Egyptian Informatics Journal 22 (2021) 329–338

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Aligning document layouts extracted with different OCR engines with clustering approach

S. Tomovic [a,1,*], K. Pavlovic [a,2], M. Bajceta [b]

[a] *Faculty of Mathematics and Natural Sciences, University of Montenegro, Cetinjska bb, 81000 Podgorica, Montenegro*
[b] *Datum Solutions, 81000 Podogrica, Montenegro*

A R T I C L E   I N F O

Article history:
Received 9 February 2020
Revised 20 October 2020
Accepted 11 December 2020
Available online 30 December 2020

Keywords:
Document layout
Document indexing
Information retrieval
Clustering
OCR

A B S T R A C T

Layout analysis is essential step in information extraction from scanned document images. In this paper we propose an algorithm for aligning layouts generated with different OCR engines. The main requirement is to always generate the same layout for the given document image regardless of OCR engine used for image processing. In that way information extraction from scanned documents, that is heavily dependent on fields positions in the document, does not depend on specific OCR engine. In other words, it is sufficient to maintain universal extractor knowledge and not necessary to train extractor explicitly with samples processed on specific OCR engine. The proposed algorithm can handle administrative documents with complex layout.

1. Introduction

The main purpose of layout aligning algorithm - LAA presented in this paper is to uniformly determine text lines on semi-structured and unstructured documents processed with different OCR engines.

LAA creates unique document layout representation from which software for information extraction is learned to identify target fields on documents. The extractor is machine learning based solution that is trained on available layouts from a training dataset. Learning procedure is heavily dependent on document layout containing positions of target fields on the document.

The motif for this research is to implement document understanding system that must be capable to extract target fields from document images processed with different OCR engines. Different OCR engines generate different layouts for the same document image. Layouts are stored in searchable PDF format. In addition, we do not have any information which OCR engine is used to generate searchable PDF which is input to the extraction module.

So, the main task for LAA is to provide unique document layout representation regardless of used OCR engine. Accordingly, extractor will not depend on OCR engine specificities and it will be sufficient to maintain only universal extractor knowledge that can handle searchable PDFs from any OCR engine.

Without LAA module, we must train and maintain specific extractor for each OCR engine. Although this solution is not efficient, it is not possible to implement it in situation where we can not know which engine is used to generate specific searchable PDF. So, we must have LAA module than converts any layout representation to unique form from which extractor can identify target fields.

LAA can be applied even in situation when only one OCR engine is used. In such case, extractor module is devoted only to learning how to extract information from documents and it is free off procedures that pre-process document layout (filtering, normalization, OCR errors correction etc.) which are now part of the LAA module. With such design document understanding system becomes more flexible because different combinations of extractor learning algorithms and pre-processing methods can be supported.

The LAA can be implemented as pre-processing step for general document understanding system that is required to process intensive document streams. Document understanding systems can be defined as software solutions which automatize immediate processing of administrative documents with minimal human intervention [21]. These systems receive document images as input. Documents can be very different in term of their structure and

* Corresponding author.
E-mail addresses: savot@ucg.ac.me (S. Tomovic), kosta@ucg.ac.me (K. Pavlovic), milija.bajceta@datum.solutions.net (M. Bajceta).
[1] www.ucg.ac.me/savot
[2] www.ucg.ac.me/kosta

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

https://doi.org/10.1016/j.eij.2020.12.004
1110-8665/© 2021 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Computers and Artificial Intelligence, Cairo University.

include invoices, forms, contracts, requests, letters etc. The task is to find and extract relevant information from these documents. For example, from the electricity invoice the system can extract and save in database information like date, total amount and customer name. Later, these data can be used for faster document searching or as input to the decision support systems. On Fig. 1, the values extracted from an invoice document are presented.

The basic use case in a document understanding system is as follows [2]. Documents are arriving from the stream. The system takes the current document and runs specific OCR engine that converts document image into searchable PDF file. Generated file is, after that, sent to the information extraction module responsible for extraction of target information. But the input stream can contain searchable PDFs instead of images. It is not possible to know source of the searchable PDF file. For instance, the system does not know which OCR engine is used to generate PDF or if the PDF is digitally created. In the case when searchable PDF file arrives from the stream, it is directly sent to the information extraction module.

According to the previous, input to the information extraction module is always document in searchable PDF format. PDF documents are stored as set of very complex instructions that determine how elements will be situated on the document. This representation does not contain complex structural elements such as sentences or paragraphs.

There are several libraries for parsing PDF documents. In this study we used PDFMiner.

Document layout represents document geometry. The main information contained in layout is position and size of text fields. Many information extraction approaches are dependent on document layout because their training consists of learning spatial relations between fields [12].

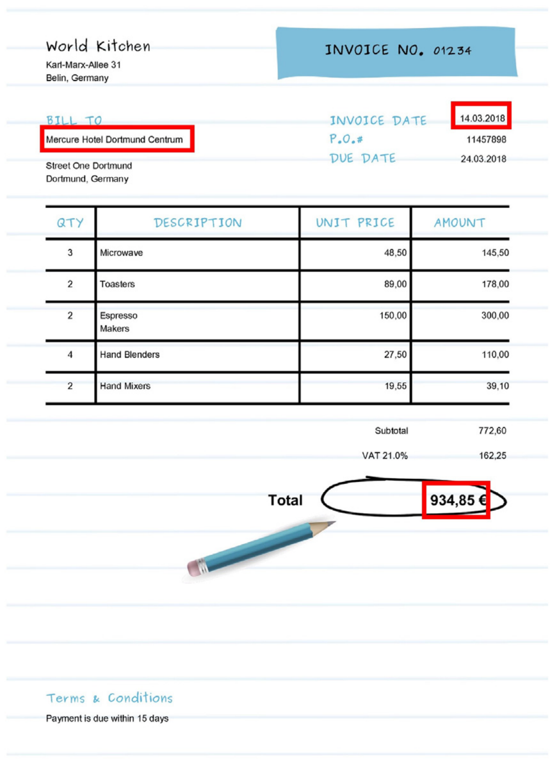
On Fig. 2 we illustrated extension of the basic use case in a document understanding system with the LAA procedure. The purpose of the LAA step is to make information extraction module independent on specific OCR engine. In this way we can use a universal model for information extraction and handle situations where different OCR engines are used for model training and document stream processing.

In the proposed flow (Fig. 2), the LAA module will process searchable PDF representation of a document image and generate corresponding layout. Document layout generated with LAA consists of text lines sorted in appropriate order for further processing.

Line extraction is essential for overall accuracy of many information extraction systems such as SROIE systems (Scanned Receipts OCR and Information Extraction) [16,15,10,14]. For example, the field Total and corresponding amount must be placed into the same line when processing document from the Fig. 1. Similarly, BILL TO must be placed immediately above *Mercure Hotel Dortmund Center* enabling the extractor to locate correct customer name from the invoice document.

Additionally, the LAA is responsible for achieving information extraction module robustness: the same layout should be generated regardless of document source. We want to eliminate any dependence on OCR engine, eventually searchable PDF can be digitally created. Further, the LAA should be capable to deal with possible anomalies regarding document geometry. For example, when document is generated and/or scanned some fields can be rotated or translated in unexpected way. The LAA must flatten and smooth such undesirable transformations and create layout that is as close as possible to ideal case.

The paper is organized as follows. The next section presents related work. Motivation, novelty and contribution of the proposed method is discussed in the third section. The forth section introduces LAA algorithm for layout alignment. Along with the main idea several modifications are considered. The fifth section is devoted to discussion about experimental protocols, performance metrics and registered results. Finally, the last section contains conclusion and proposals for possible extensions and improvements.



World Kitchen
Karl-Marx-Allee 31
Belin, Germany

INVOICE NO. 01234

BILL TO
Mercure Hotel Dortmund Centrum
Street One Dortmund
Dortmund, Germany

INVOICE DATE	14.03.2018
P.O.#	11457898
DUE DATE	24.03.2018

QTY	DESCRIPTION	UNIT PRICE	AMOUNT
3	Microwave	48,50	145,50
2	Toasters	89,00	178,00
2	Espresso Makers	150,00	300,00
4	Hand Blenders	27,50	110,00
2	Hand Mixers	19,55	39,10

Subtotal	772,60
VAT 21.0%	162,25
Total	934,85 €

Terms & Conditions
Payment is due within 15 days

Fig. 1. Date, customer name and total amount are located on invoice document.

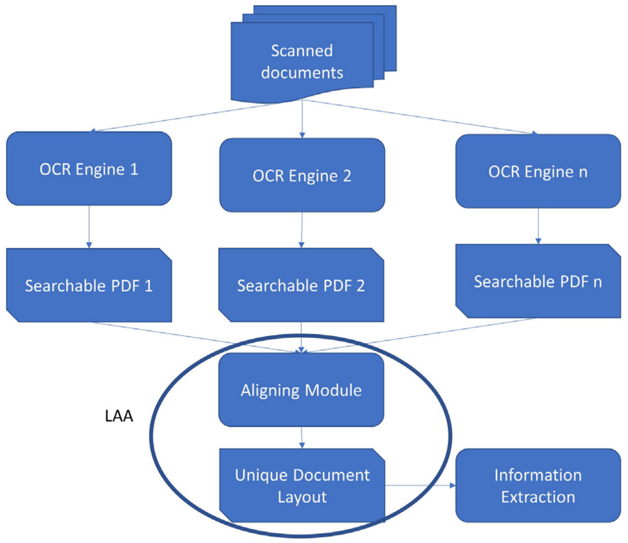


Fig. 2. Document understanding system with LAA step. LAA step is responsible to produce unique document layout regardless of OCR engine used. Information extraction module works with such layout and it is not dependant on OCR engine.

2. Related work

The most similar problem known in the literature to the problem exposed in this study is document layout analysis of document images. Document layout analysis is the field of intensively research. Briefly, it consists of document segmentation that divides document image into segments. Further, recognized segments can be classified as logos, glyphs, words, lines, paragraphs, blocks etc.

In general, algorithms for document analysis combines several procedures, such as binarization, noise removal, skew correction, page segmentation, segment classification, reading order determination [8]. Algorithms vary with the respect to how these procedures are performed.

Page segmentation is the most important step in document layout analysis. The page segmentation algorithms divide a document image into homogeneous segments. Each segment is part of physical layout structure and can represent text, typewritten text, graphics, diagram, logo, etc. Line segmentation algorithms detect the beginning, the end, the top and the bottom of each text line. As it is mentioned in the introductory section, in this study we concentrate on line detection. So, physical document layout can be represented as set of recognized text lines.

Apart physical layout, in the literature are proposed algorithms for logical layout extraction. The logical layout determines the semantic of each region in the image such as header, body, footnote, etc.

Review of representative page segmentation algorithms before 2008 can be found in [23]. Primarily, authors classified algorithms into two classes, namely zone-based algorithms and line-based algorithms.

The zone-based algorithms extract text blocks from document images. The blocks are further decomposed into text-lines. This class of algorithms includes X-Y cut [17], Whitespace Analysis [3], Docstrum [20] and Voronoi [18].

The line-based algorithms extract lines of text from the input document image. Examples of such algorithms are Constrained Text-Line Extraction [19] and Run-Length Smearing (RLSA) [25].

In [8] authors provide review of page segmentation algorithms since 2008. Algorithms included in the study provide physical layout analysis and they work on document images. The study gives very detailed overview of the field, with comprehensive discussion about algorithms' capabilities, limitations, specific parameters, types of layouts they can sufficiently recognize, etc.

Also, authors propose very articulate classification of page segmentation algorithms that allows to identify main techniques implemented in the algorithms. According to them, there are three groups of segmentation algorithms for physical layout labelling. Additionally, every method is classified as top-down or bottom-up. Top-down approaches start generating layout from the document level. Bottom-up methods create layout from the pixel level.

Algorithms from the first group must know in advance the layout type they extract. For instance, some algorithms can recognize only Manhattan layout. Another layout types can be described with set of rules or grammar. These algorithms can be used without a training step.

Algorithms from the second group are designed to be more robust with respect to variations in the document layout. In other words, single algorithm can be used to extract layouts of different types. In general, flexibility of these algorithms is achieved by introducing several parameters. Parameters are tuned during the algorithm training. Usually, a large data set is required.

Algorithms from the third group appeared last. The main approach behind them is to combine several other algorithms in one complex procedure. On the other hand, some algorithms from this group are based on methods from artificial intelligence area (neural networks, for instance) to learn significant parameters and build appropriate model for layout extraction.

In the third group approaches [24,4,5] can be classified. Authors propose approach to combine complementary techniques and apply some voting schema to determine where all techniques agree. In that way, the final output can be better than the individual results.

Similar idea is presented in [6] where authors propose method to improve OCR accuracy by including several OCR engines and performing specific voting schema.

3. Our contribution

Based on the previous discussion it is obvious that page segmentation algorithms differ considerably. So, it is natural to expect that different algorithms generate (significantly) different segmentation of the same document.

On other hand, information extraction algorithms heavily depend on document geometry and positional information. Many of them extract target information based on its position on the document and spatial relations with other elements [9,1,22,11]. In addition, their training consists of learning spatial relations between fields [12].

The main motif for this research arises from challenges in design and implementation of a document understanding system that must be capable to extract target fields from document images processed with different OCR engines. In other words, the aim is to make different page segmentation algorithms and OCR engines compatible in a sense that for the same document image they will produce the same physical layout. With this approach, document understanding systems become independent of pre-processing steps depending on page segmentation algorithm and OCR engine used.

An obvious solution is to maintain different extractors for every OCR engine. This will work only when it is known which OCR engine is used for processing document image and generating corresponding searchable PDF. The pseudo code for such solution is presented below.

```
{
d = current_searchable_PDF
switch d.source
  case FineReader:
    call InformationExtractionForFineReader(
  d)
  case Tesseract:
    call InformationExtractionForTesseract(d
  )
  default:
    call DefaultInformationExtraction(d)
}
```

The previous implementation implies that significant cost must be spent in maintaining extractor models for every OCR engine. For example, size of extractor models can be hundreds of MB. Also, it is usually mandatory to implement *active learning paradigm* meaning that each extractor model must be periodically retrained with documents processed so far. This will increase models size as well as spend significant processor time.

In this study we introduce novel step in classical document understanding use case. Fig. 2 illustrates extension of the classical scenario in a document understanding system with the LAA procedure. The purpose of the LAA step is to make information extraction module independent on specific OCR engine. In this way we can use and maintain only one universal model for information extraction and handle situations where different OCR engines are used for model training and document stream processing.

More precisely, the problem we define in this paper is to furbish physical layout contained in the given searchable PDF format in a way that if any other OCR engine is used to process the given document image the furbished layout will be always the same. Input to the algorithm is searchable PDF representation generated with different OCR engines (each implementing specific page segmentation algorithm). The algorithm does not have any information which OCR engine is used to generate searchable PDF file for its input.

To the best of our knowledge this is the first study discussing such problem. The LAA is based on KMeans clustering algorithm with specific procedures for normalization and initial centroid generation. In this research we concentrate on administrative documents. Our method does not make any assumption about document layout and it is parameter free.

The LAA can remind of voting methods mentioned in the Related work section. These methods are trying to find correct layout by voting between layouts obtained with several complementary approaches. It is possible because all of them process original document image and can involve and combine results of several external systems. In contrary, the method proposed in this study is not able to access the original document image or layout representations of every possible OCR engine to vote between them. It is provided with just one layout generated from the original image by unknown OCR engine and encoded into searchable PDF format.

The LAA method processes available elements in the given searchable PDF and ensures that they will be placed uniquely regardless of which OCR engine is used. Otherwise, all information regarding to elements position and size must be eliminated from extractor knowledge and model. Inevitably, this will degrade its capability and usability (because of lack of the most important attributes for model training).

In the best case the LAA method achieves *precision* above 98% that is comparable to results known from the literature that are between 94.62% and 97.72% for the similar problems. It is not possible to make exact comparison because this problem has not been addressed yet and because of lack of the universal dataset for testing.

To prove LAA method practical usability we proceed the following case study. We extend one commercial document understanding system with LAA module as it is suggested in the Fig. 2. Documents were obtained from one healthcare company. The extractor was trained on samples processed with one commercial engine and tested with samples from the same layout class but processed with other engines (free and commercial). Fields to extract are defined as follows: provider name, provider number, provider address, contact name, phone number, and fax number. The extractor accuracy was increased up to 10% for each target field when LAA is used as pre-processing step.

4. Text line alignment with clustering approach

At the beginning of this section we extend the motivational idea for the LAA.

We processed document image from Fig. 1 with Tesseract and FineReader and obtained corresponding searchable PDF files. After that we used PdfMiner library to parse PDFs and create XML tree representation of elements extracted from the image. Text box elements' spatial information is determined with *bbox* attribute of the following form $bbox = [x, y, width, height]$. Bounding box attribute contains position of the left bottom corner of corresponding text box, its width and height. We displayed each text box.

The results are presented on Fig. 3.

Different OCR engines generate different physical segments of the same document. Additionally, we can notice that two engines implement different measurement scales. For example, FineReader sets page area into virtual bounding box with $width = 594.950$ and $height = 841.900$. But Tesseract defines significantly different page area with $width = 929.250$ and $height = 1314.750$.

Also, there are significant differences in recognized text boxes and their positions. It can be noticed that Tesseract skipped several

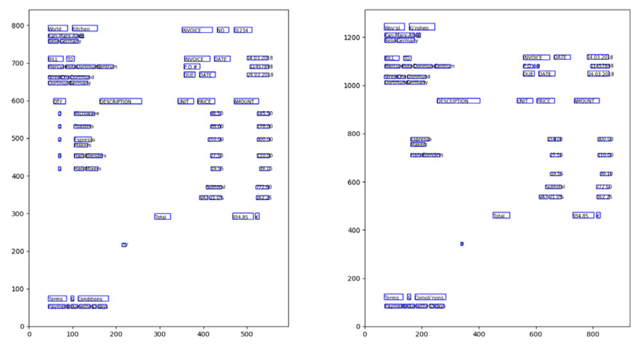


Fig. 3. Text boxes recognized by FineReader (left) and Tesseract (right). Each text box is represented with *bbox* attribute that contains position of the left bottom corner, width and height.

text boxes that are recognized with FineReader. We want to emphasize that approach presented in this study cannot reconstruct elements that OCR engine did not find.

The LAA method processes available text boxes and ensures that they will be placed uniquely regardless of which OCR engine is used.

Firstly, we must define unique measurement scale for transforming bounding box attribute. This measurement scale determines size of virtual document page and it is used for positioning text boxes on the virtual page. Such page consists of lines. The line is uniquely identified with its *y* position on the page. Each text box must be assigned to only one line.

In addition, we need an approach to assign given text box to the appropriate line. The proposed method takes line to be cluster of words. Generally, clusters are defined as groups of elements that are very close to each other and at the same time very distant from elements belonging to another clusters. Consequently, line is set of text boxes or words whose *y* positions on the page are significantly near to each other. Clustering of text boxes eventually produces lines such that the line l_y occupies all text boxes closest to the position *y*. Considering other parts from *bbox* attribute, namely x, *width* and *height* in word representation can degrade expected horizontality of text line because words from different lines can be allocated to the same cluster.

The main steps of the LAA algorithm are presented in the listing Algorithm 1. The algorithm is implemented in Python 3, so pseudocode from the listing can remind of it.

Algorithm 1 LAA algorithm

```
Require d – searchable PDF representation of document
Ensure L – physical layout consisting of text lines
  for all page in d
    //1. parse page with PDFMiner API
    words_xml_tree = PDFMiner.Extract words with
  positions(page)
    //2. normalization
    words_list = []
    for all word in words_xml_tree
      word_norm = {}
      word_norm[X] = minmaxnorm(word.X)
      word_norm[Y] = minmaxnorm(word.Y)
      word_norm[Width] = minmaxnorm(word.Width)
      word_norm[Height] = minmaxnorm(word.
  Height)
      words_list.append(word_norm)
    end for
    //3. Prepare dataset for the clustering method
    dataset = []
```

```
        for all word in words_list
            dataset.append([word["Y"]])
        end for
        //4. Run clustering algorithm, clusters represent lines
        model = KMeans(dataset, init = initial_cetroids(dataset))
        //5. Assign words to lines
        L = []
        for all cluster in model.clusters
            current_line_indexes = model.labels == cluster
            current_line_elements = dataset[current_line_indexes]
            L.append(current_line_elements)
        end for
    end for
```

In the rest of this section we expose one iteration of the proposed algorithm. During the single iteration one page of the document is processed.

The first step consists of text boxes extraction from a current page of searchable PDF file with PDFMiner library. The result is represented in the form of XML tree. Traversal of the tree is performed to generate list of text boxes corresponding to words. Every word, apart from textual content, is extended with its bounding box attribute *bbox*. As we explained earlier, the bounding box is virtual rectangle that borders word region. It is represented with position (X, Y) of the bottom left vertex as well as its width and height. Bottom left corner of the page is $(0, 0)$.

We emphasize that values associated by PDFMiner for position, width and height heavily depends on specific OCR engine that generates searchable PDF file. Apart from other factors, it is due to the measurement scale used.

Information extraction systems heavily depends on document geometry and positional information. Many of them extract target information based on its position on the document and spatial relations with other elements [9,1,22,11]. To achieve desired information extraction independence of document source it is essential to introduce unique measurement scale and units. Otherwise, all information regarding to text box position and size must be eliminated from extractor knowledge and model. Inevitably, this will degrade its capability and usability (because of lack of the most important attributes for model building).

In the second step of the LAA positional information is normalized. This implies transforming values to become part of a smaller or more appropriate range. In other words, normalization step is to transform coordinates X and Y, *width* and *height* for every word to the target interval *[newmin, newmax]*. We tested LAA with several methods for normalization.

The min–max normalization [13] procedure is default transformation in the normalization step. Let us take that A_{min} and A_{max} are the minimum and maximum values of an attribute A. Min–max normalization maps a value v of A to v' in the interval $[A_{newmin}, A_{newmax}]$ by computing the following formula [15]:

$$v' = \frac{v - A_{min}}{A_{max} - A_{min}} \times (A_{newmax} - A_{newmin}) + A_{newmin}. \tag{1}$$

Min–max normalization retains the relations between the source data values [13]. Notice that in our context it will never encounter "out-of-bounds" error. The "out-of-bounds" error occurs if a future input case for normalization falls outside of the original data range for A. Let us explain briefly.

In this step, as it is stated earlier, we must normalize coordinates X and Y, width and height for every word. For all four attributes default target interval is set to $[0, 100]$. In this case the virtual page width is 100 as well as the page height. For all attributes, $A_{min} = 0$ holds. For coordinate X, we set $X_{max} = pagewidth$. The value for source page width is known from the first step (PDFMiner associates this value). For coordinate Y, we set $Y_{max} = pageheight$. The value for source page height is known from the first step (PDFMiner associates this value). For word width, $WIDTH_{max}$ = *page width*. For word height, $HEIGHT_{max}$ = *page height*.

Instead of min–max normalization it is possible to run z-score normalization [13] or normalization by decimal scaling [13].

Z score normalization is based on the mean $\overline{A}$ and the standard deviation σ_A of an attribute A. For a value v of A normalized value v' is given by the following formula:

$$v' = \frac{v - \overline{A}}{\sigma_A}. \tag{2}$$

Normalization by decimal scaling is based on the maximum absolute value of A. Decimal scaling normalization maps a value v of A to v' by computing:

$$v' = \frac{v}{10^j}, \tag{3}$$

where j is the smallest integer such that the following expression holds

$$max(|v^j|) < 1. \tag{4}$$

Eventually, result of the normalization step is a list of JSON objects. Every word is represented as one object in the list. Apart from attributes that are read from the XML tree (created through PDFMiner interface), every word is extended with values representing normalized bounding box region. Attribute *nbbox* is of the following form *nbbox* = *x_norm, y_norm, width_norm, height_norm*.

In the third step, we implemented procedures to convert words' information from JSON to the form acceptable for clustering algorithms and interfaces provided in the *sklearn* library. With clustering we want to arrange words from the list in a way that all words belonging to the same line will form one cluster. Of course, at the same time, words from different lines will be assigned to different clusters. Normalized values are used.

Every word is represented with Y component from the normalized bounding box attribute. Intuitively, it is because text line is dominantly determined with its position on Y axes. The Y component represents normalized position of left bottom vertex. The best experimental results are achieved with this approach. Alternatives that are also implemented and tested represent word with any other vertex or centroid of the corresponding bounding region.

In the fourth step, clustering method is run to generate clusters. One cluster represents one line from a document. We tested LAA with several methods for clustering.

The best experimental results are obtained when KMeans method is called. But initial centroids must be calculated carefully. We propose two procedures. The first procedure generates initial cluster centroids, i.e. their Y coordinates, based on average word height – *awh*. The value *awh* is known from the first step and normalized in the third step. The initial clusters are created as follows. The first cluster is represented with centroid where $Y = 0$. It repre-

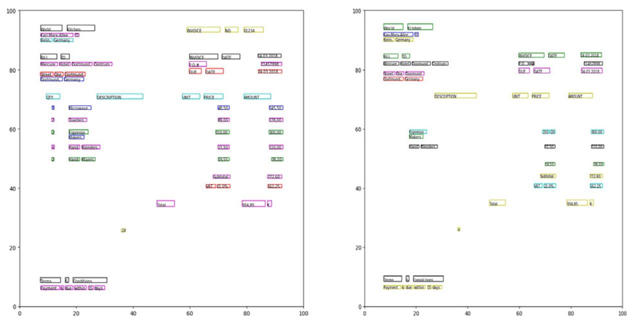


Fig. 4. Clusters: FineReader (left), Tesseract (right). Clusters are generated with default implementation of LAA algorithm: min–max method for normalization, KMeans clustering method with initial centroids generated based on *awh* parameter. Words belonging to the same cluster are depicted with the same color.

sents the very bottom line in a document. The next cluster is represented with centroid where $Y = awh$. Similarly, for the i^{th} cluster corresponding centroid is

$$Y = (i - 1) \times awh, 1 \leqslant i \leqslant 100. \tag{5}$$

The last cluster represents the very top line in a document. It will have a centroid with the highest value for Y coordinate and additionally $Y < 100$ and $Y + awh > 100$ hold. Consequently, KMeans will be started with $k = 100/awh$ centroids.

Alternative procedure for initial centroids generation will calculate $k = 100$ centroids. For the first centroid Y coordinate is 0. It represents cluster for the very bottom line in a document. The last centroid represents cluster for the very top line in a document. Consequently, in this case $Y = 99$. Obviously, for the i^{th} cluster corresponding centroid is

$$Y = i - 1, 1 \leqslant i \leqslant 100. \tag{6}$$

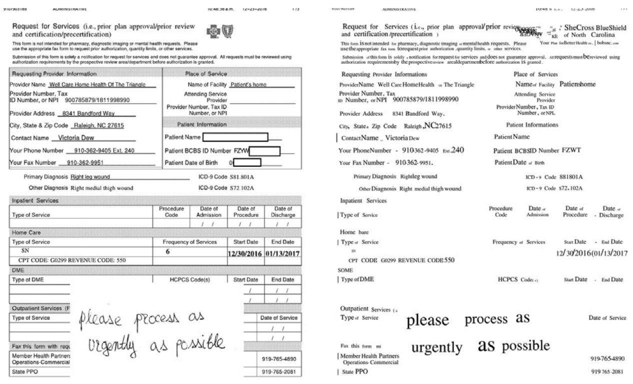
Fig. 4 illustrates clusters of words in document image from Fig. 1 generated with default implementation of LAA algorithm: min–max method for normalization, KMeans clustering method with initial centroids generated based on *awh* parameter. Words belonging to the same cluster are depicted with the same color.

We want to accent that at the first glance the same result can be achieved with the following simple *Line Scan* algorithm that avoids clustering:

```
divide the page into k height cells
  representing lines
assign words to the closest cell based on Y
  coordinate
```

But the previous approach requires post processing. Intensive experiments indicate that in general case adjacent words from the same line can be allocated to different height cells with the previous simple algorithm.

More precisely, let words $w_1, w_2, \ldots, w_m$ originally belongs to the same line. Due to OCR imperfections and errors it is expected that corresponding bounding boxes can significantly differ in Y coordinate. For example, some OCR engines will exaggerate text size and/or white spaces around the text. This can result in a situation where words $w_1, w_2, \ldots, w_m$ are allocated to several consecutive height cells $heightcell_i, heightcell_{i+1}, \ldots, heightcell_{i+k}$. In post processing such cells are usually intersecting and should be merged into one line containing all words $w_1, w_2, \ldots, w_m$.



9107903169 ADMINISTRATIVE 10:46:56 a.m. 12-23-2016 1/3

Request for Services (i.e., prior plan approval/prior review and certification/precertification)

This form is not intended for pharmacy, diagnostic imaging or mental health requests. Please use the appropriate fax form to request prior authorization, quantity limits, or other services.

Submission of this form is solely a notification for request for services and does not guarantee approval. All requests must be reviewed using authorization requirements by the prospective review area/department before authorization is granted.

Requesting Provider Information	Place of Service
Provider Name Well Care Home Health Of The Triangle	Name of Facility Patient's home
Provider Number, Tax ID Number, or NPI 900785879/1811998990	Attending Service Provider
Provider Address 8341 Bandford Way	Provider Number, Tax ID Number, or NPI
City, State & Zip Code Raleigh, NC 27615	Patient Information
Contact Name Victoria Dew	Patient Name [redacted]
Your Phone Number 910-362-9405 Ext. 240	Patient BCBS ID Number FZYW[redacted]
Your Fax Number 910-362-9951	Patient Date of Birth 0[redacted]

Primary Diagnosis Right leg wound ICD-9 Code S81.801A

Other Diagnosis Right medial thigh wound ICD-9 Code S72.102A

Inpatient Services

Type of Service	Procedure Code	Date of Admission	Date of Procedure	Date of Discharge
		/ /	/ /	/ /

Home Care

Type of Service	Frequency of Services	Start Date	End Date
SN	6	12/30/2016	01/13/2017
CPT CODE: G0299 REVENUE CODE: 550			

DME

Type of DME	HCPCS Code(s)	Start Date	End Date
		/	/ /
		/	/ /

Outpatient Services (F

Type of Service		Date of Service
		/ /
		/ /

please process as urgently as possible

Fax this form with requ

Member Health Partners Operations-Commercial		919-765-4890
State PPO		919-765-2081

Fig. 5. Example of complex form (left). Result when one OCR engine is applied (right).

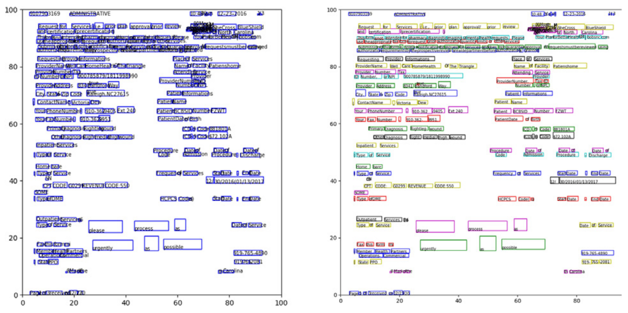


Fig. 6. Result of clustering complex form, OCR result (left), LAA clustering (right). Text boxes from the left picture are input to the LAA that creates lines. Words belonging to the same line are depicted with the same color.

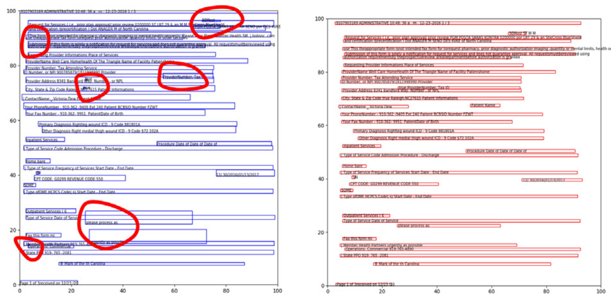


Fig. 7. Result of clustering complex form, simple algorithm (left), LAA (right). Simple algorithm produces intersecting lines, while LAA generated correct clusters (lines).

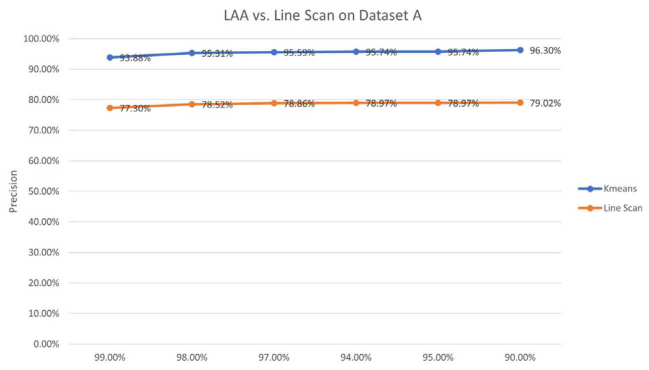


Fig. 8. Dataset A. Method LAA-KMeans achieves maximal *precision* of 96.30%, while maximal *precision* of Line Scan is 79.02%.

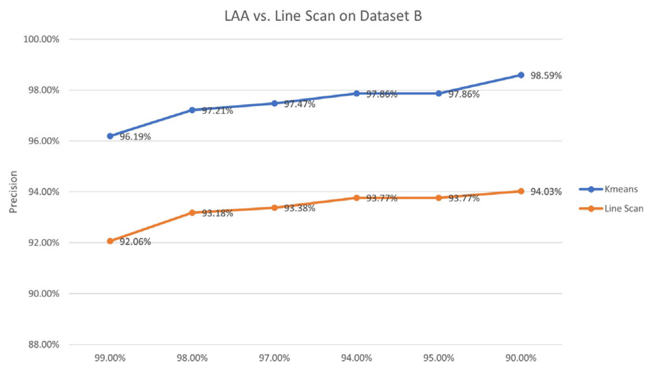


Fig. 9. Dataset B. Method LAA-KMeans achieves maximal *precision* of 98.59%, while maximal *precision* of Line Scan is 94.03%.

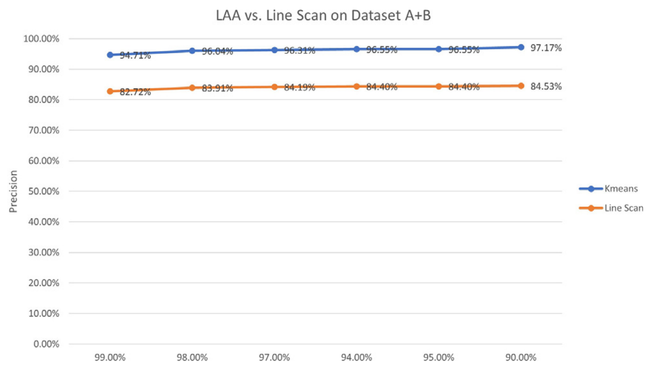


Fig. 10. Dataset A + B. Method LAA-KMeans achieves maximal *precision* of 97.17%, while maximal precision of Line Scan is 84.53%.

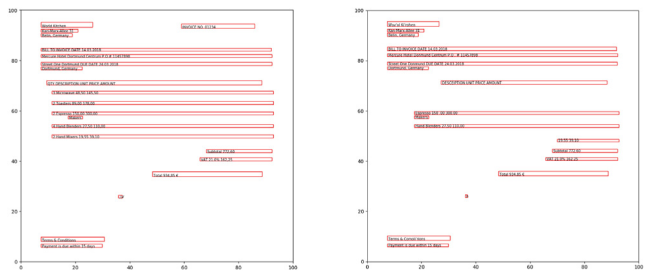


Fig. 11. Text lines: FineReader (left), Tesseract (right). Experiment shows that LAA equally aligns all text boxes recognized by FineReader and Tesseract.

It is necessary to have additional parameter ϵ that determines condition for merging. More precisely, height cells $heightcell_i$ and $heightcell_{i+1}$ with Y coordinates $heightcell_{iY}$ and $heightcell_{i+1Y}$ will be merged in the single line l if holds

$$|heightcell_{iY} - heightcell_{i+1Y}| \leqslant \epsilon. \tag{7}$$

It is impossible to assume universal value for ϵ, especially for overwhelmed forms and when OCR results are of low quality. Example of such form and OCR result is illustrated on Fig. 5. Text boxes from the left picture are input to the LAA that creates lines. Words belonging to the same line are depicted with the same color. This situation is illustrated on Fig. 6. Finally, Fig. 7 shows that it was impossible to correctly distribute text boxes to lines with simple Line Scan algorithm, while LAA generated correct clusters (lines).

In addition to previous discussion, on Fig. 8, Fig. 9 and Fig. 10 we present results of experiments where we compare *precision* of the LAA algorithm and the Line Scan algorithm. From results we can conclude that the LAA algorithm achieves higher precision in all tested cases. Especially significant difference is noticed on the more difficult datasets Dataset A and Dataset A + B. Method LAA-KMeans achieves maximal *precision* of 96.30%, while maximal *precision* of Line Scan is 79.02% on Dataset A (Fig. 8). Method LAA-KMeans achieves maximal *precision* of 97.17%, while maximal precision of Line Scan is 84.53% on Dataset A + B (Fig. 10). Detailed description of experimental protocol and used datasets are part of the *Experimental Protocol and Results* section.

The last step from the LAA algorithm loop generates list of detected lines. One line is represented with one cluster. At this moment we know for every word which cluster it belongs to. The information is included in generated clustering model. Briefly, every word is labelled with identifier of cluster it is assigned to. For example, the i^{th} cluster contains all words with label i. Roughly, this step is implemented as procedure that iterates over original list of words, checks labels assigned with clustering model and distribute equally labelled words to the same line.

Fig. 11 shows that LAA equally aligns all text boxes recognized by both OCR engines (FineReader and Tesseract), as it can yet be presumed from the found clusters (Fig. 4).

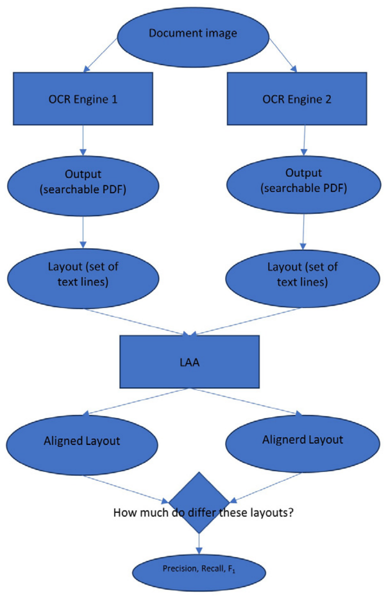


Fig. 12. LAA algorithm is applied on layouts encapsulated in searchable PDFs. Two text lines are considered equal after LAA transformation if corresponding bounding boxes overlap above a given threshold T_α.

5. Experimental protocol and results

In this section we explain experimental protocol for testing LAA module. We performed a series of experiments on the set of 56 document images of very low quality and on the set of 30 document images of higher quality but still less than 100 ppi. In the rest of this section these datasets are referred to as Dataset A and Dataset B, respectively. With FineReader we extracted 1298 text lines from the Dataset A and 1150 text lines from the Dataset B. Entirely, 2448 text lines. Tesseract recognized 1165 text lines from the Dataset A and 1106 text lines from the Dataset B. Altogether, 2271 text lines. Samples are real client documents from several different classes: forms, invoices, air tickets, contracts etc. For comparison, datasets used in several very popular competitions of page segmentation algorithms were of size between 720 and 4034 text lines [7].

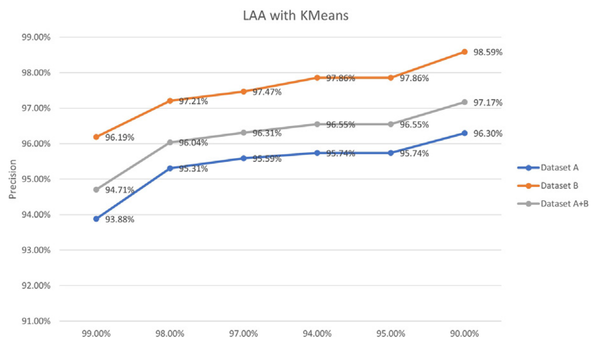


Fig. 13. Method *LAA-KMeans* achieves *precision* above 95.31 yet for $T_\alpha = 98\%$. Maximal *precision* is 98.59%.

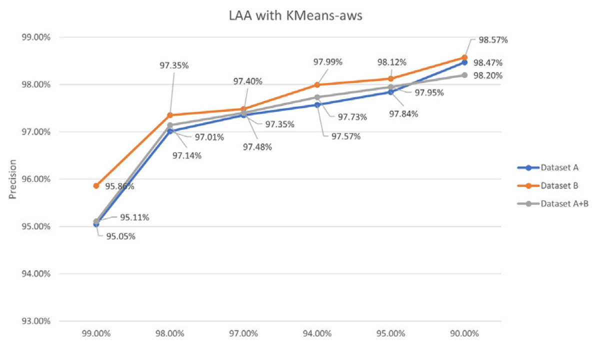


Fig. 14. Method *LAA-KMeans-aws* achieves *precision* above 97.14 yet for $T_\alpha = 98\%$. Maximal *precision* is 98.57%.

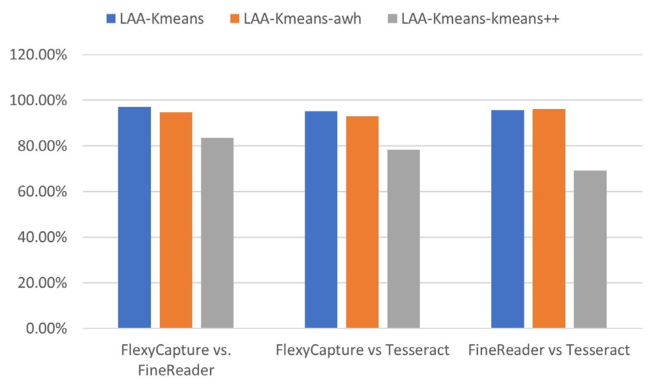


Fig. 15. The default method for initializing KMeans algorithm implemented in *sklearn*, denoted by *kmeans++*, is compared to methods *LAA-KMeans-awh* and *LAA-KMeans*. In all test cases methods *LAA-KMeans-awh* and *LAA-KMeans* are better.

Every image is processed with three OCR engines, namely Tesseract, ABBYY FineReader and ABBYY FlexiCapture.

The performance of the LAA algorithm is measured based on the Line Accuracy Measure usually denoted by FM and introduced in [7]. The FM measure (can be considered as F_1 score) is widely used to estimate line segmentation algorithms. It combines Detection Rate (can be considered as recall) and Recognition Accuracy (can be considered as precision).

Experimental protocol is illustrated on Fig. 12.

We apply LAA algorithm on layouts consisting of text lines encapsulated in searchable PDFs and generated for every pair of available OCR engines. Two text lines are considered equal after LAA transformation if corresponding bounding boxes overlap above a given threshold T_α. In [7] authors suggested to be $90\% \leqslant T_\alpha \leqslant 95\%$.

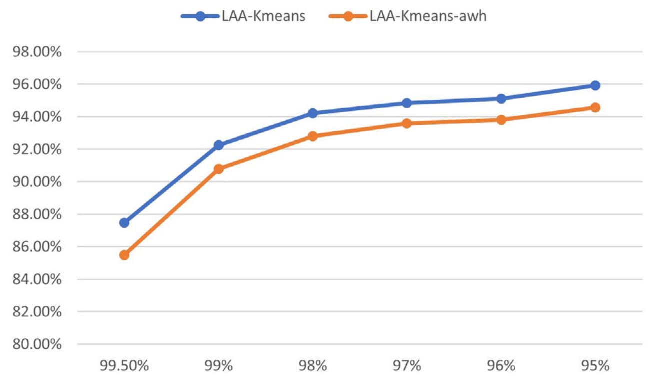


Fig. 16. F_1 with respect to the threshold T_α. LAA performances are better for smaller values of the threshold, because smaller values for T_α will produce greater number of true positive lines - *TP*.

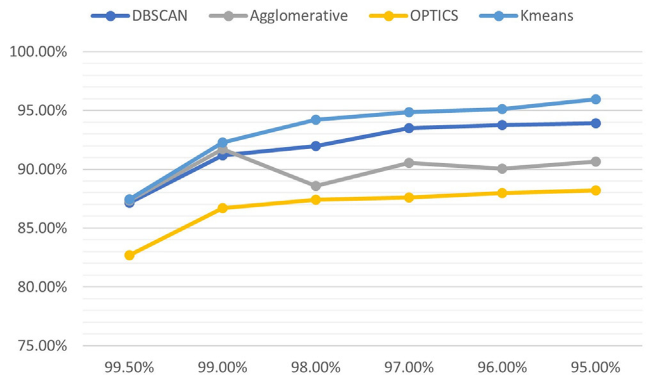


Fig. 17. The only clustering method comparable to the KMeans is DBSCAN. We also tested Agglomerative and OPTICS methods for clustering text boxes into lines.

Let us explain the performance metric used in the experiments. To the best of our knowledge the most similar problem known in the literature is line segmentation, so the FM measure serves us as the groundwork.

Consider two searchable PDFs generated with two different OCR engines from the single document image. They contain two layouts, potentially different. For this experimental protocol we can assume that each layout is represented with a set of text lines. Each line in the layout is represented with surrounding bounding box, i.e. position of the bottom left corner (X, Y), bounding box width and height. We need to estimate how good is LAA transformation

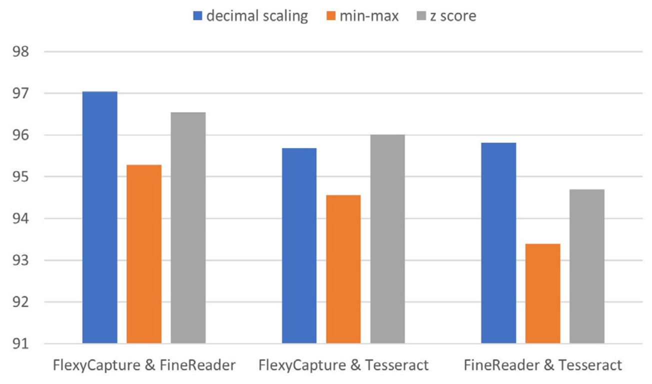


Fig. 18. The experiment shows results of comparing different procedures for normalization. In all test cases, the highest F_1 values are achieved with decimal scaling.

of the two given layouts. In the best case LAA transformation will generate the same set of text lines (i.e. the same layout). In general, LAA transformation can output different layouts.

Consider that the document image d and two OCR engines E_i and E_j are given. The engine E_i processes the document image d and generates searchable PDF file $F_d^{E_i}$ with the corresponding layout $L_d^{E_i}$ represented as set of text lines $L_d^{E_i} = \left\{ l_{1d}^{E_i}, l_{2d}^{E_i}, \ldots, l_{Md}^{E_i} \right\}$. Similarly, the engine E_j processes the document image d and generates searchable PDF file $F_d^{E_j}$ with the corresponding layout $L_d^{E_j}$ represented as set of text lines $L_d^{E_j} = \left\{ l_{1d}^{E_j}, l_{2d}^{E_j}, \ldots, l_{dN}^{E_j} \right\}$.

Layouts $L_d^{E_i}$ and $L_d^{E_j}$ are input to LAA algorithm. In the best case LAA will make the two given layouts equal, i.e. $LAA\left(L_d^{E_i}\right) = LAA\left(L_d^{E_j}\right)$, where $LAA(L)$ denotes layout resulting after LAA algorithm is applied to the given layout L. Two layout are equal if they contain the same text lines. Two text lines are equal if the corresponding bounding boxes overlap above the threshold T_α.

In general, *recall*, *precision* and F_1 score are computed:

$$Recall = \frac{TP}{N} \tag{8}$$

$$Precision = \frac{TP}{M} \tag{9}$$

$$F_1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{10}$$

where TP is the number of consistently aligned ($truepositive$) text lines from both layouts. Notice that $TP = |LAA\left(L_d^{E_i}\right) \cap LAA\left(L_d^{E_j}\right)|$.

LAA achieves the best results when clustering algorithm is KMeans and initial centroids are calculated with the expressions given in (5) and (6). We denote these implementations by *LAA-KMeans* and *LAA-KMeans-awh* respectively. On Fig. 13 and Fig. 14 we present results of series of experiments where we want to estimate if the presented methods can compete with the best solutions known from the literature. The most similar problem known in the literature is related to text line detection from document images [7]. Authors reported results for *precision* measure between 94.62 and 97.72. Method *LAA-KMeans* achieves *precision* between 93.88 and 98.59 (Fig. 13). Method *LAA-KMeans-awh* achieves *precision* between 95.05 and 98.57 (Fig. 14). We change T_α from 90% to 99.5%.

In the rest of this section we present results of experiments where we estimate performances of different implementations of specific steps and procedures in the LAA method.

Methods for initial centroids generation in LAA when clustering approach is KMeans significantly influence performances. It is illustrated on Fig. 15. The default method for initializing KMeans algorithm implemented in *sklearn*, denoted by *kmeans++*, is compared to methods *LAA-KMeans-awh* and *LAA-KMeans*. In all test cases methods *LAA-KMeans-awh* and *LAA-KMeans* are better. We set $T_\alpha = 95\%$. In this experiment we used union of both datasets.

The value of the parameter T_α also influences LAA performances. As it is mentioned earlier, two text lines are equal if their bounding boxes overlap above the threshold T_α. In the following experiment we change T_α from 95% to 99.5%. Results are shown on Fig. 16. Values for T_α are plotted on the X axis. The Y axis gives the F_1 measure. As it expected, algorithm's performances are better for smaller values of the threshold. Authors in [7] also reported that FM measure increases when the threshold T_α decreases. The union of both datasets is used in this experiment.

LAA algorithm with different clustering approaches is tested. The only clustering method comparable to the KMeans is DBSCAN.

The DBSCAN clustering algorithm is density based. It means that clusters are detected as regions of high density separated with sparse regions. Density is expressed as the number of data points in the given area. The algorithm requires two parameters *min_points* and *eps* and consequently defines three types of data points: *core, border* and *outlier*. Core points are samples such that there exists at least *min_samples* within its *eps* neighbourhood. Border points are not core but they belong to *eps* neighbourhood of some core point. Clusters are made from core points that are within *eps* neighbourhood of each other (connected points) plus their border points. Outliers are points that are neither core nor border. In our context points are text boxes or words. Connected words and their border words represent one text line. Outliers does not exist because every single word even in the isolated corner of the document must be extracted as text line itself. Because of that we must set $min\_points = 1$. Also, it is challenging to define appropriate value for the *eps*.

Results are summarized on Fig. 17. Values for T_α are plotted on the X axis. The Y axis gives the F_1 measure. Dataset A is used.

Finally, we present results of experiment where we compare different procedures for normalization. In all test cases, the highest F_1 values (the Y axis) are achieved with decimal scaling as it is shown on Fig. 18. We used union of both datasets in this experiment.

As we emphasize in previous sections, information extractor is heavily dependent on layout information. For example, it is very important for the extractor to have field label and its value in deterministic positions regardless of the document source.

To estimate if LAA method can increase the extractor accuracy we proceed the following experiment. Samples were obtained from one healthcare company. The extractor was trained on samples processed with one commercial engine and tested with samples from the same layout class but processed with other engines (free and commercial). Fields to extract are defined as follows: provider name, provider number, provider address, contact name, phone number, and fax number. The extractor accuracy was increased up to 10% when LAA is used as pre-processing step as it is suggested in the introductory section on Fig. 2.

6. Conclusion

In this paper we formulate the problem to generate equal physical layout for same document image regardless of OCR engine used. Document layout is presented as set of text lines. Also, we propose the LAA method as solution for the problem.

The LAA approach is based on clustering method. The main idea is to introduce unique measurement scale and to group words into clusters, i.e. text lines. Input to the algorithm is searchable PDF file generated by unknown OCR engine. LAA aligns given physical layout in a way that if any other OCR engine is used to process the same document image the resulting layout is always the same.

The best results are achieved when the following parameters and heuristics are applied:

- all values are previously normalized with decimal scaling method
- every bounding box is represented with just Y coordinate
- initial centroids are calculated with the formula (5).

We use performance measures *recall*, *precision* and F_1 score that were also used in the literature for the similar problems. Experiments show that the presented solution is comparable to reported performances of analogous methods.

LAA algorithm can be integrated into general document understanding software as pre-processing procedure to make information extraction module independent on specific OCR engine used in document images processing.

References

[1] Aggarwal CC, Zhai C. Mining text data. Boston: Springer; 2012.
[2] Akram S, Dar MUD, Quyoum A. Document image processing – a review. Int J Comput Appl 2010;10:35–40. doi: https://doi.org/10.5120/1475-1991. URL: http://www.ijcaonline.org/volume10/number5/pxc3871991.pdf.
[3] Baird HS. Background structure in document images. In Series in Machine Perception and Artificial Intelligence. World Scientific. vol. 16; 1994. pp. 17–34.https://doi.org/10.1142/9789812797797_0003..
[4] Boiangiu CA, Boglis P, Simion G, Ioanitescu R. Voting-based layout analysis. J Inf Syst Oper Manage 2014;8:39–47.
[5] Boiangiu CA, Ioanitescu R. Voting-based image segmentation. J Inf Syst Oper Manage 2013;8:211–20.
[6] Boiangiu CA, Ioanitescu R, Dragomir RC. Voting-based ocr system. J Inf Syst Oper Manage 2013;8:211–20.
[7] Diem M, Kleber F, Sablatnig R. Text line detection for heterogeneous documents. IEEE; 2013. p. 743–7. doi: https://doi.org/10.1109/ICDAR.2013.152. URL: http://ieeexplore.ieee.org/document/6628717/.
[8] Eskenazi S, Gomez-Krämer P, Ogier JM. A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recogn 2017;64:1–14. doi: https://doi.org/10.1016/j.patcog.2016.10.023. URL: https://linkinghub.elsevier.com/retrieve/pii/S0031320316303399.
[9] Esser D, Schuster D, Muthmann K, Berger M, Schill A. Automatic indexing of scanned documents: a layout-based approach; 2012. p. 82970H.https://doi.org/10.1117/12.908542..
[10] Everingham M, Eslami SMA, Van Gool L, Williams CKI, Winn J, Zisserman A. The pascal visual object classes challenge: a retrospective. Int J Comput Vis 2015;111:98–136. doi: https://doi.org/10.1007/s11263-014-0733-5.
[11] García-Constantino M, Atkinson K, Bollegala D, Chapman K, Coenen F, Roberts C, Robson K. CLIEL: context-based information extraction from commercial law documents. ACM Press; 2017. pp. 79–87.https://doi.org/10.1145/3086512.3086520..
[12] Ghai D, Jain N. Text extraction from document images – a review. Int J Comput Appl 2013;84:40–8. doi: https://doi.org/10.5120/14559-2661. URL: http://research.ijcaonline.org/volume84/number3/pxc3892661.pdf.
[13] Han J, Kamber M. Data mining: concepts and techniques. Haryana, India; Burlington, MA: Elsevier; 2012.
[14] Karatzas D, Gomez L, Nicolaou A, Rusinol M. The robust reading competition annotation and evaluation platform. IEEE; 2018. p. 61–6. doi: https://doi.org/10.1109/DAS.2018.22. URL: https://ieeexplore.ieee.org/document/8395172/.
[15] Karatzas D, Gomez-Bigorda L, Nicolaou A, Ghosh S, Bagdanov A, Iwamura M, Matas J, Neumann L, Chandrasekhar VR, Lu S, Shafait F, Uchida S, Valveny E. ICDAR 2015 competition on robust reading. IEEE; 2015. pp. 1156–1160. https://doi.org/10.1109/ICDAR.2015.7333942..
[16] Karatzas D, Shafait F, Uchida S, Iwamura M, Bigorda LGi, Mestre SR, Mas J, Mota DF, Almazan JA, de las Heras LP. ICDAR 2013 robust reading competition. IEEE; 2013. pp. 1484–1493.https://doi.org/10.1109/ICDAR.2013.221..
[17] Kaur S, Mann P, Kaur S. Page segmentation using XY cut algorithm in OCR systems - a review. Int J Comput Technol 2007;6:436–40. doi: https://doi.org/10.24297/ijct.v6i3.4465. URL: https://rajpub.com/index.php/ijct/article/view/4465.
[18] Kise K. A computational geometric approach to text-line extraction from binary document images. Proc Document Anal Syst 1998:346–55.
[19] Lopresti D, Hu J, Kashi R, Goos G, Hartmanis J, van Leeuwen J. (Eds.). Document Analysis Systems V. volume 2423 of Lecture Notes in Computer Science. Springer Berlin Heidelberg: Berlin, Heidelberg; 2002.https://doi.org/10.1007/3-540-45869-7..
[20] O'Gorman L. The document spectrum for page layout analysis. IEEE Trans Pattern Anal Mach Intell 1993;15:1162–73. doi: https://doi.org/10.1109/34.244677. URL: http://ieeexplore.ieee.org/document/244677/.
[21] Poibeau T, Saggion H, Piskorski J, Yangarber R. (Eds.). Multi-source, multilingual information extraction and summarization. theory and applications of natural language processing. Springer Berlin Heidelberg: Berlin, Heidelberg; 2013.https://doi.org/10.1007/978-3-642-28569-1..
[22] Schuster D, Muthmann K, Esser D, Schill A, Berger M, Weidling C, Aliyev K, Hofmeier A. Intellix – end-user trained information extraction for document archiving. IEEE; 2013. pp. 101–105. URL: http://ieeexplore.ieee.org/document/6628593/.https://doi.org/10.1109/ICDAR.2013.28..
[23] Shafait F. (Ed.). Geometric Layout Analysis of Scanned Documents. dissertation, Technical University of Kaiserslautern, Technical University of Kaiserslautern; 2008. URL: https://kluedo.ub.uni-kl.de/frontdoor/deliver/index/docId/2008/file/shafait-dissertation.pdf..
[24] Stamatopoulos N, Gatos B, Perantonis SJ. A method for combining complementary techniques for document image segmentation. Pattern Recogn 2009;42:3158–68. doi: https://doi.org/10.1016/j.patcog.2008.10.020. URL: https://linkinghub.elsevier.com/retrieve/pii/S003132030800441X.
[25] Wong KY, Casey RG, Wahl FM. Document analysis system. IBM J Res Develop 1982;26:647–56. doi: https://doi.org/10.1147/rd.266.0647. URL: http://ieeexplore.ieee.org/document/5390486/.