

An overview of Tesseract OCR Engine

Nguyen An Binh

Cấu trúc

Bước đầu tiên là Adaptive Thresholding, ở bước này, ảnh input sẽ được chuyển về dạng binary sử dụng phương pháp của Otsu.

Bước tiếp theo là **Page layout analysis**, bước này để nhận biết các text blocks trong ảnh input.

Tiếp theo, các **baselines** của các dòng sẽ được detect, sau đó text sẽ được chia ra thành nhiều từ dựa vào definite spaces và fuzzy spaces.

Tiếp theo các character outline sẽ được detect từ các từ.

Bước cuối là nhận dạng ký tự, bước này sẽ được chia làm 2 lần chạy, lần thứ nhất dùng static classifier, *mỗi từ đạt yêu cầu* sẽ được chuyển đến bộ phân lớp thứ 2, adaptive classifier làm training data. Lần chạy thứ 2 sẽ dùng adaptive classifier để chạy lại toàn bộ trang, để detect lại những từ mà lúc trước dự đoán chưa chính xác.

Page Layout Analysis

Sau khi chuyển ảnh về dạng binary ở bước đầu tiên, chúng ta sẽ dùng bước thứ 2 này để phân chia những phần chứa text và không chứa text trong văn bản.

1. Đầu tiên sẽ sử dụng thư viện cho image processing tên là Leptonica để detect ra được các vertical lines và các images, xóa các đối tượng này đi khỏi văn bản trước khi move sang bước kế tiếp là connected component analysis.
2. Các tab-stops nằm ở ngoài cùng của các vùng chứa text sẽ được group lại với nhau tạo thành các tap-stop lines.
3. Các connected components sẽ được chạy từ trái sang phải, từ trên xuống dưới để tập hợp và tạo thành các Column Partition.
4. Mỗi chuỗi Column Partition (CP) tiếp đó sẽ được chia nhỏ ra thành các dòng, và mỗi CP này sẽ được xem là một text block

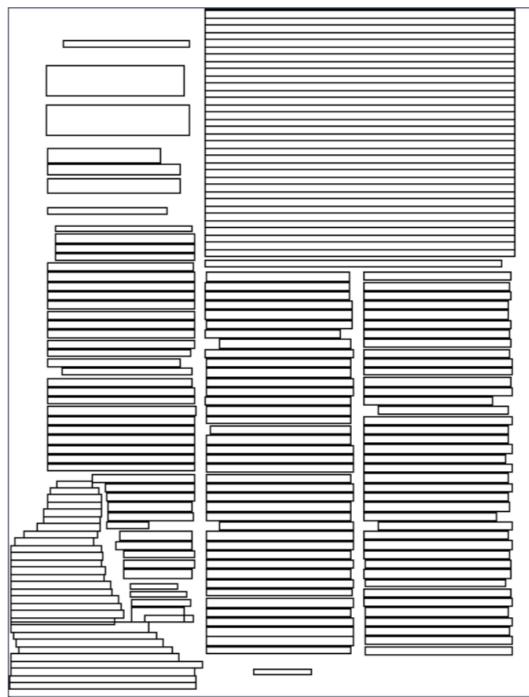


Figure 1 Column Partitions

Baseline Fitting and Word Detection

Thuật toán dùng để tìm lines của Tesseract không cần phải de-skew lại văn bản. Các bước để detect line:

1. Connected components analysis, đây là bước detect ra các blobs (bounding box) của các từ trong văn bản.
2. Xóa những blobs có kích thước nhỏ hơn mức trung bình đi. Vì những blobs này thường là dấu chấm câu hoặc nhiễu.
3. Những blobs trong trang sẽ được sắp xếp theo thứ tự tăng dần với key sort là tọa độ x bên cạnh trái.
4. Tạo dòng dựa trên blobs này, từ những dòng này chúng ta cũng sẽ biết được skew của trang.
5. Sau khi lines đã được tạo, các baselines sẽ được hình thành dựa vào least squares fit.

Word recognition

Mục đích của bước này là tách các từ ra thành ký tự (thuật toán không thể biết được đâu là từ, nó chỉ biết đến các blobs, các khoảng trắng, ...)

Tesseract kiểm tra xem các dòng text có fixed pitch không (khoảng cách giữa các từ trong text line và các ký tự trong từ cách đều nhau), nếu có thì các từ sẽ được cắt ra thành các ký tự và để dùng cho quá trình word recognition.

Nếu không có fixed pitch thì Tesseract dựa vào khoảng cách từ baseline tới meanline làm ngưỡng để tách từ, nhưng quyết định tách ký tự cuối cùng vẫn phải dựa vào quá trình word recognition.

Sau khi các fix pitched và tách từ theo tỷ lệ baseline meanline, các blob được tách ra sẽ được mang đi so khớp với bộ dictionary (linguistic analysis). Nếu kết quả vẫn tệ thì các blobs sẽ tiếp tục được cắt ra bằng cách xem các blob như là 1 đa giác, và điểm cắt là các đỉnh lõm của đa giác mà đối diện với nó cũng là một đỉnh lõm.

Sau khi thực hiện cắt ra kết các khả năng như trên, mà kết quả so khớp vẫn tệ, các mảnh được cắt ra sẽ được mang vào một associator. Associator sẽ có nhiệm vụ phối hợp và tạo các combinations của các blobs lại xem nó thuộc từ nào trong bộ dictionary không. Những combination của các blobs mà khớp (được phân loại) dựa vào dictionary sẽ được lưu giữ lại trong dictionary.

Character Classifier

A. Static classifier

Trong mô hình classification này, features chính là các outlines của các đa giác được cắt ra. Trong quá trình training, features này là các vector 4 chiều (x, y, direction, length), được lấy từ các element của các đa giác, và tạo thành các features vector.

Trong quá trình nhận dạng (recognition), các elements được phâ vỡ ra thành các mảnh nhỏ hơn hoặc bằng với các features, do đó, chiều thứ 4 (length) bị xóa bỏ. Quá trình nhận dạng sẽ là matching các elements nhỏ hơn hoặc bằng với các element lớn hơn, do đó rất dễ dàng nhận dạng được các chữ hoặc ký tự bị vỡ. Vấn đề duy nhất của mô hình này là chi phí tính toán khoảng cách matching giữa các unknow (broken pieces) với các features lớn hơn.

Để giảm chi phí tính toán, Tesseract bước đầu tách tập character ra thành 10 lớp sử dụng **Locality Sensitive Hashing**. Ở bước thứ 2, classifier sẽ tính khoảng cách từ các unknow pieces tới các features trong prototype gần nhất. Cuối cùng sẽ dùng KNN cho classification.

B. Adaptive Classification

Sử dụng output của static classifier ở bước trên để train cho quá trình adaptive classification này, ở bước trước, static classifier tạo ra các unknow pieces của tất cả các ký tự khác nhau, các loại font khác nhau, do đó khả năng phân biệt giữa các loại font hoặc các characters là yếu, đó là lý do Tesseract dùng adaptive classification để improve. Adaptive classification học sau khi run lần đầu với từ output của static classifier. Sau đó chúng ta chạy lại 1 lần nữa toàn bộ văn bản với adaptive classification để các từ, ký tự chưa được nhận dạng tốt ở lần đầu sẽ có kết quả nhận dạng tốt hơn ở lần thứ 2.

Kết quả từ tesseract trên handwritten và printed text image.

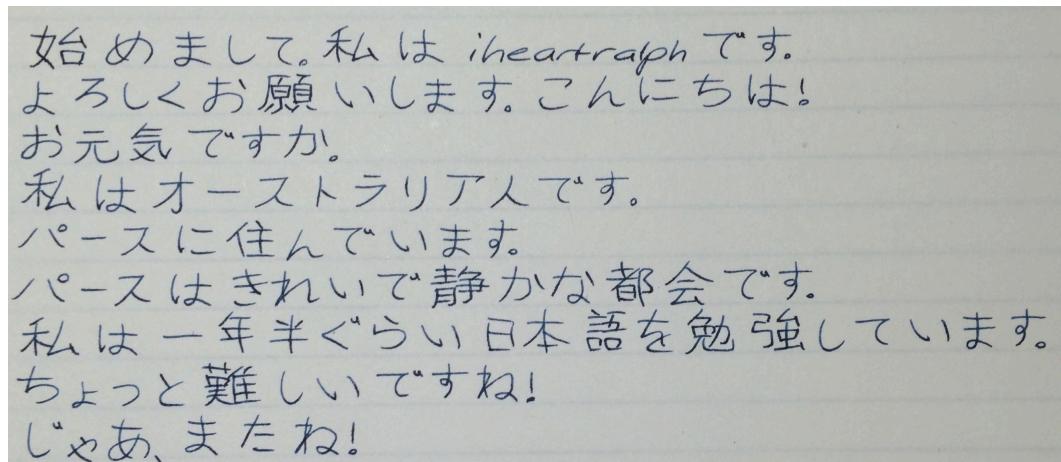


Figure 2 japanese handwritten example

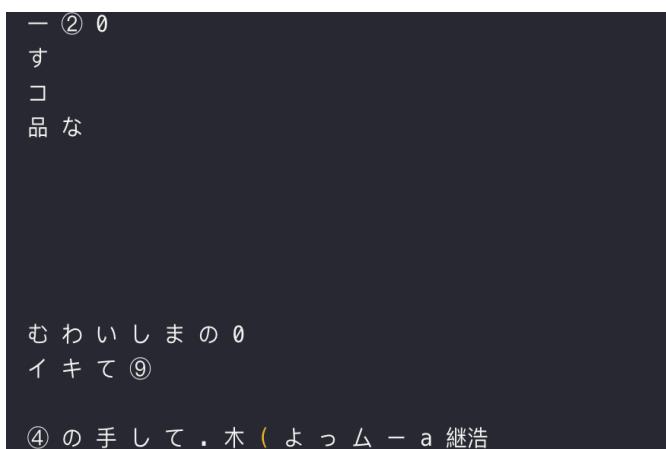


Figure 3 without dilation and erosion to remove some noise

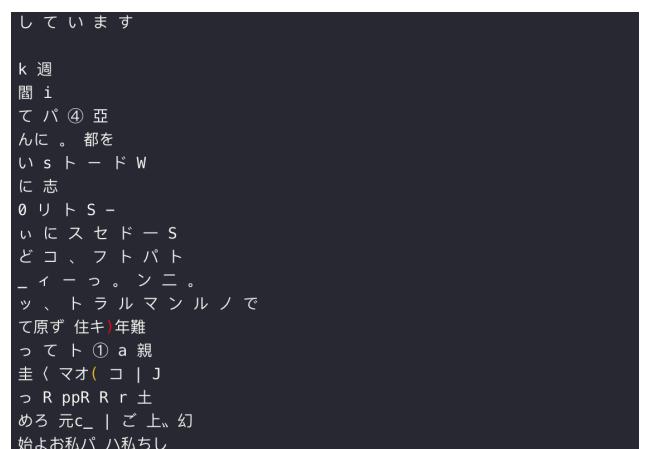


Figure 4 dilation and erosion, binarization and blur

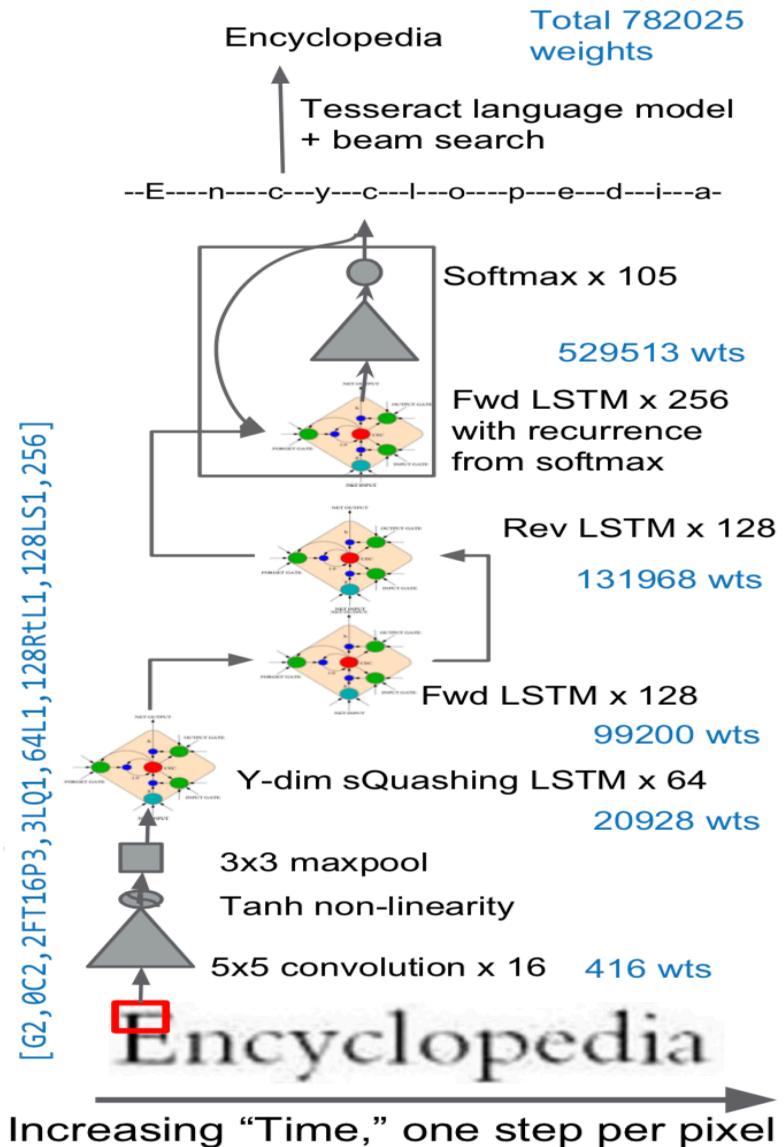
日本語能力試験

Figure 6 printed text



Figure 5 default tesseract

LSTM



Mỗi text line image được sử dụng làm sequence input cho LSTM, mỗi input image w x h trong text line image trước hết được đưa vào lớp CNN, sau đó output lớp CNN được feed vào làm input của bi-directional LSTM. Apply softmax layer cuối cùng để ra chuỗi characters, kết hợp ***Tesseract language model and beam search*** để ra final output.

Mỗi horizontal pixel position trong input text line image được dự đoán với vector probability distribution cho các ký tự (các lớp). Như vậy, output của LSTM sẽ là chuỗi ký tự, tuy nhiên vì dự đoán trên mỗi horizontal pixel cho nên output sẽ rất nhiều blank “-“ (như trên hình), cuối cùng final decoding result sẽ được rút ra bằng cách dựa vào các ký tự lân cận và xóa các blank đi.

Ref:

1. Calamari – AHigh-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition
2. Adapting the Tesseract Open Source OCR Engine for Multilingual OCR.
3. An Overview of the Tesseract OCR Engine
4. Hybrid Page Layout Analysis via Tab-Stop Detection
5. A Simple and Efficient Skew Detection Algorithm via Text Row Algorithm
6. <https://github.com/tesseract-ocr/tesseract/wiki/VGSLSpecs>
7. https://github.com/tesseract-ocr/docs/tree/master/das_tutorial2016
8. <https://github.com/tesseract-ocr/tesseract/wiki/4.0-with-LSTM>
- 9.