IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology
Vol. 3, Issue 5, May 2016

Document Image Analysis Using Imagemagick and Tesseract-ocr

Prof. Smitha M L[1], Dr. Antony P J[1], Sachin D N[1]

KVG College of Engineering, Sullia, D.K, Karnataka, India[1]

Abstract: Document image analysis is the field of converting paper documents into an editable electronic representation by performing optical character recognition (OCR). In recent years, there has been a tremendous amount of progress in the development of open source OCR systems. The tesseract-ocr engine, as was the HP Research Prototype in the UNLV Fourth Annual Test of OCR Accuracy, is described in a comprehensive overview. Emphasis is placed on aspects that are novel or at least unusual in an OCR engine, including in particular the line finding, features/classification methods, and the adaptive classifier. OCRopus is one of the leading open source document analysis systems using tesseract-ocr with a modular and pluggable architecture. Imagemagick is an open source image processing tool. This paper presents an overview of different steps involved in a document image analysis system and illustrates them with examples from Combination of imagemagick and OCRopus.

Keywords: Document Image Analysis, Imagemagick, tesseract-ocr, open source OCR, Free Software.

I. INTRODUCTION

Paper documents like books, handwritten manuscripts, magazines, newspapers, etc. have traditionally been used as the main source for acquiring, disseminating, and preserving knowledge. The advent of personal computers has given birth to another class of documents called electronic documents. An electronic document is a representation of a document using data structures that can be understood by computers. Typical examples of electronic documents are PDF, Word, XML, E-mails, Web pages, etc.

Electronic documents offer several advantages over traditional paper documents like easier editing, retrieval, indexing, and sharing since document contents can be accessed electronically. Therefore, most documents are created today by electronic means [1]. An electronic document can be converted into a paper document by means of a printing device. Converting a paper document into electronic form, on the other hand, needs a way to transform the document into data structures that can be understood by computers. A scanning device can be used to obtain a digital image of a paper document. The transformation of a scanned document image into a structured electronic representation is a complex artificial intelligence task and is the focus of research in the field of document analysis and recognition. A document imagemay contain different types of contents like text, graphics, half-tones, etc. The goal of Optical character recognition (OCR) is to extract text from a document image. This is achieved in three steps. The first step locates text-lines in the image and identifies their reading order. This step is called geometric layout analysis. In the second step, text-lines identified by the layout analysis step are fed to a character recognition engine which converts them into an appropriate format (ASCII, UTF-8, . . .). Finally, an appropriate language model is usually applied to correct for OCR errors.

Owing to the central role of optical character recognition in digitizing paper documents, several commercial and open source systems are available for OCR. The most notable commercial systems are ABBYY FineReader [2] and Nuance Omnipage [3]. An overview of the leading open source OCR systems is given in Section II. Performing different steps of OCR using the OCRopus open source OCR system will be shown in Section III followed by a summary in Section IV.

II. OPEN SOURCE OCR SYSTEM

Efforts in developing an open source OCR system started in late 90's. GOCR [4] was among the first open source character recognition engines. Other engines e.g. Clara OCR [5], and Ocrad [6] followed in the last decade. However, the performance and capabilities of these engines are very limited as compared to commercial OCR software. The launch of Google Print project (now called Google Book Search) stimulated a lot of interest in open source OCR systems. HP labs open-sourced their OCR engine called Tesseract in 2005. This followed by the development of a new high-performance OCR system at DFKI in 2007 based on funding from Google Inc. One year later, Cognitive Technologies released the kernal of its Cuneiform OCR system as opensource. These developments in the open source OCR systems over the last few years have made them competitive to commerical OCR systems and hence several research and commercial applications have started using them. A brief history of each of the three leading open source OCR systems is given in the following with a more detailed description of OCRopus OCR system.

A. Tesseract-OCR

Tesseract [7], [8] is an open source OCR engine that was developed at HP between 1985 and 1995. In 1995, it was one of the top three OCR engines at the OCR accuracy contest organized by University of Nevada in Las Vegas

(UNLV) [9]. Unfortunately, the project was not developed further and in 2005, HP released the code to UNLV's Information Science Research Institute (ISRI), an academic center doing ongoing research into OCR, as open source for the benefit of the community. The code was then picked up by Google for further development and is still maintained and extended at Google.

B. Cuneiform /Open OCR

Cuneiform [10], developed by Cognitive Technologies in the 90's was one of the first OCR packages to include adaptive character recognition. It remained a competitor of ABBYY FineReader, but finally lost its market share. After several years with no development, its OCR engine was released as open source in 2008 under the name of OpenOCR [11].

C. OCRopus

OCRopus [12], [13] is a state-of-the-art document analysis and OCR system, featuring pluggable layout analysis, character recognition, statistical language modeling, and multilingual capabilities. The OCRopus engine is based on two research projects: a high-performance handwriting recognizer developed in the mid-90's and deployed by the US Census bureau, and novel high-performance layout analysis methods. Among leading open source OCR systems, OCRopus is unique in the sense that its design and development was done from scratch focusing on a modular and pluggable architecture. Besides, the interfaces have been designed to be able to handle any script and language. Since we will be seeing examples of different document processing steps using OCRopus later in this paper, more detail about its architecture and interfaces is given here.

1) Color Coded Segmentation Representation: The purpose of using color-coded segmentation representation [15] is not only to simplify the interface by avoiding to deal with complicated data structures representing boundaries of segmented regions, but also to give a pixel-accurate representation of segmentation. The segmentation follows a particular color coding convention1 to represent different levels of segmentation results. For instance, for representing the physical layout of a text document, the red channel encodes the column number of the pixel, the green channel represents the paragraph number of the pixel counted within its column, and the blue channel represents the line number within its paragraph. Special R,G,B values are assigned to other page elements like half-tones, layout separators etc. An image schematic diagram of a document layout and structure analysis system. Figure 1.

2) hOCR Output Format: The hOCR format [16] targets the major writing systems and languages of the world, and defines markup to represent different typographic and linguistic phenomena across a wide variety of languages and scripts. The format represents the output of OCR as an HTML document and therefore can represent these phenomena with already well-defined, widely understood markup. Additional tags are added to embed the OCR engine specific information into the HTML document. One of the key advantages of the hOCR format over other OCR formats is that it can reuse the expertise that has gone into the development of textual representations for HTML. Generally speaking, all common style-, font-, script-, language-, and typesetting-specific phenomena (hyphenation, spacing, ruby, kashida, etc.) are to be represented using their HTML or Unicode representations.

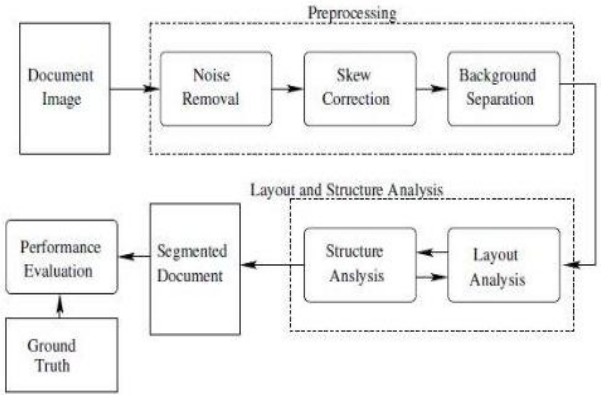

Fig 1. Schematic diagram of a document layout and structure analysis system.

III. IMAGE PRE PROCESSING

The preprocessing stage in document understanding primarily involves the following processes: i) removal of noise and other artifacts that are introduced during the scanning phase and during the production of the document, ii) separation of background regions of the document from the foreground, and iii) correction of skew that is introduced in the document image during scanning/acquisition. We will look into each of these problems and present commonly used techniques in this section. Some of these preprocessing algorithms could also be implemented as part of other modules during layout and structure analysis. However, pre-processing algorithms that are more specific to other modules of document understanding, such as character recognition are not studied here. Such methods might include binarization of a gray-level image and scaling of characters to a standard size

A. Binarization

Is the process that converts a given input grayscale or color document image into a bi-level representation [17]. Since scanners produce a grayscale image by default, it is usually the first step in any document processing pipeline. Imagemagick tool perform this function in proposed model .

B. Noise removal

Noise Removal Noise is a common problem in most of the image understanding problems. These involve white noise that is introduced by interferences in the sensors and amplification circuits of the digitization mechanism (scanner, camera). The common sources of noise include white noise, salt and pepper noise, quantization artifacts, etc. These are well known noise sources that are compensated for by using imagmagick framework

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 3, Issue 5, May 2016

C. Foreground Detection
One of the important problems in document structure understanding is that of separating the foreground from the background image. The problem is relatively simple in case of many documents that have a white or plain background. Even in such documents, determining the exact pixels that belong to the foreground is a challenging problem due to sampling and quantization of slant edges and lines.

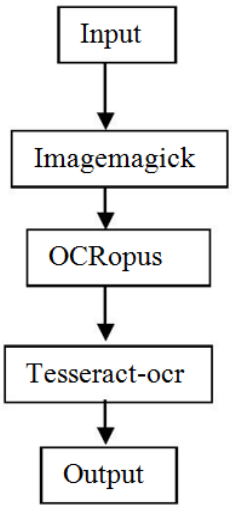

Fig2. Proposed method for document image analysis

D. Skew Correction
Skew is introduced in a document image when a document is scanned or imaged at an angle with respect to the reference axes. The problem of skew correction plays an important role in the effectiveness of many document analysis algorithms, such as text line estimation, region boundary detection, etc. this problem deal with Imagemagick and output of imagemagick is a input of OCRopus as shown in the figure 2.

IV. TEXT RECOGNITION USING OCROPUS

A typical OCR system consists of three key components:
1) Geometric layout analysis identifies the location of text-lines in the scanned documents.
2) Text-line recognition classifies the characters in the text-lines into letters of a pre-defined alphabet.
3) Language modeling attempts to correct text-line recognition output by using language specific information.

A. Geometric Layout Analysis
Geometric layout analysis of a document image typically involves different processes. The exact order in which these processes are applied varies from one algorithm to another. Also, some algorithms might skip one or more of these processes or apply them in a hybrid way. However, most of the layout analysis systems use these processes in some form. Therefore, a brief outline of these processes is given here.

• **Text/Image segmentation:** Text/Image segmentation is a process take a input from imagemagick as shown in the figure 2 and process that classifies page regions into one of a set of predefined classes (e.g. text, image, graphics, . . .) [21], [22]. Methods for separating text and non-text regions in an image implement the ITextImageClassification interface.

• **Layout Analysis:** Layout Analysis is a process that idenitifies text-lines in a document image while respecting the columnar structure of the document and extracts them in an appropriate reading order [23], [24]. In OCRopus, layout analysis algorithms implement the ISegmentPage interface.
Note that for each of the above mentioned steps, implementations of several state-of-the-art algorithms are available in OCRopus. Each method provides a constructor of the form make_...(). Hence replacing one component by another is simply achieved by calling the constructor of the new component. For instance, to use Sauvola [25], [26] binarization algorithm instead of Otsu, we just need to replace the make_BinarizeByOtsu() argument with make_BinarizeBySauvola() in Line 19 of Figure 4. Note that the interface class pointers are held in an autodel object, which is a smart pointer class that automatically deletes the objects its pointing to as soon as it goes out of scope.

B. Text Line Reorganization
Textline recognition is the process of classifying the characters in the text-line image obtained as a result of layout analysis. It proceeds into two steps. The first step is to segment the line image into isolated character images. In the second step, the isolated character images are fed to a trained classifier for recognition. The segmentation of a textline into characters is achieved by algorithms implementing ISegmentLine interface, whereas recognition of these components individually is performed by classes inheriting from ICharacterClassifier interface. However, this approach has certain limitations when applied to real world documents scanned under a wide variety of conditions. Due to scanning artifacts, characters might end up broken or merged with other characters. In such cases accurately spotting characters in a text-line is not possible. Secondly, when isolated characters are fed to the character classifier without the knowledge of their relative size w.r.t. other characters in the line and their vertical position w.r.t. the baseline; distinctions between many uppercase and lowercase letters are not possible (for instance “O” and “o”, “S” and “s”, “P” and “p” etc.).

To handle these cases a more sophisticated text-line recognition technique is required. OCRopus provides a generic interface for text line recognition called IRecognizeLine. This interface provides support for storing segmentation hypothesis of a text-line. These segmenatation
hypotheses are represented as a weighted finite state transducer (FST) in which different segmentation possibilities are encoded as different paths in the transducer. Such a graph is termed as “segmentation graph”. The segmentation graph is implemented as an instance of the IGenericFst interface. The classifier tries to recognize each of the segmentation hypotheses and

DOI 10.17148/IARJSET.2016.3523

IARJSET
ISSN (Online) 2393-8021
ISSN (Print) 2394-1588
International Advanced Research Journal in Science, Engineering and Technology
Vol. 3, Issue 5, May 2016

associates a cost with it. The most likely recognition result is then obtained by the lowest cost path through the segmentation graph.

Although well-trained text-line recognizers tend to be very accurate in recognizing text, there are some ambiguities that cannot be resolved with high confidence without further linguistic analysis. One such example is the distinction between "l" (lower case letter L), "I" (upper case letter I), and "1" (digit). In some fonts like Sans Serif, upper case I and lower case L are represented with the same ligature making it impossible for the text-line recognizer to distinguish between the two. Therefore language modeling is used to post-process the output of text-line recognition. All documents have been processed with Tesseract (version 3.02), a state-of-the-art open source document image analysis system. Tesseract provides an API to analyse an image and access the results. Layout and text content were exported to the PDF, XML, HTML etc.original rectangular outlines as produced by Tesseract (Fig. 4)

C. Statistical Language Model

The purpose of statistical language modeling is to provide linguistic information to the OCR system for better recognition results. The language model might consist of a simple dictionary based lookup. Alternatively, it may provide statistical information about the frequency of occurrence of different characters/words (uni-gram language model). A even more sophisticated language model might provide information about the frequency of occurrence of a sequence of letters/words (n-gram language model). All of these language models can also be conveniently represented as a weighted finite state transducer. Hence language models in OCRopus implement the IGenericFst interface. This representation has the advantage that applying a language model to the output of text-line recognition result (also an FST) is a simple "compose" operation of two FSTs.

V. PERFORMANCE EVOLUTION

OCRopus is an on-going, active open-source project and has just (October 2007) had its alpha release. OCRopus may already be useful in some applications: according to our benchmarks (Figure 3), it is currently the best available open source OCR system for English, when using the combination of imagemagick with the Tesseract text line recognizer.

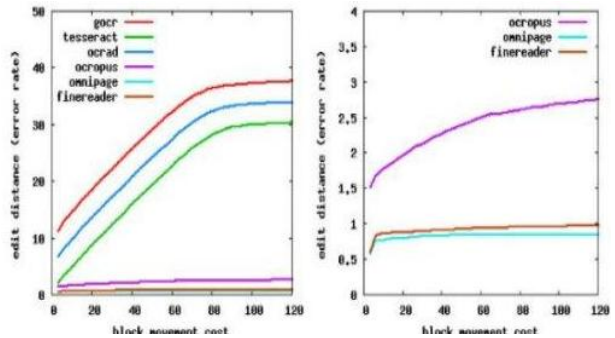

Figure 3. Evaluation of the OCR and layout analysis .

Figure 4 Example result of layout analysis by Tesseract 3.02.

Performance of the OCRopus system on scanned pages. The performance measure is string edit distance with unit costs for single character errors and parameterized costs for block moves. Performance near the left end of the graph indicates character-level accuracy, while performance towards the right indicates layout analysis errors. The OCRopus version is from February 2007. Systems compared are the open source gOCR and OCRad systems, and the commercial Omnipage and Abbyy Finereader systems. Tesseract is evaluated once as a stand-alone system, and once as a text line recognizer within the OCRopus system.

Overall, our hope is that OCRopus will become a widely adopted standard for research in OCR, as well as the basis for many open source and commercial applications, in areas ranging from digital libraries to affordable reading machines for the visually impaired. Potential contributors can find out more about joining the project at www.ocropus.org; we are particularly interested in new text line recognizers for a variety of scripts and language modeling tools for different languages. The source code for the system can be downloaded at the same address.

VI. CONCLUSION

In this paper, we described the Document Image Analysis process step by step and Existing Open Source OCR Systems. Proposed new method implements the combination of Imagemagick (image preprocessing framework) and tessercat-ocr for better document layout analysis compared to existing tools. Apparently, approximate results can done by improved training tools, extensive testing and overall speed and error rate improvements, and the ability to create a usable version of OCRopus with no external library dependencies.

REFERENCES

[1] A. Holmes. Publishing trends and practices in the scientific community. Canadian Journal of Communication, 29:359–368, 2004.
[2] http://finereader.abbyy.com/.
[3] http://www.nuance.com/imaging/products/omnipage.asp.
[4] http://jocr.sourceforge.net/.
[5] http://freshmeat.net/projects/claraocr/.
[6] http://www.gnu.org/software/ocrad/.
[7] R. Smith. An overview of the Tesseract OCR engine. In Proc. 9th Int. Conf. on Document Analysis and Recognition, pages 629–633, Curitiba, Brazil, Sep. 2007.

[8] http://code.google.com/p/tesseract-ocr/.
[9] S. V. Rice, F. R. Jenkins, and T. A. Nartker. The fourth annual test of OCR accuracy. Technical report, Information Science Research Institute, University of Nevada, Las Vegas, 1995.
[10] http://www.cuneiform.ru/eng/.
[11] http://en.openocr.org/.
[12] T. M. Breuel. The OCRopus open source OCR system. In Proc. SPIE Document Recognition and Retrieval XV, pages 0F1–0F15, San Jose, CA, USA, Jan. 2008.
[13] http://code.google.com/p/ocropus/.
[14] http://code.google.com/p/iulib/.
[15] F. Shafait, D. Keysers, and T. M. Breuel. Pixel-accurate representation and evaluation of page segmentation in document images. In 18th Int. Conf. on Pattern Recognition, pages 872–875, Hong Kong, China, Aug. 2006.
[16] T. M. Breuel. The hOCR microformat for OCR workflow and results. In Proc. Int. Conf. on Document Analysis and Recognition, pages 1063– 1067, Curitiba, Brazil, Sep. 2007.
[17] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imag-ing, 13(1):146–165, 2004.
[18] N. Otsu. A threshold selection method from gray-level histograms. IEEE Trans. Systems, Man, and Cybernetics, 9(1):62–66, 1979.
[19] F. Shafait, J. van Beusekom, D. Keysers, and T. M. Breuel. Document cleanup using page frame detection. Int. Jour. on Document Analysis and Recognition, 11(2):81–96, 2008.
[20] D. S. Bloomberg, G. E. Kopec, and L. Dasari. Measuring document image skew and orientation. In Proc. SPIE Document Recognition II, pages 302–316, San Jose, CA, USA, Feb. 1995.
[21] Y. Wang, I. Phillips, and R. Haralick. Document zone content classification and its performance evaluation. Pattern Recognition, 39(1):57–73, 2006.
[22] D. Keysers, F. Shafait, and T. M. Breuel. Document image zone classification - a simple high-performance approach. In 2nd Int. Conf. on Computer Vision Theory and Applications, pages 44–51, Barcelona, Spain, Mar. 2007.
[23] T. M. Breuel. Two geometric algorithms for layout analysis. In Proc. Document Analysis Systems, volume 2423 of Lecture Notes in Computer Science, pages 188–199, Princeton, NY, USA, Aug. 2002.
[24] F. Shafait, D. Keysers, and T. M. Breuel. Performance evaluation and benchmarking of six page segmentation algorithms. IEEE Trans. on Pattern Analysis and Machine Intelligence, 30(6):941–954, 2008.
[25] J. Sauvola and M. Pietikainen. Adaptive document image binarization. Pattern Recognition, 33(2):225–236, 2000.
[26] F. Shafait, D. Keysers, and T.M. Breuel. Efficient implementation of local adaptive thresholding techniques using integral images. Proc, of SPIE Electronic Imaging: Document Recognition and Retrieval, 6815:81510–81510, 2008.