A Hierarchical Document Description and Comparison Method

Burak Bitlis, Xiaojun Feng, Jacob L. Harris, Ilya Pollak, Charles A. Bouman, Mary P. Harper, Jan P. Allebach
School of Electrical and Computer Engineering, Purdue University
1285 EE Building, West Lafayette, Indiana 47907

Abstract

Determining the similarity of document images is an important first step for several document retrieval tasks, such as document classification, information extraction, and retrieval based on visual similarity. In this paper, we propose a method to describe and compare the content and layout of a document given only an image of the document. A tree structure is used to capture the hierarchical structure of the document. Two documents are then compared using a tree matching strategy.

1. Introduction

Determining the similarity of document images is an important first step for several document retrieval tasks, such as document classification, information extraction, and retrieval based on visual similarity [5]. These document retrieval tasks are important when dealing with large databases of document images. For example, document retrieval is necessary when a database user wants to find documents in the database which are similar to a particular example document. Document classification could be useful in deciding where to file a document image being added to the database, or in choosing the most appropriate document model, if the models are different for various classes. All of these tasks require some notion of document similarity. When assessing what makes document images similar, we consider both of the following to be important: the content of the document (words, pictures, etc.) and the layout of the document (how the content of the document is organized). In this paper, we propose a method to describe and compare the content and layout of a document given only an image of the document. A tree structure is used to capture the document's hierarchical structure. Two documents are then compared using a tree matching strategy.

The document comparison problem has been studied for some time. Some methods depend only on the text content of the document, as in [2, 3, 7]. Such methods have proven to be very effective for comparing documents which are all or mostly text, but they do not take into account either the pictures and graphics that might be present in the document or the physical layout of the document. Srihari, Zhang, and Rao [10] give an example of a method that uses both text and picture information to query a document database, but this method still does not make use of the layout of the document. The method of Hu *et al.* [5] classifies documents based on the layout of text regions, but does not take into account the content of the text regions or picture regions. All of these methods either do not take the document layout into account or do not describe the layout in a hierarchical manner. An example of describing the layout of a document using a hierarchical structure appears in [6], which uses a stochastic regular grammar to produce a segmentation of the document.

We introduce a method to describe both the layout structure and the content of a single-page document image. We begin by partitioning the document into meaningful regions and calculating several features of these regions. We then define a distance between regions, and as in [8], we recursively merge the two nearest regions until all regions have been merged. Thus we create a binary partition tree whose nodes correspond to regions in the document and contain features to describe these regions. The leaves of the tree correspond to the various small regions of a document such as paragraphs, images, captions, etc., while an internal node of the tree corresponds to a logical grouping of all the leaf regions descending from it. For example, an internal node could correspond to a column of text, which is the union of several paragraphs. While the general idea of describing an image with a binary tree is similar to [8, 9], our definition of features and of the rules for merging regions are novel and specific to our application.

We then compare two document images by comparing their document trees. The criterion for this comparison is based on an extension of the concept of edit distance between trees [11].

The organization of this paper is as follows. Section 2 describes how the document trees are constructed, Section 3 describes how the distance between two documents is measured, Section 4 presents preliminary examples illustrating our algorithm, and Section 5 contains some concluding remarks.

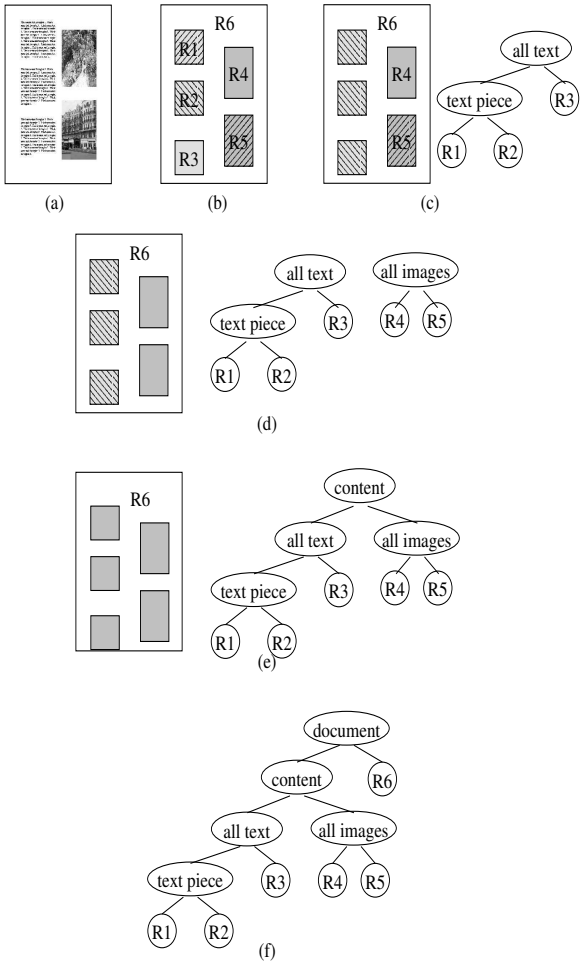

Figure 1: *Document tree illustration. (a) A document with three text regions and two image regions. (b) The regions of the document before region merging. (c,d,e) The document tree and remaining regions of the document after two, three, and four merges, respectively. (f) The final document tree describing the image in (a).*

2. Forming the Document Tree

The document tree is a binary tree which describes both the content and the layout of the document. It is formed by first segmenting the document into meaningful regions, then calculating features of these regions, and finally defining a distance between regions based on these features. The distance is then used to merge the regions recursively into a binary tree describing the document. Fig. 1 illustrates this concept. Given a document and a partition of the document into meaningful regions, we recursively merge similar regions. In Fig. 1(f), the root node corresponds to the entire document, which is split into background and content. The content is split into text and image regions, which are then split into their smaller components.

The recursive region merging procedure used to create the document tree implies the following: (i) the document has been segmented into regions, and (ii) there exists a way to discern which regions are "closest", that is, which regions should be merged first. Task (i) can be accomplished with any image segmentation program that partitions the image into several regions. To accomplish task (ii), we first define for each region several features to describe the region. We then define a distance function between regions that depends only on the feature vectors of the two regions being compared. This distance function is the criterion for deciding the order in which regions are merged while forming the document tree. The region pair with minimum distance is merged recursively until only one region remains.

Figure 2: *Block diagram of segmentation procedure*

2.1. Segmentation

The procedure for segmenting a document is illustrated in Fig. 2. The raw input file goes through a preprocessing step to produce an image that can be passed to the particular segmentation algorithm that was chosen. The output of this segmentation algorithm is then passed through a postprocessing step that removes any unwanted noise or insignificant details in the segmentation, yielding a useful segmentation of the document. The regions defined by this useful segmentation of the original document become the leaves of the document tree.

Note that this strategy does not specify the use of one specific segmentation procedure. Any reasonable segmentation procedure with appropriate preprocessing and postprocessing steps could be used. We chose to use the Trainable Sequential Maximum a Posteriori (TSMAP) segmentation algorithm in [1] to perform the segmentation of the documents. The TSMAP algorithm uses Bayesian multiscale techniques to find an estimate of which class should be assigned to each pixel in a grayscale image. We trained the TSMAP algorithm to assign one of the following four classes to the pixels of the image being segmented: header text, body text, image, and background. The output of the TSMAP segmentation step is then an image whose pixels contain the class numbers assigned to the corresponding pixels in the original document.

2.2. Features

Once the document has been split into regions, we need to define the feature vectors and a corresponding distance function which will be used to create the document tree. To do this, we need to have some measure of similarity between regions. Of course, this measure of similarity is dependent on the application, giving rise to many possible feature vectors and distance functions. For this paper, we

restrict our measure of similarity to depend on the following five characteristics of the regions: class, size, position, color, and text content.

Class: The segmentation program assigns to each region one of several classes. Our measure of similarity depends on class in the following way. The classes of the regions being compared determine how heavily to weight the other four closeness factors. For instance, the text content of the regions is much more important in a comparison of two body text regions than in a comparison of two image regions.

Size: Small regions should be merged before large regions. For instance, we want the various text regions of a document such as paragraphs to merge with each other into one large text region before any of them merge with a huge region such as the background.

Position: Region pairs which are spatially close should be merged before region pairs which are spatially far from each other. For instance, in a column of three paragraphs, the first merge should not be the top-bottom pair.

Color: Regions with similar color characteristics should be considered close.

Text Content: Regions with similar text content should be considered close. This applies more to regions containing much text than to regions containing little or no text.

We emphasize that these five measures of similarity are assumptions specific to our particular example, and that for other applications, other assumptions guiding the measure of similarity could be more appropriate.

The regions of the image are defined as the connected components of the postprocessed segmentation of the image. Several features are defined in [4] that describe the region in terms of the five similarity characteristics above. Once the features are calculated, they are the only information about the regions that is saved and used. The features alone determine the distance [4] between regions when the document tree is formed, and each node of the document tree contains the aggregate features that describe the section of the document descending from that node. We define the distance function between regions in [4] to implement the above assumptions about what makes regions similar.

3. Distance Between Document Trees

In a document database, each document can be characterized by a binary tree as described in Section 2. Then the similarity between documents can be defined by the distance between their corresponding trees. This section describes the similarity criteria that we use in our document processing system. To define the distance between two trees, we extend the concept of edit distance [11].

We let $r(T)$, $L(T)$, and $R(T)$ be the root node, the left subtree, and the right subtree of a tree T, respectively. $|T|$ denotes the number of nodes in the tree T. We define the yield $Y(T)$ to be the set of all leaf nodes of tree T. A *yield-partitioning set* of any tree T is defined to be a set of subtrees of T whose yields form a partition of $Y(T)$.

We first define the distance between two trees for several simple scenarios, and then generalize to any pair of trees using a recursive formula. First, the distance between a tree and itself is 0, $\|T,T\| \stackrel{\Delta}{=} 0$. If two trees T and $\hat{T}$ consist of a single node each, we set the distance between them to $\chi(r(T), r(\hat{T}))$ where χ [4] is an application-specific function which is symmetric ($\chi(x,y) = \chi(y,x)$). For the general T and $\hat{T}$, we use the recursive definition in Eq. (1), where $\gamma(T,\hat{T})$ is a function which penalizes the dissimilarity between yield-partitioning sets of T and $\hat{T}$; $\delta(R(T))$ and $\delta(L(T))$ represent the costs of pruning $R(T)$ and $L(T)$, respectively; and w is an application-specific weight such that $0 \leq w \leq 1$.

The intuition behind the recursive part of formula (1)–i.e., the second term–is as follows. One situation when we would like to say that T and $\hat{T}$ are similar is if their roots correspond to similar regions, and their set of leaves describe similar sets of subregions. We would therefore like to compare the respective roots of T and $\hat{T}$ as well as their yield-partitioning sets. This is what the last line of Eq. (1) does. Another case when we would like to say that two regions are similar is when one of them is a subregion of the other, provided that the difference between these two regions is not significant. The rest of the recursion is an implementation of this comparison.

As suggested by Eq. (1), the distance between two trees can be calculated using a recursive algorithm. To avoid redundant computation, whenever the distance between two subtrees is computed, it is stored. The distance between each pair of subtrees is thus computed only once. The time complexity of this algorithm is $O(N\hat{N})$ where N and $\hat{N}$ are the numbers of nodes in the trees T and $\hat{T}$, respectively.

4. Preliminary Evaluation

In the previous sections, we have described a general strategy for forming and comparing hierarchical document descriptions called document trees. This section describes some results of applying this specific strategy to documents in our database.

The database we used for the experiment consisted of 41 one-page documents in PDF format. We converted each one to a 400 dpi (157.5 dpcm) color image to obtain 41 page images to test our method. We then formed a document tree to describe each of these document images. The first step of forming the document tree is to segment the document into regions. The images were first put through the segmentation procedure whose block diagram appears in Fig. 2. Once we had a segmentation of the original im-

$$\left\|T,\hat{T}\right\| = \left\|\hat{T},T\right\| \triangleq \begin{cases} \min\begin{pmatrix} \chi(r(T),r(\hat{T})), \\ \chi(r(L(T)),r(\hat{T}))+\delta(R(T)), \\ \chi(r(R(T)),r(\hat{T}))+\delta(L(T)) \end{pmatrix}, & \text{if } |T|>1, |\hat{T}|=1, \\ \min\begin{pmatrix} \delta(L(T))+\left\|R(T),\hat{T}\right\|, \\ \delta(R(T))+\left\|L(T),\hat{T}\right\|, \\ \delta(L(\hat{T}))+\left\|T,R(\hat{T})\right\|, \\ \delta(R(\hat{T}))+\left\|T,L(\hat{T})\right\|, \\ w\cdot\chi(r(T),r(\hat{T}))+(1-w)\cdot\gamma(T,\hat{T}) \end{pmatrix}, & \text{if } |T|>1, |\hat{T}|>1, \end{cases} \quad (1)$$

age, we found the connected components of the segmentation to define the regions of the document and recursively merged these regions to form the document tree. Using these trees, we then calculated the distances between documents using the method explained in Section 3.

For testing our system, we chose a document and calculated the distance between its document tree and all the other document trees. Fig. 3 shows the test document and the document closest to this test document. Also below each document, their corresponding document trees are provided. In Fig. 4, the 2nd through 10th closest documents to the test document are shown, arranged in the order of increasing distance from the test document.

5. Conclusions

We have proposed a general framework for capturing and comparing the hierarchical structure of document images. Our preliminary experiments show that our method produces results that make sense intuitively. Our future work will focus on training the weights in the distance functions and evaluating the importance of features.

6. Acknowledgments

This work was supported in part by a National Science Foundation (NSF) CAREER award CCR-0093105, NSF grants BCS-9980054 and RCV-0329156, a Xerox Foundation grant, ARDA VACE II grant MDA904-03-C-1788, and funds from a Michael J. and Katherine R. Birck Chair in Electrical and Computer Engineering. Part of this work was carried out while the sixth author was on leave at NSF. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the view of NSF. The authors thank E. Maia and Prof. J.M. Siskind for helpful discussions. The tree matching code we wrote and used for the experiments in the paper was an extension on an algorithm developed by Prof. Siskind.

References

[1] H. Cheng and C.A. Bouman, Multiscale Bayesian Segmentation Using a Trainable Context Model. *IEEE Trans. on Image Processing*, 10(4):511-525, April 2001.

[2] J.W. Cooper, A.R. Coden, and E.W. Brown. Information retrieval models: Detecting similar documents using salient terms. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pp. 245-251. ACM, November 2002.

[3] S. Deerwester, S.T. Dumais, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391 – 407, 1990.

[4] J. Harris. *A hierarchical document description and comparison method.* MS thesis, School of Electrical and Computer Engineering Purdue University, West Lafayette, IN, 2003.

[5] J. Hu, R. Kashi, and G. Wilfong. Comparison and classification of documents based on layout similarity. *Information Retrieval*, 2:227-243, May 2000.

[6] T. Kanungo and S. Mao. Stochastic language models for style-directed layout analysis of document images. *IEEE Trans. on Image Processing*, 12(5):583 – 596, May 2003.

[7] L. Ma, J. Shepherd, and A. Nguyen. Document classification via structure synopses. In X. Zhou and K.-D. Schewe, editors, *Proceedings of the Fourteenth Australasian Database Conference on Database Technologies*, vol. 17, Adelaide, Australia, 2003.

[8] J.-M. Morel and S. Solimini. *Variational Methods in Image Segmentation*, vol. 14 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhäuser, Boston, 1995.

[9] G. Qiu and S. Sudirman. Color image coding, indexing and retrieval using binary space partitioning tree. In *Proc. of IEEE Int'l Conf. on Image Proc.*, 1:682-685, October 7-10 2001.

[10] R. K. Srihari, Z. Zhang, and A. Rao. Intelligent indexing and semantic retrieval of multimodal documents. *Information Retrieval*, 2:245-275, May 2000.

[11] K. Zhang, D. Shasha, Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6), December 1989.

A. Presenter Bios

Burak Bitlis received his B.S. degree in 2001 from Electrical and Electronics Engineering department of Bogazici University, Istanbul. Currently, he is pursuing his Ph.D degree at Purdue University and he is a research assistant in Electronic Imaging Systems Laboratory. His research interests are document management and image databases. He is a member of IEEE Signal Processing Society.

Xiaojun Feng received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2000, and the M.S. degree from Purdue University, West Lafayette, Indiana in 2002. She is currently a Ph.D. student of electrical and computer engineering at Purdue University.