Journal of Universal Computer Science, vol. 18, no. 5 (2012), 623-649
submitted: 29/12/10, accepted: 21/2/12, appeared: 1/3/12 © J.UCS

Improving the Extraction of Text in PDFs by Simulating the Human Reading Order

Ismael Hasan
(Information Retrieval Lab, Computer Science Department
University of a Coruña, Spain
ihasan@udc.es)

Javier Parapar
(Information Retrieval Lab, Computer Science Department
University of a Coruña, Spain
javierparapar@udc.es)

Álvaro Barreiro
(Information Retrieval Lab, Computer Science Department
University of a Coruña, Spain
barreiro@udc.es)

Abstract: Text preprocessing and segmentation are critical tasks in search and text mining applications. Due to the huge amount of documents that are exclusively presented in PDF format, most of the Data Mining (DM) and Information Retrieval (IR) systems must extract content from the PDF files. In some occasions this is a difficult task: the result of the extraction process from a PDF file is plain text, and it should be returned in the same order as a human would read the original PDF file. However, current tools for PDF text extraction fail in this objective when working with complex documents with multiple columns. For instance, this is the case of official government bulletins with legal information. In this task, it is mandatory to get correct and ordered text as a result of the application of the PDF extractor. It is very usual that a legal article in a document refers to a previous article and they should be offered in the right sequential order. To overcome these difficulties we have designed a new method for extraction of text in PDFs that simulates the human reading order. We evaluated our method and compared it against other PDF extraction tools and algorithms. Evaluation of our approach shows that it significantly outperforms the results of the existing tools and algorithms.

Key Words: PDF, text preprocessing, text extraction, ordered text extraction, text mining, evaluation
Category: I.7.4, I.7.5, J.1

1 Introduction

Nowadays an important portion of the information published in the web is available in PDF format [McKinley 1997]: presentations in slides, official bulletins, articles, news, technical reports, etc. Much of this information is presented with complex layouts. For instance, several official government organisations publish a lot of documents using this format; we can cite, as examples, the *Bulletin of the European Union* or the *Federal Register*. The access to this kind of information is a general request by the citizens, companies and institutions. Because of that, lately, great efforts have been taken in building DM and IR systems that deal with this kind of sources. But, usually these documents are very long and contain a lot of information, and the users usually do not want a huge document with several tens of pages as a result of a query. In order to provide search over these documents the text contained in them must be segmented. When approaching the task of document segmentation one crucial point is text extraction from PDF files. However, the existing PDF toolkits tend to make mistakes in the extraction of ordered text from multiple columns. We think that a heuristic approximation may be suitable to solve this issue because the PDF documents are generated using computer applications, so the degree of variability can be captured by heuristics.

The problem with the PDF text extractors is that sometimes the text is not correctly processed. For instance, it can happen that the text is not extracted in the order a human would expect, but in the order in which the PDF files are rendered. This issue may cause a bad impression to the users accessing the results. The reaction of a user when the result of a system is not the expected (unordered paragraphs, for instance) usually is to think that the system is immature or inaccurate.

Moreover, the cited kind of documents are almost always divided in sections. The pre-processing of their information is advisable, so the users can search inside the sections instead of inside entire documents. The pre-processing of the information becomes much more important in the case of government publications (regulations, etc): these documents are usually very long, and they can be segmented in logical parts: title, articles, etc; each one of these parts should be individually searchable in an IR system. It is crucial for this segmentation task that the textual content is recovered in the same order as it is displayed in the PDF document.

The main implication of the previous remarks is that, in order to build a competitive IR system, the text obtained through a PDF text extraction tool should be ordered in the same way as in the original document. Therefore the correct extraction of text from PDFs is an important problem and a critical feature for IR systems which have to deal with passages, sentences, generation of summaries and snippets, etc.

With this motivation, we have developed an algorithm which extracts the text in the “right order” from PDF documents with multi-column layout in which the text blocks are separable by horizontal and vertical cuts, commonly named *Manhattan layout*. The algorithm was successfully applied to a collection of official legal bulletins, but it also can be used with other kinds of documents such as research papers, patent documents, and others.

The algorithm uses high-level operations over the PDF files to simulate the way in which a document would be read by a person. As an added benefit, the algorithm also returns the column structure of the documents, so the information can be offered to an user in a way similar to the original format if desired.

The rest of the paper is organised as follows. In section 2, some PDF text extraction tools and previous works are introduced. Section 3 describes the particularities of the domain documents. Section 4 describes the algorithm and show a detailed description of each step (including pseudocode). In section 5 the algorithm is evaluated in terms of effectiveness and complexity, defining a metric, the test datasets, and comparing the results with the results of other tools. Finally, in the last section, conclusions and future work are presented.

2 Previous Works

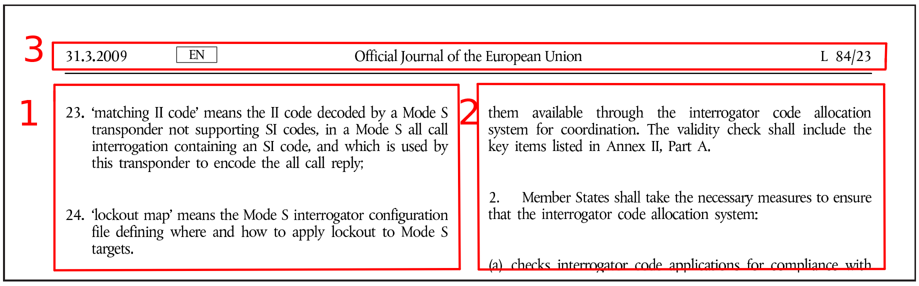
2.1 Existing Tools

There are a lot of free and commercial tools offering text extraction from PDF, so reviewing all of them is unfeasible. As an example, a list of utilities can be found at [PlanetPdf 2010].

As free tools, we can name [PDFBox 2010], a project recently taken by the *Apache Software Foundation*, [Multivalent 2010], a Java tool which offers text extraction from command-line, [Pstotext 2010], a freeware utility which offers text extraction from PDF and Postscript files, and [PDF-Analyzer 2010], a desktop freeware application for Windows. As purely commercial tools we can name [PDF-File-Converter 2010], offering also format extraction (font type, size, etc), and [TextFromPdf 2010], offering layout extraction.

Sadly, these tools make mistakes. When applying any of them, it can happen that the text is extracted in wrong order. An actual example of bad extraction using *Multivalent* is shown in Fig. 1, where the bordered areas are extracted in the order indicated by the numbers. An explanation of this behaviour can be found in the documentation of the *pstotext* tool, where it is explicitly stated that the ordered extraction may not be the expected: "*pstotext outputs words in the same sequence as they are rendered by the document. This usually, but not always, follows the order that a human would read the words on a page*". We have found that most of the aforementioned tools (*PDFBox*, *Multivalent*, *PDF-Analyzer* and *PDF-File Converter*) use a similar approach, since the results offered by these utilities are very similar to the one obtained using *pstotext*: for instance, when using these tools to extract the text from the *Federal Register* bulletins, the footnotes always appear at the beginning of the page.

Regarding *TextFromPdf*, it uses a different technique; studying the output of this tool, it seems that it generates paragraphs, and then it sorts them first


31.3.2009 EN Official Journal of the European Union L 84/23

23. 'matching II code' means the II code decoded by a Mode S transponder not supporting SI codes, in a Mode S all call interrogation containing an SI code, and which is used by this transponder to encode the all call reply;

24. 'lockout map' means the Mode S interrogator configuration file defining where and how to apply lockout to Mode S targets.

them available through the interrogator code allocation system for coordination. The validity check shall include the key items listed in Annex II, Part A.

2. Member States shall take the necessary measures to ensure that the interrogator code allocation system:

(a) checks interrogator code applications for compliance with

Figure 1: *Order of extraction using Multivalent. Taken from the Official Journal of the European Union; March, 31th 2009; page 4.*

by y-coordinate (the upper elements are returned in the first place), and then by x-coordinate, from left to right. The main problem of this approach is that a paragraph does not correspond with a column, so the paragraphs belonging to different columns are mixed in the output according to their y-coordinates.

2.2 Previous Literature

This section is divided in two parts; the layout detection in OCR algorithms, and the layout detection from PDF documents. The reason to include the reference to the OCR techniques is because they provide a clear explanation of the different approximations to layout detection; and, despite the fact that image files and PDF documents are processed in completely different ways, some of the ideas behind the OCR techniques can be applied to PDF text extraction.

In the OCR field there are several works which present techniques to obtain the layout of images containing text. Basically, there are three main approaches to the identification of the layout: *bottom-up* techniques [Mitchell 2000, Simon et al. 1997, O'Gorman 1993], which first identify minimum items (black pixels), and then recursively construct larger items (words, phrases, etc); *top-down* [Baird et al. 1990, Chakraborty et al. 2003, Krishnamoorthy et al. 1993], which first identify global elements of a page (black and white stripes, etc), and then split them into smaller items; and hybrid techniques, which combine both approaches. Particularly, *top-down* approaches are suitable to analyse documents with a *Manhattan layout* (documents in which the pages are separable into blocks by vertical and horizontal cuts) in linear time; *bottom-up* approaches can be applied to any layout, but they usually are quadratic in time and space [Simon et al. 1997]. Because of this, the use of generic bottom-up techniques would be unwarranted when we are dealing with *Manhattan layout* documents. It is important to mention that most of the cited articles are focused on the detection of layout, being [Mitchell 2000] the only one which approaches the problem of the order in which the layout items should be extracted.

In the field of PDF text extraction we must emphasise that works on *bottom-up* techniques are used, [Rosenfeld et al. 2002] and [Lovegrove et al. 1995]; regarding to *top-down* techniques, it is very remarkable the work in [Meunier 2005]. Finally, a hybrid technique to detect the layout of PDF files can be found in [Bloechle et al. 2009]. Some brief explanations about them follow:

In [Rosenfeld et al. 2002] the authors propose a text extraction system which preserves documents structure using a *bottom-up* strategy. The characters of a page are grouped to build bigger entities, until all of the text has been retrieved. This grouping is done by using neighbourhood metrics. But, the decisions about the results may present some problems when working in complex scenarios:

- In order to build lines, the characters are grouped. One of the premises used to decide if two characters are neighbours is the similarity between y-coordinates. A direct consequence of this would be that a superscript character and the superscripted character are not neighbours (which can mean that they are extracted in different lines).
- In order to build paragraphs, it is required that lines belonging to the same paragraph share the same font. There are documents which do not follow this rule, indeed, there are documents in which the same line can contain several font types.
- In order to build columns font similarity is assumed. Also, a threshold is defined to decide if two paragraphs belong to the same column, according to the vertical separation between them.

The documents we want to deal with, publications from official organisations, usually do not follow these premises; so, this method would not be suitable for our purposes.

In [Lovegrove et al. 1995] the authors propose a *bottom-up* strategy based upon the type of fonts. The steps of the algorithm regarding text extraction proceed as follows:

1. The words (previously obtained) are grouped into lines.
2. Each line is characterised with a key consisting of the point size and the font. A bin is created for each unique pair of point size and font.
3. Operations over the bins are done to separate them into sub-bins, and the space separation between lines is searched using these sub-bins. With this information, *blocks* are found.
4. The following steps of merging of blocks, and identification of hierarchy between blocks, are done according to font characteristics.

The problems of the algorithm (addressed by the authors) are that the font comparisons may fail (for instance, a line may have, at most, italic characters, so it will not be merged with the upper and lower lines if the italic characters do not prevail in them). Also, the characterisation of the spaces between lines may be poorly recognised.

In [Meunier 2005] the author proposes a *top-down* method based on the popular XY-cuts algorithm [Nagy and Seth 1984] [Mao and Kanungo 2001] to segment the page and offer its contents in reading order. The original method works by dividing the page with horizontal or vertical cuts and by recursively applying the algorithm on the divisions. Meunier modifies the original method so the result is more similar to the common order of reading in formal documents (guided by columns). The modification consists in giving more importance to the vertical cuts instead of to the horizontal cuts: once it gets all of the available combinations of horizontal cuts, the combination which is most conducive to the creation of long vertical cuts is selected. This method seems suitable to our problem, the extraction of content from legislative documents. But, as the author mentions, to deal with non uniform layouts (for instance, with varying interlinear space) it may be necessary to set the parameters of the algorithm so each line is extracted as a block. Taking into account that the algorithm computational time depends on the number of horizontal cuts, having as many cuts as lines will greatly increase the execution time of the algorithm. The author reports that the detection of the layout of a complex page may take even days. Meunier deals with this issue by applying dynamic programming, which considerably reduces the time of execution of the algorithm. However, the algorithm is not robust on the input parameters (minimum widths for vertical and horizontal separations).

In [Bloechle et al. 2009] the authors developed a canonical format for representing structured electronic documents, *OCD*. They present an hybrid technique to detect the layout of PDF documents in order to convert them to the *OCD* format. The first step is a *bottom-up* process, based on the previous works of the authors [Bloechle et al. 2006], in which the structures of the document are detected (i.e., paragraphs, images). The novelty, regarding their previous work, is to include a second *top-down* step, to overcome the under-segmentation problem of their initial technique. The next step proposed by the authors is to use this layout to generate the logical structure of the documents using a learning system, *Dolores* [Bloechle et al. 2008]. Once the logical structure is obtained, it is pretty straightforward to obtain the text of the document in reading order. However, we think that a heuristic approximation is more suitable for the domain we are working with. These documents are built in a semi-automated way, using PDF tools for instance, so the variability is limited.

Finally, it is fair to mention that the detection of tables inside the documents is beyond of the scope of our work.

3 Data Characteristics

This work focuses on official publications in PDF with a *Manhattan layout*, like patent documents, research papers and publications from official organisations (i.e., *official bulletins*). Fig. 2 shows a page of an official bulletin. The method works with documents with single column format or multiple column format. Some issues to take into account when analysing these documents are:

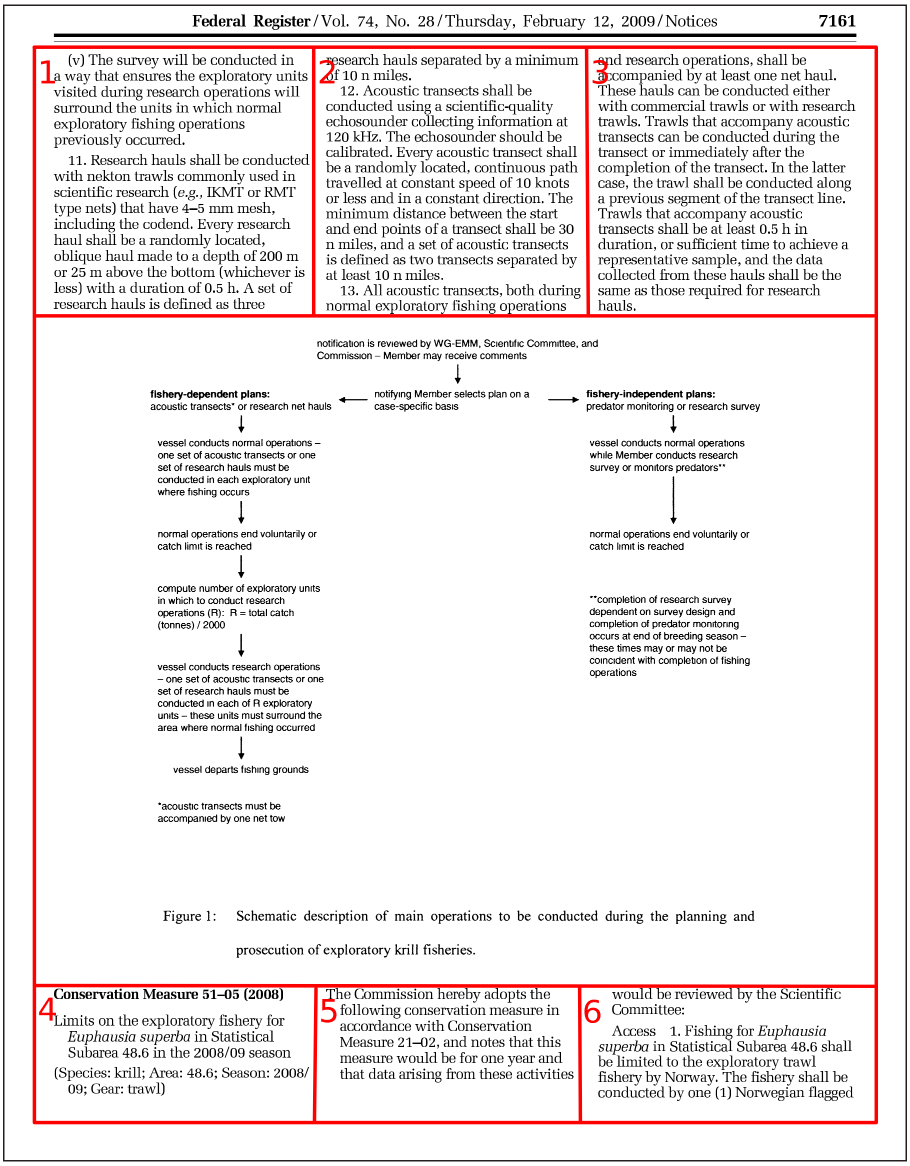
Page Headers
In the official publications, it is very usual that pages have a header including the name and number of the publication, the date, the number of page, etc.

Columns and regions
A document can be structured using columns of text; for instance, the two-column format is widely used. We will refer to these text columns just as *columns*. A *region* is a part of a page defined by its coordinates inside the page; we will refer as *text regions* to the ones containing text, as *image regions* to the ones containing images and as *blank regions* to the empty ones. The regions have the following properties:

- They are rectangular in shape.
- There must be a minimum vertical separation between regions (meaning the separation between columns). On the other hand, it is not required a minimum horizontal separation between regions.
- Two regions can not overlap each other. This also entails that nested regions can not exist.

Multiple Column
Different parts of a document's page can contain different number of columns.

Documents with Images
It is very usual that the documents contain images: graphs, photos, scanned texts, etc. These items sometimes determine the way in which a person reads the document. This issue is clearly shown in Fig. 2, where the correct order of extraction of the regions is marked with numbers.



Federal Register / Vol. 74, No. 28 / Thursday, February 12, 2009 / Notices **7161**

1 (v) The survey will be conducted in a way that ensures the exploratory units visited during research operations will surround the units in which normal exploratory fishing operations previously occurred.

11. Research hauls shall be conducted with nekton trawls commonly used in scientific research (*e.g.*, IKMT or RMT type nets) that have 4–5 mm mesh, including the codend. Every research haul shall be a randomly located, oblique haul made to a depth of 200 m or 25 m above the bottom (whichever is less) with a duration of 0.5 h. A set of research hauls is defined as three

2 research hauls separated by a minimum of 10 n miles.

12. Acoustic transects shall be conducted using a scientific-quality echosounder collecting information at 120 kHz. The echosounder should be calibrated. Every acoustic transect shall be a randomly located, continuous path travelled at constant speed of 10 knots or less and in a constant direction. The minimum distance between the start and end points of a transect shall be 30 n miles, and a set of acoustic transects is defined as two transects separated by at least 10 n miles.

13. All acoustic transects, both during normal exploratory fishing operations

3 and research operations, shall be accompanied by at least one net haul. These hauls can be conducted either with commercial trawls or with research trawls. Trawls that accompany acoustic transects can be conducted during the transect or immediately after the completion of the transect. In the latter case, the trawl shall be conducted along a previous segment of the transect line. Trawls that accompany acoustic transects shall be at least 0.5 h in duration, or sufficient time to achieve a representative sample, and the data collected from these hauls shall be the same as those required for research hauls.

notification is reviewed by WG-EMM, Scientific Committee, and Commission – Member may receive comments

fishery-dependent plans: acoustic transects* or research net hauls

notifying Member selects plan on a case-specific basis

fishery-independent plans: predator monitoring or research survey

vessel conducts normal operations – one set of acoustic transects or one set of research hauls must be conducted in each exploratory unit where fishing occurs

normal operations end voluntarily or catch limit is reached

compute number of exploratory units in which to conduct research operations (R): R = total catch (tonnes) / 2000

vessel conducts research operations – one set of acoustic transects or one set of research hauls must be conducted in each of R exploratory units – these units must surround the area where normal fishing occurred

vessel departs fishing grounds

*acoustic transects must be accompanied by one net tow

vessel conducts normal operations while Member conducts research survey or monitors predators**

normal operations end voluntarily or catch limit is reached

**completion of research survey dependent on survey design and completion of predator monitoring occurs at end of breeding season – these times may or may not be coincident with completion of fishing operations

Figure 1: Schematic description of main operations to be conducted during the planning and prosecution of exploratory krill fisheries.

4 **Conservation Measure 51–05 (2008)**

Limits on the exploratory fishery for *Euphausia superba* in Statistical Subarea 48.6 in the 2008/09 season

(Species: krill; Area: 48.6; Season: 2008/09; Gear: trawl)

5 The Commission hereby adopts the following conservation measure in accordance with Conservation Measure 21–02, and notes that this measure would be for one year and that data arising from these activities

6 would be reviewed by the Scientific Committee:

Access 1. Fishing for *Euphausia superba* in Statistical Subarea 48.6 shall be limited to the exploratory trawl fishery by Norway. The fishery shall be conducted by one (1) Norwegian flagged

Figure 2: *Page crossed by an image. Taken from the Federal Register; February, 12th 2009; page 7161.*

4 The Algorithm

4.1 General features

The idea behind our method is that the text belonging to the same region must be extracted together. The method relies on tools which can provide some high-level functionalities over PDF documents. The main functionalities used are:

– A way of obtaining general parameters of the document, for instance the width and height of a page, the number of pages, and similar features.

– Extraction of the text from a rectangle, which is defined using Cartesian coordinates inside a page.

– Retrieval of the images of a document, obtaining their coordinates.

We coined our algorithm as "*Left Regions Expansion*" (LRE) because the key of the algorithm is to detect the top of each region on the left of a page, and then extend it as much as possible. The next step is to find the regions on the right of the previously found regions, and finally the images contained in the page are used to split the regions (if necessary). The algorithm applies the next steps to each page in the document:

1. Detecting the regions which are present in the page.
2. Retrieving the list of images and creation of regions using the images coordinates.
3. Splitting the text regions which are crossed by image regions.
4. Sorting the regions of a page in the following way:
   (a) The region comprising the header of the page.
   (b) The top left region.
   (c) The regions on the right of the one obtained in (b).
   (d) The region on the left of the page which is below the previously found regions.
   (e) The regions on the right of the one obtained in (d).
   (f) Steps (d) and (e) are repeated until no more regions are found.
5. Extracting the text of each region, in the order stated in (4).

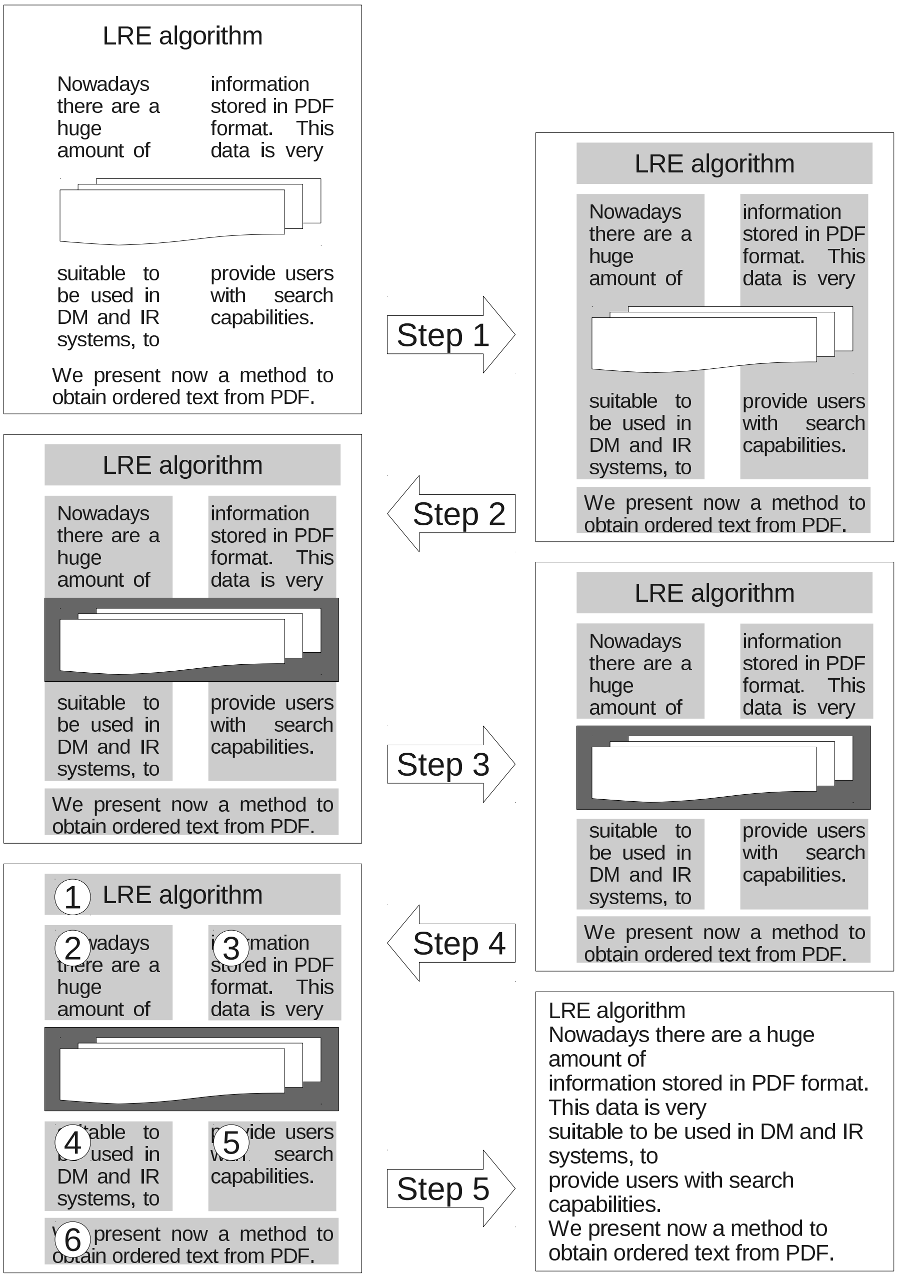


Figure 3: *LRE algorithm steps. Text regions are marked in light gray, image region is marked in dark grey and final order of extraction is circled.*

To get the text regions (step 1), the algorithm uses the separation between columns of text. Baird et al. used a similar method in [Baird et al. 1990]: they developed a technique to get the text from an image by detecting the blank regions. A similar method can be found in [Krishnamoorthy et al. 1993].

We will show an example of the algorithm over Fig. 2. The central part of the page (the diagram) is not text, but a scanned image. The algorithm works as follows:

1. It detects the header, and three more regions: one comprising the blocks 1 and 4, one comprising the blocks 2 and 5, and one comprising the blocks 3 and 6.

2. It retrieves the image to create a new region.

3. It splits the previously found regions, due to the fact that they are crossed by an image. The layout shown in the image is obtained in this step.

4. It sorts the regions in the following order:

   (a) The region comprising the header of the page.

   (b) The top left region (1).

   (c) The regions on the right of the one obtained in (b) (2 and 3).

   (d) The region on the left of the page which is below the previously found regions (4).

   (e) The regions on the right of the one obtained in (d) (5 and 6).

5. It extracts of the text of each region, in the order previously stated.

Next, we will go in depth with the algorithm steps. The functions explained in sections 4.2, 4.3, 4.4 and 4.5 implement the first step of the algorithm (Fig. 3); they are sequentially applied to each page to obtain the text regions. The second and third steps, the ones referring to images, are covered in 4.6. The last steps of the algorithm, the extraction of the text of each region, is explained in 4.7.

It must be noted that the coordinates used in the algorithm are Cartesian coordinates, which means that the bottom left of a page will be (0,0). In the pseudocode, the coordinates of a region will be referred as $region_{x.left}$, $region_{x.right}$, $region_{y.top}$ and $region_{y.bottom}$. We will also refer to the limit values these coordinates can have as x_{min} (left), x_{max} (right), y_{min} (bottom) and y_{max} (top). The next operations will be used in the pseudocode:

– *increase(coordinate)* - increase the value of a coordinate by 1.

– *decrease(coordinate)* - decrease the value of a coordinate by 1.

– *pdfTool.getTextFrom(region)* - get the text inside a region (it returns the text of the lines of the region from the top line to the bottom one).

4.2 Retrieval of the Header

In this article we are mostly working with official bulletins; the pages of these documents often contain a header, which includes the number of the page, the name and number of the bulletin, the date, etc. Since the goal of the algorithm is to provide a text extraction method to be used in specialised DM and IR systems, it can be assumed that we know in advance if the documents of the domain contain headers.

The location of the header is based in two properties: first, the header is a standard region which always appears at the top of a page. Second, it does not contain several columns, so its extraction requires no complex processing. An example of a page with header is shown in Fig. 2.

The process in detail is shown in the algorithm of Fig. 4. The algorithm initialises a region at the top of the page; this region has the width of the page, and has no height. Then, the region is extended towards the bottom of the page until it contains a line of text, which contains the header.

```
GetHeader(page, pdfTool)
region ← region with full width and no height, at the top of the page
text ← ∅
while text = ∅ and region_y.bottom ≠ y_min
do
    decrease(region_y.bottom)
    text ← pdfTool.getTextFrom(region)
endwhile
if text ≠ ∅
  then
        return region
endif
```

Figure 4: *Algorithm of extraction of the header of a page*

In order to simplify the explanation we have made the assumption that we know if the documents to be extracted have page headers. However, if this assumption does not hold, the presence of a header can be inferred quite easily. By applying the function *GetHeader* to all of the pages, we obtain all of the supposed headers. Since this is a standard part of the documents, all of the headers should be very similar; this can be checked using text similarity techniques. The method to check if a document has header can be used to check if the header contains several lines: we can compare the first line of every page; if they are similar enough to be the header, then we proceed to compare the second line of each page. The process is repeated until line n, which does not hold the similarity comparison: the header will consist of the first $n-1$ lines of each page. A simple improvement of the method can be done by splitting this comparison in even and odd pages, and comparing them independently, as it is very usual that headers in even pages slightly differ from headers in odd pages.

4.3 Computation of the Separation between Columns

LRE requires knowing the size of the separation between text columns in order to obtain the regions of each page. This size will be used to decide the horizontal extension of the regions, as it will be explained in section 4.4.

In order to get the column separation size the algorithm searches for blank regions, with a minimum width, along the x-axis. If the document does not have page headers the height of the blank regions will be the height of the page; otherwise, it will be the distance between the end of the header and the end of a page.

Once the blank regions are obtained, the width of the narrowest region is selected as the column separation size for the page.

The algorithm is shown in Fig. 5; in Fig. 6 the column separations are shown over an image. To keep a certain level of abstraction, the following issues were not considered in the pseudocode:

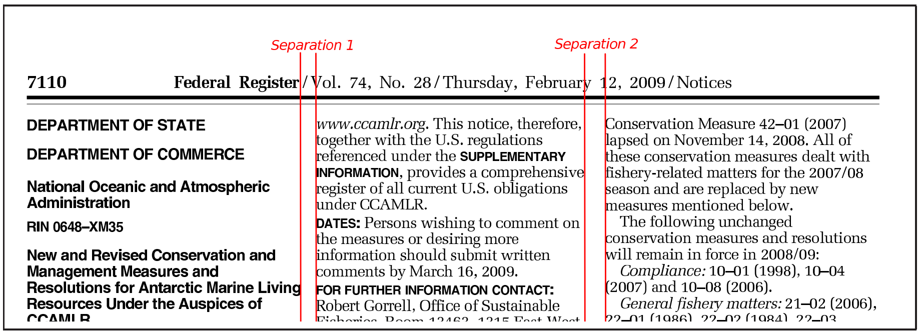
- The left and right indentation of the page are not considered as separations between columns.
- A default value is returned in the case we can not find any separation using this method. This value can be calculated in several ways; the mean of the separations of the other pages, a percentage of the width of the page, the separation of the previous or next page, etc.
- In the pseudocode the regions are initialised to cover the height of the page. However, if the page has a header, the top of these regions should be the bottom of the header, and not the top of the page.

Threshold

The only parameter of the algorithm is the threshold used to decide if a separation is wide enough to be considered as a candidate for the value of the separation between columns. This value is crucial: a high value for the threshold would imply that no separation between columns can be found, and the document would be under-segmented, and a low value for the threshold would imply that too much separations are found, and the document would be over-segmented. We used a fixed value in the complexity and efficiency tests, due to the limited variability of the domain this approximation shows good results. However, to apply

```
GetColumnSeparation(pdfTool, threshold)
result ← default value
region ← region with full height and no width, at the left of the page
text ← ∅
separations ← empty list of separation widths
while region_x.right ≠ x_max
do
    increase(region_x.right)
    text ← pdfTool.getTextFrom(region)
    if text ≠ ∅
      then
            decrease(region_x.right)
            separation ← region_x.right − region_x.left
            if separation ≥ threshold
              then
                    add separation to separations
            endif
            beginPosition ← region_x.right + 1
            region ← region with full height and no width, at beginPosition
    endif
endwhile
if separations ≠ ∅
   then
         result ← min(separations)
endif
return result
```

Figure 5: *Algorithm to obtain the width of the separation between columns*



7110 Federal Register / Vol. 74, No. 28 / Thursday, February 12, 2009 / Notices

DEPARTMENT OF STATE

DEPARTMENT OF COMMERCE

National Oceanic and Atmospheric Administration

RIN 0648–XM35

New and Revised Conservation and Management Measures and Resolutions for Antarctic Marine Living Resources Under the Auspices of CCAMLR

www.ccamlr.org. This notice, therefore, together with the U.S. regulations referenced under the SUPPLEMENTARY INFORMATION, provides a comprehensive register of all current U.S. obligations under CCAMLR.

DATES: Persons wishing to comment on the measures or desiring more information should submit written comments by March 16, 2009.

FOR FURTHER INFORMATION CONTACT: Robert Gorrell, Office of Sustainable

Conservation Measure 42–01 (2007) lapsed on November 14, 2008. All of these conservation measures dealt with fishery-related matters for the 2007/08 season and are replaced by new measures mentioned below.

The following unchanged conservation measures and resolutions will remain in force in 2008/09:

Compliance: 10–01 (1998), 10–04 (2007) and 10–08 (2006).

General fishery matters: 21–02 (2006),

Figure 6: *Columns separations. Taken from the Federal Register; February, 12th 2009; page 7110.*

the algorithm to domains with more variability the next approaches can be taken:

Calculate the threshold according to the width of the pages

A rule of thumb is that the separation should be at least a 5% of the page width: the width of a simple character of the content (not a character of a title) rarely exceeds a 1% of the page width, so with this setting we are assuming that a separation between columns should be larger than at least five times the width of a character. The advantage of this method is that it is very easily computed, and it is valid for most of the documents of the domain. The main disadvantage is that it would fail when a page is an outlier: for instance, the modal font size of a page may be very small, and so would be the separations between columns.

Calculate the threshold according to the font size of the page
Another rule of thumb is to compute the threshold considering the modal font size of the characters in the page. Firstly, the bounding boxes for the characters whose font size is equals to the mode are retrieved. Next the mean width of those bounding boxes is computed, let us name this mean as δ. Once this δ is obtained we can set the value of the threshold to five times δ, loosely speaking, this means that column separation should be at least five times larger than the average width of a character. The main advantage of this approximation is that the obtained threshold value is very accurate; the main disadvantage is that it adds some computational load to the algorithm.

4.4 Retrieval of the Regions Located on the Left of the Page

This step of the algorithm uses the results from the previous steps (header and width of column separation) to obtain the regions on the left of a page, which will be used in section 4.5 to obtain the remaining regions of a page. There are two main steps to obtain each left region:

- Locating the $region_{y.top}$, and initialising the region (algorithm in Fig. 7).
- Extending the region until the end of a page is reached, or a different region begins (algorithm in Fig. 8).

```
GetInitialRegion(upperlimit, pdfTool)
region ← region with full width and no height, at the top of the page
text ← ∅
while text = ∅ and region_y.bottom ≠ y_min
do
    decrease(region_y.bottom)
    text ← pdfTool.getTextFrom(region)
endwhile
if text ≠ ∅
  then
        region_x.left ← x_min
        region_x.right ← start of the first column separation inside the region

endif
return region
```

Figure 7: *Algorithm for regions' initialisation*

```
ExtendRegion(region, pdfTool, columnSeparation)
newRegion ← region with the x-axes of region and as y-axes region_y.bottom
text ← ∅
while text = ∅ and newRegion_y.bottom ≠ y_min
do
    decrease(newRegion_y.bottom)
    text ← pdfTool.getTextFrom(newRegion)
endwhile
if text = ∅
  then
        region_y.bottom ← y_min
        return region
endif
while no CS found on the right of newRegion and newRegion_x.right ≠ x_max
do
    increase(newRegion_x.right)
endwhile
if newRegion contains a CS
  then
        return region
endif
if newRegion_x.right ≤ region_x.right
  then
        region_y.bottom ← newRegion_y.bottom
        return extendRegion(region, pdfTool, columnSeparation)
endif
compareRegion ← region
compareRegion_x.right ← newRegion_x.right
text ← pdfTool.getTextFrom(region)
compareText ← pdfTool.getTextFrom(compareRegion)
if text = compareText and
  compareRegion has a CS on his right
  then
        region_y.bottom ← newRegion_y.bottom
        region_x.right ← newRegion_x.right
        return extendRegion(region, pdfTool, columnSeparation)
  else
        return region
endif
```

Figure 8: *Algorithm of extension of a region. The abbreviation CS specifies a blank space with a width equal or greater than the column separation*

First, the top of the upper left region is located. Afterwards, the region is extended by decreasing $region_{y.bottom}$ in order to obtain the initial region. When it can not be further extended, the steps are repeatedly applied to compute the region on the left below the previously found one: the top of the region is located, the region is extended and another left region below the last one is searched for.

The algorithm in Fig. 7 computes the top of a region. It is computed by initialising a blank region, without height and with the width of the page. The values of $region_{y.top}$ and $region_{y.bottom}$ are initialised to the bottom of the last found region (it may be the header). If there is not any previous found region, the values are set to y_{max}. Then, the region is extended towards the bottom of the page until it contains some text; $region_{y.bottom}$ is set to the value of the y-coordinate in which the text was found, $region_{x.left}$ is set to the value of x_{min}. Also, it is computed the x-coordinate in which the found text contains the first blank space wider than the size of the column separation: $region_{x.right}$ is set to the value of the start of that blank space.

The algorithm in Fig. 8 computes the extension of the region. Along this process, $region_{x.left}$ and $region_{y.top}$ will remain unchanged. The algorithm tries to find a text line below the current limit of the region. If there is no such line the end of the region is reached (the region is stored); otherwise, the region is extended to include the line ($region_{x.right}$ may be increased, and $region_{y.bottom}$ is decreased). An exception to this extension occurs when the line below the region has a different column structure than the region itself: in this case the region is stored, and a new region begins with the new line. The conditions to determine that the column structure is different are:

(a) The line immediately below the region contains a blank space wider than the size of the column separation for the page, and all of the width of this space is entirely below the region. Fig. 9 shows an example: the line following the region contains a blank space (filled with gray) below the region (the checking of this condition is done in lines 17 to 20 of the algorithm in Fig. 8). An example in which this conditions do not hold is shown in Fig. 10: the $region_{x.right}$ is extended towards the $x._{right}$ of the line, and the $region_{y.bottom}$ is extended towards the $y._{bottom}$ of the line.

Figure 9: *Condition (a) to end a region. Taken from the European Bulletin; October, 2008; page 8.*

(b) The line following the region is wider than the region (according to x-axis). In this case, if there is more text above that line than the text contained in the region (as shown in Fig. 11.1), or if the region would not fulfil the minimum separation between columns in the case that its x_{right} is set to the end of the next line (as shown in Fig. 11.2), the region ends (the checking of these conditions is done in lines 26 to 38 of the algorithm in Fig. 8).

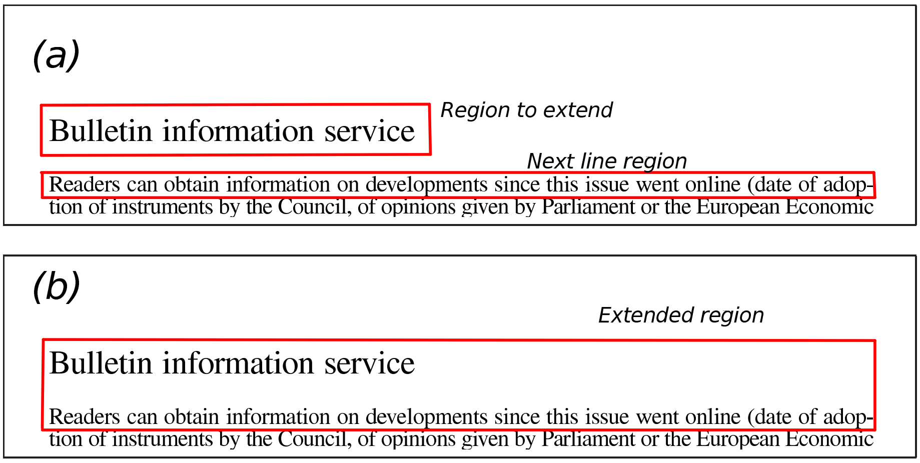


Figure 10: *(a) Region before being extended, and next line. (b) Region after being extended with the next line. Taken from the European Bulletin; April, 2008; page 2.*

4.5 Retrieval of the Remaining Regions

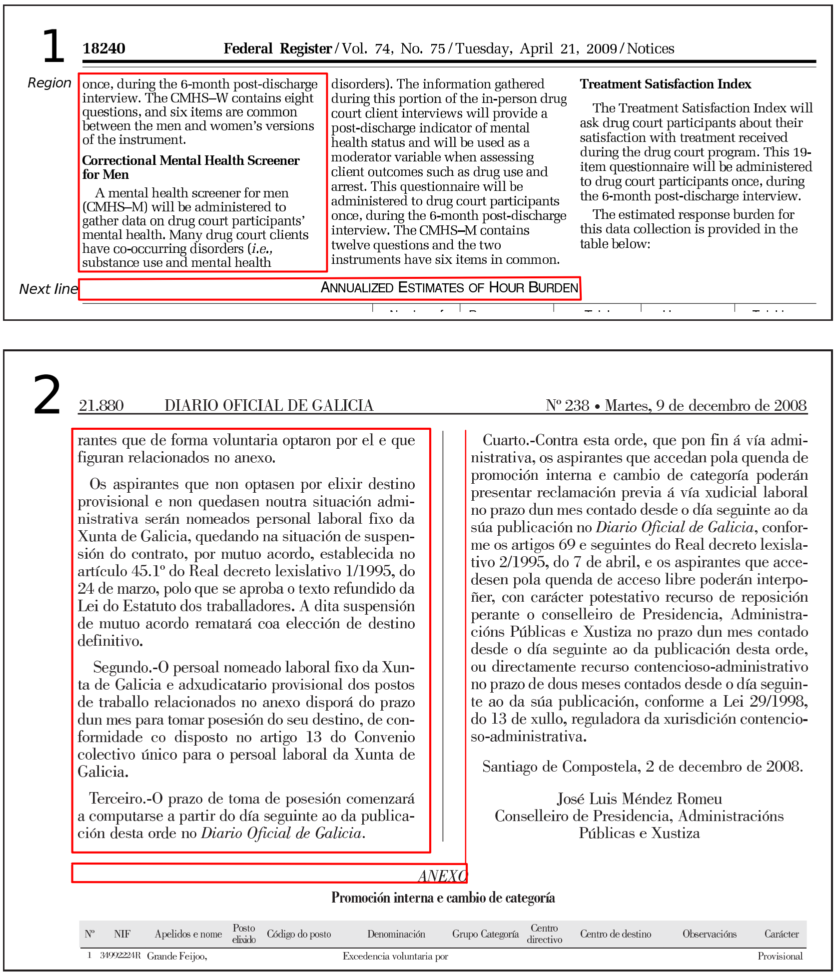
At this point of the algorithm, we have retrieved the header and the regions on the left of a page. With these left regions, the remaining regions of the page can be obtained searching for regions parallel to them (with similar y-coordinates). Also, there can be several regions parallel to a left one: they will be separated by blank spaces wider than a threshold. The algorithm is shown in Fig. 12.

4.6 Retrieval of the Images, and Modification of the Obtained Text Regions

As previously stated, the images could modify the regions previously obtained. This is easily explained using Fig. 2: if the images were not considered, regions 1 and 4 would be extracted as one region, as would be regions 2 and 5, and 3 and 6. But, using the coordinates of the image, the algorithm identifies that the image crosses the regions, meaning a horizontal cut in the layout.

When an image modifies the layout of one region, all of the regions with the same y-coordinates are affected in the same way if they are also crossed by that image. The modification consists in the split of the region into two regions, the portion above the image and the portion below it. An image will modify the regions previously found if:

– Overlaps at least two text regions.



18240 Federal Register / Vol. 74, No. 75 / Tuesday, April 21, 2009 / Notices

once, during the 6-month post-discharge interview. The CMHS–W contains eight questions, and six items are common between the men and women's versions of the instrument.

Correctional Mental Health Screener for Men

A mental health screener for men (CMHS–M) will be administered to gather data on drug court participants' mental health. Many drug court clients have co-occurring disorders (*i.e.*, substance use and mental health disorders). The information gathered during this portion of the in-person drug court client interviews will provide a post-discharge indicator of mental health status and will be used as a moderator variable when assessing client outcomes such as drug use and arrest. This questionnaire will be administered to drug court participants once, during the 6-month post-discharge interview. The CMHS–M contains twelve questions and the two instruments have six items in common.

Treatment Satisfaction Index

The Treatment Satisfaction Index will ask drug court participants about their satisfaction with treatment received during the drug court program. This 19-item questionnaire will be administered to drug court participants once, during the 6-month post-discharge interview.

The estimated response burden for this data collection is provided in the table below:

ANNUALIZED ESTIMATES OF HOUR BURDEN

21.880 DIARIO OFICIAL DE GALICIA Nº 238 • Martes, 9 de decembro de 2008

rantes que de forma voluntaria optaron por el e que figuran relacionados no anexo.

Os aspirantes que non optasen por elixir destino provisional e non quedasen noutra situación administrativa serán nomeados persoal laboral fixo da Xunta de Galicia, quedando na situación de suspensión do contrato, por mutuo acordo, establecida no artículo 45.1º do Real decreto lexislativo 1/1995, do 24 de marzo, polo que se aproba o texto refundido da Lei do Estatuto dos traballadores. A dita suspensión de mutuo acordo rematará coa elección de destino definitivo.

Segundo.-O persoal nomeado laboral fixo da Xunta de Galicia e adxudicatario provisional dos postos de traballo relacionados no anexo disporá do prazo dun mes para tomar posesión do seu destino, de conformidade co disposto no artigo 13 do Convenio colectivo único para o persoal laboral da Xunta de Galicia.

Terceiro.-O prazo de toma de posesión comenzará a computarse a partir do día seguinte ao da publicación desta orde no *Diario Oficial de Galicia*.

Cuarto.-Contra esta orde, que pon fin á vía administrativa, os aspirantes que accedan pola quenda de promoción interna e cambio de categoría poderán presentar reclamación previa á vía xudicial laboral no prazo dun mes contado desde o día seguinte ao da súa publicación no *Diario Oficial de Galicia*, conforme os artigos 69 e seguintes do Real decreto lexislativo 2/1995, do 7 de abril, e os aspirantes que accedesen pola quenda de acceso libre poderán interpoñer, con carácter potestativo recurso de reposición perante o conselleiro de Presidencia, Administracións Públicas e Xustiza no prazo dun mes contado desde o día seguinte ao da publicación desta orde, ou directamente recurso contencioso-administrativo no prazo de dous meses contados desde o día seguinte ao da súa publicación, conforme a Lei 29/1998, do 13 de xullo, reguladora da xurisdición contencioso-administrativa.

Santiago de Compostela, 2 de decembro de 2008.

José Luis Méndez Romeu
Conselleiro de Presidencia, Administracións
Públicas e Xustiza

ANEXO

Promoción interna e cambio de categoría

Nº	NIF	Apelidos e nome	Posto elixido	Código do posto	Denominación	Grupo Categoría	Centro directivo	Centro de destino	Observacións	Carácter
1	34992224R	Grande Feijoo,			Excedencia voluntaria por					Provisional

Figure 11: *Condition (b) to end a region. (1) Taken from the Federal Register; April, 21th 2009; page 18240. (2) Taken from the Diario Oficial de Galicia; December, 9th 2008; page 21880.*

– Overlaps one region, and extending the region to contain the image would not fulfil any of the properties given about the regions in section 3.

This operation could be introduced in the *ExtendRegion* algorithm, in 4.4. But, by keeping it in a separate function, it will be easier to treat the particular cases we could find. Also, despite the fact that image regions and text regions are used in the same way, the implementation of getting the images and getting the text regions is quite different.

```
getRightRegions(leftRegion, columnSeparation)
blankRegion ← region with the y-coordinates of leftRegion
blankRegion_x.left ← leftRegion_x.right + 1
blankRegion_x.right ← blankRegion_x.left + columnSeparation
separations ← empty list of blank regions
result ← empty list of regions
while blankRegion_x.right ≤ x_max
do
    text ← getTextFrom(blankRegion)
    if text ≠ ∅
       then
            increase(blankRegion_x.left)
            increase(blankRegion_x.right)
       else
            increase(blankRegion_x.right)
            text ← getTextFrom(blankRegion)
            while blankRegion_x.right ≤ x_max and text = ∅
            do
                increase(blankRegion_x.right)
                text ← getTextFrom(blankRegion)
            endwhile
            decrease(blankRegion_x.right)
             add copyofblankRegion to separations
            blankRegion_x.left ← blankRegion_x.right + 1
            blankRegion_x.right ← blankRegion_x.left + columnSeparation
    endif
endwhile
result ← list of regions delimited by separations
return result
```

Figure 12: *Algorithm to get the regions on the right of a page*

4.7 Retrieval of the Text of each Page Using the Regions

In the last step, the text of the regions is extracted in the order stated in 4.1. At this point, it could be interesting to return the original structure of the page, instead of the plain text, by returning the regions. Moreover, by returning the text regions along with the image regions, the original content of a document could be reconstructed to offer it to an user in a similar way in which it was originally published.

5 Evaluation

The algorithm was evaluated in two ways:

– Effectiveness: correction of the extraction process, i.e., the ratio of pages correctly extracted. We implemented the algorithm using Java and we compared the results of our method with *PDFBox* and XY-cuts. We also tested *Multivalent*, but the results are not reported because they were exactly the same as with the *PDFBox* tool.

– Computing complexity: the asymptotic analysis of the pseudocode shows that the algorithm is linear over the number of pages. We also ran analysis experiments to check the complexity.

5.1 Dataset

5.1.1 Dataset for the effectiveness study

This collection was built from eight sources [1]: official publications from the European Union, the United States of America, England, France and Spain. From each source, two bulletins were selected, providing a total of 1024 pages for the evaluation. This is an heterogeneous selection according to the number of pages per document; some sources contain hundreds of pages per document, and some sources tens of pages. However, the target of this dataset is to provide a complete representation of each source, thus entire bulletins were used: it may happen that the first section of a bulletin uses a particular structure (two-column, for instance), and some other section a different structure (three-column, for instance).

It must be noted that some of the documents contain empty pages; these pages were excluded from the effectiveness evaluation (they have no text to be extracted). The information about the total number of non-empty pages from each source follows:

- *The Bulletin of the European Union* (BEU): 300 pages, [`http://europa.eu/bulletin/en/welcome.htm`].
- *Official Journal of the European Union* (OJEU): 30 pages, [`http://eur-lex.europa.eu/`].
- *Federal Register* (FR): 89 pages, [`http://www.gpoaccess.gov/fr/`].
- *Command Papers* and *House of Commons Papers* (UK): 62 pages, [`http://www.official-documents.gov.uk/`].
- *Journal Officiel* (JO): 56 pages, [`http://www.journal-officiel.gouv.fr/`].
- *Boletín Oficial del Estado* (BOE): 63 pages, [`http://www.boe.es/`].
- *Diario Oficial de Galicia* (DOG): 128 pages, [`http://www.xunta.es/diario-oficial`].
- *Boletín Oficial de Castilla y León* (BOCYL): 296 pages, [`http://bocyl.jcyl.es/bocyl/`].

5.1.2 Dataset for the complexity study

The second dataset was built to check the complexity of the algorithm. A single PDF file was built by merging all of the European Bulletin publications between January 2007 and June 2009. This file provided 3949 pages to run the tests.

[1] The evaluation dataset is published in `http://www.irlab.org/files/lre_dataset.zip`

5.2 Methodology

5.2.1 Methodology for the effectiveness study

The evaluation metric to compare the result of our algorithm with the result from PDFBox was the ratio of the pages correctly extracted. We consider that the text of a page is correctly extracted when all of the text is retrieved, and in the right order. We manually inspected the results for the aforementioned collection.

To conduct the experiment, the *PDFBox* text extraction method, a XY-cuts implementation and our implemented method were applied to the pages of the bulletins presented in 5.1, in order to obtain the text. The outputs were manually checked to get data about the number of pages correctly extracted. PDFBox does not need parameters to extract the text; neither does LRE, a priori. But it may happen that the algorithm cannot infer the separation between columns for a document, and must use a default value. This value was not tunned for every source: the same value was used for every document.

It should be said that sometimes *PDFBox* repeatedly made the same extraction error. For instance, with the *UK* bulletins the header is always extracted at the end of a page. In the evaluation, these occurrences were not considered as wrong extracted pages. The reason is that they would systematically penalise the extraction with the *PDFBox* tool.

Also, we implemented the XY-cuts algorithm by adapting the pseudocode from [Mao and Kanungo 2001] to compare the detection of the layout offered by that method. Unfortunately, this technique is not stable on the input parameters [Meunier 2005]: the documents from different sources of legal information have different interlinear and intercolumn spaces, so the parameters tunned for a source of documents produce under-segmentation with some sources (ignoring vertical and horizontal separations), and over-segmentation with other sources (separating words inside the same line, lines inside the same paragraph, etc). This explains the poor behaviour of XY-cuts in several bulletins.

Finally, as previously stated, we must mention that the extraction of tables is beyond the scope of the present work, and it was not considered in the evaluation.

5.2.2 Methodology for the complexity study

The complexity of our algorithm was theoretically computed by executing an asymptotic analysis. Also, the complexity was empirically computed: the algorithm was run several times to extract the text from $10 * 2^x$ pages, being $[0, 9]$ the range of x; the computing time of each run was stored and the complexity was obtained by applying convergence analysis techniques to the obtained data.

5.3 Results

5.3.1 Effectiveness results

The results are shown in Table 1. The table shows the sources, the number of pages from each source, and the ratio of the pages correctly extracted by the XY-cuts algorithm, *PDFBox* and LRE, our algorithm. As explained earlier, the XY-cuts algorithm does not perform well, so it will not be commented again. It can be observed that the results of PDFBox are quiet acceptable, with a total ratio of 87% of the pages correctly extracted. However, our algorithm improves these results. For all of the sources, our algorithm's result stands over 90%, and *PDFBox* results can lower to 57%. Moreover, it must be noted that in the evaluation the errors produced by *PDFBox* in a systematic way, along an entire document, were ignored. The improvement is maintained over every kind of bulletin.

Bulletin	Pages	XY-cuts	PDFBox	LRE
BEU	300	0.23	0.95	0.96
OJEU	30	0.33	0.87	0.97
FR	89	0.08	0.57	0.92
UK	62	0.08	0.92	0.98
JO	56	0.77	0.80	1.00
BOE	63	0.78	1.00	1.00
DOG	128	0.19	0.77	0.91
BOCYL	296	0.83	0.91	0.98
Overall	**1024**	**0.44**	**0.87**†	**0.96**†⋆

Table 1: *Ratio of pages correctly extracted from each source, using a XY-cuts implementation, PDFBox and our method. Statistical significance differences of the mean over the XY-cuts algorithm are daggered, and over the PDFBox API are starred, both according with the Wilcoxon test, ($p < 0.05$).*

5.3.2 Complexity results

Although the efficiency was not the objective of this paper, we have to remark that, as expected, our algorithm is slower than the standard PDFBox API, but its complexity still is O(n), being n the number of pages. The linearity of the algorithm on the number of pages can be examined by reviewing the process of extraction of text from a document. For each page, it repeats a constant number of steps: detection of the header, detection of the column separation, detection of the left regions, detection of the remaining regions, detection of the images and extraction of text. These steps are executed at the level of a page, so the algorithm is linear on the number of pages. The linear complexity of the algorithm was also empirically confirmed, through an analysis of convergence. It also can be argued that the algorithm is dependent on the number of lines per page, the number of columns or even the size in pixels due to the fact that the algorithm is based on pixel steps. Despite this argumentation, we can assume maximum values for these variables, so they do not add complexity to the algorithm: to the best of our knowledge, there are not legal documents with more than 150 lines per page, neither with more than 5 columns per page nor with more than 2000 pixels per page height or per page width. Moreover this point has been thoroughly confirmed with a convergence analysis over the number of total processed lines. The results of that analysis are that the time complexity is also linear over the number of lines.

The execution time results are shown in Table 2; the collection is the one exposed in section 5.1.1, with the difference that the empty pages were included in the execution time comparison because, despite the fact they are empty, they are processed by both *PDFBox* and LRE, and so they increase the execution time. As stated before, PDFBox is faster than LRE, being capable of processing a page in 107 ms (LRE needs 1 347 ms)[2]. However, our algorithm can process more than 2 600 pages per hour, and still seems suitable to be used in a system which is daily updated.

Bulletin		PDFBox		LRE	
Name	Pages	Total	Per Page	Total	Per Page
BEU	300	20 928	69.76	196 295	654.32
OJEU	30	1 854	61.8	37 599	1 253.3
FR	89	14 667	164.8	156 329	1 756.51
UK	66	3 338	50.58	66 508	1 007.7
JO	56	3 774	67.39	89 226	1 593.32
BOE	63	5 958	94.57	90 317	1 433.6
DOG	128	28 041	219.07	208 185	1 626.45
BOCYL	400	42 689	106.72	680 287	1 700.72
Overall	**1132**	**121 249**	**107.11**	**1 524 746**	**1 346.95**

Table 2: *Execution time of PDFBox and LRE (ms)*

[2] Tests are run in a Quad CPU at 2.40GHz with 4 GB of RAM

6 Conclusions and Future Work

In this work we developed an algorithm to extract the text from PDF documents with a *Manhattan layout* which preserves the original column-format structure.

The results are better than the ones obtained with XY-cuts and *PDFBox*, an extended PDF text extraction solution; moreover, the algorithm can offer more information than this tool (the column structure of the document).

An implementation of the algorithm is currently being used in a legislative information retrieval system, where it offers good results; also, it can be used in different domains: scientific papers, papers in magazines, textbooks and handbooks.

There are several interesting issues in which we will continue with this work. Some of them are the following:

– The page headers usually can be represented with a regular expression. If the expression is known, it becomes very easy to extract the header. Also, the metainformation of the header can be obtained for further processing.

– At this moment, the algorithm does not care about the tables. However, they mean a noise factor which could affect to the correct working of the algorithm. It would be great to add some step in the algorithm so it can detect tables and extract them in a "right way". The works of Liu et al. [Liu et al. 2009, Liu et al. 2008a, Liu et al. 2008b] in the field of table boundaries detection are very interesting, and they can be combined with our approach so the tables inside the text are identified. Also, the XY-cuts algorithm seems to be very suitable to the task of segmenting the contents of a table according to the experiments we observed.

– The images of a document can contain textual information such as scanned texts, for instance. It would be interesting the use OCR techniques to analyse these images, and to integrate their content with the rest of the extracted text.

– The algorithm does not use the information about font types, colours, sizes, etc. These information could be used to improve the effectiveness of the method.

– It can be interesting to allow region nesting and to develop a reliable method to obtain them, therefore the result would contain a hierarchy of regions.

References

[Baird et al. 1990] Baird, H., Jones, S., and Fortune, S. (1990). "Image segmentation by shape-directed covers". In *ICPR '99: Proceedings of the International Conference on Pattern Recognition*, pages 1:820–825.

[Bloechle et al. 2009] Bloechle, J.-L., Lalanne, D., and Ingold, R. (2009). "OCD: An optimized and canonical document format". In *ICDAR '09: Proceedings of the 10th International Conference on Document Analysis and Recognition*, pages 236–240, Washington, DC, USA. IEEE Computer Society.

[Bloechle et al. 2008] Bloechle, J.-L., Pugin, C., and Ingold, R. (2008). "Dolores: An interactive and class-free approach for document logical restructuring". In *DAS '08: Proceedings of the 8th IAPR International Workshop on Document Analysis Systems*, pages 644–652, Washington, DC, USA. IEEE Computer Society.

[Bloechle et al. 2006] Bloechle, J.-L., Rigamonti, M., Hadjar, K., Lalanne, D., and Ingold, R. (2006). "XCDF: A canonical and structured document format". In *DAS'06: Proceedings of the 7th IAPR International Workshop on Document Analysis Systems*, pages 141–152.

[Chakraborty et al. 2003] Chakraborty, A., Liu, P., and Hsu, L. (2003). "Extracting anchorable information units from PDF files". In *ICME '03: Proceedings of the 2003 International Conference on Multimedia and Expo*, pages 173–176, Washington, DC, USA. IEEE Computer Society.

[Krishnamoorthy et al. 1993] Krishnamoorthy, M., Nagy, G., Seth, S., and Viswanathan, M. (1993). "Syntactic segmentation and labeling of digitized pages from technical journals". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(7):737–747.

[Liu et al. 2009] Liu, Y., Bai, K., Mitra, P., and Giles, C. L. (2009). "Improving the table boundary detection in PDFs by fixing the sequence error of the sparse lines". In *ICDAR '09: Proceedings of the 10th International Conference on Document Analysis and Recognition*, pages 1006–1010, Washington, DC, USA. IEEE Computer Society.

[Liu et al. 2008a] Liu, Y., Mitra, P., and Giles, C. L. (2008a). "A fast preprocessing method for table boundary detection: Narrowing down the sparse lines using solely coordinate information". In *DAS '08: Proceedings of the 8th IAPR International Workshop on Document Analysis Systems*, pages 431–438, Washington, DC, USA. IEEE Computer Society.

[Liu et al. 2008b] Liu, Y., Mitra, P., and Giles, C. L. (2008b). "Identifying table boundaries in digital documents via sparse line detection". In *CIKM '08: Proceeding of the 17th ACM Conference on Information and Knowledge Management*, pages 1311–1320, New York, NY, USA. ACM.

[Lovegrove et al. 1995] Lovegrove, W. S., David, and Brailsford, F. (1995). "Document analysis of PDF files: methods, results and implications". *Electronic Publishing*, 8(3):207–220.

[Mao and Kanungo 2001] Mao, S. and Kanungo, T. (2001). "Empirical performance evaluation methodology and its application to page segmentation algorithms". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):242–256.

[McKinley 1997] McKinley, T. (1997). "Why PDF is everywhere". *Inform, the journal of AIIM*, 11(8).

[Meunier 2005] Meunier, J.-L. (2005). "Optimized XY-Cut for determining a page reading order". In *ICDAR '05: Proceedings of the 8th International Conference on Document Analysis and Recognition*, pages 347–351, Washington, DC, USA. IEEE Computer Society.

[Mitchell 2000] Mitchell, P. (2000). "Document page segmentation and layout analysis using soft ordering". In *ICPR '00: Proceedings of the 15th International Conference on Pattern Recognition*, volume 1, page 1458, Washington, DC, USA. IEEE Computer Society.

[Multivalent 2010] Multivalent (2010). "Multivalent". `http://multivalent.sourceforge.net/`.

[Nagy and Seth 1984] Nagy, G. and Seth, S. (1984). "Hierarchical representation of optically scanned documents". In *ICPR '84: Proceedings of the 7th International Conference on Pattern Recognition*, pages 347–349.

[O'Gorman 1993] O'Gorman, L. (1993). "The document spectrum for page layout analysis". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1162–1173.

[PDF-Analyzer 2010] PDF-Analyzer (2010). "Pdf-analyzer". `http://www.pdf-analyzer.com/`.

[PDF-File-Converter 2010] PDF-File-Converter (2010). "Pdf-file converter". `http://www.pdf-file.com/`.

[PDFBox 2010] PDFBox (2010). "Pdfbox". `http://pdfbox.apache.org/`.

[PlanetPdf 2010] PlanetPdf (2010). "Planetpdf". `http://www.planetpdf.com/`.

[Pstotext 2010] Pstotext (2010). "Pstotext". `http://pages.cs.wisc.edu/~ghost/doc/pstotext.htm`.

[Rosenfeld et al. 2002] Rosenfeld, B., Feldman, R., and Aumann, Y. (2002). "Structural extraction from visual layout of documents". In *CIKM '02: Proceedings of the 11th ACM Conference on Information and Knowledge Management*, pages 203–210, New York, NY, USA. ACM.

[Simon et al. 1997] Simon, A., Pret, J.-C., and Johnson, A. P. (1997). "A fast algorithm for bottom-up document layout analysis". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):273–277.

[TextFromPdf 2010] TextFromPdf (2010). "Textfrompdf". `http://www.textfrompdf.com/`.