Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research Report
RR-03-02

Proceedings of the Third International Workshop on Document Layout Interpretation and its Applications (DLIA2003)

co-located with ICDAR2003

Edinburgh, Scotland

Andrew D. Bagdanov and Bertin Klein (Eds.)

2 August 2003

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210
E-Mail: info@dfki.uni-kl.de

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341
E-Mail: info@dfki.de

WWW: http://www.dfki.de

The editors of this work have been supported in part by the Universiteit van Amsterdam and the Deutsches Forschungszentrum für Künstliche Intelligenz, and would like to take this opportunity to express their heartfelt gratitude.

ISSN 0946-008X

Proceedings of the Third International Workshop on Document Layout Interpretation and its Applications (DLIA2003)

Andrew D. Bagdanov and Bertin Klein (Eds.)

2 August 2003

Preface

This report contains the proceedings of the Third International Workshop on Document Layout Interpretation and its Applications (DLIA2003). The workshop was held in Edinburgh, Scotland, on 2 August 2003, in conjunction with the 7th International Conference on Document Analysis and Recognition (ICDAR2003). The first DLIA workshop was held in Bangalore, India, and the second in Seattle, USA, as satellite events to ICDAR99 and ICDAR2001.

The techniques of document layout segmentation have reached such a level of maturity that researchers are beginning to explore the wealth of possibilities enabled by the availability of accurate layout information. The purpose of this international workshop is to bring together researchers, developers, and users of document understanding technology to explore the following question:

> *Assuming accurate layout information is available, what now?*

Of course, before approaching such a broad, open-ended question, it is essential to determine a task-dependent definition of accuracy for layout information. Once the interpretation purpose is fixed, the result of a specific layout analysis technique can be poor or very good.

We feel that the papers in this proceedings reflect the many views on this question prevalent in current document layout understanding research. A specific goal of this workshop series is to bring together many researchers, with many different research directions and philosophies on the nature of document layout interpretation. While the classical problem of document layout detection and segmentation remains an active area of research, the majority of the papers in this collection deal with the *utilization* of detected layout structure. By approaching problems from this viewpoint, we hope to stimulate discussion on the frontier of document understanding research, its direction and its future.

Program Committee

This workshop would not have been possible without the invaluable assistance of the program committee:

Apostolos ANTONACOPOULOS, University of Liverpool, U.K.
Henry BAIRD, Palo Alto Research Center (PARC), USA
Larry BAUM, The Boeing Company, USA
Thomas BREUEL, Palo Alto Research Center (PARC), USA
Andreas DENGEL, DFKI, Germany
David DOERMANN, University of Maryland, USA
Jianying HU, Avaya Labs Research, USA
Hiromichi FUJISAWA, Hitachi CRL, Japan
Yasuto ISHITANI, Toshiba Corporation, Japan
Koichi KISE, Osaka Prefecture University, Japan
Donato MALERBA, Universita degli Studi di Bari, Italy
Simone MARINAI, Universita di Firenze, Italy
Lawrence O'GORMAN, Avaya Labs Research, USA
Larry SPITZ, Document Recognition Technologies, USA
Luc VINCENT, LizardTech, USA
Marcel WORRING, University of Amsterdam, The Netherlands

We would to take this opportunity to express our heartfelt gratitude to all of the program committee members. Their helpful suggestions, not to mention the time taken from their busy schedules to review submissions, have helped to make this third installment of the DLIA workshop a success.

Andrew D. Bagdanov and Bertin Klein
Amsterdam, July 2003

Contents

Document Layout Problems Facing the Aerospace Industry

Lawrence S. Baum, John H. Boose, Molly Boose, Carey S. Chaplin, James Cheung, Ole B. Larsen, Monica Rosman Lafever, Ronald C. Provine, David Shema
Boeing Phantom Works
larry.baum@boeing.com, john.boose@boeing.com, molly.boose@boeing.com, carey.chaplin@boeing.com, james.cheung@boeing.com, ole.larsen@boeing.com, monica.c.rosmanlafever@boeing.com, ronald.c.provine@boeing.com, david.shema@boeing.com

Abstract

Boeing has hundreds of millions of pages of process specifications, design specifications, maintenance documentation, internal control documents, and standards. Much of the information contained in these documents is in the form of tables, technical illustrations and other formats with recognizable layout characteristics. More and more, Boeing and the aerospace industry depend upon the correct interpretation of the information contained in these documents and rely on systems that aid in human understanding of that information. This paper discusses several specific examples where automatic interpretation and understanding of documents is critical to efficient and accurate aircraft manufacturing and maintenance

1. Introduction

The success of the Boeing Company depends in large part upon millions of documents that are critical in all aspects of its operations. The design, manufacture and support of aerospace vehicles is an enormous undertaking controlled by process specifications and engineering standards that must be scrupulously adhered to. The designs themselves become central documents in the manufacturing process and determine the content of the millions of pages of maintenance documentation. Figure 1, which shows only *part* of the maintenance documentation for a single 747 aircraft, illustrates the scale of the documentation problems confronting Boeing.

Within these documents are many types of layouts that are suitable for *document layout analysis*, including tables, parts illustrations, schematics, multi-segment diagrams, and decision trees.

The purpose here is to touch on some of the potential applications of document layout analysis, and is certainly **not** intended to provide an exhaustive list of such applications. We hope this will spark additional research interest in this area.

Fig. 1: *Part* of the maintenance documentation for a single 747

2. Tables

Tabular information is a standard way to display technical information in documents. Many of these tables are highly complex and difficult to work with. It is common for columns or rows of a table to have relationships with other columns or rows of a table. It is also common to see tables within tables or tables embedded within drawings of parts or equipment. Without electronic enhancements, users spend too much time and make too many mistakes trying to understand the information contained in a table.

For example, the wiring processes for a commercial airplane must conform to the **Standard Wiring Practices Manual (SWPM)** [1]. The SWPM specifies the correct wire types, connector materials, tools and process steps for every wiring connection across the Boeing fleet. Some of the tables are massive in scope. The Wire Type Code table shown in **Figure 2** has 1073 rows. Finding the desired information in a table this size is a daunting task.

TABLE I
WIRE TYPE CODES

WTC	Model: 7( )7					Wire Part Number or Specification	#	Notes	R WTC
	2	3	4	5	6	7				
01	2	3	-	-	-	-	BMS 13-08 Type I Class A	01	High Temperature	39
08	2	3	-	-	-	-	RG8A/U, MIL-C-17D	01	Coax	10
0A	-	-	-	-	-	7	BMS 13-48 Type 15 Class 1	01	Shielded	-

Figure 2: The beginning of the Wire Type Code table – over 1000 rows follow

Tables are often embedded in engineering drawings with links to reference designators and reference marks elsewhere on the page. Users need systems that understand the relationships among these objects. **Figure 3** shows an equipment rack. Descriptions of each component in the rack are provided in the table in the center of the drawing. A system that understood this drawing's semantics could make such a drawing much more useful to the end user.

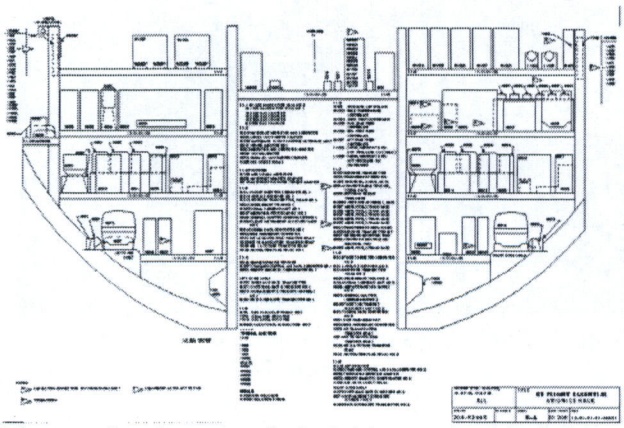

Figure 3: An unruled table embedded in a drawing

3. Technical Drawing Segmentation

Technical drawings may combine related information sources in different formats on a single page. To understand the content of such drawings, it is necessary to recognize that there are different sections or segments to the drawing. For example, in Figure 3 there is a flagnote legend in the lower-left hand corner that is referenced in dozens of places on the drawing using flagnote symbols. There is a complex title block in the lower right-hand corner that has valuable information about when the drawing was authored, what revision of the drawing this is and what aircraft this drawing is applicable to. End systems need to know about the different sections of a drawing and understand them and how they relate to each other.

In **Figure 4**, we see a typical electrical schematic. It is comprised of three main sections. On the right, there is a block of diagnostic messages. In the center is a high-level schematic view of the functional logic of the system. On the left is detailed wiring information. To make this type of drawing more useful to the end user, the system needs to know what the different parts of the drawing are.

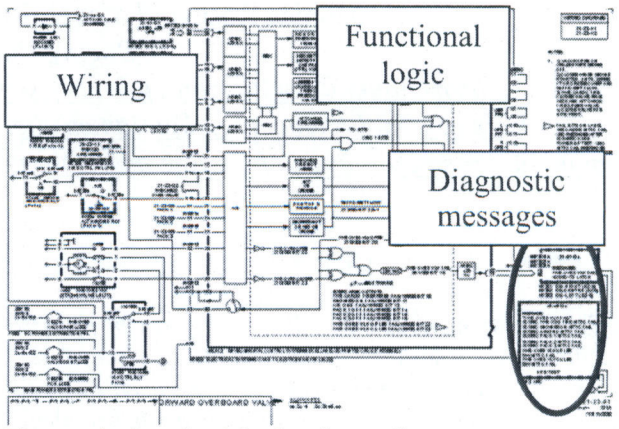

Figure 4: An electrical schematic

Another type of drawing that requires segmentation is a fault tree as shown in **Figure 5**. This figure is from a Fault Isolation Manual and consists of blocks containing diagnostic questions with arrows directing you to the next box based on the answer to the question. Systems need to understand the fault diagnostic logic implicit in these diagrams. In [2], we discussed our approach to this type of fault isolation diagram. We have achieved recognition accuracy above 99% and the software is now in production for Boeing's Portable Maintenance Aid product.

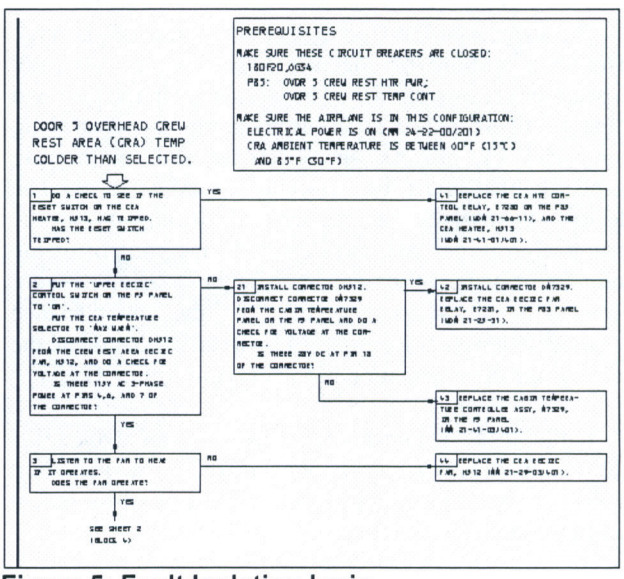

Figure 5: Fault Isolation logic

Figure 6 shows an example of a parts page. Parts pages present additional layout recognition challenges. These diagrams are used to drill down from top-level

assemblies to lower-level assemblies via 'bubbles' that are labeled. Each labeled bubble shows a single assembly. Accordingly, there is a tremendous need for layout recognizers that can accurately segment the drawings into their individual assemblies and automatically set up hyper-links among the assemblies of different levels.

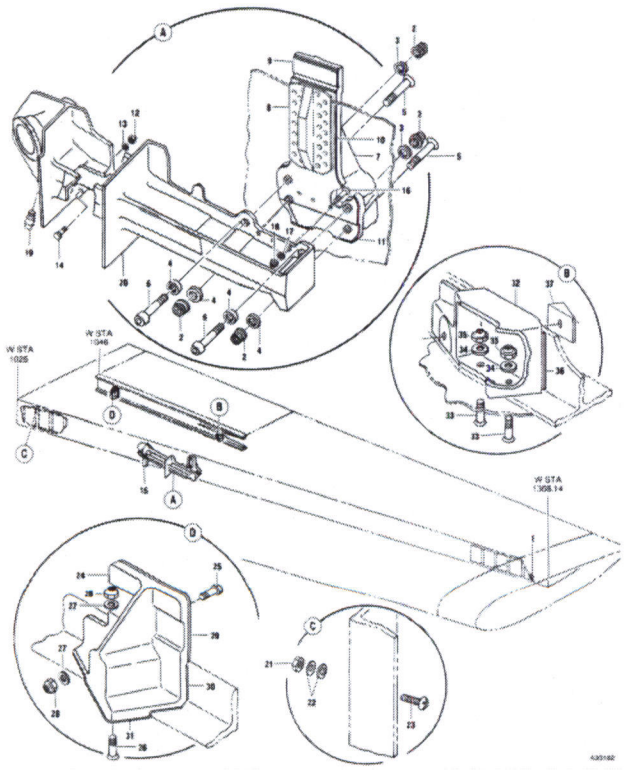

Figure 6: Parts pages require segmentation

In [2], we discussed our research on recognizing the layout of parts pages in vector format. Applying geometric proximity algorithms we achieved accuracies on the order of 84% for grouping art into the proper segments. Incorporating heuristics regarding the location of detail labels, the recognition of leader lines and attaching text to those leader lines increased our layout recognition accuracy to ~97%. Failures generally occur when parts of a detail are deliberately offset from the main body to show separation within the detail or when there is considerable space between the body of the detail and an orientation arrow (**Figure 7**). To achieve accuracies much closer to 100% will require recognition of the objects being presented, re-mapping the technical illustration to the real-world object.

4. Raster vs. vector considerations

Layout recognition research has predominantly focused on the understanding of the layout of *raster* images. While some of the examples we have discussed do occur as

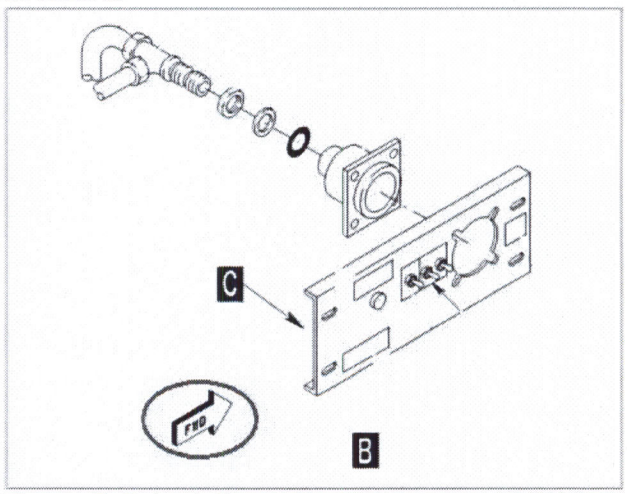

Figure 7: Orientation arrows can cause segmentation errors

raster drawings, most are vector images produced in modern illustration tools. Vector graphics are generally easier to analyze than raster – there is no need for optical character recognition, for example. However vector graphics present unique challenges that are not present in raster images and require a very different approach. For example, techniques involving histograms or ink density translate poorly, if at all, to vector diagrams. We need new recognition algorithms based on the rendering characteristics of vector graphics. This is a fertile area for new research.

5. Summary

Aerospace companies depend heavily on a wide variety of technical documentation in tremendous quantities. We have presented just a few of the types of layout recognition challenges found in technical documentation. Solutions must be highly scalable and extremely accurate. We hope to inspire new research in this relatively unexplored area of document recognition.

6. References

[1] 707, 727-777 Standard Wiring Practices Manual, D6-54446, The Boeing Company

[2] L.S. Baum, J.H.Boose, and R.J. Kelley, "Graphics Recognition for a Large-Scale Airplane Information System", *Graphics Recognition, Algorithms and Systems, Second International Workshop, GREC'97*, Springer, Nancy, France, August 1997, pp. 291-301.

Poorly Structured Handwritten Documents Segmentation using Continuous Probabilistic Feature Grammars

T. Artières
LIP6, Université Paris 6,
8 rue du Capitaine Scott, 75015, France
Thierry.Artieres@lip6.fr
Tel : (33)-1-44-27-72-20 Fax : (33)-1-44-27-70-00

Abstract

This work deals with poorly structured handwritten documents segmentation such as pages of handwritten notes produced with pen-based interfaces. We propose to use a formalism, based on Probabilistic Feature Grammars, that exhibit some interesting features. It allows handling ambiguities and to taking into account contextual information such as spatial relations between objects in the page.

KEYWORDS: Segmentation, Poorly structured documents, Note-taking, Probabilistic Feature Grammars.

1. Introduction

A number of new mobile terminals have appeared in the last few years, e-books, note-taking devices, portfolios. More recently, new terminals have been commercialized that are a mix between classical wearable computers and digital tablets. These objects, being most often mobile devices, have come with pen-based interfaces, allowing the user to write, store, edit, manage handwritten documents, and in a limited way to recognize these.

Documents produced using pen-based interfaces may be very hetreogeneous both in their structure (e.g. handwritten notes, maps etc) and in their content that may include text, drawings, images, equations etc. Such documents cannot be efficiently used in their rough form of electronic ink; one needs a higher level representation both for their structure (identification of lines, drawings,...) and for their content (e.g. partial recognition of textual regions).

Up to now, segmentation of written documents (e.g. pages) has been most often studied for off-line documents. Lots of works has been done in this context and various methods have been developed for e.g. newspapers and table segmentation, see in [2, 6, 9, 10, 11] for popular techniques. Besides, the case of on-line documents has been investigated very recently [5, 12]. However, existing segmentation techniques tuned for off-line documents are not well adapted to on-line documents. Actually, on-line document segmentation and off-line document segmentation do not deal with the same kind of documents. Off-line document segmentation deals with documents with a complex structure (e.g. newspaper) but with an homogeneous local structure (e.g. purely horizontal and parallel text lines, uniform inter-line distance etc). At the opposite, on-line handwritten documents produced with pen-based interfaces (using a mobile device) are often much simpler in their structure (handwritten notes in a single column form ...) but also much more heterogeneous. In a same page, lines may be differently slanted; characters size may vary in a same line etc. The main difficulty lies then in the inherent ambiguity and heterogeneous of such documents.

As a consequence methods developed for off-line documents segmentation relying on global features of the documents (uniform slant of the lines,...) are adapted to well structured and homogeneous documents. In this study, we investigate the development of generic segmentation tools for much less structured documents as those typically acquired through pen-based interfaces with note-taking devices. This is a prospective work and, up to now, no strict evaluation has been conducted.

The paper is organized as follows. In section §2, we discuss the choice of a generic model to express handwritten documents structure and present Probabislitic Feature Grammars (PFGs), which is the formalism we built on. In section §3, we detail how we have extended and adapted PFGs to deal with 2D handwritten documents. In §4, we discuss the use of such models for on-line documents segmentation.

2. Document structure model

2.1. Dealing with on-line documents

A few previous works have been published on the segmentation and recognition of poorly structured handwritten documents [5, 9, 12]. Most often, techniques operate in two steps. First, the page is segmented into text and non text; then text areas are segmented into lines, lines into words etc. This is done using bottom-up or top-down approaches (for example with histogram projection techniques to detect lines and then to detect words in lines). In a second step, segmented elements (i.e. word) are recognized using an isolated word recognition engine.

In our opinion, the main drawback of these techniques lies in that the segmentation process is performed using global features. This is damageable in the context of on-line handwritten notes where contextual information is essential for disambiguation. Also, none of these studies provide a generic tool for dealing with various documents structure; these are rather ad-hoc systems. Our work is an attempt to overcome these limitations. We aim at providing a generic formalism for segmentation of handwritten pages taking into account contextual information.

For dealing with spatial relationships, a number of techniques have been proposed in the literature, visual grammars, 2D Hidden Markov Models, or grammars that include spatial operators [1, 3, 4, 7, 13, 14]. However, there is no general agreement today on the best formalism for dealing with two-dimensional documents. Following the idea in β], we choose to extend a grammar formalism originally designed to deal with one-dimensional data (sentences in language modeling), and to integrate the 2-dimensional feature of our data (handwritten pages) through the use of spatial operators. There are two main reasons for doing so. First, grammars are a kind of intuitive description language. Then, to deal with particularly structured documents one needs only changing some rules of the grammar while the remaining of the program remains the same. The second reason lies in that, at the end, an on-line handwritten document is somehow more one-dimensional (following more or less the writing/reading order) than 2-dimensional. We investigated a formalism named Probabilistic Feature Grammars (PFGs) [4]. PFGs are a probabilistic framework that allow taking into account contextual information during the parsing process, at the opposite of more classical Probabilistic Context Free Grammars [13].

2.2. Formalism of PFGs

This section is much inspired from Goodman's paper [4]. In PFGs, the integration of contextual information consists in propagating information about the way non terminal are instantiated. Rules are of the form:

$$A:(a_1,\ldots a_g)\rightarrow B:(b_1,\ldots b_g),C:(c_1,\ldots c_g) \quad (1)$$

where A, B and C are terms (non terminals or terminals), $(a_1,a_2,\ldots,a_g)$, $(b_1,b_2,\ldots,b_g)$ and $(c_1,c_2,\ldots,c_g)$ are vectors of g features associated to terms. In our case, terms in the grammar are constitutive parts of a handwritten page: paragraph, line, word, etc. We note $C(X)$ the feature vector associated to a term X, features are assumed discrete here. Feature vectors are considered as random variables, this allows associating a derivation probability to a rule. For example the probability for using rule (1) is:

$$P(C(B),C(C)/C(A)) \quad (2)$$

PFGs may be viewed as probabilistic generative models. Based on the knowledge of features for A, one derivates terms B and C and their feature vectors with a probability computed with (2). This probability may be rewritten:

$$P(C(B)/C(A),C(C)).P(C(C)/C(A)) \quad (3)$$

To make computation tractable, [4] assumes independent features so that (3) is computed through:

$$\prod_{i=1}^{g} P(b_i/a_i,c_i).\prod_{i=1}^{g} P(c_i/a_i) \quad (4)$$

In the discrete features case described in [4], one can approximate probabilities $P(c_i/a_i)$ and $P(b_i/a_i,c_i)$ with bigrams and trigrams probabilities estimated with a training corpus. The algorithms used by Goodman, to compute the probability of an input data sequence, or to compute the derivation tree with maximum probability, are variations of the CKY algorithm in [8].

3. PFGs for handwritten documents

We describe now continuous PFGs (i.e. with continuously valued features) for the segmentation of 2D on-line handwritten documents. We investigated the definition of a set of elementary rules and operators from which one can define various grammars corresponding to different document structures. We detail here a simple grammar suitable for simple handwritten texts. We then describe the features that are propagated during the parsing and the probability laws associated with the rules. Then, we discuss the decoding algorithm.

3.1. A Grammar for simple handwritten texts

In this preliminary study, we restricted ourselves to simple grammars although the formalism could be used for a wide variety of grammars. Below is an example of a grammar for simple handwritten texts:

R1. Page → Section
R2. Section → Paragraph [Above] Section
R3. Section → Paragraph
R4. Paragraph → Line [Above] Paragraph
R5. Paragraph → Line
R6. Line → word [On the left of] Line
R7. Line → <u>word</u>

This grammar defines a page as a series of paragraphs that themselves consist in series of lines, that themselves consist in series of words. We use the terminology word for any stroke written without pen-up movement. The operators [*Above*] and [*On the left*] are spatial operators that are used to define rule probabilities.

3.2. Feature vectors

Our grammars are based on a hierarchy of terms. For example, in the grammar described above, a paragraph consists in a series of lines, a line consists in a series of words etc. This led us to define for each instantiated term X in the grammar three subsets of features:

- *ownf(X) =(ulc, w, h, s)*: The own features of X. These features are: the position of X (x and y coordinates of the upper left corner, *ulc*, of X), its dimensions (width w and height h), its slant s.
- *cef(X)=(ulce, wce, hce, sce)*: the average own features *ownf* of the composite entities of X (e.g. average own features for the lines that compose a paragraph).
- *srce(X)=(dce)*: The average spatial relation between the composite entities of X: *dce*. It is the average 2D displacement between the composite entities of X (e.g. average inter-line in a paragraph).

A feature vector is associated to each instantiated term of the grammar, and consists in a triplet:

$$C(X)=(ownf(X), cef(X), srce(X)) \quad (5)$$

3.3. Rule Probabilities

The probability to activate a rule is computed based on the features of the terms that appear in the rule. We identified a few typical rule families, depending upon the *levels* (in the hierarchy of terms) of the terms involved in the rule. For each family, we have defined the rule probability form. For example, the form of rule R6 is:

$A \rightarrow B\ op\ C$

where *op* is a spatial operator (*On the left*, *Above*, etc), C is a term that has the same level as A and B is a term with a level immediately below that of A. The probability associated to such a rule is rewritten as in (3):

$$P(C(B)/C(A),C(C)) * P(C(C)/C(A)) \quad (6)$$

The first part of right member may be defined as:

$$P(C(B)/C(A),C(C)) = P_{op}(ownf(B)/cef(C),srce(C)) \quad (7)$$

Let illustrate this with rule *R6*; Eq (7) says that the probability of a word B added to an existing line C is computed as the probability of the *own* features of the word (its position, height, slant etc) conditionally to the average height, slant ... of words in C, and to average spatial relationship between words in C. This latter probability is computed according to the expected spatial relationship defined by the spatial operator *op*. All rule probabilities have been defined in a similar way according to their underlying semantic. To compute these probabilities, we assumed independence assumption between features. For example the probability in (7) is computed as:

$$P_{op}(ownf(B)/cef(C),srec(C)) =$$
$$P(w(B)/wce(C)) * P(h(B)/hce(C)).P(s(B)/sce(C)) *$$
$$P(ulc(B)/ulc(C),srce(C),op) \quad (8)$$

where probability laws for each feature have been assumed Gaussian laws. For now, the parameters of these Gaussian laws have been determined empirically.

Fig. 1 illustrates this. A partial line (C) has been derived and its extension (by aggregating another word, B) is investigated (Rule R6). To do this, the extended line (A) is built (its features are computed), and its probability is computed through equation (8), according to the features of A, B and C. Note that the second member of equation (6), $P(C(C)/C(A))$, is not considered here since A features $C(A)$, are deterministically computed based on B and C features.

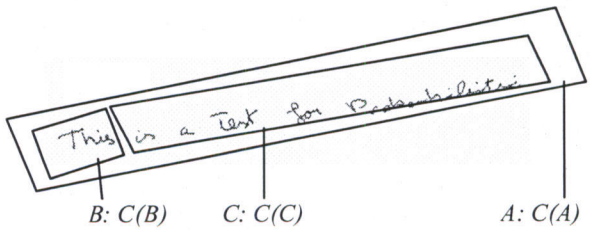

Figure 2. Example of rule R6.

3.4. Optimal derivation tree

Classical parsing algorithms could be adapted for our continuous probabilistic grammars, e.g.[13]. However, to limit combinatorial problems we used a simplified parsing algorithm by taking into account the *hierarchical* nature of our grammars. It processes the grammar level by level from the bottom. For the grammar in §3.1, first, all possible (most probable) lines are parsed. Then, from these possible lines, most probable paragraphs are determined etc until whole page segmentation hypotheses are obtained. This algorithm allows easier integration of beam search strategy in order to prune less probable hypotheses.

4. Examples

We show in this section an example of segmentation of a handwritten page. It is a very simple one but it helps to understand how contextual information may be useful and even necessary to deal with poorly structured documents. We do not provide quantitative experimental results on a collection of documents since we did not perform such evaluation yet, it is our objective to do so as soon as

possible. Figure 2.a shows a handwritten page with two paragraphs, the first one being slightly slanted, the second one not. When processing such a document, a global segmentation method would expect an uniform slant of all lines in the page, performing histogram projection to detect lines and would fail at identifying these lines. Furthermore, even if the lines were correctly identified, it would not be so clear that the first three lines are a paragraph and the two other lines are a second paragraph. These two paragraphs appear to us as two distinct paragraphs because they are, alone but not together, homogeneous aggregation of lines.

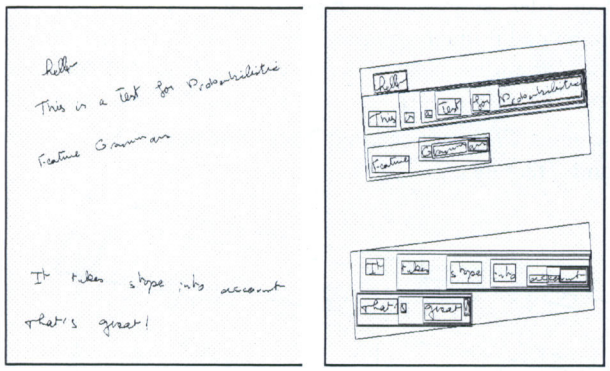

Figure 2. Example of a handwritten page (left) and its most probable segmentation according to our system (right).

Figure 2.b shows the result of the most probable segmentation of this page with our system. For illustration, we added the bounding boxes of all entities that have been instantiated in this segmentation. One can see the incremental construction of the lines (self-including bounding boxes of words) corresponding to the iteration of rule 6, as well as the incremental construction of the paragraphs (self-including bounding boxes of lines). As may be seen the two paragraphs have been correctly identified and these have been identified as two different paragraphs because of the propagation of contextual information. Since the average slant of the lines of the first paragraph, together with the average inter-line distance between the lines of the first paragraph do not correspond to the fourth line, a new paragraph is initiated that begins with the fourth line.

5. Conclusion

We presented a preliminary study for the segmentation of poorly structured handwritten documents as typically produced with pen-based interfaces. We adapted and extended Probabilistic Feature Grammars in order to deal with 2D data. Our approach presents some interesting features. Being a probabilistic formalism, it allows to deal with ambiguities, and the decoding algorithm outputs a number of most probable segmentations. Second, the use of contextual information makes possible the segmentation of poorly and ambiguously structured documents where traditional global methods would fail. We believe that the first results of our prospective work is promising, we are currently working at a extensive evaluation of the system.

6. References

1. Abney S., *Stochastic Attribute-Value Grammars*, Computational Linguistics, 23 (4), pages 597-617, 1997.
2. Aiello M., Monz C., Todoran L., Worring M., *Document understanding for a broad class of documents*, International Journal on Document Anlysis and Recognition (IJDAR), Vol. 5.1, 2002.
3. Couasnon B., Brisset P., and Stephan I., *Using logic programming languages for optical music recognition*, in International Conference on the Practical Application of Prolog, pages 115-134, Paris, France, April 1995.
4. Goodman J., *Probabilistic feature grammars,* International Workshop on Parsing Technologies, 1997.
5. Jain K., Namboodiri A., Subrahmonia J., *Structure in on-line documents*, ICDAR, 2001.
6. Kise K., Sato A., and Iwata M., *Segmentation of page images using the area Voronoi diagram*, Computer Vision and Image Understanding, 70, pages 370-382, 1998.
7. Kosmala A., Rigoll G., Lavirotte S., and Pottier L., *On-line handwritten formula recognition using hidden Markov models and context dependent graph grammars*, ICDAR'99.
8. Lari K, Young S., The estimation of stochastic context-free grammars using the inside-outside algorithm, Computer Speech and Language, 4, pages 35-56, 1990.
9. Marti U., Bunke H., *Text line segmentation and word recognition in a system for general writer independent handwriting recognition*, ICDAR, 2001.
10. Nagy G., Seth S., *Hierarchical representation of optically scanned documents*, ICPR, 1984.
11. O'Gorman L., *The document spectrum for page layout analysis*, IEEE Trans. PAMI, Vol. 15, 1993.
12. Ratzlaff E., *Inter-line distance estimation and text line extraction for unconstrained online handwriting*, IWFHR, 2000.
13. Stolcke A., *An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities*, Computational Linguistics, Vol. 11, no2, 1995.
14. Tokuyasu T.A., *Applications of the turbo recognition approach to layout analysis*, DLIA, 2003.

Mining spatial association rules from document layout structures

Margherita Berardi Michelangelo Ceci Donato Malerba
Dipartimento di Informatica – Università degli Studi di Bari
via Orabona 4 - 70126 Bari
{berardi, ceci, malerba}@di.uniba.it

Abstract

In this paper we investigate the discovery of spatial association rules from a particular kind of images, namely document images. Document images are initially processed to extract both their layout structures and their logical structures. To take into account the inherent spatial nature of the layout structure, a spatial data mining algorithm is applied, which returns spatial association rules. We present possible applications of spatial association rules detected from document layout. We also illustrate and comment experimental results on a set of multi-page documents extracted by IEEE PAMI.

1. Introduction

The discovery of association rules has attracted a great deal of attention in data mining research. Association rules are a class of regularities introduced by [1] that can be expressed by an implication:

$$X \rightarrow Y$$

where X and Y are a sets of *items*, such that $X \cap Y = \emptyset$. The meaning of such rules is quite intuitive: Given a database D be of transactions, where each *transaction* $T \in D$ is a set of items, $X \rightarrow Y$ expresses that whenever a transaction T contains X than T probably contains Y also. The conjunction $X \wedge Y$ is called *pattern*.

Two parameters are usually reported for association rules, namely the *support*, which estimates the probability $p(X \subseteq T \wedge Y \subseteq T)$, and the *confidence*, which estimates the probability $p(Y \subseteq T \mid X \subseteq T)$. The goal of association rule mining is to find all the rules with support and confidence exceeding user specified thresholds, henceforth called *minsup* and *minconf* respectively. A pattern $X \wedge Y$ is *large* (or *frequent*) if its support is greater than or equal to *minsup*. An association rule $X \rightarrow Y$ is *strong* if it has a large support (i.e. $X \wedge Y$ is large) and high confidence.

The traditional application of association rules is in the business world, where they are used to take more precise marketing actions on the basis of what products are frequently bought together. In this application, the items are products and the transactions are customer purchases at the checkout. However, it is becoming clear that association rules are not restricted to market basket analysis, but can be successfully applied to a wide range of domains, such as web access patterns discovery [3] and building intrusion detection models [6].

An interesting application is faced in the work by Ordonez and Omiecinski [10] where a method for mining knowledge from images is proposed. The method is an association rule miner that automatically identifies similarities in images on the basis of their content. The content is expressed in terms of objects automatically recognized in a segmented image. The work shows that even without domain knowledge it is possible to automatically extract some reliable knowledge. Mined association rules refer to the presence/absence of an object in an image, since images are viewed as transactions while objects as items. No spatial relationship between objects in the same image is considered.

In this paper we investigate the discovery of association rules from a particular kind of images, namely document images. Document images are initially processed to extract their layout structures, which describe the geometrical arrangement of content portions on a page. Then the logical structures are extracted. The logical structure of a document image consists of a hierarchy of segments of the document, each of which corresponds to a visually distinguished semantic component of the document (e.g., title, paragraph, caption or heading) [12]. Some layout components with no visually distinguished semantic are labelled as *undefined*. The extraction of both layout and logical structures is performed by means of the system WISDOM++ [2], whose main characteristic is the application of machine learning algorithms in several document processing steps. In WISDOM++ documents are grouped into classes (e.g. paper published on IEEE Transactions of Pattern Analysis and Machine Intelligence), such that document images in the same class show approximately the same layout/logical structure.

The discovery of association rules follows the processes of layout structure extraction (layout analysis)

and logical structure extraction (document image understanding). We are interested in association rules expressing regularities among logical components of a set of document images belonging to the same class.

2. The approach

Differently from the work by Ordonez and Omiecinski [10], we also intend to take into account the inherent spatial nature of the layout structure, that is, we intend to discover, if any, spatial patterns between logical components. Therefore, association rule mining methods developed in the context of spatial data mining are considered [9].

There are three main peculiarities of the proposed approach. First, the spatial property of logical components is considered. Logical components are described in terms of their content type (e.g., text, graphics, etc.), their logical meaning (e.g., title, authors, etc.) and their geometrical features, which can be either relational or attributional. Relational features relate two logical components on the basis of their mutual position in the document (e.g. *on_top(A,B)*, *to_right(A,B)*). Attributional features refer to geometrical properties of layout components, such as width and height, as well as locational properties (position along x/y axis).

Second, the hierarchical structure of logical components is considered as well. It is possible to look at the set of logical components of a document image as a hierarchy where each single logical component is related to another one by a *is_a* or a *part_of* relation (e.g *title* is part_of *identification*, *page_number* is_a *page_component*). The levels in the hierarchy are called granularity levels.

Third, the logical components can play different roles. Indeed, in spatial data mining attributes of some spatial objects in the neighborhood of, or contained in, a unit of analysis[1] may affect the values taken by attributes of the unit of analysis. Therefore, it is necessary to distinguish units of analysis, which are the *reference* objects of an analysis, from other *task-relevant* spatial objects, and it is important to represent interactions between them. In our application, some logical components play the role of *reference objects* while other logical components play the role of *task relevant objects*.

To mine spatial association rules we use SPADA (Spatial Pattern Discovery Algorithm) [9] which is based on a multi-relational data mining approach and permits the extraction of multi-level spatial association rules, that is, association rules involving spatial objects at different granularity levels. An example of association rule discovered by SPADA is:

is_a(A,running_head) →
on_top(A,B) , *is_a(B,content)* ,*type_text(A)*
support: 90.9% confidence: 90.9%

This rule means that if a logical component (*A*) is a *running_head* then it is textual and it is on top of another layout component (*B*) which is a component of type *content*. This rule has a high support and a high confidence (both expressed as percentages).

3. Using association rules

The extracted association rules can be used in a number of ways. First, new documents can be recognized as satisfying the constraints that define the domain template (document classification and retrieval). Indeed, recent approaches propose to use discovered association rules for classification tasks [7].

Second, the rules can be profitably used in the automatic layout correction. Currently, in WISDOM++ the automatic correction of the layout is performed by means of a set of rules learned from the sequence of user actions [8]. By formulating the problem as a planning problem, it is necessary to define both a goal and a metric evaluating the distance between the current state (layout structure) and the goal. This metric can be based on the number of association rules supported by the extracted layout structure.

Third, the rules can be also used in a generative way. For instance, if a part of the document is hidden or missing, strong spatial association rules can be used to predict the location of missing layout/logical components [5]. Moreover, a desirable property of a system that automatically generates textual documents is to take into account the layout specification during the generation process, since layout and wording generally interact [11]. Spatial association rules can be useful to define the layout specifications of such a system. Finally, this problem is also related to document reformatting [4].

4. Mining spatial association rules

The problem of mining association rules by means of SPADA can be formally stated as follows:

Given

- a set of descriptions of the labelled documents (result of the document understanding)
- a set of reference objects S,
- some sets R_k, $1 \leq k \leq m$, of task-relevant objects,

[1] The unit of analysis is the basic entity or object about which generalizations are to be made based on an analysis and for which data are collected in the form of variables.

- a background knowledge *BK* including some spatial hierarchies H_k on objects in R_k and a domain specific knowledge,
- *M* granularity levels in the descriptions (1 is the highest while M is the lowest),
- a set of granularity assignments ψ_k which associate each object in H_k with a granularity level,
- a couple of thresholds *minsup*[*l*] and *minconf*[*l*] for each granularity level,
- a declarative bias *DB* that constrains the search space,

Find

strong multi-level spatial association rules.

The set of descriptions of the labelled documents is expressed in the form of first-order logic conditions. The use of the first order logic is necessary because the feature-vector representation, typically adopted in statistical approaches, cannot render the relational features.

Spatial features (relations and attributes) are used to describe the logical structure of a document image. In particular, we mention *locational* features such as the coordinates of the centroid of a logical component (*x_pos_center*, *y_pos_center*), *geometrical* features such as the dimensions of a logical component (*width*, *height*), and *topological* features such as relations between two components (*on_top*, *to_right*, *alignment*). We use the *non-spatial* feature *type_of* that specifies the content type of a logical component (e.g. *image*, *text*, *horizontal line*). In addition there are other non-spatial features, called *logical* features which define the label associated to the logical components. They are: *affiliation, page_number, figure,caption, index_term, running_head, author, title, abstract, formulae, subsection_title, section_title, biografy, references, paragraph, table, undefined*. In the following we present an example of document description on which run SPADA:

```
class(h,tpami),
is_a(a,running_head), is_a(b,title),...
page(h,first),
part_of(h,a),part_of(h,b)=true,...
width(a,390),width(b,490),...
height(a,7),height(b,54),...
type_text(a),type_text(b),...
x_pos_centre(a,215),x_pos_centre(b,288),...
y_pos_centre(a,26),y_pos_centre(b,83),...
on_top(a,b),on_top(b,c),on_top(b,d),...
to_right(e,f),...
only_left_col(a,1),only_left_col(b,e),
only_right_col(b,e),only_middle_col(b,e),
...
```

where *h* represents the page and *a,...,g* represent the logical components of the page. It is noteworthy that the features *class* and *page* are used to describe the class and the order page and the relation *part_of* is used to express the membership of a component to a page. Numerical features are automatically discretized before inferring spatial association rules.

The specification, by means of a set of rules, of the following domain specific knowledge:

```
at_page(X,first) <- part_of(Y,X),
page(Y,first)
at_page(X,intermediate) <-
        part_of(Y,X),page(Y,intermediate)
at_page(X,last_but_one) <-
       part_of(Y,X), page(Y,last_but_one)
at_page(X,last)        <-        part_of(Y,X),
page(Y,last)
```

permits to automatically associate information on page order to layout components. Since the presence of some logical components may depend on the page order (e.g. *author* is in the first page), the above rules allows SPADA to discover associations where this information is taken into account. The specification of the hierarchy (Figure 1) allows the system to extract spatial association rules at different granularity size.

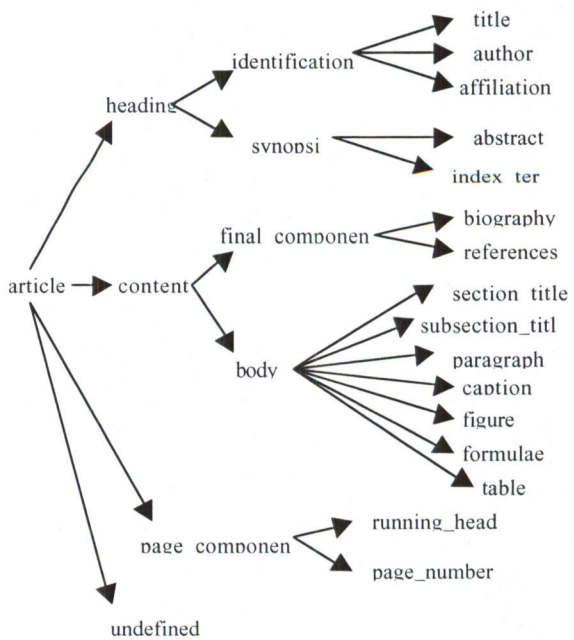

Figure 1. Hierarchy of logical components.

The declarative bias *DB* constrains the search space and aims at defining the *reference objects* (*ro*) and *task relevant objects* (*tro*). In our task, the *ro* are the logical components to which at least one satisfied logical feature, different from *undefined*, is associated. The *tro* are all the logical components.

5. Experimental results

To investigate the applicability of the proposed solution we considered six papers, published as either regular or short, in the IEEE Transactions on Pattern Analysis and Machine Intelligence, in the January and February 1996 issues. Each paper is a multi-page document and has a variable number of pages and layout components for page. A user of WISDOM++ labels some layout components of this set of documents according to their logical meaning. Those layout components with no clear logical meaning are labelled as *undefined*. All logical labels belong to the lowest level of the hierarchy reported in the previous section. We processed 54 document images in all.

In Table 1 logical components distribution on the processed documents is shown.

Table 1. Labels distribution.

Document ID / Label	*1*	*3*	*4*	*6*	*7*	*9*	*Total*
affiliation	1	1	0	1	2	2	7
page_number	8	13	12	1	5	5	44
figure	19	13	12	3	12	12	71
caption	13	17	7	3	11	5	56
index_term	1	1	1	0	0	0	3
running_head	14	15	14	1	6	5	55
Author	1	1	2	1	1	1	7
Title	1	1	1	1	1	1	6
abstract	1	1	1	1	1	1	6
formulae	24	19	21	0	4	5	73
subsection_title	3	1	1	0	0	0	5
section_title	4	4	2	0	1	1	12
biografy	2	1	1	0	0	0	4
references	3	3	2	1	1	2	12
paragraph	54	55	50	3	19	21	202
table	0	9	1	0	2	1	13
undefined	21	26	27	10	14	16	114

The number of features to describe the six documents presented to SPADA is 17,880, about 331 features for each page document. The total number of logical components is 690 (114 of which are *undefined*) about 318 descriptors for each page document.

An example of association rule discovered by SPADA is:

is_a(A,author) → only_middle_col(A,B) ,
is_a(B,heading), height(B,[1..174]), type_text(A)
support: 85.71% confidence: 85.71%

The spatial pattern of this rule involves six out of seven (i.e. 85.71%) blocks labelled as authors. This means that six logical components which represent the author of some paper are textual components vertically centered with a logical component B at the heading of the paper, with height between 1 and 174.

At a lower granularity level, a similar rule is found where the logical component B is specialized as abstract:

is_a(A,author) → only_middle_col (A,B),
is_a(B,abstract), height(B,[1..174]), type_text(A)
support: 85.71% confidence: 85.71%

The rule has the same confidence and support reported for the rule inferred at the first granularity level.

Conclusions

This work presents an application of spatial data mining techniques to the problem of finding associations between logical components extracted from document images by means of document analysis and understanding methods. As future work, we intend to investigate the application of mined association rules in three different contexts, namely document classification and retrieval, automated layout correction, and automated generation of documents.

Acknowledgements

This work fulfills the research objectives set by the IST-1999-20882 project COLLATE (Collaboratory for Annotation, Indexing and Retrieval of Digitized Historical Archive Material) funded by the European Union.

References

1. R. Agrawal, and R. Srikant, Fast Algorithms for Mining Association Rules.*Proc.20th Int.Conf.Very Large Data Bases*, VLDB, 1994
2. O. Altamura, F. Esposito, and D. Malerba, Transforming paper documents into XML format with WISDOM++, *Int. Journal on Document Analysis and Recognition*, 4(1), 2-17, 2001.
3. M. S. Chen, J. S. Park, and P. S. Yu, Data Mining for Path Traversal Patterns in a Web Environment, *In Proceedings of the 16th International Conference on Distributed Computing Systems*, IEEE press, in 385-392, 1996.
4. L. Hardman, L. Rutledge, and D. Bulterman, Automated generation of hypermedia presentation from pre-existing tagged media objects, *Proc. Of the 2nd. Workshop on Adaptive Hypertext and Hypermedia*, 1998.

5. K. Hiraki, J.H. Gennari, Y. Yamamoto, and Y. Anzai, Learning Spatial Relations from Images, *Machine Learning Workshop,* Chicago, pages 407—411, 1991.
6. W. Lee, S. Stolfo, and K. Mok, Mining audit data to build intrusion detection models. *In KDD-98*, Agrawal, Stolorz, and PiatetskyShapiro, Eds., AAAI Press, pp. 66—72, 1998.
7. B. Liu, W. Hsu, and Y. Ma., Integrating classification and association rule mining. *In KDD'98*, New York, NY, 1998.
8. D. Malerba, F. Esposito, O. Altamura, M. Ceci, and M. Berardi, Correcting the Document Layout: A Machine Learning Approach, ICDAR 2003.
9. D. Malerba, and F.A. Lisi, Discovering Associations Between Spatial Objects: An ILP Application, in C. Rouveirol & M. Sebag (Eds.), *Inductive Logic Programming*, Lecture Notes in Artificial Intelligence, 2170, Springer, Berlin, 2001.
10. C. Ordonez, and E. Omiecinski, Discovering association rules based on image content, *Proceedings of the IEEE Advances in Digital Libraries Conference 99.*
11. K. Reichenberger, K. J. Rondhuis, J. Kleinz, and J. Bateman, Effective Presentation of Information Through Page Layout: a Linguistically-Based Approach. *Proceedings of the ACM Workshop on Effective Abstractions in Multimedia.* San Francisco, California, 1995.
12. K. Summers, Toward a taxonomy of logical document structures. *In Electronic Publishing and the Information Superhighway: Proceedings of the Dartmouth Institute for Advanced Graduate Studies (DAGS)*, pages 124--133, 1995.

Selection of Table Areas for Information Extraction

Ana Costa e Silva *
Banco de Portugal, Portugal
acsilva@bportugal.pt

Alípio Jorge
Universidade do Porto, Faculdade de Economia do Porto, LIACC, Portugal
amjorge@liacc.up.pt

Luís Torgo
Universidade do Porto, Faculdade de Economia do Porto, LIACC, Portugal
ltorgo@liacc.up.pt

Abstract

Information contained in companies' financial statements is valuable to support a variety of decisions. In such documents, much of the relevant information is contained in tables and is extracted mainly by hand. We propose a method that accomplishes a preliminary step of the task of automatically extracting information from tables in documents: selecting the lines which are likely to belong to the tables containing the information to be extracted. Our method has been developed by empirically analysing a set of Portuguese companies' financial statements, using statistical and data mining techniques. Empirical evaluation indicates that more than 99% of the table lines are selected after discarding at least 50% of them. The method can cope with the complexity of styles used in assembling information on paper and adapt its performance accordingly, thus maximizing its results.

1. Introduction

Companies worldwide have the legal obligation of producing and publishing, on at least a yearly basis, reports accounting for their activities. These reports, called financial statements, contain tables with information that supports the decisions of a variety of economic agents, for whom it is often necessary to combine information from several documents. However, such aggregate analysis requires the time consuming activity of capturing the data manually to a database.

Although it is expected that in the future these statements will be published in a predefined structured format such as XBRL (eXtensible Business Reporting Language, [9]), in many countries it will most likely take a long time before they are fully adopted.

To automatically extract information from the tables contained in financial statements, or in any document, five basic tasks have to be fulfilled [2]:

Location: "differentiating tables from other elements such as body text, heading, titles, bibliographies, line drawings and bitmap graphics", [7].

Segmentation: delimiting the table's physical structures -its columns and lines, its simple and spanning cells (those spreading over more than one column/ line).

Functional analysis: identifying the function each physical unit plays; there are two basic functions: containing data and describing data, any titles and footnotes should be identified as such.

Structural analysis: detecting the relationships between the cells, identifying the attribute-value threads which have to be read conjointly.

Interpretation: going one step beyond simply identifying relationships to knowing what they mean and deciding upon them. More than knowing the cells containing strings "Total assets" - "5000" - "Real" have to be read conjointly, interpreting means being able to affirm "Total assets are 5000 PTE".

The task we approach in this paper is a preliminary step of this process: given a text document, we want to automatically select groups of lines that are likely to belong to tables. This step reduces the search effort in the subsequent tasks. To serve as training examples, we downloaded from the Web 19 financial reports published by 13 Portuguese companies in a period ranging from 1997 to 2000. On the whole there were 87,843 lines, of which 70,584 are not table lines and 9,685 are part of the tables containing the information we wish to extract. We then proceeded to convert the original files to plain text files using the pdftotxt linux utility, and imported them to a database table. Each line in the report became a distinct record.

2. Selecting target areas

Current software to locate tables in documents is generally based on detecting the alignment of certain characters, either words, e.g. [3], non-space characters, e.g. [6], or white spaces, e.g. [5], being vertical alignment often the key feature searched for. Other methods, such as [4], examine the contents of each line in the document, character by character, to detect the presence of groups of contiguous spaces, count them, among other characteristics, and present the counts to a decision tree to determine whether or not the line belongs to a table.

*The opinions expressed in this article are the responsibility of the authors and not necessarily coincide with the Banco de Portugal's.

However, in order to classify an area as table, existing strategies take each line (either vertical or horizontal) and perform a character-by-character scan to detect the presence of the characteristics used as input. As such, all parts of the document are treated as having equal likelihood of containing tables.

2.1 The main criterion for table line location

To classify each line as being likely to belong to a table or not, our method basically relies on one feature – the total number of contiguous inner spaces in the line. To determine this, we first remove the leading and trailing spaces; we then obtain all the substrings of more than one consecutive space characters; the sum of the lengths of these substrings is the total number of contiguous inner spaces in the line. The effectiveness of this feature relies on the notion that a line having more inner spaces than what is "normal" will hold a distinctive graphical element (such as a table or a chart), which is worth inspecting more closely. A Chi-square test was able to prove the validity of this empirical observation: with significance 10^{-6} and on the basis of our sample, a statistical dependence was found between the number of inner spaces in a line and whether the lines belong to a table 1. Thus, our main criterion will be of the form

$$X_\ell > \lambda \rightarrow \ell \text{ is a } \textit{candidate table line} \qquad (1)$$

where X_ℓ is the number of contiguous inner spaces in line ℓ, and λ is a threshold to be determined.

2.2 Robustness of the criterion to different layouts

Having this established, we have to find an appropriate threshold to distinguish the two groups. Intuitively, one could foresee that the limit to distinguish table from non-table lines can be much smaller in a text written in one single column than in a text entirely written in two or more columns. To verify this observation, we applied the Kruskal Wallis test 0 and proved, with 10^{-6} significance and on the basis of this sample, that the distribution of inner spaces is statistically different within the reports in the sample. To determine whether a given pair of reports had contributed to this result, we compared each pair at a time and found that only 22,8% of the total number of possible pairs were similar. A Chi square test also revealed statistical evidence of independence between whether or not a line is in a table and the document it originates from. As such, the threshold to distinguish the two types of lines should be established differently according to its source document. In other words, we want the criterion to be generalized to

$$X_\ell > \lambda(D) \rightarrow \ell \text{ is a } \textit{candidate table line} \qquad (2)$$

where D is the document being analysed. If we label each line of a report as being in a table or not and count the number of inner spaces it has, we can determine the optimal threshold by using the J48 algorithm of the data mining suite Weka [8] to build a one-node decision tree, which will classify an unlabelled line according to the logical value of a test of the form "Number of contiguous inner spaces < Threshold". The value of that threshold corresponds to the most informative test.

We would like to be able to determine how this optimal threshold can be obtained for a document given its global features, without prior knowledge of which of its lines belong to tables. In Figure 1, we compare the thresholds established through the process described above with certain global features of their source reports. In the leftmost chart, the global feature is *the number of lines in the document*. As can be observed, documents with more lines tend to have higher thresholds, because their authors probably invested more resources to make them graphically more appealing, and used a more diversified style for accommodating elements on page. The rightmost chart shows that, for documents where the *most frequent number of positive inner spaces* (or mode) is high enough (but not too high), the optimal division tends to be increasingly higher. This happens because the mode occurs farther from zero when the text is organized in two or more columns, for which too small a limit would cause the misclassification of a considerable number of text lines.

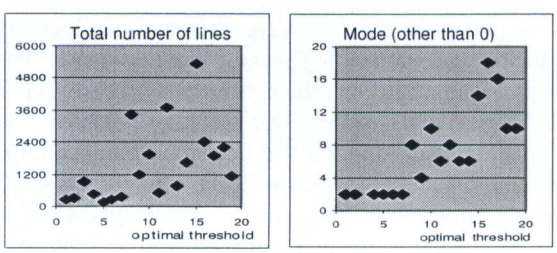

Figure 1. The optimal threshold *versus* characteristics of source documents: total number of lines (leftmost), mode (other than zero) (rightmost)

As such, it is possible to use measurable features of source document to estimate the best threshold. These reflect the different styles used by the authors to organize graphical elements on a page, thus the method can adapt to different document layouts. Alternatives for defining $\lambda(D)$ are given in Section 3.1. All lines with more spaces than the threshold are marked as *candidate table lines*.

One extra characteristic has to be taken into account when deciding whether a line belongs to a table: its neighbourhood. No matter how many inner spaces a line has, it is not in a table if lines around it are plain text; and

a line with no inner spaces is not plain text, if table lines surround it. Ng et al. [4] found an interesting way of incorporating this: when deciding about a line, they considered characteristics of the lines that fall within a fixed 1-line band around it (the lines before and after it).

Rather than considering a band of invariable size, our method defines the size of the band according to the characteristics of the page it belongs to, because the information we want to extract is more likely to be in pages with a high concentration of table lines, either one big table or several small tables. In fact, in our sample, a statistically significant at 1% positive correlation was found between the *proportion of table lines* in a page and a variable taking value 1 when the page contains a table with relevant information. We estimate this proportion by counting the *candidate lines* identified by criterion (2).

3. The algorithm

For each page p with more than a given minimum of m candidate table lines, we identify *Windows* of lines within which to look for table lines. If the proportion of candidates on the page, $Weight_p$, is high enough, all lines between the first and the last candidates, c_a and c_z, are included in the window. Otherwise, we may have more than one window per page, each corresponding to a sequence of at least m candidate lines no more than k lines apart. Windows always include i lines above the first candidate line c_a and f lines below the last c_z.

The distance between any two lines is measured in terms of the total number of non-empty lines that separate them. This makes the method robust to the existence of long sequences of empty lines within tables, which can be quite common in this context.

Each line in a window will be classified as *target area* if it has any contiguous inner spaces or if its total string length is smaller than x% of the maximum length of the candidate lines on the page. The procedure is iterated until there are no more sufficiently close candidate lines on the page and no more pages with enough candidate lines in the whole document. The values of k, i and f, and thus the size of the windows, are determined according to the proportion of candidate lines on the page.

Instead of simply marking each line as belonging to a target area or not, an absolute index is assigned as a first approach to the delimitation of different tables. Consecutive non-empty lines classified by the algorithm as *target areas* are assigned the same table identifier, and this facilitates the subsequent table processing steps.

3.1 Parameters

Based on the sample of 19 reports, we considered three different manners of estimating the threshold $\lambda(D)$ [6]:

- γ standard deviations, $S_X(D)$, from the mean, $\overline{X}(D)$, of inner spaces per line in report D, determining γ as the value that minimizes the average square error when estimating the ideal threshold of each document;

$$\lambda_1(D) = \overline{X}(D) + 0{,}46 * S_X(D) \quad (3)$$

- an equation based on the most common number of positive inner spaces in each report ($Mode_X(D)$), using the ordinary least square method, which minimizes the average square error of the estimates of $\lambda_2(D)$ given $Mode_X(D)$, to determine the coefficients;

$$\lambda_2(D) = 17{,}9 + 1{,}4 * Mode_X(D) \quad (4)$$

- and the minimum of equations (3) and (4). We achieved better experimental results with the last option.

The parameters of the algorithm were set at default values, which we have empirically tested: m=3 or m=2 (m=3 was found to be better), x=75% and y=50%. We determine k, i, and f as a function of the weight of candidate lines per page, as described in Figure 2.

	k	i	f
$Weight < 30\%$	1	1	1
$30\% \leq Weight < 50\%$	2	2	2
$Weight \geq 50\%$	3	4	2

Figure 2. Default rule for the definition of k, i, and f for a given the proportion of candidate lines per page

The user can manipulate the values of m, k, i, f, and their relationship with the concentration of candidate lines in the page, and as such influence the performance of the method, by providing the parameters that can best seize the tables holding the information to be extracted.

4. Evaluation

We evaluate the tasks of recognizing only tables with relevant information (*context-specific purpose*) and recognizing all tables (*generic purpose*). We measure the percentage of lines we have been able to discard (*economy*) and the proportion of actual table lines that are correctly identified by the method (*recall*).

For the group of reports used as training examples, with 87,843 lines (70,584 non-table lines; 9,685 lines of tables containing the information we wish to extract) using default values, the algorithm was able to discard in average 74,4% of the lines in 17 of 19 reports, while keeping 99,6% of the interesting lines. In the other two reports, by manipulating the default values, the algorithm detected 100% of the desired information while discarding in average 62,3% of the examples. In an unseen test group of three reports, with a total 9,470 lines (7,388 non-table lines; 1,386 table lines containing the information we wish to extract), average recall was

99,4% and economy 73,6%. All interesting tables were at least partially detected in both groups.

For a generic purpose evaluation, and because we wish to detect the presence of any type of table, whatever type of page it belongs to, we have chosen for parameter values $k = f = 31$, $i = 5$, $m = 3$, independently of the weight of candidate lines on the page. Average results were 98,5% recall and 62,6% economy over the 19 examples; and 99,7% recall and 63,6% economy in the unseen group. The only types of tables this method could not detect were very narrow tables standing horizontally isolated in its page (e.g. a slightly narrower Figure 2), when in source documents mostly written in two or more columns and in pages with no other distinctive graphical elements. These are undetected for not enough candidate lines exist on the page. If we used m=1, we would be able to detect 99,9% of table lines, with a 47,7% economy.

Figure 3 shows the performance of the methods parameterised for context specific and generic evaluation. As can be seen, economy is practically always above 50% and recall is consistently above 95% (except for one financial statement, which showed a table using dots as delimiters, instead of inner spaces; the method can easily be adapted to detect column separators other than white space, including dots).

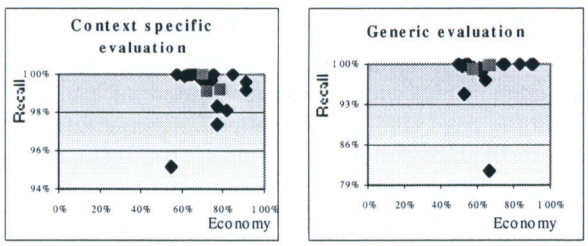

Figure 3. Performance of the methods parameterised for context specific and generic evaluation in the 22 documents (the three squares represent the unseen reports)

5. Conclusion

We have developed a method to locate portions of ASCII documents with high likelihood of holding tables displaying the information we wish to extract. This is a necessary step in a longer process of information extraction from tables in text. We have applied the method in the context of financial statements, which are mandatory reports published by companies accounting for their activities on at least a yearly basis. Information extraction from such reports is relevant as a support for the decisions of many economic agents, but current approaches are mainly manual and time consuming.

The method, which works on ASCII inputs, has been developed by applying data analysis techniques, from statistics to data mining, on a set of 19 financial statements published on the Web by Portuguese companies. It identifies sequences of potential table lines based on the characteristics of neighbouring lines, the current page and the source document as a whole. The method is able to adapt to different styles used in organizing the documents' graphical components. Styles can be very diverse in this highly and highly marketing influenced context. Apart from this, the method allows parameterisation to further adapt it to the specificities of the tables to detect. The performance of the method was measured on the given set of reports and on a separate test set. Results show that a lot of work can be saved right away, without losing relevant information.

Another advantage of this method is that virtually any type of document can be converted to ASCII, including paper documents, after digitalizing and using an optical character recognition software (OCR) that preserves the relative positions of the items on the page. Awareness to certain keywords and recognition of titles has the potential of bringing recall closer to 100% with very low economy costs, and will be sought in future work; as will the other tasks for extracting information from tables.

6. References

[1] Conover, W. J, *Practical nonparametric statistics*, John Wiley, New York, USA, 1999.

[2] Hurst, Mathew, "The interpretation of tables in text", PhD. Thesis, School of Cognitive Science, Informatics, The University of Edinburgh, UK, 2000.

[3] Kieninger, Thomas, "Table structure recognition based on robust block segmentation", *Proceedings of IS&T/SPIE's 10th Annual Symposium Electronic Imaging '98: Science Technology – Document Recognition V*, San Jose, USA, 1998.

[4] Ng, Hwee Tou, Chung Yong Lim and Jessica Li Teng Koo, "Learning to recognize tables in free text", *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, Maryland, USA, 1999, pp. 443-450.

[5] Pyreddy, Pallavi, W. Bruce Croft, "A system for retrieval in text tables", Technical report 105, Center for Intelligent Information Retrieval, Dep. Computer Science, University of Massachussets, USA, 1997.

[6] Silva, Ana Costa e, "Extracção da informação de tabelas em texto", MSc Thesis, Faculdade de Economia, Universidade do Porto, Portugal, 2003.

[7] Tupaj, Scott, Zhongwen Shi, C. Hwa Chang and Hassan Alan, "Extracting tabular information from text files", EECS, Tufts University, Medford, USA, 1996.

[8] Weka software, The University of Waikato, New Zealand, http://www.cs.waikato.ac.nz/ml/weka/

[9] XBRL, http://xbrl.org

Indexing and Retrieval of Document Images Using Term Positions and Physical Structures

Koichi Kise, Keinosuke Matsumoto
Dept. of Computer and Systems Sciences, Osaka Prefecture University
1-1 Gakuencho, Sakai, Osaka 599-8531, Japan
kise@cs.osakafu-u.ac.jp

Abstract

This paper presents some methods of indexing and retrieval of document images based on their physical (layout) structures and term positions in pages. Documents are divided into blocks that are physically de ned for the purpose of indexing with terms. The simple vector space model (VSM) and the latent semantic indexing (LSI) are employed as retrieval models. Experimental results on the retrieval of 129 documents show that LSI with blocks consisting of overlapping pages outperforms an ordinary method of retrieval based on the VSM.

1. Introduction

Document image databases (DIBs) are the databases that provide ef cient storage of and access to document images [1]. As it becomes more popular to equip copiers and printers with devices for the storage of document images, DIBs gain importance in our society.

An open issue of DIBs is how to achieve content-based retrieval based on queries given by users. In the case that queries are given as image features such as layout of documents [2], indexing by layout analysis should be applied. If queries are given as keywords, it is required to apply OCR for indexing. In addition to the issue of OCR errors [3], we have another issue of how to index document images based on recognized characters and words.

A simple way is to employ the Bag of Words model, which is common in the eld of information retrieval. In this model, documents are regarded as collections of words. In the context of DIBs, therefore, the rest of information obtained through the process of OCR, e.g., positions of characters / words, and physical (layout) structures of documents, is discarded.

This paper presents methods of utilizing some of the discarded information in addition to words (terms) themselves so as to improve the accuracy of retrieval. To be precise, documents are indexed based on the information of physical structures such as pages and columns, as well as positions of terms in pages. As an example of such methods, we have already proposed the method called density distributions of terms [4, 5], which is an application of passage retrieval in the IR eld to document images. The methods proposed in this paper can be viewed as simpli ed versions of this method for the reduction of computational costs.

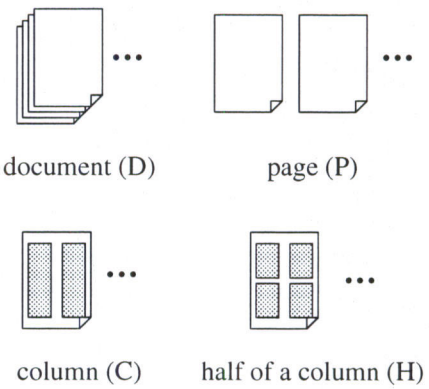

Figure 1. Units of indexing.

2. Indexing

Our methods of indexing are de ned by *units* and *blocks* of indexing. As units of indexing, we consider page (P), column (C) and half of a column (H) in addition to document (D) as shown in Fig. 1.

Columns are obtained in a brute-force manner in order to avoid complicated problems in layout analysis: each page region is cut into two pieces at a xed position. As a result, regions laid out in a single column format as well as some wider gures and tables are split into different pieces. Halves of columns are likewise de ned by splitting at the physical vertical center of columns.

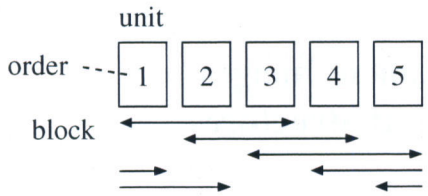

Figure 2. Blocks of indexing. A block consists of one or more units sorted in the reading order. Blocks are regarded as pseudo documents and retrieved.

The units except for document are ordered by pages within each document, and within each page from left to right and top to bottom. Blocks of indexing are determined based on the order. Figure 2 illustrates blocks consisting of three units. The unit at the end is connected to the beginning so that all units are treated equivalently. As shown in this gure, blocks are de ned by shifting a starting unit one by one. In this paper, we consider $1 \sim 5$ as the numbers of units in a block. Note that blocks consisting of a single unit do not have overlaps with other blocks.

Each block is regarded as a pseudo document and indexed by terms with tf-idf weights as follows. First, terms are de ned by applying stemming and elimination of stop words to all words included in documents in the database. Next, term frequency is counted in each block. Let f_{ik} be the frequency of a term t_i in a block b_k. A block b_k is represented as a vector:

$$\boldsymbol{b}_k = (w_{1k}, ..., w_{mk})^T \ , \tag{1}$$

where T indicates the transpose, and w_{ik} is the weight of a term t_i in a block b_k. The de nition is as follows.

$$w_{ik} = \mathrm{tf}_{ik} \cdot \mathrm{idf}_i \ , \tag{2}$$

$$\mathrm{tf}_{ik} = \begin{cases} \log(f_{ik}/U) + 1 & \text{if } f_{ik} > 0 \ , \\ 0 & \text{otherwise} \ , \end{cases} \tag{3}$$

$$\mathrm{idf}_i = \log(B/B_i) \ , \tag{4}$$

where U, B and B_i are the number of units in a block, the total number of blocks for all documents in the database, and the number of blocks that include a term t_i, respectively.

3. Retrieval

Retrieval of documents is based on a user s query q, which is also treated as a pseudo document and represented as a vector:

$$\boldsymbol{q} = (w_{1q}, ..., w_{mq})^T \ , \tag{5}$$

where w_{iq} is a weight of a term t_i in the query q. The value is calculated similarly to the case of blocks.

The result of retrieval is represented as a ranked list of documents sorted according to the degree of similarity between a document $d_j (1 \leq j \leq n)$ and the query q. Let $\boldsymbol{b}_{j1}, ..., \boldsymbol{b}_{jr}$ be vectors of blocks in a document d_j. Then we de ne the degree of similarity as

$$\mathrm{sim}(d_j, q) = \max_{1 \leq k \leq r} \{\mathrm{sim}(\boldsymbol{b}_{jk}, \boldsymbol{q})\} \ , \tag{6}$$

where $\mathrm{sim}(\boldsymbol{b}_{jk}, \boldsymbol{q})$ is a degree of similarity between a block b_{jk} and the query q.

In this paper, we employ two kinds of degrees of similarity: a degree obtained by the simple vector space model (VSM), and a degree by latent semantic indexing (LSI). In the simple vector space model, the degree of similarity is de ned as the cosine of the angle between two vectors [6]. Latent semantic indexing is the retrieval model which makes use of singular value decomposition to reduce the dimensionality of document vectors [6]. In this method, the dimensionality of a term-by-block matrix $D = [\boldsymbol{b}_{11}, \boldsymbol{b}_{12}, ..., \boldsymbol{b}_{nr}]$ is reduced to κ ($< \mathrm{rank}(D)$), which is a parameter to be predetermined.

In total, we have de ned 32 methods as combinations of a retrieval model (VSM or LSI), a unit (D, P, C, H), and the number of units in a block (1, 2, ..., 5). Note that the number of units is limited to 1 for the unit D . In the following, methods are denoted like VSM(D1) .

4. Experiments

4.1. Data set

Preliminary experiments were carried out to test the proposed methods. A data set employed in the experiments was prepared based on the proceedings of ICDAR2001 [7]. The titles of oral sessions were utilized for de ning queries shown in Table 1. All the papers for the sessions were put in the document database and the relevance judgment was de ned as follows: documents are relevant to a query if they are assigned to its corresponding session. Statistics about the data set are shown in Table 2.

4.2. Evaluation

4.2.1 Mean average precision

For evaluating experimental results, we employ the mean average precision over all relevant documents [6]. The definition is as follows.

As described in Sect. 3, the result of retrieval is represented as the ranked list of documents. Let $r(i)$ be the rank of the i-th relevant document counted from the top of the

Table 1. Queries.

1	handwriting classi ers
2	image processing
3	OCR and document understanding
4	handwriting features and writer identi cation
5	evaluation and datasets
6	word recognition
7	classi ers and learning
8	graphics recognition
9	document analysis systems, check processing and postal automation
10	applications
11	online handwriting
12	tables
13	classi cation and language models
14	forms
15	segmentation
16	web documents and video
17	information retrieval and word spotting
18	color and multimedia documents
19	image quality and style

Table 2. Statistics about the test collection.

no. of terms	7022	
no. of doc.	129	
no. of pages	673	
ave. doc. len.	1507	terms
ave. page. len.	289	terms
no. of queries	19	
ave. query len.	2.37	terms
ave. no. of rel. doc. per query	6.79	

list. The precision at retrieving this document is calculated by $i/r(i)$. Average precision p_k for a query q_k is obtained by averaging precision values for all documents relevant to the query:

$$p_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \frac{i}{r(i)} \ , \tag{7}$$

where N_k is the number of relevant documents for the query q_k. The mean average precision over all relevant documents is de ned as the mean of the respective values over all queries:

$$p = \frac{1}{N} \sum_{k=1}^{N} p_k \ , \tag{8}$$

where N is the number of queries.

For example, consider two queries q_1 and q_2 which have two and three relevant documents, respectively. Suppose the ranks of relevant documents for q_1 are 2 and 5, and those for q_2 are 1, 3 and 10. The average precision for q_1 and q_2 is computed as $(1/2 + 2/5)/2 = 0.45$ and $(1/1 + 2/3 + 3/10)/3 = 0.66$, respectively. Then the mean average precision which takes into account both queries is $(0.45 + 0.66)/2 = 0.56$.

4.2.2 Statistical test

The next step for the evaluation is to compare values of the mean average precision obtained by different methods. An important question here is whether the difference in the mean average precision is really meaningful or just by chance. A way to make such a distinction is to apply a statistical test. In this paper, we utilize a test called the paired t-test [8]. The following is the summary of the test.

Let a_k and b_k be the scores (e.g., the average precision) of retrieval methods A and B for a query q_k , and de ne $\delta_k = a_k - b_k$. The test can be applied under the assumptions that the model is additive, i.e., $\delta_k = \mu + \varepsilon_k$ where μ is the population mean and ε_k is an error, and that the errors are normally distributed. The null hypothesis here is $\mu = 0$ (A performs equivalently to B in terms of the scores), and the alternative hypothesis is $\mu > 0$ (A performs better than B).

It is known that the Student s t-statistic

$$t = \frac{\bar{\delta}}{\sqrt{s^2/N}} \tag{9}$$

follows the t-distribution with the degree of freedom of $N - 1$, where N is the number of samples (queries), $\bar{\delta}$ and s^2 are the sample mean and variance:

$$\bar{\delta} = \frac{1}{N} \sum_{k=1}^{N} \delta_k \ , \tag{10}$$

$$s^2 = \frac{1}{N-1} \sum_{k=1}^{N} (\delta_k - \bar{\delta})^2 \ . \tag{11}$$

By looking up the value of t in the t-distribution, we can obtain the P-value, i.e., the probability of observing the sample results δ_k $(1 \leq k \leq N)$ under the assumption that the null hypothesis is true. The P-value is compared to a predetermined signi cance level α in order to decide whether the null hypothesis should be rejected. In this paper, we utilize $\alpha = 5\%$.

4.2.3 Parameter

The last thing we should consider for evaluating the methods is how to determine the value of the parameter κ, i.e., the number of dimensions used in LSI. In order to make a comparison unbiased, we apply *leave-one-out* cross-validation: the average precision for a query q_i is obtained using the dimension $\kappa_i \in \{10, 20, ..., 600\}$ which produces the maximum mean average precision for the rest of queries $q_1, ..., q_{i-1}, q_{i+1}, ..., q_N$.

Table 3. Mean average precision.

unit	no. of units in a block	VSM	LSI
D	1	0.315	0.293
P	1	0.310	0.350
	2	0.323	*0.352*
	3	0.313	**0.358**
	4	0.327	0.293
	5	0.329	0.320
C	1	0.294	0.325
	2	**0.339**	0.327
	3	**0.339**	0.334
	4	*0.336*	0.318
	5	0.333	0.327
H	1	0.293	0.340
	2	0.331	0.292
	3	0.314	0.314
	4	0.330	0.332
	5	0.326	0.307

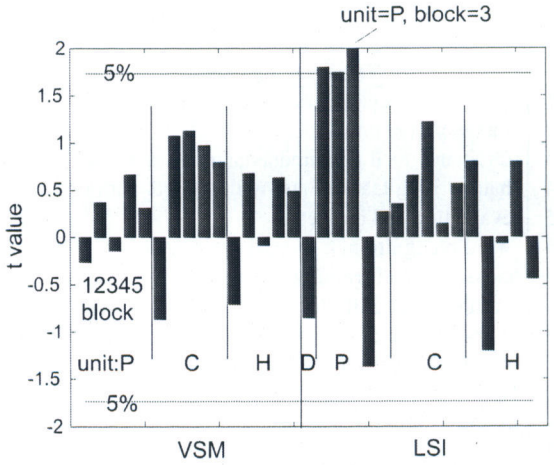

Figure 3. Results of the paired t-test. The horizontal axis indicates methods represented as combinations of a retrieval model, a unit, and the number of units in a block. The methods which exceed the upper 5% line are signi - cantly better than the baseline VSM(D1).

4.3. Results and discussion

Table 3 shows the mean average precision for all methods. For the VSM, no good results were obtained for the blocks without overlap (no. of units = 1). On the other hand, the blocks of size more than 1 often yielded better results as compared to VSM(D1). Although there is no such clear tendency for LSI, units smaller than D often resulted in improvement as compared to LSI(D1).

In order to clarify the signi cance of differences in the mean average precision, we applied the paired t-test taking VSM(D1) as a baseline. The results are illustrated in Fig. 3 where the vertical axis indicates the t-value and the horizontal lines in the gure show the signi cance level of 5%. Although most of the methods were not signi cantly better than the baseline, the methods LSI(P1), LSI(P2) and LSI(P3) were the exceptions.

5. Conclusion

We have proposed the methods of indexing based on term positions and physical structures of documents. The results of preliminary experiments show that some methods based on LSI with pages as units of indexing yielded significantly better results as compared to the ordinary VSM.

Although the numbers of documents and queries are too small to draw a nal conclusion, there is some possibility of improving the ordinary VSM by taking pages or other units for indexing. Thus the future work includes experiments with larger datasets and improvement of retrieval methods.

Acknowledgment

This research was supported in part by Grant-in-Aid for Scienti c Research (C) from Japan Society for the Promotion of Science (No.14580453).

References

[1] Doermann, D.: The Indexing and Retrieval of Document Images: A Survey, *Computer Vision and Image Processing*, Vol. 70, No. 3, pp.287–298, 1998.

[2] Hu, J., Kashi, R. and Wilfong, G., Document Image Layout Comparison and Classi cation, *Proc. 5th ICDAR*, pp.285–288, 1999.

[3] Jones, G.J.F. and Lam-Adesina, A.M., Examining the Effectiveness of IR Techniques for Document Image Retrieval, *Proc. SIGIR Workshop on Information Retrieval and OCR: From Converting Content to Grasping Meaning*, 2002.

[4] Kise, K., Tsujino, M. and Matsumoto, K., Spotting Where to Read on Pages Retrieval of Relevant Parts from Page Images, in *Proc. DAS 02*, pp.388–399, 2002.

[5] Kise, K., Yin, W. and Matsumoto, K., Document Image Retrieval Based on 2D Density Distributions of Terms with Pseudo Relevance Feedback, in *Proc. ICDAR 03*, to appear.

[6] Baeza-Yates, R. and Ribeiro-Neto, B., *Modern Information Retrieval*, Addison-Wesley Pub. Co., 1999.

[7] `http://icdar.djvuzone.org/`

[8] D. Hull, Using Statistical Testing in the Evaluation of Retrieval Experiments , in Proc. SIGIR 93, pp.329–338, 1993.

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-

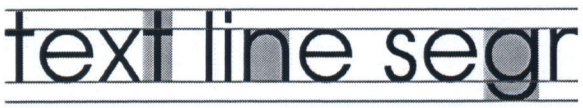

Figure 1. Text line model used by the gapped text line segment finding algorithm.

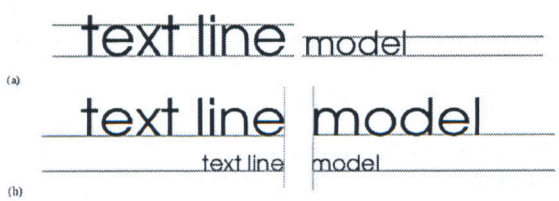

Figure 2. Changes in text line models that cause text line breaks to be detected.

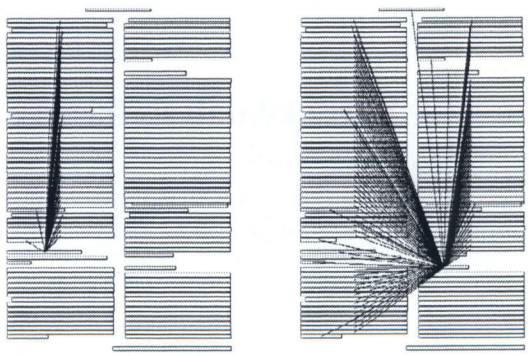

Figure 3. Examples of pairwise reading order relationships determined for text line segments in actual documents.

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

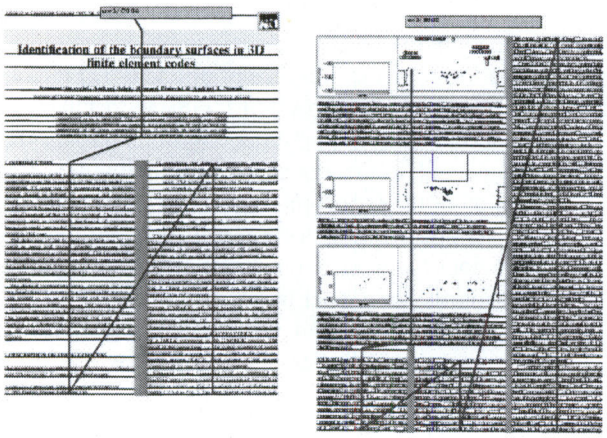

Figure 4. Examples of reading order recovery by topological sorting. Blue lines indicate text line segments, and the red line connects the center of successive text line segments in reading order.

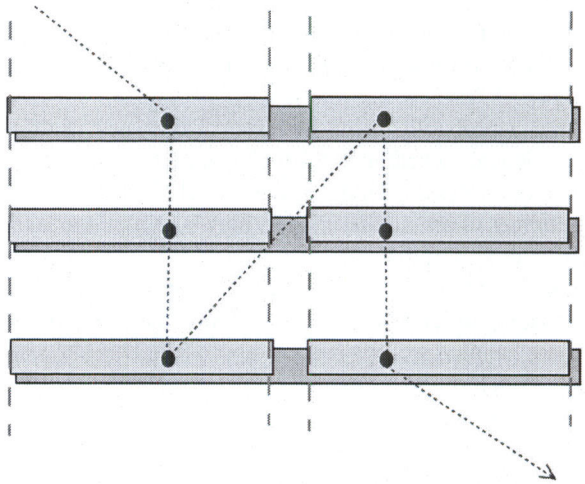

Figure 5. Schematic illustration of the probabilistic segmentation framework described in the text. A gapped text line segmentation algorithm will return alternative interpretations of the input in terms of text line segments (shown as green rectangles). A maximum likelihood interpretation of the page 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.

Assuming Accurate Layout Information for Web Documents is Available, What Now?

Hassan Alam, Rachmat Hartono, Aman Kumar, Fuad Rahman[§], Yuliya Tarnikova and Che Wilcox
BCL Technologies Inc.
fuad@bcltechnologies.com

Abstract

Re-authoring of multi-modal documents, such as web pages, has proved itself to be a difficult problem. With the widespread recognition and subsequent adoption of XML, expectations were high that the WWW will see a transformation in terms of web page authoring and sharing. Unfortunately, that has not happened. Today, as was last year, and the year before that, the majority of the web pages are still written using HTML, with little or no content descriptions and explicit relationship between data and structure. That has made the task of web page re-authoring difficult. But what if we had accurate layout information for these very troublesome web pages? This paper discusses some interesting ideas about the future of web page re-authoring with this assumption.

1. Introduction

Web page re-authoring is a problem that has been studied for a number of years. This has attracted renewed attention recently as various handheld devices such as Personal Digital Assistants (PDAs) and cell phones have become capable of browsing web pages. The limited form factor of these devices in terms of display area is a major obstacle to mass adoption of this technology. The majority of the web pages are still designed to be viewed by full resolution desktop monitors and any attempt to effectively browse and locate information using tiny viewing displays becomes very difficult. Keeping this in mind, researches have turned their attention to web page re-authoring to make the rendering most appropriate given local display area restrictions.

Since most of the web pages in the world are still written using HTML, any re-authoring requires transformations into various different formats including HDML, HTML 3.2, WAP, iMode (NTT DoCoMo), J-Sky (J-Phone) and EZweb (KDDI) to name a few. On top of that, format translation is only part of the problem. The content can be often encoded in tabular (HTML table construct) format, and different multi-media objects such as graphics, hyperlinks, flash, java codes etc. can be embedded into it. This makes the task of accurate transformation very difficult.

In [1], to be presented in ICDAR 2003, we have discussed how it is possible to use a combination of natural and non-natural language techniques to produce high quality re-authoring solutions for web pages written in HTML. These so-called 'legacy' documents are all we have before a universal high level formatting is adopted by everyone (and the dream is on!). Now if we can *assume* that accurate layout information is available, information such as "this is a table, rectangle coordinates xx, yy, ...", "this is a paragraph of texts, rectangle coordinates xx, yy, ...", "this is a graphic, rectangle coordinates xx, yy, ... ", etc., what do we do then? How do we use this information? How do that information help us in getting better re-authoring solutions? This paper discusses some of the pathways to this future.

2. Related Work

Before we start, here is a very brief outline of past work. Over the years, researchers have proposed different solutions to the problem of web page re-authoring. The possible solutions can be categorized as:

- **Handcrafting:** Handcrafting involves typically crafting web pages by hand by a set of content experts for device specific output. This process is labor intensive and expensive. BBC (www.bbc.co.uk) web site is an example, where handcrafted web content is served in text only format.
- **Transcoding:** Transcoding [2] replaces HTML tags with suitable device specific tags, such as HDML, WML and others.
- **Adaptive Re-authoring:** The research on web page re-authoring can be broadly separated into two parts: approaches that explicitly use natural

[§] Corresponding author. The authors gratefully acknowledge the support of a Small Business Innovation Research (SBIR) award under the supervision of US Army Communications-Electronics Command (CECOM).

language processing (NLP) techniques based on computational linguistics [3,4], and the approaches that use non-NLP techniques [5,6].

3. Web page summarization for handheld devices

Web browsing on a handheld device can be a very annoying task. The principal reason for this is the display area available on these devices. Typically, a PDA (such as a Palm V) can have 160X160 pixels of resolution, whereas a cell phone might be as low as 120X80. PocketPCs have much higher resolution, but the price point is still prohibitive. Trying to use these devices to browse or try to find information on the web is a tiring task. On top of that, since a page is downloaded as a whole before viewing, this can take up significant time. Summarization is a very attractive solution in these cases.

In [1], we propose a web page re-authoring approach based on a combination of NLP and non-NLP techniques. The following is a brief outline of the proposed approach:

- **Web page data structure:** If HTML is used to compose a web page; the data ("content") is arranged using an HTML data structure.
- **Content analysis:** Content analysis aims to decompose a web document based on the extracted structure from the tree hierarchy.
- **Content processing for re-authoring:** The aim is to re-author the content in a format that is most suited for displaying in a target device.
    - **Verbatim. T**he content is not processed any further and is displayed in the target device 'as-is'.
    - **Transcode. T**he content is re-flowed by replacing HTML tags with suitable device specific tags (HDML, WML etc.).
    - **Summarize. T**he content is summarized using NLP and non-NLP techniques.
- **Node merging and segmenting:** In cases where the node is a neighbor of other nodes with similar type of content, the nodes are merged. In cases, where the content is too large, it is split into smaller coherent segments.
- **Representing the complete web page:** Once merging and segmentation is completed, it is possible to recreate the original web page by combining these merged and segmented nodes.
    - **"To summarize or not to summarize":** It so happens that not all the segments of a web page are good candidates for NLP summarization.
    - **Creating a label:** For creating a label, non-NLP techniques are adequate. Visual clues are used to detect the most important segments of the content, or a 'label'.
    - **Creating a summary:** NLP techniques need to be employed to create short summaries of the content [7,8].

However, although this hybrid of NLP and non-NLP approaches has shown great promise, it still lags the relationship between the structure and the content.

As is evident from the work of researchers in the last few years, HTML is hugely exciting, and easy to adopt, but is not very receptive to data descriptions, sharing, automatic conversion, machine readability and interoperability. Since web page re-authoring needs to transform and adapt existing web pages to the display capabilities of other devices, in general of smaller display area, an explicit relationship (i.e. ontology) between the content and the data structure is extremely important.

Assuming accurate layout information is available, we explore how adoption of linguistic knowledge and semantics with a combination of explicit ontology and XML representation can solve this problem gracefully.

4. The Future: Marrying Ontology with XML

We assume that now have information regarding the structure of the web page, but what we do not have is an ontology defined for that domain. Web pages can be very variable, defining ontology applicable to all web pages, therefore, can be a daunting task. Here we attempt to define a generic ontology defining all web pages.

The approached described in [1] does not need to extract very detailed structure of web pages in the re-authoring process, not do we require it. A high level structure is all that is required to apply these ideas. The extracted elements then can be conveniently used to generate an intermediate XML description of the web page. On the other hand, it is entirely possible to map these high level structures and their relationship to the data using a simple ontology.

Ontology is a specification of a conceptualization [9]. Ontology establishes a joint terminology between members of a community of interest. These members can be human or automated agents. In order to define such ontology for the domain of web pages, we need to define four things, a list of the elements, concept hierarchy, concept association and finally rules or axioms [10]. **Table 1** has a list of possible elements of a web page.

web address	Ticker	icon	Navigati-on	Map
Title	graphic banner	box	bulletlist	Section
Heading	side burst	content	panel	Highlights bannerad (advertisement) (border, tab, height, width)
sub-heading	Image	text	menu	Coordinates
banner ad	graphic	link	author	Table
directions				

Table 1: A possible list of elements

The concept hierarchy based on these elements can be described in the following way:

```
Object []
address :: Object
  computerAddress :: address
      WebAddress :: computerAddress

text :: Object
  title :: text
  line :: text
    headine :: line
      subheading :: heading
    banner :: line

message :: Object
      promotion :: message
        ad :: promotion
          bannnerAd :: promotion

representation :: Object
  symbol :: representation
    ticker :: symbol

artifact :: Object
  creation :: artifact
    represenation :: creation
      display :: representatin
      image :: display
      icon  :: display
        graphic :: image
    illustration :: representation
      graph :: illustration

form :: Object
  figure :: form
    rectangle :: figure
      box :: rectangle

communication :: Object
  message :: communication
    content :: message

relation :: Object
  communication :: relation
    writing :: communication
      coding system :: writing
        computerCode :: codingSystem
          command :: computerCode
            link :: command

activity :: Object
  control :: activity
    direction :: control
    steering :: direction
      navigation :: direction

relation :: Object
  communication :: relation
    content :: communication
      information :: content
        database :: information
          list :: database
            bulletlist :: list
        menu :: database
    writtenCommunication :: communication
      writing :: writtenCommunication
        section :: writing
    genre :: communication
      prose :: genre
        nonfiction :: prose
          article :: nonfiction

artifact :: Object
  display :: artifact
    window :: display
      panel :: window
  creation :: display
    representation :: creation
      map :: representation

animateThing :: Object
  person :: animateThing
    author :: person
```

Similarly, the concept association can be conveniently expressed as:

WebAddress[name =>> STRING; colon =>> STRING; slash =>> STRING; domainName => STRING;
newspapername =>> webNewspaper; portal =>> webPage]
text[font =>> NUM; format =>> STRING; size =>> NUM]
banner[color =>> STRING; background; sharp;]
ticker[font =>> NUM; format =>> SRING; size =>> NUM; case;]
graph[color =>> STRING; background; size =>> NUM]
image[color =>> STRING; background; size =>> NUM]
box[text =>> STRING; size =>> NUM; font =>> NUM; format =>> STRING]
link[color =>> STRING; size =>> NUM]
bulletlist[size =>> NUM; font =>> NUM; format]
article[font =>> NUM; format =>> STRING; size =>> NUM; link; author =>> STRING; subheading; section]
map[format =>> STRING; size =>> NUM; link; text]

Finally, the rules or axioms are expressed as follows:

FORALL Pers1, Art1
Art1: Article[author ->> Pers1] <->
Pers1: Person[Article ->> Art1]

FORALL Rep1, Rep2
Rep2: Banner[containsAd ->> Rep1] <->
Rep1: BannerAd[banner ->> Rep2]

FORALL Dis1, Dis2
Dis1: Graph[illustration ->> Dis2] <->
Dis2: Image[illustrates ->> Dis1]

FORALL Tex1, Fig1
Fig1: Box[Line ->> Tex1]
Tex1: Text[Line ->> Fig1]

This shows that the ontology for web pages can be conveniently expressed in terms of the derived XML structure. Since the hybrid method of web page re-authoring, presented in [1], exploits all these structures, it is very easy to exploit this ontology to generate the most appropriate format of a web page for a specific device. So the re-authoring takes a different path than was described earlier: The structure is generated, the processing mode is selected, content of each block is either summarized or re-flowed, output level (either single or double levels) decided, and then the ontology is used to re-format the source web page. This improves the quality of the output in many ways. Not only that it becomes possible to capture the contextual relationship among various components within the document, it also leads to better understanding of the information contained within the document. This additional information can be used in other processes, such as document categorization and contextual search.

Future mobile web browsing will be very much simpler as semantic web becomes more widely used and accepted.

5. Conclusion

This paper has presented some ideas about how to produce better web page re-authoring solutions by using linguistic knowledge and ontology assuming accurate layout information for web pages is available. It is shown that such an approach will produce high quality intelligent summary for web pages allowing fast and efficient web browsing on small display handheld devices such as PDAs and cell phones.

One assumption of this approach is the use of explicit classification of document elements in addition to accurate layout information. In a multi-modal web page, specific identification and classification of component structures, such as headings, text blocks, images, graphics, audio, video, flash, image maps etc. leads to more precise re-authored output. Since each of these components are labeled at early stages of the process, putting them together while maintaining the correct contextual relationships among these components becomes possible. During the presentation of the paper, demonstrations will be arranged to show this functionality.

It is also implicitly assumed that the future of mobile browsing lies in the adoption of semantic web technology. However, before that realizes, the proposed approach offers a workable compromise to generate high fidelity re-authored web pages suitable for viewing on a host of display devices automatically from currently available multi-modal web pages.

As is stated in the beginning, this is an exploratory paper offering a specific pathway to the future of web page re-authoring provided accurate layout information is available. Currently, it is beyond the capability of any algorithm to achieve this level of accuracy. However, approximations to that accuracy are attainable and even practical. It will be interesting to discuss other possibilities in this space during the DLIA workshop.

References

1. Alam, H., Hartono. R., Kumar, A., Tarnikova, Y., Rahman, F. and Wilcox, C., "Web Page Summarization for Handheld Devices: A Natural Language Approach", 7th Int. Conf. on Document Analysis and Recognition (ICDAR'03). In press.
2. Hori, M., Mohan, R., Maruyama, H. and Singhal, S., "Annotation of Web Content for Transcoding". E3C Note. http://www.w3.org/TR/annot.
3. Berger, A. and Mittal, V., "OCELOT: A System for Summarizing Web Pages". Research and Development in Information Retrieval, pages 144-151, 2000.
4. Buyukkokten, O., Garcia-Molina, H., and Paepcke, A., "Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices". Proc. of the Tenth Int. World-Wide Web Conference, 2001.
5. Bickmore, T., Girgensohn, A. and Sullivan, J., "Web page filtering and re-authoring for mobile users". The Computer Journal, 42(6):534-546, 1999.
6. Zhang, H., "Adaptive content delivery: A new research in media computing". Proc. Int. Conf. on Multimedia data Storage, retrieval, Integration and Applications, MDSRIA, 2000.
7. McKeown, K., Barzilay, R., Evans, D., Hatzivassiloglou, V., Schiffman, B., and Teufel, S., "Columbia Multi-Document Summarization: Approach and Evaluation". Proc. of the Workshop on Text Summarization, ACM SIGIR Conference, 2001.
8. McKeown, K., Klavans, J., Hatzivassiloglou, V., Barzilay, R., Eskin, E., "Towards Multidocument Summarization by Reformulation: Progress and Prospects". AAAI/IAAI, pages 453-460, 1999.
9. Gruber, T., A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993.
10. Erdmann, M. & Studer, R., Ontologies as conceptual models for XML documents. Twelfth Workshop on Knowledge Acquisition, Modeling and Management, 1999.

Ground-Truth Production and Benchmarking Scenarios Creation With DocMining

Eric Clavier[1], Pierre Heroux[2], Joel Gardes[1], Eric Trupin[2]
[1] *FTR&D,FTR&D Lannion 2 Avenue Pierre Marzin, 22307 Lannion CEDEX, France, {rntl.ball001,gardes}@rd.francetelecom.com*
[2] *PSI Laboratory, University of Rouen, 76821 Mont Saint Aignan, France, {Pierre.Heroux, Eric.Trupin}@univ-rouen.fr*

Abstract

In this paper we present the DocMining platform and its application to ground-truth datasets production and page segmentation evaluation. DocMining is a highly modular framework dedicated to document interpretation where document processing tasks are modelized with scenarios. We present here two scenarios which use PDF documents, found on the web or produced from XML files, as basis of the ground-truth dataset.

1. Introduction

Algorithm performance evaluation has become a major challenge for document analysis systems. In order to choose the right algorithm according to the domain or to tune algorithm parameters, users must have evaluation scenarios at their disposal. But efficient performance evaluation can only be achieved with a representative ground-truth dataset. Therefore users must have the possibility to create, access or modify ground-truth datasets too.

Many approaches and tools have been proposed for benchmarking page segmentation algorithm and producing ground-truth datasets. Former architectures and environment can be found in [5] [6] [8]. The ground-truth datasets are usually defined with image regions features like location, label, reading order, contained text. But information needed for a ground-truth dataset depends more on the user's evaluation intention than on a formal definition. Therefore it must be possible to upgrade a ground-truth dataset according to the evolving needs for evaluation.

Performance evaluation criteria, we found in previous works, are various and reflect those needs. Noticeable criteria are overlap ratio, regions alignment, split/merge errors.

This paper describes the DocMining platform and its application to ground-truth dataset production and performance evaluation.

This paper is organized as follow : first, we present the architecture of the DocMining platform and its major components. This platform is aimed at providing a framework for document interpretation, but its modular architecture allows multi-purpose applications based on scenarios. Then we present an application of the architecture where scenarios are designed for ground-truthing and benchmarking page segmentation algorithms.

2. The DocMining platform

The DocMining project is supported by The DocMining consortium, including four academic partners, PSI Lab (Rouen, France), Project Qgar (LORIA, Nancy, France), L3i Lab (La Rochelle, France), DIUF Lab (Fribourg, Switzerland), and one industrial partner, GRI Lab from France Telecom R&D (Lannion, France). DocMining is a multi purpose platform and is characterized by three major aspects.

At first, the DocMining architecture relies on a document-centered approach. Document processings communicate through the document itself; such an approach avoids the problems of data scattering usually met in classical document processing chains.

Second, the DocMining framework is based on a plug-in oriented architecture. Developers can conveniently add new processings, making thus the platform easily upgradeable. Document visualization and manipulation tools are also designed according to this approach, so that a user is able to fully customize the interactions with the document structure.

Third, the platform handles scenario-based operations. Running a scenario collects users' experience, which becomes part of the scenario itself. The scenario may then be transformed into a new processing corresponding to a higher-level granularity.

So the DocMining architecture is really modular because a user can create his own objects, integrate his own processings into the platform, design his own interfaces, define and run his own scenarios. In this way, the platform may be used for various interesting purposes such as benchmarking scenario creation, knowledge base creation, parameters tuning, *etc.*

2.1. Architecture overview

The platform is based on a Java/XML architecture and relies on four major components:

The PSI Library, deriving from different research works at PSI laboratory, proposes processing chains using statistical and/or structural approaches [2]. It contains a classification tools library and a XML data management library.

The Qgar software system [3] is developed by the same-named project at LORIA (www.qgar.org). It is aimed at the design of document analysis applications.

The XMillum (for XML Illuminator) platform [4] is developped by the Software, Image & Document Engineering team, at the Departement of Computer Science of the University of Fribourg. It is a general and extensible tool for the edition and visualization of all kinds of document recognition data, that are transformed into XML using XSLT stylesheets. Display and editing functionalities are delegated to plugins.

The ImTrAc package, developed by the GRI Lab at FranceTelecom R&D Lannion, provides a process engine to control processing execution and a scenario engine to control scenario execution, as well as tools for processing integration and scenario creation. An overview of the platform architecture is given in figure 1.

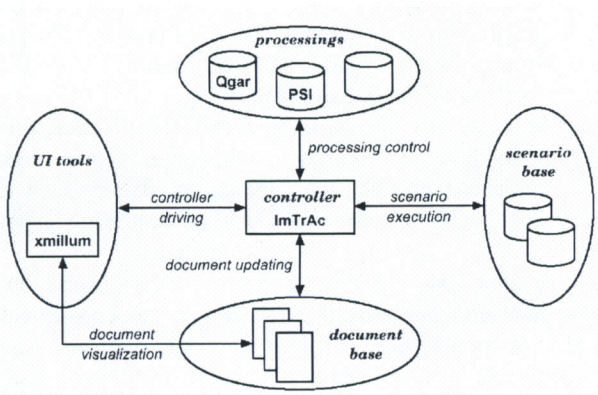

Figure 1 : DocMining platform architecture

2.2. The document structure

The DocMining architecture is based on a document centered approach. A document is represented by an XML tree built according to an XML schema. Basic elements are graphical objects defined by their type (`Binary Image`, `Connected Component`, `Text Block`, *etc*), their source (the document they are extracted from), and their location in the image. We did not try to build a complete predefined taxonomy of possible types: The users of the platform can define their own graphical object types when necessary. A graphical object includes intrinsic data describing the way the object is physically represented. The XML schema we have defined for that is based on basic data types such as `Freeman Chain`, `Feature Vector`, etc., but, just like previously, a user can define its own data types if necessary. However a document is more than a simple description in terms of graphical objects and data. Its structure also contains information (name, parameters, *etc*) about processings which have been applied to the document and which have provided the objects.

As shown in figure 1, objects included in the document structure are visualized with XMillum. XSLT stylesheets define what objects may be visualized, how they are visualized, and how events involving objects (e.g. mouse clicks) are handled. Each object is associated to a Java class, which performs the rendering.

2.3. Interaction between processings

As shown in figure 1, a processing has no direct access to the document structure and cannot modify it if a so-called contract, defined according to an XML schema, has not been established with the document. The contract describes the processing behavior: the way the processing modifies the XML document structure (by adding, removing or updating nodes), the kind of graphical objects it produces, and parameters that do not require access to the document structure. The objects a processing may modify or access are defined by specifying the "trigger" node (the node that enables the execution of the processing) and the "updated" nodes (the nodes which are modified by the processing).

2.4. Scenarios

In order to achieve interpretation tasks, users can interactively build scenarios, which are defined as structured combinations of document processings. There are two ways of creating a scenario. The first way is based on the contracts of the processes. As objects inputs and outputs are specified for all processings in the corresponding contracts, it is possible to determine which processings can feed a given process and then to combine processings. The other way relies on a XMillum component that we have specifically developed for the DocMining platform. It provides means to interact with the ImTrAc processing engine and to visualize the structure of the document. For each object of a document, the ImTrAc engine is able to supply the list of processings that may be applied. Once the user has chosen a processing, the engine supplies its parameters list so as to be able to launch the corresponding process. When the process terminates, the document structure is updated and the user can then interact with the newly created objects.

Each user action on the document is recorded in a scenario, which may be applied later to another document. Each step of a scenario acts as a trigger and includes an XPath expression describing the way the required objects have to be extracted from the document.

3. Page Segmentation evaluation

3.1. Obtaining the document base

PDF (Adobe® Portable Document Format) documents serve as basis for our ground-truth dataset. Indeed, the PDF format is widely used in many applications (newspaper, advertising, slides, ...) and PDF documents can be easily found on the web. Moreover search engines (like google) allow to refine a search according to the document format. So it is very easy to build a PDF document base where many domains are represented.

A ground-truth dataset can also be built with newly created PDF documents. Figure 2 shows different ways to create a PDF representation of an XML document [1]. Each of the tools is freely available for download. The input XML document may be or may contain an instance of a widely used DTD such as DocBook, TEI, MathML, *etc.* Stylesheets (DSSSL or XML) are given for some of theses DTD. These can be modified so that several PDF documents with different formatting attributes may be created from a unique XML document.

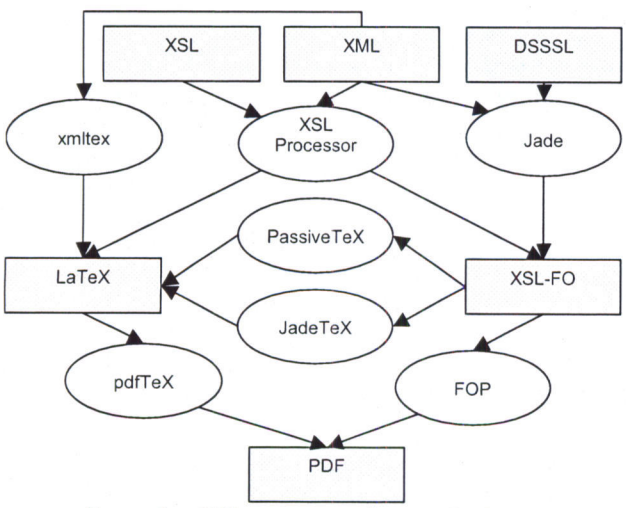

figure 2 : different ways for producing a PDF document from an XML source

With those different approaches, it is possible to build a document base which contains "real life" documents obtained through an internet search and "problem specific" documents built from an XML source.

3.2. Building the ground-truthing scenario

The main drawback of the PDF format is that it is based on a pure display approach, structural and logical information is not directly accessible, those information must be computed from the low level objects contained in the PDF document.

The ground-truth dataset is obtained through a three steps scenario:

- select the PDF documents.
- extract the physical structure from their PDF representation.
- save the generated ground-truth structure.

For each scenario step we have defined and developed a processing observing our contract approach. For example, figure 3 shows the contract we defined for the extraction of the physical structure of a PDF document. Contract noticeable elements are bold typed:

- `handled_object` : the object which is processed. Here the trigger node and the updated node are the same (see 2.3). The document must contain a `PdfDoc` element to launch the processing.
- `process_config` : the parameters of the processing
- `produced_object` : the processing produces three kind of objects (text lines, words and images) with an unknown cardinality (indicated by the list attribute).

-

```
<process_property class_name="PdfSeg">
<service name="nodeAdd">
 <handled_object>
    <object_doc type="PdfDoc"/>
    <process_config>
  <param type="ParamIn" name="ExtractImage"
support="Data" param_value="0" info="if 1
extract the images"/>
  <param type="ParamIn" name="RemoveWord"
support="Data" param_value="1" info="if 1
remove word textpieces (make the document
lighter)"/>
 </process_config>
<produced_object>
  <object_doc type="TextLine" list='yes'/>
  <object_doc type="Word" list='yes'/>
  <object_doc type="Image" list='yes'/>
</produced_object>
</handled_object>
</service>
</process_property>
```

figure 3 : contract of the PDF segmentation processing

Structure extraction from the PDF is done by using the PDF parsing API provided in the Multivalent

package [8] a java platform dedicated to the visualization of various formats documents

The resulting structure is then marshalled into an XML file observing our XML schema. Therefore, this file contains the ground-truth information of the document Figure 4 shows an excerpt of the ground truth structure and the figure 5 its visualization with XMillum.

```
<object_doc object_id="56" type="TextLine">
<object_pos h="11" w="127" x="14" y="660"/>
  <object_data>
    <ascii_data>daz, et créé ce week-end à
    </ascii_data>
  </object_data>
</object_doc>
<object_doc object_id="59" type="TextLine">
<object_pos h="11" w="127" x="14" y="670"/>
  <object_data>
    <ascii_data>Bonnefontaine dans le cadre
    des </ascii_data>
  </object_data>
</object_doc>
<object_doc object_id="67" type="TextLine">
```

figure 4 : ground-truth structure extracted from a PDF File

figure 5 : ground-truth structure visualization with Xmillum

Finally, our ground-truth dataset contains information concerning text lines, words (content and location) and images contained in the document. Thus we obtain a partial ground-truth that is sufficient for the first tests.

3.3. Building the benchmarking scenario

The benchmarking scenario is composed by three steps :

- Transformation of the PDF document into an image.
- Physical structure extraction using a page segmentation processing.
- Structure matching evaluation by extracting the corresponding ground-truth in the ground-truth dataset.

Transformation of the PDF document in an image is done by a processing that encapsulates a ghostscript command. At the present time, we have two segmentation algorithms, one based on a classical top down approach and the other one based on an hybrid approach[7]. Both algorithms produce a resulting XML structure which is matched with the ground-truth dataset to measure the regions overlap ratio. Ground-truth information is extracted from the dataset by using Xpath expressions which allows to select the desired corresponding structure.

Figure 6 shows the contract we defined for this node matching step. The major parameters are bold typed :

- The parameter named `KnowledgeBase` refers to the ground-truth file.
- The parameter named `XpathSelector` refers to an Xpath expression used to extract the desired objects from the ground-truth file.
- The parameter `SegmentedObjects` which is another Xpath expression, refers to the graphical objects obtained after the segmentation step.

The flexibility of Xpath expressions allows the user to select exactly what he needs, he can modify those selection expressions by choosing another kind of object (words for example) or by adding constraints (for example small areas may be filtered)

Node matching itself is done with Yanikoglu's method based on the ON pixels contained in a zone [9]. In order to ignore insignificant differences between the ground-truth regions and the segmented ones, only the black pixels content of the areas are taken into account

```
<process_property class_name="NodeMatch">
<service name="nodeAdd">
<handled_object>
  <object_doc type="BinaryImage"/>
   <process_config>
    <param type="ParamIn"
    name="KnowledgeBase" support="Data"
    param_value="gd_base.xml" info="tree
    base"/>
    <param type="ParamIn"
    name="XPathSelector" support="Data"
    param_value=
    "//object_doc[@type='TextLine']"/>
    <param type="ParamIn"
    name="SegmentedObjects"
    support="ObjectDoc" param_value=
    "object_doc[@type='TextLine']"/>
  </process_config>
```

```
<produced_object>
<object_data>
<feature name= "NodeMatching"/>
</object_data>
</produced_object>
</handled_object>
</service>
</process_property>
```

figure 6 : Contract of the Node Matching process

4. Conclusion and future works

Although many page segmentation evaluation problems are not yet addressed in this paper (blocks labeling, reading order, errors evaluation), we think that the DocMining architecture is well suited to tackle many aspects of ground-truthing and benchmarking. Indeed, DocMining's strong modularity can help building ground-truthing benchmarking scenarios according to users needs Editing and visualization tools for manipulating ground-truth datasets may be added as well. Moreover, processing modularity allows user to design their own performance evaluation algorithm. Therefore, the platform architecture allows future works to include errors evaluation processings and tools for adding new features to the datasets (labels, reading order).

In this paper, the solution we use to generate ground-truth and benchmarking scenarios for page segmentation is based on PDF documents. As shown on figure 2, PDF documents can be produced from XML data. The markup of an XML document often describes its logical structure. Therefore, a ground-truth dataset may be built with XML documents representing the logical structure and their associated PDF version corresponding to the physical structure.

5. References

[1] D. Carlisle, M. Goossens et S. Rahtz, "De XML à PDF via xmltex, XSLT et PassiveTeX" Cahiers GUTenberg n°35-36, pp. 79-114, 2000.

[2] M. Delalandre, S. Nicolas, E. Trupin and J.M. Ogier. "Symbols Recognition by Global-Local Structural Approaches, Based on the Scenarios Use, and with a XML Representation of Data", *International Conference on Document Analysis And Recognition (ICDAR)*, 2003.

[3] Ph. Dosch, K. Tombre, C. Ah Soon, G. Masini, "A complete system for the analysis of architectural drawings", *International Journal on Document Analysis and Recognition, 3(2)*, December 2000, 102-116.

[4] O. Hitz, L. Robadey, R. Ingold. An architecture for editing document recognition results using XML. Proceedings of the 4th IAPR International Workshop on Document Analysis Systems (DAS'2000), Rio de Janeiro, Brazil, December 2000.

[5] T. Kanungo, C. H. Lee, J. Czorapinski, and I. Bella. "TRUEVIZ: a groundtruth/metadata editing and visualizing toolkit for OCR". *In Proc. of SPIE Conference on Document Recognition and Retrieval,* Jan. 2001.

[6] S. Mao and T. Kanungo, "Software architecture of PSET: a page segmentation evaluation toolkit". *International Journal on Document Analysis and Recognition (4) 3*, 2002, 205-217.

[7] P.Parodi and G. Piccioli, "An efficient pre-processing of mixed-content document images for OCR systems", *Proceedings of the 13th International Conference on Pattern Recognition*, Volume: 3 , 1996, 778 -782 .

[8] T. A. Phelps and R. Wilensky, "The Multivalent Browser : A Platform for New Ideas", proc of Document Engineering 2001, Atlanta, Georgia, USA, 2001.

[9] B. A. Yanikoglu and L. Vincent, "Pink panther: a complete environment for ground-truthing and benchmarking document page segmentation,", *Pattern Recognition 31*, September 1998, 1191-1204.

Assuming Accurate Layout Information is Available: How do we Interpret the Content Flow in HTML Documents?

Hassan Alam and Fuad Rahman[§]
BCL Technologies Inc.
fuad@bcltechnologies.com

Abstract

Accessing multi-modal HTML documents, such as web pages, by devices of lower form factor is a growing practice. There are three problems of using a small handheld device for accessing web pages, the display area is small, the network speed is low, and the device capability is lower. Often the devices are unable to run IE or other browsers, and when they can, the browsers have lower levels of functionality. A direct consequence is that web pages written for desktop devices with HTML 4 with Cascading Style Sheets (CSS), image maps, and flash multi-media plug-ins are often not viewable in these smaller devices. In order to solve this problem, researchers have been using techniques such as transcoding and re-authoring to re-format web pages and re-flowing them to the small devices. Some of these techniques use secondary information about the HTML data structure and content association in terms of semantics of the web pages, but none has access to accurate layout information before it is rendered. Assuming we do have access to this type of information, we can take a whole new perspective to this web page re-authoring problem. This paper discusses some ideas about the possible use of accurate layout information while automatically re-authoring web pages.

1. Introduction

Web pages written in HTML do not provide adequate information about the final layout. This is only known when the page is rendered using one of the standard web page browsers such as IE®, Netscape® or Opera®. This is so as many of the elements in the web pages are dynamic. For example, the width of a column may vary based on the browser window size. This makes calculating accurate layout information very difficult. But what if we did have access to this information? How do we use this to improve web page re-authoring? This paper discusses some ideas about how we can exploit this information to produce better re-flow of web pages targeted to small handheld devices.

2. Related Work

There is a growing demand for viewing web pages on small screen devices. Mobile viewing allows keeping in touch with the rest of the world while on the move. So web page re-authoring can be of great interest. Another motivation is to summarize web content to help in rapid viewing, as time is a very important commodity and the amount of information available these days makes it impossible to browse through entire web sites [1]. Often there is a demand for alternative browsing, such as the use of voice [2, 3]. Most email browsing software now support HTML pages, which make the problem of universal email accessibility a problem of web page manipulation. Last, but not the least, web page manipulation is often required in order to extract content and transfer it to other formats, such as PDF and others. In this scenario, web page manipulation supported with document analysis techniques is the best choice.

Before we start, here is a very brief outline of past work. Over the years, researchers have proposed different solutions to the problem of web page re-authoring. The possible solutions can be categorized as:

- By Hand: Manipulate web sites by hand.
- Transcoding: Automatically replace tags with device and target specific tags. In this case, the content flow is identical to the flow in the HTML data structure [4].
- Re-authoring: Automatically capture the structure and intelligently re-flow content in a different flow other than in the original document. There are many variations of this approach:
    - Table of Content (TOC): the document is re-created based on extracted content and a *heading* is created. This heading hides a hyperlink, which if selected, can load up details associated with the headline [5].

[§] Corresponding author. The authors gratefully acknowledge the support of a Small Business Innovation Research (SBIR) award under the supervision of US Army Communications-Electronics Command (CECOM).

- Summarization: the content is not simply separated into separate layers; the textual part of the content is summarized using natural language techniques [6].
- Hybrid: TOC and summarization approaches are combined [1].

3. The Future: Interpreting Content Flow with Accurate Layout Information

None of the approaches mentioned above have access to accurate information about the layout of the web pages. Let us assume we do have access to this information. Let us also assume that Figure 1 shows a hypothetical web page with accurate layout information.

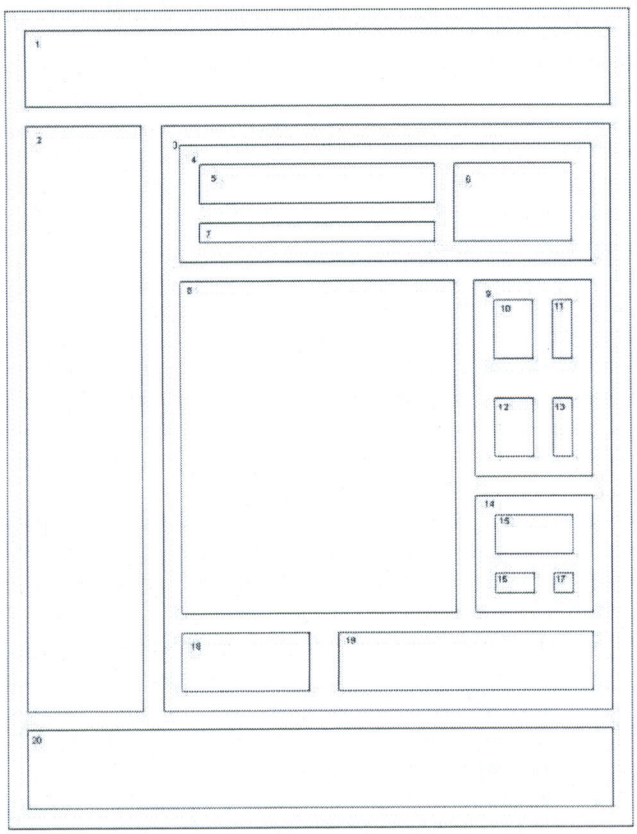

Figure 1: A sample hypothetical HTML page in terms of table structure

In Figure 1, each rectangle represents a region, which is in turn, is represented internally as an HTML table construct. As the HTML code dictates, the actual content is in the columns of these tables. It is a safe assumption to make as most web pages are created using the table construct. Assuming we know the absolute position of these tables, tagged 1, 2, etc. in the figure, let us design a solution to extract the content flow of this web page without knowing anything about the semantic relationship of the actual content. It should be pointed out in this regard that this assumption automatically makes the content flow extraction algorithm free from any dependence on the language used in the web site.

The following is an outline of the proposed content flow extraction algorithm. For ease of description, we assume the Table of Content (TOC) approach for automatic web page re-authoring [7]:

- Form the sides belonging to the tables.
- Scanning order => Top to Bottom, left to Right:
  - Pair up sides,
  - Identify and mark the tables.
- Calculate xPreferance list:
  - Sort the top most sides of each table in terms of increasing x value, i.e. tables beginning at left are placed before tables beginning at right etc., and assign sequential weights beginning with 0, with the final weight equal or less than the number of total tables on that page.
  - Tables beginning at the same x value are given the same weight.
- Calculate yPreferance list:
  - Sort the left most sides of each table in terms of increasing y value, i.e. tables beginning at top are placed before tables beginning at bottom etc., and assign sequential weights beginning with 0, with the final weight equal or less than the number of total tables on that page.
  - Tables beginning at the same y value are given the same weight.
- Perform a Proximity Analysis:
  - For each side of a table, find immediate neighbors.
- Determine the Inclusion Criterion:
  - Calculate offset of each side to the closest side, which is larger than the offset.
  - Determine which table satisfies having those four sides as its sides.
- Calculate the offset of each side of each table on the outermost table.
- Calculate area of each table.
- Calculate table hierarchy (tree) based on Inclusion Criterion and Proximity information beginning from the outer-most table.
- Preference (Order of Display):
  - Tables surrounded by other tables.
  - Tables appearing near the top, but not the top most.
  - Tables with higher area.

Table No	Position on x axis	Position on y axis	Inclusion	Proximity	Area
1	1	1	0	2, 3	?
2	1	2	0	1, 3, 20	?
3	2	2	0	1, 2, 20	?
4	3	3	3	8, 9	?
5	4	4	4	6, 7	?
6	5	4	4	5, 7	?
7	4	5	4	4, 5, 6	?
---	---	---	---	---	---
---	---	---	---	---	---

Table 1: A map based on Figure 1

- Level of Table of Content (TOC) determining at which level should each label representing each table be displayed.
    - Labels coming from the tables having the same inclusion criterion should be at the same level.
- Merging Criteria (when we can safely merge tables):
    - Only tables which have the same inclusion criteria can be merged provided:
        - Only at the lowest levels in the hierarchy
        - Only when they share identical sides
        - Can happen in both left/right or top/bottom neighbors
        - But not if a border exists

This algorithm provides a map as shown in Table 1. This shows a partial analysis of the example web page. This map can decide how the content is to be displayed and in which order, based on the best-guess scenario by exploiting accurate layout information of the web page.

In the absence of such information, the content would have been displayed based on other criteria, such as semantic relationship [8], absolute size of the tables in terms of the number of words, and others. This algorithm can also help us in picking the most important part of the content based on geometric position. For example, Tables 4 and 8 are strategically placed in the upper middle part of the page, so chances are that they are the centerpieces of the page. In addition, this also allows us to label other tables, such as Table 2 is a sidebar, Table 20 is the lower bar, and so on.

It is also interesting to see how the semantic relatedness factor [8] helps in determining which tables should be merged. This is something that we will explore in the future.

4. Extension of the Argument

Estimating accurate layout information from HTML pages may be hard, but not impossible. This will require writing a parser that will be based on geometric analysis of web pages and will be able to deliver considerably richer information about the content, their relationships with each other, and their relative importance within the context of that page compared to what information is available from a traditional parser.

The next question will be to represent this rich set of information gathered from the geometry-based parser. The information collected from such a geometry based parser will help in determining the "sense" and meaning of the main "message", which is turn will help in making the re-authoring, more accurate. Recently XML is being successfully used in many applications to mark up important information according to application-specific vocabularies. To properly exploit the flexibility inherent in the power of XML, industry associations, e-commerce consortia, and the W3C need to develop their own vocabularies to be able to transform information marked up in XML from one vocabulary to another. Two W3C Recommendations, XSLT (the Extensible Stylesheet Language Transformations) and XPath (the XML Path Language), meet that need. They provide a powerful implementation of a tree-oriented transformation language for transmuting instances of XML using one vocabulary into either simple text, the legacy HTML vocabulary, or XML instances using any other vocabulary imaginable. It is probably better to use the XSLT language, which itself uses XPath, to specify how an implementation of an XSLT processor is to create a desired output from a given marked-up input.

This representation of information will probably take a form where the content of each page is represented in a

hierarchical structure along with additional information concerning them. Specific information that can be harvested from this may include:

- Exact location of each block, in rectangular coordinates, equivalent to rendition using a standard browser.
- Size of each block of content.
- Type of content, e.g. text, graphics etc.
- Weight of content, in terms of size and placement within a page.
- Continuity information, derived from physical association in terms of geometrical collocation.
- Classification of content into a set of pre-defined classes, e.g. main story, sidebars, links and so on.
- Linkage information from the XML representation, indicating the layers of information that can be hidden at a level of summary. This can represent the content in many levels, but more than two or three levels are unsuitable for easy navigation.

5. Conclusion

This is a concept paper discussing a specific pathway to the future of web page re-authoring provided accurate layout information is available. This in no way represents a state of the art discussion about the possible use of layout information. Rather, it focuses on one small part within an array of possibilities. It will be interesting to discuss other possibilities in this space during the DLIA workshop.

References

1. Alam, H., Hartono. R., Kumar, A., Tarnikova, Y., Rahman, F. and Wilcox, C., "Web Page Summarization for Handheld Devices: A Natural Language Approach", 7th Int. Conf. on Document Analysis and Recognition (ICDAR'03). In press.
2. Brown, M., Glinski, S. and Schmult, B., "Web page analysis for voice browsing", First International Workshop on Web Document Analysis (WDA2001), Seattle, Washington, USA, September 8, 2001.
3. Takagi, H. and Asakawa, C., "Web Content Transcoding For Voice Output", Technology And Persons With Disabilities Conference, 2002.
4. Hori, M., Mohan, R., Maruyama, H. and Singhal, S., "Annotation of Web Content for Transcoding", E3C Note. http://www.w3.org/TR/annot.
5. Jones, M., Marsden, G., Mohd-Nasir, N. and Buchanan, G., "A site based outliner for small screen Web access", Proceedings of the 8th World Wide Web conference, pages 156-157, 1999.
6. Berger, A. and Mittal, V., "OCELOT: A System for Summarizing Web Pages", Research and Development in Information Retrieval, pages 144-151, 2000.
7. Rahman, A., Alam, H. and Hartono, R., "Content Extraction from HTML Documents", Proceedings of the Web Document Analysis Workshop (WDA01), Seattle, USA, 8 September, 2001, pp. 7-10. Online Proc. at http://www.csc.liv.ac.uk/~wda2001/.
8. Alam, H., Rahman, F. and Tarnikova, Y., "Web Document Analysis: How can Natural Language Processing Help in Determining Correct Content Flow?", WDA 2003. Submitted.

Background pattern recognition in multi-page PDF document

Hui Chao
Hewlett-Packard Labs
1501 Page Mill Road, ms 1203, Palo Alto, CA 94304
hui.chao@hp.com

1. Introduction

Documents are often edited on templates or with certain background patterns. For example, Microsoft Power Point allows the user to build their presentations based on a slide master or a title master. Those patterns are often used to enhance the appearance of documents and to help to communicate a message more clearly and consistently. Background or template patterns here are the page components that if being removed, will not alter the flow or meaning of a document. The elements in the background pattern or template can be static, staying exactly the same from page to page like a company logo in Power Points slides. The elements can be also dynamic, such as page number. They appear on some of the pages or some of their features such as shape, color or orientation vary to make the elements more suitable for the content or the orientation of the page, while other features of the elements stay the same to preserve the consistency in style. Although the information about the templates or background patterns may be available in the original document and with the original editing software, it is often lost when the document is converted to a printer-ready format such as the Portable Document Format (PDF)(1). Since the printer-ready format is the last stage of a publish workflow and is platform independent in the case of PDF it is often the only available format for a document. Without the ability to distinguish the template or background pattern of a document and its foreground content, document reuse can be compromised. For Example, the background graphics pattern may interfere with the foreground illustration during graphics extraction (2), and converting a document from one format to another may require treating background differently from the foreground content.

There have been many approaches to understand the structure of documents for scanned documents (3-8). Most of the studies were concentrated on the understanding of the text blocks on a page. There also have been approaches to separate the foreground and background of a document for compression reason (9, 10), where text and drawing are considered to be the foreground. A couple of research groups have worked on PDF documents (10-13) to understand the semantics and layout of documents. The documents being analyzed were mostly newspapers, technical manuals and business letters. Elements overlay was not considered.

In this paper we present a technique that separates the template or background pattern of a document from its foreground content in a multi-page PDF document. It analyzes the document as a whole, compares elements across pages, sets soft constraints for a similarity check and obtains background pattern or template for each page in a document. This technique could be used as a preprocessing step for document structure understanding mentioned above.

2. Methods

First we preprocess PDF documents by flattening the PDF page structure and segmenting text elements. In order to search for not only the static elements but also the dynamic elements, we selectively obtain and compare features of elements from page to page. Similar elements across pages are marked, recorded, and analyzed to check if they belong to background pattern.

2.1 Preprocessing

2.1.1 Flatten page structure

A PDF page display list consists of page elements (14, 15). Elements can be simple types like text, image or graphic elements. They can also be compound types like container elements and XObject form elements, which themselves may contain a list of elements including text, image and graphic elements or even another level of container and form elements. Elements in a compound element are extracted and replaced in the page display list to substitute for the compound element while the layout order of the elements is preserved. The extraction and substitution process continues until there are no more compound elements on the page display list.

2.1.2 Text segmentation

Text Elements in PDF often contain unrelated bodies of text and text with different attributes (1, 15). Little semantic information can be derived from the way text

elements being organized. During preprocessing, we break the text elements into homogenous segments of text, in which all the characters have the same font and font size.

2.1.3 Finding the smallest bounding boxes that enclose the vector graphics elements

The element's bounding box, the smallest rectangle that encloses each page element, is used to represent the size and position of a page element. The bounding box of a graphic element is calculated by finding the smallest rectangle that encloses all the sub-paths in a graphics element. There are three types of drawings or paths: lines, rectangles and Beize Curves. The coordinates for the bounding boxes of lines and rectangles can be easily obtained from the control points, the starting and ending points of a path. In the case of the Beize curve:

$$R(t)=(1-t)^3P_0 + 3t(1-t)^2P_1+3t^2(1-t)P_2 + t^3P_3,$$

Where $0<t<1$, $P_0(x_0, y_0)$, $P_3(x_3, y_3)$ are the starting point and ending point and $P_1(x_1, y_1)$, $P_2(x_2, y_2)$ are the two control points.
The coordinates of the bounding boxes are obtained by searching the points with maximum and minimum values in the horizontal and vertical directions along the path of the curve.

2.2 Defining the structure of element's attributes for the comparison

The data structure of elements attributes or features is defined as:

```
{
        float           Width,
        float           Height,
        float           CenterX,
        float           CenterY,
        float           DistToEdge,
        int             Type,
        bool            flag,
        ImageAttrs      imageAttr,
        GraphicAttrs    graphicAttr,
        TextAttrs       textAttr
}
```

Where

- Width and Height are width and height of the bounding box of an element. They are normalized against the width and height of size of the page.
- CenterX, CenterY are the coordinates of the center of the bounding box in horizontal and vertical direction.
- DistToEdge is the smallest distance from the top or bottom edge of the bounding box to the top or bottom edge of the page. It is used for finding the page number, header and footer.
- Flag is used to indicate if the element is marked as common.
- Type represents the type of the object. Depending on the type of the object, image, graphic, and text attributes could include different information such as image data stream, graphics shape, and text font information.

2.3 Searching for similar elements across page

In multi-page documents, elements for the background pattern tend to appear on every page, every other page, or some of the pages in a document. Elements are classified as "common" based on the recurrence rate of elements with similar attributes such as width, height, type, and position on the page. Following are the steps used in searching elements of the background pattern.

Assuming the total number of pages in a document is N. ELEM[ ][ ] is the matrix of the page elements attributes structure. The first index is the page number and the second index is the element number:

- Select a reference element ELEM[i0][j0] that has not marked as "common", This process starts at the first element on the first page, that is i0=0, j0=0; continues to the next elements on the first page, ELEM[0][1], and so on. When all unmarked elements on the current page have been referenced, it moves to the first unmarked elements on the next page.
- Go through elements on the rest of pages in the document, on each page searching for first element that are similar to the reference element, that is, on each subsequent page, find and mark the first similar element. The similarity is defined as follows:
    a. Same types of elements that are less than 1% different in size, less than 1% different in position, and with same data stream or same shape.
    b. For small text elements whose size is less than 1% of the size of the page, that are less than 1% different in height and distance to the edge, less than 5% different in width, and whose distance to the top or bottom edge is within 10% of the height of the page;

  The last criterion is for finding headers, footers and page numbers.
- If similar elements occur on a third of the pages (N/3) in a documents, or more than 2 pages in a document in the case of N/3 <2, the elements are

marked as common elements and classified as background elements.

3. Results and Discussion:

We have tested our technique on 7 different documents ranging from marketing brochures, interior design books, a catalogue, and a weekly journal, for total of 130 pages. As shown in table 1, among all the pages, there are four false negative cases where the header, footer or the pattern is slightly different from ones on the rest of the pages in the document. There are three false positive cases, where one pattern belonging to the foreground content is accidentally repeated on three different pages. Figure 1 shows examples of background pattern recognition and extraction from a catalogue. Figure 1a shows the original pages from the document, and figure 1b shows the extracted background pattern from the document.

We have also tested this algorithm on documents that are a collection of tickets for different tourist attractions and supermarket weekly advertisements. Because of the repetitive text patterns in the content, text elements that belong to content were mistakenly classified as background pattern elements. For those types of document, further restriction of the elements type may be required.

Documents	Total number of pages	Number of false negative pages(missing Background elements)	Number of false positive pages(picking elements in the foreground content)
Catalogue	21	2, missing a header which is slightly longer than ones in the rest of pages	0
Book	16	0	0
Marketing brochure	4	0	0
Marketing brochure	10	0	3, adding one graphic pattern in content to the background pattern)
Book	22	1, the footnote pattern is slightly longer than the ones in the rest of pages	0
Marketing brochure	2	0	0
Marketing brochure	32	0	0
Weekly Journal	24	1, missing footer which is longer and in slightly different position than ones in the rest of the pages.	0

Table 1: Results from background pattern extraction

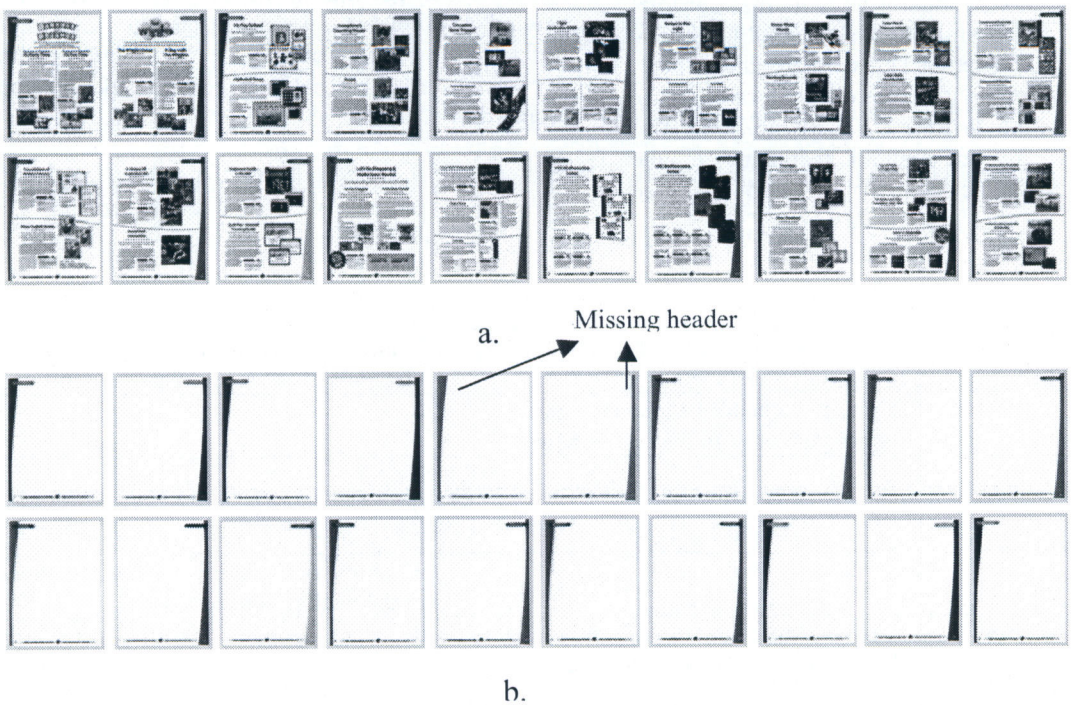

Figure 1. (a) The original pages of a catalogue. (b) The background pattern identified. Missing header can be seen on page 5 and 6, because it is slightly bigger than header pattern on the rest of the pages.

4. References

1. "PDF Document Layout Study with Page Elements and Bounding Boxes", Chao, H, Beretta, G, Sang, H. Proceedings of Workshop on Document Layout Interpretation and its Applications, 2001(DLIA2001).
2. "Graphics Extraction in PDF Document", Chao. H., Proceedings of Electronic Imaging, 2003, SPIE vol. 5010, no. 317.
3. "Automatic knowledge acquisition for spatial document interpretation", Walischewski, H., Proceedings of the Fourth International Conference on Document Analysis and Recognition, (pp. 243-247), 1997, Los Vaqueros, CA: IEEE Computer Society Press.
4. "Computer understanding of document structure" Dengel, A.,Dubiel, F. International Journal of Imaging Systems and Technology, 1996, vol.7, no.4.,
5. "Incremental acquisition of knowledge about layout structures from examples of documents", Kise, K. Proceedings of the Second International Conference on Document Analysis and Recognition,1993, (pp. 668-671), Los Vaqueros, CA: IEEE Computer Society Press.
6. "Comparison and classification of documents based on layout similarity." Hu, J., Kashi, R., Information Retrieval, 2(2):227-243, May 2000.

7. "Content Features for logical document labeling", Liang, J., Doermann, D., Proceedings of Electronic Imaging, 2003, SPIE Vol 5010.
8. "A Pattern-Based Method for document structure recognition", Robadey, L., Hitz, O., Ingold R., Proceedings of Workshop on Document Layout Interpretation and its Applications, 2001(DLIA2001). (DLIA2001).
9. "DjVu: a compression method for distributing scanned documents in color over the internet", Cun, Y. L., Bottou, L., Haffner, P., Howard P., Color 6, IST 1998,
10. "A scalable DSP-architecture for high-speed color document compression", Thierschmann, M., Barthel, K, Martin, U., Proceedings of SPIE, Vol. 4307, 2001.
11. "AIDAS: Incremental logical structure discovery in PDF documents". Anjewierden, A., Proceedings of the International Conference on Document Analysis and Recognition, 2001. (ICDAR2001).
12. " Document analysis of PDF files; method, results and implications", Lovegrove, W.S., Brailsford, D.F. Electronic publishing: origination, dissemination and Design, Vol. 8 no. 2-3.
13. "Towards structured, block-based PDF", Smith, P.N., Brailsford, D.F. Electronic publishing: origination, dissemination and Design, Vol. 8 no. 2-3.
14. "Adobe Acrobat core API reference", Technical Notes #5191, Adobe Systems Incorporated.
15. "PDF Reference" Second Edition, Adobe Systems Incorporated.