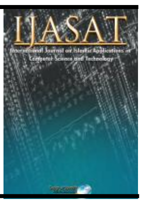
A Semi-Automatic Annotation Tool for Arabic Online Handwritten Text

Randa Elanwar
Electronics Research Institute
19 PUBLICATIONS **126** CITATIONS
SEE PROFILE

Mohsen A. Rashwan
Cairo University
152 PUBLICATIONS **993** CITATIONS
SEE PROFILE

Samia A. Mashali
Electronics Research Institute
41 PUBLICATIONS **155** CITATIONS
SEE PROFILE



A Semi-Automatic Annotation Tool for Arabic Online Handwritten Text

Randa I. Elanwar, PhD[1,a], Mohsen Rashwan, Professor[2,b], Samia Mashali, Professor[1,c]

[1] Electronic Research Institute, Cairo, Egypt (phone: 202-33310515)

[2] Faculty of Engineering, Cairo University

[a]eng_r_i_elanwar@yahoo.com, [b]mrashwan@rdi-eg.com,[c]samia@eri.sci.eg

ABSTRACT

Large amounts of ground truth data is vital for building, testing, analyzing and improving the performance of character recognizers especially those using segmentation based routines. Ground truth information, the annotation, can be associated with the document images at the paragraph level, the sentence level, the word level, and up until the character or stroke level. Providing huge annotated datasets for this purpose manually is a very taxing and error prone procedure. Therefore, it is important to complement the automatic tools for metadata extraction with tools that provide an efficient human-computer interface to experts for validation and correction to simplify the creation of recognizers. In this paper we present the first semi-automatic tool for annotation Arabic online handwritten documents. A tool provided to automate and simplify document visualization, manipulation and annotation of documents at the character level generating transcription files ready for use by any handwriting recognizer. The tool is a set of interactive user interfaces guiding the user along the whole process and reducing the human effort and time by the activation of smart segmentation utilities offering satisfying performance and allowing intervention for validation.

***Keywords*: Arabic, Document Annotation, Ground truthing, Interactive user interface, Online handwriting, Segmentation.**

1. Introduction

Annotation is the process of identifying data of particular type using additional data of different type, which precisely describes the entities that make up the former data. Annotation is a natural human data management activity. Frequent readers, who are used to read excessively and regularly, are accustomed to underline, highlight, comment on, summarize and organize information as they read to help effectively comprehending the reading material. This annotation action serves also in cutting down the search time of information because the reader doesn't have to re-read the whole page. Instead he can only read the lines he highlighted or the annotation/comment he previously wrote beside.

With the fast technology development and the spread access of information on the World Wide Web made it easier to search, upload, download, send and receive all kinds of personal and public data. This caused people to face the problem of information overflow and need sufficient support for its organization and making them accessible to the users who would in fact like to search them with relative ease.

Consequently, the need for annotation is increasingly getting important. Information grouping, filing, indexing and retrieval are studied by different researchers. From this, the idea of constructing digital libraries has emerged. Today's digital libraries increasingly include not only printed text or books but also scanned handwritten pages and other multimedia material.

As illustrated by Balasubramanian A. (2006) documents can be primarily divided into three different categories: (1) Online: Documents that consist handwriting data captured by a digitizer (e.g., personal digital assistants 'PDA' or tablet PCs), (2) Offline: documents consist of scanned copies of handwriting information that were a priori written on a sheet of paper, and (3) Printed: documents contain textual information in the Printed form which are scanned copies from a book.

All of Online, Offline, and Printed documents do not contain searchable text as is, but contain their image or ink information which cannot be searched by existing conventional text search engines which depend on matching or comparison of textual description (say in ASCII/UNICODE). Thus, annotation of such documents images allows searching and accessing of these documents using the conventional textual search.

In manual annotation, users spend time annotating huge corpus of documents manually, every region of interest (image of a line, word or character) that needs to be annotated. Similar objects need to be manually identified one after another using graphic selection tools and are annotated appropriately. It is a laborious, time-consuming and error-prone process, especially at the character level.

On the other hand, Semi-Automatic and Automatic schemes of annotation reduce the manual effort considerably. If plain transcripts of handwritten data can be made available, method to automatically align the handwritten data with the text data can be used to generate annotation of handwritten data at character level. Another way is to use robust handwriting recognizers for converting these scanned images/ink of handwritten words into textual form but current recognizers haven't reached their maturity so far. In fact it is a double-face problem as in order to create and evaluate recognition engines for handwriting, significant resources in the form of annotated datasets are required. Lack of linguistic resources for many scripts (especially Arabic), in the form of annotated handwriting datasets has been one of the major hurdles to research in these scripts.

The availability of annotation tool kits will accelerate the development of new handwriting recognizers because it saves the time wasted by researchers in their repetitive tasks of collecting and manually annotating large datasets to work on, thus, maximize efficiency, productivity and profitability. Furthermore, toolkits can be used integrate multiple recognizers through standard interfaces.

2. Background

Annotation is achieved by associating specific details, called metadata, to the main information source. Once the metadata extracted, they are tagged into XML documents called "meta-document". The XML representation is a hierarchical organization of data and annotation. It typically contains the root of the annotation hierarchy defined by the user. Each level of hierarchy contains a label element that captures annotation information at that level. The leaf elements of the hierarchy refer to raw image/ink traces.

The metadata may be related to the dataset as a whole and writer elements. It can contain some or all the following elements: *Name*, name for referring to the dataset, *Category*, type of dataset, *Version*, version number and/or date stamp of publication, *Contact*, contact info for dataset-related queries, *Source*, source of collected data, *Script*, language/script captured in dataset, *Quality*, quality of handwriting data captured in dataset, *Ground truth*, truth of what is captured, *Methodology*, design of data and collection procedure, *Annotation Scheme*, description of annotation scheme, *Date*, date when writing occurred, *Hand*, left/right handedness, *Gender*, Gender, *Age*, age at the time of capture, *Style*, predominant writing style, *Region*, native region, *Writer Name*, name of the human/automated source of labels

The categories of the metadata described above are essential for document retrieval problem, whereas, for the handwriting recognition problem the metadata that is of the main importance is the corresponding ground truth of image/ink trace.

Ground-truthing a document image, i.e., annotating the regions, text lines, words and characters, is important for document analysis research, particularly, for algorithm design and performance evaluation.

Often, document images were ground-truthed by humans since automatic tools were not available for giving desired accuracy. Automatic ground-truthing was available only for restricted cases such as documents with characters written in boxes or with large inter-line and inter-character spaces. In recent years, some ground-truthing tools have been developed for annotating document images with less restriction. For printed document images synthesized by text editing, printing and scanning, the text description (transcription) can be matched with the scanned image to generate ground-truth data. Handwritten document images can be similarly matched with the transcript, but the matching process is much more complicated due to the irregularity of document layout and written word/character shapes. Usually, a dynamic time warping (DTW) or linear hidden Markov model (HMM) optimizing a match score between a text line image and its transcript is used to align the sequence of image segments and the words/characters (Yin et al. (2009)).

3. Literature Review

In literature, few approaches that do automatic and semi-automatic ground-truthing annotation of handwriting data have been reported:

Jawahar et al. (2007) have developed a semi-automatic annotation tool for Indian printed documents by aligning image and textual content parallely at word level. Scanned documents are segmented to paragraphs, lines and then words, which are labeled with corresponding textual content. When the segmentation algorithms fail, user intervention is asked for. For the process of akshara (one or more connected components in the word image) level annotation, the best alignment between components and aksharas is found using DTW. Automatic annotation of akshara is verified and validated using a validation user interface.

Strassel (2009) has introduced a five year DARPA (Defense Advanced Research Projects Agency) program that will produce systems to automatically convert foreign language text images into English transcripts MADCAT (Multilingual Automatic Document Classification Analysis and Translation). The first two phases of MADCAT focus on offline handwritten Arabic. Documents are first manually segmented into sentences. Data is then annotated by drawing a bounding box around each line, word or other targeted element on the handwritten page. Each bounding box, or zone, consists of a unique ID, the contents and coordinates indicating the location of the zone on the page.

Yin et al. (2009) have presented an automatic annotation tool for offline Chinese document where a handwritten document image and the corresponding transcription are loaded, and it outputs the ground-truth data (annotated image). The system first extracts the connected components (CCs) from the input document image. The CCs are grouped into text lines. After segmenting the document image into the same number of text lines as the text transcription, each text line image is then aligned with its transcript. For aligning character boundaries, the optimal match can be found by DTW. After alignment, mis-segmentation and mis-labeling of characters (such errors are inevitable) are corrected manually. The tool has been used to annotate a large number of Chinese handwritten document images in the HIT-HW database (383 text line images, 8424 characters in total). A recall rate of 89.47% is observed over 8,424 transcript characters and a precision of 89.13% over 8,453 aligned characters.

Bhaskarabhatla et al. (2004) have presented a tool for online English handwriting annotation. The tool implements an open and extensible architecture using plug-ins for different operations. Segmentation plug-ins are implemented for strokes, words, and lines levels. Annotation proceeds by segmenting a digital ink document to lines, words or even strokes using an available plug-in. The segments (stroke groups) obtained are displayed in different colors using a coloring scheme. These segments generally need to be manually corrected, and the interface supports the use of both mouse clicks and keyboard shortcuts for adjustment and annotation.

Kumar et al. (2006) have presented an annotation tool for online Indian handwriting. A synthesis module is used to convert the text sequence to the corresponding handwritten form using a model of handwriting of the writer. This forms the reference handwriting, where the segmentation and ground truth are accurately known. The input handwriting is then matched with the reference handwriting using a two-stage elastic matching module. After the best match is identified, the ground truth is propagated to the words and characters of the input handwriting. The algorithm also allows manual correction of errors. The error rate at the word level was 26.5%, tested over 425 words. The character-level annotation was 96.4% when tested on a set of over 3500 characters.

Obviously, the main tasks that every annotation tool should accomplish to facilitate the creation and enhancement of handwriting recognizers and suggestions of how to do them are illustrated by Nagy et al. (2006): first, document segmentation, Secondly, Line finding and thirdly, Word extraction. Segmentation is complex and automated systems have not yet reached human accuracy. Skew and orientation problems together with the non uniform spacing between lines or between words that often occur in handwritings need a simple interface to be designed to allow humans to correct linked and broken words after line segmentation. This operation requires a display of all or part of the document image, a point-

and-drag mechanism to select regions (in most applications, rectangular regions), and a set of buttons to associate region coordinates with component types.

They also declare that the most significant variable, which is responsible for most of the expense of human annotation, is human time. As machines take over the simpler tasks, more and more expertise will be required of the operator. The cost of human labor, already the dominant factor in most information processing tasks, will become even more significant. Higher-level segmentation algorithms tend to be more error prone, and therefore require higher reject thresholds. Even a 30% reduction in overall human time will be significant in an operational application (Nagy et al. (2006)).

4. Overview of the Annotation Tool

Our annotation tool we present in this paper a semi-automatic annotation tool for Arabic online handwritten datasets. It provides a number of tools and utilities to support the tasks of handwriting data segmentation and data annotation for the training and evaluation of word recognizers.

These utilities are provided through user friendly interfaces that give the options to perform each in either automatic or manual methodology. Different approaches have been used to implement each utility. Each is detailed in the following sections:

4.1 Tool architecture and description

The architecture of our tool is composed of sequential modules of document segmentation to finer entities ending with the ink data together with the corresponding truth on the character level. Each module is implemented using different procedure and visualized on independent user interface. Each module performs the task automatically and allows human interaction for validation and error correction. The user interface (UI) design guides the user through the tool modules step by step giving warning if he tries to skip one. It can be used by non-specialists to visualize data, segment it, and automatically generate a transcription file. This can cut the time duration needed for manual data transcription/annotation to the fifth or less thus enable working on much bigger database files.

The user interface set consists of the following pages: Main UI, Word Extraction and Add Transcription UIs. The Main UI opens at the start up showing the user all operations that can be performed on a document: Text line extraction, word extraction, document transcription. Each time the user chooses an operation, he is allowed to browse and locate the document to be manipulated. When the user chooses to extract the document text lines, the text line extraction module is activated using the procedure described in section 4.2, the results are visualized as separate text lines on different UI. Then, when the user chooses to do word extraction, each of the extracted text lines is visualized in sequence allowing both automatic and manual extraction. The automatic word extraction module performs using the procedure described in section 4.3. Finally, when the user chooses to segment the document words, each of the extracted words is displayed in sequence on different UI allowing automatic and manual segmentation and truth alignment and visualizing the annotation result for validation and correction. The automatic word segmentation proceeds as described in section 4.4. The different UI design and functionality is described in section 4.5.

4.2 Text line extraction utility

Elanwar et al. (2011) give an overview of the text line detection system. The input is Arabic handwritten text in online format. After preprocessing to remove the diacritical marks (dots), the document is divided/shredded vertically into adjacent strips. Spatially related strokes (overlapping or touching) in each strip are clustered to form units. These stroke units make the same sense of a connected component, or component for short, that consists of one or several horizontally overlapping strokes. These units/components have to be enlarged/extended to form segments. Dynamic programming (DP) is used to merge a pair of text line segments and optimize the global cost function to merge a complete text line. Between each two consecutive text line segments the dynamic programming paths will start. It searches an optimal path of merging segments guided by heuristic cost function that takes different criteria into account to form complete text lines: Direction Penalty, Cross-Over Penalty, Distance Penalty, and Width Penalty. The direction penalty is used to emphasize the effect of merging direction of text line segments (we have 8 possible directions). The cross over penalty insures correct successive merging spatially. There shouldn't be stand alone segments crossed over in between the two segments to be merged. The distance penalty is to insure merging close segments rather than distant ones. The width penalty is used to favor long text lines. Its value decreases as the sum of widths of the two merged segments approach the value to the whole document width. Finally the dots are restored and text lines are detected. The details of the penalties values selection and the DP stopping threshold selection are given in Elanwar et al. (2011). The extraction results are discussed in details in section 5.

4.3 Word extraction utility

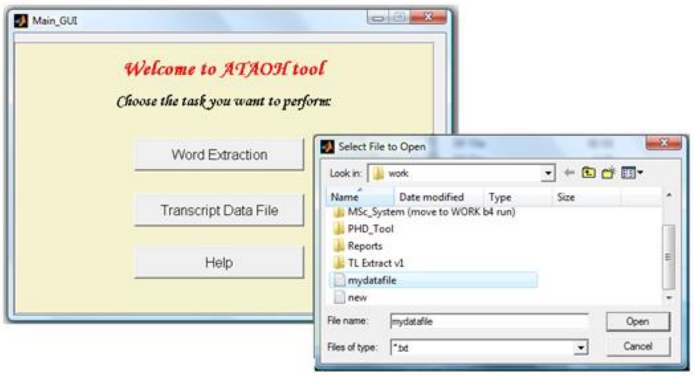
The word extraction procedure first classifies the white gaps between words connected components into either intra-word or inter-word gap according to some local and global online features extracted from each gap together with the groups of strokes encompassing the gap (detailed by Elanwar et al. (2012)). The classifier is a polynomial kernel support vector machine (SVM) which decisions are used for initial word extraction. A post stage is added to the system to test the extracted words for under-segmentation and resolve this under-segmentation by reconsidering the gap type decisions for the stuck word. Rule-based stick detection stage is employed, then stick resolution is achieved using classifiers fusion. Classifiers decision fusion takes place by consulting five different classifiers (four SVM and a radial basis function neural network 'RBF NN') and feeding their decisions to a separate pre-trained SVM to make the final decision. Most stuck words are correctly detected and a lot of them have been correctly resolved. The post stage leads to remarkable error reduction compared to single classifiers performance as detailed in section 5.

4.4 Word segmentation utility

The word segmentation procedure proceeds by simultaneous segmentation-alignment using hidden Markov model (HMM). The input word strokes undergo the main preprocessing operations of smoothing, re-sampling, normalization and complementary strokes removal where dots, hamza and other secondary (delayed) strokes are identified and filtered out from the main word strokes based on heuristic rules. Local and vicinity features are computed for a window of samples moving in the samples writing order direction. These features are discussed in details by Elanwar et al. (2012). Frames made up from the extracted features are passed to pre trained HMM to align the word strokes ink data with the user entered ground truth. The HMM output characters boundaries represent the proposed segmentation points (PSP). These PSPs are then validated by rules-based post stage to relocate their position on the word strokes for bad- segmentation error reduction and segmentation enhancement. Most errors cases are cured and remarkable segmentation enhancement is achieved as shown in section 5.

4.5 User interfaces

The tool is composed of a set of interfaces designed and implemented on MATLAB that guide the user along the whole process of data manipulation. The first is Main UI which is the entrance to the tool utilities (figure 1). The Word Extraction Pushbutton is the entrance to “Word Extraction UI” to enable user to make word separation (automatically or manually). The Transcript Data File Pushbutton is the entrance to “Add Transcription UI” to enable user to annotate the extracted words and generate transcription files. The Help Pushbutton is the entrance to the tool documentation for helping the user activate and access the UIs.



Look in: work

Name	Date modified	Type	Size
MSc_System (move to WORK b4 run)			
PHD_Tool			
Reports			
TL Extract v1			
mydatafile			
new			

File name: mydatafile
Files of type: *.txt

Figure 1: The tool Main UI

The Word Extraction UI which appears when the user chooses the word extraction process, displays the document text lines sequentially allowing the user to automatically/manually extract words (figure 2). The Auto Segmentation Pushbutton segments the current text line automatically using an embedded word extraction module described in section 4.3. The Manual Segmentation Pushbutton segments the current text line manually by specify the segmentation locations between words using mouse clicks. The Undo/Redo Pushbuttons undo/redo the last manual segmentation mouse click action. The Back Pushbutton turns back to the previous text line with/without extracting words of the current text line. The Continue Pushbutton turns over to the next text line after saving all extracted words of the current text line. The Refresh Pushbutton re-plots the current text line for segmentation correction. The Skip Pushbutton turns over to the next word page without extracting words of the current text line. The Check Pushbutton plots each extracted word separately for check. The Finish Pushbutton generates an output file to which all extracted words information is written.

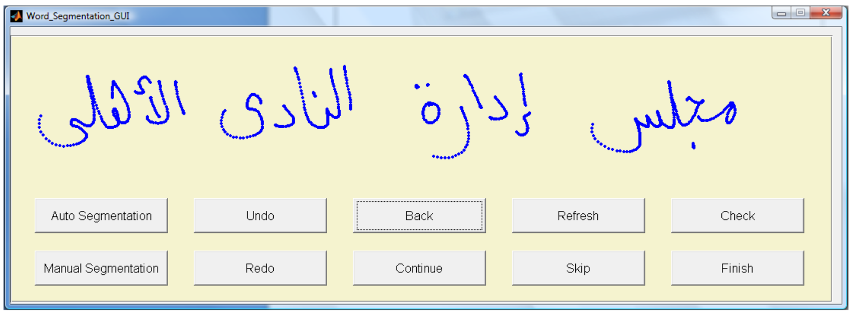


Figure 2: The Word Extraction UI

The Add Transcription UI which appears when the user chooses the transcription process, displays the document words extracted previously in sequence allowing the user to automatically/manually segment words (figure 3).

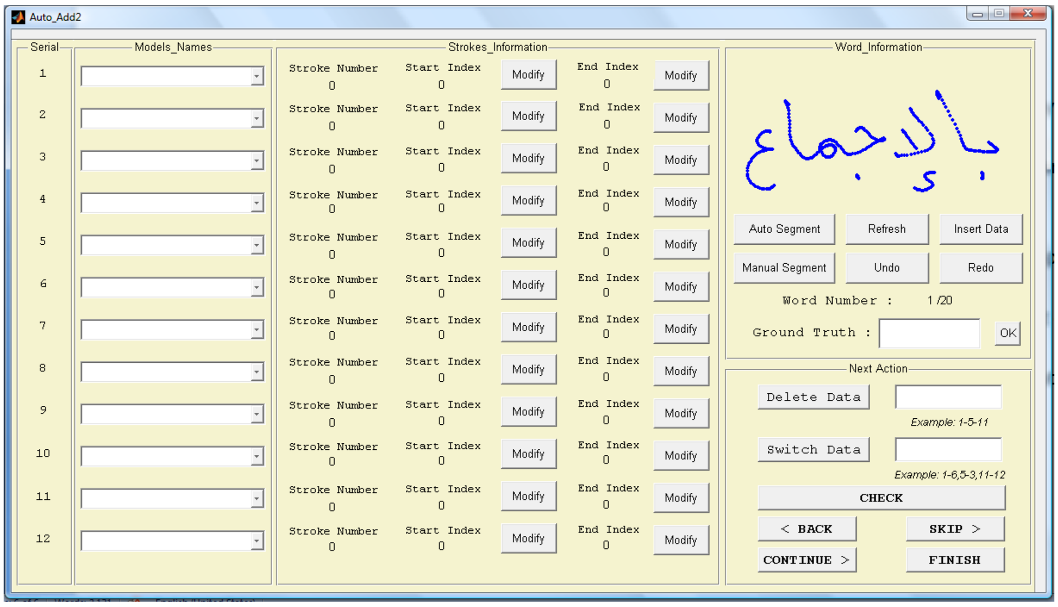
The Ground Truth text area is used to enter the ASCII code text corresponding to the current word. Once filled the model name menus are filled with the corresponding character names.

The Auto Segment Pushbutton allows user to automatically segment the current word into characters using the procedure described in section 4.4. The Manual Segment Pushbutton allows user to manually segment the current word into characters using successive mouse clicks. The Undo/Redo Pushbuttons undo/redo the last manual segmentation mouse click action.

Once segmentation is done, each segmented character's ink information (stroke number(s), start point(s) and end point(s)) is computed and displayed in the corresponding text areas by pressing the Insert Data Pushbutton. The Refresh Pushbutton re-plots the current word for segmentation correction.

The Delete Pushbutton deletes the characters information specified by the user in the 'Delete text area'. The Switch Pushbutton switches the characters information (Names or ink information) specified by the user in the 'Switch text area'. The Check, Back, Continue, Skip and Finish Pushbuttons function the same like those in the Word Extraction UI but on word level rather than text line level.

The annotation output is in the form of transcription file where each word in the document is transcribed on the character level. Each character's information: name, number of composing strokes, stroke indices, start point(s) and end point(s) is listed ready to be used by any recognizer.



Serial	Stroke Number	Start Index		End Index	
1	0	0	Modify	0	Modify
2	0	0	Modify	0	Modify
3	0	0	Modify	0	Modify
4	0	0	Modify	0	Modify
5	0	0	Modify	0	Modify
6	0	0	Modify	0	Modify
7	0	0	Modify	0	Modify
8	0	0	Modify	0	Modify
9	0	0	Modify	0	Modify
10	0	0	Modify	0	Modify
11	0	0	Modify	0	Modify
12	0	0	Modify	0	Modify

Word Number : 1 /20

Ground Truth :

Figure 3: The Add Transcription UI

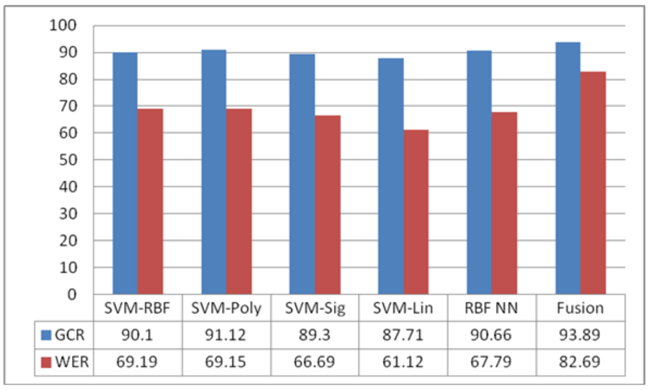
5. Experiments and Discussion

The Arabic language differs greatly from other Latin languages not only in its characters cursiveness but also in the language structure as well. For Arabic language, we don't yet have public text line data sets for online Arabic handwritten documents. For this reason, we collected a dataset of our own, OHASD dataset previously presented by Elanwar et al. (2010).

The OHASD dataset is the first sentence dataset collected for Arabic online handwriting research purposes. 48 volunteers' handwritings of different subjects are collected on tablet PCs. The handwriting is natural and unconstrained. The data set is composed of 154 documents with total 670 text lines, more than 3800 words and more than 19,400 characters. We divide our dataset into 3 parts: training, validation and test sets. The experiment results obtained by each module of the annotation tool are detailed as follows:

First, the tool performs text line detection based on dynamic programming with accuracies 96.67% document accuracy (DocAcc) and 98.5% text line accuracy (TLAcc). Experiments on the training dataset are conducted for parameters values optimization. Final results are taken for test dataset. Rules-based stick detection post stage is employed to detect and resolve stuck text lines.

Secondly, the tool performs word extraction based on white gap classification to inter- or intra- word gaps. The word extraction module is designed using the validation data. First the SVM classifier pre-stage proposes initial word extraction. The pre-stage output is either correct extraction, over-segmentation (due to obviously wide intra-word gaps) or under-segmentation (due to strokes overlapping). The stick resolution post-stage (using multiple classifier decision fusion) achieved a remarkable rise in the word extraction rate (WER) compared to single classifier performance (figure 4). Applying the test data to our system resulted at gap classification rate (GCR) of 88.4% and WER of 71.5%.



	SVM-RBF	SVM-Poly	SVM-Sig	SVM-Lin	RBF NN	Fusion
🟦 GCR	90.1	91.12	89.3	87.71	90.66	93.89
🟥 WER	69.19	69.15	66.69	61.12	67.79	82.69

Figure 4: Comparing stick resolving by decision fusion to single classifiers performance

Thirdly, the tool performs word segmentation based on HMM. The HMM design, features and parameters optimization are chosen according to excessive experiments using the validation dataset. HMM is employed as a recognizer once and as an aligner another time for parameter optimization. The best performing HMM recognizer design was experimentally found to be composed of 36 states with the first 8 states having 16 Gaussian mixtures and a single Gaussian elsewhere. In other words, instead of having a HMM with all its states having equal number of Gaussian mixtures, we define a new HMM with variable Gaussian mixtures number per state. The performance comparison can be obviously noted in the charts in figure 5. The best performing HMM aligner design was experimentally found to be composed of 45 states with the first 8 states having 16 Gaussian mixtures and a single Gaussian elsewhere.

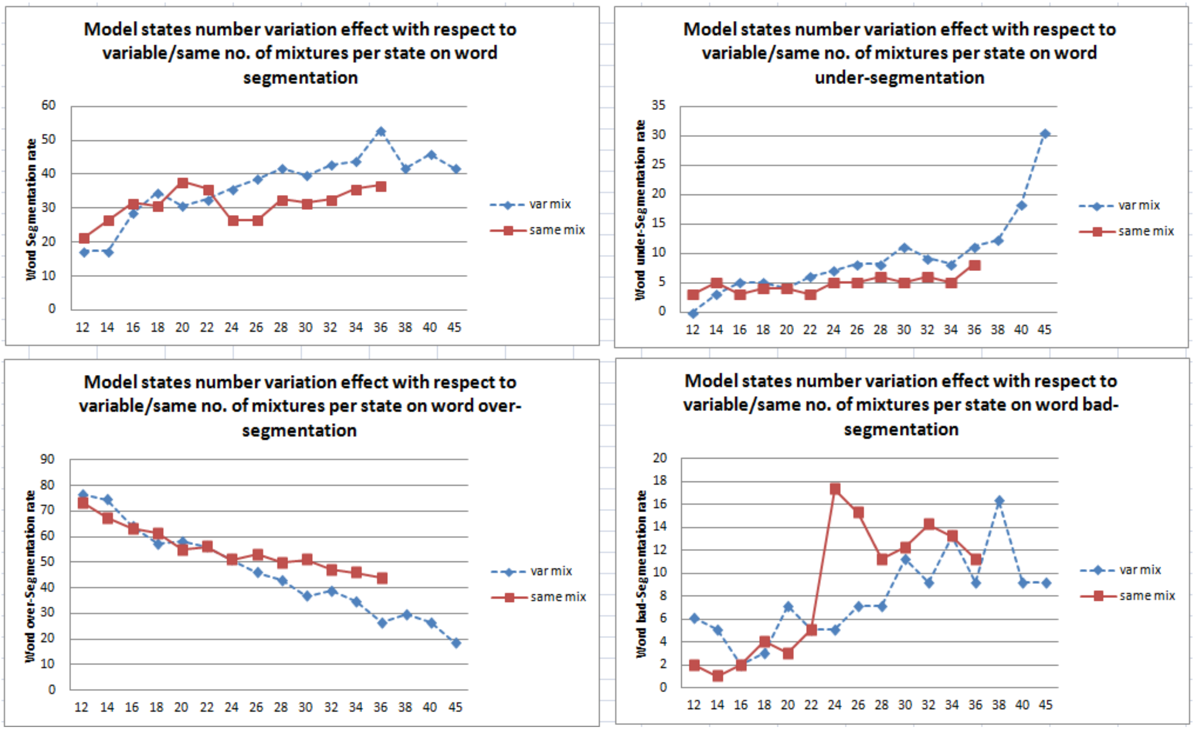


Figure 5: The effect of location of variable-mixture states within the HMM

The average system results on both validation and test datasets in both HMM modes are detailed in table 1.

Table 1: Word and character segmentation results obtained by the tool.

Test data Aligner	Word Correct seg.	Word Under seg.	Word Over seg.	Word Bad seg.	Correct character seg.
HMM Output	52.23	3.38	4.06	40.32	74.47
After PSP validation	75.64	4.19	8.12	12.04	89.42
After dot restoration	74.42	3.52	8.25	13.80	87.04

Validation data Aligner	Word Correct seg.	Word Under seg.	Word Over seg.	Word Bad seg.	Correct character seg.
HMM Output	73.35	0.00	0.00	26.65	88.85
After PSP validation	94.91	0.00	0.60	4.49	97.83
After dot restoration	94.61	0.00	0.60	4.49	97.10

Test data Recognizer	Word Correct seg.	Word Under seg.	Word Over seg.	Word Bad seg.	Correct character seg.
HMM Output	37.05	21.61	21.61	19.73	72.46
After PSP validation	51.54	23.09	17.05	8.19	80.68
After dot restoration	36.64	18.66	28.72	15.84	71.35

Validation data Recognizer	Word Correct seg.	Word Under seg.	Word Over seg.	Word Bad seg.	Correct character seg.
HMM Output	46.41	18.56	23.95	11.08	80.75
After PSP validation	78.44	17.96	1.80	1.80	91.97
After dot restoration	57.19	16.77	4.79	20.96	80.88

The average word segmentation rate (WSR) for the validation dataset in HMM aligner mode is 95%. The average character segmentation rate (CSR) is 97%. The average WSR for the test dataset in HMM aligner mode is 74.4%. The average CSR is 87%. The PSP validation stage is mostly adjusting the segmentation points locations more than eliminating them because these points proposed by HMM are much smarter than those proposed by heuristics used in literature (figure 6). The overall annotation automation results seem very promising regarding the fact of the difficulty of unconstrained Arabic handwriting processing and not using contextual help or other language resources in most of the job.

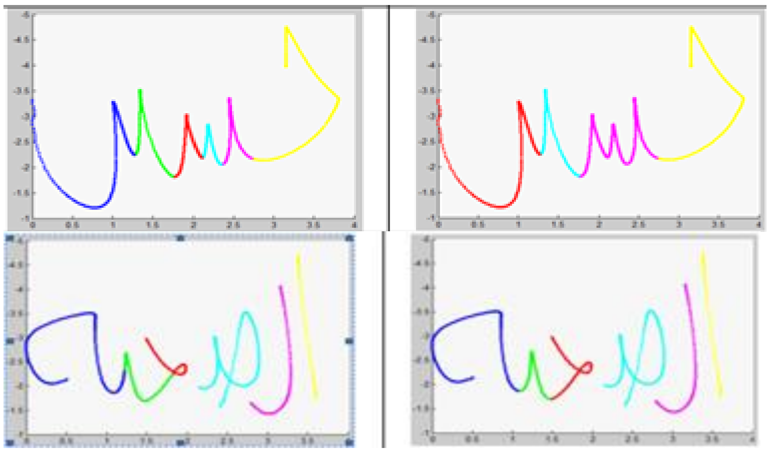


Figure 6: Examples of segmentation errors corrected by the validation rules

6. Status

The first version of the annotation tool has been implemented at Electronic Research Institute, Computers and Systems department, Egypt, and was released internally in Dec 2011. This first version is aimed at online document segmentation and annotation, and includes generic tools for data visualization, manipulation and annotation and sample data (OHASD dataset) and MATLAB code.

The included segmentation algorithms based on dynamic programming, RBF NN, SVM and HMM have been shown to perform in the range of 73-98% average accuracy on Arabic online handwritten text lines, words and characters. These algorithms cannot be benchmarked on standard datasets so far as we still don't have online datasets of Arabic scripts or even sentences.

In all the tool modules, it is clear that the algorithms employed offer a reasonable first level of annotation, additional methods together with other features and language resources are necessary to achieve higher accuracy and provide more utilities.

7. Evaluation

As declared before by Nagy et al. (2006) the most significant variable, which is responsible for most of the expense of human annotation, is human time. Even a 30% reduction in overall human time will be significant in an operational application. Thus, for evaluation, we computed the human time save using our tool compare to manual methods. A group of 12 volunteers not familiar to the annotation tool was asked to perform segmentation-annotation tasks to samples of the test dataset documents using our tool, first time using the tool manual option and the second time using automatic segmentation (and correcting errors if any). The automatic solution results came with average word annotation time of 16.18 sec. and average document annotation time 9.89 min. The manual solution results came with average word annotation time of 32.75 sec. and average document annotation time 16.20 min. The results show that our tool has achieved word & doc. time save in average of 51.5% and ~40% respectively.

8. Summary and Future plans

In summary, the tool effort aims to facilitate development of online handwritten annotated datasets for recognition engines for Arabic scripts. The first version of the tool provides robust implementations of tools, algorithms, scripts and sample code necessary to support the entire process starting from the text line extraction to the segmentation and annotation, for a particular set of shapes or characters. Most of the algorithms and tools are implemented using MATLAB. The annotation tool supports the tagging of digital ink with labels corresponding to ground truth, character names at different levels of an appropriate hierarchy of annotation.

However, our major focus at present is to validate the design and utility of the tool with different datasets. We are also interested in collaborative projects with university research groups using the tool. We hope that some of these users can contribute by trying to use the tool and providing feedback, while others may contribute to the tool by way of new utilities and algorithms. We also aim for the availability of large public test dataset for Arabic online hand writing as IAM-onDB (Liwicki et al. (2005)) for English to help benchmark our tool.

Furthermore, we aim at upgrading the tool to create a generic toolkit whose components can be used to build online handwriting recognition engines, and simplify integration of the resulting engines into real-world application contexts. We aim to add plug-in tools for handwriting data collection, standard algorithms for preprocessing and feature extraction, algorithms for training and pattern classification, and tools for subsequent evaluation and error analysis.

References

Balasubramanian A. (2006) Document Annotation and Retrieval Systems. Master Thesis. International Institute Of Information Technology. Hyderabad, India.

Bhaskarabhatla A.S., Madhvanath S. (2004) Representation and Annotation of Online Handwritten Data. International Workshop on Frontiers in Handwriting Recognition. pp. 136-141.

Bhattacharya, U., Banerjee, R. ; Baral, S. ; De, R. ; Parui, S.K. (2012) A Semi-automatic Annotation Scheme for Bangla Online Mixed Cursive Handwriting Samples. International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 680-685

Bhattacharya, N. ; Pal, U. (2012) Stroke Segmentation and Recognition from Bangla Online Handwritten Text. International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 740-745

Elanwar R.I., Rashwan M.A. & Mashali S.A. (2010) OHASD: The first online Arabic sentence database handwritten on tablet PC. International Conference on Signal and Image Processing (ICSIP 2010), Vol. 72, pp.710-715.

Elanwar R.I., Rashwan M.A. & Mashali S.A. (2011) On-Line Arabic Handwriting Text Line Detection Using Dynamic Programming. International Conference on Computer Mathematics and Natural Computing ICCMNC 2011, vol. 74, Malaysia, p. 588-593.

Elanwar R.I., Rashwan M.A. & Mashali S.A., (2012) Arabic online word extraction from handwritten text using SVM-RBF classifiers decision fusion. International Conference on Signal Processing, Robotics and Automation (ISPRA'12), Clare College, Cambridge University, England, pp. 68-73.

Elanwar R.I., Rashwan M.A. & Mashali S.A., (2012) Unconstrained Arabic online handwritten words segmentation using new HMM state design. International Conference on Computer Vision, Image and Signal Processing (ICCVISP 2012), Paris, France.

Jawahar C.V. & Kumar A. (2007) Content-level Annotation of Large Collection of Printed Document Images. International Conference of Document Analysis and Recognition (ICDAR 2007) pp. 799-803.

Kumar A., Balasubramanian A., Namboodiri A. & Jawahar C.V. (2006) Model-Based Annotation of Online Handwritten Datasets. International Workshop on Frontiers in Handwriting Recognition (IWFHR'06), October 23-26, La Baule, France.

Liwicki M., Bunke H. (2005) IAM-OnDB - an online English sentence database acquired from handwritten text on a whiteboard, In the Proceedings of 8th International Conference on Document Analysis and Recognition, 2:956–961.

Nagy G. & Lopresti D. (2006) Interactive Document Processing and Digital Libraries. Proceedings of the Second International Conference on Document Image Analysis for Libraries (DIAL'06)

Richarz, J.; Vajda, S. ; Fink, G.A. (2012) Annotating handwritten characters with minimal human involvement in a semi-supervised learning strategy. International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 23-28

Strassel S. M. (2009) Linguistic Resources for Arabic Handwriting Recognition. Proceedings of the Second International Conference for Arabic Handwriting Recognition

Yin F., Wang Q., Liu C. (2009) A Tool for Ground-Truthing Text Lines and Characters in Off-Line Handwritten Chinese Documents. 10th International Conference on Document Analysis and Recognition, pp. 951-955.