Thêm Dấu Cho Văn Bản Tiếng Việt Sử Dụng Phương Pháp Học Sâu

Trần Quang Linh[1,2,*], Dương Văn Bình[1,2,*], Lâm Gia Huy[1,2,*], Đỗ Trọng Hợp[1,2,†]

[1] Trường Đại học Công nghệ Thông tin
[2] Đại học Quốc Gia Thành Phố Hồ Chí Minh
Email: {18520997, 18520505, 18520832}@gm.uit.edu.vn,*
†hopdt@uit.edu.vn

Tóm tắt nội dung Trong bài báo này, chúng tôi đề xuất phát triển một công cụ cho việc tự động thêm dấu Tiếng Việt sử dụng các thuật toán học sâu trong xử lý ngôn ngữ tự nhiên. Chúng tôi đã tiến hành thực nghiệm trên nhiều phương pháp học sâu như Gated Recurrent Unit, Bidirectional Long-short Term Memory, Bidirectional Gated Recurrent Unit. Kết quả thực nghiệm của chúng tôi cho thấy thuật toán Bidirectional Gated Recurrent Unit cho kết quả tốt nhất với Blue-score đạt 88.06%. Cùng với đó, chúng tôi cũng thực nghiệm trên 2 phương pháp tách từ cho tiếng Việt và kết quả đạt được là cách tách từ theo khoảng trắng cho kết quả tốt hơn. Việc đưa công cụ thêm dấu này vào thực tế sẽ tạo một sự cải tiến trong cách giao tiếp bằng nhắn tin, giúp mọi người tiết kiệm thời gian trong đánh máy nhưng vẫn đảm bảo người nhận hiểu đúng nghĩa của câu.

Keywords: Thêm Dấu Tiếng Việt · Mạng Nơ-Ron · Dịch Máy · Xử Lý Ngôn Ngữ Tự Nhiên .

1 Giới thiệu

Tiếng Việt là một trong những ngôn ngữ khó học trên thế giới, điều này đã gây nhiều trở ngại cho những người mới bắt đầu học tiếng Việt. Một trong những nguyên nhân chính khiến tiếng Việt khó là bộ dấu trong tiếng Việt. Không giống như tiếng Anh, tiếng Việt có năm dấu thanh là huyền, sắc, hỏi, ngã, nặng. Mỗi từ khi được ghép vào một dấu khác nhau sẽ tạo thành một từ mới với nghĩa nhau (Ví dụ: từ tư thêm dấu huyền thành từ từ hoặc từ tư thêm dấu sắc thành từ tứ). Vì vậy, trong bài báo này, chúng tôi xin giới thiệu một công cụ thêm dấu cho tiếng Việt để giúp người mới học tiếng Việt bắt đầu dễ dàng hơn cũng như giúp đơn giản hóa quá trình đánh máy tiếng Việt.

Với sự phát triển của trí tuệ nhân tạo nói chung và xử lý ngôn ngữ tự nhiên nói riêng, các vấn đề trong cuộc sống xã hội đã dần được máy tính giải quyết như nhận dạng bình luận độc hại[1][2], tổng hợp tiếng nói[3], v.v. Tuy nhiên, nguồn dữ liệu tiếng Việt gặp phải nhiều vấn đề như từ viết tắt, teencode, và đặc biệt là từ không dấu, làm ảnh hưởng đến chất lượng các mô hình. Đặc biệt là

trong bài toán nhận dạng bình luận, nơi mà rất nhiều người dùng mạng xã hội sử dụng các từ không dấu để bình luận. Vì vậy, chúng tôi đã xây dựng công cụ thêm dấu cho tiếng Việt để có thể áp dụng vào bước tiền xử lý cho dữ liệu không dấu, giúp tạo một bộ dữ liệu sạch, có đầy đủ dấu và hoàn chỉnh để làm đầu vào cho các bài toán như phân tích sắc thái văn bản hay dịch máy trên tiếng Việt.

Với sự phát triển của công nghệ và áp lực đẩy nhanh công việc, một bộ phận người chọn cách đánh máy không dấu để đẩy nhanh tốc độ đánh máy, đặc biệt là những người làm việc cần giao tiếp bằng hai ngôn ngữ Anh-Việt (vì khi chuyển chế độ gõ trên unikey tốn thời gian và thường dễ gây nhầm lẫn). Nhưng việc đánh máy không dấu cũng gây nhiều hậu quả khá nghiêm trọng như gây hiểu nhầm, hiểu sai ý người nói. Chính vì những vấn đề này mà chúng tôi đề xuất một công cụ để chuyển đổi từ câu không dấu thành một câu có dấu hoàn chỉnh. Như vậy, có thể giải quyết bài toán vừa đẩy nhanh tốc độ đánh máy khi đánh máy không dấu, vừa đảm bảo tính chính xác nghĩa của câu.

Ở các phần sau của bài báo, chúng tôi sẽ giới thiệu các công trình liên quan trong Mục 2. Bộ dữ liệu được sử dụng trong thực nghiệm được trình bày ở Mục 3. Trong Mục 4, chúng tôi giới thiệu các phương pháp sử dụng trong bài toán. Chuẩn bị các mô hình huấn luyện sẽ được trình bày trong Mục 5. Phần kết quả sẽ được trình bày trong Mục 6 và phần cuối cùng của bài báo sẽ đưa ra kết luận và hướng phát triển.

2 Các công trình nghiên cứu liên quan

Với đề tài này, đã có một số công trình nghiên cứu liên quan sử dụng những phương pháp khác nhau và cũng đã đạt được một số thành quả ban đầu.

Luan Nghia Pham và các cộng sự với bài báo Vietnamese Text Accent Restoration With Statistical Machine Translation[4]. Đây là bài báo sử dụng framework MOSES toolkit để thực hiện dịch từ câu không dấu sang câu có dấu trong Tiếng Việt. Bài báo này sử dụng bộ dữ liệu gồm 206000 câu Tiếng Việt và hệ số Accuracy đạt 92%. Nhưng nhược điểm lớn nhất của MOSES toolkit là cần cấu hình máy tính mạnh và thời gian huấn luyện mô hình rất lâu. Vì vậy đây là một trong những khía cạnh mà chúng tôi mong muốn sẽ cải thiện.

Thai Hoang Pham và các cộng sự với bài báo On the Use of Machine Translation-Based Approaches for Vietnamese Diacritic Restoration[5]. Đây là bài báo sử dụng framework OpenNMT toolkit để dựa vào các thuật toán mạng nơ-ron. Toolkit này sử dụng mạng Encoder-Decoder để dịch từ câu không dấu sang câu Tiếng Việt có dấu. Độ chính xác của mô hình dịch máy ở bài báo này đạt 96.15%. Quá trình huấn luyện cho mô hình này mất 8 tiếng với một bộ dữ liệu 140474 câu Tiếng Việt.

Nhìn chung, mặc dù các bài báo này đã đạt hiệu suất rất cao nhưng đều dựa vào các phương pháp không còn mới, dẫn đến cần nhiều tài nguyên tính toán và thời gian huấn luyện mô hình. Thêm vào đó là bộ dữ liệu không đủ lớn nên có thể không bao quát được hết tất cả các trường hợp của Tiếng Việt. Vì vậy, chúng tôi muốn xây dựng một mô hình dịch máy dựa trên một bộ dữ liệu lớn và sử dụng những phương pháp hiện đại để giải quyết bài toán.

3 Bộ dữ liệu

3.1 Thông tin chung

Bộ dữ liệu mà chúng tôi sử dụng để huấn luyện và đánh giá mô hình có tên là A Large-scale Vietnamese News Text Classification Corpus[3], được công bố trong bài báo A Comparative Study on Vietnamese Text Classification Methods[6]. Bộ dữ liệu này có hơn 26451 bài báo Tiếng Việt được phân vào 27 chủ đề là bóng đá, âm nhạc, giải trí, v.v. Mục đích ban đầu của bộ dữ liệu này là để phân loại văn bản nhưng chúng tôi nhận thấy đây là một bộ dữ liệu khá lớn, bao quát đầy đủ các lĩnh vực của đời sống xã hội nên chúng tôi tiến hành xử lý bộ dữ liệu để phù hợp cho bài toán của chúng tôi.

3.2 Xây dựng dữ liệu huấn luyện

Chúng tôi đã xử lý tách các câu trong bộ dữ liệu A Large-scale Vietnamese News Text Classification Corpus theo các dấu hết câu ('.') và có được 542107 câu Tiếng Việt. Chúng tôi tiếp tục một số xử lý trên bộ dữ liệu này. Đầu tiên, loại bỏ các kí hiệu đặc biệt cũng như các chữ số trong câu vì các thành phần này không ảnh hưởng đến dấu câu. Sau đó, chúng tôi tiếp tục bỏ các từ Tiếng Anh trong câu. Cuối cùng, chúng tôi chuyển tất cả các từ có chữ cái in hoa thành in thường vì các từ này có nghĩa giống nhau nhưng khác về cách biểu diễn. Chúng tôi sử dụng thư viện unidecode[4] để loại bỏ dấu câu trên 542107 câu này và có được một bộ dữ liệu với dữ liệu là các câu không dấu và nhãn là các câu có dấu tương ứng. Chúng tôi tiến hành phân chia bộ dữ liệu thành các tập huấn luyện, tập kiểm thử và tập đánh giá với kích thước lần lượt là 500000 câu, 21107 câu và 21000 câu. Trong Bảng 1 là một vài điểm dữ liệu trong bộ dữ liệu sau khi xử lý.

Bảng 1: Bảng ví dụ một vài điểm dữ liệu trong bộ dữ liệu đã xử lý

STT	Dữ liệu được tách thành các câu	Dữ liệu sau khi bỏ dấu
1	tại câu lạc bộ nguyễn du	tai cau lac bo nguyen du
2	tham gia chương trình lần này có ca sĩ hồ ngọc hà	tham gia chuong trinh lan nay co ca si ho ngoc ha
3	khéo léo kết hợp thành viên chủ chốt của nhóm tay cự phách	kheo leo ket hop thanh vien chu chot cua nhom tay cu phach
4	âm thanh hiện đại nhất trong các đĩa nhạc từng phát hành	am thanh hien dai nhat trong cac dia nhac tung phat hanh

[3] https://github.com/duyvuleo/VNTC
[4] https://pypi.org/project/Unidecode/

4 Phương pháp tiếp cận

Chúng tôi tiếp cận bài toán thêm dấu tương tự như bài toán dịch máy, nhưng điểm khác biệt là đầu vào của bài toán luôn luôn có độ dài bằng với đầu ra.

Với mục tiêu tìm ra được phương pháp tốt nhất cho việc phát triển một mô hình học sâu trên bộ dữ liệu trên, chúng tôi đã áp dụng hai phương pháp tách từ khác nhau để xử lý trên dữ liệu: tách từ theo khoảng trắng và sử dụng thư viện tách từ vncorenlp để tách thành các từ đơn và từ ghép. Và chạy thử nghiệm trên nhiều biến thể của các mô hình dịch máy khác nhau để có được những đánh giá khách quan nhất. Kết quả từ các bộ tách từ sẽ được sử dụng cho việc xây dựng đầu vào dùng cho bước huấn luyện mô hình.

4.1 Tiền xử lý dữ liệu

Với bộ dữ liệu ban đầu, sau khi xử lý để tạo ra các điểm dữ liệu đơn lẻ, chúng tôi tiến hành tạo bộ dữ liệu sạch để huấn luyện mô hình như Mục 3.2. Tuy nhiên, nhằm có thêm những đánh giá không chỉ trên việc xây dựng mô hình mà còn trên phương diện xử lý dữ liệu nên chúng tôi có thêm một bước áp dụng kỹ thuật tách từ trên các điểm dữ liệu trước khi loại bỏ dấu của chúng để tạo dữ liệu huấn luyện.

4.1.1 Tách từ theo khoảng trắng: Ưu điểm của phương pháp này là nhanh, đơn giản và phổ biến khi xử lý dữ liệu đầu vào có dạng chuỗi. Ví dụ khi tách từ: 'trường đại học công nghệ thông tin'->['trường', 'đại', 'học', 'công', 'nghệ', 'thông', 'tin'].
Việc sử dụng phương pháp trên có hạn chế là làm cho ý nghĩa các cặp từ ghép bị mất đi, ví dụ như 'đại học' hay 'công nghệ thông tin' khi các từ đơn lẻ đứng cạnh nhau sẽ tạo nên một ý nghĩa riêng biệt so với khi đứng đơn lẻ. Với phương pháp này tập huấn luyện có kích thước tập từ vựng là 1625 từ.

4.1.2 Tách từ theo loại từ Tiếng Việt: Để khắc phục những hạn chế của phương pháp tách từ thứ nhất và việc giữ được ý nghĩa của các từ ghép trong câu cũng giúp mô hình hiểu được ngữ nghĩa của các cặp từ ghép trong Tiếng Việt. Chúng tôi đã sử dụng thư viện vncorenlp[5] để tiến hành tách một câu thành các cặp từ đơn và ghép như: ['trường', 'đại_học', 'công_nghệ_thông_tin'].
Với phương pháp này số lượng từ cho việc huấn luyện mô hình cũng tăng đáng kể (kích thước của bộ từ vựng trên tập huấn luyện là 27692 từ).

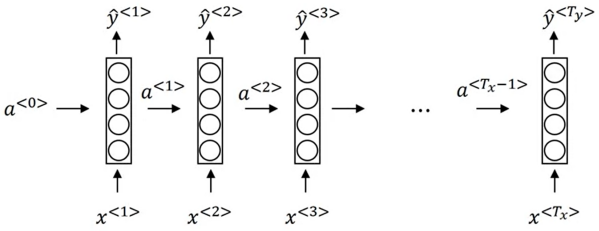
4.2 Mô hình dịch máy

4.2.1 Kiến trúc mạng Recurrent Neural Networks[7][8][9][10]: RNN là một kiến trúc mạng học sâu xử lý dạng chuỗi thông tin được dùng cho các bài toán có dữ liệu đầu vào dạng chuỗi và được ứng dụng rất nhiều trong việc

[5] https://github.com/vncorenlp/VnCoreNLP

giải quyết các bài toán xử lý ngôn ngữ tự nhiên cũng như các bài toán xử lý âm thanh.

Kiến trúc cơ bản của RNN[6] là việc cho phép các đầu ra của lớp nơ-ron trước trở thành đầu vào của lớp kế tiếp. Ưu điểm của kiến trúc mạng này là có thể xử lý chuỗi đầu vào có độ dài bất kỳ, kích thước mô hình không thay đổi, thông tin của các lớp trước sẽ có ích cho các lớp sau và các bộ trọng số được tái sử dụng.



Hình 1: Kiến trúc mạng RNN.

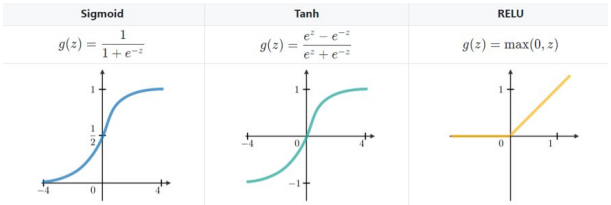
Trong đó, hàm kích hoạt cho mỗi trạng thái t với mỗi đầu vào x^t:

$$a^{<t>} = g(W_{aa}a^{t-1} + W_{ax}x^t + b_a), \tag{1}$$

và đầu ra của mỗi trạng thái:

$$y^{<t>} = g(W_{ya}a^t + b_y). \tag{2}$$

Với g là hàm phi tuyến, có thể là hàm Tanh, RELU hoặc Sigmoid.



Hình 2: Các hàm phi tuyến phổ biến.

Tuy nhiên, ở bất cứ mô hình nào cũng sẽ có những hạn chế, với kiến trúc RNN cơ bản thì việc tính toán sẽ chậm, khó khăn trong việc truy tìm lại các thông tin từ những từ đã duyệt qua quá lâu ở các trạng thái trước đó (vấn đề phụ thuộc xa[11]). Do đó những phần tử đầu tiên của chuỗi đầu vào sẽ không có nhiều ảnh hưởng và không thể kiểm soát đầu vào cho trạng thái hiện tại.

[6] https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

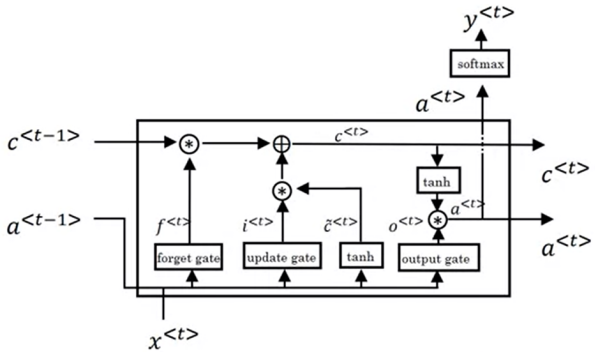
Có nhiều loại RNN khác nhau (*one to one, one to many, many to many,...*) được ứng dụng để giải quyết những bài toán khác nhau. Loại RNN được sử dụng trong bài toán dịch máy là *many to many*.

Vì những đặc trưng của RNN khi xử lý dữ liệu nên gặp phải hai vấn đề chính trong khi huấn luyện mô hình trên dữ liệu[11] là *vanishing gradient*[12] và *exploding gradient*. Với *exploding gradient* thì có thể khắc phục bằng kỹ thuật là *gradient clipping*[13] trong khi đó LSTM và GRU đã được giới thiệu để giải quyết vấn đề *vanishing gradient*.

4.2.2 Long Short-Term Memory Unit (LSTM): Được giới thiệu lần đầu tiên bởi Hochreiter và Schmidhuber[14] đã tạo nên nhiều cải tiến cho các mô hình mạng RNN truyền thống. Khi sử dụng LSTM, cấu tạo các cổng (gate) sẽ có thể được điều chỉnh khác nhau tùy mục đích để có được các kết quả thử nghiệm khác nhau với mục tiêu xây dựng được mô hình tốt nhất.

Việc áp dụng LSTM cho các bài toán dịch máy đã đạt được những kết quả khả quan qua một dẫn chứng là nhóm tác giả trong bài báo LSTM Neural Networks For Language Modeling[15] đã tiến hành thử nghiệm và đạt kết quả cải thiện 8% so với mô hình dịch máy thông thường.

Cấu tạo của LSTM cơ bản[7] gồm có các cổng là *update gate, forget gate* và *output gate* với các chức năng tương ứng: chọn lọc những thông tin được thêm vào bộ nhớ, loại bỏ thông tin không cần thiết, xác định những thông tin được chọn làm đầu ra. Việc tinh chỉnh trong LSTM chính là việc điều chỉnh các cổng này (thêm cổng hay điều chỉnh hàm tính toán trong các cổng). Các thông tin từ mỗi trạng thái sẽ được lưu dữ lại tại các *memory cell* c^t.



Hình 3: Kiến trúc của LSTM.

Các thông tin có thể được lưa trữ vào *cell* từ đầu vào x^t:

$$\tilde{c}^t = tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c), \tag{3}$$

[7] https://www.coursera.org/lecture/nlp-sequence-models/long-short-term-memory-lstm-KXoay

Cổng *update gate* được tính toán như sau:

$$\Gamma_u = \sigma(W_u[c^{t-1}, x^{<t>}] + b_u), \tag{4}$$

Cổng *forget gate* có công thức:

$$\Gamma_f = \sigma(W_f[a^{t-1}, x^t] + b_f), \tag{5}$$

Công thức tổng quát của *output gate*:

$$\Gamma_o = \sigma(W_o[a^{t-1}, x^t] + b_o), \tag{6}$$

Memory cell được cập nhật theo công thức:

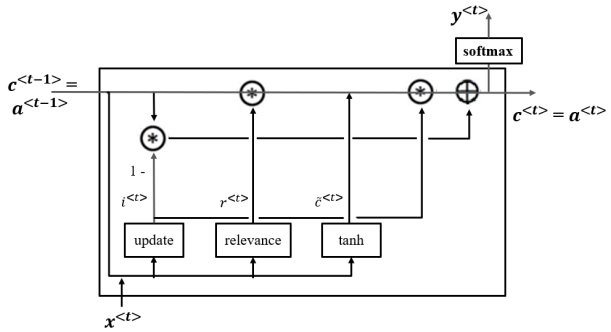
$$c^{<t>} = \Gamma_u * \tilde{c}^t + \Gamma_f * c^{t-1}, \tag{7}$$

Hàm kích hoạt:

$$a^{<t>} = \Gamma_o * tanh(c^t). \tag{8}$$

4.2.3 Gated Recurrent Unit (GRU): Là một kiến trúc tương tự như LSTM được giới thiệu để khắc phục những hạn chế của mạng RNN truyền thống. Cũng đã được thử nghiệm và đánh giá trong bài báo Empirical Evaluation Of Gated Recurrent Neural Networks On Sequence Modeling[16]. GRU cũng có cấu tạo là các cổng như LSTM tuy nhiên hạn chế hơn là chỉ có hai cổng là *update gate* và *relevance gate* vì vậy việc tính toán trong GRU cũng đơn giản hơn. Ở GRU có cổng *relevance* với mục đích tìm ra thông tin phía trước có quan trọng hay không để loại bỏ. Trong GRU, *memory cell* và hàm kích hoạt là như nhau:

$$c^{<t>} = a^{<t>}, \tag{9}$$



Hình 4: Kiến trúc của GRU.

Cổng *relevance gate* được tính như sau:

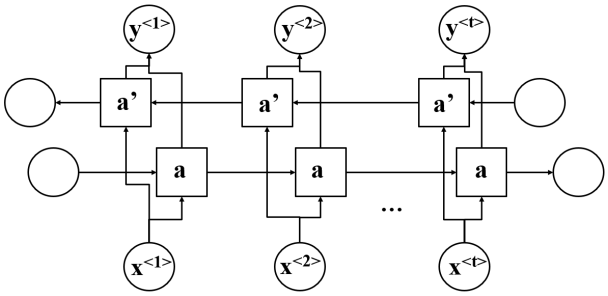
$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_c), \tag{10}$$

Các $\tilde{c}$ và $c^{<t>}$ cũng sẽ thay đổi:

$$\tilde{c}^t = tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c), \tag{11}$$

$$c^{<t>} = \Gamma_u * \tilde{c}^t + (1 - \Gamma_u) * c^{t-1}. \tag{12}$$

4.2.4 Bidirectional RNN[17]: Ý tưởng của Bidirectional RNN (BRNN) cũng giống như bài tập điền từ vào vị trí trống trong môn Tiếng Anh, để tìm được từ thích hợp để điền thì ta cần phải xem xét cả từ phía trước chỗ trống lẫn phía sau để có thể dữ đoán chính xác từ loại (danh từ, tính từ, động từ hay trạng từ) và lựa chọn từ có ý nghĩa phù hợp. Một mạng BRNN cơ bản gồm 2 mạng RNN độc lập nhau và kết quả của 2 mạng RNN này sẽ được tổng hợp thành một kết quả duy nhất cho mỗi trạng thái của mạng.



Hình 5: Kiến trúc mạng BRNN.

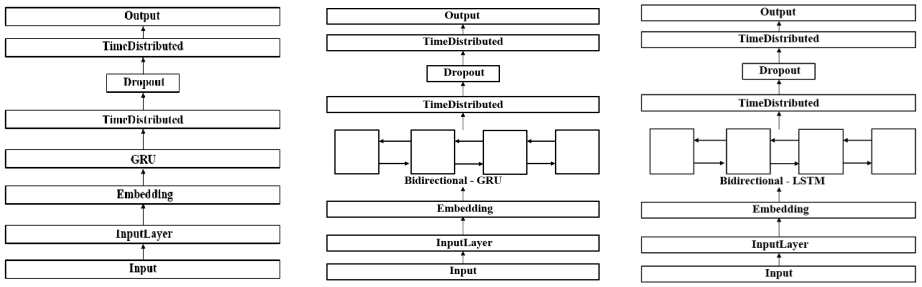
Với BRNN đầu ra của trạng thái hiện tại sẽ phụ thuộc vào thông tin phía trước và sau trạng thái đó. Vì vậy, với bài toán của chúng tôi thì việc thêm dấu cho một từ bất kỳ trong một câu cần thông tin của toàn bộ các từ xung quanh để điền dấu phù hợp để tạo ra một câu có ý nghĩa. Lấy ví dụ như chữ 'an' nếu chỉ đứng một mình thì sẽ có nhiều cách thêm dấu khác nhau ('ăn', 'án', 'ân', v.v) nên nếu chỉ dùng RNN thì mô hình sẽ điền dấu cho chữ này khi duyệt qua mà không cần biết chữ tiếp theo có phù hợp với chữ đã được điều dấu hay không (trong hoàn cảnh xấu hơn nếu chữ này bị điền sai dấu với dấu được kỳ vọng, có thể chữ tiếp theo sẽ bị điền dấu sai ý nghĩa để phù hợp với chữ đã được điền dấu trước đó), việc này sẽ gây ra phản ứng di truyền dẫn đến ý nghĩa một câu sẽ bị sai lệch so với kỳ vọng ban đầu. Trong trường hợp lấy thêm từ phía sau 'com' thì ta có từ 'an com', tuy vẫn có rất nhiều cách điền dấu cho từ này nhưng theo chủ quan thì bản thân chúng ta cũng đã giới hạn được việc điền dấu cho chữ 'an' và 'com'. Cứ như vậy, chúng ta thấy được rằng thông tin các chữ phía trước và sau của chữ hiện tại là rất giá trị. Do đó, BRNN sẽ được dự đoán là một mô hình khả quan cho bài toán.

5 Chuẩn bị mô hình huấn luyện:

Các mô hình được chúng tôi thử nghiệm đều được cài đặt dựa trên sự hỗ trợ của Framework Keras[8].

Để có thể tìm ra được mô hình phù hợp nhất dựa trên việc kiểm nghiệm thực tế trên bộ dữ liệu, chúng tôi chuẩn bị 4 mô hình huấn luyện khác nhau (với lớp embedding kích thước 256) gồm:

- **GRU-RNN:** Lớp GRU-RNN có 256 đơn vị, sử dụng hàm kích hoạt Softmax cho lớp đầu ra.
- **Bidirectional GRU-RNN và Bidirectional LSTM-RNN:** Hai mô hình này gồm lớp Bidirectional GRU và Bidirectional LSTM với cùng 128 đơn vị. Và lớp đầu ra sử dụng hàm kích hoạt Softmax.
- Lớp **TimeDistributed** (hoạt động giống như một lớp fully-connected) cũng sẽ được thêm vào để có thể tổng hợp kết quả của các đơn vị LSTM cho mỗi time-step. Với mô hình mà chúng tôi xây dựng thì sẽ có 2 lớp TimeDistributed được thực hiện liên tục để tối ưu kết quả dự đoán của mô hình.
- Để tránh hiện tượng quá khớp xảy ra với mô hình, chúng tôi có sử dụng thêm các lớp dropout = 0.5 cho mỗi mô hình nhằm giảm bới thông tin nhận vào của mỗi lớp.



(a) GRU. (b) Bidirectional-GRU. (c) Bidirectional-LSTM.

Hình 6: Trực quan các mô hình.

Với những kiến trúc khác nhau của mạng RNN, chúng tôi đặt mục tiêu sẽ tìm ra được mô hình phù hợp nhất cho bộ dữ liệu với tiêu trí sử dụng ít tài nguyên phần cứng và thời gian huấn luyện ngắn nhất có thể.

[8] https://www.tensorflow.org/guide/keras/rnn

6 Phân tích và đánh giá kết quả

6.1 Kết quả

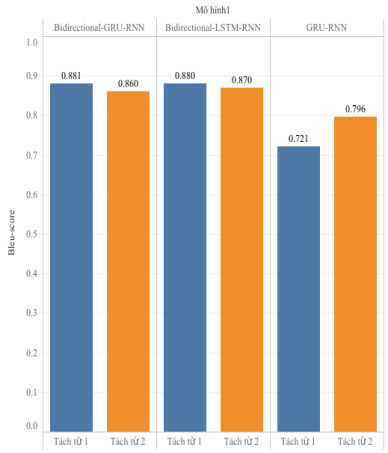
Sau khi thực nghiệm trên tất cả các mô hình nêu trên, chúng tôi có được kết quả của hai phương pháp tách từ 1 (Mục 4.1.1) và 2 (Mục 4.1.2) trong Bảng 2 và 3.

Bảng 2: Bảng kết quả đánh giá mô hình với phương pháp tách từ 1.

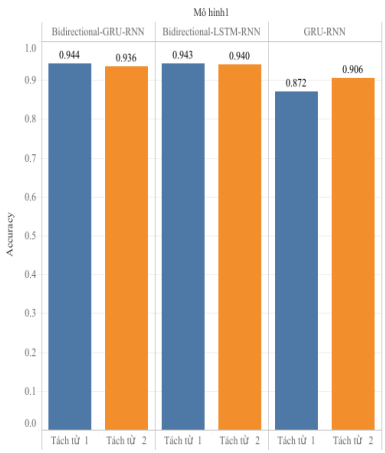
Mô hình	Train accuracy	Test Accuracy	Bleu score
GRU-RNN	0.9892	0.8790	0.7347
Bidirectional-LSTM-RNN	0.9956	0.9432	0.8798
Bidirectional-GRU-RNN	0.9954	0.9436	0.8806

Bảng 3: Bảng kết quả đánh giá mô hình với phương pháp tách từ 2.

Mô hình	Train accuracy	Test Accuracy	Bleu score
GRU-RNN	0.9925	0.9032	0.7909
Bidirectional-LSTM-RNN	0.9952	0.9399	0.8702
Bidirectional-GRU-RNN	0.9930	0.9357	0.8602



(a) Blue_score.



(b) Test Accuracy.

Hình 7: Các độ đo đánh giá trên các mô hình với hai phương pháp tách từ.

Chúng tôi nhận thấy hai phương pháp tách từ có ảnh hưởng khá lớn tới độ chính xác của mô hình trên tập kiểm thử và điểm Bleu_score.

Xét về mô hình: Trong cả 2 phương pháp tách từ, mô hình Bidirectional-LSTM cho kết quả tốt hơn 2 mô hình còn lại. Với kết quả độ chính xác trên tập kiểm thử đạt 94% thì đây là một kết quả rất tốt cho mô hình thêm dấu khi đến 94% các từ được thêm dấu đúng. Mô hình GRU là mô hình cho kết quả thấp nhất trong 3 mô hình khi chỉ đạt 87% và 90% trên 2 phương pháp tách từ.

Xét về độ đo Bleu_score: Các kết quả không quá cao nhưng có thể chấp nhận được ở mức từ 72% đến 88%. Mô hình Bidirectional-GRU cho kết quả cao nhất (88%) trên bộ tách từ 1 nhưng kết quả của mô hình Bidirectional-LSTM cũng rất sát (87%).

Khi tiến hành phân tích trên các câu khác nhau, chúng tôi nhân thấy dấu của một từ tại một vị trí bất kỳ phụ thuộc rất nhiều vào các từ đứng trước và sau từ đó nên thông tin bối cảnh trong câu có ảnh hưởng rất lớn tới kết quả điền dấu cho từ. Vì vậy, các mô hình Bidirectional RNN tốt hơn mô hình RNN thường là điều đã được dự đoán trước.

Xét về 2 phương pháp tách từ: Đối với phương pháp tách từ 2, phương pháp này giúp độ chính xác trên tập kiểm thử và Bleu_score của mô hình GRU-RNN được cải thiện đáng kể. Tiếng Việt có nhiều từ ghép nên việc tách từ đơn có thể không thực sự tốt nhưng mà qua kết quả trong Bảng 3 có thể thấy bộ dữ liệu đã đủ lớn nên đã giải quyết được phần nào các trường hợp thiếu hụt từ ghép.

Đối với phương pháp tách từ 2: Mô hình tốt nhất theo phương pháp tách từ này là Bidirectional-LSTM với độ chính xác trên tập kiểm thử và Bleu_score lần lượt là 93% và 87%. Đây là kết quả rất tốt, cho thấy phương pháp tách từ này cũng hiệu quả cho bài toán thêm dấu. Độ chính xác trên tập kiểm thử của cả 3 mô hình cũng ít có sự chênh lệch hơn khi sử dụng phương pháp 2.

6.2 Phân tích lỗi

Trong quá trình đánh giá mô hình, chúng tôi cũng tiến hành đánh giá các lỗi thêm dấu của mô hình trên dữ liệu kiểm thử và cũng tìm ra được những lỗi sai phổ biến của mô hình chúng tôi xây dựng.

Có 2 nhóm lỗi sai chính là sai do chữ đầu vào thiếu chữ đi kèm và sai do mô hình thiên vị việc gán nhãn:

- Sai do đầu vào thiếu chữ đi kèm: Ví dụ như chữ 'thay' ('thầy') khi đưa và mô hình nếu chữ 'thay' đứng đầu câu hay chỉ đứng một mình thì sẽ bị thêm dấu sai thành 'thấy'. Tuy nhiên, nếu các chữ 'nguoi' hay 'giao' (trong 'thay giao' và 'nguoi thay') được đưa vào chung với chữ 'thay' thì việc thêm dấu của mô hình sẽ luôn chính xác. Điều này cho thấy mô hình đã học được những cụm từ ghép và những chữ hay xuất hiện cùng nhau trong câu.
- Sai do mô hình thiên vị: Có những trường hợp mô hình sẽ thêm dấu mặc định cho một số chữ mà không cần biết các chữ đó có ý nghĩa gì trong câu, ví dụ như từ 'ban than' luôn được gán nhãn là 'bản thân' mặc dù dấu mong muốn là 'bạn thân'.

Lỗi sai do mô hình thiên chỉ xuất hiện trong dự đoán của mô hình GRU-RNN, vì những đặc trưng riêng về ưu điểm và hạn chế của các mô hình. Từ đó,có thể thấy rằng các mô hình Bidirectional RNN đã cho kết quả tốt như kỳ vọng.

6.3 Thời gian huấn luyện và kiểm tra

Trong cả 6 mô hình mà chúng tôi đã tiến hành thực nghiệm, việc tinh chỉnh mô hình là rất quan trọng để có thể đạt được kết quả tốt nhất. Nhưng nhờ vào thời gian huấn luyện chỉ mất 1.5 giờ để huấn luyện mô hình nên chúng tôi đã có thể có được mô hình tốt nhất với thời gian huấn luyện ngắn nhất.

Trong kiểm tra mô hình, tốc độ dịch từ câu không dấu sang câu có dấu của các mô hình của chúng tôi nhanh hơn đáng kể so với MOSES toolkit ở các công trình nguyên cứu trước. Dưới đây là bảng so sánh tốc độ kiểm tra.

Bảng 4: Hiệu năng của các mô hình khi chạy thực nghiệm trên điểm dữ liệu.

Mô hình / Toolkit	**Thời gian thêm dấu/câu (giây)**	**Số câu/giây**
MOSES[4]	0.125	8
OpenNMT[5]	0.045	22
Bidirectional-GRU-RNN	0.043	23

7 Kết luận và hướng phát triển

Trong bài báo này, chúng tôi giới thiệu và so sánh 3 mô hình dịch máy dựa trên các thuật toán mạng nơ-ron nhân tạo để giải quyết bài toán thêm dấu cho các câu Tiếng Việt không dấu. Với các phương pháp mà chúng tôi tiến hành thực nghiệm, mô hình Bidirectional-GRU cho kết quả tốt nhất với Bleu_score đạt 88.06% và mô hình Gated Recurrent Unit cho kết quả thấp nhất với Blue_score đạt 73.47%. Đây là một kết quả khá tốt đối với một khoảng thời gian huấn luyện và tài nguyên tính toán hạn chế. Cùng với đó, chúng tôi cũng so sánh các phương pháp tách từ trên tiếng Việt về bài toán thêm dấu và nhận thấy có sự chênh lệch nhỏ về hiệu suất mô hình giữa hai phương pháp tách từ. Ở phương pháp tách từ 1 có Bleu_score cao nhất là 88.06% và phương pháp tách từ 2 có Bleu_score cao nhất là 87.02%.

Các mô hình Bidirectional cho kết quả cao hơn vượt trội so với các mô hình GRU bình thường với Bleu_score chênh lệch cao hơn 15% ở phương pháp tách từ 1 và 8% ở phương pháp tách từ 2. Như vậy, việc thêm dấu cho một từ Tiếng Việt cần phải yêu cầu thông tin từ cả trước và sau từ đó để có thể dự đoán dấu đạt chính xác nhất.

Trong tương lai, chúng tôi sẽ phát triển thêm bài toán để có thể đạt độ chính xác cao nhất với chi phí thấp nhất để có thể ứng dụng vào cuộc sống. Việc thêm dữ liệu với đa dạng hóa bối cảnh dữ liệu là một cách để nâng cao hiệu suất mô hình. Bên cạnh đó, áp dụng các thuật toán học sâu mới như Transformer có thể giúp mô hình dịch máy chính xác hơn.

Tài liệu

1. Tin Van Huynh, Vu Duc Nguyen, Kiet Van Nguyen, Ngan Luu-Thuy Nguyen, and Anh Gia-Tuan Nguyen. Hate speech detection on vietnamese social media text using the bi-gru-lstm-cnn model. *arXiv preprint arXiv:1911.03644*, 2019.
2. Luan Thanh Nguyen, Kiet Van Nguyen, and Ngan Luu-Thuy Nguyen. Constructive and toxic speech detection for open-domain social media comments in vietnamese. *arXiv preprint arXiv:2103.10069*, 2021.
3. Yuchen Fan, Yao Qian, Feng-Long Xie, and Frank K Soong. Tts synthesis with bidirectional lstm based recurrent neural networks. In *Fifteenth annual conference of the international speech communication association*, 2014.
4. Luan-Nghia Pham, Viet Hong Tran, and Vinh Van Nguyen. Vietnamese text accent restoration with statistical machine translation. In *Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC 27)*, pages 423–429, 2013.
5. T. Pham, X. Pham, and P. Le-Hong. On the use of machine translation-based approaches for vietnamese diacritic restoration. In *2017 International Conference on Asian Language Processing (IALP)*, pages 272–275, 2017.
6. Vu Cong Duy Hoang, Dien Dinh, Nguyen Le Nguyen, and Hung Quoc Ngo. A comparative study on vietnamese text classification methods. In *2007 IEEE International Conference on Research, Innovation and Vision for the Future*, pages 267–273. IEEE, 2007.
7. David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
8. Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.
9. Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.
10. Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.
11. Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
12. Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.
13. Tomáš Mikolov et al. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 80:26, 2012.
14. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
15. Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.
16. Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
17. Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.