Training TESSERACT Tool for Amazigh OCR

Chapter · May 2015

CITATIONS
2

READS
5,340

3 authors:

Khadija El Gajoui
Mohammed V University of Rabat
7 PUBLICATIONS **14** CITATIONS
SEE PROFILE

Fadoua Ataa Allah
Institut Royal de la Culture Amazighe
63 PUBLICATIONS **202** CITATIONS
SEE PROFILE

Mohammed Oumsis
High School of Technology-Salé, Mohammed V-Agdal University, Rabat, Morocco
57 PUBLICATIONS **281** CITATIONS
SEE PROFILE

Training TESSERACT Tool for Amazigh OCR

KHADIJA EL GAJOUI[1], FADOUA ATAA ALLAH[2], MOHAMMED OUMSIS[3]

[1]Laboratory of research in Informatics and Telecommunications, Faculty of Sciences – Rabat, Mohammed V University, Rabat, MOROCCO
[2]CEISIC, The Royal Institute of Amazigh Culture, Rabat, MOROCCO
[3]Department of Computer Science, School of Technology-Sale, Mohammed V University, Sale, MOROCCO
khadija.gajoui@gmail.com, ataaallah@ircam.ma, oumsis@yahoo.com

Abstract: - The Optical Character Recognition is the operation of converting a text image into an editable text file. Several tools have been developed as OCR systems. Techniques used in each system vary from one system to another, therefore the accuracy changes. In this paper, we present an example of available OCR tools, and we train TESSERACT tool on the Amazigh language transcribed in Latin characters.

Key-Words : OCR; Amazigh; Tesseract; Training.

1. Introduction

Over the last five decades, machine reading has grown from a dream to reality. Optical character recognition has become one of the most successful applications of technology. Many systems for performing OCR exist for a variety of applications, although the machines are still not able to compete with human reading capabilities.

The Amazigh language is spoken by a significant part of the population in North Africa. It became official in Morocco since 2011. Yet few studies on OCR systems have been interested in that language either written in Tifinagh alphabet or transcribed in Arabic or Latin letters.
With the development experienced by research on optical character recognition, field of research in pattern recognition, artificial intelligence and computer vision, various tools have been designed to achieve a conversion from text image to editable text with a quite high recognition rate [1]. The tools dedicated to OCR are either open source or paid according to their license.

In the remaining of this paper, we define, in Section 2, the OCR system architecture and we present different approaches developed for each modules of the system. In Section 3, we introduce the Amazigh language writing. In Section 4, we present examples of OCR tools. In Section 5, we list the training steps of Tesseract tool for the Amazigh language. Then, we show, in section 6, the evaluation of the system tested on a set of documents extracted from different books. Finally, in Section 7, we draw conclusions and suggest further related research.

2. Optical character recognition systems

An OCR system is a system that takes a text image as input and applies certain treatments through modules making up the system in order to output editable file with the same text [2][3].
The architecture of an OCR system varies from one system to another as needed.
Optical Character Recognition systems are usually composed of the following phases [4]:

- Preprocessing phase: prepares the sensor data to the next phase. It is a set of treatments allowing image quality increasing.
- Segmentation phase: delimits document elements (line, word, character ...). By applying good segmentation techniques, we can increase the performance of OCR systems.
- Feature extraction phase: defines features characterizing the delimited elements of a document.
- Feature extraction is one of the most important steps in developing a classification system. This step describes the various features characterizing the delimited elements of a document.
- Classification phase: recognizes and identifies each element. It is performed based on the extracted features.
- Post-processing phase: it is an optional phase. It may be automatic or manual.

Several approaches and techniques have been developed for each module [1].

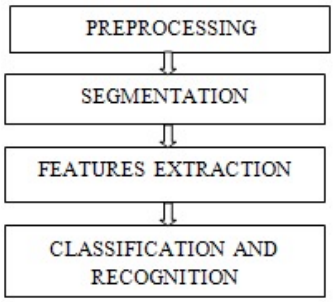

Fig 1. Steps of OCR Systems

3. Amazigh writing

The Amazigh language, or Tamazight, is present today in a dozen of countries across the Maghreb-Sahel-Sahara: Morocco, Algeria, Tunisia, Libya, Egypt, Niger, Mali, Burkina Faso and Mauritania. But Algeria and Morocco are by far the two countries with the largest Amazigh population.
Since antiquity, Amazigh people have developed their own writing system [5]. But when it comes to write consistent documents, the Amazigh has used language and / or script of dominant peoples in contact: Punic, Latin or Arabic.
To transcribe Amazigh language, in Morocco, three writing systems are used [6]:

- Tifinagh is the authentic alphabet, attested in Libyan inscriptions since antiquity, and the official script in Morocco since 2003.
- Arabic alphabet used since the Arab arrival on the 6th century.
- Latin used since the end of the 19th century by colonial scholars, and later by national researchers.

In this work, we focus on Amazigh language transcribed in Latin.

After exploring a set of Amazigh documents transcribed in Latin, such as “CHOICE OF BERBER TALK VERSION OF SOUTHWEST MOROCCAN” by Arsène Roux [5] “MOTS ET CHOSES BERBERES” by Emile Laoust [7] and “THE ARGAN TREE AND ITS TASHELHIYT BERBER LEXICON” by Harry Stroomer [8], we found that the Latin characters used in the transcription are represented in Latin, Extended-A Latin and Extended Additional Latin encoding blocks.
The figure and table below show respectively an example of text written in Amazigh language transcribed in Latin and an example of characters used in this transcription. These characters are composed of Latin alphabet and diacritics that represent a set of marks accompanying a letter or grapheme. Diacritics can be placed above (superscript diacritic), below (subscribed diacritic) or after (adscript diacritic).

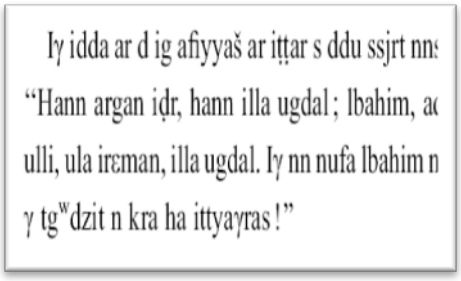

Fig 2. An example of text excerpt from the book ” THE ARGAN TREE AND ITS TASHELHIYT BERBER LEXICON”

Ā	ā	Ă	ă	Aᶜ	aᶜ	Bʷ	bʷ	Bᶜ	bᶜ
Ḅ	ḅ	Ḍ	ḍ	Dᶜ	dᶜ	Ĕ	ĕ	Fʷ	fʷ
Ġ	ġ	Gʷ	gʷ	Ḡ	ḡ	Ḥ	ḥ	Ḫ	ḫ
Kʷ	Kʷ	Lᶜ	Lᶜ	Ḷ	ḷ	Mʷ	mʷ	Ŏ	ŏ
Ṛ	ṛ	Ṡ	ṡ	Tᶜ	tᶜ	Ṭ	ṭ	Ů	ů
Ŭ	ŭ	Ẓ	ẓ						

Table 1. An example of characters used in Amazigh transcription in Latin

4. OCR tools

Automatic document reading techniques have evolved and matured over the past decade. Thus, Test stands are commonly performed on the latest versions of software packages showing the new opportunities and accurate assessment of their recognition ability in terms of confidence levels, accuracy and speed of execution, by paper type and typography used.

The tools available for OCR systems are either commercial or open source. In both cases the tool is based on the OCR’s architecture composed of 3 phases: preprocessing, features extraction and classification. Since the classification is the most determined phase in the system, several approaches have been developed, used and tested in various tools.

4.1. Commercial tools

Commercial systems were developed primarily as university projects, shareware or free tools and converted to high quality sophisticated products meeting the high expectations of today’s OCR market.

Several commercial systems come with a rich set of image processing utilities capable of transforming the input images to the most appropriate format where their ability to work on different types of image and to adapt to various spatial resolutions and pixel depths. The most popular commercial tool is FineReader [9].

This kind of Tools does not allow access to the system code in order to modify it and improve its performance which is very important for research. Hence, the advantage of using open source tools.

4.2. Open source tools

"Open source" refers to software in which the source code is available to the general public, and this is usually a collaborative effort in which programmers improve all the source code and share the changes within the community as well as other members can contribute.

The ability to study how the program works helps in adapting it to the needs. Another important advantage is that any potential improvements can be discussed with developers and added by any interested parties. Thus, the work done has a chance to be reused and extended by anybody.
Below we give an example of such popular open source systems which are OCRopus [10] and Tesseract [11].

OCROPUS

OCRopus is a free document analysis and optical character recognition (OCR) system, released under the Apache License and currently developed under the lead of Thomas Breuel from the German Research Centre for Artificial Intelligence. It is sponsored by Google.

OCRopus is becoming a powerful tool for optical character recognition, capable of analyzing a complex layout (containing columns and boxes…). It does not reconstitute the page layout after processing, but performs the recognition in a logical order after analyzing the layout.

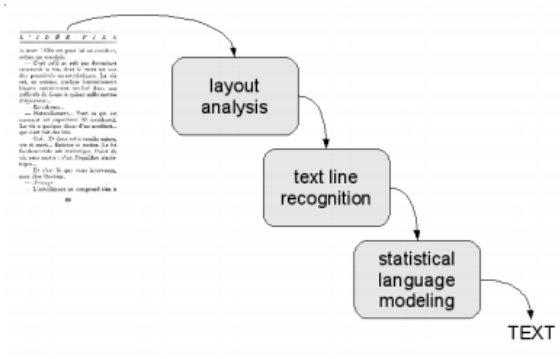

Fig 3. Architecture of the OCRopus system

The overall architecture of OCROPUS consists of three major components [12]:

- Layout analysis: OCRopus contains two modules responsible of the layout analysis: a simple text-image segmentation system to separate different regions and RAST-Based Layout Analysis for the Column finding, the text line modeling and the reading order determination.
- Text line recognition: separates text line of images into a collection of characters. Then, performs the character recognition based on a hypothesis graph.
  OCRopus is based on a statistical approach using multi-layer perceptrons (MLPs) for character recognition.
- Statistical language modeling: integrates alternative recognition hypotheses with prior knowledge about language, vocabulary, grammar, and the document domain.

TESSERACT

Tesseract is an open source optical character recognition engine for various operating systems. It was originally developed at HP between 1984 and 1994 [13] [14]. It was modified and improved in 1995 with greater accuracy. In late 2005, HP released Tesseract for open source. Now it is developed and maintained by Google. It uses a statistical approach based on the polygonal approximation and the calculation of distance between extracted features [13].

Tesseract is considered as one of the most accurate free software OCR engines currently available. A large variety of other OCR software now uses it as a base. It is an excellent quality OCR program, with a large amount of flexibility, a solid codebase, and a large, engaged community of interested people around it [15].

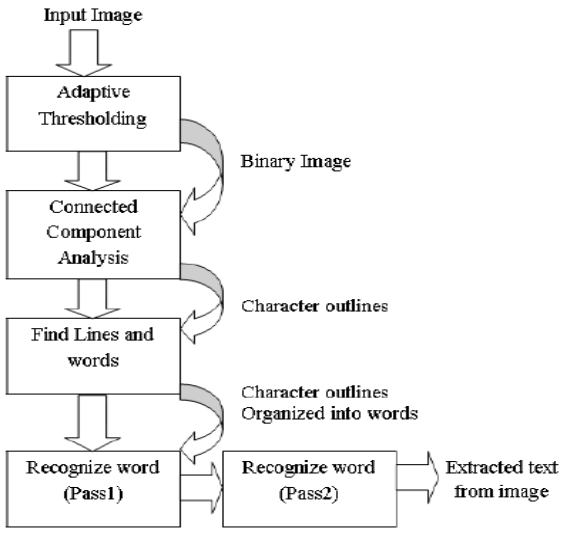

Fig 4. Architecture of Tesseract OCR

Tesseract OCR follows a traditional step-by-step pipeline processing [13]. Those steps are:

- Adaptive Thresholding: It converts images into binary images.
- Connected component analysis: It is used to extract character outlines. This method is very useful because it applies OCR for images with white text and black background. Tesseract was probably the first to provide this kind of processing.

- At this stage, outlines are gathered together, purely by nesting, into *Blobs*.
- Blobs are organized into text lines. Lines and regions are analyzed for some fixed pitch or proportional text. Text is divided into words using definite spaces and fuzzy spaces.
- Recognition proceeds as two-pass process:
  - In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory pass to an adaptive classifier as training data. Then, the adaptive classifier gets a chance to more accurately recognize text lower down the page.
  - In the second pass, the adaptive classifier run over the page to recognize words that were not well enough recognized in the first pass. A final phase resolves fuzzy spaces, and checks alternative hypotheses for the x-height to locate smallcap text.

5. Training Tesseract on Amazigh language

5.1. Why Tesseract?

Examples of tools mentioned above and many others are good quality OCR programs, but Tesseract has important criteria for which he was chosen.

The first relevant criterion in Tesseract is the fact that is free and open source (FOSS), which is an advantage and a key point in the research development.

Usually, whenever Tesseract is compared to another free OCR tool, it is the best whether in terms of recognition rate or speed [16]. Even, when it is compared with the Finereader commercial tool, Tesseract arrives to rub it and managed to overtake for handwritten writing [17].

The specificity of the Amazigh language transcribed in Latin characters is the presence of diacritic below and above a large number of characters. The experiments on Tesseract for diacritical languages, such as ancient Greek [14] and Urdu [18], have shown that it is strong enough for this type of languages.

Hence, the interest to train this tool on Amazigh language transcribed on Latin characters. The process of training passes by tree steps: generation of corpus, creation of the traineddata file and the training [13].

5.1.1. Box generation

The first step is to generate corpus composed of different characters used in the transcription of Amazigh in Latin. For this purpose, we use jTessBoxEditor.

jTessBoxEditor is a box editor and trainer for Tesseract OCR. It provides box data editing for both Tesseract 2.0x and 3.0x formats, and full automation of Tesseract training. It can read images of common image formats, including multi-page TIFF. The program requires Java Runtime Environment 7 or later.

This interface (Figure 5) allows adding text file containing characters to train, define the font desired and specify noise degree in order to generate boxes.

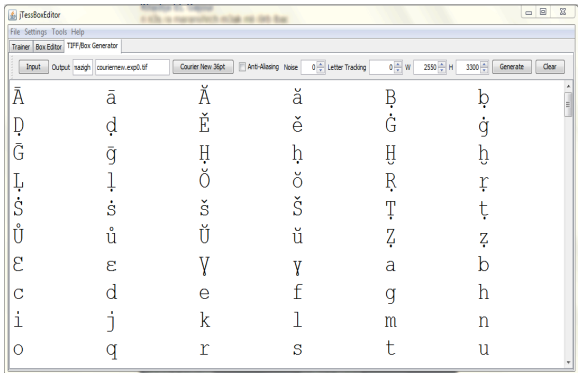

Fig 5. Box generator tab from jTessBoxEditor tool

After creating our file containing the characters, we upload it and we choose the appropriate fonts to the characters' type. The fonts used are Arial, Calibri, Cambria, Charis SLI, Tahoma and Times new roman in bold or/and italic. In total, we have 24 different writing formats according to font.

We define the maximum level of noise in order to increase the recognition quality. Then, we generate the boxes. Each box corresponds to a specific writing format.

Figure 6 presents a box. It contains the coordinate of each character in the created image.

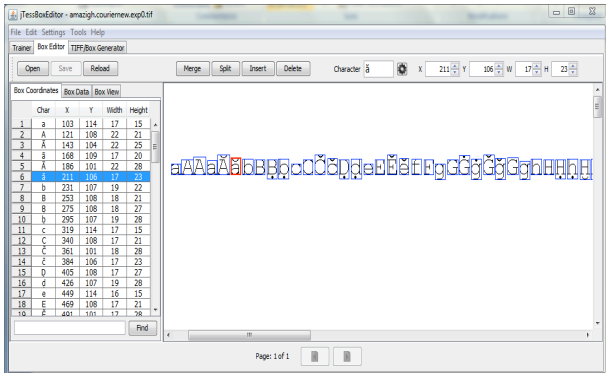

Fig 6. A box example in the box editor tab

5.1.2. Creation of traineddata file

The traineddata file allows the tool training. To create this file, we need a set of files which are:

- BOX file: the set of box files generated previously.
- FONT_PROPERTISES file: contain the used fonts with their properties. Respectively <italic>, <bold>, <fixed>, <serif> and <fraktur> are all simple 0 or 1 flags indicating whether the font has the named property.
- FREQUENT_WORD_LIST file (.frequent_words_list): file containing frequently used words in the learning language. Such as “nna” (wich), “n” (of), “nns”(his), “id”(it) and “Iγ” (If) in Amazigh language.
- WORDS_LIST file (.words_list): list of words, contain at least one word of the language.

All files must begin with the name of the language that was set in the box generation phase.

The traineddata file is created in the Trainer tab of jTessBoxEditor tool. In our case the files are:

- Amazigh.arial.exp0.box: for the “arial” font.
- Amazigh.font_properties
- Amazigh.frequent_words_list
- Amazigh.words_list

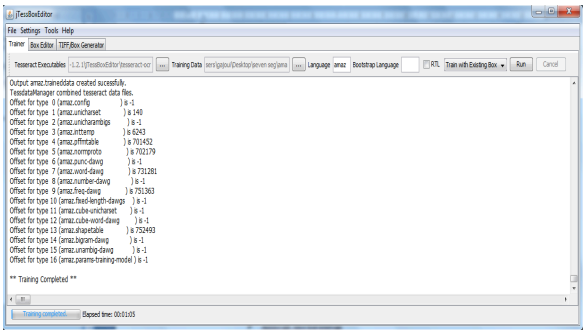

Fig 7. The Trainer tab from jTessBoxEditor tool

5.1.3. Training Tesseract

To train Tesseract, we need to copy the traineddata file generated after the execution of the Trainer, and place it in the tessdata folder in Tesseract.

6. Experiments & Results

6.1. Experiments

To test the Tesseract on our corpus, we use the VietOcr GUI. VietOCR, is a GUI frontend for Tesseract OCR engine, available in Java and .NET executable.

After using the traineddata file to train the Tesseract in VietOCR, we use a set of document extracted from different books to evaluate the system.

The documents used are 220 pages collected from 4 different books [8] [19] [20] [21] written in Amazigh language transcribed into Latin.
Part of this collection has undergone a pretreatment to increase the image quality while the other is kept with low quality in order to view the system behavior in both cases.

The documents are divided into two parts:

- Doc 1: documents in good quality.
- Doc 2: documents in low quality.

6.2. Results & Analysis

Our goal is to test the system on the corpus created for the Amazigh language transcribed into Latin. For this, we varied two properties. The first is the training font size and the second is the quality of the document to recognize.
Recognition rates are given in the following table:

		Font size variation		
		14 pt	36pt	48pt
Document quality variation	Doc 1	25%	92%	85%
	Doc 2	18%	75%	70%

Table2. Recognition rates

The results show that the improved recognition is 92% obtained on documents of good quality with the font size 36pt.

We note that the percentage of recognition varies remarkably between the corpus based on size 14pt, 36pt and 48pt.

For the font size 14pt, we remarked many classification errors. For example:

- The capital letters are confused with lowercase.
- One single character is recognized by several characters: "n" is recognized as "rr", "m" as "rrr" and "a" as "zt".
- The subscribed dot is generally ignored, "ṭ" confused with "t", "ḍ" with "d" and "ḥ" with "h".
- Other errors such as "a" recognized as "z", and "ɣ" as "Y".

For the font size 48pt, there are some classification errors such as:

- The capital letters are confused with lowercase.
- The subscribed dot is ignored in "ẓ" and "ṛ"
- The diacritic unrecognized, "ḫ" is confused with "ḥ"

While, experiences for the font size 36pt show that this size is the best for the training. Classification errors with this font size have been reduced to:

- The capital letters are confused with lowercase.
- The character "ḍ" is confused with "d", "ṭ" with "t" and "ɣ" with "Y".

On the other hand, the quality of the document influences the result of recognition. We can notice that even with a low quality, the system reaches an important recognition of 75% but it is even better with pretreatment. Hence, the importance of improving the system pretreatment phase.

7. Conclusion

In this paper, we are interested in the optical character recognition of documents, which is an active area of research today. We have introduced the OCR system and its components. Then, we have presented the Amazigh language. Given the success that the Tesseract tool has approved, we chose to apply it for the Amazigh language transcribed into Latin. This language that has not been explored in depth on the OCR field.

This study opens several perspectives such as improving the preprocessing phase by applying different treatments to increase the percentage of recognition, and enriching the used corpus. This enrichment consists on adding a new set of characters and executing the training for new fonts.

References:

[1] El Gajoui K., Ataa Allah F. Optical Character Recognition for Multilingual Documents: Amazighe-French, *The 2nd World Conference on Complex Systems, Agadir*, Morocco, November 2014

[2] Line Eikvil . OCR, Optical Character Recognition, Norsk Regnesentral, 1993

[3] Belaïd A. Reconnaissance automatique de l'écriture et du document, *Campus scientifique*, Vandoeuvre-Lès-nancy, 2001

[4] Charles P., Harish V., Swathi M., Deepthi CH. A Review on the Various Techniques used for Optical Character Recognition, *International Journal of Engineering Research and Applications*, 2012
[5] Roux A. Choix de Version Berbères Parler du Sud-Ouest Marocaine, France, 1951
[6] Skounti A., Lemjidi A., Nami M. Tirra aux origines de l'écriture au Maroc, Publications de l'Institut Royal de la Culture Amazigh, Rabat, 2003
[7] Laoust E. Mots et Choses Berbères, Paris, 1920
[8] Stroomer H. THE ARGAN TREE AND ITS TASHELHIYT BERBER LEXICON, Université de Leyde, Etudes et document berbères, 2008
[9] http://Finereader.abbyy.com
[10] https://code.google.com/p/ocropus
[11] http://code.google.com/p/tesseract-ocr
[12] Breuel T. M. The OCRopus open source OCR system, Proc. IS&T/SPIE 20th Annu. Symp., pp.1 -15 2008
[13] Ray S. An Overview of the Tesseract OCR Engine, *International Conference on Document Analysis and Recognition,* 2007
[14] Nick W. Training Tesseract for Ancient Greek OCR, *Google Inc "eutypon28-29"*, October 2012
[15] Patel C., Patel A., Patel D. Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study, *International Journal of Computer Applications*, Volume 55 – No.10, October 2012
[16] Dhiman S., Singh A. Tesseract Vs Gocr A Comparative Study, *International Journal of Recent Technology and Engineering (IJRTE)* ISSN: 2277-3878, Volume-2, Issue-4, September 2013
[17] Heliński M., Kmieciak M., Parkoła T. Report on the comparison of Tesseract and ABBYY FineReader OCR engines, Improoving Access to Text.
[18] Qurat ul Ain A., Sarmad H, Aneeta N., Umair A, Faheem I. Adapting Tesseract for Complex Scripts: An Example for Urdu Nastalique, *11th IAPR Workshop on Document Analysis Systems (DAS 14)* 2014
[19] Justinard C. MANUEL De BERBERE MAROCAIN (Dialecte Rifain), Librairie Paul Geuthner, Paris 1926
[20] Lasri Amazigh B. IJAWWAN N TAYRI, Marrakech, Imp Imal, 2008
[21] Leguil A. CONTE BEREBER GRIVOIS DU HAUT ATLAS, L'Harmattan, Paris 2000
[22] Muaz A. Urdu Optical Character Recognition System, Thesis, 2010
[23] Mithe R., Indalkar S., Divekar N. Ravina Mithe, Supriya Indalkar, Nilam Divekar, *International Journal of Recent Technology and Engineering (IJRTE)* ISSN: 2277-3878, Volume 2, Issue-1, March 2013
[24] Kinhekar S., Govilkar S. Comparative Study of Segmentation and Recognition Methods for Handwritten Devnagari Script, *International Journal of Computer Applications* (0975 – 8887) Volume 105 – No. 9, November 2014