Kỷ yếu Hội nghị Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Đà Nẵng, ngày 17-18/08/2017
DOI: 10.15625/vap.2017.00068

MỘT PHƯƠNG PHÁP PHÂN ĐOẠN KÝ TỰ CHO BÀI TOÁN NHẬN DẠNG CHỮ VIẾT TAY ONLINE

Nguyễn Thị Thanh Nga[1], Bùi Thu Lâm[2], Nguyễn Đức Dũng[1]

[1]Viện Công nghệ thông tin, Viện Hàn lâm Khoa học và Công nghệ Việt Nam

[2] Khoa Công nghệ thông tin, Học viện Kỹ thuật Quân sự

thanhnga.mta@gmail.com, lambt@lqdtu.edu.vn, nddung@ioit.ac.vn

TÓM TẮT: *Bài báo đề xuất một phương pháp phân đoạn ký tự trong nhận chữ viết tay trực tuyến (online) tiếng Anh. Phương pháp đề xuất việc sử dụng thuật toán lấy mẫu Ramer để tạo over-segmentation kết hợp với kết quả nhận dạng và thống kê ngữ cảnh nhằm cải thiện độ chính xác của vị trí phân đoạn ký tự. Một engine nhận dạng sử dụng mạng học sâu để đưa ra điểm phân đoạn tốt nhất. Phần thực nghiệm được tiến hành trên bộ dữ liệu IAM-OnDB, là bộ chữ viết tay tiếng anh online được thu nhận bằng thiết bị "smart whiteboard". Các kết quả nhận được cho thấy phương pháp đề xuất có khả năng phân đoạn chính xác ranh giới các ký tự trong bài toán nhận dạng chữ viết tay online tiếng Anh.*

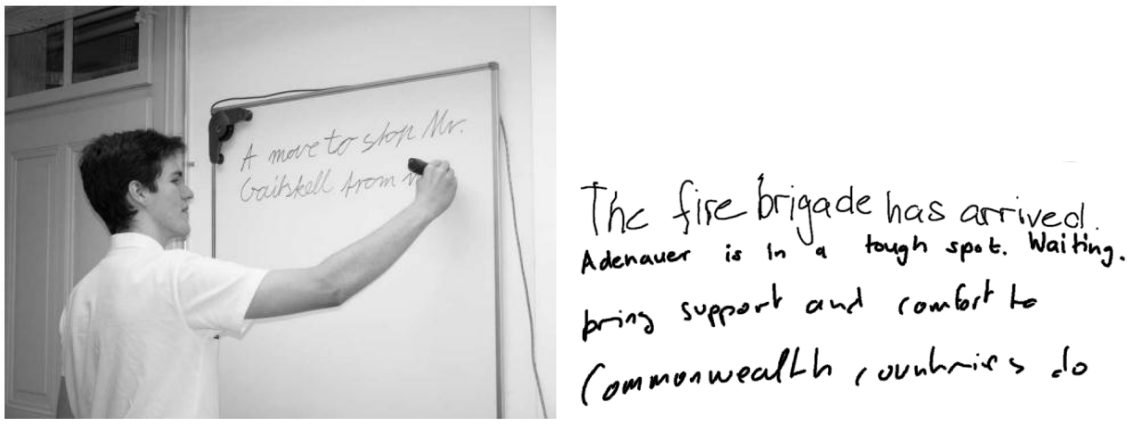
Từ khóa: *Phân đoạn ký tự, Recurrent neural network (RNN), Long short-term memory (LSTM), Chữ viết tay online, IAM-onDB***.**

I. GIỚI THIỆU

Mặc dù vấn đề nhận dạng chữ viết tay đã được nghiên cứu trong hơn 30 năm [2, 3, 4], nhưng hiện nay nó vẫn còn rất nhiều vấn đề mở. Nhận dạng chữ viết tay được chia thành 2 loại: nhận dạng trực tuyến (*on-line*) và nhận dạng *off-line*. Trong đó, chữ viết tay *on-line* là tập các tọa độ điểm theo thời gian, đại diện cho sự chuyển động của đầu bút khi viết trên thiết bị thu nhận, trong khi đó, chữ viết tay *off-line* chỉ có hình ảnh của văn bản có sẵn. Trong bài báo xem xét việc phân đoạn kí tự đối với văn bản viết tay *on-line* trên một tấm bảng, gọi là "smart boad" [7]. Đây là một nhiệm vụ khó, vì đối với chữ viết tay *on-line* thu nhận được từ một tấm bảng, người viết đứng chứ không ngồi tựa tay trên bảng như viết trên thiết bị máy tính bảng, do đó các dòng văn bản trên bảng thường không có cùng kích thước, độ xiên, nghiêng. Đồng thời kích thước của các chữ cái là không giống nhau ở đầu và ở cuối của 1 dòng (Hình 1.b). Do đó, cần phải phân đoạn các dòng văn bản thành các thành phần nhỏ hơn để nhận dạng.

Phân đoạn ký tự trong nhận dạng chữ viết tay *on-line* là vấn đề khó bởi điểm phân tách giữa các ký tự thường không rõ ràng do sự dày thưa của các tọa độ điểm dữ liệu. Nếu không có các dấu hiệu nhận dạng ký tự thì không thể phân tách rõ ràng các ký tự được. Do đó, một cách để thực hiện là kết hợp cả phân đoạn và nhận dạng. Trong bài báo đề xuất một phương pháp phân đoạn ký tự dựa trên phân đoạn *over-segmentation*, sử dụng thuật toán lấy mẫu Ramer [1]. Khi thực nghiệm trên bộ dữ liệu IAM-ONDB cho thấy thuật toán phân đoạn đề xuất có khả năng phân đoạn tốt đối với những trường hợp các nét bị viết liền nhau, không được phân tách thành các kí tự riêng biệt.

Nhóm tác giả xây dựng một *engine* nhận dạng ký tự sử dụng mạng học sâu Long short-term memory (LSTM) huấn luyện trên bộ dữ liệu UJPEN-OnDB[14] và bộ kí tự trích xuất từ bộ dữ liệu IAM-OnDB [7] cho kết quả nhận dạng kí tự lên đến 99,67% khi lấy *top-4* kí tự gần nhất để đưa ra các điểm phân tách tốt nhất.



Hình 1. (a) Quá trình thu nhận dữ liệu (b) Dữ liệu viết tay online

Cấu trúc của bài báo với các phần còn lại như sau: Phần II. Trình bày mô hình nhận dạng chữ viết tay *on-line*. Phần III. Mô hình mạng Recurrent neural network (RNN)/ Long Short Term Memory (LSTM). Phần IV. Trình bày phương pháp phân đoạn đề xuất. Phần V. Trình bày về phần thực nghiệm và các kết quả đạt được, đánh giá. Phần VI. Kết luận và hướng phát triển tương lai.

II. MÔ HÌNH NHẬN DẠNG CHỮ VIẾT TAY ONLINE

A. Tổng quan về nhận dạng chữ viết tay

Nhận dạng chữ viết tay *on-line* được thực hiện trên cơ sở lưu lại các thông tin về nét chữ cũng như thứ tự nét viết hướng và tốc độ của nét viết trong quá trình đang viết. Đây chính là cơ sở để máy tính nhận diện được các chữ cái. Ngược lại, đối với nhận dạng chữ viết tay *off-line*, dữ liệu đầu vào là ảnh văn bản đã được quét. Nhận dạng chữ viết tay *on-line* là nhiệm vụ mới, đặc biệt là kiểu dữ liệu *on-line* thu được từ thiết bị "smart board". Khó khăn lớn nhất khi nghiên cứu bài toán nhận dạng chữ viết tay *on-line* là sự biến thiên quá đa dạng trong cách viết của từng người trên một thiết bị thu nhận. Cùng một người viết nhưng đôi khi cũng có nhiều sự khác biệt trong cách viết tùy thuộc vào từng ngữ cảnh, kiểu viết của một người cũng có thể thay đổi theo thời gian hoặc theo thói quen… Điều này gây ra nhiều trở ngại trong việc trích chọn đặc trưng cũng như lựa chọn mô hình nhận dạng.

Hiện nay, có rất nhiều công trình công bố cho bài toán nhận dạng chữ viết tay *on-line*, như phương pháp nhận dạng chữ viết tay *on-line* sử dụng Hidden Markov Model (HMM) của Marcus Liwicki và HorstBunke [11] cho kết quả nhận dạng đạt 65,4% khi không sử dụng mô hình ngôn ngữ và 70,8% có sử dụng mô hình ngôn ngữ trên bộ dữ liệu IAM-OnDB, hay phương pháp Connectionist Temporal Classification (CTC) kết hợp mô hình Bidirectional Long Short Term Memory (BLSTM) [10,12] cho kết quả nhận dạng đạt 74,0% trên bộ dữ liệu IAM-OnDB. Sự khác biệt đáng kể giữa bài toán nhận dạng ký tự riêng biệt và nhận dạng cả từ hoàn chỉnh xuất phát từ sự thể hiện đa dạng của các kí tự khi xuất hiện cùng nhau và phương pháp phân đoạn kí tự sao cho chính xác trong một từ đã khiến cho việc huấn luyện các thuật toán học máy gặp rất nhiều khó khăn.

Một số mô hình nhận dạng chữ viết tay *on-line* thường chuyển kiểu dữ liệu dạng tập hợp các điểm sang dạng hình ảnh để tiến hành phân đoạn và áp dụng các phương pháp nhận dạng *off-line* hiện có. Như trong mô hình nhận dạng chữ viết tay *on-line* sử dụng mạng Multi Convolution Neural Networks (MCNN)[13], tác giả chuyển kiểu dữ liệu từ *on-line* sang ma trận các *fixel* ảnh để thực hiện, tiến hành thử nghiệm trên bộ dữ liệu MNIST cho kết quả nhận dạng 89% cho nhận dạng ký tự.

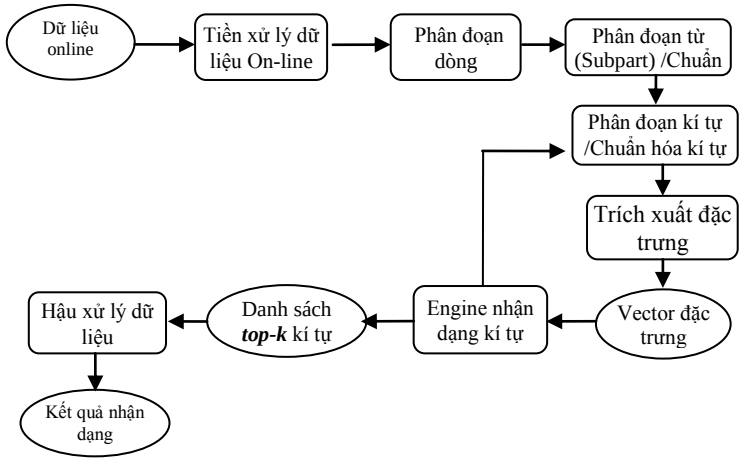
Bài báo của tác giả đề cập đến một khía cạnh khác trong nhận dạng chữ viết tay *on-line*, nhận dạng trực tiếp trên tập dữ liệu thô mà không cần phải chuyển kiểu dữ liệu từ tập các tọa độ điểm sang ảnh, cố gắng cải thiện phương pháp phân đoạn kí tự để tăng chất lượng nhận dạng.

B. Mô hình nhận dạng chữ viết tay on-line

Để nhận dạng văn bản viết tay nói chung, cần xác định các dòng, sau đó xác định các từ rồi nhận dạng từng từ, để nhận dạng từ, ta cần phải phân tách các kí tự trong từ bằng cách xác định các vị trí cắt. Tiếp theo ta sẽ tiến hành nhận dạng các phần được tách ra đồng thời điểm tra xem kí tự vừa nhận dạng được có hợp lý hay không bằng cách sử dụng thống kê ngữ cảnh, nếu thỏa mãn sẽ ghi nhận vị trí cắt hiện tại và lặp lại các thao tác này với các phần còn lại.

Như vậy, quá trình nhận dạng sẽ được thực hiện đối với mỗi phân đoạn, mỗi cách phân đoạn kí tự tương ứng với một đường đi, mỗi điểm trên đường đi tương ứng với một kí tự được nhận dạng đi kèm với độ chính xác của kí tự đó. Sau khi quá trình nhận dạng kết thúc, ta ghép các kí tự trên mỗi đường đi sẽ thu được một danh sách các từ ứng cử viên (từ có khả năng được lựa chọn) tương ứng với một cách phân đoạn đối với từ đầu vào. Với phương pháp này nếu xác định được từ có ***n*** vị trí cắt thì số từ ứng viên tối đa sinh ra sẽ là ***n!*** từ. Khi đó, ứng viên có xác suất nhận dạng và thống kê ngữ cảnh cao nhất sẽ được chọn. Dữ liệu thống kê ngữ cảnh được lưu vào hai từ điển: từ điển *bi-gram* mức kí tự và từ điển *uni-gram* mức từ.

Mô hình nhận dạng chữ viết tay được mô tả trong Hình 2.



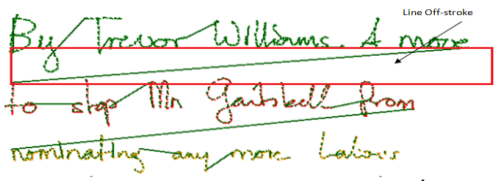
Hình 2. Mô hình nhận dạng chữ viết tay Online English (tiếng Anh)

Hệ thống bao gồm 6 module chính:

- Tiền xử lý dữ liệu *on-line (online preprocessing):* thực hiện lọc nhiễu của dữ liệu thô;
- Tách dòng *(line segmentaion):* Thực hiện tách các dòng văn bản khỏi đoạn văn bản;
- Tách từ (hay còn gọi là các *SubPart*) và chuẩn hóa các từ *(SubPart segmentaion and nomalization);*
- Tách ký tự và chuẩn hóa ký tự *(character segmentation and nomalization)*;
- Trích chọn đặc trưng *(feature extraction):* chuỗi các điểm dữ liệu được chuyển thành chuỗi các vector đặc trưng;
- Nhận dạng *(recognition):* ở đây sử dụng LSTM để nhận dạng ký tự, đưa ra được danh sách các kí tự gần giống nhất với dữ liệu đưa vào;
- Hậu xử lý *(post-processing).*

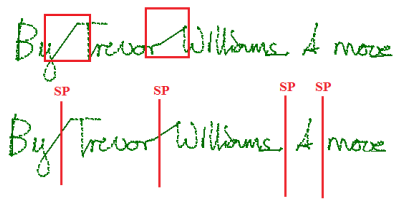
Trước khi trích rút đặc trưng, dữ liệu thô ban đầu cần được tiền xử lý và chuẩn hóa. Điều này rất quan trọng trong hệ thống nhận dạng chữ viết tay, bởi vì mỗi người viết sẽ có độ nghiên, xiên, chiều cao, rộng của nét chữ là khác nhau. Các kỹ thuật tiền xử lý dữ liệu chữ viết tay *on-line* được trình bày rõ ràng và chi tiết trong các tài liệu [3, 5, 7].

Sau khi tiền xử lý dữ liệu, ta phân tách dữ liệu thành các dòng, dựa vào khoảng cách của tọa độ điểm x của điểm cuối cùng của dòng trên và điểm đầu tiên của dòng dưới (hay còn gọi là *LOS feature- Line off-stroke*). Từ đó, ta đưa ra được danh sách các dòng.



Hình 3. Tách các dòng trong văn bản viết tay

Sau khi phân tách được các dòng, do đặc điểm của dữ liệu trên một dòng có độ to nhỏ khác nhau, thường thì sẽ to ở phần đầu, và nhỏ hơn ở cuối dòng (Hình 1.b), do đó, cần phải phân đoạn dòng thành các thành phần nhỏ hơn (gọi là các *SubPart* hay các từ), các thành phần sẽ được tách ở những khoảng trống có kích thước lớn hơn khoảng trống trung bình trên toàn dòng, đồng thời kích thước của cả 2 *SubPart* phải lớn hơn 1 ngưỡng xác định trước.



Hình 4. Tách các subpart (hoặc từ) của dòng

Sau khi tách được các thành phần con (*SubPart* hoặc *từ*), tiến hành phân đoạn ký tự, các phần tách được có thể là 1 ký tự hoặc 1 phần của ký tự, kết hợp các thành phần này lại với nhau ta sẽ có các phương án phân đoạn cho mỗi *SubPart*, với mỗi phương án phân tách, sử dụng *engine* nhận dạng từng phần để đưa ra điểm phân tách tốt nhất, nhận dạng từng phần phân tách và kết hợp với các thông tin ngữ cảnh để đưa ra kết quả nhận dạng chữ viết tay *on-line* tốt nhất.

Tuy nhiên, việc phân tách ký tự trong chữ viết tay *on-line* là vô cùng khó khăn do số lượng điểm dày đặc, cùng với tốc độ viết dày mỏng của các nét và thiết bị thu nhận dữ liệu, làm cho việc phân tách bằng các phương pháp thông thường hết sức khó khăn.

Ở phần V sẽ trình bày rõ phương pháp đề xuất phân đoạn ký tự trong các thành phần con (*SubPart*) dựa trên phương pháp ***over-segmentation*** và phương pháp lấy mẫu Ramer.

III. MẠNG RECURRENT NEURAL NETWORK VÀ LONG SHORT TERM MEMORY (RNN/ LSTM)

Đặc điểm của chữ viết tay *on-line* là chuỗi các điểm có tọa độ *(x,y)* đại diện cho vị trí của đầu bút khi viết trên thiết bị thu nhận, đây là một kiểu dữ liệu liên tục theo thời gian. Để nhận dạng được kiểu dữ liệu này, một số mô hình trước đây sử dụng Hidden Markov Model (HMM), HMM là một phương pháp hiệu quả trong việc mô hình hóa quan hệ của dữ liệu đầu vào và thời gian. Tuy nhiên do tính chất Markov, đó là HMM chỉ mô hình được sự tương tác cục bộ của dữ liệu (đặc trưng của đầu vào thời điểm *t* chỉ phụ thuộc vào đặc trưng của đầu vào tại thời điểm *t-1*). Do vậy mạng Recurrent Neural Network (RNN) và mạng cải tiến Long- Short Term Memory sẽ là một cách tiếp cận tốt hơn.

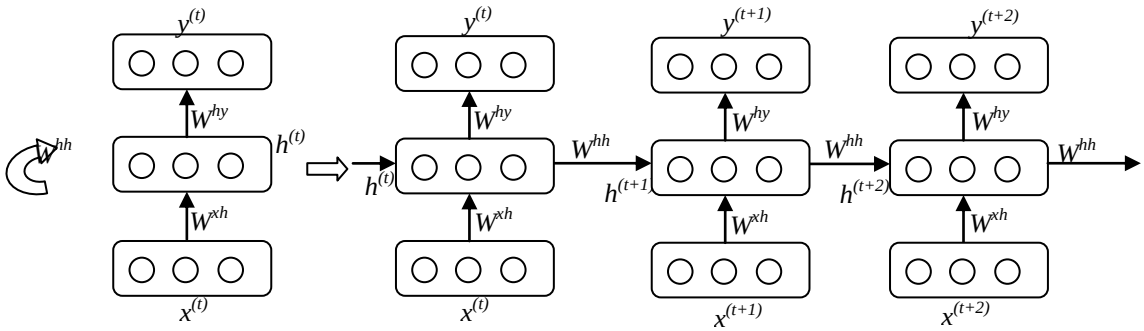
A. Mạng Recurrent Neural Network (RNN)

Mô tả cách thức hoạt động của mô hình RNN như sau: cho trước một chuỗi đầu vào $x = (x^{(1)}, x^{(2)}, ..., x^{(T)})$. Mô hình RNN sẽ tính chuỗi vector tầng ẩn và đầu ra tương ứng $h = (h^{(1)}, h^{(2)}, ..., h^{(T)})$ và $y = (y^{(1)}, y^{(2)}, ..., y^{(T)})$ bằng cách lặp các phương trình sau từ $t = 1$ đến T:

$$h^{(t)} = f(W^{xh}x^{(t)} + W^{hh}h^{(t-1)} + b^h)$$
$$y^{(t)} = softmax(W^{hy}h^{(t)} + b^y) \quad (1)$$

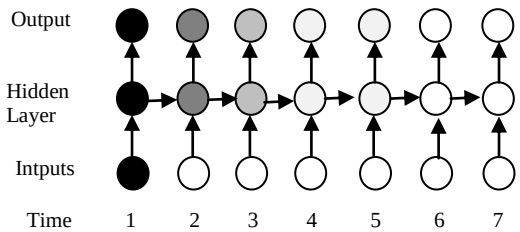
Trong đó: $x^{(t)}$ là input tại thời điểm thứ t. T là độ dài của chuỗi đầu vào. Hàm f là hàm nonlinearity (*Sigmoid, tanh* hoặc *ReLU*). $h^{(t-1)}$ là hidden ở thời điểm *t-1*, W là ma trận trọng số, đầu ra $y^{(t)}$ là vector chứa xác suất của đối tượng cần phân loại. softmax tương ứng với hàm *softmax function* như hàm độ lỗi (*loss function*). Đối với hàm *softmax function* ta dùng hàm *cross-entropy loss* (hay còn gọi là *negative log likelihood*).

Một cách hiệu quả để hình dung mạng RNN đó là trải mạng theo chuỗi đầu vào như được mô tả trong hình 5.



Hình 5. Mạng RNN được trải ra theo thời gian

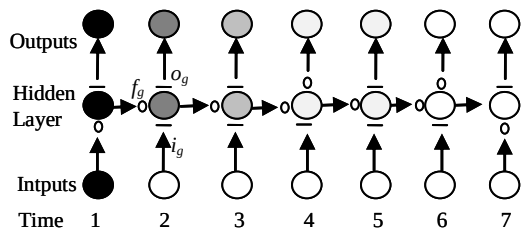
Theo như hình, mỗi nút biểu diễn một tầng của mạng tại một thời điểm. Tuy nhiên trên thực tế, phạm vi ngữ cảnh mà mô hình RNN chuẩn có thể truy cập được là rất hạn chế. Một vấn đề đó là ảnh hưởng của dữ liệu đầu vào lên tầng ẩn và sau đó là tầng đầu ra có thể bị giảm hoặc tăng đáng kể khi ta lặp các kết nối hồi quy của RNN. Vấn đề này được gọi là 'vanishing/exploding gradient' như mô tả trong hình 6. Trong phần tiếp theo, báo cáo sẽ trình bày mô hình Long-Short Term Memory (LSTM), là một đề xuất trong để giải quyết vấn đề trên.



Hình 6. Vấn đề ***'vanishing gradient'*** đối với RNN

B. Mô hình Long-Short Term Memory (LSTM)

Kiến trúc Long-Short Term Memory (*LSTM*) bao gồm một tập các khối nhớ (*memory block*). Mỗi khối này chứa một hoặc nhiều đơn vị nhớ (*memory cells*) và 3 cổng là: cổng ghi vào (i_g - *input gate*), cổng xóa (f_g - *forget gate*) và cổng ghi ra (o_g- *output gate*).



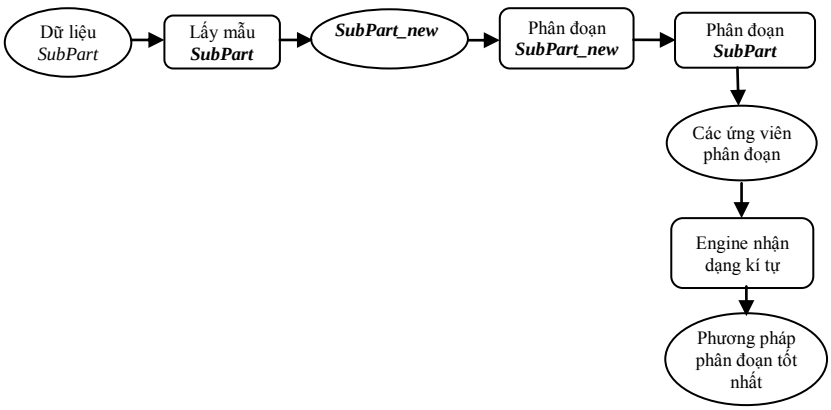
Hình 7. LSTM giúp giảm nhẹ tác động của vấn đề ***'vanishing gradient'***

Một mạng LSTM có thể được xem như một mạng RNN, tuy nhiên thay vì sử dụng nút với phép tổng hợp tuyến tính và hàm kích hoạt, mô hình LSTM sẽ sử dụng các khối nhớ với mục đích điều khiển các luồng thông tin ra vào mỗi khối nhớ. Cơ chế hoạt động của các khối nhớ này giúp cho các đơn vị nhớ LSTM có thể lưu giữ được các thông tin ngữ cảnh trong một khoảng thời gian dài, và giảm bớt được vấn đề '*vanishing gradient*'. Ví dụ như trong hình 7, mô tả sự ảnh hưởng của nút vào lên các nút ở tầng ẩn và tầng đầu ra (màu đen thể hiện sự ảnh hưởng mạnh và màu trắng là sự ảnh hưởng yếu). Để đơn giản, có thể hiểu các cổng chỉ ở 2 trạng thái, đóng hoặc mở. Đơn vị nhớ sẽ nhớ *input* đầu tiên khi cổng *forget* mở và cổng *input* đóng và các nút ở đầu ra sẽ bị ảnh hưởng nếu như cổng *output* mở.

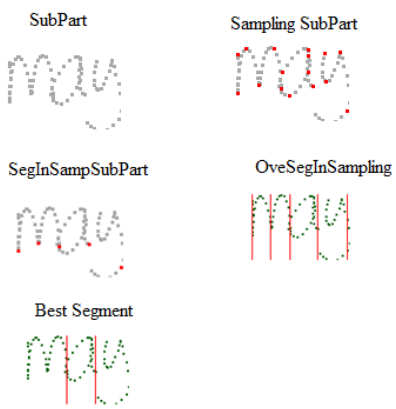
IV. ĐỀ XUẤT PHƯƠNG PHÁP PHÂN ĐOẠN

Đặc điểm của kiểu dữ liệu *on-line* được thể hiện dưới dạng tập hợp các điểm tọa độ của mỗi nét chữ. Mỗi nét (*stroke*) có thể là một kí tự, một phần của kí tự hoặc cũng có thể là các kí tự liền kề nhau. Do đó, để phân đoạn kí tự, ta xét các trường hợp một nét là một kí tự, một phần của kí tự và các kí tự liền nhau. Đối với trường hợp một nét là một kí tự, tiến hành nhận dạng và sử dụng xác suất nhận dạng để đưa ra kí tự đúng nhất, đối với trường hợp là một phần của kí tự, tiến hành ghép các nét liền kề và tiến hành nhận dạng để đưa ra trường hợp ghép hợp lý nhất, đối với nét là các kí tự liền kề (hay còn gọi là một *SubPart*), ta phải phân tách thành các ký tự, việc phân tách sử dụng phương pháp phân đoạn được đề xuất dưới đây.

Đối với việc phân đoạn *SubPart,* một số các phương pháp cơ bản thường chuyển dữ liệu *on-line* sang dữ liệu *off-line* (dữ liệu ảnh) để thực hiện phân đoạn, sau đó, sử dụng phương pháp *Histogram* theo chiều ngang để phân đoạn dòng và theo chiều dọc để phân đoạn kí tự. Trong trường hợp này, đường phân cách là khoảng điểm có số lượng điểm ảnh cắt hai dòng hoặc hai kí tự là ít nhất, khi đó, trên lược đồ *histogram*, các đường phân cách là các đường thấp nhất trên lược đồ *histogram*. Tuy nhiên, điểm hạn chế khi dùng phương pháp này là dữ liệu *off-line* thường có các nét đậm, mảnh khác nhau tùy vào việc nhấn đầu bút xuống giấy, nhưng dữ liệu *on-line* thì không, nếu chuyển đổi 2 kiểu dữ liệu này với nhau sẽ khiến thông tin dữ liệu bị mất mát, phân bố điểm ảnh trên các nét không đều nhau, dẫn đến việc phân đoạn sẽ không chính xác. Do đó, phương pháp phân đoạn mà tác giả đề xuất được thực hiện ngay trên tập các tọa độ điểm *(x, y),* mỗi một *SubPart* có độ dài *n* điểm, *n* thường lớn và phân bố điểm thường không đều nhau. Trong bài báo, tác giả đề xuất tên gọi là thuật toán ROVS (*Ramer Over Segmentation*).



Hình 8. Phương pháp phân đoạn đề xuất



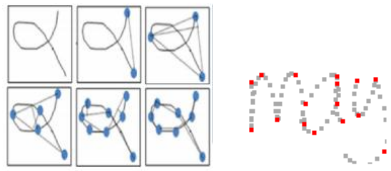
Hình 9. Ví dụ về phương pháp phân đoạn đề xuất

Tư tưởng chính của phương pháp là sử dụng thật toán làm giảm số lượng dữ liệu mà không làm mất mát thông tin của dữ liệu gốc ban đầu và đảm bảo vẫn đại diện cho các đường nét của *SubPart* ban đầu. Thuật toán để giảm số lượng dữ liệu hay còn gọi là lấy mẫu được sử dụng của Ramer [1], sẽ được trình bày cụ thể ở dưới đây. Sau khi giảm kích thước dữ liệu, tiến hành phân tách trên dữ liệu mới. Có một đặc điểm của dữ liệu *on-line* là thứ tự điểm thu nhận được sắp xếp theo thứ tự nét bút của người viết, do đó, để tiến hành phân đoạn ký tự trên *SubPart*, ta sắp xếp lại dữ liệu theo chiều tăng của tọa độ *x* của mỗi điểm. Việc sắp xếp này nhằm mục đích phân tách tốt các khoảng trắng đại diện tiêu biểu cho các ranh rới giữa các kí tự hoặc một phần ký tự mà không làm thay đổi hình dạng của nét chữ. Việc tách các phần trên tập dữ liệu mới dựa trên khoảng cách giữa các điểm liên tiếp lớn hơn một ngưỡng cho trước. Sau khi tách được các phần con trên dữ liệu mới, đánh dấu các điểm phân tách và quay lại phân đoạn trên tập dữ liệu gốc ban đầu.

Ở đây có các tham số như việc giảm kích thước dữ liệu xuống còn bao nhiêu là đủ, hoặc ngưỡng phân tách nhận giá trị bao nhiêu là phù hợp, sẽ được trình bày ở các phần tiếp theo.

A. Lấy các điểm dữ liệu đặc trưng

Mục đích là giảm kích thước số điểm dữ liệu trong các nét của *SubPart*, chỉ giữ lại những điểm dữ liệu đặc trưng của mỗi nét. Việc trích chọn các điểm đặc trưng sử dụng phương pháp của Rammer [1].



Hình 10. (a) Phương pháp Ramer (b) Kết quả sau khi lấy mẫu

Thuật toán Ramer:

Thuật toán sử dụng 2 ngăn xếp OPEN và CLOSED. Nếu các đỉnh được kiểm tra sẽ được đẩy vào stack OPEN, nếu sau khi kiểm tra thỏa mãn điều kiện thì đỉnh sẽ được đẩy từ ngăn xếp OPEN sang ngăn xếp CLOSED. Thuật toán sẽ được lặp cho đến khi ngăn xếp OPEN rỗng. Cuối thuật toán, ta sẽ thu được những điểm cần lấy nằm trong ngăn xếp CLOSED.

Bắt đầu bằng một đa giác (Đa giác có thể kín hoặc không kín). Tạo một đường thẳng giữa 2 điểm đầu cuối (Đây là phép xấp xỉ ban đầu cho đa giác đơn giản). Tính toán khoảng cách giữa cạnh và các đỉnh còn lại. Nếu đỉnh có khoảng cách lớn nhất và lớn hơn một ngưỡng ε cho trước thì thêm điểm vừa xét vào danh sách các điểm cần lấy mẫu. Lặp lại bước 2 cho đến khi không còn điểm nào có thể được thêm vào danh sách các điểm cần lấy mẫu.

Thuật toán: Ramer cho chuỗi dữ liệu không khép kín ***(Open Curves)***

1. **PUSH** (closed, first point)
2. **PUSH**(open, last point)
3. **while** (NOT(EMPTY(open)))
4. **do**
5. A ←STACKTOP(closed);
6. B ← POP(open);
7. **Find** $C \mid d(P, A^{-}B) \le d(C, A^{-}B),\ \forall P \in A^{\wedge}B, P \ne C$;
8. **if** $d(C, A^{-}B) < \varepsilon$
9. **then**
10. PUSH(closed,B);
11. **else**
12. PUSH(open, B);
13. PUSH(open,C);

Thuật toán: Ramer cho chuỗi dữ liệu khép kín ***(Closed Curves)***

1. **PUSH** (closed, first point);
2. **PUSH** (open, first point);
3. **PUSH** (open, last point);
4. **while** (NOT(EMPTY(open)))
5. **do**
6. A ←STACKTOP(closed);
7. B ← POP(open);
8. **Find** $C \mid d(P, A^{-}B) \le d(C, A^{-}B),\ \forall P \in A^{\wedge}B, P \ne C$;
9. **if** $d(C, A^{-}B) < \varepsilon$
10. **then**
11. **PUSH**(closed,B);
12. **else**
13. **PUSH**(open, B);
14. **PUSH**(open,C);

Điểm đầu - điểm cuối là điểm đầu của ***SubPart***; điểm cuối là điểm kết thúc của ***SubPart***.

$$\varepsilon = \frac{Feature\_Level}{\frac{w\_sp * h\_sp}{count\_point\_sp}} \quad (2)$$

Trong đó:

Mặc định ***Feature_Level=300***; ***w_sp*** là chiều rộng của ***SubPart***; ***h_sp*** là chiều cao của ***SubPart***; ***count_point_sp*** là số điểm dữ liệu trong ***SubPart;*** ε là ngưỡng cho phép của khoảng cách giữa các đỉnh và đường thẳng nối giữa 2 điểm đầu cuối.

B. Thực hiện Over-Segmentation trên dữ liệu đã được lấy mẫu

Sau khi đã lấy được những điểm đặc trưng, ta phân tách thành các thành phần con (hay còn gọi là *SubChar*). Mỗi *SubChar* được phân tách nếu khoảng cách giữa chúng nhỏ hơn một ngưỡng xác định trước. Thông thường việc xác định ngưỡng sẽ dựa vào khoảng cách trung bình trên toàn bộ *SubPart***.**

Mỗi *SubChar* được tách ra có khả năng là một ký tự hoặc một phần ký tự.

Hình 11. (a) Một trường hợp của phân đoạn ký tự (b) Minh họa kết quả phân đoạn tốt nhất

Thuật toán: ROVS cho phân đoạn kí tự

INPUT: Tập hợp điểm đại diện cho từ cần phân đoạn (Supart) $P=\{p_1,p_2,...p_t\}$.
OUTPUT: Danh sách các ứng viên có khả năng lựa chọn (tương ứng với các cách phân đoạn khác nhau)

---

1. Sử dụng thuật toán Ramer lấy mẫu trên tập dữ liệu *P*, thu được *P' (Sampling SubPart)*
2. Thực hiện phân đoạn trên tập P'

- Trên tập *P'*. Tìm các điểm p_i thỏa mãn: $p'_i.X<Height/2$; Trong đó: *Height* là chiều cao của từ (được tính bằng khoảng cách giữ điểm có tọa độ *Y* cao nhất trừ đi *Y* có tọa độ thấp nhất). Khi đó, ta có tập *P''*

- Trên tập *P''*.Tìm các điểm ứng cử viên phân đoạn p''_i thỏa mãn: $|p''_i-p''_{i+1}| \geq \beta$. Trong đó: β là ngưỡng cho trước, trong bài báo tác giả đang để $\beta = 10$.

- Khi đó, ta thu được tập *P** là tập các điểm ứng viên cho việc phân đoạn.

3. Tạo các ứng viên phân đoạn

Ta dựa từ tập $P^* = \{SP_1,SP_2,...,SP_n\}$ .Với *n* là số điểm phân đoạn có được từ bước 2

- Từ tập *P** chưa các điểm ứng viên phân đoạn, ta sinh ra các trường hợp phân đoạn ($List_{candSeg}$) bằng cách kết hợp 2, 3 các đoạn liên tiếp lại với nhau ta có:

$List_{candSeg}=\{ (SP_1, SP_2, SP_3, SP_4,..., SP_{n-1}, SP_n),$
$(SP_1+SP_2, SP_3, SP_4,...,SP_{n-1}, SP_n),$
$(SP_1+SP_2+SP_3, SP_4,...,SP_{n-1}, SP_n),$
$...,$
$(SP_1, SP_2, SP_3, SP_4,...,SP_{n-1}+SP_n)\}$

Trong đó: Mỗi hàng của $List_{candSeg}$ tương ứng với 1 cách phân đoạn cho từ đầu vào.

4. Tính xác suất của mỗi cách phân đoạn

Từ dang sách các cách phân đoạn $List_{candSeg}$. Với mỗi trường hợp phân đoạn $Cand_i$ tính xác suất *Prob_Cand* :

Giả sử *SubPart* S có một cách phân đoạn $CandSeg=\{SP_1, SP_2,...SP_n\}$, *n* là số phân đoạn trên một *SubPart*

$$P(CandSeg \mid S) = \prod_{i=0}^{n} P(SP_i) \prod_{i=0}^{n-1} P(SP_iSP_{i+1}) P(SP_1SP_2...SP_n) \quad (3)$$

Trong đó: $P(CandSeg|S)$ là xác suất của cách phân đoạn *CandSeg* đối với *SubPart S*

$P(SP_i)$ là xác suất kết quả tốt nhất đưa ra từ mô hình với đầu vào SP_i.

$P(SP_iSP_{i+1})$ là xác suất xuất hiện của cặp SP_iSP_{i+1} (sử dụng từ điển *bi-gram* mức kí tự)

$P(SP_1SP_2...SP_n)$ là xác suất xuất hiện của 1 từ (sử dụng từ điển *uni-gram* mức từ)

5. Tìm cách phân đoạn có xác suất lớn nhất

Tính: $Best_{CandSeg}$

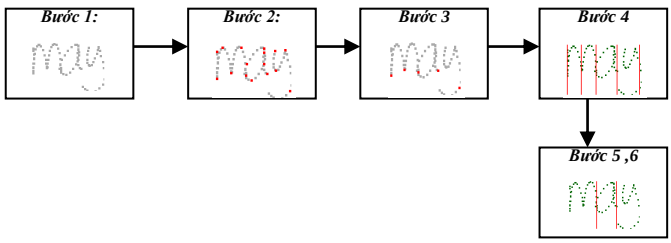
$$Best_{CandSeg} = ArgMax(P(CandSeg_i)) \quad (4)$$

Với $i = \overline{0,k}$ *, k là số lượng các phần tử của* $List_{CandSeg}$

6. Đưa ra kết quả phân đoạn

(Có thể đưa ra top-1, top-2 hoặc top-k các phân đoạn tốt nhất dựa vào xác suất của mỗi ứng viên phân đoạn đối với đầu vào SubPart)

Như vậy, toàn bộ quá trình thực hiện được thể hiện dưới đây:



Hình 12. Minh họa các kết quả đạt được sau mỗi bước thực hiện thuật toán ROVS

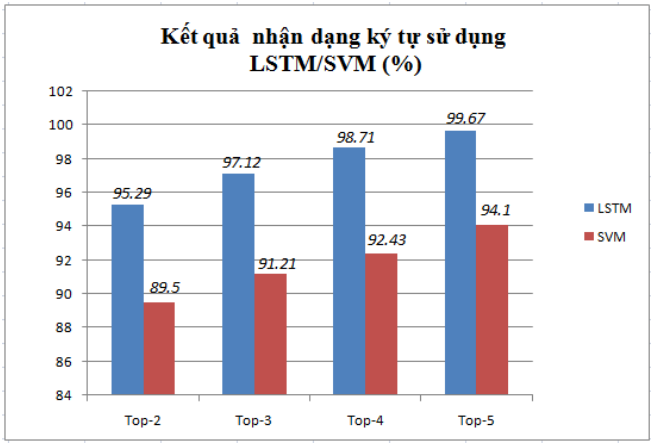
V. THỰC NGHIỆM VÀ ĐÁNH GIÁ

A. Hệ thống nhận dạng

Để kiểm thử cho phương pháp đề xuất, bài báo sử dụng mô hình nhận dạng engine nhận dạng LSTM cho nhận dạng chữ viết tay online Tiếng Anh. Cơ sở dữ liệu được sử dụng là bộ ký tự được trích xuất từ bộ dữ liệu IAM-OnDB. Bộ dữ liệu được phân thành 35 lớp, tương ứng với 35 ký tự (a-z, 1-9) trong tập dữ liệu. Bộ phân loại LSTM có số đầu vào: 8, số đầu ra 35, số noron lớp ẩn: 350. Bộ dữ liệu huấn luyện kí tự gồm: 61260 kí tự, chia làm 6 phần: 3 phần *train*, 1 phần làm *validation train* , 1 phần *test train*, 1 phần để *test* lại hệ thống. Đồng thời, cài đặt bộ phân lớp SVM với hàm hạt nhân RBF để so sánh các kết quả với LSTM.

Theo kết quả trong Hình 13, bộ phân lớp LSTM cho kết quả tốt hơn so với bộ phân lớp SVM, do đó, nhóm tác giả sử dụng kết quả nhận dạng của bộ phân lớp LSTM để tiến thành phần thực nghiệm phân đoạn ký tự và nhận dạng từ trong mục B dưới đây.

Kết quả nhận dạng của engine nhận dạng kí tự LSTM và SVM:



Hình 13. Biểu đồ kết quả nhận dạng kí tự khi sử dụng LSTM/SVM

B. Thực nghiệm phân đoạn từ, phân đoạn ký tự

Để đánh giá hệ thống cũng như hiệu quả của phương pháp đề xuất, chúng tôi sử dụng bộ dữ liệu IAM-OnDB [9], đây là bộ dữ liệu bao gồm tập hợp các điểm theo chiều chuyển động của đầu bút khi viết trên *"smart whiteboard"* của 221 người viết khác nhau. Bộ dữ liệu gồm 12,1279 dòng văn bản, 727,674 từ và có 3,638,370 ký tự. ε

Tiến hành thực nghiệm trên bộ dữ liệu IAM-OnDB với 2 phần thực nghiệm phân đoạn từ và phân đoạn ký tự với các chỉ số đánh giá sau:

Với mỗi dòng văn bản, gọi:

N_Char_Truth và N_Subpart_Truth là số *ký tự và SubPart* của dòng văn bản

N_Char_Segment và N_Subpart_Segment là số *ký tự và SubPart* mà phương pháp tìm ra.

N_Char và N_Subpart là số *ký tự và SubPart(Word)* mà thuật toán đề xuất phân đoạn đúng so với dữ liệu gốc.

Khi đó:

Precision là tỉ lệ giữa số *ký tự/ SubPart* mà thuật toán đề xuất đúng với số *ký tự/ SubPart* mà thuật toán tìm ra.

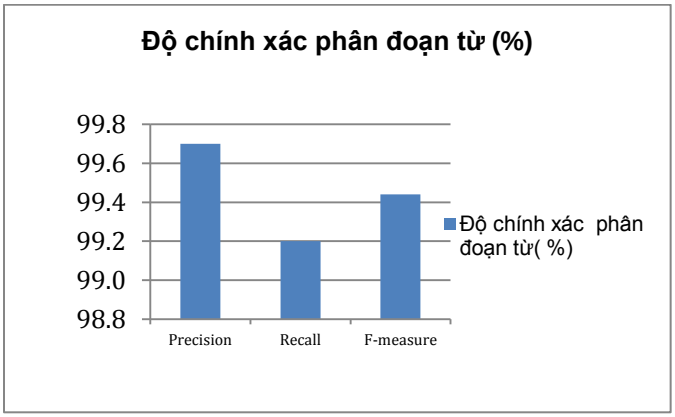
Recall là tỉ lệ giữa số *ký tự/ SubPart* mà thuật toán đề xuất với số *ký tự/ SubPart* của dòng văn bản.

$$Precision = \frac{N\_Char}{N\_Char\_Segment} \times 100\% \quad \text{hoặc} \quad Precision = \frac{N\_SubPart}{N\_SubPart\_Segment} \times 100\% \qquad (5)$$

$$Recall = \frac{N\_Char}{N\_Char\_Truth} \times 100\% \quad \text{hoặc} \quad Recall = \frac{N\_SubPart}{N\_SubPart\_Truth} \times 100\% \qquad (6)$$

F-measure được tính từ *Precision* và *Recall* như sau: $F\text{-}measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$ (7)

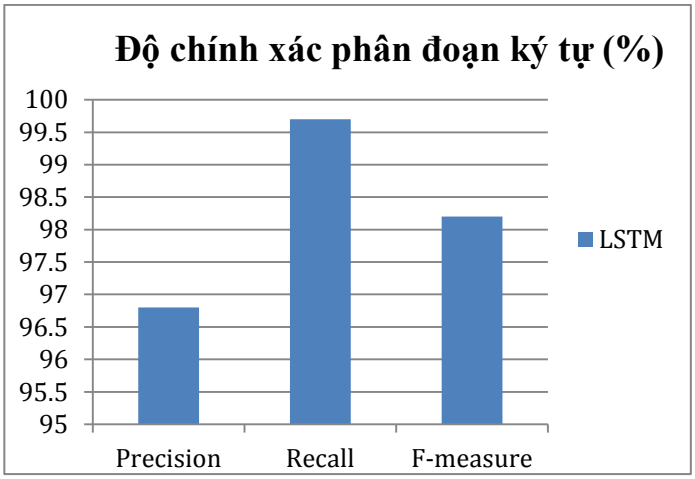
❖ Thực nghiệm tiến hành thống kê kết quả phân đoạn từ trên 200 dòng, mỗi dòng có 5 từ trích rút từ bộ dữ liệu IAM-OnDB.



Precision	*Recall*	*F-measure*
99,7	99,2	99,4

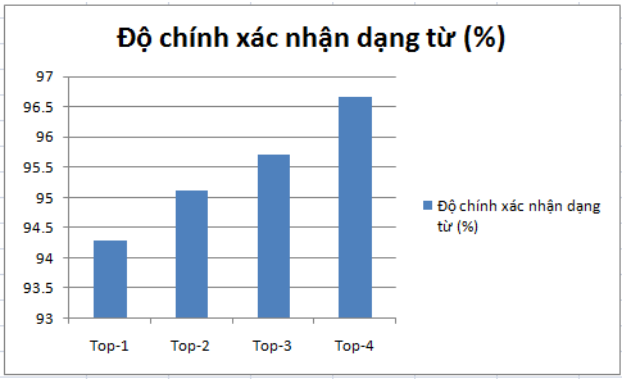
Hình 14. Biểu đồ thể hiện kết quả phân đoạn từ trên bộ dữ liệu IAM-OnDB

❖ Thực nghiệm tiến hành thống kê kết quả phân đoạn kí tự sử dụng LSTM trên 1000 từ trích rút từ tập dữ liệu IAM-OnDB.



Char Acc.	*Precision*	*Recall*	*F-measure*
LSTM	96.8	99.7	98.2

Hình 15. Biểu đồ thể hiện kết quả phân đoạn ký tự trên bộ dữ liệu IAM-OnDB



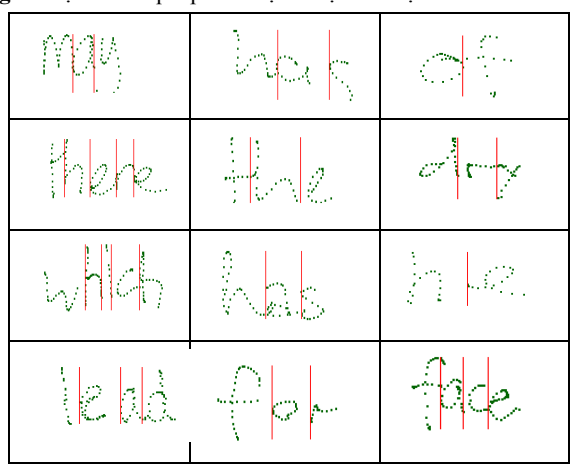
Top-1	*Top-2*	*Top-3*	*Top-4*
94.29	95.12	95.71	96.67

Hình 16. Biểu đồ kết quả nhận dạng từ khi kiểm thử trên bộ dữ liệu từ trích xuất từ dữ liệu IAM-OnDB

Do chưa thu thập được tập dữ liệu thử nghiệm chuẩn giữa các phương pháp phân đoạn khác nhau trên cùng một tập dữ liệu nên tại thời điểm hiện tại báo cáo chưa thể đưa ra được một đối sánh cụ thể giữa cách tiếp cận của chúng tôi với các cách tiếp cận khác. Tuy nhiên, từ đặc trưng của phương pháp cũng như thông qua một số kết quả phân đoạn dữ liệu thực nghiệm trên tập chữ viết tay *on-line* trích rút từ tập IAM-OnDB với các kiểu chữ khác nhau cho thấy đây là một cách tiếp cận khả thi cho việc phân đoạn ký tự.

Việc xác định các điểm đặc trưng của mỗi một từ sẽ giảm thiểu các điểm dữ liệu dư thừa, lấy được các điểm cơ bản đại diện cho mỗi nét chữ. Việc xác định nhiều điểm cắt ở cùng một thời điểm, kết hợp với sử dụng các kết quả thống kê ngữ cảnh ngay trong quá trình nhận dạng đảm bảo cho thuật toán luôn chọn được vị trí cắt chính xác nhất tương ứng với kết quả nhận dạng tốt nhất.

Bảng 1. Một số kết quả phân đoạn kí tự cho một số kiểu từ điển hình



Từ bảng kết quả ở trên, ta thấy thuật toán phân đoạn đề xuất đưa ra kết quả phân đoạn khá tốt đối với các kiểu chữ viết tay điển hình.Việc phân đoạn trực tiếp trên dữ liệu dạng chuỗi hạn chế các nhiễu hay các thông tin sai lệch trong quá trình chuyển đổi kiểu dữ liệu từ *on-line* sang *off-line*. Phương pháp cho kết quả tốt đối với hầu hết các kiểu chữ dính nhau mà đường nối giữa các kí tự có kích thước tương đối (không quá nhỏ).

Tuy nhiên, vẫn còn một số vấn đề chưa giải quyết được:

- Chưa giải quyết được trường hợp hai kí tự dính chồng lên nhau hoặc bị lồng nhau, chẳng hạn trường hợp dính nhau giữa chữ *"O"* và *"P"* như dưới đây:

- Đối với những kí tự có 1 nét như ***"l","i" hoặc "t"*** , khi lấy điểm đặc trưng, thường sẽ khiến phân đoạn bị sai do các điểm đặc trưng được lấy nằm trên cùng một nét thẳng đứng, do đó, khi thực hiện phân đoạn, nhát cắt phân đoạn sẽ dọc theo nét thẳng đứng của kí tự. Ví dụ dưới thể hiện cho điều đó.

Như vậy, thuật toán phân đoạn ký tự ROVS cho nhận dạng chữ viết tay *on-line* mà bài báo đề xuất cho độ chính xác khá cao. Kết quả của quá trình phân đoạn phụ thuộc ít nhiều vào engine nhận dạng ký tự mà người dùng sử dụng. Phương pháp đề xuất dễ hiểu, dễ cài đặt. Phương pháp đề xuất chạy tốt cả đối với những tập dữ liệu bị mất mát dữ liệu (hay số điểm dữ liệu ít). Tuy nhiên, đối với nhưng trường hợp viết liền nét, gộp nét thì phương pháp không thể phân đoạn ký tự được.

VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN TƯƠNG LAI

Nhận dạng chữ viết tay online thu được trên thiết bị "smart board" là một nhiệm vụ khó, vì đặc điểm của dữ liệu viết tay phức tạp hơn nhiều so với việc viết trên một thiết bị smart phone hay tab,.... Hiện nay độ chính xác của các phương pháp nhận dạng cho bộ dữ liệu viết tay online English mới chỉ dừng lại ở 74,0% với việc sử dụng CTC [10, 12] và 65,4% với việc sử dụng HMM [11]. Việc đề xuất phương pháp phân đoạn ký tự có độ chính xác cao sẽ giúp cải thiện chất lượng của nhận dạng chữ viết tay online.

Trong thời gian tới, tác giả sẽ tiếp tục nghiên cứu các phương pháp phân đoạn cũng như nhận dạng để giải quyết trường hợp viết liền nét, gộp nét và cải thiện độ chính xác của phương pháp nhận dạng trên bộ dữ liệu IAM-OnDB. So sánh đánh giá với các phương pháp khác trên cùng một tập dữ liệu để đưa ra nhận xét tổng quan về thuật toán đề xuất với các nghiên cứu có liên quan.

TÀI LIỆU THAM KHẢO

[1] U. Ramer, "An iterative procedure for the polygonal approximation of plan closed curves", Computer Graphics and Image Processing vol.1, no.3, pp.2440256, 1972.

[2] H. Bunke. "Recognition of cursive roman handwriting -past present and future". In Proc. 7th Int. Conf. on Document Analysis and Recognition, volume 1, pages 448-459,2003.

[3] R. Plamondon and S. N. Srihari. "On-line and off-line handwriting recognition: A comprehensive survey". IEEE Trans. Pattern Anal. Mach. Intell., 22(1):63-84, 2000.

[4] A. Vinciarelli. "A survey on off-line cursive script recognition." Pattern Recognition, 35(7):1433-1446, 2002.

[5] S. Jaeger, S. Manke, J. Reichert, and A. Waibel."Online handwriting recognition: the NPen++ recognizer". Int. Journal on Document Analysis and Recognition, 3(3):169-180, 2001.

[6] M. Schenkel, I. Guyon, and D. Henderson."On-line cursive script recognition using time delay neural networks and hidden Markov models". Machine Vision and Applications, 8:215-223, 1995.

[7] M.Liwicki and H.Bunke, "IAM-OnDB an On-line English sentence database acquired from handwritten text on a whiteboard", Prob.8th Int. Conf. Document Anal. and Recognit. (ICDAR), pp 956-961, 2005.

[8] Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., and Schmidhuber, J. "A novel connectionist system for improved unconstrained handwriting recognition". IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(5), 2009.

[9] Y. SJ, R. NH, and T. JHS. "Token passing: A simple conceptual model for connected speech recognition systems". Technical report, Cambridge University Engineering Dept, 1989.

[10] A. Graves, S. Fern' andez, F. Gomez, and J. Schmidhuber. "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks". In Proc. Int. Conf. on Machine Learning, pages 369–376, 2006.

[11] M. Liwicki and H. Bunke. "HMM-based on-line recognition of handwritten whiteboard notes". In Proc. 10th Int. Workshop on Frontiers in Handwriting Recognition", pages 595–599, 2006.

[12] Liwicki, Marcus, Graves, Alex, Bunke, Horst, and Schmidhuber, Juergen. "A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks". In Proceedings of ICDAR, 2007.

[13] D. V. Phạm, "Online handwriting recognition using multi convolution neural networks," in Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning, pp. 310-319, Springer, Hanoi, Vietnam, 2012.

[14] D. Llorens et al.: "The UJIpenchars Database: A Pen-Based Database of Isolated Handwritten Characters" Proc. of the 6th International Conference on Language Resources and Evaluation. 2008.

A CHARACTER SEGMENTATION METHOD FOR ONLINE HANDWRITING RECOGNITION

Nguyen Thi Thanh Nga, Bui Thu Lam, Nguyen Duc Dung

TÓM TẮT: *In this paper, we propose an online characters segmenting for Online handwriting recognition. The proposed method uses the Ramer sampling algorithm to create over-segmentation to improve the accuracy and speed of character segmentation. A recognition module uses deep neural network to provide the best segmentation score. The experiment was conducted on the IAM-OnDB data set, an online English handwritten character set acquired by the "smart whiteboard". The results show that the proposed method is capable of accurately segmenting characters in online handwriting recognition.*