

## Chapter 8

# STATISTICAL BASED APPROACH TO WORD SEGMENTATION

### **8.1 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 chapter describes 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.

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 [55] 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 page images within the UW III document database. The evaluation result is reported in this chapter.

The remainder of this chapter is organized as follows: In Section 8.2, we review the previous work on word segmentation problem. In Section 8.3, we present the abstract problem formulation. In Section 8.4, we describe the detail of our word segmentation algorithm. Our experimental protocol and results are given in Section 8.5. We conclude this chapter with future work by Section 8.6.

## **8.2 Literature Review**

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 [38] 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 ([16]) used the recursive morphological closing transform to segment the words. He reported a 95% accuracy using hundreds of test images. Bapst et al.([4]) 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. In [57], Liang et. al. presented a performance evaluation protocol for the layout analysis. A set of layout analysis algorithms which detect different entities have been tested based on the data set and the performance metric. In the ISL document layout toolbox [62], there are two different word segmentation methods: top-down and bottom-up. Their performance evaluation results were reported in [57].

### 8.3 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  $\Pi$  be the partition of  $\mathcal{A}$  to be determined 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  $\Pi$  with a label.  $V : \wp(\Pi) \rightarrow \Lambda$  specifies the measurement made on the subset  $\Pi$ , where  $\Lambda$  is the measurement space. Let  $A = (A_1, A_2, \dots, A_M)$  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( [55]):  
*Given an initial set  $\mathcal{A}$ , find a partition  $\Pi$  of  $\mathcal{A}$  of input glyphs, and a labeling function  $f : \Pi \rightarrow L$ , that maximizes the probability:*

$$\begin{aligned} P(V(\tau) : \tau \in \Pi, f, \Pi | \mathcal{A}) \\ &= P(V(\tau) : \tau \in \Pi | \mathcal{A}, \Pi, f) P(\Pi, f | \mathcal{A}) \\ &= P(V(\tau) : \tau \in \Pi | \mathcal{A}, \Pi, f) P(f | \Pi, \mathcal{A}) P(\Pi | \mathcal{A}) \end{aligned} \quad (8.1)$$

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  $P(\Pi | \mathcal{A}) = P(r | \mathcal{A})$  and let  $N$  be the number of elements in  $\mathcal{A}$ . We can decompose the probability (8.1) as follows:

$$\begin{aligned} P(V(\tau) : \tau \in \Pi, f, \Pi | \mathcal{A}) \\ &= \prod_{\tau \in \Pi} P(V(\tau) | f(\tau)) P(f | \Pi, \mathcal{A}) \times \prod_{i=1}^{N-1} P(r(i) | A_i, A_{i+1}) \end{aligned} \quad (8.2)$$

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 (8.2).

#### **8.4 Text Word Segmentation Algorithm**

An iterative searching method can find the consistent partition and labeling that maximizes the joint probability (8.2). 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 8.1. The detailed description of the consistent partition and labeling algorithm can be found at [55]. We describe how we compute the three conditional probabilities in (8.2) in the subsections below.

##### *8.4.1 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:

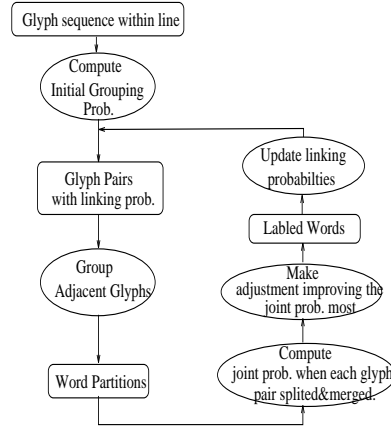


Figure 8.1: Overview of the word segmentation algorithm

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

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 ([74]).

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

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

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

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

$$P(\text{SameWord}(i, i + 1) | d_i, lt_i, lb_i, rt_i, rb_i)$$

#### 8.4.2 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

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

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

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

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:

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

#### 8.4.3 Context Checking Probability

Given a line and its segmented words, we can expect that 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:

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

### 8.5 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

Table 8.1: 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%)

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 ([76]). 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 ([16]). 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 8.1. Figure 8.2 shows one example page of the segmented word entities.

For comparison, we listed the word segmentation performance evaluation results reported in [57] in Table 8.2 and 8.3. The results were obtained using the same dataset and same performance evaluation protocol. We also listed the three algorithms' correct detection rates with respect to the ground truth data in Table 8.4. For the statistical-based and top-down algorithms, the detection rates are 97.43%

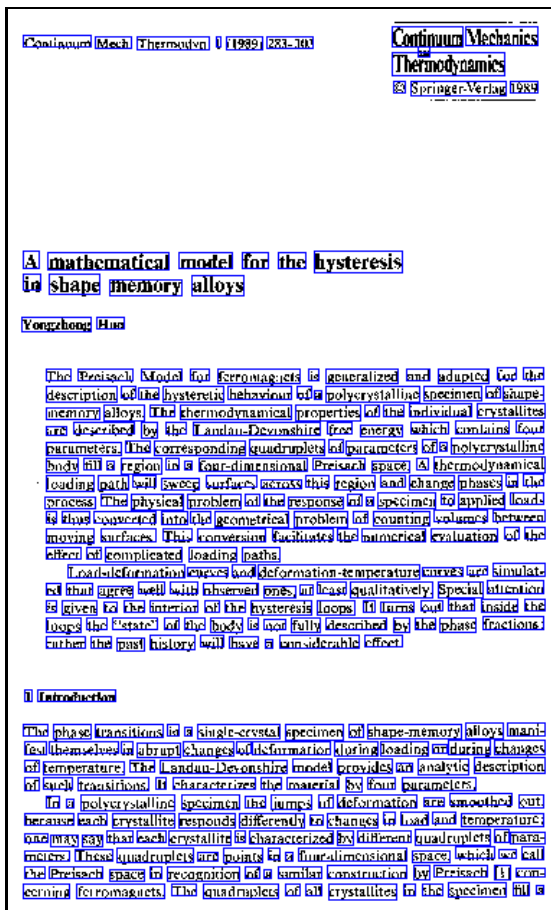


Figure 8.2: Example of the word segmentation result

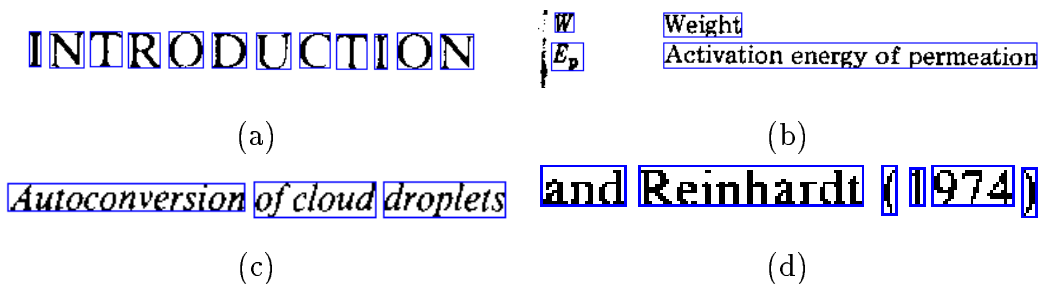


Figure 8.3: Illustrates examples that the word segmentation algorithm failed.

Table 8.2: Performance of the projection cut word segmentation algorithm(from [57]).

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	828201	800135 (96.61%)	3288 (0.40%)	16548 (2.00%)	7892 (0.95%)	338 (0.04%)
Detected	813419	800135 (98.37%)	7782 (0.96%)	5174 (0.64%)	147 (0.02%)	181 (0.02%)

Table 8.3: Performance of the morphological closing word segmentation algorithm (from [57]).

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	828201	656067 (79.22%)	19024 (2.30%)	100851 (12.18%)	47008 (5.68%)	5251 (0.63%)
Detected	786713	656067 (83.40%)	46654 (5.93%)	35699 (4.54%)	44007 (5.60%)	4286 (0.54%)

and 96.61%, respectively. We use the total ground truth number as 827,433( The difference between 828,201, reported in [57] and 827,433, reported here might be due to the update of the CDROM ). The  $Z$ -test [68] gives  $Z = 31.03$  with the  $p$ -value is far less than  $10^{-10}$ . Since the performance of bottom-up algorithm is even lower, we will get the similar  $Z$  value when we compare the statistical-based algorithm with the bottom-up algorithm. Hence there is strong evidence that the detection performance of the statistical-based algorithm is significantly better than the other two algorithms. From this calculation, we can also easily conclude that the statistical-based algorithm has very significant better performance than other algorithms in this data set.

Table 8.4: Three word segmentation algorithms' correct detection rates with respect to the ground truth data

Statistical-based	Top-down	Bottom-up
97.43%	96.61%	79.22%

## 8.6 Summary

This chapter presents a statistical-based word segmentation algorithm based on the methods developed in [55]. 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 8.3 give a few 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 8.3(a). Our current context checking favors large inter-word distance, which gives us the kind of error shown in Figure 8.3(b). Other errors are due to the Italic fonts( Figure 8.3(c)) and the thin characters(Figure 8.3(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.