Layout Analysis based on Text Line Segment Hypotheses

Thomas M. Breuel

Abstract *This paper describes on-going work in the development of a document layout analysis system based on text line segments. It describes two novel algorithms: gapped text line finding, which can identify text line segments, taking into account per-text line font information for the determination of where text line segments break, and reading order recovery for text line segments using topological sorting. An extension of the approach to a probabilistic maximum likelihood framework is discussed.*

1 Introduction

Document layout analysis has usually been formulated as a top-down problem, finding large scale, two-dimensional features of page layout, such as columns, whitespace, and other features. While such approaches have proved very successful for a large fraction of document layouts, they fail to take advantage of some linguistic and statistical information. This becomes evident on atypical pages, for example the last page of a two-column article when it contains only a small number of text lines.

More recently, a number of authors have proposed bottom-up statistical approaches to layout analysis. For example, [4] proposes estimating statistics of the relative geometric arrangement of individual characters, lines, and paragraphs, and jointly optimizing this model to arrive at a maximum likelihood interpretation of the document layout. Closely related to such ideas is the document image decoding (DID) approach [3] to combined layout analysis and optical character recognition (OCR): DID also proposes treating the OCR and document layout analysis problem as a single, combined maximum likelihood estimation problem. Both these approaches are computationally expensive and involve difficult parameter estimation problems.

This paper first describes two algorithms that permit a geometric approach to layout analysis that starts with text line segments and recovers reading order and layout information from that. The first algorithm can either yield an single partition of the input text into text line segments, or a collection of alternative segmentations into text line segments. These text line segments are then put in reading order by extending a known partial order of the text line segments to a total order.

The paper then describes how these methods can be applied in a probabilistic framework in order to achieve a maximum likelihood interpretation of the input according to a statistical model derived from training data. The idea is to use geometric algorithms to find a collection of plausible candidate components of a document layout analysis, represent the spatial relationship among those candidate components as a graph structure, and then to combine those candidates into a globally optimal interpretation of the document layout.

2 Finding Gapped Line Segment Candidates

The basic framework for text line finding used in this work is that described in [2]. The idea is to model text lines as several parallel lines and find statistically optimal and robust matches of these models against page images; this is shown in Figure 1. Individual characters are represented by their bounding boxes (or, more generally, a convex polygonal hull). In Latin scripts, each bounding box rests either on the baseline or line of descenders. Furthermore, the top of each character reaches either the x-height or the ascender height. Each line model is therefore determined by five parameters: two parameters, (θ, r), angle and distance from the origin, for the baseline, and three parameters giving the offset for the line of descenders, the x-height, and the ascender, (h_d, h_x, h_c). Furthermore, a user-specified parameter gives the maximum amount of error permissible under a robust least-square error model.

Matches against this text line model are found using the line finding algorithm described in [2], implemented using interval arithmetic. The approach is a branch-and-bound global optimization that explores the five-dimensional space of transformation parameters. For its application, all that needs to be specified is an evaluation function $Q(\theta, r, h_d, h_x, h_c)$; the algorithm will automatically find the globally optimal matches against that model in decreasing order of quality of match. For details of the algorithms, the reader is referred to the references.

This basic text line finding algorithm is known to work reliably for single column documents. However, for multi-column documents, it is essential that text lines do not span different columns. First of all, characters in differ-

Figure 1. Text line model used by the gapped text line segment finding algorithm.

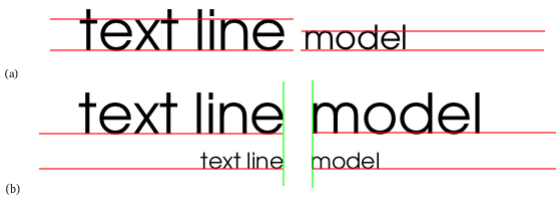

Figure 2. Changes in text line models that cause text line breaks to be detected.

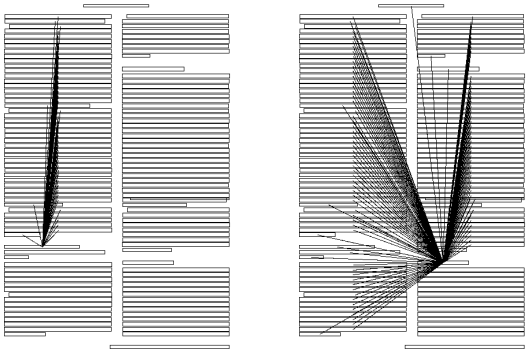

Figure 3. Examples of pairwise reading order relationships determined for text line segments in actual documents.

ent columns do not belong together logically. Secondly, misidentifying characters that are part of different columns as belonging to the same text line may lead to incorrect estimates of the baseline.

One approach to this problem has been proposed in [2], which formulates the problem of multi-column text line finding in two separate steps: in the first step, a column boundary finder identifies whitespace that is present between text columns; then, a constrained text-line finder finds text line segments that do not intersect any of the identified column boundaries. While a simplistic implementation of this constrained text line segment problem would appear to add two extra parameters, the start and end of the text line segment, to the evaluation function Q, it turns out that the use of matchlists permits a more efficient implementation [2]. The idea is to define, in addition to Q, a splitting function S. This splitting function takes the set of characters matching a text line and either returns the same set, or returns two disjoint subsets indicating a split of the line into two subsegments.

While this approach works reliably for most pages, a few pages have unusual layouts in which text columns cannot be identified reliably from whitespace analysis and need to be inferred based on other properties. Two examples of this are shown in Figure 2. In the first example, a likely break in a text line is indicated by the fact that words on the left and on the right are significantly different in font size. In the second example, a likely text line break is indicated by the presence of a large whitespace gap relative to the font size; note that only the second text line is likely to contain a break–for the font size in the first line, the whitespace gap is consistent with inter-word spacing.

The case depicted in Figure 2(a) is already detected by the use of a five parameter text line model in this work (previous methods only modeled the baseline and line of descenders, which cannot detect changes in font size reliably): even if the baselines of two text line segments using fonts of different sizes align, their differing x-heights will mean that they match separate text line segments. To address the case depicted in Figure 2(b), we define a splitting function S that estimates the inter-word whitespace from the font size of the characters present in the currently matching set of characters and then splits the line at gaps that are wide relative to the inter-word whitespace.

Using a quality of match function Q and splitting function S gives us a single segmentation of the connected components on a page into text line segments. In order to obtain alternative possible segmentations, when S indicates a possible split, we can add both the split and the unsplit alternatives to the priority queue used in the search, and the gapped text line matching algorithm will return a collection of alternative segmentations of the input into text line segments.

3 Reading Order by Topological Sorting

Let us assume, for the time being, that the text line segment finder has returned a unique segmentation of the connected components on the page into text line segments. For many kinds of documents, such a unique segmentation can, in fact, be determined based on text line segment finding with obstacles (e.g., column boundaries, explicit column separators) [2] or a combination of such techniques with gapped text line segment finding described above. Even if we have such a unique segmentation, we are still left with the problem of grouping those text line segments into paragraphs and columns, and then to determine the order of those layout elements.

While other approaches to document layout analysis attempt to determining paragraphs and columns directly from

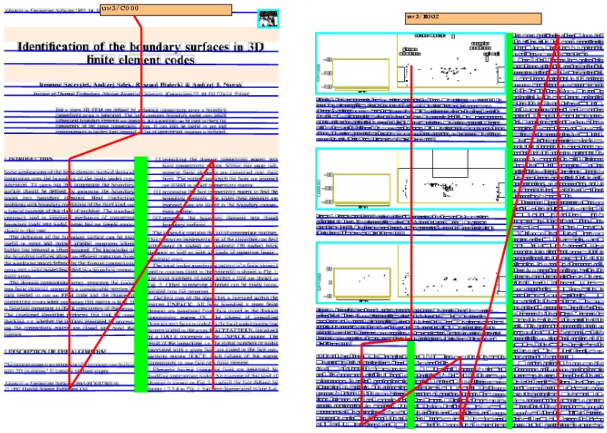

Figure 4. Examples of reading order recovery by topological sorting. Blue lines indicate text line segments, and the red line connects the center of successive text line segments in reading order.

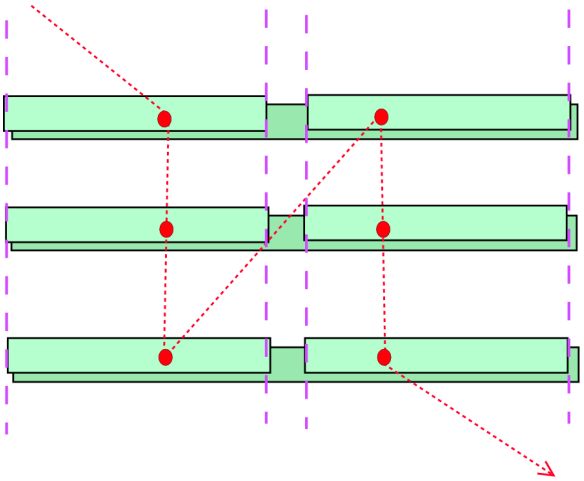

Figure 5. Schematic illustration of the probabilistic segmentation framework described in the text. A gapped text line segmentation algorithm will return alternative interpretations of the input in terms of text line segments (shown as green rectangles). A maximum likelihood interpretation of the page takes into account that each character can only participate in one text line segment, the likelihood of columnar alignments of the beginnings and ends of text line segments (shown as dashed purple lines), as well as the likelihood (according to a language model) of the text that results from concatenating the text lines in the given order. The red, dotted line shows what would probably the most likely interpretation of the input in terms of text lines.

whitespace analysis or the spacing of connected components, the approach taken in this work is different: after finding text lines, our method determines reading order of the text line segments. Columns, paragraphs, and other layout features then follow from the spatial arrangement of text line segments in reading order. For example, paragraph boundaries are indicated by relative indentation of consecutive text lines in reading order.

The idea behind determining reading order of text line segments is the following. While it is impossible in general to determine for two arbitrary text line segments what their reading order is, for some pairs of text line segments, reading order is easy to determine.

While those rules give us pairwise order relationships among certain text line segments, in order to obtain a total reading order of text line segments, we need to extend this partial order into a consistent total order of all text line segments. The algorithm for doing this is topological sorting: topological sorting takes a collection of objects (text line segments) and a partial order, and finds at least one global order consistent with this partial order.

Examples of reading order recovery by topological sorting are shown in Figure 4. For those documents, as well as almost all other documents in the "A" and "C" section of the University of Washington database (the only sections in the UW3 database examined), four simple rules turn out to be sufficient to recover reading order accurately.

4 Probabilistic Extension

While the deterministic approach to layout analysis described in the previous sections–text line segment findings followed by reading order recovery using topological sorting–works quite well on many documents, a significant fraction of documents are ambiguous at the layout level and may require integration of multiple sources of information, including OCR and font information, layout information, and language model information, in order to arrive at a good overall interpretation of the document layout. For example, in the absence of clearcut cues from the geometric layout, determination textual continuity may be required to determine which text line segment follows which other text line segment.

A theoretically sound framework for integration of this kind of disparate evidence is given by Bayesian statistics, and, in particular, Bayesian networks. In fact, such integration has been used in a number of previous systems. For example, the document image decoding (DID) approach [3] uses two-dimensional hidden Markov models with a channel model for document degradation and an integrated lan-

guage model for finding the most likely interpretation of both the layout and the content of a document image. More recently, an approach equivalent to Markov random fields, using learned estimates of joint probabilities has been proposed [4].

While a probabilistic or Bayesian framework itself is well motivated, whether a particular approach is feasible depends on the exact nature of the models being adopted: some choices of probability models may be difficult to learn and difficult to optimize.

Experience with the text line segment models and recovery of reading order described above shows that many documents are handled well by such an approach. This suggests taking the deterministic model as a starting point, but resolving ambiguities using learned statistical models when they occur.

More formally, this approach can be justified by observing that what we would like to identify is the segmentation $\mathcal{S}$ of the document $D = \{d_1, \ldots, d_N\}$ (where the d_i are, for example, the connected components of the page image), that has the maximum *a posteriori* probability, $\arg\max_{\mathcal{S}} P(\mathcal{S}|D) = P(\mathcal{S}|d_1, \ldots, d_N)$. The document image decoding approach postulates a certain Markovian structure to this model, while Markov random field approaches postulate that $P(\mathcal{S}|d_1, \ldots, d_N)$ can be approximated well as a function f of marginals of a small number of (usually) neighboring components, $P(\mathcal{S}|d_1, \ldots, d_N) \approx f(\{P(\mathcal{S}|d_i, d_j, d_k)|i, j, k \in \text{local neighborhood}\})$.

In contrast, the underlying statistical model for the probabilistic framework to document analysis proposed here represents a very different approximation of $P(\mathcal{S}|D)$: the low-level segmentation and grouping processes (gapped text line segments, etc.) generate "chunks" S_i that can be combined into an overall segmentation $\mathcal{S}$. That is, $P(\mathcal{S}|D)$ is approximated as a function of $P(D|S_i)$, where the $P(D|S_i)$ are, for example, the likelihood associated with matching a particular text line segment model against the page (the gapped text line finder described above estimates these likelihoods), the probability that a collection of three text line segments are in a particular reading order (a probabilistic analog of the "four rules" mentioned above), and the probability assigned to a particular reading order according to a language model. This is illustrated schematically in Figure 5.

5 Discussion and Conclusions

The gapped text line finding algorithm described at the beginning of this paper differs from previous methods for text line segment finding based on branch-and-bound methods in that it can return alternative interpretations of the input, and that it can take into account during segmentation the font size and other properties associated with each individual text line. This allows it to detect gaps as shown in Figure 2 The gapped text line finding algorithm has been implemented and runs in time comparable to the text line finding algorithm in the presence of obstacles (of the order of 1 sec on modern PC hardware for typical documents).

The deterministic reading order algorithm has also been implemented and, as mentioned above, found to work for a large fraction of the documents in the UW3 database[1]

Work on implementing the probabilistic algorithm is ongoing. Previous has work demonstrated the learning of pairwise probabilities in a discriminative framework for probabilistic segmentation in an OCR-by-clustering task [1], which is very similar in structure to the layout analysis framework presented here. That work used simulated annealing to obtain the maximum likelihood solution. However, because formulating the problem of probabilistic layout analysis as that of recovering reading order transforms it into a one-dimensional problem, it is possible to use a Viterbi algorithm to search for a maximum likelihood solution for the given geometric and linguistic constraints, providing a convenient and efficient way to incorporate large language models.

This paper has described two novel algorithms for layout analysis, gapped text line segment finding and reading order recovery by topological sorting. These algorithms can already be used for building document layout analysis systems that work on a wide variety of documents. However, layout analysis cannot be performed unambiguously based on distribution of connected components, and an integration of multiple sources of evidence is needed. The paper has also described on-going work in our lab to integrate different sources of evidence into a maximum likelihood interpretation of the document layout in a way that takes advantage of the algorithmic efficiencies of previous methods while approximating the Bayes-optimal decision criteria.

References

[1] T. Breuel. Classification by probabilistic clustering. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (ICASSP 2001)*, pages 1333–1336, 2001.

[2] T. M. Breuel. Two algorithms for geometric layout analysis. In *Proceedings of the Workshop on Document Analysis Systems, Princeton, NJ, USA*, 2002.

[3] G. E. Kopec and P. A. Chou. Document image decoding using Markov source models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):602–617, June 1994.

[4] Y. Wang, I. T. Phillips, and R. Haralick. Statistical-based approach to word segmentation. In *Proceedings of 15th International Conference on Pattern Recognition (ICPR2000), Barcelona*, pages 555–558, 2000.

[1] More experimental details are contained in a journal article currently under review.