A Machine Learning approach for Graph-based Page Segmentation

Ana L. L. M. Maia[*][†], Frank D. Julca-Aguilar[*] and Nina S. T. Hirata[*]
[*]Department of Computer Science, Institute of Mathematics and Statistics
University of São Paulo (USP)
São Paulo, Brazil
[†]Department of Exact Science, State University of Feira de Santana (UEFS)
Feira de Santana, Brazil

***Abstract*—We propose a new approach for segmenting a document image into its page components (e.g. text, graphics and tables). Our approach consists of two main steps. In the first step, a set of scores corresponding to the output of a convolutional neural network, one for each of the possible page component categories, is assigned to each connected component in the document. The labeled connected components define a fuzzy over-segmentation of the page. In the second step, spatially close connected components that are likely to belong to a same page component are grouped together. This is done by building an attributed region adjacency graph of the connected components and modeling the problem as an edge removal problem. Edges are then kept or removed based on a pre-trained classifier. The resulting groups, defined by the connected subgraphs, correspond to the detected page components. We evaluate our method on the ICDAR2009 dataset. Results show that our method effectively segments pages, being able to detect the nine types of page components. Furthermore, as our approach is based on simple machine learning models and graph-based techniques, it should be easily adapted to the segmentation of a variety of document types.**

I. Introduction

Documents are constituted by structural elements arranged both spatially and hierarchically following some layout to facilitate understanding by humans. Layout analysis is therefore an important step in machine based document understanding, and it strongly depends on the detection of structural elements contained in the documents. When restricted to document pages, structural elements correspond to the page components such as blocks of text, tables, graphics, photos, among others. Recognizing these components is in general referred to as page segmentation or document image segmentation [1]. In this work we address the page segmentation problem.

The page segmentation problem is sometimes divided into two parts [2]: (i) partitioning an image into homogeneous regions according to some criteria, where each region is regarded as a page component, and (2) classifying each of these regions to attribute them a class label. Page segmentation is also sometimes referred to as page layout analysis [3], [4]. To avoid confusion, we use the term page segmentation to refer to the problem of recognizing, i.e., detecting and labeling, the structural elements of a document page. We understand layout analysis as a broader process that not only performs page segmentation but also establishes the spatial and hierarchical relation among the page components (for instance, reading order, or a hierarchical relation between a text and a diagram when the text is a sentence inside the diagram). Examples of page segmentation, highlighting six distinct categories of page components, are shown in Figure 1. All images of page documents shown is this manuscript are from the ICDAR2009 dataset [5].

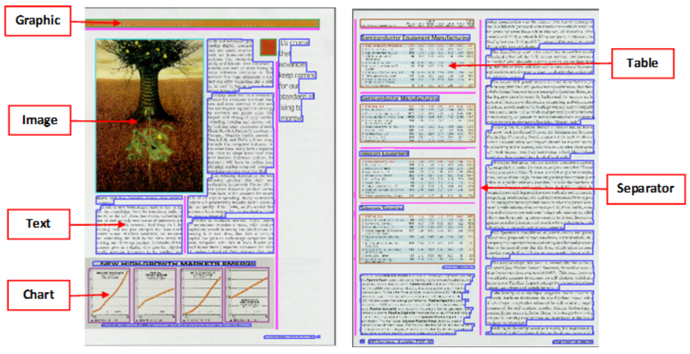


Fig. 1. Examples of page segmentation (images from the ICDAR2009 dataset [5]). Polygons overlaid on the original input image delimit its page components. Class labels are highlighted in red.

A major challenge in page segmentation is layout complexity. Complexity arises not only due to layout variability, but also to the variability of the structural components (e.g., text in different fonts, sizes and styles, multiple page component categories, orientation of the components) as well as imaging conditions (e.g., noise, skew, perspective distortion, uneven illumination). This diversity has affected the diversity of existing methods for page segmentation and layout analysis [1], [6] and most of them are tailored for very specific cases. For instance, works that mention page segmentation are usually restricted to the segmentation of specific types of components [7]–[9].

In this work, we propose a new flexible method for page segmentation. From the input-output point of view, our method receives a document page image and outputs a list of polygons that delimit the regions corresponding to each of the recognized page components (see Figure 1). The main steps of the proposed method are:

1) Computation of building blocks (in our case, connected components) and their classification as part of one of the classes (category of page components);
2) Grouping of the building blocks (taking into consideration their spatial relations and class labels) and assignment of a unique class label for each group; polygons are then computed as the enclosing contour for each of the groups.

The main idea of the method is to first attribute the most likely class labels for each building block in the document image. In this work we consider connected components as the building blocks. However, it is noteworthy to mention that other components such as pixels, superpixels or flat zones can be also considered. Then, in the second step the main idea is to explore structural information aiming to (i) solve the classification ambiguities and errors, and (ii) group the building blocks into larger regions so that the resulting final regions correspond to the expected page components.

Although similar two-step methods are not new in pattern recognition problems, we are not aware of any method similar to our formulation for the page segmentation problem. We highlight two features of the proposed method that make it suitable to tackle issues related to layout complexity and diversity. First, we model spatial relationships and similarities among the building blocks by means of a region adjacency graph. Graphs are known to be suitable to represent structural information in general, including structures of document images [10]. Grouping building blocks that correspond to a page component can then be viewed as a graph segmentation problem, where each connected subgraph would define a page component. The second aspect to be highlighted is the fact that both of the steps in our method are modeled as machine learning problems. In the first step, classification of the building block is done using a machine learned classifier and, in the second step, subgraphs corresponding to page components are obtained by removing undesired edges while keeping the desired ones. This is also done by means of a machine learned classifier. Machine learned classifiers add flexibility regarding the adaptation of the method to diverse types of layouts and page components. Details of the proposed method are presented in Section II.

In spite of some efforts in the research community to provide venues for comparative analysis of different methods [11], existing page segmentation algorithms are still mostly developed for specific cases and validated on particular datasets. The same holds with respect to available ground-truth annotations (for instance, they are either at pixel level, or at region level with bounding boxes or at region level with enclosing polygons, and usually limited to a few page component categories) and evaluation metrics [12].

One of the few available datasets with structural annotations is the PRImA Layout Analysis Dataset (https://www.primaresearch.org/datasets/Layout_Analysis). Part of it has been used in the ICDAR Page Segmentation and Layout Analysis competitions in the last years [11]. There is no separation of the dataset into training and test sets and different subsets have been used as test images in distinct editions of the competition. Besides that, although there are annotations for nine page component types, we did not find works that address the segmentation of all of them simultaneously. Thus we have opted to evaluate our method on the ICDAR2009 dataset [13], since it is the one we have used in our previous work [14]. We use the same evaluation metrics described in [11]. Our results, although not directly comparable as noted above, are competitive with the ones achieved in the last competition [11]. The experimental setup including dataset description, training and evaluation procedures, as well as the results of the experiments are described in Section III.

Conclusions are presented in Section IV.

II. Proposed approach

In this section we detail the method proposed for page segmentation. As briefly described in the introduction, it includes two main steps: (i) classification of building blocks of the images, which we assume in this work to be the connected components, and (ii) page component recognition based on a graph representation of the connected components. The method is outlined in the pipeline presented in Figure 2.

The input of the pipeline is a document page image to be segmented and the output is its segmentation result. The path in the top part of the diagram corresponds to the first step where connected components are individually classified. The other path, in the bottom part of the diagram, corresponds to the second step. Information regarding the connected components and classification labels generated in the first step are used to build a graph representation of the image. Page components are then computed on the graph and a list of the recognized components is generated. Details of the two steps are presented in the next subsections.

A. Connected Components Classification

In this step of the process, prior to connected component computation, images must be binarized. We used the well known Otsu's method. Connected components are labeled with unique identification numbers to allow easy retrieval later. Then each connected component is individually classified, with class labels corresponding to the categories of page components of interest. For instance, in the ICDAR2009 dataset there are nine page components, namely *text*, *chart*, *graphic*, *image*, *maths*, *noise*, *separator*, *table* and *other*. Six of them occur in the images shown before in Figure 1.

Works that explore connected components in document image analysis usually perform connected component classification based on its features (related to, for instance, shape, size, color, or texture) or based on small image patches containing the component [8], [14]–[16]. For classification of the connected components we follow the same idea proposed in our previous work [14].

The main idea is to crop, for each connected component, a square image patch containing the component at the center together with its surroundings. Since connected components may present a large range of sizes, a size normalization is

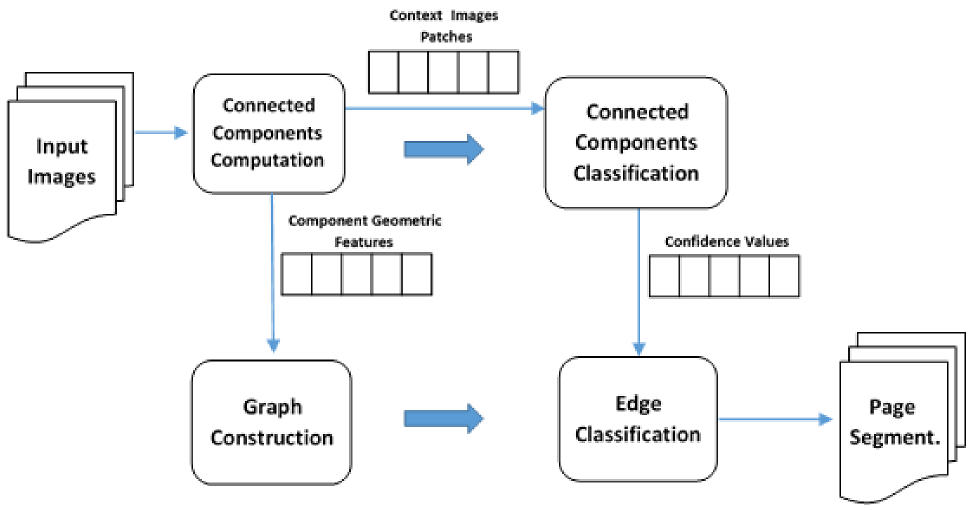


Fig. 2. Outline of the method proposed for page segmentation.

also employed. For each connected component a scaling factor to fit the component inside a 8×8 square, without changing its aspect ratio, is first determined. Then the input image is cropped around the connected component in such a way that the rescaled size of the cropped patch reduces to 40×40 and the connected component stays at its center inside a 8×8 square. Note that the scaling is applied only if the connected component does not fit inside a 8×8 square. The area surrounding the connected component carries some context information and that is why these patches are called context image patches. Figure 3 shows examples of context image patches.

Context image patches are then classified using a convolutional neural network (CNN). The main differences from our previous work [14] are: (i) in this work we consider the nine page component categories listed above, whereas in our previous work we have considered only two classes (*text* and *non-text*), and (ii) we use RGB images as the input to the CNN classifier whereas in our previous work we have considered binary images. Additional information regarding the training of the CNN classifier is presented in Section III-B. Since the CNN is trained with softmax function on its output layer, given an input context image patch, it outputs a likelihood score for each of the nine class labels.

The final result of the first step of the method is the assignment of the nine scores to each of the connected components in the image.

B. Page Segmentation based on a Graph Representation

After connected components are individually classified, those that are part of a same page component must be grouped. This problem could be addressed, for instance as a clustering problem, where not only the features but also the spatial coordinates of the connected components would play an important role in the clustering process. However, clustering is essentially based on similarities computed between pairs of elements and there is no simple way to enforce structural properties to the resulting clusters. Graphs, on the other hand, are generally accepted as adequate models for encoding structural information. Thus, we adopt a graph representation of document pages to perform the grouping.

In our representation, connected components are the nodes of a graph G. Edges can be added between any pair of two nodes. If we were able to add only edges linking pairs of nodes that belong to a same page component, then page segmentation would be reduced to the problem of computing the connected subgraphs of G. Conversely, we could also start from a complete graph and remove all edges that link nodes that are in distinct page components. In both cases, adding or removing edges can be modeled as a classification problem: given an edge, in the first case, one must decide whether the edge should be added or not, and in the second case, one should decide whether it should be removed or not.

A region adjacency graph (RAG) where the connected components are the nodes can be built by first computing the area Voronoi diagram [2] and then linking the nodes that share a common edge in the area Voronoi diagram. This results in a connected planar graph. Assuming that page components do not overlap each other, RAGs seem to be an adequate choice. Figure 4 shows part of a RAG corresponding to a page image.

Given the RAG representation of a page as described above, we model the problem of page segmentation as a problem of removing edges that link nodes belonging to distinct page components. After removal, it is expected that any two nodes in a connected subgraph belong to a same page component. The task of removing edges would be relatively easy if all connected components were classified correctly in the previous step (Section II-A). However, in a general case there is no such guarantee and there might be nearby components with conflicting labels. Therefore, to remove edges, we rely on machine learning and train an edge classifier. To that end, we assign geometric information of the connected components

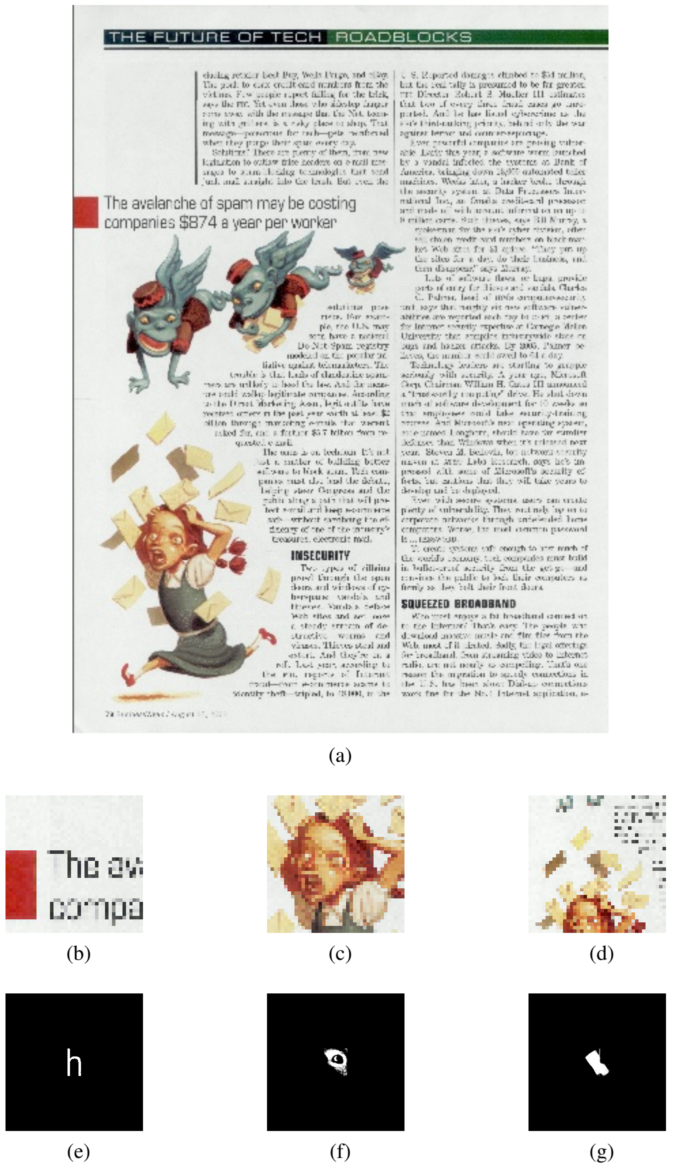

THE FUTURE OF TECH ROADBLOCKS
The avalanche of spam may be costing companies $874 a year per worker
INSECURITY
SQUEEZED BROADBAND
(a)
The av compa
h
(b)
(c)
(d)
(e)
(f)
(g)

Fig. 3. Examples of context images (3b, 3c and 3d) with the connected component rescaled within a 8×8 square at the center of a 40×40 image (patches extracted from the image in Fig. 3a). The binary images 3e, 3f and 3g correspond to the connected components of the patches 3b, 3c and 3d, respectively.

to its corresponding nodes in the graph. We consider the following geometric features[1] related to size and shape of the connected components.

- *Height:* Connected component bounding box height,
- *Width:* Connected component bounding box width,
- *Aspect Ratio:* Width divided by height,
- *Elongation:* Ratio between the height and the width, computed as $min(\text{height}, \text{width})/max(\text{height}, \text{width})$,
- *Solidity:* Area of the connected component (in pixels) divided by the area of its convex hull,
- *Area:* Number of pixels in the connected component.

[1]Note that if images are of different resolution or size, then size metrics should be normalized across images.

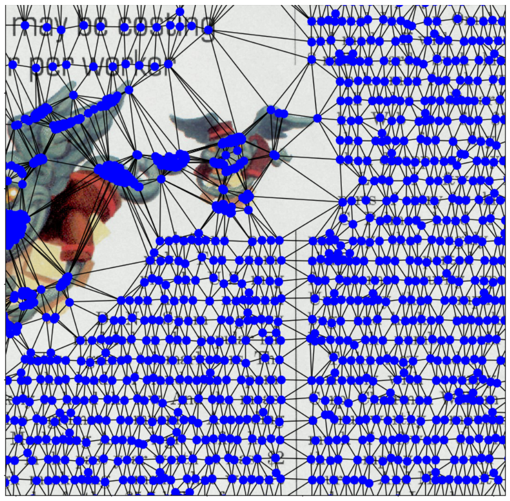


Fig. 4. Part of the region adjacency graph, restricted to a cropped region of the image shown in Fig. 3a.

In addition to these geometrical features, each node also carries the nine likelihood scores relative to the nine page component categories which were attributed by the CNN classifier in the previous step of the pipeline. We have then a binary classification problem, where each edge (u, v) must be classified as positive or negative, based on the set of features consisting of:

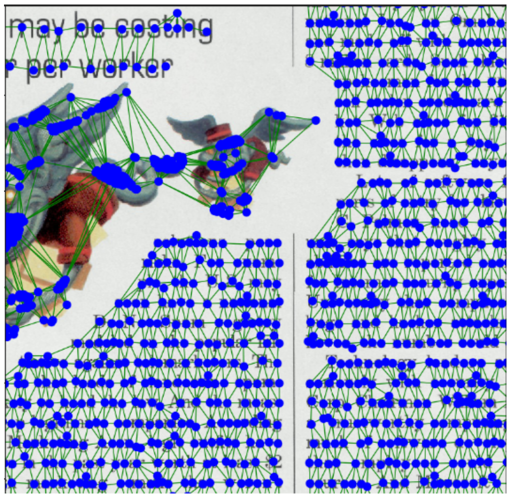
- the nine likelihood scores of node u,
- the nine likelihood scores of node v,
- the geometrical features of u,
- the geometrical features of v,
- the Euclidean distance between the centroids of u and v,
- absolute value of the angle between the X-axis and edge (u, v),

Positive edges are those that we would like to keep whereas negative are those that we would like to remove from the graph.

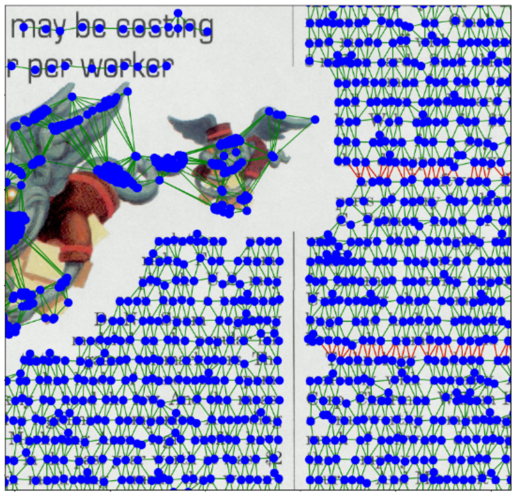
An important characteristic of the RAGs created as described above is the fact that we generally obtain a high quantity of edges linking connected components belonging to a same page component (e.g., characters within a paragraph) – positive edges, and a small quantity of edges linking connected components belonging to distinct page components – negative edges. Training classifiers with these data would tend to favor the correct classification of positive edges. However, in the context of our problem the false positives are more important than false negatives. For instance, in Figure 4, we can see many edges linking pairs of characters that are part of a same text component in the page. Even if most of those edges were classified as negative, the accuracy of the segmentation would not be affected if enough edges were classified as positive. In contrast, a false positive would group nodes that belong to distinct page components, resulting in a single large mixed group. In the example of Figure 4, any edge linking a letter and part of the image should be classified as negative because otherwise it would join a *text* and an *image* page components.

This problem of highly unbalanced classes can be partially mitigated by training classifiers with a weighted cross entropy cost function. The weight, denoted w, penalizes more one type of the error over the other. Additional details on training edge classifiers are given in Section III-B.

Figure 5 shows examples of false positives. In the first image the set of edges is correct, whereas in the second image edges shown in red correspond to false positives. Note, however, that in this case although the false positives end up joining two paragraphs, this is a less critical error since both page components are of the same type.



(a)



(b)

Fig. 5. Edge classification: (a) Ground-truth of the image in Fig. 3a, (b) example of classification result (false positives in red).

After edge classification, a set of connected subgraphs will emerge. Each of these subgraphs will be regarded as a detected page component. As there might be conflicting class labels assigned to the nodes of a subgraph, a majority vote based on the highest score label of each connected component is used to decide the final page component label. The polygon that delimits each detected page component is computed as the polygon enclosing all of its connected components. In this work, we calculate the convex hull to serve as an approximation of the enclosing polygon.

III. Experimental Results

The proposed method has been implemented using the `Python` programming language, together with `Tensorflow` library for the classifiers, `NetworkX` library for building the RAG and `Scikit Image` library for image processing.

A. Dataset

We evaluated our approach on the ICDAR2009 competition dataset [13]. Recall that reasons for this choice are explained in the introduction. We considered the same 53 images used in [14], where 28 images (almost half of the data) are for training and 25 are for testing. Unlike in [14], where only two classes (*text* and *non-text*) were considered, here we considered all 9 classes of page components: *text*, *chart*, *graphic*, *image*, *maths*, *noise*, *separator*, *table* and *background*.

B. Training and selection of classifiers

1) Connected component classification: A total of $137,862$ connected components from the training images were extracted, and they were randomly divided into validation (20%) and training sets (80%). To train a CNN, we used the same architecture and hyperparameters of the CNN that achieved the best validation results in our previous work [14]. It consists of two sequences of convolution–ReLU–max-pooling layers, followed by two fully connected layers, plus an output layer with softmax function. We kept the dropout regularization in the penultimate layer and used the Adam algorithm [17] for optimization, as done before. We also evaluated larger patch sizes, but since no significative gain was observed on the validation set, we decided to keep the 40×40 patch size.

The results of this CNN and the results achieved in our previous work are shown in Table I. It should be noted, however, that while here we used RGB input patches and considered all the nine classes, in our previous work we used binary input patches and considered only two classes (*text* and *non-text*). Nevertheless, the overall accuracy has improved as well as the precision and recall with respect to the *text* class. These results indicate that color information is relevant for the classification of the connected components.

TABLE I
Connected component classification results on the validation set

CNN classifier	Accuracy (over all classes)	Precision (*text* class)	Recall (*text* class)
Previous [14]	98.68 (2 classes)	99.30	98.88
Current work	99.31 (9 classes)	99.59	99.89

2) Edge classification: To train the edge classifier, RAGs were built for each of the training images as described in Section II-B, and then edges from all of the training images were pooled together and randomly split into two subsets, 20% for validation and 80% for training, keeping the percentage of positive and negative examples in both subsets. We used standard neural networks (Multilayer Perceptron Networks) for edge classification.

Since there are more positive examples than negative ones and since false positives are critical (see Section II), we adjusted the edge classifier to favor true negatives (correct classification of edges linking components in distinct regions) by playing with the positive class weight w in the cost function.

More specifically, we varied the weight parameter w in the range $[0.01 - 0.9]$ and found out that best results with respect to the validation set were in the range $[0.01 - 0.09]$. This means that errors in classifying positive edges have small importance while errors in classifying negative edges are heavily penalized. Recall and accuracy for weights in this range are shown in Figure 6. We denote negative edges (those that link connected components belonging to different page components) as class 0 and positive edges (those that link connected components within a same page component) as class 1. Note that the overall accuracy is close to the recall of the positive class since most of the edges are positive ones.

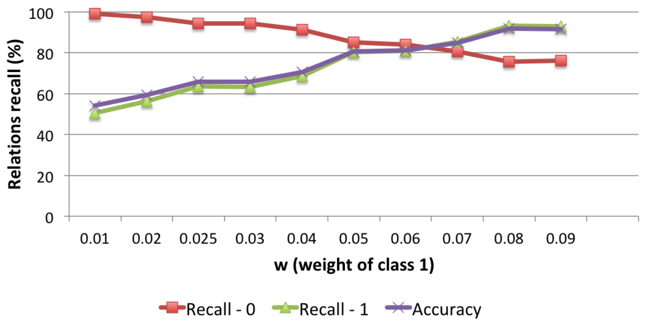


Fig. 6. Class balancing parameter $w \in [0.01 - 0.09]$: recall of negative edges (Recall-0), recall of positive edges (Recall-1), and accuracy.

After this preliminary evaluation, the ten classifiers corresponding to weight range $[0.01 - 0.09]$ were further evaluated to choose one final classifier. The second evaluation aimed at choosing the classifier that generates the best page segmentation results. To quantify page segmentation we used a score based on the *segmentation + classification* metric defined in [11]. The metric is used to define a *success rate* score that represents the percentage of area overlap between regions of the detected page components and the ones defined by the ground-truth. This score also takes into consideration the class label of the regions.

Figure 7 shows the mean success rate, with respect to a subset of four training images, of each of the ten classifiers. Based on this evaluation, we find that the class balancing weight parameter $w = 0.07$ yields better results and thus the corresponding classifier was chosen as the final edge classifier.

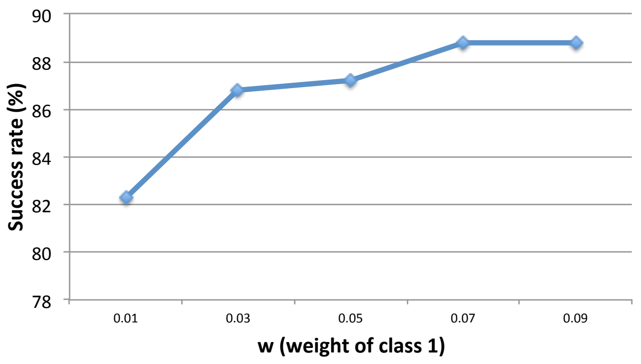


Fig. 7. Success rate at segmentation + classification level for weights w in the range $[0.01 - 0.09]$ over a subset of four training images.

C. Results

Results presented in this section were computed on the test set employing the proposed method, with connected component classifier and edge classifiers trained and chosen as detailed above.

Figure 8 shows the *segmentation* and *segmentation + classification* success rates per image on the test set. The *segmentation* success rate is similar to the one computed based on *segmentation + classification* except that it does not consider class labels. Success rates, for both metrics, on individual images are all above 80%. Furthermore, scores based on both metrics are very close each other for every image, indicating that only a small percentage of the correctly detected regions are classified wrongly.

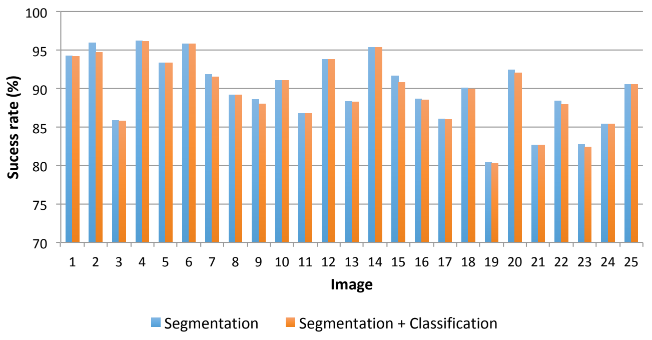


Fig. 8. Success rate with respect to *segmentation* and *segmentation + classification* metrics for each of the 25 test images.

As we mentioned in the introduction, our results are not directly comparable to the ones reported in the last competition [11] because, to the best of our knowledge, none of the methods in the competition and others described in the related literature attempted classification of all nine page component types and also because the test set we have used is not the same used in the competition. Nevertheless, we point that

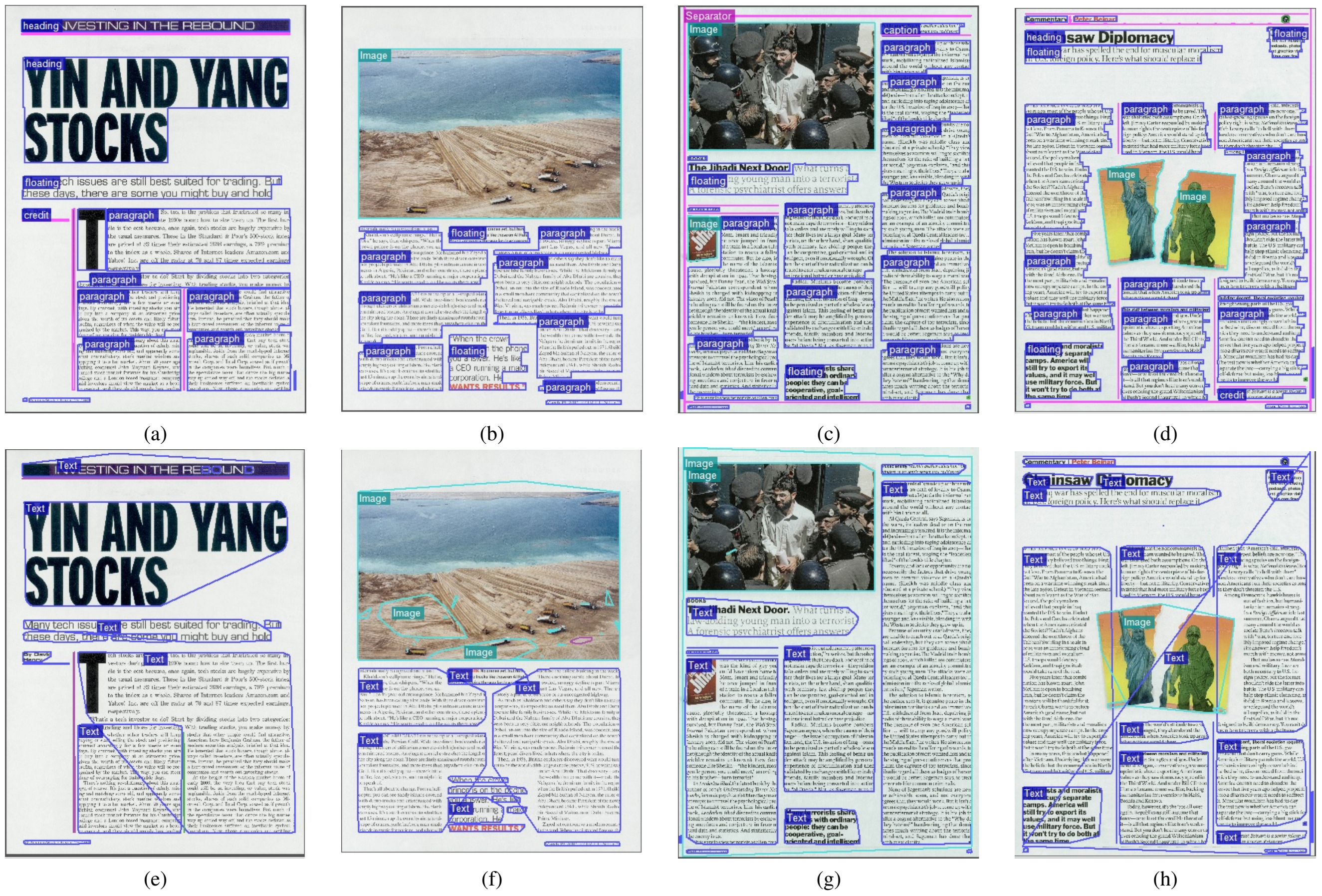


Fig. 9. Examples of results: ground-truth at the top row (9a - 9d) and respective results below (9e - 9h). Images 9g and 9h correspond to the two worst results. See details in the text.

some of our test images are included in the last competition's test set and, therefore, comparison provides some information on how our method is performing. The success rates based on the *segmentation* and *segmentation* + *classification* metrics reported in [11] are in the ranges $[75-92]\%$ and $[72-90]\%$, respectively. The results we obtained for individual test images are all in these ranges. The average *segmentation* and *segmentation* + *classification* success rates of our method on the 25 test images was 89.83% and 89.64%, respectively. This is a very interesting result, considering that we have considered all nine page component classes and we have not implemented some immediate improvements ss the ones described below.

By visual inspection, we note that some errors are due to the use of the convex hull to compute the enclosing polygon, which ends up including all the undesired area corresponding to the concave parts. Error of these type can be seen for instance in Figures 9e and 9f, where some page components of type *text* (for instance, the one that has a large bold "T" at the beginning) is described by a convex polygon that overlaps another page component. We also note that a simple post-processing could fix some of the errors. For instance, in Figure 9f, multiple components of type *image* are detected inside another large *image* component. These interior components could be removed by applying non-maximum suppression-like techniques.

The two worst results, shown in Figures 9g and 9h, were due to the misclassification of separators as *image* or as *text* components. In Fig. 9g, note the large green rectangle covering almost the entire image, and in Fig. 9h, there is a large triangular component at the right side detected as a *text*. These errors could be solved, for instance, by adding additional shape information to the set of component features. Concave polygons would also reduce the spurious overlapping areas.

IV. Conclusions and Future work

We proposed an effective, simple, and flexible machine learning based method for page segmentation. All 25 tested images resulted in success rates above 80% with regard to the *segmentation* and *segmentation* + *classification* scores. This is an important result since all nine page component types available in the ICDAR2009 dataset [13] were considered.

The proposed method is flexible with respect to several aspects. First, although we have used connected components as the building blocks in the first step of the pipeline, alternative components such as superpixels or regions of a oversegmentation can be used as well. These latter ones may be interesting when distinct page components touch each other. Secondly, both building block classifier in the first step as well as the edge classifier in the second step are trained using machine learning techniques. This allows an easy adaptation of the method to different types of documents and page components.

Several points of the method are amenable to improvements. We used the Otsu binarization to compute the connected components but other algorithms can be employed as well. For edge classification, we did not evaluate how distinct features affect the results. There is room for exploring additional features and classification algorithms, including deep learning approaches. Replacing the convex hull algorithm with any other that computes a more tight polygon will definitely improve the results.

To better evaluate the potential of the method, in addition to the improvements above, we plan to evaluate our method on other datasets. A challenging future work is to include structural constraints in the second step of the method, so that not only local but more global information is also considered for edge removal.

Acknowledgment

A. L. L. M. Maia thanks State University of Feira de Santana, F. D. Julca-Aguilar thanks FAPESP (grant 2016/06020-1), and N. S. T. Hirata thanks CNPq (305055/2015-1). This work was supported by FAPESP (grant 2015/17741-9) and CNPq (grant 484572/2013-0).

References

[1] K. Kise, *Page Segmentation Techniques in Document Analysis*. London: Springer London, 2014, pp. 135–175.

[2] K. Kise, A. Sato, and M. Iwata, "Segmentation of page images using the area Voronoi diagram," *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 370 – 382, 1998.

[3] J. Liang, I. T. Phillips, and R. M. Haralick, "Performance evaluation of document layout analysis algorithms on the UW data set," in *Document Recognition IV*, vol. 3027. International Society for Optics and Photonics, 1997, pp. 149–161.

[4] A. O. Shigarov and R. K. Fedorov, "Simple algorithm page layout analysis," *Pattern Recognition and Image Analysis*, vol. 21, no. 2, pp. 324–327, Jun 2011.

[5] A. Antonacopoulos, D. Bridson, C. Papadopoulos, and S. Pletschacher, "A realistic dataset for performance evaluation of document layout analysis," in *10th International Conference on Document Analysis and Recognition*. IEEE, 2009, pp. 296–300.

[6] A. Dengel and F. Shafait, *Analysis of the Logical Layout of Documents*. London: Springer London, 2014, pp. 177–222.

[7] Z. Shi and V. Govindaraju, "Multi-scale techniques for document page segmentation," in *Eighth International Conference on Document Analysis and Recognition (ICDAR)*, vol. 2, 2005, pp. 1020–1024.

[8] S. S. Bukhari, M. I. A. Al Azawi, F. Shafait, and T. M. Breuel, "Document image segmentation using discriminative learning over connected components," in *9th IAPR International Workshop on Document Analysis Systems (DAS)*. ACM, 2010, pp. 183–190.

[9] T. Kasar, P. Barlas, S. Adam, C. Chatelain, and T. Paquet, "Learning to detect tables in scanned document images using line information," in *12th International Conference on Document Analysis and Recognition*, 2013, pp. 1185–1189.

[10] H. Bunke and K. Riesen, "Recent advances in graph-based pattern recognition with applications in document analysis," *Pattern Recognition*, vol. 44, no. 5, pp. 1057 – 1067, 2011.

[11] C. Clausner, A. Antonacopoulos, and S. Pletschacher, "ICDAR2017 Competition on Recognition of Documents with Complex Layouts - RDCL2017," in *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, Nov 2017, pp. 1404–1410.

[12] M. Cote and A. B. Albu, "Layered ground truth: Conveying structural and statistical information for document image analysis and evaluation," in *23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 3258–3263.

[13] A. Antonacopoulos, S. Pletschacher, D. Bridson, and C. Papadopoulos, "ICDAR 2009 Page Segmentation Competition," in *10th International Conference on Document Analysis and Recognition*. IEEE Computer Society, 2009, pp. 1370–1374.

[14] F. D. Julca-Aguilar, A. L. L. M. Maia, and N. S. T. Hirata, "Text/non-text classification of connected components in document images," in *30th Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2017, pp. 450–455.

[15] V. P. Le, N. Nayef, M. Visani, J. M. Ogier, and C. D. Tran, "Text and non-text segmentation based on connected component features," in *International Conference on Document Analysis and Recognition (ICDAR)*, Aug 2015, pp. 1096–1100.

[16] Y. Aramaki, Y. Matsui, T. Yamasaki, and K. Aizawa, "Text detection in manga by combining connected-component-based and region-based classifications," in *IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 2901–2905.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." *ICLR*, 2014.