

**TRƯỜNG ĐẠI HỌC HÀNG HẢI  
KHOA CÔNG NGHỆ THÔNG TIN**



**THUYẾT MINH  
ĐỀ TÀI NCKH CẤP TRƯỜNG**

**ĐỀ TÀI**

**ỨNG DỤNG THƯ VIỆN LẬP TRÌNH MÃ NGUỒN MỞ XÂY  
DỰNG CHƯƠNG TRÌNH NHẬN DẠNG VĂN BẢN  
CHỮ VIỆT, ANH TỬ ẢNH SỐ.**

**Chủ nhiệm đề tài: Th.S PHẠM TUẤN ĐẠT  
Thành viên tham gia: Th.S NGUYỄN VĂN THỦY**

**Hải Phòng, tháng 5/2016**

# MỤC LỤC

MỤC LỤC .....	i
MỞ ĐẦU .....	1
CHƯƠNG 1 CƠ SỞ LÝ THUYẾT .....	3
1.1. Nhị phân hóa ảnh văn bản.....	3
1.2. Cải thiện hình ảnh văn bản.....	4
1.3. Xác định góc nghiêng ảnh văn bản .....	5
1.4. Tách dòng văn bản, ký tự.....	7
1.5. Giải thuật nhận dạng ký tự quang học .....	8
1.5.1. Ứng dụng logic mờ trong nhận dạng mẫu .....	8
1.5.2. Ứng dụng mạng nơ – ron trong nhận dạng mẫu .....	10
CHƯƠNG 2 THƯ VIỆN NHẬN DẠNG TESSERACT.....	15
2.1    Ứng dụng nhận dạng ký tự quang học .....	15
2.2    Thư viện Tesseract .....	16
2.2.1    Quá trình hình thành Tesseract .....	16
2.2.2    Chức năng của Tesseract.....	17
2.2.3    Kiến trúc giải thuật nhận dạng chữ in .....	17
2.3    Huấn luyện dữ liệu nhận dạng với Tesseract .....	20
2.3.1    Tạo dữ liệu huấn luyện.....	21
2.3.2    Thiết lập các tệp cấu hình huấn luyện.....	24
2.3.3    Huấn luyện dữ liệu .....	24
CHƯƠNG 3 CHƯƠNG TRÌNH NHẬN DẠNG VĂN BẢN .....	26
3.1    Ngôn ngữ lập trình và những thư viện được sử dụng .....	26
3.1.1    Ngôn ngữ lập trình .....	26
3.1.2    Những thư viện được sử dụng.....	28
3.2 Chức năng chương trình.....	30
3.2.1 Thu nhận ảnh.....	30

3.2.2 Tiền xử lý .....	30
3.2.3 Nhận dạng .....	30
3.2.4 Hậu xử lý .....	31
3.2.5 Hiện thị và lưu trữ .....	31
3.3 Giao diện chương trình .....	31
KẾT LUẬN .....	35
I. Đánh giá kết quả .....	35
II. Hướng phát triển của đề tài .....	35
TÀI LIỆU THAM KHẢO .....	36

## DANH SÁCH BẢNG BIỂU

<b>Thứ tự</b>	<b>Tiêu đề bảng</b>	<b>Trang</b>
Bảng 1.1	Tập ký tự số	9
Bảng 1.2	Tập véc tơ đặc trưng	9
Bảng 1.3	Kết quả đối sánh ký tự số	10
Bảng 2.1	Thuộc tính phong chữ	24
Bảng 3.1	Nhận dạng một vùng văn bản	32
Bảng 3.2	Nhận dạng ảnh văn bản có góc nghiêng $10^\circ$	33
Bảng 3.3	Nhận dạng ảnh văn bản với phong và cỡ chữ khác nhau	33
Bảng 3.4	Nhận dạng ảnh văn bản có các dòng cong	34

## DANH SÁCH HÌNH ẢNH

<b>Thứ tự</b>	<b>Tiêu đề hình ảnh</b>	<b>Trang</b>
Hình 1.1	Đường thẳng và góc nghiêng	6
Hình 1.2	Đường thẳng đi qua 3 điểm	6
Hình 1.3	Văn bản nghiêng	6
Hình 1.4	Tách dòng và xác chọn vùng ký tự	7
Hình 1.5	Nút nơ – ron nhân tạo	11
Hình 1.6	Mạng truyền thẳng nhiều tầng	13
Hình 2.1	Quy trình xử lý của một ứng dụng nhận dạng ký tự quang học	15
Hình 2.2	Kiến trúc nhận dạng văn bản chữ in trong Tesseract	17
Hình 2.3	Đường cơ sở hình cong	18
Hình 2.4	Cắt các ký tự liền nhau	18
Hình 2.5	Sơ đồ nhận dạng từ	19
Hình 2.6	Các đặc trưng ký tự được nhận dạng	19
Hình 2.7	Sơ đồ huấn luyện dữ liệu của Tesseract	20
Hình 2.8	Các chức năng chính của bộ biên tập văn bản mẫu	21
Hình 2.9	Nhận dạng phác thảo ký tự	23
Hình 2.10	Kết quả huấn luyện dữ liệu	25
Hình 2.11	Ứng dụng Java chạy trên nhiều hệ điều hành	26
Hình 2.12	Cơ chế thông dịch java	27
Hình 2.13	Chức năng chính trong chương trình	30
Hình 3.1	Giao diện chương trình chính	32

## DANH MỤC THUẬT NGỮ, CHỮ VIẾT TẮT

<b>Thuật ngữ, chữ viết tắt</b>	<b>Ý nghĩa cụm từ</b>	<b>Trang</b>
Activation function	Hàm kích hoạt	11
Actual response	Đáp ứng ra thực tế	11
Adaptive classifier	Phân loại thích ứng	20
Adaptive thresholding	Ngưỡng thích nghi	4
Anti-aliasing	Khử răng cưa	21
ANN-Artificial neural network	Mạng nơ – ron nhân tạo	10
Associate word	Ghép từ	19
Back – propagation learning algorithm	Giải thuật lan truyền ngược	13
Baseline	Dòng cơ sở	18
Bounding box	Hộp biên	23
Clustering	Tách cụm	10
Connected component analysis	Phân tích thành phần liên thông	18
De-skew	Khử độ nghiêng	5
Desired response	Đáp ứng mong muốn	11
Features of character	Những đặc trưng của ký tự	19
Feed-forward neural network	Mạng nơ – ron truyền thẳng	13
Find text lines and words	Tìm dòng và từ	18
Fundamental pattern	Mẫu cơ sở	9
Fuzzy logic	Logic mờ	8
Gradient	Độ dốc	14
Hidden layer	Tầng ẩn	13
Hough transform	Biến đổi Hough	5
Input layer	Tầng vào	13
JDK – java development kit	Bộ phát triển Java	27
JNA, Tess4J, JOrtho	Tên 3 thư viện lập trình mã nguồn mở sử dụng với Tesseract	29

<b>Thuật ngữ, chữ viết tắt</b>	<b>Ý nghĩa cụm từ</b>	<b>Trang</b>
JRE – java realtime environment	Môi trường thời gian thực Java	27
JVM – java virtual machine	Máy ảo Java	27
Letter tracking	Dính chập ký tự	21
Matching	Đối sánh	9
OCR-Optical character recognition	Nhận dạng ký tự quang học	15
Output layer	Tầng ra	13
PSM - Page Segment Mode	Chế độ phân đoạn trang	30
Parse number	Phân tách số	19
Pattern Recognition	Nhận dạng mẫu	10
Post-processing	Hậu xử lý	31
Pre-processing	Tiền xử lý	4
Sharpening filter	Bộ lọc tăng cường độ sắc nét	4
Skewed document	Văn bản nghiêng	5
Smoothing filter	Bộ lọc trơn mịn ảnh	4
Static classifier	Phân loại tĩnh	20
Tolerance	Dung sai	11
Total square error	Sai số bình phương toàn phần	13

## MỞ ĐẦU

### 1. Tính cấp thiết của vấn đề nghiên cứu

Ngày nay các loại sách báo, tư liệu cần được lưu trữ dưới dạng văn bản số rất phổ biến. Văn bản số có ưu điểm như cập nhật, sửa chữa, cũng như trao đổi nhanh chóng hơn so với văn bản in giấy truyền thống. Mặt khác, qua thời gian thì chất lượng văn bản in giấy sẽ kém đi nhưng văn bản số vẫn không bị hỏng. Từ đó, nảy sinh vấn đề làm cách nào để khôi phục lại những thông tin của sách báo dưới dạng văn bản số để có thể tái bản. Đây là một nhiệm vụ thực tế trong nhiều lĩnh vực, chẳng hạn như trong các thư viện và nhà xuất bản.

Có một số cách khác nhau để giải quyết bài toán chuyển đổi trên. Một biện pháp dễ thực hiện nhất là nhập lại nội dung của văn bản thông qua bàn phím. Mặc dù vậy, đây là một công việc thủ công trong thao tác chế bản nên nếu số lượng văn bản là quá lớn và mất nhiều thời gian sẽ dẫn tới nhiều sai sót. Giải pháp khác là tạo ra một chương trình nhận dạng văn bản tự động. Theo hướng này, sách báo được máy quét lưu trữ dưới dạng ảnh số, chương trình có chức năng nhận dạng ký tự và từ, từ đó chuyển đổi thành văn bản số.

### 2. Tổng quan về tình hình nghiên cứu thuộc lĩnh vực đề tài

Hiện nay, có nhiều phần mềm nhận dạng ký tự nhưng chúng được phát triển cho mục tiêu thương mại. Tuy nhiên, một thư viện mã nguồn mở viết trên C/C++ được công bố rộng rãi là Tesseract. Dựa vào thư viện này, những lập trình viên đã phát triển chúng thành những ứng dụng nhận dạng các ngôn ngữ khác nhau. Thư viện mở được viết bằng C/C++ nên nó có thể được biên dịch để chạy trên các hệ điều hành khác nhau. Hơn nữa, người viết có thể xây dựng bộ dữ liệu từ điển để nhận dạng ngôn ngữ riêng để ngày càng trở nên hoàn thiện. Có những ngôn ngữ như tiếng Anh đã được nghiên cứu cho kết quả nhận dạng văn bản chữ in gần như hoàn hảo nhưng còn nhiều ngôn ngữ khác như văn bản in chữ Việt thì hiệu quả còn chưa cao.

### 3. Mục tiêu, đối tượng, phạm vi nghiên cứu

Trong nội dung của đề tài nghiên cứu “Ứng dụng thư viện lập trình mã nguồn mở xây dựng chương trình nhận dạng văn bản chữ Việt, Anh từ ảnh số”, mục tiêu quan trọng là áp dụng thư viện mã nguồn mở Tesseract tạo ra bộ dữ liệu từ điển tiếng



Việt và tiếng Anh, từ đó khôi phục văn bản tiếng Anh và Việt thông qua máy quét. Mặt khác, nghiên cứu đề tài còn là để nắm bắt được những giải thuật cơ sở đã sử dụng để cài đặt trong thư viện mã nguồn mở như giải thuật xử lý đối tượng ảnh, nhận dạng đối tượng dựa trên mạng nơ – ron. Thực tế trong năm trước, sinh viên ngành Công Nghệ Thông Tin Đại học Hàng Hải cũng đã tiếp cận và tìm hiểu thư viện mã nguồn mở trên và tiến hành thực hiện luận văn tốt nghiệp.

#### **4. Phương pháp nghiên cứu, kết cấu của công trình nghiên cứu**

Nội dung của báo cáo đề tài chia thành 3 chương, chương 1 giới thiệu kết quả nghiên cứu những giải thuật cơ sở đã được áp dụng trong quá trình xây dựng thư viện nhận dạng mã nguồn mở Tesseract. Chương 2 trình bày tổng quan các chức năng của thư viện mã nguồn mở và bộ biên tập dùng trong đào tạo dữ liệu. Chương 3 giới thiệu về ngôn ngữ dùng trong xây dựng ứng dụng nhận dạng văn bản chữ in và những kết quả thử nghiệm có được.

#### **5. Kết quả đạt được của đề tài**

Nhìn chung, kết quả nghiên cứu đã đáp ứng được những yêu cầu cơ bản của đề tài đã đặt ra ban đầu là hiểu được những giải thuật cơ sở đã áp dụng trong thư viện mã nguồn mở, xây dựng được bộ từ điển nhận dạng văn bản chữ in cho tiếng Việt và tiếng Anh, và thu được những kết quả tương đối tốt. Tuy nhiên, kết quả nghiên cứu còn có những hạn chế nhất định vì yếu tố thời gian, khả năng khai thác internet và máy tính.

## CHƯƠNG 1 CƠ SỞ LÝ THUYẾT

### 1.1. Nhị phân hóa ảnh văn bản

Trong thực tế, ảnh văn bản ban đầu là ảnh chứa nhiều màu hơn là trắng và đen. Vì vậy để có thể thực hiện được quá trình phân tích và nhận dạng, cần phải chuyển chúng thành ảnh nhị phân trong đó mỗi điểm ảnh được biểu diễn bởi một trong 2 giá trị là 0 hoặc 255. Đầu tiên, ảnh màu nhận vào sẽ được chuyển thành ảnh xám với các mức xám có giá trị từ 0 đến 255 dựa trên ba giá trị red, green, blue của ảnh đầu vào. Phương trình chuyển đổi ảnh màu sang ảnh xám:

$$\text{greycolor} = r * 0.299 + g * 0.587 + b * 0.114.$$

Từ ảnh xám này, so sánh mức xám của từng điểm với một ngưỡng cho trước để quyết định điểm đó sẽ là 0 hoặc 255, giá trị 0 biểu diễn cho màu đen và 255 biểu diễn cho màu trắng. Một trong các phương pháp là giải thuật Otsu đề nghị để tìm ra ngưỡng thích hợp đối với mỗi ảnh nhận vào.

Cho trước ảnh đa mức xám có L mức sáng, ký hiệu  $p(i)$  là tần suất xuất hiện của mức sáng thứ i và các trọng số tần suất  $\omega_0(t)$ ,  $\omega_1(t)$  dựa theo ngưỡng t như sau:

$$\begin{cases} \omega_0(t) = \sum_{i=0}^{t-1} p(i) \\ \omega_1(t) = \sum_{i=t}^{L-1} p(i) \end{cases} \quad 1.1$$

$$\text{Các hàm thuộc: } \begin{cases} \mu_0(t) = \sum_{i=0}^{t-1} i.p(i) / \omega_0(t) \\ \mu_1(t) = \sum_{i=t}^{L-1} i.p(i) / \omega_1(t) \\ \mu_T(t) = \sum_{i=0}^{L-1} i.p(i) \end{cases} \quad 1.2$$

Từ đó, suy ra sự liên hệ giữa những trọng số và các hàm thuộc:

$$\begin{cases} \omega_0(t) \mu_0(t) + \omega_1(t) \mu_1(t) = \mu_T(t) \\ \omega_0(t) + \omega_1(t) = 1 \end{cases} \quad 1.3$$

$$\text{Otsu định nghĩa giá trị: } \sigma_b^2(t) = \omega_0(t) \omega_1(t) (\mu_0(t) - \mu_1(t))^2, \quad \forall 0 \leq t \leq L-1 \quad 1.4$$

và Otsu cho rằng ngưỡng thích hợp là giá trị lớn nhất trong số các giá trị  $\sigma_b^2(t)$ . Như

thế, ngưỡng được chọn phụ thuộc theo đặc trưng mức sáng trong ảnh số, và do vậy ngưỡng của giải thuật Otsu được xem là ngưỡng thích nghi (Adaptive thresholding).

## 1.2. Cải thiện hình ảnh văn bản

Mục tiêu của cải thiện ảnh số là những chức năng xử lý ảnh nâng chất lượng từ ảnh ban đầu và phù hợp với ứng dụng đặc trưng. Đối với bài toán nhận dạng văn bản, ảnh số được tạo ra từ những trang văn bản với thiết bị quét không phải lúc nào cũng cho hình ảnh tốt nhất, vì văn bản gốc lâu năm nên ảnh quét có chữ viết mờ hay mất nét, có thể có nhiều. Trong mục này, ta xem xét hai kỹ thuật nâng cao chất lượng ảnh số văn bản phục vụ cho quá trình nhận dạng là làm mịn ảnh và tăng cường sắc nét ảnh.

### 1.1.1 Mịn ảnh

Mịn ảnh được thực hiện dựa trên bộ lọc trơn (Smoothing filter) nhằm loại nhiễu, bước này dùng trong quá trình tiền xử lý (Pre-processing) khi phải giảm bớt một số chi tiết không cần thiết của một đối tượng nào đó trong ảnh. Một hướng áp dụng phổ biến để giảm nhiễu là lọc tuyến tính, những bộ lọc tuyến tính theo hướng này được biết đến như là lọc thông thấp.

Ý tưởng cho những bộ lọc thông thấp là thay thế giá trị mức sáng của mọi điểm ảnh bằng giá trị mức sáng trung bình của các hàng xóm, định nghĩa theo mặt nạ lọc. Kết quả trên dẫn tới ảnh số văn bản mất đi những chi tiết nhiễu, ma trận của một bộ lọc làm mịn ảnh thường sử dụng có các hệ số như sau:

$$M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \text{ hoặc } M = \begin{bmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{bmatrix} \quad 1.5$$

### 1.1.2 Tăng cường sắc nét ảnh

Trái ngược với bộ lọc mịn ảnh, bộ lọc tăng cường độ sắc nét (Sharpening filter) để nhấn mạnh hay cải thiện chi tiết bị mờ của đối tượng đang xét trong ảnh văn bản, ví dụ như dấu của các chữ cái không rõ ràng. Qua những bộ lọc loại này, bức ảnh màu tối có mức sáng trung bình của toàn bộ điểm ảnh được tăng cường. Ma trận của một loại bộ lọc tăng cường độ sắc nét ảnh thường sử dụng có các hệ số như sau:

$$M = \begin{bmatrix} -1 & -1 & -1 \\ -1 & A+8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \text{hoặc} \quad M = \begin{bmatrix} 0 & -1 & 0 \\ -1 & A+4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad 1.6$$

Nếu cần làm sắc nét ảnh thì chọn hệ số A là 1, còn khi A lớn hơn 1 thì mức sáng trung bình của các điểm ảnh tăng.

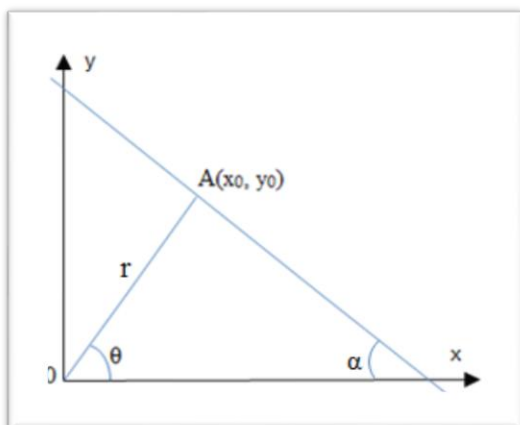
### 1.3. Xác định góc nghiêng ảnh văn bản

Quá trình sao chụp hay quét ảnh không chuẩn dẫn tới văn bản bị nghiêng (Skewed document). Nếu văn bản bị nghiêng thì nó sẽ ảnh hưởng đến các bước tiếp theo của giải thuật nhận dạng ngay khi góc nghiêng chỉ cỡ khoảng  $5^\circ$ , và khiến cho hiệu quả nhận dạng giảm sút.

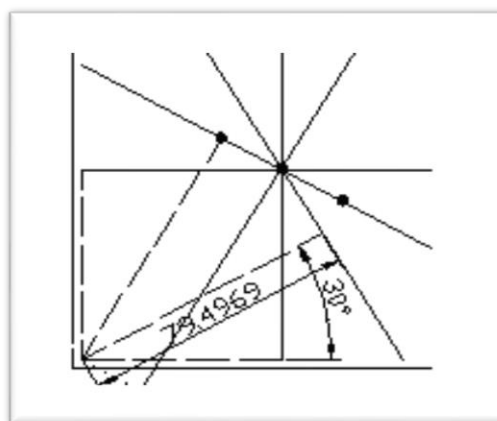
Đã có nhiều hướng tiếp cận nhằm khắc phục vấn đề trên ở nhiều mức độ khác nhau. Có hai tiêu chuẩn cơ bản để khử độ nghiêng (De-skew) của ảnh văn bản: Tiêu chuẩn đầu tiên là giới hạn góc ước lượng, ví dụ góc ước lượng của văn bản giới hạn trong khoảng giới hạn nào đó. Tiêu chuẩn còn lại là số lượng góc nghiêng trong toàn văn bản nghĩa là văn bản có một hay nhiều góc nghiêng. Trong trường hợp này, ta hãy xét văn bản cùng có một góc nghiêng nhỏ.

Nếu cho trước tập các đỉnh phân biệt trong mặt phẳng, cần kiểm tra xem chúng có tạo thành đường thẳng không. Mỗi cặp điểm khác nhau tạo thành một đường thẳng và n điểm tạo thành  $n(n-1)/2$  đường thẳng, với mỗi đường thẳng cần kiểm tra  $n-2$  điểm còn lại có thuộc vào đường thẳng đó không, như thế sẽ cần  $O(n^3)$  phép thử. Với ảnh số kích thước lớn thì hướng giải quyết trên tạo ra số lượng phép tính bùng nổ. Do đó, có một lựa chọn khác là sử dụng biến đổi Hough (Hough transform). Giải thuật đã được áp dụng trong việc phát hiện những đối tượng hình học cơ sở như đường thẳng, đường tròn hay elip. Từ đó, ý tưởng biến đổi Hough được áp dụng ước lượng góc nghiêng văn bản nhằm tối ưu số lượng phép tính.

Biến đổi Hough tính toán theo phương trình tọa độ cực:  $r = x_0 \cos \theta + y_0 \sin \theta$ , trong đó r là khoảng cách nhỏ nhất giữa gốc tọa độ và đường thẳng,  $\theta$  là góc tạo bởi trục hoành với đoạn thẳng OA. Như hình 1.1 thì góc nghiêng giữa đường thẳng và trục hoành là  $\alpha$  bằng  $90 - \theta$ .



Hình 1.1 Đường thẳng và góc nghiêng



Hình 1.2 Đường thẳng đi qua 3 điểm

Tập các đoạn thẳng trong mặt phẳng xác định từ những cặp tham số  $(r, \theta)$ . Ta xét trường hợp ảnh đầu vào có màu đen trắng, chỉ chứa các dòng văn bản và sử dụng một phong chữ. Có thể coi các dòng văn bản song song với nhau nên góc nghiêng toàn văn bản là góc nghiêng của một dòng, như trong hình 1.3. Trước tiên, ta tìm ra tập các điểm màu đen dưới chân của một dòng văn bản nào đó, với mỗi điểm như thế duyệt các góc  $\theta$  trong khoảng giới hạn và ước lượng các khoảng cách  $r$  tương ứng. Với mỗi cặp tham số  $(r, \theta)$  ghi nhận số lượng điểm trong tập các điểm màu đen đang xét thuộc vào đường thẳng tương ứng cặp tham số đó. Đường thẳng nào đi qua nhiều điểm màu đen nhất sẽ có tham số  $\theta$  cần tìm.

Hình 1.2 phía trên mô tả có những đường thẳng khác nhau tương ứng các cặp  $(r, \theta)$  đi qua một điểm, nhưng chỉ có một đường thẳng thỏa mãn điều kiện đi qua 3 điểm.



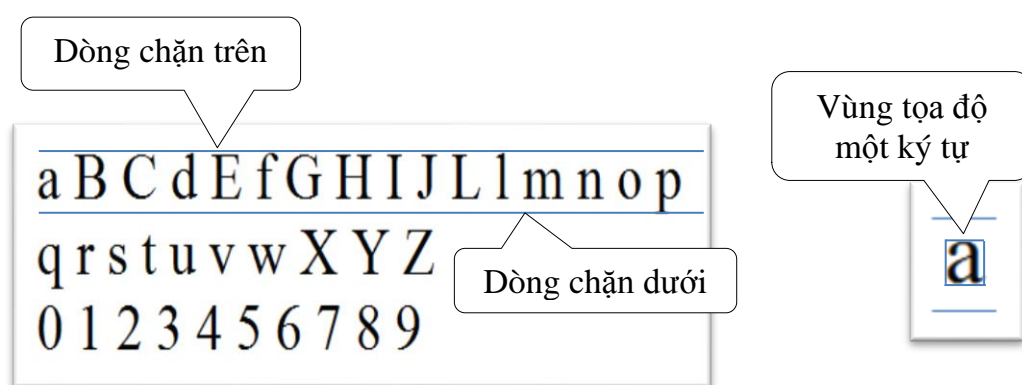
Hình 1.3 Văn bản nghiêng

Trong một dòng văn bản, số lượng ký tự có hạn nên tập các điểm màu đen dưới chân dòng đó có số lượng nhỏ. Hơn nữa,  $\theta$  được chọn giá trị rời rạc theo bước trong ngưỡng  $(-90^\circ, 90^\circ)$ . Từ đó kích thước ma trận chứa các cặp  $(r, \theta)$  sẽ không lớn.

#### 1.4. Tách dòng văn bản, ký tự

Để nhận dạng được toàn bộ văn bản trong ảnh số, ta phải nhận dạng được các dòng và các ký tự trong ảnh. Sau khi qua tiền xử lý, dựa trên các đặc trưng văn bản sẽ tách được các dòng và từ, ký tự trong ảnh. Mỗi dòng văn bản luôn có tọa độ chặn dưới và chặn trên, trong khi đó mỗi ký tự có tọa độ chặn dưới, chặn trên, giới hạn trái và giới hạn phải.

Giải thuật xác định tọa độ chặn trên và dưới của mỗi dòng văn bản được mô tả tóm tắt như sau: Từ tọa độ ban đầu  $(0,0)$ , duyệt theo chiều ngang để tìm điểm có màu đen, nếu đã hết dòng mà vẫn chưa thấy thì bắt đầu duyệt lại từ đầu dòng kế tiếp. Nếu tìm thấy điểm có màu đen thì ghi nhận tung độ điểm đó là tung độ của dòng chặn trên và dừng duyệt. Tương tự với tọa độ chặn dưới, xuất phát từ điểm có hoành độ là 0 và tung độ bằng tung độ của dòng chặn trên, cũng duyệt theo chiều ngang, nếu sau hết dòng không thấy điểm đen nào thì ghi nhận tung độ dòng đang xét là tung độ của dòng chặn dưới, còn nếu tìm thấy điểm đen thì lại xét lại từ dòng kế tiếp.



Hình 1.4 Tách dòng cơ sở và xác chọn vùng ký tự

Lặp lại các bước trên để tìm tọa độ chặn trên và chặn dưới cho những dòng còn lại trong ảnh văn bản.

Giải thuật xác định vùng tọa độ cho mỗi ký tự như sau: Có được tọa độ giới hạn mỗi dòng, xác định được tọa độ chặn dưới và chặn trên của mỗi ký tự trên dòng đó.

Trong khi đó, để tìm tọa độ giới hạn trái và phải của một ký tự, bắt đầu từ đầu dòng chặn trên, duyệt theo chiều dọc tới tung độ của dòng chặn dưới, nếu gặp điểm màu đen thì ghi nhận hoành độ điểm đó là hoành độ của cột giới hạn trái, nếu không thấy điểm màu đen thì tiếp tục lại từ đầu cột kế tiếp. Tương tự với tọa độ giới hạn phải, bắt đầu từ đầu dòng chặn trên, duyệt theo chiều dọc tới tung độ của dòng chặn dưới, nếu sau hết cột không tìm thấy điểm màu đen thì hoành độ cột đang xét là hoành độ cột giới hạn phải của ký tự, nếu tìm thấy thì tiếp tục lại từ đầu cột kế tiếp.

## 1.5. Giải thuật nhận dạng ký tự quang học

### 1.5.1. Ứng dụng logic mờ trong nhận dạng mẫu


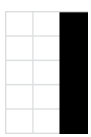


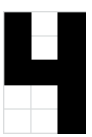


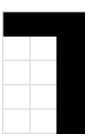


Hệ logic đơn giản nhất là logic mệnh đề, bất cứ một mệnh đề chỉ có thể nhận một trong hai giá trị là đúng hay sai. Các mệnh đề kết hợp với nhau qua các phép toán phủ định, và, hoặc, kéo theo... Nhược điểm của logic mệnh đề là nó thiếu cơ chế diễn tả các quan hệ giữa các đối tượng, nó cũng không tổng quát hóa được các đối tượng trong tự nhiên.

Logic vị từ là một phương tiện để nâng cao tính rõ nghĩa của logic mệnh đề. Sự tổng quát hóa của nó cho phép ta biểu diễn tri thức cũng như lập luận về các đối tượng và các thực thể quan hệ. Cần phải nhấn mạnh rằng, phát biểu trong logic vị từ không mang giá trị đúng hoặc sai trừ phi các đối số nhận giá trị rõ. Tuy nhiên, logic vị từ vẫn là hệ logic hai giá trị, điều này dẫn tới sự hình thành hệ logic đa trị có giá trị thứ ba là không xác định (0.5).

Logic mờ (Fuzzy logic) được xây dựng dựa trên sự tổng quát của logic đa trị, nó cho phép lập luận trên các đối tượng thực tế được định nghĩa không rõ ràng như các thực thể quan hệ. Trong logic mờ, chỉ có các đối tượng xấp xỉ chứ không có đối tượng chính xác, do đó lập luận cũng là xấp xỉ. Một chân trị là một điểm trong khoảng  $[0, 1]$  trường hợp giá trị là số hay là cụm từ như đúng, rất đúng, sai, kém... trường hợp giá trị chân lý là ngôn ngữ. Ví dụ như thông tin dự báo thời tiết “Có mưa rải rác vài nơi” không thể biểu diễn bằng một trị chân lý 0 hay 1, nhưng nó vẫn có giá trị đúng theo số phần trăm nào đó theo công tác nghiên cứu thống kê. Trong trường hợp này, một khẳng định A kèm theo giá trị độ thuộc  $0 \leq \mu(A) \leq 1$  đo sự chính xác của A, ký hiệu là  $(A, \mu(A))$ .

Nếu như logic mệnh đề ban đầu chỉ được nghiên cứu như một ngành khoa học hình thức, thì nay nó được mở rộng trong những lĩnh vực khác nhau như hệ chuyên gia và áp dụng để giải quyết các bài toán tin học như nhận dạng đối tượng, khai thác dữ liệu, ra quyết định trong điều khiển...

Để hiểu được giải thuật logic mờ trong bài toán nhận dạng ký tự quang học, ta minh họa quá trình nhận dạng 10 ký tự số lưu trữ dưới dạng ảnh:

Ký tự	0	1	2	3	4	5	6	7	8	9
Hình dạng										

Bảng 1.1 Tập ký tự số

Giả thiết các ký tự số cùng có kích thước 3x5, nhị phân hóa các ảnh mẫu được tập véc tơ bit như bảng sau:

Ký tự mẫu	Véc tơ bit	Ký tự mẫu	Véc tơ bit
0	111101101101111	5	111100111001111
1	0011001001001001	6	111100111101111
2	111001111100111	7	111001001001001
3	111001111001111	8	111101111101111
4	101101111001001	9	111101111001111

Bảng 1.2 Tập véc tơ đặc trưng

Việc nhận dạng ký tự thực hiện sự đối sánh (matching) ký tự nhận dạng với các mẫu cơ sở (Fundamental pattern) dựa trên lựa chọn mẫu trùng khớp nhất với ký tự nhận dạng. Logic mờ áp dụng nhiều loại đối sánh khác nhau, tuy nhiên với bài toán trên, ta nêu lên một số phương trình sau:

- Khoảng cách Euclit:  $d(X, Y) = \sqrt{\sum_{i=1}^n (X_i - Y_i)^2}$  1.7

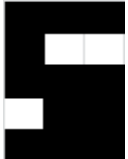
- Khoảng cách Manhattan:  $d(X, Y) = \sum_{i=1}^n |X_i - Y_i|$  1.8

- Khoảng cách Hamming:  $d(X, Y) = \sum_{i=1}^n (X_i - Y_i)$  1.9



$$- \text{Độ đo tương tự: } d(X, Y) = \frac{|\sum_{i=1}^n X_i Y_i|}{\sqrt{\sum_{i=1}^n X_i^2 \sum_{i=1}^n Y_i^2}} \quad 1.10$$

Trong đó, X và Y là đối tượng mẫu và đối tượng nhận dạng có tập véc tơ bit tương ứng là  $X_i$  và  $Y_i$ . Nếu dùng các phương trình 1.7, 1.8 hay 1.9 có hàm đối sánh nhận giá trị zero thì X và Y là đồng nhất, nhưng nếu hàm đối sánh cho giá trị gần nhất với zero thì X có thể xem như là Y. Trái lại, dùng độ đo tương tự thì ta có bảng đối sánh nhận dạng ký tự số dưới đây:

Ký tự mẫu	Véc tơ bit	Ký tự nhận dạng và véc tơ bit	Độ thuộc đối sánh
0	111101101101111	 111100111011111	0.833
1	001001001001001		0.516
2	111001111100111		0.783
3	111001111001111		0.87
4	101101111001001		0.77
<b>5</b>	<b>111100111001111</b>		<b>0.957</b>
6	111100111101111		0.917
7	111001001001001		0.655
8	111101111101111		0.881
9	111101111001111		0.917

Bảng 1.3 Kết quả đối sánh ký tự số

Độ thuộc lớn nhất xấp xỉ 1, vậy ký tự nhận dạng là ký tự 5.

### 1.5.2. Ứng dụng mạng nơ – ron trong nhận dạng mẫu

Mạng nơ – ron nhân tạo (ANN) là một mô hình tính toán được xây dựng dựa trên các đặc trưng cơ bản của nơ – ron sinh học. Mạng chứa các nút và xử lý thông tin bằng cách truyền theo các kết nối và tính giá trị mới tại các nút.

Lĩnh vực nghiên cứu điển hình của mạng nơ – ron trong phân lớp, tách cụm (Clustering), nhận dạng mẫu (Pattern Recognition) và khai thác dữ liệu (Data mining). Nhóm mô hình này nhận tín hiệu vào và nhận dạng để phân lớp chúng. Thuật toán cần phải huấn luyện sao cho thông tin vào biến dạng ít nhiều thì mạng vẫn nhận dạng đúng bản chất của nó. Lớp các bài toán tối ưu hoặc hồi quy – tổng quát hóa cũng có thể được giải quyết với mạng, qua hồi quy tuyến tính và phi tuyến người ta tìm ra các

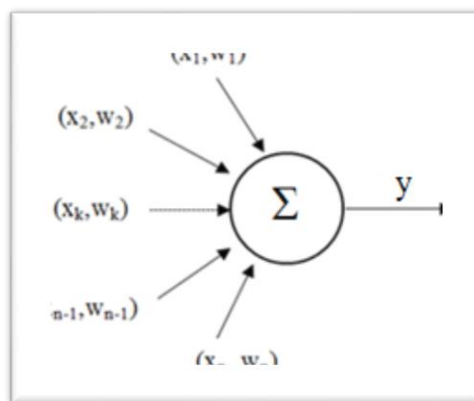
đường thẳng hoặc đường phi tuyến trơn gần khớp với mẫu. Ngoài ra là lĩnh vực hoàn chỉnh dạng, nếu dữ liệu bị mất đi một phần thì nó cần được hoàn thiện đủ so với trạng thái ban đầu.

Ưu điểm của mạng nơ – ron là khả năng xây dựng mô hình có khả năng học dữ liệu. Chỉ cần truyền vào một tập mẫu dữ liệu thì mạng tìm được ràng buộc dữ liệu và áp dụng chúng vào quá trình tính toán mà không cần có thêm các tri thức mới. Mặt khác, mạng còn có khả năng dung thứ lỗi (Tolerance), chấp nhận những mẫu dữ liệu không hoàn toàn chính xác. Với những đặc điểm trên, mô hình thích nghi được với sự thay đổi của quy luật dữ liệu thông qua quá trình học lại của mạng.

Một nút nơ – ron nhân tạo nhận đầu vào  $x$  ( $x_1, x_2, \dots, x_n$ ) và truyền một đầu ra  $y$ .

Trạng thái bên trong của nút chứa bộ tổng thực hiện: 
$$\text{Net} = \sum_{i=1}^n w_i x_i \quad 1.11$$

Hàm số  $f$  (activation function) xác định giá trị  $y$  của nút nơ – ron theo phương trình  $f(\text{Net}, \text{thres})$ . Như hình bên thì  $w$  là tập trọng số kích hoạt còn  $\text{thres}$  là ngưỡng của nút. Sự kết hợp các nút theo cấu trúc khác nhau tạo ra những mạng nơ – ron khác nhau, như mạng truyền thẳng hay mạng nối ngược.



Hình 1.5 Nút nơ – ron nhân tạo

Dưới đây là một vài hàm biến đổi dùng trong mạng nơ – ron:

$$\text{Hard-limit : } y = \begin{cases} 1 & \text{ khi } \text{Net} > \text{thres} \\ -1 & \text{ khi } \text{Net} < \text{thres} \end{cases} \quad 1.12$$

$$\text{Gause: } y = e^{-(\text{Net}-\text{thres})^2} \quad 1.13$$

$$\text{Sigmoid : } y = \frac{1}{1 + e^{-(\text{Net} - \text{thres})}} \quad 1.14$$

Bài toán học của mạng là xác định các giá trị trọng số trên các tầng dựa trên thông tin có sẵn. Thông thường, quá trình huấn luyện được thực hiện qua phép so sánh đáp ứng ra thực tế (Actual response) với đáp ứng mong muốn (Desired response) để cực tiểu hóa hiệu số trên.

Đối với hàm tuyến tính Hard-limit cho mạng nơ – ron một nút, xét phương trình

$$f(x, w) = \text{Net} - \text{thres}, \text{ nếu đặt } w_{n+1} = -\text{thres} \text{ và } x_{n+1} = 1 \text{ thì } f(x, w) = \sum_{i=1}^n w_i x_i + w_{n+1}.$$

Cho trước 2 tập mẫu và 2 lớp khác nhau, điều kiện nhận dạng là nếu hàm  $f$  của tập nào có giá trị dương thì tập đó thuộc vào lớp thứ nhất, trái lại  $f$  có giá trị âm thì tập đó thuộc lớp thứ hai. Giải thuật đào tạo tập trọng số như sau:

1. Khởi tạo ngẫu nhiên véc tơ trọng số  $w$ , hệ số  $c$  dương.
2. Tại vòng lặp thứ  $k$ ,  $k = 1, 2, \dots, N$ :  
 nếu  $x(k)$  thuộc lớp 1 và  $[w(k)]^T \cdot [x(k)] \leq 0$  thì  $w(k+1) = w(k) + c \cdot x(k)$ ,  
 nếu  $x(k)$  thuộc lớp 2 và  $[w(k)]^T \cdot [x(k)] \geq 0$  thì  $w(k+1) = w(k) - c \cdot x(k)$ ,  
 ngược lại thì  $w(k+1) = w(k)$ .
3. Giải thuật lặp cho tới khi toàn bộ mẫu nhận dạng đúng theo điều kiện trên.

Giả thiết cần nhận dạng tập  $\{(0,0); (0,1)\}$  vào lớp thứ nhất và tập  $\{(1,0); (1,1)\}$  vào lớp thứ hai, ngưỡng chọn trước là -1. Theo giải thuật trên, ta có:

- Khởi tạo  $w = (0,0,0)$ , hệ số  $c = 1$ . Ký hiệu  $x(1) = (0,0,1)$ ,  $x(2) = (0,1,1)$ ,  $x(3) = (1,0,1)$ ,  $x(4) = (1,1,1)$ .

$$- [w(1)]^T [x(1)] = [0,0,0] \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0, \text{ suy ra } w(2) = w(1) + x(1) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$- [w(2)]^T [x(2)] = [0,0,1] \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = 1, \text{ suy ra } w(3) = w(2) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

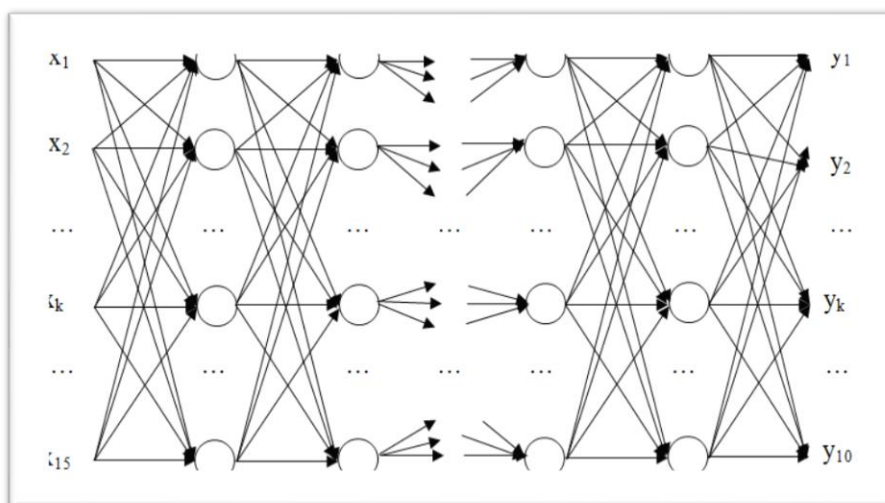
$$- [w(3)]^T [x(3)] = [0,0,1] \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = 1, \text{ suy ra } w(4) = w(3) - x(3) = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$$

$$- [w(4)]^T [x(4)] = [-1,0,0] \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = -1, \text{ suy ra } w(5) = w(4) = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$$

Tiếp tục với  $x(5) = x(1)$ ,  $x(6) = x(2), \dots$  tới khi được tập trọng số  $w = (-2, 0, 1)$ .  
 Như thế, với  $x$  là  $(0,0)$  hoặc  $(0,1)$  thì  $f(x,w)$  bằng 1, với  $x$  là  $(1,0)$  hoặc  $(1,1)$  thì  $f(x,w)$  bằng -1.

Nhược điểm của Hard-limit là nó không nhận dạng được nhiều hơn 2 lớp. Tuy nhiên, ta có thể áp dụng hàm số phi tuyến như Sigmoid trong mạng truyền thẳng (Feed-forward neural network). Ký hiệu mạng có tầng vào A có  $N_A$  nơ – ron , tầng thứ 2 là B có  $N_B$  nơ – ron,..., tầng cuối Q có  $N_Q$  nơ – ron.

Mạng nhận dữ liệu vào từ tầng nhập (Input layer) qua các tầng ẩn (Hidden layer), rồi tới tầng ra (Output layer). Số nơ – ron của tầng vào  $N_A$  là số chiều của véc tơ mẫu trong khi số nơ – ron tầng ra  $N_Q$  là số mẫu cần đào tạo để nhận dạng, như bài toán nhận dạng 10 ký tự số ở phần 1.5.1 thì mạng nơ ron có tầng đầu tiên nhận 15 giá trị vào và phân loại tối đa 10 lớp ở tầng cuối. Một mẫu ký tự thuộc lớp thứ  $i$  nếu như ở đầu ra thứ  $i$  có hàm Net bằng giá trị ngưỡng trong khi tại các đầu ra khác có hàm Net sai khác rất nhiều giá trị ngưỡng. Mặc dù vậy, cơ chế đào tạo mạng vẫn chấp nhận mẫu ký tự thuộc lớp thứ  $i$  nếu giá trị đầu ra xấp xỉ 1 trong khi các đầu ra khác có giá trị đầu ra xấp xỉ 0.



Hình 1.6 Mạng truyền thẳng nhiều tầng

Các giá trị  $x_i$  nối với các nút trên tầng thứ nhất có trọng số là  $w_{ia}$ :  $i = 1,2,..15$ ;  $a = 1,2,..N_A$ . Tập trọng số thứ hai là  $w_{ab}$  trên các cung nối tầng 2 và tầng 3,  $b = 1,2,..N_B$  ... Hai tầng cuối có tập trọng số là  $w_{pq}$ :  $p = 1,2,..N_P$ ;  $q = 1,2,..10$  . Quy trình huấn luyện tìm tập trọng số kích hoạt trong giải thuật lan truyền ngược (Back – propagation learning algorithm) được trình bày tóm tắt như sau:

Đặt hàm sai lỗi bình phương toàn phần (Total square error):

$$E_Q = \frac{\sum_{q=1}^{10} (r_q - y_q)^2}{2} \quad 1.15$$

trong đó  $r_q$  và  $y_q$  là đáp ứng mong muốn và đáp ứng tính toán tại nơ – ron thứ  $q$  tại tầng ra. Mục tiêu giờ là cần phải cực tiểu hàm  $E$ , cách thực hiện là tại mỗi tầng thì trọng số trên các cung được cập nhật theo hệ số  $\Delta w_{ij} = -\alpha \frac{\partial E_j}{\partial w_{ij}}$  sau mỗi vòng lặp cho

tới khi thỏa mãn sai số,  $\alpha$  là hệ số dương chọn trước. Trên cơ sở xác định độ dốc (gradient) của hàm sai lỗi theo các trọng số, ta xác định được hệ số điều chỉnh như sau:

$$\Delta w_{pq} = \alpha \delta_q y_p ; \delta_q = (r_q - y_q) y_q (1 - y_q) ; p = 1..N_p, q = 1..10 \quad 1.16$$

$P$  là tầng liền trước tầng cuối  $Q$ . Đối với các tầng giữa thì hệ số điều chỉnh được tính

$$\text{ngược theo phương trình } \Delta w_{ij} = \alpha \delta_j y_i ; \delta_j = y_j (1 - y_j) \sum_{k=1}^{N_k} \delta_k w_{jk} ; j = 1..N_j \quad 1.17$$

Quy ước tầng  $J$  là tầng liền trước tầng  $K$ .

Đối với một ký tự số, giải thuật học được mô tả như sau:

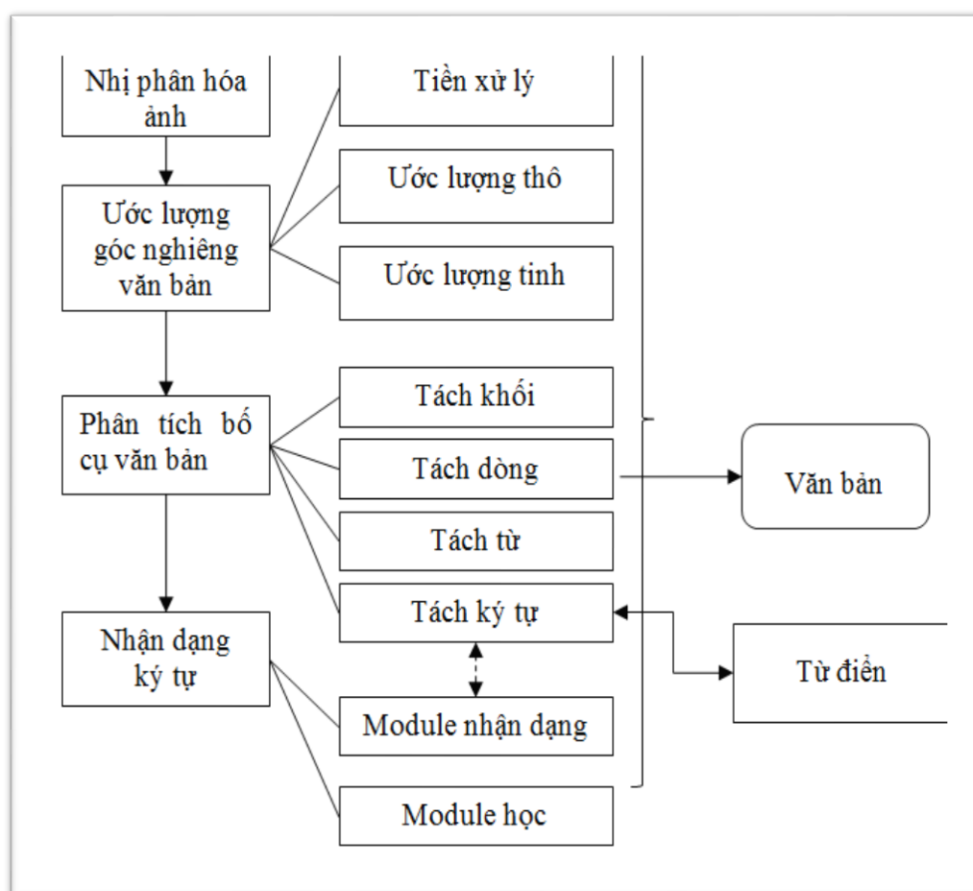
1. Nhập đầu vào  $x(x_1, x_2, \dots, x_{15})$  biểu diễn ký tự số, chọn hệ số  $\alpha$  dương trong khoảng  $(0,1)$ .
2. Với mỗi tầng trong mạng, tìm giá trị ra theo phương trình 1.14 .
3. Nếu tại tầng cuối, tìm được một đầu ra có giá trị xấp xỉ 1 tương đương đáp ứng mong muốn trong khi mọi đầu ra còn lại có giá trị xấp xỉ 0 thì dừng, tập trọng số đã xác định cho ký tự số trên. Ngược lại, sang bước 4.
4. Cập nhật trọng số kích hoạt trên các cung tại mỗi tầng:  $w_{ij} = w_{ij} - \alpha \frac{\partial E_j}{\partial w_{ij}}$  theo các trường hợp nêu trong 1.16 và 1.17 , và lặp lại 2.

## CHƯƠNG 2 THƯ VIỆN NHẬN DẠNG TESSERACT

### 2.1 Ứng dụng nhận dạng ký tự quang học

Nhận dạng ký tự quang học (OCR) là một chương trình được xây dựng để chuyển đổi các hình ảnh của chữ viết tay hoặc chữ đánh máy thành các văn bản tài liệu số. Ứng dụng OCR được bắt đầu từ một lĩnh vực nghiên cứu trong lý thuyết nhận dạng mẫu, trí tuệ nhân tạo và thị giác máy tính. Mặc dù công việc nghiên cứu học thuật vẫn tiếp tục, một phần công việc của OCR đã chuyển sang thực tế với những giải thuật đã được chứng minh. So với chữ đánh máy, nhận dạng chữ viết tay khó khăn hơn và cho hiệu quả thấp. Nhìn chung, nếu không có những thông tin thêm về ngữ pháp và ngữ cảnh thì nhận dạng chữ viết tay không dẫn tới kết quả tốt.

Ngày nay, các hệ thống nhận dạng thỏa mãn độ chính xác nhận dạng cao đối với hầu hết các phong chữ tiêu chuẩn như Unicode. Một số hệ thống còn có khả năng tái tạo lại các định dạng của tài liệu gần giống với bản gốc bao gồm hình ảnh, các cột, bảng biểu, các thành phần không phải là văn bản.



Hình 2.1 Quy trình xử lý của một ứng dụng nhận dạng ký tự quang học

Đối với văn bản in chữ Việt, ứng dụng nhận dạng chữ Việt VnDOCR có khả năng nhận dạng trực tiếp các loại tài liệu được quét từ máy quét, không cần lưu trữ dưới dạng tệp ảnh trung gian. Các trang tài liệu có thể được quét và lưu trữ dưới dạng tệp tin nhiều trang. Kết quả nhận dạng được lưu trữ sang những định dạng khác nhau như \*.txt hay \*.doc .

ABBYY, một chương trình về lĩnh vực nhận dạng ký tự quang học đã được tiến hành nghiên cứu cho tiếng Việt từ tháng 4 năm 2009. Ứng dụng nhận dạng của ABBYY chấp nhận hầu hết các định dạng ảnh đầu vào như PDF, TIFF, JPEG, GIF, PNG, BMP... Kết quả nhận dạng được lưu trữ dưới các định dạng MS Word, TXT, XML... trong đó \*. PDF là một định dạng hoàn hảo cho việc lưu trữ và khai thác tài liệu.

## **2.2 Thư viện Tesseract**

Khác với những phần mềm vì mục tiêu thương mại, Tesseract là một thư viện – không phải là chương trình – nhận dạng ký tự quang học. Nó có mã nguồn mở, được công khai dưới giấy phép Apache, phiên bản 2.0, và được phát triển dưới sự tài trợ của Google từ năm 2006. Tesseract được đánh giá là một trong số ít những thư viện nhận dạng ký tự quang học mã nguồn mở tốt nhất hiện nay.

### **2.2.1 Quá trình hình thành Tesseract**

Ban đầu, Tesseract được phát triển như một thư viện độc quyền tại phòng thí nghiệm của hãng Hewlett Packard ở Bristol - Anh và phòng thí nghiệm Greeley tại bang Colorado - Hoa Kỳ từ năm 1985 đến năm 1994. Có một số thay đổi vào năm 1996 khi Tesseract được thay đổi để viết trên hệ điều hành Windows, với phần lớn mã vẫn được viết bằng C và một số chi tiết được viết lại bằng C++. Kể từ đó cho tới nay tất cả mã của phần mềm đã được chuyển đổi hoặc ít nhất là được biên dịch với một trình biên dịch C++. Khi mà phần lõi của Tesseract viết bằng C/C++ thì nó sẽ tương thích trên nhiều nền tảng hệ điều hành khác nhau, Tuy nhiên trong vài năm tiếp theo thư viện vẫn rất ít được phát triển. Mãi cho tới năm 2005, Tesseract mới được hãng và đại học Nevada - Hoa Kỳ phát hành như một thư viện mã nguồn mở. Ngay sau đó 1 năm, Google chính thức tài trợ cho dự án này. Kể từ thời điểm này, dự án thu hút nhiều sự quan tâm của các lập trình viên cho việc xây dựng các ứng dụng nhận dạng.



## 2.2.2 Chức năng của Tesseract

Như trên đã giới thiệu, Tesseract là một thư viện mã nguồn mở để hỗ trợ xây dựng ứng dụng nhận dạng ký tự quang học được phát triển từ năm 1995. Nó có thể chạy trên Linux, Windows và Mac, tuy nhiên do nguồn lực hạn chế nên chỉ có các phiên bản chạy trên Windows được kiểm tra chặt chẽ bởi các nhà phát triển.

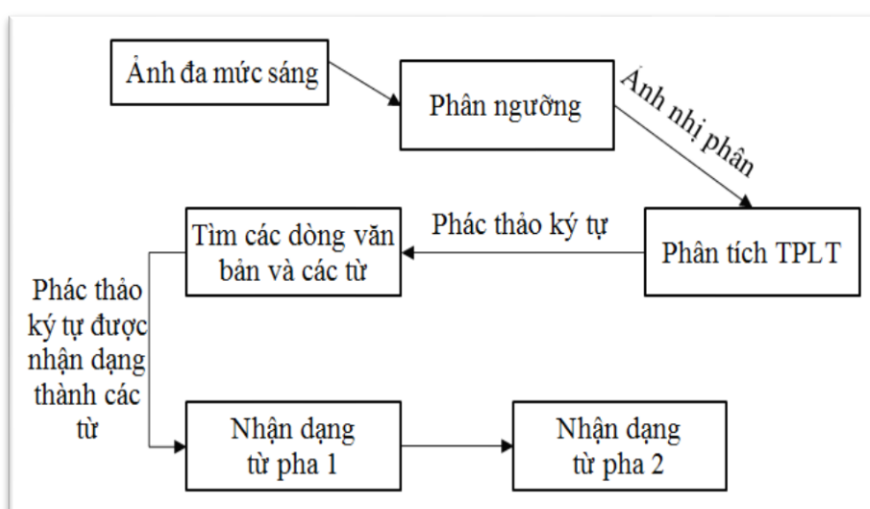
Tesseract phiên bản 2.0 chỉ có thể đọc và hiển thị hình ảnh TIFF đơn giản chứa một cột văn bản. Những phiên bản đầu tiên này cũng không cho phép phân tích bố cục đối với văn bản nhiều cột hay chứa hình ảnh hoặc các công thức toán học. Kể từ phiên bản 3.0, Tesseract đã hỗ trợ định dạng văn bản đầu ra, xác định thông tin vị trí và phân tích bố cục trang. Mặt khác, một số định dạng mới như JPG, PNG, hoặc PDF được thêm vào bằng cách sử dụng thư viện Leptonica.

Nếu như phiên bản 1.0 của Tesseract chỉ có thể nhận dạng văn bản tiếng Anh thì sau này Tesseract được xây dựng để có thể được đào tạo để làm việc trong nhiều ngôn ngữ mới. Do đó từ phiên bản 2.0, Tesseract đã cho phép nhận dạng 6 ngôn ngữ bằng cách bổ sung tiếng Pháp, Tây Ban Nha... Tới phiên bản 3.0, nó đào tạo các ngôn ngữ tương tự như ngôn ngữ Ả Rập, Trung Quốc dạng giản thể và ngôn ngữ Việt. Ngày nay, những phiên bản được cập nhật mới không chỉ mở rộng thêm nhiều ngôn ngữ mà còn nâng cao chất lượng nhận dạng.

## 2.2.3 Kiến trúc giải thuật nhận dạng chữ in

Cũng giống như hầu hết các chương trình nhận dạng ký tự quang học, Tesseract

có một kiến trúc điển hình từ trên xuống. Bước đầu bằng chức năng tiền xử lý, một ngưỡng sẽ được chọn bởi bộ phân ngưỡng thông qua một quá trình phân tích các điểm



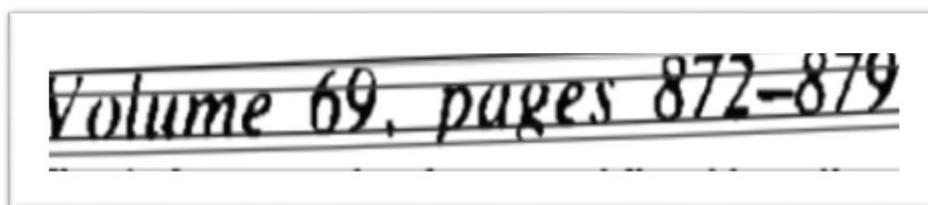
trong ảnh với giải thuật Hình 2.2 Kiến trúc nhận dạng văn bản chữ in trong Tesseract



như là Otsu, sau đó ngưỡng này sẽ được sử dụng để chuyển đổi ảnh màu hoặc ảnh xám đầu vào thành một ảnh nhị phân, giả thiết chứa các vùng văn bản hình dạng đa giác.

Giai đoạn tiếp theo là ảnh nhị phân được đưa vào bộ Phân tích thành phần liên thông (Connected component analysis) để tìm ra hình dạng phác thảo của những thành phần liên thông. Đây là một tiến trình phức tạp mất nhiều thời gian nhưng cần có để tách ra các ký tự có trong hình.

Khởi Tìm các dòng văn bản và từ (Find text lines and words) thực hiện các chức năng như xác định dòng chặn dưới và chặn trên, đối với mỗi dòng thì cắt gọn từ trước khi xác định vùng của mỗi ký tự, ngoài ra cần nhận dạng khoảng cách giữa chữ và số. Lọc dãy dòng không chỉ tìm dãy ký tự trong từng dòng mà còn phát hiện các ký tự có độ cao chênh lệch trong dòng như ký tự drop-cap, ký tự chấm câu, ký tự dấu và nhiều... Tuy nhiên, nếu ảnh số chứa các dòng có độ nghiêng hoặc cong thì giải thuật trở nên phức tạp. Để giúp giảm bớt mất thông tin khi nhận dạng ảnh nghiêng thì áp dụng giải thuật biến đổi Hough tìm góc nghiêng để đưa ảnh số trở lại vị trí thông thường. Trong trường hợp dòng có độ cong nào đó thì phải thiết lập các dòng cơ sở (Baseline) bằng cách sử dụng phương trình spline thích hợp cho nhiều phân đoạn.



Hình 2.3 Đường cơ sở hình cong

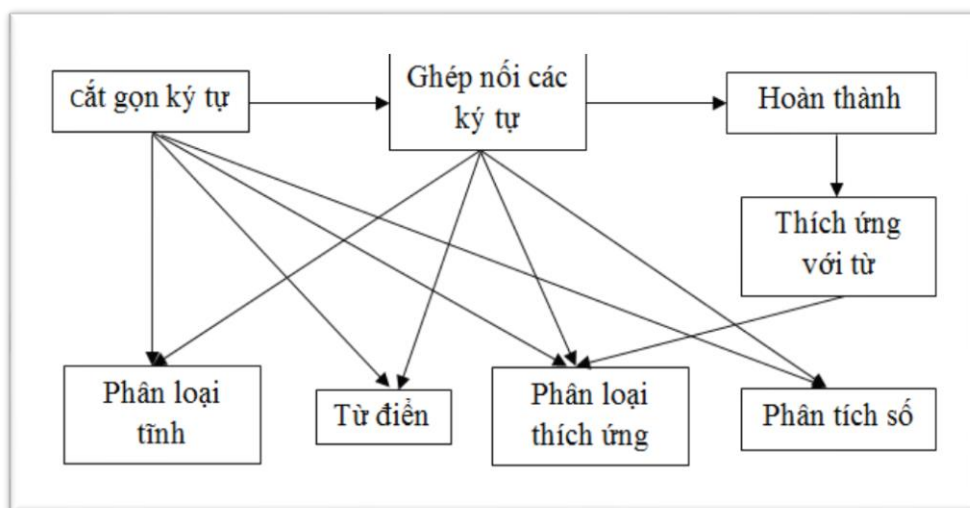
Cắt gọn từ sẽ xác định xem có các ký tự liền nhau trong một từ hay không. Nếu có nó sẽ cắt nhỏ các ký tự ra thành các ký tự riêng lẻ.



Hình 2.4 Cắt các ký tự liền nhau

Nhận dạng khoảng cách giữa chữ và số là một bài toán rắc rối. Trong văn bản có nhiều phong chữ khác nhau dẫn tới khoảng cách giữa các từ và số khác nhau. Tesseract khắc phục khó khăn trên bằng cách đo khoảng cách được chọn gần ngưỡng nào đó như là giá trị mờ với sai số.

Sơ đồ nhận dạng một từ là quy trình phân tích một từ được chia ra thành các ký tự:

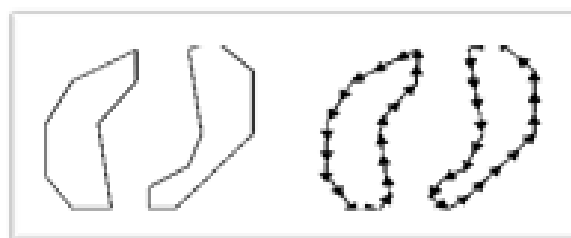


Hình 2.5 Sơ đồ nhận dạng từ

Mỗi ký tự cần nhận dạng có những đặc trưng riêng, có khoảng 50 tới 100 đặc trưng điển hình trong mỗi ký tự. Mỗi đặc trưng chứa 3 tham số là hoành độ, tung độ, và góc xoay. Trong khi đó mỗi ký tự mẫu có từ 10 tới 20 đặc trưng, mỗi đặc trưng có 4 tham số là hoành độ, tung độ, góc xoay, độ dài.

Văn bản luôn tồn tại độ dư thừa ký tự và từ vựng, vậy chức năng phân loại ký tự tạo ra danh sách rút gọn chứa các ký tự mà ký tự đối sánh có thể trùng khớp. Các lớp ký tự mẫu sinh ra các lớp véc tơ bit tương ứng với những đặc trưng của từng ký tự.

Những đặc trưng của ký tự nhận dạng (Features of character) được đối sánh với lớp véc tơ bit của ký tự mẫu, và tính toán sự khác nhau giữa các đặc trưng của chúng. Bên cạnh đó có tham số thứ hai là độ dài của ký tự nhận dạng.



Hình 2.6 Các đặc trưng ký tự được nhận dạng

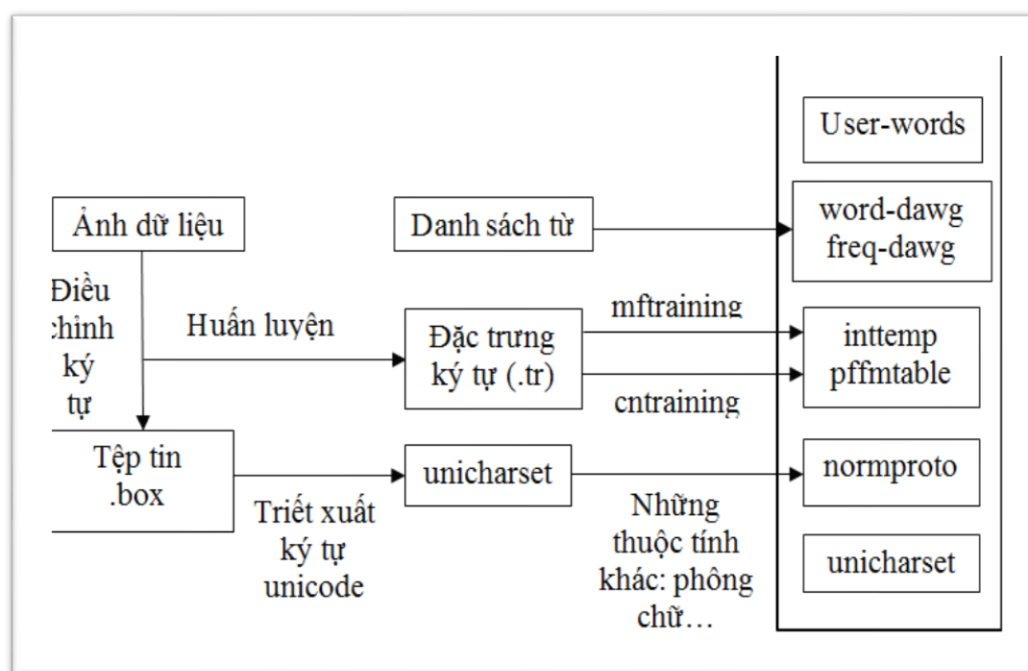
Hệ số đánh giá đối sánh là tích hai tham số trên, cặp đối sánh nào có hệ số nhỏ nhất thì xem như chúng là tương tự nhau.

Chức năng phân loại tĩnh (static classifier) phù hợp với các ký tự có phong chữ bất kỳ, nhưng nó chủ yếu được dùng để nhận dạng các ký tự riêng như ký tự chú giải, dấu ngăn cách hay kết thúc câu... trong khi chức năng phân loại thích ứng (Adaptive classifier) dùng để nhận dạng các ký tự theo phong chữ chuẩn.

Bộ từ điển dùng để lưu trữ dữ liệu cho quá trình phân loại và nhận dạng. Mỗi ngôn ngữ có một bộ từ điển chứa các ký tự theo các phong chữ khác nhau với thuộc tính như chuẩn – normal, đậm – bold, nghiêng – italic và thuộc tính kết hợp. Từ điển cũng lưu trữ các từ hay sử dụng, từ chữ cái, từ số, từ chữ hoa, từ chữ thường.

## 2.3 Huấn luyện dữ liệu nhận dạng với Tesseract

Để sử dụng Tesseract như là engine nhận dạng cho văn bản viết bằng một ngôn ngữ nào đó, ta cần đào tạo ra một bộ từ điển cho ngôn ngữ ấy. Ngày nay, gần như các bộ dữ liệu nhận dạng cho các ngôn ngữ phổ thông kể cả tiếng Việt đều đã có thể tải xuống từ trang dự án Google Code của phần mềm. Tuy nhiên, Tesseract cung cấp danh sách hàm thủ tục để những lập trình viên có thể xây dựng từ điển nhận dạng.

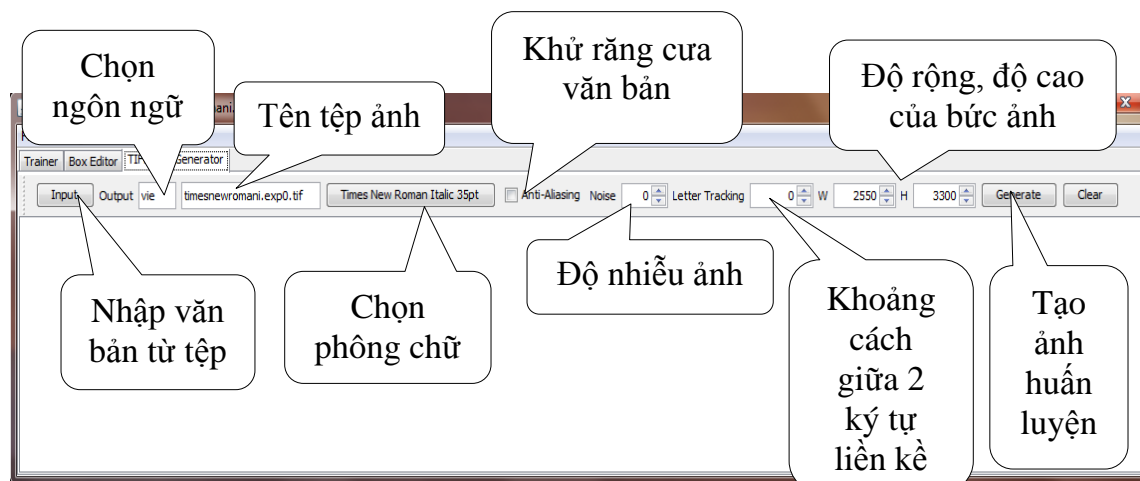


Hình 2.7 Sơ đồ đào tạo dữ liệu của Tesseract

Hơn nữa, những chuyên gia tin học có thể hoàn thiện thư viện mã nguồn mở viết bằng C/C++. Mặc dù vậy, người sử dụng không bắt buộc phải hiểu chi tiết lập trình trong các thủ tục nhưng vẫn thực hiện được công tác xây dựng từ điển vì thư viện Tesseract đã biên dịch chúng thành những chương trình thi hành được dưới dạng tệp \*.exe như mftraining, cntraining, hoặc shapeclustering... có tham số đầu vào và đầu ra. Tuy vậy, với mục đích hỗ trợ tối đa cho người dùng không là chuyên gia lập trình bằng cách cung cấp giao diện thân thiện và giảm bớt các dòng lệnh nhập trong môi trường đồ họa. Các chức năng trong sơ đồ 2.7 trên sẽ được mô tả trực quan bởi các bước theo giao diện huấn luyện với jTessBoxEditor sau đây.

### 2.3.1 Tạo dữ liệu huấn luyện

Chạy bộ biên tập để sinh ảnh văn bản, giao diện chương trình được hiển thị như sau:



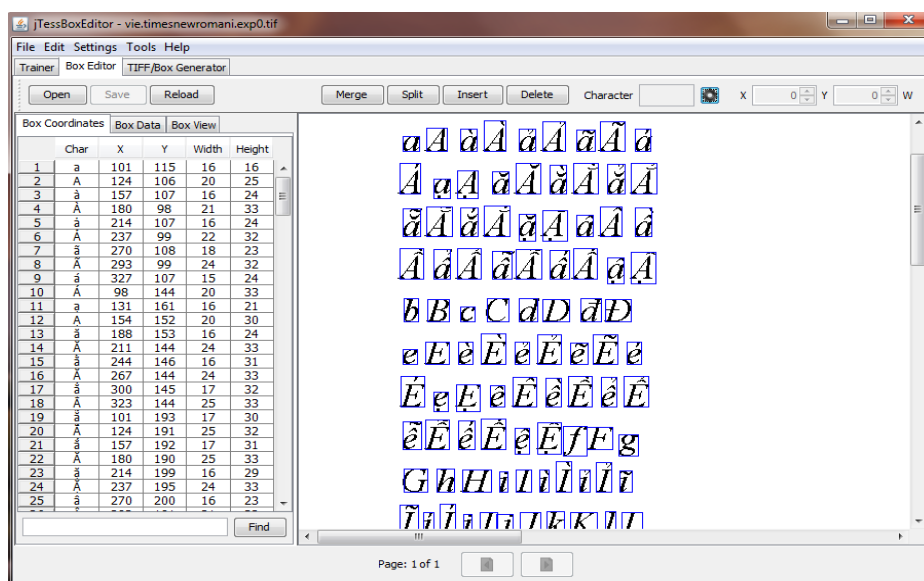
Hình 2.8 Các chức năng chính của bộ biên tập văn bản mẫu

Muốn tạo một hình ảnh huấn luyện đầu tiên ta phải có văn bản huấn luyện, ngôn ngữ đào tạo thì không có giới hạn nhưng ở đây lựa chọn tiếng Anh và tiếng Việt. Phông chữ soạn thảo cho mọi ngôn ngữ được sử dụng là phông unicode vì nếu dùng những loại phông chữ khác có thể tạo ra lỗi nhận dạng mã ký tự chuẩn UTF-8 sau đó. Cỡ chữ được Tesseract đề nghị nhỏ nhất là 15, nhưng với những phông chữ nhỏ thì chất lượng nhận dạng kém. Do đó cần chọn cỡ chữ lớn hơn để tương đương với kích cỡ chữ qua máy quét văn bản. Nếu như nhận dạng những văn bản in ấn có chất lượng tốt thì không cần để ý tới tham số letter tracking, nhưng với văn bản nhận dạng có chất lượng kém thì để tránh tình trạng dính chập các ký tự có thể phải thay đổi tham số letter tracking. Tùy chọn anti-aliasing sẽ giúp giảm tối thiểu tình trạng răng cưa trên

1. Tiến hành tạo ảnh chứa chữ in Việt cần huấn luyện:

- Ngôn ngữ: Việt
- Phong chữ: Times New Roman, chữ nghiêng, cỡ chữ 35
- Khử răng cưa: Không
- Có nhiều: Không
- Có chòong lẫn ký tự: Không
- Độ rộng: 1024 điểm ảnh
- Độ cao: 400 điểm ảnh

3. Nhấn vào nút Open để xem hình ảnh vừa tạo:



Hình 2.9 Nhận dạng phác thảo ký tự

4. Nếu thấy chưa thỏa đáng, trình biên tập cho phép nhóm (merge) hay chia (split) các ký tự, thêm (insert) hay xóa (delete) ký tự nào đó trong dữ liệu nhận dạng của ảnh. Sau khi hoàn thành việc chỉnh sửa nhấn Save để cập nhật tệp .box . Hộp biên (Bounding box) chứa tọa độ của các ký tự như hình trên, tọa độ trái trên, độ rộng, độ cao tương ứng trong ảnh văn bản, như là dữ liệu học của từ điển.
5. Từ dữ liệu trong hộp biên, bộ biên tập gọi chương trình *unicharset\_extractor* sinh ra mã unicode của các ký tự lưu trong tệp *vie.unicharset* và những thuộc tính của ký tự như chữ cái, chữ số, hoa, thường, hoặc ký tự ngăn cách câu từ... được lưu trong tệp *vie\*.tr*
6. Muốn nhận dạng văn bản bất kỳ được chính xác, Tesseract đề nghị nên sinh nhiều phong, kiểu chữ khác nhau. Số lượng các từ đa dạng nếu được huấn luyện với nhiều phong chữ khác nhau cũng cho kết quả nhận dạng cải thiện tăng thích ứng với nhiều văn bản. Tesseract không đòi hỏi tạo ra các văn bản với nhiều cỡ chữ khác nhau vì những mẫu dữ liệu nhận dạng khác kích thước với mẫu đã có thì sẽ được điều chỉnh qua phép co giãn.
7. Cuối cùng, sao chép toàn bộ các tệp dữ liệu đã tạo vào thư mục chứa đã tạo ở trên để tiến hành bước tiếp theo.

### 2.3.2 Thiết lập các tệp cấu hình huấn luyện

Cần tối thiểu 3 tệp cấu hình `vie.font_properties`, `vie.frequent_words_list` và `vie.words_list`, chúng có thể biên soạn bằng trình notepad.

#### 1. Tệp `vie.font_properties`

Tệp này khai báo những kiểu phong chữ sử dụng cho dữ liệu cần đào tạo trong các tệp `.tiff` đã tạo ra trước đó. Tệp chứa nhiều dòng và mỗi dòng chứa thông tin từng phong chữ:

	name	italic	bold	fixed	serif	fraktur
giá trị	Không chứa dấu cách trống	0: không 1: có	0: không 1: có	0: không 1: có	0: không 1: có	0

Bảng 2.1 Thuộc tính phong chữ

Phong chữ `courier new` có thuộc tính `fixed` quy ước bằng 1, như khai báo: `couriernewi 1 0 1 0 0`. Trong khi đó phong chữ `timesnewroman` luôn có thuộc tính `serif`, như khai báo: `timesnewromanb 0 1 0 1 0`.

#### 2. Tệp `vie.frequent_words_list`

Tệp này khai báo danh sách các từ thường gặp trong ngôn ngữ mà ta muốn nhận dạng, thông thường việc này đòi hỏi sự thống kê trên quy mô lớn với các tài liệu có số lượng trang nhiều. Cấu trúc của một tệp bao gồm nhiều dòng và mỗi dòng chứa một từ.

#### 3. Tệp `vie.words_list`

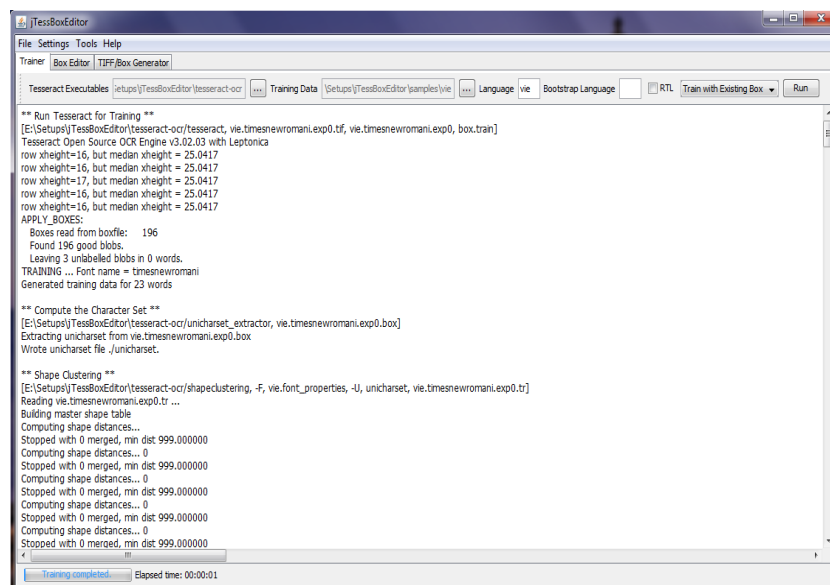
Tệp này khai báo danh sách toàn bộ các từ, nên sử dụng dữ liệu từ điển để đưa ra danh sách này. Cũng giống như `frequent_word_list`, `words_list` cũng lưu dữ liệu theo dòng, mỗi dòng có 1 từ và được mã hóa theo định dạng UTF-8.

### 2.3.3 Huấn luyện dữ liệu

Sau khi chuẩn bị xong những tệp dữ liệu trên, ta tiếp tục quy trình huấn luyện dữ liệu. Trước tiên, bộ biên tập gọi chương trình `shapeclustering`, nó tiếp nhận dữ liệu từ các tệp `*.tr` và huấn luyện cụm bằng cách tạo ra tệp `vie.shapetable`. Sau đó, chương trình `mftraining` sinh ra các tệp `vie.inttemp` và `vie.pffmtable`. Những tệp này chứa các mẫu hình dạng của các ký tự đã được đào tạo và bảng số lượng các đặc trưng kỳ vọng



của từng ký tự đó. Tương tự là chương trình *cntraining* tạo tệp *vie.normproto* chứa các mẫu ký tự chuẩn hóa.



Hình 2.10 Kết quả huấn luyện dữ liệu

Trên cơ sở những tệp dữ liệu này, bộ biên tập tiếp nhận danh sách các từ khóa trong *vie.frequent\_words\_list* và *vie.words\_list* và danh sách mã unicode để chương trình *wordlist2dawg* sinh ra tệp *vie.freq-dawg* và *vie.word-dawg*. Ngoài ra bộ biên tập cũng sinh ra một số các tệp nháp khác như: *vie.config*, *vie.punc-dawg*, *vie.bigram-dawg*...

Chương trình cuối cùng là *combine\_tessdata* sẽ kết nối các tệp *vie.\** trở thành tệp *vie.traineddata* là tệp từ điển cho nhận dạng văn bản chữ in Việt.

Mặc dù vậy, toàn bộ các chức năng trên được thực hiện tự động chỉ với một thao tác chọn nút Run trong cửa sổ ứng dụng jTessBoxEditor.



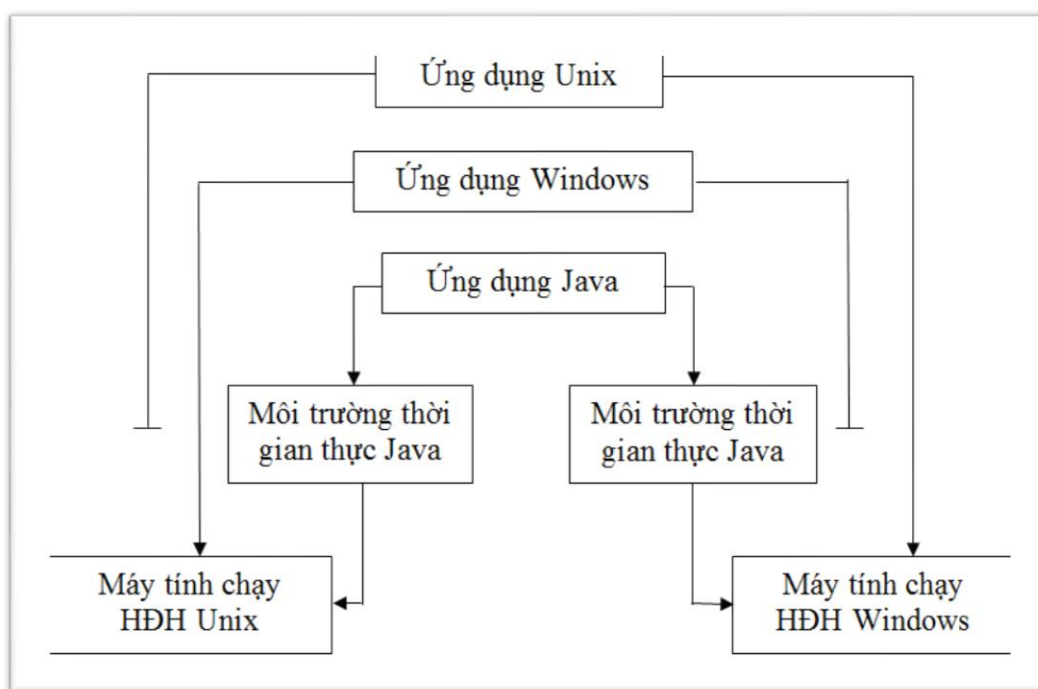
## CHƯƠNG 3 CHƯƠNG TRÌNH NHẬN DẠNG VĂN BẢN

### 3.1 Ngôn ngữ lập trình và những thư viện được sử dụng

Có nhiều ngôn ngữ lập trình viết chương trình nhận dạng văn bản, một trong số đó là Java. Java có cú pháp tương đối giống với C# và có khả năng thừa kế thư viện lập trình của C/C++. Mặt khác, Java phù hợp trong môi trường mạng truyền thông cũng như viễn thông, do đó có thể sử dụng các thư viện chuẩn của Java để lập trình cho mạng Internet và thiết bị di động. Sau đây ta sẽ giới thiệu tóm tắt về quá trình hình thành và phát triển của ngôn ngữ.

#### 3.1.1 Ngôn ngữ lập trình

Java được đề xuất bởi nhóm tác giả tại Sun Microsystems, Inc vào năm 1991. Phải mất 18 tháng để nhóm tác giả phát triển xong phiên bản đầu tiên, ban đầu ngôn ngữ được đặt tên là cây sồi nhưng sau đó đổi thành tên gọi như ngày nay. Trước thời điểm giới thiệu Java vào năm 1995, mục tiêu của nhóm tác giả khi đó chỉ là xây dựng ngôn ngữ độc lập với nền tảng để nhúng được trong các thiết bị điện tử của người tiêu dùng. Ngôn ngữ C/C++ thiết kế chỉ cho biên dịch trên một nền tảng cụ thể và nếu như phải thích ứng với nhiều bộ xử lý và hệ điều hành khác nhau thì chúng khá đắt và tốn thời gian.

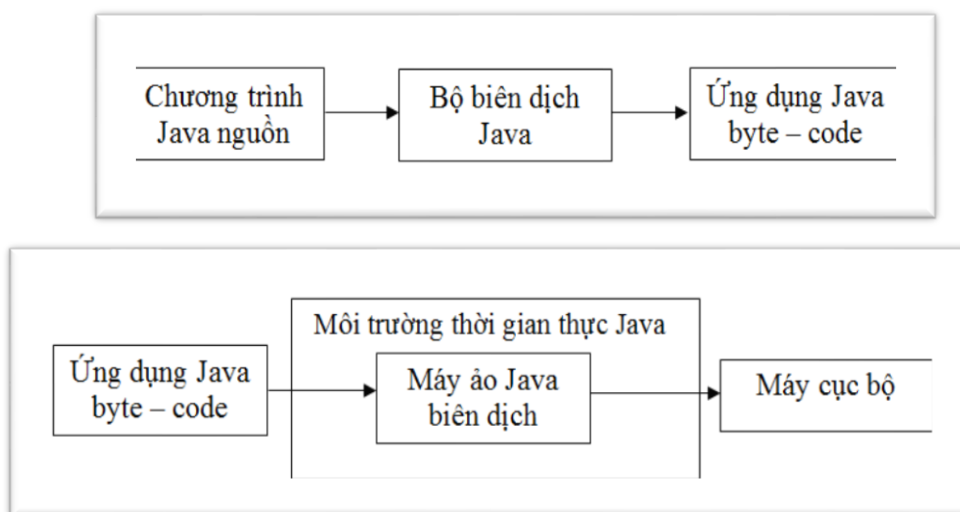


Hình 2.11 Ứng dụng Java chạy trên nhiều hệ điều hành

Giải pháp trong trường hợp này là thiết kế ngôn ngữ mới độc lập với thiết bị phần cứng cũng như hệ điều hành nhưng có chi phí tiết kiệm hơn, đó là Java. Cũng trong thời gian này, còn có một nhân tố khác vai trò quan trọng tới tương lai sau này của Java là sự hình thành của Internet và world wide web.

Java thừa kế nhiều đặc trưng của C/C++ nhưng Java không là một ngôn ngữ C/C++ cho Internet. Trái lại, nó có tính thực tiễn và triết lý riêng bên cạnh những ưu điểm của cú pháp ngôn ngữ như sự nhất quán, logic, mạch lạc. Hơn thế nữa, bên cạnh những ràng buộc khi viết mã lệnh cho môi trường internet thì ngôn ngữ hỗ trợ khả năng tạo giao diện ứng dụng thông qua bộ điều khiển để tối ưu mã lệnh. Java cũng không là ngôn ngữ tồn tại tách biệt, vì nó cho phép tương tác với các thư viện của ngôn ngữ C/C++, kết quả đó càng tăng thêm sức mạnh cho ngôn ngữ. Tổng kết, Java có những ưu điểm như khả năng hướng đối tượng, bảo mật, phù hợp môi trường di động, kiến trúc nơ – ron, có năng lực thi hành cao, và tính phân phối.

Trong kiến trúc nền tảng, Java chứa hai phần mềm là môi trường thời gian thực java (JRE) và công cụ phát triển java (JDK). JRE phục vụ để chạy các ứng dụng java và applets trong khi đó JDK cho phép phát triển các ứng dụng của Java. Một thành phần quan trọng của JRE là máy ảo java (JVM), nó hoạt động như bộ xử lý ảo, cho phép các ứng dụng java chạy trên hệ thống cục bộ. Các ứng dụng Java được sắp đặt theo định dạng tập lệnh tối ưu hóa cao gọi là byte – code để không phụ thuộc kiến trúc phần cứng, hệ điều hành... mà chúng được cài đặt trên đó. Ngày nay, đa số các ngôn ngữ hỗ trợ biên dịch nhưng Java duy trì bộ thông dịch nhằm thích ứng môi trường mạng.



Hình 2.12 Cơ chế thông dịch Java

Kiến trúc Java trước đây thực hiện thông dịch chương trình byte – code thành native – code nhưng từ phiên bản 2 trở đi, máy ảo Java tích hợp bộ biên dịch Just In Time, nhưng chỉ biên dịch ứng dụng byte – code thành ứng dụng native – code khi cần nhằm tối ưu khả năng thực hiện. Các tệp chương trình nguồn sẽ phải được dịch thành ứng dụng byte – code trước khi chạy, và bộ biên dịch cài đặt luôn có trong JDK.

Đối với các lập trình viên, hệ thống cần có JRE hợp lệ và công cụ phát triển ứng dụng với ngôn ngữ Java. Trên cơ sở nền tảng Java, một thành phần ứng dụng cho phép lập trình viên thiết kế chương trình mềm dẻo nhanh chóng, đó là Java Beans. Khi làm việc với Java Beans, phần lớn các nhà phát triển sử dụng công cụ xây dựng ứng dụng cho phép cấu hình tập Beans, kết nối chúng, và tạo thành chương trình. Sau đây là những ưu điểm của Java Beans:

- Khả năng viết một lần, chạy mọi nơi.
- Công cụ xây dựng ứng dụng hỗ trợ điều khiển được các thuộc tính, sự kiện, phương thức, giúp lập trình viên giảm bớt quá trình viết mã.
- Phần mềm phụ trợ có thể được cung cấp để giúp cấu hình một Bean. Nó cần thiết nếu thành phần có các tham số “design-time” được thiết lập. Chẳng hạn như bộ biên tập hay bộ tùy biến có thể tạo ra một môi trường riêng cho một Bean.
- Suu tập của các Bean được cấu hình và liên thông với nhau, để có thể lưu trữ nhất quán và có thể khôi phục sau này cho trạng thái của ứng dụng...

Có hai công cụ để xây dựng ứng dụng Bean, loại thứ nhất là BeanBox, loại sau này là Bean Builder. Những ứng dụng Bean phiên bản mới hay sử dụng công cụ Bean Builder.

### ***3.1.2 Những thư viện được sử dụng***

Bên cạnh các thư viện chuẩn của ngôn ngữ, chương trình nhận dạng văn bản số sử dụng một số thư viện khác. JNA là một thư viện mã nguồn mở do cộng đồng phát triển cung cấp khả năng truy cập dễ dàng vào các thư viện chia sẻ, được viết bằng ngôn ngữ C/C++ mà không cần sử dụng các giao diện Java bản địa. Thiết kế của JNA nhằm cung cấp khả năng truy cập bản địa một cách tự nhiên và không yêu cầu tạo ra các đoạn mã kết nối.

Thư viện JNA sử dụng một thư viện giao diện chức năng ngoài để tự động gọi mã nguồn cơ sở. Giống như cách truy nhập thư viện liên kết động, chức năng này cho phép mã để tải một thư viện theo tên và lấy một con trỏ đến một hàm trong thư viện đó và sử dụng thư viện để gọi nó mà không cần tập tin tiêu đề hoặc bất kỳ giai đoạn biên dịch nào. Các nhà phát triển sử dụng một giao diện Java để mô tả chức năng và cấu trúc trong thư viện mục tiêu. Điều này làm cho nó khá dễ dàng để tận dụng lợi thế của tính năng trên nền tảng bản địa mà không cần phát sinh chi phí trong việc cấu hình và xây dựng mã JNI.

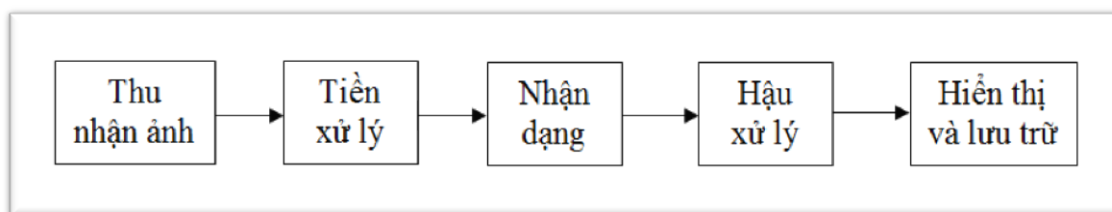
JNA được xây dựng và thử nghiệm trên nhiều hệ điều hành khác nhau như Mac, Windows, Windows Mobile, Android... Nó cũng có thể được tinh chỉnh và biên dịch lại các cấu hình để làm cho nó làm việc trên hầu hết các nền tảng chạy Java.

Một thư viện khác là Tess4J, chứa các hàm đã lập trình của Tesseract API. Hiện tại, nó đang được phát triển và thử nghiệm với Java 32-bit trên Windows và Linux. Chức năng của Tess4J cung cấp các hàm cho phép nhận dạng ký tự quang học từ những tệp ảnh số có định dạng TIFF, JPEG, GIF, PNG và BMP, và tài liệu PDF.

Cuối cùng là Jortho, đây là một thư viện mã nguồn mở kiểm tra chính tả tiếng Anh viết bằng Java. Từ điển của nó được dựa trên dự án từ điển Wikidictionary miễn phí và do đó có thể được cập nhật cho hầu như bất kỳ ngôn ngữ nào. Thư viện làm việc với bất kỳ JTextComponent từ Swing framework; bao gồm JTextPane, JEditorPane và JTextArea. Các khả năng của thư viện bao gồm:

- Đánh dấu từ có khả năng sai chính tả.
- Cung cấp một menu ngữ cảnh với các đề xuất cho một dạng thức đúng của từ.
- Menu ngữ cảnh với các tùy chọn để thay đổi ngôn ngữ kiểm tra.
- Hộp thoại để tiếp tục kiểm tra chính tả văn bản.
- Người dùng tự định nghĩa từ điển các từ (chức năng này là một lựa chọn xuất hiện trong hộp thoại kiểm tra liên tục)
- Dịch kiểm tra chính tả giao diện người dùng sẽ thấy các nút và nhãn trong ngôn ngữ bản địa.
- Hàm API độc lập, không cần có máy chủ nhưng applet là có thể.
- Phát hiện sai sót.
- Bật hoặc tắt mỗi tính năng riêng biệt.

### 3.2 Chức năng chương trình



Hình 2.13 Chức năng chính trong chương trình

#### 3.2.1 Thu nhận ảnh

Thông thường, ảnh văn bản được thu nhận dưới rất nhiều dạng bao gồm ảnh quét/chụp bằng các công cụ quang học (máy ảnh, máy quét,...), ngoài ra còn có ảnh được tạo ra bởi các ứng dụng số như các trình biên tập hình ảnh (Photoshop, Corel Draw,...) hay ảnh chụp màn hình. Vì vậy chất lượng định dạng ảnh đầu vào sẽ rất khác nhau từ tập tin PDF đến các định dạng ảnh thông dụng khác như JPG, PNG, BMP,... đòi hỏi chúng phải cùng được đưa về một định dạng chung nhất để tiện cho việc xử lý.

#### 3.2.2 Tiền xử lý

Ảnh đầu vào như là ảnh chụp từ camera thì cần được tiền xử lý như chuyển đổi ảnh đen trắng. Nếu ảnh từ máy quét có thể chứa độ nghiêng thì phải được khử nghiêng. Mặt khác, ngôn ngữ Java cung cấp thư viện xử lý ảnh hỗ trợ nhiều lớp lọc ảnh như tăng độ tương phản, trơn ảnh... cải thiện chất lượng ảnh.

#### 3.2.3 Nhận dạng

Tesseract phiên bản hiện nay hỗ trợ chế độ nhận dạng văn bản ảnh số theo chế độ phân đoạn trang (PSM):

- PSM\_AUTO\_ONLY: phân đoạn trang tự động.
- PSM\_AUTO: phân đoạn trang tự động đầy đủ, hỗ trợ nhận dạng toàn văn bản.
- PSM\_SINGLE\_COLUMN: nhận dạng văn bản có một cột với những kích cỡ phông chữ khác nhau.
- PSM\_SINGLE\_BLOCK: nhận dạng hình ảnh chứa một khối văn bản chuẩn.
- PSM\_SINGLE\_LINE: nhận dạng hình ảnh văn bản nếu chỉ chứa một dòng.
- PSM\_SINGLE\_WORD: nhận dạng hình ảnh văn bản nếu chỉ chứa một từ.

Tuy vậy, chỉ cần sử dụng một số chế độ như PSM\_AUTO, PSM\_SINGLE\_BLOCK, PSM\_SINGLE\_LINE, PSM\_SINGLE\_WORD. Trong mọi trường hợp, nên để thông số này ở chế độ PSM\_AUTO mặc định.

### 3.2.4 Hậu xử lý

Chức năng này nhằm xử lý các lỗi hay gặp, giả thiết có đoạn văn bản được nhận dạng như sau: “mOng là sẽ khÔng sao”. Ở đây ta nhận thấy các ký tự ‘O’ ở từ “mong” và ký tự ‘Ô’ ở từ “không” bị nhận dạng sai. Nhận thấy ký tự bị nhầm lẫn là ký tự in hoa kẹp giữa 2 ký tự viết thường hoặc nằm sau ký tự đầu của 1 từ.

Tương tự như thế, giả thiết có đoạn văn bản được nhận dạng như sau: “HoA”. Ở đây ta nhận thấy ký tự ‘A’ ở từ hoa bị nhận dạng sai. Xâu ký tự chỉ có ký tự đầu viết hoa trong khi các ký tự khác viết thường, do đó ta tiến hành thay thế đơn giản bằng cách đổi các ký tự thường của mẫu tìm được.

Trong một vài trường hợp, Tesseract thường dễ bị nhầm lẫn giữa các ký tự có cách viết giống nhau, do đó số ‘11’ thay bởi cụm từ ‘ll’, xâu ký tự “rr” thay bởi ký tự ‘n’, ký tự ‘l’ thay bằng số ‘1’, số ‘0’ thay bằng ký tự ‘o’... điều này đã được phân nào khắc phục bởi tập tin vie.DangAmbigs.txt của người dùng huấn luyện:

Khai báo ví dụ:      11 = ll  
                              rr = n  
                              l = 1

Chương trình tự động tìm và thay thế các cụm ký tự trong văn bản nhận dạng với các mẫu từ sửa chữa.

### 3.2.5 Hiện thị và lưu trữ

Văn bản được nhận dạng hiển thị trong cửa sổ chương trình, dữ liệu có thể được lưu dưới dạng .txt dưới theo chuẩn mã utf-8, sau đó có thể biên tập trong bất cứ trình soạn thảo nào, như Microsoft Word.

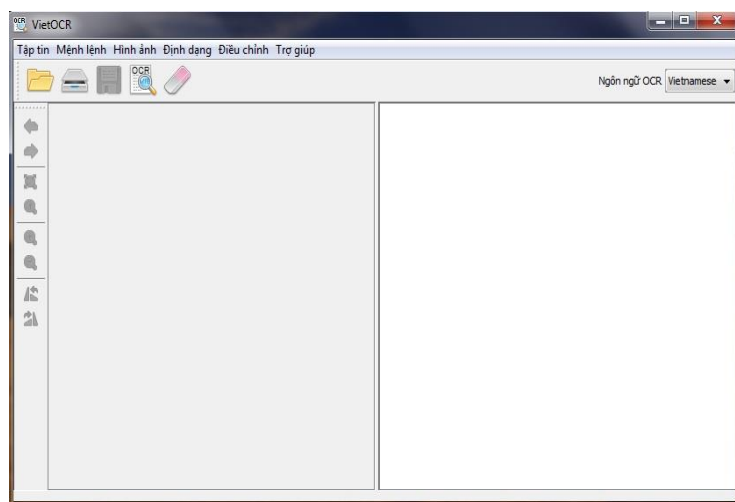
## 3.3 Giao diện chương trình

Sau khi ứng dụng khởi động xong, một giao diện bắt đầu sẽ hiển thị để người dùng tạo ra một tác vụ nhận dạng văn bản mới.

Giao diện này chứa menu ngang và dọc để truy cập nhanh các tính năng của chương trình và 3 khung làm việc chính:

- Khung bên trái hiển thị hình ảnh.
- Khung bên phải chứa văn bản được nhận dạng.
- Một khung panel phía dưới hiển thị thông tin làm việc của chương trình.

Một số kết quả thử nghiệm nhận dạng ảnh văn bản chữ in Việt, Anh:



Hình 3.1 Giao diện chương trình chính

Ảnh số	<p>Many problems can be solved by considering them as special cases of general problems. For instance, consider the problem of locating the largest integer in the sequence 101, 12,144,212,98. This is a specific case of the problem of locating the largest integer in a sequence of integers. To solve this general problem we must give an algorithm, which specifies a sequence of steps used to solve this general problem.</p> <p>We will study algorithms for solving many different types of problems in this book. For example, in this chapter we will introduce algorithms for two of the most important problems in computer science, searching for an element in a list and sorting a list so its elements are in some prescribed order, such as increasing, decreasing, or alphabetic. Later in the book we will develop algorithms that find the greatest common divisor of two integers, that generate all the orderings of a finite set, that find the shortest path between nodes in a network, and for solving many other problems.</p> <p>We will also introduce the notion of an algorithmic paradigm, which provides a general method for designing algorithms. In particular we will discuss brute-force algorithms, which find solutions using a straight forward approach without introducing any cleverness. We will also discuss greedy algorithms, a class of algorithms used to solve optimization problems. Proofs are important in the study of algorithms. In this chapter we illustrate this by proving that a particular greedy algorithm always finds an optimal solution.</p>
Kết quả nhận dạng	<p>We will study algorithms for solving many different types of problems in this book. For example, in this chapter we will introduce algorithms for two of the most important problems in computer science, searching for an element in a list and sorting a list so its elements are in some prescribed order, such as increasing, decreasing, or alphabetic. Later in the book we will develop algorithms that find the greatest common divisor of two integers, that generate all the orderings of a finite set, that find the shortest path between nodes in a network, and for solving many other problems.</p>

Bảng 3.1 Nhận dạng một vùng văn bản



Ảnh số	Các mục tiêu của an toàn bảo mật thông tin không thể đạt được nếu chỉ đơn thuần dựa vào các thuật toán toán học và các giao thức, mà để đạt được điều này đòi hỏi cần có các kỹ thuật mang tính thủ tục và sự tôn trọng các điều luật. Chẳng hạn sự bí mật của các bức thư tay là do sự phân phát các lá thư đã có đóng dấu bởi một dịch vụ thư tín đã được chấp nhận.
Kết quả nhận dạng	Các mục tiêu của an toàn bảo mật thông tin không thể đạt được nếu chỉ đơn thuần dựa vào các thuật toán toán học và các giao thức, mà để đạt được điều này đòi hỏi cần có các kỹ thuật mang tính thủ tục và sự tôn trọng các điều luật. Chẳng hạn sự bí mật của các bức thư tay là do sự phân phát các lá thư đã có đóng dấu bởi một dịch vụ thư tín đã được chấp nhận.

Bảng 3.2 Nhận dạng ảnh văn bản có góc nghiêng 10°:

Ảnh số	<b>Trao đổi khóa gồm có thỏa thuận khóa và phân phối khóa; TA là nơi thực hiện việc phân phối, cũng là nơi quản lý khóa.</b> Thỏa thuận khóa nói chung không cần có sự tham gia của một TA nào và chỉ có thể xảy ra khi các hệ bảo mật mà ta sử dụng là hệ có khóa công khai, còn phân phối khóa thì có thể xảy ra đối với các trường hợp sử dụng các hệ khóa đối xứng cũng như các hệ có khóa công khai. <b><i>Việc phân phối khóa với vai trò quản trị khóa của một TA là một việc bình thường, đã tồn tại rất lâu trước khi có các hệ mật mã khóa công khai.</i></b> Ta sẽ bắt đầu với một vài hệ phân phối khóa như vậy, sau đó sẽ giới thiệu một số hệ phân phối hoặc trao đổi khóa khi dùng các sơ đồ an toàn và bảo mật với khóa công khai.
Kết quả nhận dạng	Trao đổi khóa gồm có thỏa thuận khóa và phân phối khóa; TA là nơi thực hiện việc phân phối, cũng là nơi quản lý khóa. Thỏa thuận khóa nói chung không cần có sự tham gia của một TA nào và chỉ có thể xảy ra khi các hệ bảo mật mà ta sử dụng là hệ có khóa công khai, còn phân phối khóa thì có thể xảy ra đối với các trường hợp sử dụng các hệ khóa đối xứng cũng như các hệ có khóa công khai. Việc phân phối khóa với vai trò quản trị khóa của một TA là một việc bình thường, đã tồn tại rất lâu trước khi có các hệ mật mã khóa công khai. Ta sẽ bắt đầu với một vài hệ phân phối khóa như vậy, sau đó sẽ giới thiệu một số hệ phân phối hoặc trao đổi khóa khi dùng các sơ đồ an toàn và bảo mật với khóa công khai.

Bảng 3.3 Nhận dạng ảnh văn bản với phông và cỡ chữ khác nhau



Ảnh số	Kết quả nhận dạng
<p>Logic is the basis of all mathematical reasoning, and of all automated reasoning. It has practical applications to the design of computing machines, to the specification of systems, to artificial intelligence, to computer programming, to programming languages, and to other areas of computer science, as well as to many other fields of study.</p>	<p>Logic is the basis of all mathematical reasoning, and of all automated reasoning. It has practical applications to the design of computing machines, to the specification of systems, to artificial intelligence, to computer programming, to programming languages, and to other areas of computer science, as well as to many other fields of study.</p>

Bảng 3.4 Nhận dạng ảnh văn bản có các dòng cong

## KẾT LUẬN

### I. Đánh giá kết quả

Sau khi thử nghiệm thư viện Tesseract để nhận dạng tập ảnh số văn bản chữ in tiếng Anh và tiếng Việt, có những nhận xét sau đây:

Ưu điểm của chương trình:

- Bộ từ điển tiếng Anh có khả năng nhận dạng có độ chính xác cao đối với văn bản chữ in tiếng Anh.
- Cho phép dữ liệu nhận dạng được huấn luyện, do đó có thể tạo ra từ điển riêng với có phong chữ phù hợp với văn bản nhận dạng cụ thể, trường hợp này thì kết quả nhận dạng tương đối tốt.
- Nếu tạo ra bộ từ điển tiếng Việt nhận dạng các loại văn bản chữ in tiếng Việt theo một số loại phong chữ phổ biến như Times New Roman, Arial, Microsoft Sans Serif, Tahoma được quét từ máy quét thì hiệu quả cũng tạm được.
- Nhận dạng được ảnh văn bản chứa góc nghiêng hoặc dòng có độ cong nhỏ.

Nhược điểm của chương trình:

- Kết quả nhận dạng văn bản theo định dạng phong chữ Việt còn mắc nhiều lỗi, chủ yếu là lỗi dấu... Hơn nữa, chương trình chưa nhận dạng chính xác văn bản chữ in tiếng Việt có cỡ chữ nhỏ hơn 15 hoặc ảnh văn bản chất lượng thấp.
- Còn có lỗi xử lý nhận dạng dữ liệu hoặc chỉ nhận dạng được với khối dữ liệu nhỏ từ tệp PDF.
- Tỷ lệ lỗi nhận dạng phụ thuộc kích thước văn bản.

### II. Hướng phát triển của đề tài

- Tăng cường khả năng xử lý đầu ra cho phép nhận dạng văn bản trộn bảng biểu và hình ảnh.
- Một trong những hạn chế của kết quả nhận dạng như lỗi từ là do số lượng từ học của văn bản còn ít. Nguyên nhân trên có thể được cải thiện nếu tốc độ và bộ nhớ máy tính được nâng cấp sau này. Hiện nay, từ điển tiếng Việt mới thử nghiệm được khoảng 30 loại chữ khác nhau, nếu cập nhật văn bản mới để đào tạo thì sẽ nâng cao được chất lượng nhận dạng chữ in tiếng Việt.
- Nhận dạng văn bản từ tệp PDF.

## TÀI LIỆU THAM KHẢO

- [1]. Rafael C.Gonzalez, “Digital Image Processing”, Prentice-Hall, 2005
- [2]. Tesseract OCR <http://code.google.com/p/tesseract-ocr/>
- [3]. Ray Smith, “An Overview of the Tesseract OCR Engine”, IEEE Computer Society, 2007
- [4]. Chirag Patel , Atul Patel , Dharmendra Patel , “Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study”, International Journal of Computer Applications, 2012
- [5]. Java Platform <http://en.wikipedia.org>
- [6]. Herbert Schildt, "The Complete Reference Java 2", Mc – Graw Hill, 2002
- [7]. Nhận dạng ký tự quang học <http://en.wikipedia.org>
- [8]. Lương Xuân Mạnh, “Luận văn tốt nghiệp Đại Học”, ĐHHH, 2014
- [9]. Nguyễn Hữu Tuấn, “Giáo trình An toàn bảo mật thông tin”, NXB ĐHHH, 2008
- [10]. Kenneth H.Rosen, “Discrete Mathematics and Its Applications”, Mc-Graw Hill, 2012