Printed and Handwritten OCR for Arabic-script Languages

Technical Notes

CiyaSoft OCR Team
Initial Version 4/2004
Latest update: 5/2018

1. Definitions

OCR: Optical Character Recognition for printed documents.
HOCR: OCR for "unconstrained" Handwritten documents.
Arabic-based Languages: Languages such as Farsi (Persian), Arabic, Pashto, Dari and Urdu which are based on Arabic alphanumeric characters (or its extensions).
AHOCR: HOCR for Arabic-based Languages.
CiyaICR: Trade name for CiyaSoft's OCR (ICR=Intelligent Character Recognition).
MT: Machine Translation.
CiyaTran: Trade name for CiyaSoft's MT for Arabic/Farsi to/from English.
A Segment: A combination of connected characters (cursive form) that does not necessarily have a meaning but occurs in actual words.
Dotless Segment: A segment with dots and diacritics removed.
An Object: A segment, a dot-less segment, or a single character (special case of a segment).
WSC: Writing Style Class.

2. Overview

Language tools are integral, are often a serious and consequential operational deficiency, especially in operations that involve Arabic-based languages. There is an acute need to convert unconstrained handwritten (handprint) documents--frequently of degraded quality--in Arabic-based languages into digital forms so that MT, Concept Extraction and Summarization software as well as human translators can analyze them for their potential value. These documents have been, and are being acquired at tremendous cost and effort, but until the process, quality enhancement and conversion into machine-readable form can be automated, their value cannot be exploited. The backlog of unprocessed documents is enormous and growing. This problem cannot be solved by simply re-keying and throwing more manpower at it.

3. The Purpose of this document

This document is a summary of Ciyasoft's research efforts in AHOCR which identified the issues, and developed the software.

In 2002, CiyaSoft took up the task to investigate whether its existing technology, experience and software tools of MT and printed OCR could be utilized to design and implement an AHOCR. A series of related research tasks were subsequently defined and conducted, and the collected data were tested for simulation.

The findings from the collected information and the combination of the results of the simulations and test cases showed that implementation of an AHOCR was not only possible, but the peculiarities of the script would have allowed for innovative designs of recognition engines which could yield a smaller error rate than OCR for unconstrained English scripts.

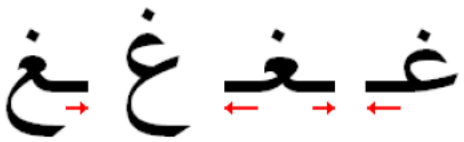
This document provides a high-level background on the issues CiyaSoft faced in developing the AHOCR.

4. Characteristics of letters and words in Arabic-based Languages

There are over 30 alphabetic characters in Arabic-based languages. Here are some examples:

هـ قـ ثـ چ ژ شـ مـ غ ؤ ض

Because characters can connect to each other, most words consist of one or more segment of two or more connected characters. Most characters have four shapes (about 10 have only two shapes, i.e. they either connect from the right, or do not connect at all.) A character can connect from the left, right, both ends, and it can be in solitude shape (not connected). The following is an example of four shapes of one character:



A word may consist of a single character (example 1 below), a single segment consisting of 2 or more connecting characters (example 2), or two or more segments (examples 3 and 4):

5. Challenges of OCR and HOCR for Arabic-based Languages

5.1 Separating Characters

While OCR for English must recognize letters, OCR for Arabic-based languages would require schemes to separate the characters first before recognition. Separating the characters is not always as easy as using a vertical vector scanning from right to left; in fact, our study showed that scanning vector failed to properly separate connecting characters of over 50% of the cases of a randomly-selected printed text with a random font, size and style. Jagged vertical vector scanning, head-and-tail identifying, end-to-end tracing and neural networks are used independently to yield a higher success rate in character separation.

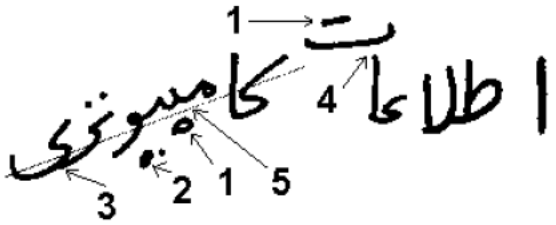
5.2 Dots

The location (top or bottom) and number of dots create yet additional challenges that do not exist in the English OCR counterpart (except for the two lower case letters "i" and "j").

5.3 Other Difficulties

CiyaSoft's implementation of OCR for both printed Arabic-based documents and AHOCR use complex dot identification and character separation algorithms. But AHOCR has to solve a few additional problems; some of the more important ones are:

1) Variety of dot shapes,
2) Irregular ratio of weight of dots to weight of the body of the cursive form,
3) Crossing characters,
4) Irregular kerning, and
5) local skews:



6. Using Peculiarities to Increase Overall Recognition Accuracy

More than half of Arabic alphabetic characters have dots (numerals do not have dots and do not connect). In a typical contemporary Arabic or Farsi text, however, less than a third of characters are "dot-less". Also, more than two-thirds of characters have four shapes, that is, they can connect to other characters in three different ways; and although nearly 80% of words in the dictionary include one or more segments, this figure is more than 90% for words of the contemporary text. These percentages show significant peculiarities that may be of little or no value from a linguistic point of view but were fully taken advantage of in the design of the CiyaSoft's recognition engines.

AHOCR uses the same successful techniques employed in CiyaICR to separate characters and to handle dots using its character-based engine. To solve the additional problems, four independent segment-based recognition engines exploit the statistical information about segments that would have otherwise contributed to recognition errors.

7. Use of Fuzzy Logic

Fuzzy logic is used in that each engine is designed to identify and solve one or more classes of problems by rendering the result with a probability of accuracy. An "object" may be recognized with 100% accuracy by one engine, a 0% accuracy by another, or a percentage in between, by a third engine. The character-based engine is less error-prone to loops and curves, while engines 2 and 3 render more accurate results recognizing segments with mutually exclusive unique dot patterns. Engine 5 ignores dot information, so it is strong in recognizing dot-less segments. Each segment is submitted to all engines and fuzzy logic algorithms analyze the returned probabilities of accuracy.

8. Preprocessing

Preprocessing includes support for manual scanning, automatic scanning (batch processing), document enhancement, zoning (identification of text areas), line separation,

and segment separation. Separated segments are processed again by segment-based engines to detect and further separate intersecting segments.

While scanned image may be corrected by de-skewing and enhanced by noise removal, degraded pages could significantly affect recognition accuracy and such images should go through another layer of processing. Before low-quality images could be upgraded, the language of the document should be identified as one of the measures of degradation.

8.1 Language Identification

Once a document is scanned, a preliminary analysis should be performed to determine the prominent language of the text zones. This determination is necessary to measure the quality and to decide if document enhancement would be necessary. It is also needed to load the appropriate language-dependent databases.

It is assumed that AHOCR software is, for the most part, executed in batch mode to process a large number of documents automatically. Therefore, the same successful algorithms used in CiyaTran are utilized to identify the language. The algorithm first determines base alphabet, e.g. English, Arabic, Farsi, etc. Then a few text zones are selected at random and recognized. The recognized segments are checked against all databases of the 1,000 most frequently-used segments for the relevant alphabet group. Our research shows that this method is successful in over 95% of randomly-selected texts by examination of only 10 randomly-selected words from the text zones. If the result is not conclusive, the database of the letters, most-frequently present in contemporary text for each of the languages, is used to strengthen the determination.

8.2 Degradation Detection

Unless the user informs the software that a document is degraded, in batch mode, where thousands of pages of document are expected to be processed automatically, AHOCR needs to measure the degree of degradation of each page automatically and individually. Sources of image degradation include perspective distortion which occurs while scanning thick, bound documents; optical perturbation of scanning device, digitization process, speckle, blur, jitter and threshold. Image degradation detection and correction is a common problem among all applications of digital image processing for which there are classic solutions. However, scanning of handwritten documents usually introduces additional factors contributing to degradation, such as low-quality paper, fragility of paper due to exposure to the environment (humidity, light and temperature variation), presence of cuts and fold lines, non-standard sizes, ink smudges, lack of continuity in color or gray level, line thickness and sharpness, etc. which have to be detected and resolved.

Image-processing algorithms that would allow easy location of text zones, even when the document has been degraded are used. Degraded areas of the scanned image are also identified using similar algorithms. Therefore, one way of degradation detection is as follows: Once the language is identified, the quality is measured by finding the ratio of the areas of recognizable text zones and total text zones. If the ratio is smaller than a threshold, the text zones are treated by different enhancement techniques to see if the ratio has changed, allowing measurement of degradation.

8.3 Quality Enhancement

To enhance the quality of a degraded document, other than the well-known standard techniques, the scanned image is passed through various digital filters whereby its colors may be adjusted and the page may be rotated (de-skewed) until the ratio of unrecognizable text zones becomes less than a threshold.

9. Engines

CiyaSoft's AHOCR takes advantage of most of the tools, engines and databases designed for CiyaICR. But it employs four additional proprietary recognition engines, which are based on completely new approaches suited for Arabic-based languages.

First, preprocessing section scans each line of the image from right to left and submits one segment to each engine at a time. In addition to the four engines that work directly on the segments, one engine, namely the character-based engine, separates and tries to recognize each of its characters.

Depending on the requirements of each engine, a number of functions operate on each segment before analysis by an engine can begin. These functions perform such tasks as normalization, smoothing, thinning, dot removal and filtering.

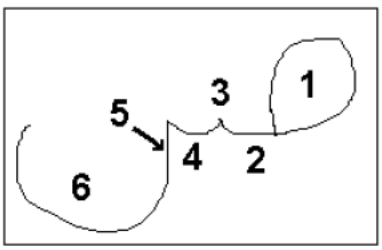
9.1 The Character-based Engine (Engine 1)

Engine 1 works with the features of a character. First, a number of classes of features are defined and mathematically described. Then each character is defined by an appropriate list of features. For example, there may be 100 different definitions of loops. These 100 definitions constitute a class, namely the "loop class". There are also classes of lines and curves of various type. During the training of a new character, the training software will try to mathematically define the character by a list of features that best describe it. If a loop is found, for example, the class of loops is searched for a match. If a match is not found, the definition of the new loop is added to the class statically.

Each feature is defined by ranges of distances (in pixels) and angles. Therefore, a match is not required to be exact in length, or angle, so long as the lengths and angles are within the defined ranges.

In the following example, the letter "SAD", scanning from right to left, is defined as follows:

1) Loop Type 24
2)Horizontal Line Type 5
3) Edge Type 32
4) Dent Type 14
5)Vertical Line Type 4
6)Concave Type 43



Obviously, a given character could be defined by more than one set of features. However, with a large sample size, many of the definitions would overlap and the number of features of a class would eventually converge.

9.2 Dot Analysis

The character-based engine works on the dots separately. Since, presence of dots is meaningful only at certain locations (top or bottom) and the count (one to three), not all combinations of count and location are valid. The presence of dots in invalid locations provides additional information on the type of the filter to use for further noise removal. If a dot is not noise, its thickness could point to the size, and sometime, type and style of the font.

Dot analysis takes place before normalization and thinning, to preserve the information about its shape and dimensions.

9.3 Segment-based Engines (Engines 2, 3, 4 and 5)

Extensive research on composition, count and distribution of segments naturally occurring in corpus has revealed highly useful information that fundamentally affects the design of optimum segment-based engines. One of the most useful finding is the distribution of segments size in the Arabic corpus:

2-charcter long segments 3%
3-charcter long segments 20%
4-charcter long segments 38%
5-charcter long segments 26%
6-charcter long segments 9%
7-charcter long segments 4%

In addition, word construction rules limit the generation of new segments and certain segments tend to be present in more words than others. Therefore, sample collection should allow more variations of those most common segments to increase the probability of correct recognition.

Engines 2 through 5 work on segments. While recognition of a character would require identification from a reasonably small number of shapes; the number of possible segments is astronomical. The total number of segments that can be constructed is over 20 billion, if a segment can be up to 7 characters long (the maximum size in the corpus).

Since each segment can be represented at least 200 different ways (see section 12.1), an exhaustive database would require over 4 trillion definitions. If each definition takes an average of 3 Kbytes, the size of the database would be over 12,000 terabytes, which is not practical with today's storage devices.

An examination of the dictionaries of each of the Arabic-based languages shows that the number of actual segments is far less than the number of segments that can be constructed. Further, segments that may or may not have a meaning are present repeatedly to construct the words. In response to creation of new words, if the language expands without importing words from foreign sources, the number of these segments will not increase. Therefore, the only new segments that may enter the database of segments are proper nouns, or technical terms from foreign sources, e.g. English, that must be written as they are pronounced.

If we take all the words in an Arabic dictionary into account, separate the segments and discard the duplicates, 13,000 segments will remain. When segments are extracted from words of foreign source, the total will be 15,000. If obsolete words are discarded and only contemporary words are considered, this figure drops to 5,000. With dots removed, we end up with less than 1,000 segments, which would require about 600MB of storage. Therefore, engines 2 through 5 can comfortably work with a limited number of shapes, while segments that are not considered would still have a chance to be recognized by the character-based engine.

Engine 2 uses the weight distribution to uniquely define a segment. This, in part is done through analyzing the segment by its horizontal, vertical and angular histograms.

Engine 3 circumscribes a segment into a box whose dimensions would not exceed values determined to be optimum empirically. The segment is first treated by smoothing at the edges, normalization and thinning. It is further cleaned up of its extra features to render an optimal shape. When the size of the box is small, many of the variations of a given segment will be forced to render the same, or similar characteristics. The optimal size of the box has been determined to be 13x7 pixels for Arabic. Engine 3 can successfully recognize all characters in solitude shape, because the biggest character would fit into an 8x7 box, and smaller characters would require an even smaller box. With 2,000 samples of the same segment, it is expected that many of the samples would render the same representation once the image has been reduced to its most prevalent features.

Engine 4 uses Bezier curves and Fourier transformation which is the subject of another CiyaSoft white paper.

While dot information heavily contributes to the recognition of a character, it is removed before Engine 5 can work on it. Engine 5 which works on "dotl-ess" segments, identifies feature of a segment (similar to Engine 1 which focuses on features of a character). It analyzes the count, type and order of loops, vertical and horizontal lines and weights and location of features. The segment is traced from right to left and features are identified and listed as the representation of the given segment.

It must be noted that engines complement each other, as each one may be better suited in recognizing certain groups of segments.

10. Training

Each engine has separate training software, which uses its respective engine statically to generate the databases as the samples are fed into it. The training process is automatic for the most part, except for engine 1 where new classes of feature may sometimes have to be defined manually.

11. Databases

The databases are strictly language-dependent, e.g. those developed for Pashtu will not work for Arabic.

Databases contain the information that can be extracted from segments, or characters. The information includes feature list (in case of Engine 1 and 5), statistical, trace and transformation data of segments. Each engine has its own set of databases.

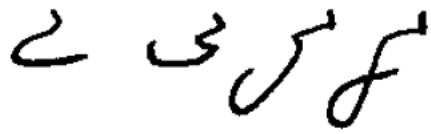
Although they should be adjusted and calibrated for each language, the engines may be used interchangeably among Arabic-based languages. The engines have been tested based on a large database of handwriting samples, encompassing a comprehensive

number of character combinations. Our investigation shows increase in accuracy to be a function of sample size and is semi-parabolic once more than 1,000 samples have been collected.

12. Sample Collection

12.1 Writing Style Classes

The similarities, which exist in the characteristics of handwriting among individuals, can be compared with the similarities between various fonts. These characteristics or Writing Style Classes (WSC) can be thought of as fonts. Our research showed that a given segment written by one individual had a 0.5% probability to be similar (in extractable features and statistical information) to the same segment written by another (randomly selected) individual. This means that the number of WSC's for Arabic, for example, needed to be less than 200 and by collecting sample handwritings of 2,000 or more different individuals, we nearly had all the different ways a segment could be written by hand. In addition, only a small percentage of segments had too many variations that necessitated defining a new set of features to describe it. For example, segments ending in letter "YEH" can be handwritten four different ways, because that letter has four distinct shapes:



12.2 Sample Collection Requirements

The samples included all possible segments that could be constructed. It did not need include all segments that were used in the corpus. It only included high frequency, dot-less segments that had been determined to be present in over 90% of the contemporary text. The character-based engines were used for recognizing other (excluded) segments.

Once the information was extracted and categorized into databases, a recognition engine was able to recognize any (included) segment with nearly 100% accuracy. We have been able to prove this on a large scale.

a. Each sample included approximately 400 to 1,000 handwritten segments without the dots.
b. The list of 400-1,000 segments was predefined.
c. The gender, nationality (the country they learned to write Arabic), level of education, age, left- or right-handedness, dexterity and other personal attributes of the writers that affect his/her handwriting style was randomly distributed. Further, the time of the day, whether the writer was tired, the way he/she sat and held the pen, speed and other factors could have affected his/her writing style. So, for best results, three separate samples were collected from each writer.
d. The writer wrote naturally. He/she was not restricted in the type of pen to use, or the size of individual characters or segments.
e. The predefined segments did not necessarily represent a word, particularly since the dots had been removed. This increased writer error and caused the rejection of the sample. Therefore, it was necessary to draft a meaningful text (a collection of meaningful sentences) that included all the required segments, with dots included.

After scanning, dots were removed, and the additional segments were isolated and discarded.

f. Before scanning and processing, each sample was carefully reviewed manually, and rejected if it contained any errors. Examples of error included:
   - Missing words
   - Lines running into each other (This is different from the issues involving crossing and local kerning in the input, which are resolved by a separate unit.)
   - Individually skewed lines (This is different from skewed page, or local skewed segment issues of the input, which are resolved by the local de-skewing algorithms.)
   - Presence of extra characters, words or symbols
   - Faded or smudged areas (problems with the pen)
   - Crossed out words
   - Folded, torn, unclean or otherwise low-quality pages

13. Testing

13.1 Regression Testing

Data collection and training, as well as optimization and debugging of the engines are gradual processes during which, accuracy is not always necessarily increasing. As the databases grew, or new algorithms were implemented, recognition of certain segments that were previously recognized correctly, were negatively affected. For example, even though a given segment may have thousands of different representations, the opposite situation was a potential source of error. Also, while an improvement in one aspect of an engine may have resolved incorrect recognition for a given class of segments, it lessened the accuracy of another class.

To have a handle on this, regression tests were executed in conjunction with training as follows: Each time a new set of data was successfully trained; a sample used in the training was added to the regression test database. Also, whenever an engine was changed in response to a situation, samples representing that situation were added to the regression test database. Subsequently, the regression test software, which used the same engines and databases, was run after each instance of engine change, or training of a new group of data.

Each engine had its own regression test software and database and was capable of offering suggestions or point to areas of code that could have contributed to the problem. At each new change, the cycle of database build, code compilation and regression test, was repeated until all problems were identified and resolved.

13.2 Quality Control

In addition to regression testing, to control the quality, AHOCR was regularly tested against random samples that were not included in training. These samples included words that were not part of the training and should have been written by individuals other than those whose handwritings were included.

13.3 Simulation Testing

Images stored in regression test databases were transformed and tested as an additional test of the software's robustness. Transformation included vertical and

horizontal stretching or compression, rotation, sharpening, fading and blurring the images.

13.4 Continuous Testing

As tests were performed continuously (in parallel) while samples were collected and learned by the engines, we saw a slow rate of increase in accuracy, as in curve B, compared to the desired rate (curve A). This was proven in both our small-scale and large-scale test cases where samples were collected and trained. The information obtained was used to make more accurate estimates on database development for other languages such as Farsi.