể được xem là kết quả khá tốt giữa bộ Embedding tự xây dựng và bộ Embedding có sẵn. Mô hình GRU cho ra kết quả không cao cho thấy kiến trúc của GRU không phù hợp với bài toán đã đặt ra (GRU không có cơ chế ghi nhớ các quan sát trước đó). Ngược lại Bi-LSTM có kết quả cao do đã khắc phục được hạn chế của GRU khi có thể lưu trữ được các quan sát, thích hợp cho các dữ liệu đầu vào dài hơn và có thể xử lý các từ mới một cách dễ dàng hơn.

Khi sử dụng bộ Embedding có sẵn (Pre-trained Embedding) như FastText và Word2Vec, kết quả ở các mô hình đã có cải thiện hơn so với trước đó. Đặc biệt cao nhất ở RNN (FastText) với 0.82 và Bi-LSTM (Word2Vec) với 0.85. Kết quả được cải thiện vì đa số các pre-trained embedding đã được huấn luyện tốt với nguồn dữ liệu đạt chuẩn với số chiều của các bộ Embedding lên đến 300 chiều và từ điển lên đến 1,5 triệu từ. Bộ FastText có số lượng từ vựng ít hơn nên mô hình RNN đã xử lý tốt hơn so với các mô hình khác trên cùng bộ Embedding này. RNN tuy không có cơ chế lưu trữ thông tin nhưng có thể xử lý tốt với đầu vào ngắn, ít. Ngược lại Word2Vec có số lượng từ nhiều hơn nên Bi-LSTM đã có thể phát huy lợi thế lưu trữ được thông tin và cuối cùng đạt kết quả tốt nhất trong các mô hình.



Hình 9: Biểu đồ thể hiện phân bố số lượng câu trên tập test

Ở mô hình cho kết quả tốt nhất là Bi-LSTM, ta thấy độ chính xác trên từng nhãn rất cao, chỉ có ba nhãn có độ chính xác bé hơn 50%, các nhãn còn lại có độ chính xác trải dài từ 70 đến 100%. Với 3 nhãn có độ chính xác thấp là nhãn B (0%), Nc (32%) và Nu (40%), các nhãn này có số lượng từ trong tập test chiếm rất ít, lần lượt là 0 từ, 2 từ và 47 từ, kể cả trong tập huấn luyện các nhãn này chỉ chiếm số lượng chưa đến 0.02%.

Ta có thể đưa ra nhận xét rằng các nhãn có số lượng từ ít hơn thường cho kết quả gán nhãn kém khả quan hơn các nhãn có số lượng từ đa dạng và phong phú. Điều này thể khắc phục được bằng cách làm giàu thêm bộ dữ liệu để việc huấn luyện mô hình trở nên hiệu quả hơn.

Nhãn	Precision	F1
Nu	40	57
A	91	87
T	89	77
Cc	99	99
N	93	94
E	96	97
Ni	100	98
V	94	93
X	80	82
Z	79	81
M	99	95
L	100	100
Y	91	70
R	90	92
CH	98	99
I	100	100
B	0	0
P	99	99
C	90	90
Nc	32	47
Np	83	75

Bảng 5: Kết quả độ đo Precision và F1 của từng nhãn trên mô hình Bi-LSTM kết hợp với bộ Word2Vec. (Đơn vị: %)

7 Kết luận

Trong nghiên cứu này, nhóm chúng tôi đã tiến hành thực nghiệm các mô hình phổ biến trong xử lý ngôn ngữ tự nhiên như GRU, LSTM, Bi-LSTM để gán nhãn từ loại tiếng Việt dựa trên bộ dữ liệu VLSP2013. Ngoài ra chúng tôi cũng sử dụng thêm một mô hình xác suất đặc biệt đó là Hidden Markov kết hợp với thuật toán Viterbi để tìm được đường đi (dãy) của các nhãn từ loại. Kết quả thu được khá tốt ở các mô hình với độ đo f1 xấp xỉ 0.7 đến 0.8. Dựa vào kết quả này, chúng tôi bước đầu đã xây dựng được nền tảng cho tác vụ gán nhãn từ loại tiếng Việt, trong tương lai chúng tôi sẽ nghiên cứu các hướng phát triển mô hình mới phù hợp hơn nhằm tăng độ chính xác cho kết quả.

References

Ben Athiwaratkun, Andrew Gordon Wilson, and Anima Anandkumar. 2018. Probabilistic fasttext for multi-sense word embeddings. *arXiv preprint arXiv:1806.02901*.

Phil Blunsom. 2004. Hidden markov models. *Lecture notes, August*, 15(18-19):48.

Kenneth Ward Church. 2017. Word2vec. *Natural Language Engineering*, 23(1):155–162.

Dan Jurafsky. 2000. *Speech & language processing*. Pearson Education India.

Qimin Zhou and Hao Wu. 2018. Nlp at iest 2018: Bilstm-attention and lstm-attention via soft voting in emotion classification. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 189–194.