WISDOM++: An Interactive and Adaptive Document Analysis System

Oronzo Altamura Floriana Esposito Donato Malerba
Dipartimento di Informatica, Università degli Studi di Bari
via Orabona 4, I-70125 Bari, Italy
{*altamura* | *esposito* | *malerba*}*@di.uniba.it*

Abstract

WISDOM++ is a document analysis system whose main design requirements are real-time user interaction and adaptivity. This paper presents the two-phased skew estimation algorithm and the adaptive document block segmentation and classification techniques. An evaluation of the performance of some of these tasks is also conducted according to a benchmarking procedure.

1. Introduction

WISDOM++ is a document analysis system that operates in four steps: *document analysis*, *document classification*, *document understanding*, and *text recognition* with an OCR.[1] One of its distinguishing features is the use of a rule base in order to support some tasks performed in the first three steps. The rule base is automatically built from a set of training documents using symbolic machine learning tools and techniques, which make the system highly *adaptive* [1]. *Real-time* user interaction is another relevant feature that affected the design of the whole system.

From a functional point of view, WISDOM++ presents several novelties with respect to its predecessor, PLRS [2], namely: 1) the two-phased skew estimation; 2) the application of decision tree learning tecniques to the *block classification* problem, that is the separation of text blocks from graphics; 3) the extension of the learning techniques for document classification and understanding in order to deal with both numeric and symbolic data; 4) the treatment of *multi-page* documents, each of which is a *sequence* of pages; 5) the generation of the corresponding XML format.

This paper illustrates the first two novelties: the deskewing algorithm is described in Section2, while the adaptive block classification is reported in Section 3. Finally, some experimental results are reported in Section 4.

[1] WISDOM++ is a newer version of the system WISDOM (Windows Interface System for DOcument Management), originally written in C [3].

2. Data capture and preprocessing

In WISDOM++ each page of a multi-page document is optically scanned with a resolution of 300 dpi and thresholded into a binary image, which is stored in TIFF format. The image is associated with a coordinate system whose origin is in the top-left hand corner; the X-coordinate increases from the leftmost to the rightmost column, while the Y-coordinate increases from to uppermost to the lowermost row.

The segmentation of the page is performed by a top-down method, which is quite fast but generally ineffective when applied to skewed documents. Consequently, the skew angle has to be estimated and corrected by inverse rotation of the document image.

The skew angle of a document image I is the orientation angle θ of its text baselines: It is positive when the image is rotated counter-clockwise, otherwise it is negative. In WISDOM++ the estimation $\hat{\theta}$ of the actual skew angle θ is obtained as composition of two functions: $S(I)$, which returns a *sample region* R of the document image I, and $E(R)$, which returns the *estimation* of the skew angle in the sample region R. The selection of a sample region is necessary to reduce the computational cost of the estimation step. The system always shows the selected region of the bitmap and allows the user to change it interactively. WISDOM++ is not able to autonomously manage documents with many unknown skews $\theta_1, \theta_2, \ldots, \theta_n$ for text lines, since the function S returns a text region whose skew is not guaranteed to be the most frequently occurring one (*dominant skew*).

In order to select the sample region, WISDOM++ computes both the horizontal projection profile H of the document image and the average number of pixels per row (*avpx*). Then, it extracts a set of *regions* from H: A region R_i is a sequence of adjacent rows in H whose height is greater than *avpx/4*. In this way, only regions with prominent peaks will be considered, since $E(R_i)$ is more likely to be close to the true skew angle θ. Each region is associated with one of the following three classes: Horizontal line, text and image. The classification is based on the height of the region:

$height(R_i) < Text\_min$ ➔ horizontal line,
$Text\_min \le height(R_i) \le Text\_max$ ➔ text,
$height(R_i) > Text\_max$ ➔ image.

The constants *Text_min* and *Text_max* are set to 24 and 125, respectively, in order to take into account 6-point to 30-point text lines. Since the focus is on the estimation of the skew angle of text regions, the system selects, if any, the text region R_i with the maximum average density of black pixels per row. Otherwise, the system returns the region, classified as horizontal line or image, satisfying the following conditions: Its base is smaller than 310 pixels and it has the maximum average density of black pixels per row.[2]

Once the sample region *R* has been selected, *E(R)* is computed. Let H_θ be the horizontal projection profile of *R* after a virtual rotation of an angle θ. The histogram H_θ shows sharply rising peaks with base equal to the character height when text lines span horizontally, while it is characterized by smooth slopes and lower peaks in the presence of a large skew angle. This observation is mathematically captured by a real-valued function,

$$A(\theta) = \sum_{j \in R} H_\theta^2(j),$$

which has a global maximum at the correct skew angle. Thus finding the actual skew angle is cast as the problem of locating the global maximum value of A(θ). Since this measure is not smooth enough to apply gradient techniques, the system adopts some peak-finding heuristic. Initially, it takes thirty-two samples of A(θ) for rotation steps of 20 pixels.[3] Then it selects the angle θ' maximizing A(θ) and rotates the sample region at finer steps (10 pixels), starting from $\theta' - 30$, until a peak is found. The final estimate of the skew angle is determined by computing the vertex of the parabola interpolating three points around the peak in the space (θ, A(θ)). The main advantage of this approach is its low computational cost with respect to methods based on Hough transforms, Fourier transforms, or connected components. The estimated skew angle is used as default value when the user asks the system to rotate the document. The user can repeat the loop "skew estimation - document rotation" until satisfying results are obtained.

Another parameter computed during the preprocessing phase is the *spread factor* of the document image. It is defined as the ratio of the average distance between the regions R_i (*avdist*) and the average height of the same regions (*avheight*). In quite simple documents with few sparse regions, this ratio is greater than 1.0, while in complex documents with closely written text regions the ratio is lower than the unit. This factor is used to define some smoothing parameters of the segmentation algorithm.

At the end of the preprocessing phase, the resolution of the document image is reduced from 300 to 75 dpi: This is deemed a reasonable trade-off between the accuracy and the speed of the segmentation process. Moreover, noisy black specks on white background are filtered out in this way.

3. Adaptive block classification

WISDOM++ segments the reduced document image into rectangular blocks by means of an efficient variant of the Run Length Smoothing Algorithm [5]. The segmentation algorithm returns blocks that may contain either textual or graphical information. In order to facilitate subsequent document processing steps, it is important to classify these blocks according to the type of content: *text block*, *horizontal line*, *vertical line*, *picture* (i.e., halftone images) and *graphics* (e.g., line drawings).

The classification of blocks is performed by means of a decision tree automatically built from a set of training examples (blocks) of the five classes. The choice of a "tree-based" method is due to its inherent flexibility, since decision trees can handle complicated interactions among features and give results that can be easily interpreted. The numerical features used by the system to describe each block are the following: *height* (height of the reduced image block); *length* (length of the reduced image block); *area* (*height*length*); *eccentricity* (*length/height*); *blackpix* (total number of black pixels in the reduced image block); *bw_trans* (total number of black-white transitions in all rows of the reduced image block); *pblack* (percentage of black pixels in the reduced image block, *blackpix/area*); *mean_tr* (average number of black pixels per black-white transition, *blackpix/bw_trans*); *F1* (short run emphasis); *F2*: (long run emphasis); *F3* (extra long run emphasis).[4]

In order to make WISDOM++ interactive and adaptive, users are allowed to train the system on-line: When they are dissatisfied with the classification made by the decision tree, they can ask the system to revise the classifier without starting from scratch. In this way, some blocks, such as the logo of a business letter, can be considered text for some users and graphics for others. The decision tree learning system currently used by WISDOM++ is ITI 2.0 [7]. It can operate in three different ways. In the *batch* mode, it cannot change the decision tree when some blocks are misclassified, unless a new tree is generated from scratch using an extended training set. In the other two modes ITI supports the *incremental* construction of decision trees by revising the current tree in response to each newly observed training

[2] When no full region satisfying these conditions exists, a sub-region of exactly 310 pixel is selected.

[3] For an A4-sized document with 2,496 columns, this corresponds to a rotation of an angle equal to $arctan(20/2496) \approx 0.46°$. The system can detect skews smaller than ±7.2°, thus only 32 rotations of step 0.46° are possible.

[4] Computed using the following thresholds: T1=10 and T2=20 [8].

instance. In particular, in the *normal* operation mode, it first updates the frequency counts associated to each node of the tree as soon as a new instance is received. Then it restructures the decision tree according to the updated frequency counts. In the *error-correction* mode, frequency counts are updated only in case of misclassification of the new instance. The main difference between the two incremental modes is that the normal operation mode guarantees to build the same decision tree independently of the order in which examples are presented, while the error-correction mode does not.

Our preference for ITI 2.0 is also due to its exclusive capability to handle numerical features. From the practical point of view, the main problem we observed is that ITI creates large files since it needs storing examples in the nodes of the induced tree. This inefficiency can be contained when the system operates in the *error-correction* mode, since training examples are stored only in case of misclassification. Currently, two new functions have been added to the interface of WISDOM++: The interactive correction of the results of the classification, and the incremental update of the block classifier.

4. Benchmarking of WISDOM++

By considering WISDOM++ as a chain of modules, we intend to assess the absolute performance of two of them, namely the skew evaluation module and the block classification module, by following the evaluation method proposed by Märgner *et al.* [4].

A set of 112 real, single-page documents have been considered as input data. Documents are distributed as follows: Thirty are the first page of articles appeared on the proceedings of the International Symposium on Methodologies for Intelligent Systems (*ISMIS94*),[5] twenty-eight are the first page of articles published on the proceedings of the 12th International Conference on Machine Learning (*ICML95*), thirty-four are the first page of articles published on the IEEE Transactions on Pattern Analysis and Machine Intelligence (*TPAMI*) during the period January-June 1996, and twenty documents of different type or published on other proceedings, transactions and journals (*Reject*). The text is organized into one column for the first class of documents and into two columns for the second and third class. No uniform layout can be detected for the documents in the fourth class.

Benchmarking of the deskew algorithm is performed in two different ways. Firstly, the optimal output (*ground truth*) is determined by a human expert, who decides which is the right skew angle of the scanned document image. This *real* assessment may contain some imprecision, that we tried to keep under control by providing the expert with two effective tools in the interface: Zooming functions and a straight-edge. The second way aims at having an error-free *ideal* assessment through artificial rotations of the document image for some known angles. In both cases, the *evaluation function* computes the error made with respect to the real/ideal assessment value.

Results concerning the real assessment are reported in Table 1, where all measurements are absolute values of angles expressed in degrees. The following observations should be made. Documents have been scanned by a careful user, who placed documents on the flat surface of a scanner with a skew less than three degrees. The smallest non-null skew angle that can be detected for A4 documents is 0.023°, which corresponds to a tilt of one pixel in a bitmap with 2496 columns. For single-column documents (ISMIS94), the error made by WISDOM++ is two pixels on average, while for documents organized in two columns (ICML95 and TPAMI), it is about nine pixels on average.

Table 1: Errors made w.r.t. the ground truth.

	Ismis	Tpami	Icml	Reject
Average error	0,0470	0,2166	0,2045	0,1120
Average Real Skew	0,3115	0,4530	0,5130	0,3659
Standard Deviation	0,2827	0,3489	0,5121	0,3896

Document images correctly rotated by the expert are used in the second benchmarking of the deskew algorithm. Each of them is rotated at various degrees using the rotation algorithm implemented in WISDOM++. The absolute errors averaged on all documents of a class are reported Table 2.

A comparison with other published results is not easy. A similar benchmarking was performed by Smith [6], whose best results seem not to be different from ours. Our experimental design allows us to observe that the skew estimation procedure exhibits a good performance for single-column documents, but it is not always reliable for documents organized in two or more columns. This limit is more evident with clockwise rotations, and is generally due to the difficulty in selecting a good sample region. Moreover, the error generally increases with the size of the skew angle, so that for a relatively large tilt it would be necessary to repeat the deskew process more than once. As to the time performance, which is a critical factor for a real-time system, it is always lower than 0,41 s on a Pentium PC 200MMX with 64Mb SDRam.

In order to test the performance of the block classifier, the set of documents has been split into a training set (70%) and a test set (30%) according to a stratified random sampling. The number of training blocks is 9,429 while the number of test blocks is 4,670. Three experiments have been organized. ITI 2.0 has been trained in the batch mode in the first experiment, in the pure error-correction mode in the second, and in a mixed mode

[5] Published by Springer Verlag in the series "Lecture Notes in Artificial Intelligence," Vol. 869.

in the third (incremental mode for the first 4,545 examples and error-correction mode for the remaining 4,884).

The main characteristics of the learned trees are reported in Table 3, where the last column refers to the number of examples stored in the nodes of the induced trees.The decision tree built in the batch mode takes more than 24Mb, since all instances have to be stored, while the decision three obtained in the error-correction mode requires only 982Kb. The total number of instances incorporated in the third experiment is 4,670, where 4,545 are the training examples used in the first incremental step, while the remaining 125 have been incorporated in the subsequent error-correction step. Nevertheless, this difference in tree size corresponds to a very little difference in predictive accuracy estimated on the independent test set (see Table 4). This justifies the use of the decision tree built according to the error-correction mode in this application.

Table 3. Main characteristics of the learned decision trees.

	Size Kb	No. Nodes	No. leaves	No. incorporated examples
Batch	24,320	229	114	9,429
Pure Error-correction	982	159	79	277
Mixed	13,150	235	117	4,545+125

Table 4. Predictive accuracy of the learned decision trees.

	ISMIS94	ICML95	TPAMI	Reject	Total
Batch	95.78	97.74	98.26	97.00	97.48
Pure EC	97.05	98.39	98.01	95.06	97.45
Mixed	95.99	97.63	98.18	97.35	97.51

The learned trees are not shown in the paper because of their huge size. We limit ourselves to report that the set of features actually used in the decision tree built by the pure error-correction procedure does not contain two features, namely width of a block and F3. The latter exclusion is probably due to the documents considered in this benchmarking, with few occurrences of large text blocks for which the long run emphasis actually helps.

Acknowledgments

The authors thank Giacomo Sidella and Ignazio Sardella for their help in conducting benchmarking.

References

[1] F. Esposito, D. Malerba, and G. Semeraro, "Automated acquisition of rules for document understanding," *Proc. of the Second Int. Conf. on Document Analysis and Recognition*, 650-654, IEEE Computer Society Press, 1993.

[2] F. Esposito, D. Malerba, and G. Semeraro, "Multistrategy learning for document recognition," *Applied Artificial Intelligence*, 8, 1, 33-84, 1994.

[3] D. Malerba, F. Esposito, G. Semeraro, and L. De Filippis, "Processing paper documents with WISDOM," in M. Lenzerini (Ed.), *AI*IA 97: Advances in Artificial Intelligence*, LNAI - 1321, Springer, 439-442, 1997.

[4] V.F. Märgner, P. Karcher, and A.-K. Pawlowski, "On benchmarking of document analysis systems," *Proc. of the Fourth Int. Conf. on Document Analysis and Recognition*, 331-336, IEEE Computer Society Press, 1997.

[5] F.Y. Shih, and S.-S. Chen, "Adaptive document block segmentation and classification," *IEEE Trans. on Systems, Man, and Cybernetics - Part B*, 26, 5, 797-802, (1996).

[6] R. Smith, "A simple and efficient skew detection algorithm via text row accumulation," *Proc. of the Third Int. Conf. on Document Analysis and Recognition*, 1145-1148, IEEE Computer Society Press, 1995.

[7] P.E. Utgoff, "An improved algorithm for incremental induction of decision trees," *Proc. of the Eleventh Int. Conf. on Machine Learning*, Morgan Kaufmann, 318-325, 1994.

[8] D. Wang and R.N. Srihari, "Classification of newspaper image blocks using texture analysis," *Computer Vision, Graphics, and Image Processing*, 47, 327-352, 1989.

Table 2. Average absolute error in the skew angle for the four classes.

Counter-clockwise					Clockwise			
Reject	**TPAMI**	**ICML95**	**ISMIS94**	**Degrees**	**ISMIS94**	**ICML95**	**TPAMI**	**Reject**
0,1260	0,2516	0,2303	0,0377	**0,5**	0,0390	0,3563	0,2384	0,1798
0,1928	0,5957	0,6265	0,0514	**1,0**	0,0505	0,6099	0,5119	0,3670
0,2214	0,2049	0,4495	0,0699	**1,5**	0,0709	0,4096	0,2846	0,2630
0,1639	0,0736	0,1841	0,0606	**2,0**	0,0562	0,1505	0,0743	0,1040
0,1587	0,1097	0,5090	0,0526	**3,0**	0,1123	0,4837	0,0789	0,5205
0,4534	0,2639	0,2781	0,0734	**4,0**	0,1318	0,5521	0,2180	0,7142
0,8134	0,5126	0,4883	0,1482	**5,0**	0,5046	1,3710	0,1220	0,8743
1,3723	0,5874	0,5403	0,4071	**6,0**	0,4546	1,7466	0,4971	0,6652
1,3564	0,6439	1,4304	0,4440	**7,0**	0,4340	2,2474	1,2674	0,7813