Statistical-based Approach to Word Segmentation

Yalin Wang[†] Ihsin T. Phillips[‡] Robert Haralick[†]

[†] Department of Electrical Engineering
University of Washington Seattle, WA 98195 U.S.A.

[‡] Department of Computer Science/Software Engineering
Seattle University Seattle, WA 98122 U.S.A.

{ylwang, yun, haralick@george.ee.washington.edu}

Abstract

This paper presents a text word extraction algorithm that takes a set of bounding boxes of glyphs and their associated text lines of a given document and partitions the glyphs into a set of text words, using only the geometric information of the input glyphs. The algorithm is probability based. An iterative, relaxation-like method is used to find the partitioning solution that maximizes the joint probability. To evaluate the performance of our text word extraction algorithm, we used a 3-fold validation method and developed a quantitative performance measure. The algorithm was evaluated on the UW-III database of some 1600 scanned document image pages. An area-overlap measure was used to find the correspondence between the detected entities and the ground-truth. For a total of $827,433$ *ground truth words, the algorithm identified and segmented* $806,149$ *words correctly, an accuracy of* 97.43%.

1. Introduction

A document structure analysis system converts a scanned document page or a document encoded by a Page Description Language (PDL), such as PostScript and Portable Document Format (PDF), into a well partitioned hierarchical representation that reliably identifies the basic document components – text words, text lines, and text blocks. Thus, extracting words (word segmentation) from a scanned document page or a PDF is an important and basic step in document structure analysis and understanding systems, but the task is not trivial. Incorrect word segmentation could lead to OCR errors and could also lead to errors in information retrieval and in understanding of the input document. This paper presents a text word extraction algorithm that takes a set of bounding boxes of glyphs and their associated text lines of a given document and partitions the glyphs into a set of text words, using only the geometric information of the input glyphs.

There are many document layout analysis algorithms in the literature; however, only several word segmentation methods can be found. Baird et al.'s word segmentation method [3] assumed that the distribution of the inter-symbol distances parallel to the text-line orientation is bimodal and segmented the words by finding an appropriate threshold. No performance evaluation of their text word segmentation was reported. Chen et al.'s method ([2]) used the recursive morphological closing transform to segment the words. He reported a 95% accuracy using hundreds of test images. Bapst et al.([1]) used typographic information to improve the existing word segmentation method and has shown good result. However, no quantitative performance evaluation was reported in this paper.

Our algorithm takes a set of glyph bounding boxes and their associated text lines of a given document. It partitions the glyphs into a set of text words. We adopt an engineering approach to systematically characterizing the text word based on a large document image database, and use the statistical methods developed in [4] to extract the text words from the image. All the probabilities are estimated from an extensive training set of various kinds of measurements among the glyphs and among the text words in the training data set. The off-line probabilities estimated in the training then drive all decisions in the on-line text word extraction. An iterative, relaxation-like method is used to find the partitioning solution that maximizes the joint probability. The algorithm was tested on 1600 document pages within the UW III document database. The evaluation result is reported in this paper.

The remainder of this paper is organized as follows: In Section 2, we present the abstract problem formulation. In Section 3, we describe the detail of our word segmentation algorithm. Our experimental protocol and results are given in Section 4. Our conclusions and statements of future work are discussed in Section 5.

Proceedings of the International Conference on Pattern Recognition (ICPR'00)
1051-4651/00 $10.00 @ 2000 IEEE

2. The Word Segmentation Problem Statement

Given a set of bounding boxes of glyphs and their associated text lines, the word segmentation problem is to partition the input glyphs into a set of text words that maximizes the probability of glyphs to word assignment.

Let $\mathcal{A}$ be the set of input glyphs. Let Π be a partition of $\mathcal{A}$ where each element of the partition is a word. Let L be a set of labels. The function $f : \Pi \rightarrow L$ associates each element of Π with a label. $V : \wp(\Pi) \rightarrow \Lambda$ specifies the measurement made on the subset of Π, where Λ is the measurement space. Let $A = (A_1, A_2, \cdots, A_M)$ be a linearly ordered set (chain in $\mathcal{A}$) of input entities. Let $\mathcal{R} = \{Y, N\}$ be the set of grouping labels. Let A^P denote a set of element pairs, such that $A^p \subset A \times A$ and $A^p = \{(A_i, A_j) | A_i, A_j \in A \text{ and } j = i + 1\}$. The function $r : A^p \rightarrow \mathcal{R}$, associates each pair of adjacent elements of A with a grouping label, and $r(i) = r(A_i, A_{i+1})$.

The consistent partition and labeling problem can be formulated as follows([4]): *Given an initial set $\mathcal{A}$, find a partition Π of $\mathcal{A}$, and a labeling function $f : \Pi \rightarrow L$, that maximizes the probability:*

$$\begin{aligned} & P(V(\tau) : \tau \in \Pi, f, \Pi | \mathcal{A}) \\ & \quad = P(V(\tau) : \tau \in \Pi | \mathcal{A}, \Pi, f) P(\Pi, f | \mathcal{A}) \\ & \quad = P(V(\tau) : \tau \in \Pi | \mathcal{A}, \Pi, f) P(f | \Pi, \mathcal{A}) P(\Pi | \mathcal{A}) \end{aligned} \tag{1}$$

We make an assumption of conditional independence: when the label $f(\tau)$ is known, no knowledge of other labels will alter the probability of $V(\tau)$. We use $P(\Pi|\mathcal{A}) = P(r|A)$ and let N be the number of elements in $\mathcal{A}$. We can decompose the probability (1) as follows:

$$\begin{aligned} & P(V(\tau) : \tau \in \Pi, f, \Pi | \mathcal{A}) \\ & = \prod_{\tau \in \Pi} P(V(\tau) | f(\tau)) P(f | \Pi, \mathcal{A}) \times \prod_{i=1}^{N-1} P(r(i) | A_i, A_{i+1}) \end{aligned} \tag{2}$$

The search space for the above equation is 2^{N-1}, where N is the number of input glyphs. Fortunately, the glyphs within words follows a particular sequential order, Thus, with the ordering constraint, the partitioning problem can be done iteratively. The next section describes an iterative search method of order $O(N)$ that finds the consistent partition labeling by monotonically maximizing the joint probability in equation (2).

3. Text Word Segmentation Algorithm

An iterative searching method can find the consistent partition and labeling that maximizes the joint probability (2). First, the grouping probability $P(r(i)|A_i, A_{i+1})$ between each pair of adjacent input entities is computed by observing the spatial relationship between the two input entities. An initial partition is determined based on the grouping probabilities. Then, we adjust the partition and assign labels to the members of the partition by optimizing the joint probability. At each iteration, the adjustment that produces the maximum improvement of the joint probability is selected. The iteration stops when there is no improvement on the joint probability.

The overview of our word segmentation algorithm is shown in Figure 1. The detailed description of the consistent partition and labeling algorithm can be found at [4]. We describe how we compute the three conditional probabilities in (2) in the subsections below.

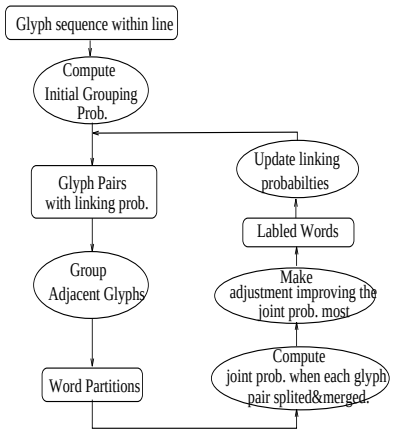


Figure 1. Overview of the word segmentation algorithm

3.1. Initial Grouping Probability

Without loss of generality, we assume that the reading direction of the text words in the given line is left-right. The text word segmentation algorithm starts with a set of the bounding boxes of the text glyphs within the given text line.

We first construct the reading order of the input glyphs. For each pair of adjacent glyphs within the same text line, g_i and g_{i+1}, we compute the probability that they are within the same text word:

$$P(r|g_i, g_{i+1}) = P(SameWord(g_i, g_{i+1}) | g_i, g_{i+1})$$

For each pair of horizontally adjacent glyphs g_i and g_{i+1}, the glyph is represented by a bounding box (x, y, w, h). Given the line l, where the line is represented by a bounding box (x_l, y_l, w_l, h_l), we make the following measurements:

- inter-glyph distance: $d(i, i+1) = x_{i+1} - x_i - w_i$
- left-top offset: $e_{lt} = y_i - y_l$
- left-bottom offset: $e_{lb} = y_i + h_i - y_l$

- right-top offset: $e_{rt} = y_{i+1} - y_l$
- right-bottom offset: $e_{rb} = y_{i+1} + h_{i+1} - y_l$

The inter-glyph distance $d(i, i+1)$ is normalized by the threshold, $thre_{otsu}$, which is calculated from the distance set for the given line using Otsu's algorithm([5]).

$$d_i = \frac{d(i, i+1)}{thre_{otsu}}$$

The other four measurements are all normalized by the given line height.

$$lt_i = \frac{e_{lt}}{h_l}, \qquad lb_i = \frac{e_{lb}}{h_l}, \qquad rt_i = \frac{e_{rt}}{h_l}, \qquad rb_i = \frac{e_{rb}}{h_l}$$

Given the above measurements, we compute the probability that g_i and g_{i+1} belong to the same word,

$$P(SameWord(i, i+1)|d_i, lt_i, lb_i, rt_i, rb_i)$$

3.2. Labeling Checking Probability

Given the initial word segmentation result, we have two sets of different types of horizontal distance. Let D_{iw} be the set of distances between the horizontally adjacent words and D_{ig} be the set of distances between the horizontally adjacent glyphs which belong to the same word. We have:

$$\begin{aligned} D_{iw} &= \{d_{iw}(i, i+1)|w_i, w_{i+1} \text{are horizontally adjacent words}\} \\ D_{ig}(j) &= \{d_{ig}(i, i+1)|g_i, g_{i+1} \text{are horizontally adjacent glyphs} \\ &\quad \text{and belong to the same word } j\ \} \end{aligned}$$

A text word usually has homogeneous inter-glyph distance and the inter-word distance is usually larger than the maximum inter-glyph distance of its adjacent words. Given one detected word W, we compute the conditional probability that W has homogeneous inter-glyph distance and appropriate inter-word distance. Assuming their conditional independence, we have

$$\begin{aligned} &P(V(W)|TextWord(W)) \\ &\quad = P(V(W)|IntG(W), IntW(W)) \\ &\quad = P(V_1(W)|IntG(W))P(V_2(W)|IntW(W)) \end{aligned}$$

Assuming that W is the jth segmented text word and it has m glyphs, we can estimate the conditional probability of it having homogeneous inter-glyph distance by:

$$\begin{aligned} &P(V_1(W)|IntG(W)) = \\ &P(\sum_{i=1}^{m-1}(|d_{ig}(i, i+1) - Median(D_{ig}(j))|)|IntG(W)) \end{aligned}$$

In a line, one inter-word distance should be larger than the maximum inter-glyph distance in its two adjacent words. Assuming that W is not the last glyph in the given line, we can estimate the conditional probability on its following inter-word distance by:

$$\begin{aligned} &P(V_2(W)|IntW(W)) = \\ &\quad P(d_{iw}(j, j+1) - Max(D_{ig}(j) \cup D_{ig}(j+1))|IntW(W)) \end{aligned}$$

3.3. Context Checking Probability

Given a line and its segmented words, we can expect the minimum inter-word distance should be larger than the maximum inter-glyph distance. Let $\mathcal{G}$ be the set of the glyphs in the given line, $\mathcal{W}$ be the set of the segmented words. Assuming that there are S words in the line, we can do the context checking by computing:

$$P(TextWord|\mathcal{W}, \mathcal{G}) = P(Min(D_{iw}) - Max(\bigcup_{j=1}^{S} D_{ig}(j))|\mathcal{W}, \mathcal{G})$$

4. Experimental Result

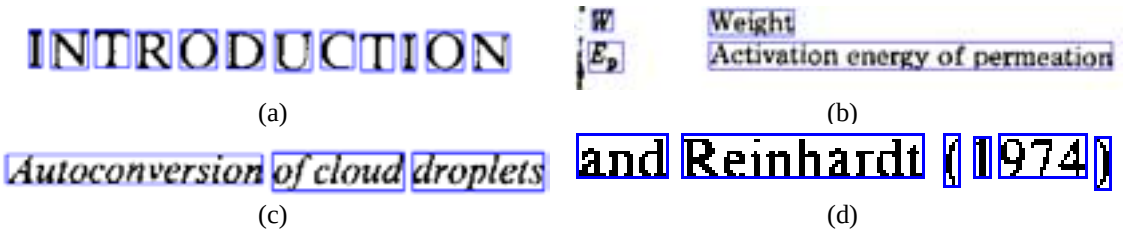
We use discrete contingency tables to represent the joint and conditional probabilities used in the algorithm. A tree structure quantization is used to partition the value of each variable into bins. At each node of the tree, we search through all possible threshold candidates on each variable, and select the one that gives minimum value of entropy of the resulting distribution. The total number of terminal nodes, which is equivalent to the total number of cells, is predetermined. For each joint or conditional probability distribution, a cell count is computed from the ground-truthed document images in the UW-III Document Image Database([6]). The cell count is simply the number of units in the sample whose quantized measurement vector falls in the given cell. The joint probability can be computed directly from the cell count. For the performance evaluation, we use an area-overlap measure to find the correspondence between the detected entities and the ground-truth([2]). We applied our word segmentation algorithm to the total of 1600 images from the UW-III Document Image Database using the cross validation method. Of $827,433$ ground truth words , the numbers and percentages of miss, false, correct, splitting, merging and spurious detections are shown in Table 1. Figure 2 shows one example page of the segmented word entities.

5. Conclusion and Future Work

This paper presents a statistical-based word segmentation algorithm based on the methods developed in [4]. The algorithm uses only the geometric information of the bounding boxes input glyphs. The algorithm was tested on the 1600 pages within UW-III Document Image Database and achieved a 97.43% accuracy rate. Figure 3 give a few

Table 1. Performance of the statistical-based word segmentation algorithm.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	827433	806149 (97.43%)	7602 (0.92%)	12193 (1.47%)	630 (0.08%)	859 (0.10%)
Detected	834048	806149 (96.65%)	21715 (2.60%)	4911 (0.59%)	367 (0.04%)	906 (0.11%)



(a) (b) (c) (d)

Figure 3. Illustrates examples that the word segmentation algorithm failed.

examples at which our algorithm failed. Our algorithm finds the global optimization by searching for the local optimization. When they do not match, glyphs may be segment as word individually, as shown in Figure 3(a). Our current context checking favors large inter-word distance, which gives us the kind of error shown in Figure 3(b). Other errors are due to the Italic fonts(Figure 3(c)) and the thin characters(Figure 3(d)). So our future work will include using a polygon instead of a rectangle as the entity enclosing box, doing the context checking in a larger region, and dealing with the small width characters and the various inter-word distances within one line.

References

[1] F. Bapst and R. Ingold. Using typography in document image analysis. *Electronic Publishing, Artistic Imaging, and Digital Typography. EP'98 & RIDT'98 Proceedings.*, pages 240–251, Mar./Apr. 1998.

[2] S. Chen, R. M. Haralick, and I. Phillips. Simultaneous word segmentation from document images using recursive morphological closing transform. *Proceedings of the 3rd ICDAR*, pages 761–764, Aug. 1995.

[3] D. J. Ittner and H. S. Baird. Language-free layout analysis. *Proceedings of the 2rd ICDAR*, pages 336–340, Oct. 1993.

[4] J. Liang. *Document Structure Analysis and Performance Evaluation*. Ph.D thesis, Univ. of Washington, Seattle, WA, 1999.

[5] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on SMC, Vol. SMC-9*, pages 62–66, 1979.

[6] I. Phillips. Users' reference manual. *CD-ROM, UW-III Document Image Database-III*, 1995.

Figure 2. Example of the word segmentation result

1051-4651/00 $10.00 @ 2000 IEEE