The University of Nottingham

School of Computer Science

# Offline printed Arabic character recognition

Ashraf AbdelRaouf

Thesis submitted to the University of Nottingham

for the degree of Doctor of Philosophy

May 2012

*To my family...*

# Abstract

Optical Character Recognition (OCR) shows great potential for rapid data entry, but has limited success when applied to the Arabic language. Normal OCR problems are compounded by the right-to-left nature of Arabic and because the script is largely connected. This research investigates current approaches to the Arabic character recognition problem and innovates a new approach.

The main work involves a Haar-Cascade Classifier (HCC) approach modified for the first time for Arabic character recognition. This technique eliminates the problematic steps in the pre-processing and recognition phases in additional to the character segmentation stage. A classifier was produced for each of the 61 Arabic glyphs that exist after the removal of diacritical marks. These 61 classifiers were trained and tested on an average of about 2,000 images each.

A Multi-Modal Arabic Corpus (MMAC) has also been developed to support this work. MMAC makes innovative use of the new concept of connected segments of Arabic words (PAWs) with and without diacritics marks. These new tokens have significance for linguistic as well as OCR research and applications and have been applied here in the post-processing phase.

A complete Arabic OCR application has been developed to manipulate the scanned images and extract a list of detected words. It consists of the HCC to extract glyphs, systems for parsing and correcting these glyphs and the MMAC to apply linguistic constrains. The HCC produces a recognition rate for Arabic glyphs of 87%. MMAC is based on 6 million words, is published on the web and has been applied and validated both in research and commercial use.

# Acknowledgements

First I would like to thank my main supervisor, Dr. Colin Higgins, for his participation throughout this research. His ideas and suggestions were of great benefit to me. I would also like to thank my supervisors, Dr. Tony Pridmore, Dr. Mahmoud Khalil, and Prof. Dave Elliman. I appreciate their guidance and comments on this thesis and for their honest support along my research.

Special appreciation to Prof. Graham Kendall whose help was very important at several times. He is also one of the best friends I got to know during my study time in Nottingham.

Special thanks to Essam Eliwa who is a very good friend. He made life much easier for me at Nottingham. It was a pleasure, not only to share the same lab with him over the last few years, but also to know him.

I would like to thank all my friends and colleagues at my home university, Misr International University (MIU). Many special thanks to my manager, Mr. Hassan ElRashidy, for his continues encouragement and support that I appreciate so much. His support was essential to help me pass many difficult times throughout my PhD study.

There are no adequate words to express my love and gratitude to my beloved wife, Meral, who has been by my side throughout the years. Supporting me, encouraging me and carrying more responsibilities during my study. She made me happy during difficult times. My sweet kids; Eraj, Nadine, Mona and Mohamed, I would like to tell you that all I do is for you and for a better future for you.

I am indebted and grateful to my whole family, especially to my father for his backing and advice that motivated me to study for a Ph.D. degree. I would also like to thank my mother for her continuous support. They both made me what I am today.

Thank you all, Ashraf AbdelRaouf

# Table of Contents

# List of Figures

# List of tables

# Chapter 1.

# Introduction

Computer processing of data has increased dramatically over time, to the extent that there are now massive demands for the rapid transfer of printed and handwritten information into computer readable format. Frequently this data must be accurately typed by human operators, which is time consuming and prone to error. This operation has recently been ameliorated by the invention of Optical Character Recognition (OCR) techniques. An OCR application reads written or printed data by recognizing the text at high speed. OCR applications are still quite limited in their use and can only recognize small amounts of the available data. Hence, more effort is needed in order to enable OCR applications to read printed and handwritten characters more quickly and accurately (Cheriet, Kharma et al. 2007). OCR applications for non-western characters, particularly partially connected scripts such as Arabic and Persian, are still relatively weak. The work reported here is concerned with the OCR of machine printed Arabic text.

*Why do we need OCR generally?* OCR is an essential technique that enables the transfer of printed documents to a computer readable form. OCR allows for the automated archiving and retrieving of old documents, especially old books. OCR has various applications at this time and still has great potential, as it offers a variety of applications for mail sorting (Govindan and Shivaprasad 1990), bank cheque processing (Senior 1992) and reading machines for the blind (Abdelazim 2006) among other uses. OCR research studies have a great influence on pattern recognition applications, for example; face recognition, fingerprint recognition and iris recognition. Such applications are used for some security issues such as criminal tracking. Recently, some systems have integrated OCR with new research topics such as automatic translation and voice commands. These systems would play an important role in developing such new topics (Chang, Chen et al. 2009).

*Why do we need Arabic OCR specifically?* Arabic OCR lacks research studies when compared to Latin character OCR. The recognition accuracy of Latin character OCR is far beyond that of Arabic OCR because the Arabic language is different in its cursive script and its letter shape, which is context sensitive (Khorsheed 2002). The population of those using the Arabic language is 350 million speakers (Statistics 2010) and it is the fourth most spoken language in the world after Chinese, English and Spanish (Education 2009). This indicates the need for more research studies in Arabic OCR. An improved and more powerful approach for Arabic OCR would be useful for the continuity of the language and in order to keep its history recorded.

## 1.1 Background

Pattern recognition classifies data based on knowledge extracted from known patterns. Pattern recognition is concerned with defined pattern domains (such as: character, iris, fingerprint, voice, face) and extracts some features from this pattern (either geometrical or structural) and then applies a classification algorithm in order to identify the pattern based on previously known features.

Pattern recognition is important in many applications and fields, for example; speech recognition for customer authentication, voice commands to computers and for converting speech to text (Cheriet, Kharma et al. 2007); for the role it takes in understanding and recognising engineering drawings (Elliman 2001); for medical and bioinformatics applications such as the understanding of the electrocardiogram (ECG) (Jafari, Noshadi et al. 2006); for human identification, such as face recognition (Bowyer, Chang et al. 2005; Tan, Chen et al. 2006; Mian, Bennamoun et al. 2007), iris recognition (Bowyer, Hollingsworth et al. 2008; He, Tan et al. 2009; Kumar and Passi 2010) and fingerprint recognition (O'Gorman 1998; Maltoni and Cappelli 2009); for research and industrial use of robotics (Bradski and Kaehler 2008) and for text understanding in character recognition.

Character recognition is the transformation of text images, generally captured by a scanner, into machine-editable text. This process is known as Optical Character Recognition (OCR). The objective of OCR is to emulate the human capability of accurately reading a document but at a higher speed.

OCR is currently very important and has many applications. One of the main uses of OCR is for office automation and document archiving as some companies tend to replace paper archiving for electronic archiving (Khorsheed 2002). Normally they keep two electronic copies of the same old paper document; an image for archiving and a text document for further use. Another usage of OCR is obvious in banks as they utilize the offline handwritten recognition applications with cheques in order to recognize names and values written on the cheques. Also, mail sorters use the same type of applications in order to recognize the names and addresses from the mail (Govindan and Shivaprasad 1990).

### 1.1.1 Motivation

The Arabic language has been used since the fifth century when written forms were stimulated by Islam. It is the native language of 350 million speakers (Statistics 2010) and it is the fourth most spoken language in the world after Chinese, English and Spanish (Education 2009). It is one of the six main languages of the United Nations (beside Chinese, English, French, Russian and Spanish) (Nations 2007). It has been estimated to be one of the ten most used languages on the Internet (Statistics 2010) and is the official language of 22 countries located in the geographical area of the Middle East (UCLA 2006).

The Arabic language is a particularly important language for all Muslims around the world. There are 1.5 billion Muslims, representing around a quarter of the world’s population (worldwide 2009). The Arabic language is the original language of the holy Quran and, although the holy Quran is translated into many other languages, Muslims use the Arabic language in their religious rituals.

Research studies regarding OCR have been widely developed during the last five decades in common languages like English, Chinese and Japanese (Cheriet, Kharma et al. 2007). These research studies concentrated mainly on Latin based languages, while Arabic based languages tend to be neglected (Harty and Ghaddar 2004; Cheriet, Kharma et al. 2007; Lorigo and Govindaraju May,  2006). Although there has been a recognizable increase in Arabic OCR research studies during the last decade, there is still a need for more (Abdelazim 2006). Specific features of the Arabic language (shown in section 2.2) create uniqueness in the necessary OCR applications. It is this uniqueness that increases the demand for more research regarding Arabic language recognition (Hamada 2007).

Language engineering is the use of computer technologies for the creation, archiving, processing and retrieval of machine processed language data and is a common research topic involving computer science and linguistics (Maynard, Tablan et al. 2002). The research studies of language engineering in the Arabic language are very limited (AbdelRaouf, Higgins et al. 2010). So, for instance, the Arabic language lacks a robust Arabic corpus. The creation of a well established Arabic corpus encourages Arabic language research and enhances the development of Arabic OCR applications. Therefore, it is essential for the corpus to be flexible regarding its continuous updating of size and contents in such a way as to meet the innovative needs of research.

### 1.1.2 Scope

The scope of this research is to introduce an innovative approach for offline printed Arabic recognition, while the wider scope is the machine recognition of readable information. Figure 1.1 shows the different categories of machine recognition of readable information. The two main types of readable information are "offline (OCR)" or "online". The online character recognition recognizes the dynamic handwriting motion during typing which is widely used in tablet PCs, handheld PDAs, smart phones and tablet devices (Plamondon and Srihari 2000). The online readable information is not included in this research. The offline readable information (OCR) is either machine printed or handwritten (Abdelazim 2006). Handwritten OCR uses a computer application in order to translate scanned paper documents written by hand into electronic word documents (Lorigo and Govindaraju May, 2006). Handwritten OCR is outside of the scope of this research.

The machine printed OCR contains different categories (Govindan and Shivaprasad 1990); Firstly, cursive / partially cursive - such as the Arabic OCR - which includes character segmentation-free recognition, as in the case of this research. It also includes the recognition of isolated characters only. Moreover, it includes recognition by segmenting the word into its characters or primitives. It can recognize the whole word without character segmentation (Khorsheed 2002; Abdelazim 2006).

Secondly, separated characters; for example, in the case of Latin character based languages, it recognizes alphabets, numerals and symbols. Thirdly, the mathematical formula recognizes symbols, numerals and so on. Fourthly, the ideogram languages such as Chinese and Japanese are totally symbolic languages. This type is out of the scope of this research. Finally, other types which include; for example, the Indian, Russian and Greek languages (see section 2.1.2).

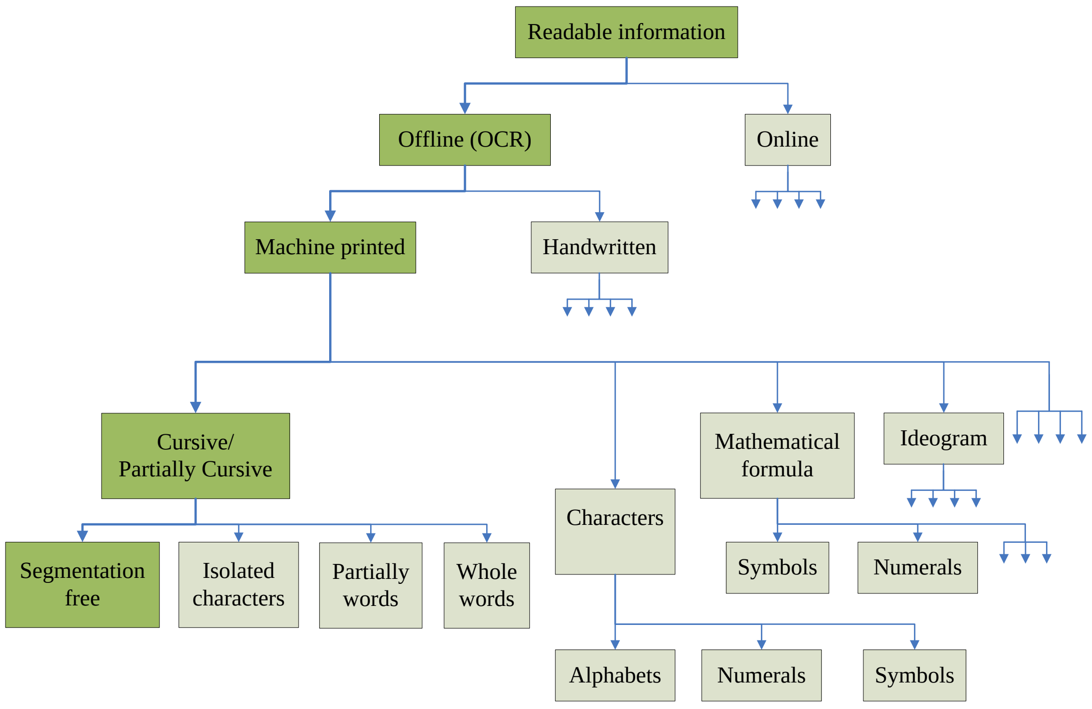


Figure 1.1: The machine recognition categories of readable information

The established approach of solving the Arabic OCR problem was investigated in order to achieve the scope of this research. An OCR application normally consists of three main parts: pre-processing, recognition and post-processing as shown in Figure 1.2 (Khorsheed 2002; Cheriet, Kharma et al. 2007; Lorigo and Govindaraju May, 2006). The pre-processing part is responsible for preparing the document image for the recognition part. The more successful the pre-processing algorithms are, the better the recognition rate of the application. The recognition part handles the document image after line finding. In turn, the recognition part outputs the machine readable recognized form of a document image. Finally, the post-processing part receives the series of recognized letters / words and it indicates which possible words are best in this situation, based on linguistic or statistical data.

The pre-processing phase cleans the image using image processing techniques. It converts the image to a more concise representation. The pre-processing part normally includes the following steps (Senior 1992; Lorigo and Govindaraju May, 2006):

1. Page layout analysis is responsible for analysing the contents of the document image. It defines the blocks of data inside the image whether text or non-text (Cheriet, Kharma et al. 2007).

2. Document decomposition separates the document image into its main components (paragraphs or non-text). This step is considered to be a preliminary step of dealing with the document image before the pre-processing stages that enhance the document image (Cheriet, Kharma et al. 2007).

3. Binarization is responsible for receiving the scanned image and converting the document from a grey-scale image to a binary, black and white image. Noise removal removes small erroneous regions and pixels from the binarized image.

4. A thinning step is normally used with handwritten input and is sometimes used with printed input. This step removes the stroke width and leaves a skeleton stroke with a width of one pixel.

5. Slant and skew detection and correction detect the slant and skew in the document image and correct them. Slant is the sheer inclining of the word from a horizontal line as in *italic* words. Skew is the rotation of the whole document image which results from the rotation of the document image while scanning.

6. Line finding identifies the baselines of the text lines in the paragraph of text. The baseline is the horizontal line that passes through the connected primitives of the line of text (Khorsheed 2002) or just under the non-descending letters for non-connected text.

The recognition part normally includes the following steps:

1. Lines and words segmentation separate the image of a paragraph of text to its images of text lines. Separate the line of text to its images of words. It ends up with the images of separate words inside the paragraph.

2. Word normalization scales the word image to fit a required standard height for the training process.

3. Feature detection and extraction defines the type of features that can be extracted from the document images and extracts these features during the training process of the application.

4. Character segmentation segments the word into its letters or sub-letters (primitives) contents.

5. Classification & recognition: classifies the letters of the words based on the features extracted in the previous step. This step ends with defining the letters of the words or all the words.

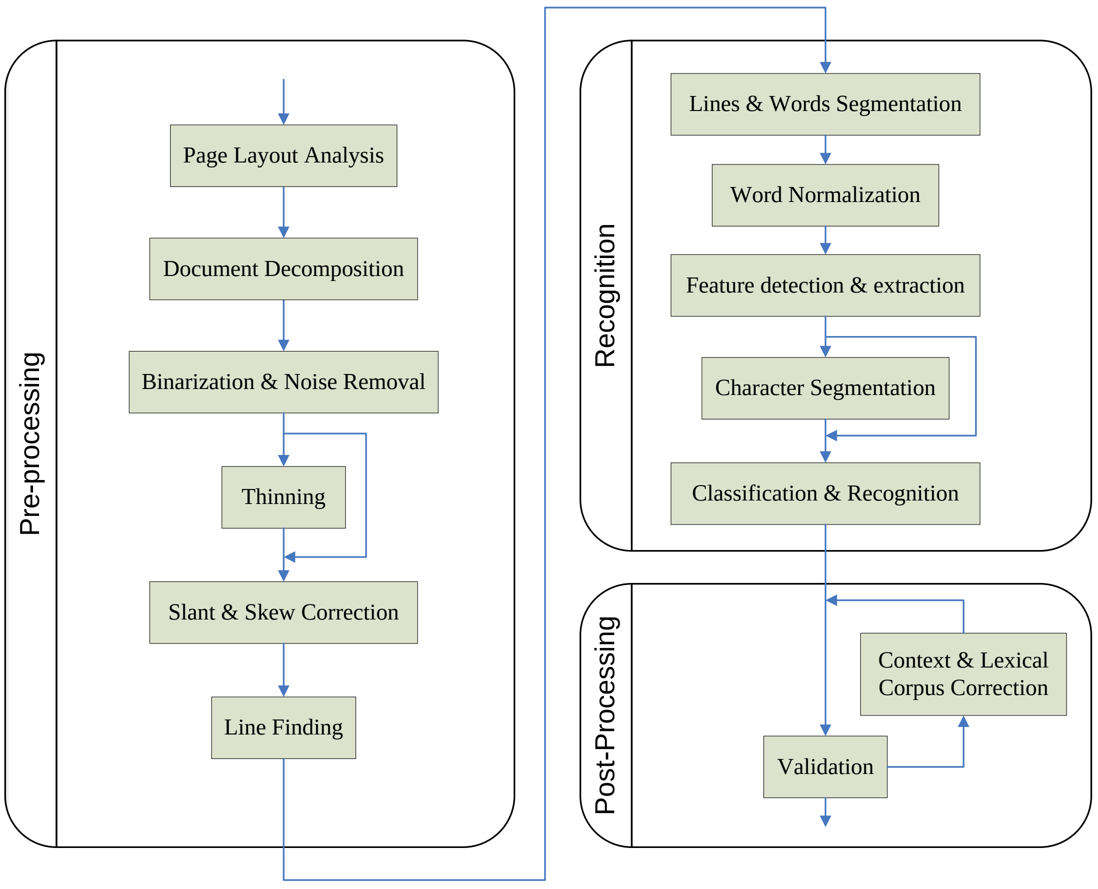


Figure 1.2: A general framework for character recognition

The post-processing part includes the following two steps:

1. Validation checks the word resulting from the classification and recognition step from a statistical, lexicographic or grammatical point of view, deciding whether it is the best suite word or not.

2. Context and lexical corpus correction step supplies the OCR application with suggested likelihood alternative words, driven from the corpus context.

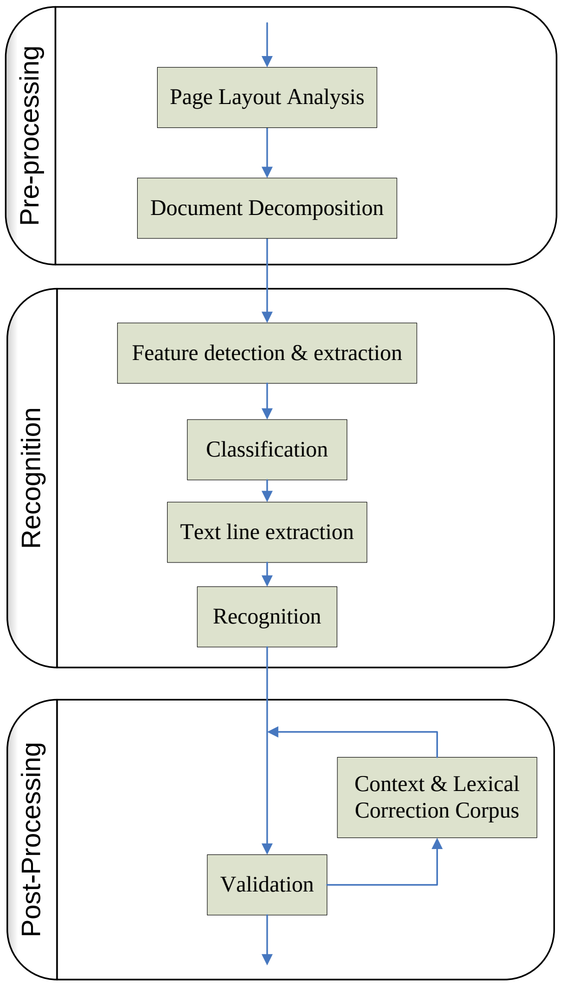


Figure 1.3: The framework of the proposed Arabic OCR approach

Figure 1.3 shows the framework of the proposed approach which defines the scope of this research. It is clear from the proposed approach that some of the pre-processing and recognition steps (Binarization & Noise Removal, Thinning, Slant & Skew Correction, Line Finding, Lines & Words Segmentation and Word

Normalization) are eliminated and the character segmentation stage is also totally eliminated. This indicates that a lot of steps might not be needed for the recognition of Arabic OCR.

The scope of this research is wide enough to include other languages that use Arabic alphabets, for example; Pashto, Persian, Sindhi, and Urdu. All languages that use Arabic alphabets have common features such as partial cursive script, context sensitivity of letter shapes and reading direction (Khorsheed 2002). These common features facilitate the opportunity to share research studies.

## 1.2 Brief Overview

This section introduces the general aims and objectives of this research with a detailed definition of the problems and specific objectives. The approach followed in solving the problem is defined. The contributions of this research are explained briefly.

### 1.2.1 General aims and objectives

The aim of this research is to investigate, suggest, design and analyze a set of OCR techniques in order to significantly increase the recognition accuracy and speed of offline printed Arabic character recognition. A number of practical questions relating to Arabic OCR applications set the foundation and objectives for this research:

1. *What are the advantages and disadvantages of the established way of solving the problem of printed Arabic OCR?*

2. *To what extent are the steps in the pre-processing and recognition stages essential and can these steps be eliminated totally or partially?*

3. *Can Arabic printed text be recognized without the character segmentation phase?*

4. *If some steps from the pre-processing and recognition stages in additional to character segmentation are eliminated, then can a complete Arabic OCR application be developed?*

5. *Can the connected segments or pieces of Arabic words (PAWs) as well as naked pieces of Arabic words (NPAWs); PAWs without diacritical marks shown in section 2.2 (AbdelRaouf, Higgins et al. 2008) be considered as the smallest Arabic tokens to replace words and naked words in Arabic OCR applications?*

6. *Are the available Arabic corpora sufficient to cover Arabic OCR research?*

These represent the research questions that were tackled during the development of this research. Acquiring answers to these questions is the final objective of this research.

### 1.2.2 Problems and specific objectives

Three major problem areas need to be considered in order to answer the set of questions presented in section 1.2.1.

The first problem area is concerned with the pre-processing and recognition stages, which affects the recognition accuracy according to the quality of the document image (Khorsheed 2002). All the steps of the pre-processing and recognition stages have many algorithms and different approaches available. Sometimes one approach achieves good results with a document image and fails with another (Baird 1987). The variety of approaches in each step sometimes misleads the researchers. These stages are considered to be one of the reasons for the slowness in the character recognition time (Cheriet, Kharma et al. 2007). Initially, this research intended to reduce the pre-processing and recognition steps as much as possible for the sake of enhancing the recognition accuracy and speed.

The second problem area is concerned with the character segmentation phase. It is one of the reasons for reducing recognition speed and accuracy. It is clear that the character segmentation phase is always the major bottle neck in Arabic OCR research (Abdelazim 2006). Although there are some research studies trying to achieve character segmentation-free Arabic OCR, this is still limited to a certain number of words and fonts (Abdelazim 2006). The current research aims to eliminate the character segmentation phase in the recognition of printed Arabic text. The character segmentation problem was tackled by attempting a completely new approach that can detect the glyph image from the document image without any character segmentation.

The third problem area is concerned with Arabic corpora. A corpus is a large structured set of texts, electronically sorted and processed. Corpora have become very important in the study of languages and they are the key to the development of OCR applications. Excellent corpora have been developed for Latin based languages, but not for the Arabic language. Access to a corpus of both language and images is essential during OCR development, particularly while training and testing a recognition application (AbdelRaouf, Higgins et al. 2010). The aim of constructing and providing a comprehensive study and analysis of a multi-modal Arabic corpus was essential for this research and for all the research studies of printed Arabic OCR.

These represent the research problems that were tackled during the development of this research. Acquiring a solution to these problems is the final target of the research.

### 1.2.3 Approach

The previous two sections (1.2.1, 1.2.2) describe the aims and objectives of this research. In order to achieve these aims, this research investigated a novel approach that disregarded steps from the pre-processing and recognition stages and recognized the glyphs without a character segmentation phase as shown in Figure 1.3. This novel approach achieved at the end a full Arabic OCR application using the proposed research objectives.

The research studies and applications of Arabic OCR were investigated in order to identify the advantages and disadvantages of the established way of solving the Arabic OCR problem as shown in Figure 1.2.

A multi-modal Arabic corpus (MMAC) was constructed in order to facilitate the generation of Arabic OCR applications. Unlike other corpora, MMAC also includes tokens such as PAWs and NPAWs that are beneficial to OCR research (AbdelRaouf, Higgins et al. 2010). A lexicon / stemming algorithm had been developed for providing a list of alternatives for uncertain words to improve the performance of the corpus. The lexicon / stemming algorithm also assures the continuous updating of the size and contents of the corpus, which copes with the innovative needs of Arabic OCR research.

This research depended on utilising an already existing approach that had been used and tested regarding face detection applications (Viola and Jones 2001). This approach is applied on character recognition using a cascade of boosted classifiers based on Haar-like features. The approach used the rotated Haar-like wavelet feature extraction algorithm and the cascade of boosted classification algorithm. The feature extraction algorithm obtains rectangular regions of the image and sums up the pixels in that region. This summing up is used in order to categorize images. Thus, it could categorize all images having Haar-like features in a certain range of values as one category and those deviating out of this range in another category. This procedure could be recursively carried out by dividing the image clusters (Viola and Jones 2001).

The boosting classification is a powerful machine learning technique. It merges the performance of many "weak" or base classifiers in order to construct a powerful one whose performance is significantly better than that of any of its base classifiers (Bishop 2006). A “weak” or base classifier only needs to perform better than chance, and thus can be straightforward and computationally inexpensive (Lienhart, Kuranov et al. 2002). This Haar Cascade Classifier (HCC) approach will be explained later in Chapter 4. The HCC approach has been selected for the following reasons:

1. It can be used with grey-scale images, so there is no need to apply binarization and noise removal algorithms that convert a document from a grey-scale to a black and white image. Nor is there a need to remove noise.

2. It uses a document image without slant and skew corrections as the rotated Haar-like features accept rotated images up to ±45°.

3. It handles a document image without segmenting each word to its letters or primitives content; which means that it can pick out any glyph image from inside the document image without character segmentation. This novel approach represents a real character segmentation free approach.

4. It manages different font sizes. The approach starts detecting the glyph from the width and height given in the training parameter up to the size of the whole document image, hence there is no need for any word normalization.

5. It can be extended to any cursive handwritten language not only the Arabic language. (Using this approach on handwriting could avoid the thinning step in the pre-processing stage.)

6. This approach can be applied to create a Portable Document Format (pdf) file upon extracting the position of the recognized glyph. This could be used with text from any scanned Arabic document.

A simple approach was applied to implement an HCC classifier and recognizer that was built based on the classifiers generated from the HCC glyphs classifiers. The confusion matrix was applied in order to enhance the recognition accuracy of the HCC approach. A simple post-processing technique was finally applied in order to enhance the recognition rate of the HCC approach.

### 1.2.4 Contributions

The main contribution of this research is to develop an Arabic OCR application completely without character segmentation and pre-processing. This research contributes to the research area of Arabic OCR in the following aspects:

1. A multi-modal Arabic corpus (MMAC) was collected, generated, verified, tested and justified. MMAC was tested with a commercial Arabic OCR application and provided variety of results which proof its usability. The MMAC was used by a commercial company to make a comparison between different Arabic OCR applications. The corpus is complemented by a newly developed lexicon / stemming algorithm in order to improve the performance of MMAC. MMAC has an application that allows adding and verifying any new lists of words to the corpus. (see Chapter 3)

2. A new approach to solving the problem of Arabic OCR was followed. This approach works without steps from the pre-processing and recognition phases to enhance recognition accuracy and time. (see Chapter 4)

3. Other research studies have attempted to use character segmentation-free recognition (see section 2.5.5), but these attempts failed either to present a concrete application or to achieve reasonable results (Abdelazim 2006). This research followed a new approach in order to achieve true character segmentation-free recognition depending on Haar-like features, which use the visual object detection algorithm. This approach led to a true character segmentation-free recognition application. (see Chapter 4)

4. A complete Arabic OCR application has been successfully created using a Haar Cascade Classifier (HCC) approach. The application has similar functionality to, and has been tested against, another Arabic OCR commercial application. (see Chapter 5 and Chapter 6)

5. The HCC approach is for the first time to be applied with character recognition and specifically Arabic character recognition. Although this approach was originally developed for face detection, it is usually used in face detection, eye detection, pedestrian detection and even in bowls detection (Adolf 2003). (see Chapter 4)

## 1.3 Organization of the thesis

This chapter presented the background information, including motivation and scope of this research. A brief overview was introduced to include general aims and objectives of this research. It introduced the three main problem areas, sets out the general objectives, explains the approach and highlights the contributions.

Chapter 2 introduces key concepts in the area of printed character recognition with the focus on the Arabic language. It presents an overview of OCR history and research topics related to OCR. It also presents different categories of OCR applications and OCR commercial applications. It provides an overview of the written Arabic language and its recognition features. It highlights and considers a detailed survey study of different phases of established OCR applications. It demonstrate the survey study of the new approaches applied in this research.

Chapter 3 reports the construction of a Multi-Modal Arabic Corpus (MMAC). The MMAC is generated from both text and images of existing documents. A ground truth annotation is offered for corpus images. Statistical analysis and verification of the corpus has been carried out and is presented. Moreover, the MMAC was tested using commercial OCR software. A new lexicon / stemming algorithm is proposed in order to enhance corpus information retrieval. A tool is added to continually improve the corpus by adding new words and justifying them using the lexicon / stemming algorithm.

Chapter 4 describes the approach of the Haar-Cascade Classifier (HCC), with an explanation of how to apply the HCC approach to Arabic character recognition. The relationship between Arabic glyphs and faces is explained. The usefulness of the approach to Arabic character recognition is also indicated. It shows the steps and experiments that were followed in order to justify that the HCC approach is appropriate for Arabic character recognition applications. An experiment was introduced to include all the Arabic glyphs in the HCC approach and its testing results and conclusions are presented.

Chapter 5 presents an HCC tokens recognition. It explains the generation of testing dataset for testing the next steps. It describes the generation of a single tokens classifier from the different classifiers that were produced in Chapter 4. The generation of the tokens recognizer is given with its different components, including the testing process. It describes the generation and applying of a confusion matrix with an overview of how it can enhance the recognition rate of the OCR application. The testing of the HCC approach after applying the confusion matrix was presented against commercial software.

Chapter 6 describes the application of a post-processing technique to the HCC approach. A comparison between naked words and naked PAWs is presented. Applying trie structure in searching inside the look-up dictionaries of the two token types (Nwords and NPAWs) is presented. Implementing the Levenshtein distance algorithm in order to enhance the recognition rate of the HCC approach is explained using the two different token types. Testing the post-processing technique with the two token types is introduced with the results and conclusion of the testing.

Chapter 7 presents the conclusions and suggestions for future work of the research. How the objective of the research was met is explained. The future work that can enhance the final results of the research is introduced and explained for each part of the thesis. The conclusion of the research is explained briefly. Closing remarks of the thesis are presented.

# Chapter 2.

# A review of character recognition

This chapter introduces a survey across OCR research studies in general. The survey includes OCR applications and related research. It provides an overview of the written Arabic language and discusses the problems occurring to the developer of an OCR application. It introduces the main stages involved in developing any OCR application. The main focus of this research is Arabic printed OCR, therefore a large proportion of the survey presented was centred on this. A survey regarding commercial OCR applications is also presented.

OCR is the automatic reading of text which is then saved onto the computer. The text can be in any format or language. Normally, the OCR application goes through many phases to generate the overall application where each phase includes different steps. Each step can usually apply one of many different algorithms. The importance of the steps differs from one OCR classification type to another. Figure 1.2 shows the phases and steps of an OCR application. A survey study is introduced in this chapter containing all the OCR phases with their normal steps.

## 2.1 Introduction

### 2.1.1 Overview

The history of OCR demonstrates the progress made in the field from the early beginnings of this research topic. Moreover it tells the enhancements made in the recognition rate and algorithms over time. Research topics related to OCR gives more information regarding the approaches and techniques used. These approaches may be of benefit to the development of OCR.

#### *2.1.1.1 History of OCR*

The history of OCR explains how it was developed throughout early research from the first OCR device to the current improved OCR applications. Interestingly OCR began as a hardware device. The problem at that time was finding a way to enable the computer to scan document images.

The origins of OCR were founded in early 1870 as an aid to the visually handicapped, and the first successful version was created by the Russian scientist Tyurin in 1900 (Govindan and Shivaprasad 1990). In 1928, Gustav Tauschek of Austria had a patent of a basic OCR "reading machine". It was based on using light-detecting photocells to recognize patterns on paper or card (Woodford 2010).

The photomultiplier machine was invented in the early 1950s. The machine captured the images of characters and text by mechanical and optical means of rotating disks and photomultipliers. It was the first real recorded OCR application. The development of OCR applications started to show real improvement beginning with the invention of scanner machines (Cheriet, Kharma et al. 2007).

The modern computerized version of OCR appeared in the mid 1950s. Following this, the flatbed scanner was invented which was capable of high speed scanning. These important developments accelerated the speed of character recognition, reduced the cost and allowed a wide variety of forms and documents to be scanned (Govindan and Shivaprasad 1990). Since then extensive research has been carried out and a large number of research studies have been published by various researchers in the area of character recognition (Govindan and Shivaprasad 1990). At that time, the motive for developing OCR applications was governed by the need to process an enormous amount of documents such as bank checks, government records, credit card imprints and for mail sorting (Cheriet, Kharma et al. 2007).

#### *2.1.1.2 Research topics related to OCR*

OCR research topics are part of many other broad research areas. This section serves to explore the relationship between OCR and those areas. These broad research areas are artificial intelligence, machine learning, computer vision, digital image processing, pattern recognition and natural language processing.

*Artificial intelligence* facilitates computers to carry out tasks which are usually done by people, thus, enabling computers to recognize documents rather than people is considered to be an artificial intelligence approach (Senior 1992).

*Machine learning* is concerned with the design and development of algorithms that allow computers to evolve behaviours based on empirical data, such as databases (Bishop 2006). The feature detection and extraction phase is considered to be a machine learning methodology.

*Computer vision* aims to duplicate the effort of human vision by electronically perceiving and understanding an image. The machine extracts important information from the image which enables the solving of some tasks in order to understand the image (Sonka, Hlavac et al. 1998). The pre-processing stage includes computer vision algorithms (Abdelazim 2006).

*Digital image processing* is the processing of digital images by digital computers (Gonzalez and Woods 2007). The digital image is composed of a fixed number of elements. Each element is a pixel on the image and has a location and value. Most of the steps of OCR handle the document image as a digital image and use different algorithms - usually applied in image processing.

*Pattern recognition* is concerned with defined pattern domains (such as iris, fingerprint, voice and face) and extracts some features from this pattern (either geometrical or structural) and then applies a classification algorithm to identify the pattern based on previous known examples. OCR is a branch of pattern recognition and the domain, in the case of OCR, is the text (Khorsheed 2002).

*Natural language processing (NLP)* is a common research topic involving computer science and linguistics. It is concerned with the computers’ processing of human (natural) languages (Bates 1995). This computer processing is generated using language engineering algorithms (Maynard, Tablan et al. 2002). OCR uses language engineering in generating and exploiting corpora.

## 2.1.2 Different categories of readable information

Different categories of readable information explain the main features of machine recognition as shown in Figure 1.1. Readable information can be categorized based on either online versus offline (OCR) or on being machine printed versus handwritten. The latter categorization is based on the type of language used.

### *2.1.2.1 Online versus offline classifications*

Online character recognition is an application that saves input at the time of writing via direct input from the user stylus. Examples of this application include the tablet PC, smart phones and tablet devices. This category depends mainly on the direction of the movement of the stylus during writing in order for the application to recognize the characters in real time (Plamondon and Srihari 2000; Abdelazim 2006). The feature extraction and classification phases in this category use some definite algorithms in text recognition.

Offline character recognition is the transformation of text images, generally captured by a scanner, into machine-editable text. These inputted text images can be machine printed or handwritten (Senior 1992; Lorigo and Govindaraju May, 2006). Handwritten OCR is the most sophisticated category and usually has a low recognition rate (Abdelazim 2006). Machine printed OCR for non-cursive languages is the most straightforward category and achieves a high recognition accuracy (Senior 1992).

### *2.1.2.2 Machine printed versus handwritten categories*

Machine printed OCR operates on characters created using a typewriter or computer printer. Machine print is usually consistent in the size of the characters and the recognition rate is extremely high. The shapes of a single character are extremely limited compared to handwriting. The distance between characters is uniform and the distance between words is also consistent. The baseline is uniform and can be picked out easily. Non-cursive languages (e.g. Latin based) need no character segmentation phase for recognition (Khorsheed 2002; Abdelazim 2006).

Handwritten OCR is more complicated than machine printed. The thinning step in the pre-processing phase is more important than in the case of machine print. The shapes of a single character can vary hugely and the application is continually required to learn these shapes. The inter-distance between letters and words is also inconsistent and the baseline detection process requires a complicated algorithm. In addition to this, a complicated character segmentation algorithm is required to segment the characters from the words. The post-processing phase is extremely important. Hence, the influence of prediction is higher in this category (Senior 1992; Lorigo and Govindaraju May, 2006).

#### *2.1.2.3 Language based category*

OCR is categorized based on the languages' alphabets. The Latin based languages (for example English, French and German) can be included in one category which is based on the Latin characters' alphabets. The Far Eastern (Ideogram) languages (for example Chinese and Japanese) can be included in another category (Govindan and Shivaprasad 1990). The South Eastern Asian languages (such as Indian, Tamil and Tai) can also be included in another category (Shanthi and Duraiswamy 2010). The last category is the Middle Eastern languages which are based on Arabic alphabets, Hebrew can be included in this category (Govindan and Shivaprasad 1990).

Each language category has its own different properties. Latin based languages is the simplest category, it needs no character segmentation for recognition. The Far Eastern category includes around 5,000 characters which is why two levels are used to classify the characters; one to group them and the second to classify the character (Govindan and Shivaprasad 1990). The South Eastern Asian category uses also two levels of classification. The number of characters are less than that for the Far East category and are cursive characters (Shanthi and Duraiswamy 2010). The Middle Eastern category has cursive letters and is read from right to left. Normally it uses character segmentation algorithms for recognition.

### 2.1.3 OCR Applications

There are many commercial OCR applications on the market with different prices and degrees of accuracy. These applications are commonly used with the English language. The character recognition accuracy, page layout reconstruction, support for languages, support for PDF output, speed and user interface are the main features that differentiate between OCR applications (Simple Software 2012).

From varies OCR applications on the market, it was settled for ABBYY FineReader 11 (ABBYY 2012), OmniPage Professional 18 (ScanStore 2011) and Readiris Pro 12 (IRIS 2011) to be the best OCR applications on the market (World 2010; Network 2012; Simple Software 2012). NovoVerus from Novo Dynamics (Dynamics 2012) and OCR from Sakhr software (Software 2011) were first known Arabic OCR applications.

#### *2.1.3.1 ABBYY FineReader 11 - ABBYY*

ABBYY is one of the leader companies in optical character recognition (OCR), document processing, linguistic and translation technologies. The ABBYY FineReader 11 application is the latest OCR version produced by the AABBYY Company (ABBYY 2012).

ABBYY FineReader 11 version enhances the speed and accuracy of recognition. It creates editable and searchable files from scans. Productivity enhancement include one-click wizard which enable doing more tasks with fewer steps. Supports multi-language documents with 189 supported language including Arabic (ABBYY 2012).

#### *2.1.3.2 OmniPage Professional 18 - Nuance*

Nuance is one of the leader world companies from speech technologies, to healthcare solutions, to imaging technologies that convert physical documents into searchable digital files. The OmniPage Professional 18 application is the latest OCR application produced by the Nuance Company (ScanStore 2011).

OmniPage 18 is fast and precise application that deals with paper, PDF files, and digital camera pictures. OmniPage 18 ensures that the converted document is look like the original with its text, graphics, columns, and tables. It offers many advanced features, including Digital Camera 3DC technology that automatically corrects any distortions of images. Automatically detects Asian languages, which suits documents contain mixed languages, but not include Arabic language (ScanStore 2011).

#### *2.1.3.3 Readiris Pro 12 - IRIS*

Readiris Pro is manufactured by a large worldwide global company called I.R.I.S. started in 1987 and specialized in OCR applications. The company manufactures a number of other optical character recognition products including pen scanners, card scanners, mobile scanners and a digital pen (Network 2012).

Readiris Pro 12 is a competitive OCR package with a number of features including table recognition, handwritten recognition and picture enhancement tools. However, it is in the middle of the recognition accuracy. Readiris pro12 recognizes more than 120 different languages. Dedicated versions for Asian, Hebrew and Arabic characters are also available (IRIS 2011).

#### *2.1.3.4 NovoVerus – Novo Dynamics*

NovoDynamics had been established since two decades and is specialized in the OCR applications. It provides fast and accurate 64-bit, multi-core, cross-platform document capture applications. NovoDynamics delivers intelligent applications that add clarity and insight to help customers (Dynamics 2012).

NovoVerus automatically detect and extract different languages. In addition, to apply image enhancement to damaged pages, poor photocopies and low quality documents. NovoVerus supports Roman, Asian, Cyrillic and Middle Eastern languages. Convenient user interface features facilitate zone and text editing prior to processing with machine translation systems, which can easily be integrated (Dynamics 2012).

#### *2.1.3.5 OCR – Sakhr Software*

Sakhr was established in 1982 to support Arabic language for information technology. Sakhr has produced many industry first commercial products and solutions with outstanding accuracy and performance. U.S. government evaluators assess Sakhr as the best available Arabic OCR (Software 2011).

Sakhr OCR recognizes scanned Arabic text or handwriting with high accuracy. Supports Arabic, Farsi, Pashto, Jawi, and Urdu. It supports bilingual documents.

## 2.2 The written Arabic language

This section provides an overview of the written Arabic language and discusses the problems occurring to the developer of an OCR application. These problems also have an impact upon the design of successful corpora. The Unicode naming convention has been adopted here, though other schemes are in use (Consortium 2003; Unicode 2007).

### 2.2.1 Key features of written Arabic

Written Arabic is both rich and complex, features of note are:

- The Arabic language consists of 28 Arabic letters (Consortium 2003) and is written from right to left. Arabic script is cursive even when printed and Arabic letters are connected by the baseline of the word (see Figure 5.4). The Arabic language makes no distinction between capital and lower-case letters; it contains only one case.

- The widths of letters in Arabic are variable (for example س and ا).

- The connecting letter known as Tatweel or Kashida is used to adjust the left and right alignments; this letter has no meaning in the language, it does not exist at all in any semantic sense.

- Arabic alphabets depend on dots to differentiate between letters. There are 19 “joining groups” (Unicode 1991-2006). Each joining group contains more than one similar letter which differs in the number and place of the dots, as for example (ج ح خ) which have the same joining group (ح) but with differences in the place of dots. Table 2.1 lists the joining groups, their schematic names and their group letters. The Hamza (ء) is used as a joining group, although it is not included in Unicode. Table 2.2 shows the same joining groups but the letters are shown in their location in the word. The character in its different location is called a glyph.

- The Arabic language incorporates ligatures such as (Lam Alef لا) which actually consist of two letters (ل ا) but when connected produce another glyph. In some fonts like Traditional Arabic there are ligatures like (يـــ) which come from two letters (يمـ).

Table 2.1: Arabic joining groups and group letters, with their English name

| Schematic Name | Joining Group | Group Letters |
|---|---|---|
| ALEF | ا | آ أ إ ا |
| BEH | ب | ب ت ث |
| HAH | ح | ج ح خ |
| DAL | د | د ذ |
| REH | ر | ر ز |
| SEEN | س | س ش |
| SAD | ص | ص ض |
| TAH | ط | ط ظ |
| AIN | ع | ع غ |
| FEH | ف | ف |
| QAF | ق | ق |
| KAF | ك | ك |
| LAM | ل | ل |
| MEEM | م | م |
| NOON | ن | ن |
| HEH | هـ | هـ |
| WAW | و | و ؤ |
| YEH | ى | ئ ى ي |
| TEH MARBUTA | ه | ة ه |
| HAMZA | ء | ء |

- Arabic script can use diacritical marking above and below the letters such as ( عِلم عُلم) termed Harakat (Consortium 2003) to help in pronouncing the words and in indicating their meaning (Fahmy and Ali 2001). These diacritical markings are not considered in this research.

- The three letters Ain, Ghain and Heh (ع غ هـ) have four different glyph shapes according to their location in the word, while the rest of the Arabic letters have two different glyphs in different locations inside the PAW. This is shown in Table 2.3.

- An Arabic word may consist of one or more sub-words. We have termed these disconnected sub-words PAWs (Pieces of Arabic Word) (Amin 1997; Lorigo and Govindaraju May, 2006). For example (رسول) is an Arabic word with three PAWs (ل) (سو) (ر). The first and inner PAWs of the word must end with one of the six letters (ا د ذ ر ز و) as these are not left connected. Hamza is a separate PAW.

Table 2.2: Arabic joining groups and group letters, defined with letters location

| Characters | Isolated | Start | Middle | End |
|---|---|---|---|---|
| ا | ا | | | ـا |
| ب - ت – ث | ب ت ث | بـ تـ ثـ نـ يـ ئـ | ـبـ ـتـ ـثـ ـنـ ـيـ ـئـ | ـب ـت ـث |
| ج - ح – خ | ج ح خ | جـ حـ خـ | ـجـ ـحـ ـخـ | ـج ـح ـخ |
| د – ذ | د ذ | | | ـد ـذ |
| ر – ز | ر ز | | | ـر ـز |
| س – ش | س ش | سـ شـ | ـسـ ـشـ | ـس ـش |
| ص – ض | ص ض | صـ ضـ | ـصـ ـضـ | ـص ـض |
| ط – ظ | ط ظ | ط ظ | ـط ـظ | ـط ـظ |
| ع – غ | ع غ | عـ غـ | ـعـ ـغـ | ـع ـغ |
| ف | ف | فـ قـ | ـفـ ـقـ | ـف |
| ق | ق | | | ـق |
| ك | ك | كـ | ـكـ | ـك |
| ل | ل | لـ | ـلـ | ـل |
| م | م | مـ | ـمـ | ـم |
| ن | ن | | | ـن |
| ه | ه ة | هـ | ـهـ | ـه ـة |
| و – ؤ | و | | | ـو ـؤ |
| ى - ي – ئ | ى | | | ـى ـي ـئ |
| ء | ء | | | |
| لا | لا | | | ـلا |

Table 2.3: Arabic letters and their shapes at different locations within words

| English Name | Arabic Letter | Isolated | Start | Middle | End |
|---|---|---|---|---|---|
| ALEF | ا | ا | | | ـا |
| BEH | ب | ب | بـ | ـبـ | ـب |
| THE | ت | ت | تـ | ـتـ | ـت |
| THEH | ث | ث | ثـ | ـثـ | ـث |
| JEEM | ج | ج | جـ | ـجـ | ـج |
| HAH | ح | ح | حـ | ـحـ | ـح |
| KHAH | خ | خ | خـ | ـخـ | ـخ |
| DAL | د | د | | | ـد |
| THAL | ذ | ذ | | | ـذ |
| REH | ر | ر | | | ـر |
| ZAIN | ز | ز | | | ـز |
| SEEN | س | س | سـ | ـسـ | ـس |
| SHEEN | ش | ش | شـ | ـشـ | ـش |
| SAD | ص | ص | صـ | ـصـ | ـص |
| DAD | ض | ض | ضـ | ـضـ | ـض |
| TAH | ط | ط | طـ | ـطـ | ـط |
| ZAH | ظ | ظ | ظـ | ـظـ | ـظ |
| AIN | ع | ع | عـ | ـعـ | ـع |
| GHAIN | غ | غ | غـ | ـغـ | ـغ |
| FEH | ف | ف | فـ | ـفـ | ـف |
| QAF | ق | ق | قـ | ـقـ | ـق |
| KAF | ك | ك | كـ | ـكـ | ـك |
| LAM | ل | ل | لـ | ـلـ | ـل |
| MEEM | م | م | مـ | ـمـ | ـم |
| NOON | ن | ن | نـ | ـنـ | ـن |
| HEH | هـ | ه | هـ | ـهـ | ـه |
| WAW | و | و | | | ـو |
| YEH | ي | ي | يـ | ـيـ | ـي |

- Arabic letters have four different shapes according to their location in the word (Lorigo and Govindaraju May, 2006); Start, Middle, End and Isolated. For the six letters (ا د ذ ر ز و) there is no Start or Middle location shape. The letter following these six letters must be used in its Start or Isolated location shape. In the joining type defined by the Unicode standard all the Arabic letters are Dual Joining, except the previous six letters and the Teh Marbota (ة) which is joined from the right side only. The Hamza (ء) is not a joining letter. Table 2.3 shows the different shapes Arabic letters adopt in different locations and gives their English names.

- Fifteen Arabic letters have dots: 12 letters have dots above the baseline; while three have dots below it. 10 of the fifteen letters use one dot, three use two dots and two use three dots (Khorsheed 2002). Table 2.4 shows the dots different distribution among letters.

Table 2.4: Arabic letters with dots distribution

| Description | Arabic Letters |
|---|---|
| Letters with dots above baseline | ق ن ف غ ظ ض ش ز ذ خ ث ت |
| Letters with dots below baseline | ي ج ب |
| One dot Letters | ن ف غ ظ ض ز ذ ب ج خ |
| Two dots letters | ي ق ت |
| Three dots letters | ش ث |

## 2.2.2 Recognizing written Arabic

The Arabic language is not an easy language for automatic recognition. Some of the particular difficulties that must be faced by the developers of OCR applications are:

- Characters are cursive and not separated, as in the case of Latin script. Hence, recognition normally requires a sophisticated character segmentation algorithm.

- Characters change shape depending on their location in the word. The distinction between isolated characters is lost when they appear in the middle of a word.

- Sometimes Arabic writers neglect to include whitespace between words when the word ends with one of the six letters (ا د ذ ر ز و). This is known as the connected words problem (Buckwalter 2004). For example, (الأستاذاحمد) should be

(ألاستاذ احمد). On the other hand they sometimes separate a single word into two words when the word is long or is pronounced in two parts; (سبع مائه) is actually (سبعمائه) (Buckwalter 2004).

- Repeated characters are sometimes used, even if this breaks Arabic word rules. This is especially common in online “chat” sites; for example (مووووووت) is actually (موت).

- There are two ending letters (ي ى) which sometimes indicate the same meaning but are different characters. For example (الذي) and (الذى) have the same meaning, the first is correct but the second form is often encountered. The same problem exists with the character pair (ه ة).

- There is often misuse of the letter Alef (ا) in its different shapes (أ إ آ).

- The individual letter (و) which means "and" in English is often misused. It is a separate word and should have whitespace before and after it, but most of the time Arabic writers do not use the whitespace. This is a particularly common instance of the connected word problem.

- The Arabic language contains a number of similar letters like Alef (ا) and the number 1 (١), and also the full stop (.) and the Arabic number 0 (٠) (Harty and Ghaddar 2004).

- Arabic font files exist which define character shapes similar to the old form of Arabic writing. These fonts are totally different from the popular fonts. For example a statement with an Arabic Transparent font like (احمد يلعب في الحديقة) when written in the old shape font like Andalus becomes (احمد يلعب في الحديقة).

- It is common to find transliterations of English-based words, especially proper names, medical terms, and Latin-based words.

- The Arabic language is not based on the Latin alphabet and therefore requires different encoding for computer use. This is a practical problem, and a source of confusion, as several incompatible alternatives may be used. A code page is a sequence of bits representing a certain character (Beebe 1990; contributors 2006).

There are three main code pages used: Arabic Windows 1256, Arabic DOS 720, and ISO 8859-6 Arabic. Upon selecting any files it must first check the code page used. Most Arabic files use Arabic Windows 1256 encoding. Recently, files have been encoded using the Unicode UTF-8 code page. The standard code page for Arabic is Unicode UTF-16 and also the Unicode UTF-32 Standard. The use of Unicode may be regarded as best current practice.

### 2.2.3 PAWs and NPAWs

Arabic is an intricate language. It is therefore crucial that an equally rich and complex corpus is available to support development of Arabic OCR applications. In particular, the importance of PAWS and NPAWS is such that they must be included in any realistic text/image corpus. The addition of PAWs and naked PAWs is a novel extension of the previous word-based approach to Arabic corpora that specifically facilitates Arabic OCR. PAWs and NPAWs are included to the Multi-Modal Arabic Corpus (MMAC) shown in Chapter 3 for the following reasons:

- PAWs are the smallest separable token in the Arabic language.
- Each PAW is one glyph, making written Arabic a disconnected sequence of PAWs. Using PAWs is a reasonable way to overcome the problem of connected words.
- Removing the dots to leave naked words and naked PAWs reduces the number of letters that needs to be considered from 28 to 19.
- The total number of PAWs representing the language in MMAC corpus is 66,725, while the total number of NPAWs is comparatively small (32,804). Using NPAWs reduces the combinatory aspects of the problem.
- PAWs consisting of one and two letters can be extracted first, as if they are disconnected letters. These small PAWs represent a significant proportion of the language and are the most commonly used as shown in section 3.5.2 below. Their recognition can provide a firm foundation for later processes.

### 2.2.4 Arabic language morphology

The Arabic language depends mainly on the root of a word. The root word can produce either a verb or a noun for instance (عمل) - a root word – can be a noun as in (عامل معمل) or a verb as in (يعمل تعملون).

Stemmers, in general, tend to extract the root of the word by removing affixes. English stemmers remove only suffixes whereas Arabic stemmers mainly remove prefixes and suffixes, some of them also remove infixes.

Lexica on the other hand create a list of alternative words that can be produced by that root (Al-Shalabi and Evens 1998; Jomma, Ismail et al. 2006).

Arabic words (verbs and nouns) change according to the following variables: (Al-Shalabi and Evens 1998; Al-Shalabi and Kanaan 2004)

- **Gender:** Male or female, like (يعمل تعمل).
- **Tense (verbs only):** Past, present or future, like (عمل يعمل سيعمل).
- **Number:** Singular, pair or plural, like (تعمل تعملان يعملون).
- **Person:** First, second or third, like (عملت عملتن عملن).
- **Imperative verb:** like (اعمل لاتعمل).
- **Definiteness (nouns only):** Definite or indefinite, like (عمل العمل).

The Arabic language, in addition to verbs and nouns, contains some other words like prepositions, adverbs, pronouns and so on.

## 2.3 Pre-processing Phase

The pre-processing phase is the first phase of OCR as it receives the scanned image of the document for pre-processing and prepares the image for the feature extraction phase as shown in Figure 1.2. Pre-processing is responsible for cleaning the image with image processing techniques by converting the image to a more concise representation. The more successful the pre-processing algorithms are, the better the recognition rate of the application.

The proposed approach in this research focuses upon eliminating some steps from the pre-processing and recognition phases as shown in the framework of the proposed Arabic OCR as shown in Figure 1.3. This approach successfully generated a complete OCR application without these steps. This indicates that some of these steps are no longer required for this type of research. The following are the steps normally included in the pre-processing phase.

### 2.3.1 Document layout analysis and decomposition

Document layout analysis is a technique used to identify the physical structure of the page. It describes the page as a set of components. These components can be paragraphs or non-text components. There are two main approaches to document layout analysis: the top-down and the bottom-up approaches. The top-down approach seeks to find global information on the page, such as black and white stripes. It splits the page into columns, then blocks, lines and finally words. The bottom-up approach starts with local information consisting of black pixels and connected components. It firstly recognizes the words, then merges words into lines and finally lines into blocks (Simon, Pret et al. 1997).

Ittner and Baird (D.J.Ittner and Baird 1993) proposed a robust, multi-lingual top-down method. The method segments pages, after skew- and shear-correction, into blocks of text. This method locates large elongated white rectangles and pieces them together to separate the blocks of text. It generates the minimal spanning tree in order to detect the text line orientation, and finally uses the projection profiles of the blocks to find the text lines.

O’Gorman (O'Gorman 1993) introduced the document spectrum or ‘docstrum’, where the relationship of the objects is expressed with polar coordinates (distance and angle). It is a method used for structural page layout analysis based on bottom-up, k-nearest neighbour assembling of page components. The method measures skew, within-line, and between-line spacing and locates text lines and text blocks.

Breuel (Breuel 2002) presented geometric algorithms for solving the two main problems in page layout analysis. The first is to find a cover of the whitespace between texts depending on the maximum empty rectangles. The second is to find the text line of the maximum likelihood of the presence of geometric constrains. He also introduced an evaluation function that identifies maximal empty rectangles corresponding to column boundaries. The evaluation function was combined with the two geometric algorithms to produce an easy-to-implement layout analysis system.

### 2.3.2 Binarization and noise removal

Binarization is responsible for receiving the scanned image and converting the document from a grey-scale image to a binary, black and white, image. This process is sometimes referred to as thresholding. Noise removal removes small erroneous pixels from the binarized image.

A global grey level threshold which classifies each pixel's grey level depends on a global view and not local features (Cheriet, Kharma et al. 2007). Jianzhuang *et al.* (Jianzhuang, Wenqing et al. 1991) presented a global grey level threshold histogram using the Otsu method. His objective was to extend the technique to a 2-dimensional histogram. The 2-dimensional Otsu method enhances the grey level information of each pixel and its spatial correlation within the neighbourhood.

An approach for handling problems with the global grey level threshold is to use the local grey level threshold. This divides the image into sub-images and utilizes a different threshold for each sub-image. The key here is to estimate the threshold for each sub-image which in some cases gives a better threshold.(Gonzalez and Woods 2007). Trier and Taxt (Trier and Taxt 1995) presented a comparative study between eleven different types of binarization methods.

Noise removal is a process to enhance the quality of the image. It deals with what is called salt-and-pepper - which refers to dots in the image which occur from re-copying or from ageing. Dilation algorithms are used with broken characters. Erosion algorithms are used with touching characters. Souza *et al.* (Souza, Cheriet et al. 2003) presented a method for selecting the best filter to enhance poor documents using image quality assessment. They introduced five quality measures to get information about the quality of the images, and morphological filters to improve their quality.

### 2.3.3 Thinning

A thinning step is normally used with handwritten input and is also sometimes used with printed. This step removes the stroke width and leaves a skeleton stroke with a width of one pixel. The skeleton obtained must be as thin as possible, connected and centred (Cheriet, Kharma et al. 2007). This thinning step is generally applied after the binarization step of the pre-processing. A line-based feature extraction algorithm is normally used when applying the thinning algorithm, which extracts endpoints, holes, corner points and fork points (Cheriet, Kharma et al. 2007).

Harty and Ghaddar (Harty and Ghaddar 2004) presented a thinning method which converted the document image into binary representations. A matrix of ones (1's) for black pixels and zeros (0's) for white pixels were used. Thinning an image reduces character blocks to their skeletons. The thinning transforms characters into arc segments one pixel thick, preserving connected components and the number of cavities and holes. Line-based feature extraction was used by Artificial Neural Networks (ANN) for classification.

### 2.3.4 Slant / skew detection and correction

Slant is the sheer inclining of the word from a horizontal line as in *italic* words. The general purpose of slant correction is to reduce the variation of the script and to improve the quality of the words which leads to higher recognition accuracy (Cheriet, Kharma et al. 2007). Skew is the rotation of the whole document image which results from rotating the document while scanning. Skew can be recognized visually as a slope of the text lines with respect to the x-axis, and it mainly concerns the orientation of the text lines (Cheriet, Kharma et al. 2007).

Baird (Baird 1987) presented an algorithm for detecting the skew angle based on the projection profile of the maximum energy function. The best estimate of skew occurs at the global maximum of an energy function on sets of projection counts of character locations. Experiments showed that the algorithm worked well on a wide variety of page layouts. Both the speed and accuracy of the algorithm reported better results than had previously been reported.

Hull (Hull 1998) presented a survey study of four broad classes of skew detection techniques. These included methods that either calculated skew from a horizontal projection profile, a distribution of feature locations, a Hough transform, or the distribution of responses from local, directionally sensitive masks. The basic method used by each class of techniques was presented and individual algorithms within each class were discussed.

Slavik and Govindaraju (Slavik and Govindaraju 2001) proved the theoretical equivalence of two different methods for slant and skew corrections. They showed that skew correction, followed by slant shear transformation in a horizontal direction, is equivalent to first correcting slant shear transformation in the horizontal direction and then correcting skew by a shear transformation in the vertical direction of the other.

### 2.3.5 Textline / Baseline detection

A text line is a group of words and characters that are adjacent to each other and through which a straight line can be drawn (O'Gorman 1993). This straight line is the baseline which passes through all the connected components of an Arabic text line (Cheriet, Kharma et al. 2007). A text line is defined by the location of its baseline. The baseline is defined graphically as the line with the maximum value of black pixels in that text line (Shafait, Adnan-ul-Hasan et al. 2006). The baseline is the horizontal line that passes through the connected primitives of the line of text (Khorsheed 2002) or just under the non-descending letters for non-connected text. Figure 2.1 shows the location of the baseline in both the Arabic and English languages.

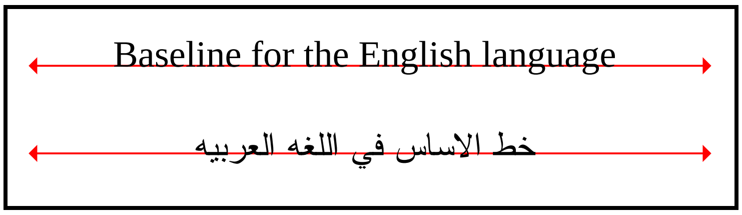


Figure 2.1: The baseline location of text in English and Arabic

Breuel (Breuel 2002) presented a new algorithm to address the identification of both page rotation and text lines problems. He used a branch-and-bound approach for finding the optimal global line and modelled baseline and descender line using a Gaussian error/robust least square model.

O’Gorman (O'Gorman 1993) introduced the document spectrum, or docstrum as previously discussed. His method yields an accurate computed skew, within-line, and between-line spacing and locates text lines and text blocks.

### 2.3.6 Word normalization

Word normalization scales the word image to fit a required standard height for the classification process. Normally, the character image is mapped onto a standard size so as to give a unified dimension for classification (Cheriet, Kharma et al. 2007). Basically there are two different approaches for normalization; moment-based normalization and nonlinear normalization.

Among the various methods of normalization, moment-based normalizations are known for enhancing the character recognition rate. Miyoshi *et al.* (Miyoshi, Nagasaki et al. 2009) presented moment normalization methods that use the moments of character contours rather than character images themselves to calculate the scaling parameters. Their experiments showed that the proposed method is effective, particularly for printed character recognition.

Maddouri *et al.* (Maddouri, Amiri et al. 2000) presented a local normalization method for global recognition of Arabic handwritten scripts. This normalization method is based on Fourier coefficients of a chain-encoded contour. The method begins with boundary detection and then generates a Freeman chain code. A Fast Fourier Transform (FFT) algorithm calculates the Fourier coefficients of a chain-encoded contour. Finally these coefficients are normalized to cope with the orientation and size variation of a handwritten character. Experimental tests help to evaluate distances between the normalized character and its point of reference.

## 2.4 Feature extraction Phase

The purpose of the feature extraction phase is to calculate the attributes of patterns that are most suitable for the classification phase. When the input pattern is too large to be processed as one set and is suspected to be particularly redundant (with a lot of data, but little information) then the input data must be transformed to a compact representational set of features. The method of transforming the input data into a set of features is called feature extraction. Generally, if the features are carefully selected, it is expected that the relevant information will be extracted from the input data. Feature extraction methods are designed for different representations of characters. The feature extraction method is selected based on invariance properties, re-constructability, expected distortions and variability of the characters (Cheriet, Kharma et al. 2007).

The feature extraction algorithms can be categorized into geometric; structural and space transformation features. An example of the geometric features is moments; of the structural features is Fourier descriptors and for space transformation features an example would be linear transformation (Cheriet, Kharma et al. 2007). Trier *et al.* (Trier, Jain et al. 1996) presented a survey study of feature extraction methods for off-line recognition of segmented (isolated) characters. Their study included the feature extraction methods of grey-scale images, binary contour images and vector representation.

The feature extraction algorithm applied in this research has Haar-like features which consider as one of the geometric features. The approach is explained with the following commonly used feature extraction methods.

### 2.4.1 Moments

Moments and functions of moments as feature extraction methods have been used in character recognition. These methods extract global geometric properties of the image such as shape area, centre of mass and moment of inertia (Cheriet, Kharma et al. 2007).

Teh and Chin (Teh and Chin 1988) presented various types of moments used to recognize image patterns as features. A number of moments were surveyed and some issues were addressed, such as image-representation ability, noise sensitivity, and information redundancy. Moment type evaluations include regular moments, Legendre moments, Zernike moments, pseudo-Zernike moments, rotational moments, and complex moments.

### 2.4.2 Hough transform

The Hough transform is a popular feature extraction method used in digital image processing. It is able to detect straight lines, curves, or any shape used in parametric equations. The Hough transform uses the polar parametric form to identify possible initial centre of a circle, given an edge point and a radius to search over. It maps figure points from the feature space to parameter space and therefore extracts the features (Gonzalez and Woods 2007). There are two types of Hough transform; standard and randomized Hough transform. Touj *et al.* (Touj, Amara et al. 2003) presented a generalized Hough transform technique which is known for its ability to absorb the distortions in the document image as well as noises. They described an approach proving the efficiency of the Generalized Hough Transform to recognize Arabic printed characters in their different shapes.

### 2.4.3 Fourier descriptors

Fourier descriptors (FD) is one of the most popular and efficient feature extraction methods used in digital image processing. It represents the shape of the object in the frequency domain. It includes a strong discrimination facility, low noise sensitivity, easy normalization and information preservation (Cheriet, Kharma et al. 2007; Gonzalez and Woods 2007). Zhang and Lu (Zhang and Lu 2001) presented a comparative study on shape retrieval using Fourier descriptors with different shape signatures. The methods achieved a high level of representation and normalization. Different shape signatures exploited were of complex coordinates, centroid distance**,** **c**urvature signature and cumulative angular functions. They built a Java retrieval application to compare shape retrieval using Fourier descriptors which resulted from the use of different signatures. General techniques for shape representation and normalization were also analyzed.

### 2.4.4 Linear transforms

Feature space linear transformation methods are used to change the feature representation of patterns in order to improve the classification performance. Linear transforms usually reduce the dimension of the features. Linear transforms include

Principal Component Analysis (PCA) and Linear Discriminate Analysis (LDA) (Gonzalez and Woods 2007).

Principal Component Analysis (PCA) is a statistical technique which belongs to the category of factor analysis methods. PCA is a mathematical method that uses an orthogonal transformation to convert a set of correlated observations into a set of uncorrelated variables called principal components. Zidouri (Zidouri 2007) presented two level recognition processes for Arabic characters. In his proposed approach, recognition is applied at two levels with different approaches. The first level is applied after word segmentation to recognize isolated characters, whilst the second level is applied to segmented characters. A PCA feature extraction algorithm reported great results with Arabic character recognition.

Linear Discriminate Analysis (LDA) is a parametric feature extraction method used to find a linear combination of features which differentiate two or more classes of objects. The resulting mixture can be used as a linear classifier or, more commonly, for dimensionality reduction before later classification. Kurt *et al.* (Kurt, Turkmen et al. 2009) presented Linear Discriminate Analysis (LDA) based on the Ottoman Character Recognition system. LDA reduces the size of the data whilst keeping the variation present in the original dataset. The training set images were normalized to reduce the variations in illumination and size. The features were extracted by LDA. Principal Component Analysis (PCA) was applied as an intermediate step. The described processes were also applied to the unknown test images.

### 2.4.5 Haar-like feature extraction

Haar-like feature extraction is a simple method depending on the basic visual features of the objects. It uses grey-scale differences between rectangles in order to extract the object features (Viola and Jones 2001).

#### *2.4.5.1 Wavelet*

The wavelet is a simple data representation algorithm that is used to generate the Haar-like feature extraction. The wavelet is a mathematical method for decomposing the data in a hierarchal way and can represent any type of data whether it be an image, curve or surface. Wavelets were originally applied in signal processing and in approximation theories. It has recently been applied in computer graphics and image processing (Stollnitz, DeRose et al. 1995) and provides a powerful method for representing all levels of details in an image. Wavelet transform is the representation of the wavelet function by small signals. Wavelet transform has the advantages of using discontinuities, sharp peaks and accurately deconstructing and reconstructing finite signals (Gonzalez and Woods 2007).

#### *2.4.5.2 Haar wavelet*

The Haar wavelet was first invented by Alfred Haar in 1909 (Haar 1910) and is considered to be the simplest wavelet function. The Haar wavelet is applied by calculating the sums and differences of adjacent elements and is equal to its inverse. The Haar wavelet works with either one or two dimensional data. It is applied first on adjacent horizontal elements and then on vertical adjacent elements (Stollnitz, DeRose et al. 1995).

To explain how the Haar wavelet works, we will look at a simple example of a one dimensional image with four pixels of values (5 9 7 3). The Haar wavelet simplifies the data by averaging pairs together to create a new lower resolution level (7 5). This data can then be transformed to an even lower resolution level (6). This method shows that some information is lost during the transformation and can’t be recovered from the original data (Stollnitz, DeRose et al. 1995). Recovering the original data (reconstructing data) requires the preservation of detail coefficients which preserves the missing information. In the previous example; in the first resolution level, the first coefficient must be -2 as the first value 5 is less than the average 7 by 2. The second coefficient is 2. The lower resolution level coefficient is 1 as the first value 7 in the upper level is more than the average 6 by 1. The final wavelet transformation will be (6 1 -2 2) (Stollnitz, DeRose et al. 1995).

#### *2.4.5.3 Haar-Like Features (wavelets)*

Haar-like features are a class of simple local features that are calculated by subtracting the sum of a sub-window of the feature from the sum of the remaining window of the feature (Messom and Barczak 2006). The Haar-like features are computed in short and constant time rather than using a pixel-based system.

According to Lienhart *et al.* (Lienhart, Kuranov et al. 2002) assume that the Haar-like features for an object lies within a window of $W\ X\ H$ of pixels, which can be defined in the following equation:

$$features = \sum_{i \epsilon I}^{N} \omega_{\mathrm{i}} \cdot RecSum(r_i)$$

Where $\omega_i$ is the chosen weighting factor which has a default value of 0.995 (Lienhart, Kuranov et al. 2002). A rectangle is specified by five parameters. $r = (x, y, w, h, \alpha)$ with

$$0 \leq x, x + w \leq W, \qquad 0 \leq y, y + h \leq H, \qquad x, y \geq 0, w, h > 0 \quad \alpha \epsilon \{0°, 45°\}$$

and its pixel sum is denoted by $RecSum(r_i)$. Two examples of such rectangles are given in Figure 2.2 for the upright and the 45° rotated rectangles.

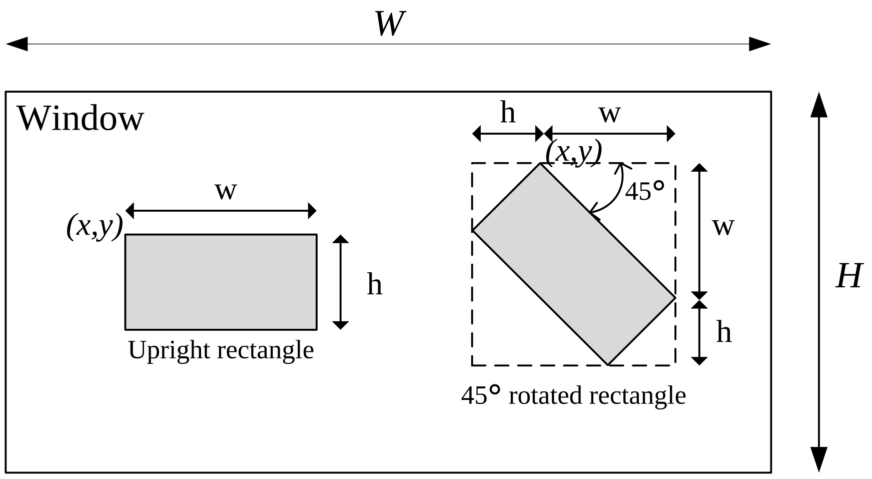


Figure 2.2: Upright and 45° rotated rectangle of detection window

The previous equation generates an almost infinite feature set. It has been reduced for the practical application of the approach. This reduction generates 15 feature prototypes shown in Figure 2.3: (1) four edge features, (2) eight line features, (3) two centre-surround features, and (4) a special diagonal line feature. The generated set of features was scaled in the horizontal and vertical directions. Note that the line features can be calculated by only two rectangles. Edge features (a) and (b), line features (a) and (c) and the special diagonal line features are the features that were used in the beginning by (Papageorgiou, Oren et al. 1998; Mohan, Papageorgiou et al. 2001; Viola and Jones 2001). They used the value of a two-rectangle feature (edge features) as the difference between the sum of the pixels between the two rectangular regions. The two regions are of the same size and shape and are either horizontally or vertically adjacent. A three-rectangle feature (line features) subtracts the sum of the two outside rectangles from the sum of the middle rectangle. A four-rectangle feature (special diagonal line feature) subtracts the sum of the two diagonal pairs of rectangles (as in Figure 2.3).

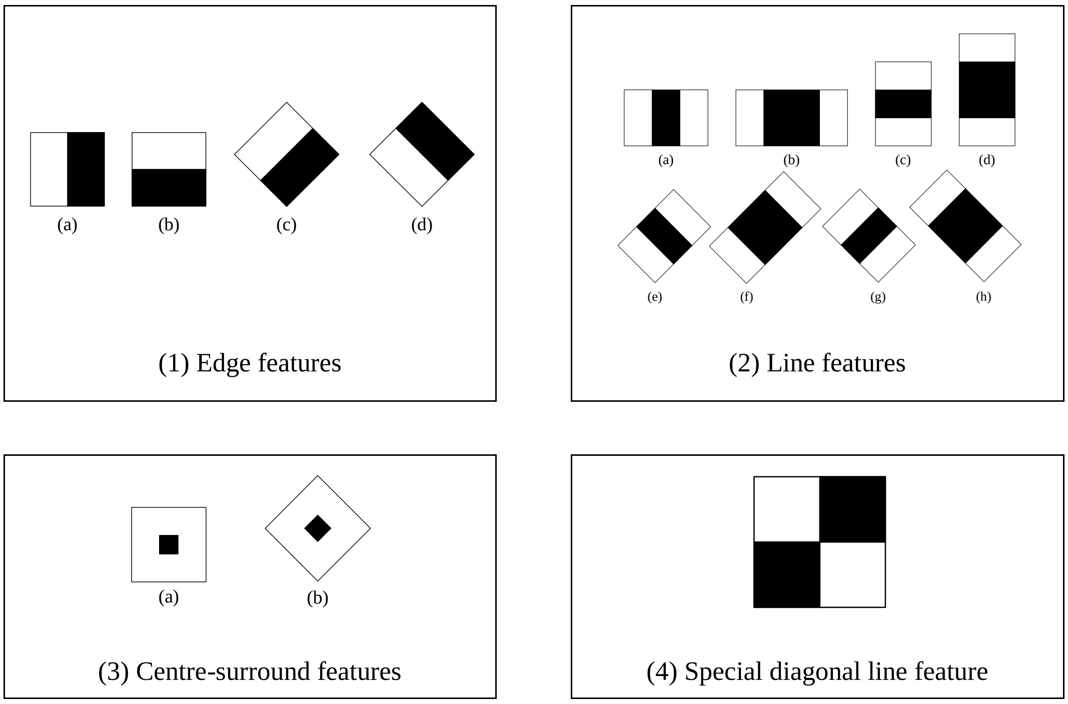


Figure 2.3: Haar-like features prototype

In the experiments of Lienhart and Maydt (Lienhart and Maydt 2002) rotated features had been added which significantly enhanced the learning system and improved the performance of the Haar Cascade Classifier. These rotated features had a significant enhancement when applied to objects with diagonal shapes. It produces a remarkable increase in accuracy when used with Arabic character recognition.

The total number of features differs from prototype to prototype. It can be calculated as explained by Lienhart *et al.* (Lienhart, Kuranov et al. 2002); as shown in Figure 2.2.

$X = [W \div w]$, $Y = [H \div h]$ is the maximum scaling factor in $x$, $y$ directions. The upright features is equal to $XY \cdot (W + 1 - w \cdot (X + 1)/2) \cdot (H + 1 - h \cdot (Y + 1)/2)$ features for an image of size $W \times H$. The 45° rotated features is equal to $XY \cdot (W + 1 - z \cdot (X + 1)/2) \cdot (H + 1 - z \cdot (Y + 1)/2)$ for $z = w + h$. As an example for a window of size $24 \times 24$ , the total number of features is 117,941 (Lienhart, Kuranov et al. 2002).

### 2.4.5.4 Integral image

Integral image was first used in feature extraction by Viola and Jones (Viola and Jones 2001). Lienhart and Maydt (Lienhart and Maydt 2002) developed the algorithm by adding the rotated integral image. It is also called Summed Area Table (SAT). This is an algorithm for calculating a single table in which pixel intensity is replaced by a value representing the sum of the intensities of all pixels contained in the rectangle. This is defined by the pixel of interest and the upper left corner of the image (Crow 1984). It can be calculated very quickly and efficiently for an image in a few operations per pixel. Haar-like features at any scale can be simply calculated in constant time, whenever the integral image is calculated (Kasinski and Schmidt 2010).

The integral image for the upright rectangle at any location $(x, y)$ is equal to the sum of the pixels' intensities (grey-scale) from the $(0, 0)$ to $(x, y)$ as shown in Figure 2.4 a. and known is as $SAT(x, y)$.

$$SAT(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

Where $I(x, y)$ is the intensity value at $(x, y)$.

The $RecSum(r)$ for the upright rectangle $r = (x, y, w, h, 0°)$ as shown in Figure 2.4 (b) is:

$$RecSum(r) = SAT(x - 1, y - 1) + SAT(x + w - 1, y + h - 1) \\ - SAT(x + w - 1, y - 1) - SAT(x - 1, y + h - 1)$$

The integral image for the 45° rotated rectangle at any location $(x, y)$ is equal to the sum of the pixels' intensities at a 45° rotated rectangle with the bottom most corner at $(x, y)$ and extending upward till the boundary of the image as shown in Figure 2.4 (c) and is known as R$SAT(x, y)$.

$$RSAT(x, y) = \sum_{y' \le y, y' \le y - |x - x'|} I(x', y')$$

Where $I(x, y)$ is the intensity value at $(x, y)$.

The $RecSum(r)$ for the upright rectangle $r = (x, y, w, h, 45°)$ as shown in Figure 2.4 (d) is:

$$RecSum(r) = RSAT(x - h + w, y + w + h - 1) - RSAT(x - h, y + h - 1) - RSAT(x + w, y + w - 1) + RSAT(x, y - 1)$$

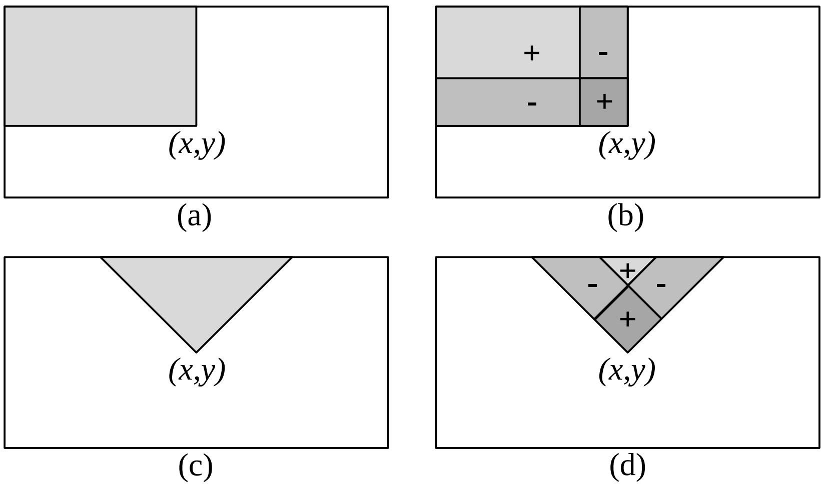


Figure 2.4: Supplementary image illustration SAT and RSAT

## 2.5 Character segmentation Phase

Character segmentation refers to the process of splitting a word into its characters, strokes or primitives (Lorigo and Govindaraju May, 2006). Character segmentation is an important phase in the case of handwriting and in the cursive languages. It is very important to mention here that the contribution of this research is towards handling printed Arabic character recognition totally without character segmentation.

Although there were previous trials to have character segmentation-free systems, their practical effect was limited (Abdelazim 2006).

### 2.5.1 Isolated / pre-segmented characters

This type of character segmentation recognizes numerals and isolated characters only. This type might only be useful for academic purposes and is not practical in real life OCR applications (Khorsheed 2002; Abdelazim 2006).

Touj *et al.* (Touj, Amara et al. 2003) presented research on using a Hough Transform to recognize isolated Arabic characters. The research described the robustness of the Hough Transform rather than trying to tackle overall Arabic character recognition.

### 2.5.2 Segmenting a word into characters

This type of character segmentation transfers a word, sub-word or connected component into characters. These systems dealing with this type of character segmentation by connect the characters together to construct a sub-word or a word (Khorsheed 2002).

Sarfraz *et al.* (Sarfraz, Nawaz et al. 2003) presented a technique for the automatic recognition of Arabic printed text using artificial neural networks. The technique segments the text into individual characters. The segmentation process proposed consists of three main steps: the segmentation of lines of text into different zones, segmentation of lines of text to different words and the segmentation of words into individual characters. The segmentation process depends on horizontal and vertical histograms.

### 2.5.3 Segmenting a word into primitives

This type of character segmentation transfers a word, sub-word or connected component into symbols. This symbol may be a character, a stroke, ligature or sub-word. The systems dealing with this type of character segmentation either connect the symbols together to construct a character or detect the symbols themselves (Khorsheed 2002).

Trenkle *et al.* (Trenkle, Gillies et al. 2001) improved a system that recognizes Arabic and Farsi text. This module is an oversegmenter, designed to generate atomic image segments with a size no larger than a single character. Hence, each atomic segment should come from only a single character of the ideal character segmentation. Character segmentation can be produced by combining the atomic segments into suitable groups. The combination must be done in such a way as to keep the spatial relationships between the atomic segments in the group. A Viterbi algorithm generates appropriate groups as a by–product of the recognition process.

### 2.5.4 Integration of recognition and character segmentation

Here, character segmentation is performed after or during recognition. The approach is to scan the word from right to left. At each step, the process either relates a column to one of the codebook entries or calculates cumulative moment invariants. The system is not always able to recognize all characters, which means that the following characters in the same sub-word would not be processed either (Khorsheed 2002).

Lu *et al.* (Lu, Bazzi et al. 1999) presented a language-independent optical character recognition system that is able, in principle, to recognize printed text from different languages. The system uses a Hidden Markov Model (HMM) developed for continuous speech recognition in order to model each character. The approach demonstrated a language-independence for Arabic, English, and Chinese. The approach depends on automatic training on non-segmented data, and simultaneous character segmentation and recognition. The training algorithm scans the document line by line using vertical slices. The features extracted from the vertical slices are attached with the ground truth information as an input to train the character models.

### 2.5.5 Character segmentation-free systems

Character segmentation-free systems are motivated by insight in psychological studies of the human reading process and were originally introduced for speech recognition. These systems attempt to recognize the whole word without trying to segment and recognize characters or primitives individually (Khorsheed 2002). In other words they tend to avoid intra-word recognition. The main reason for this type of research is that reliable character segmentation is not always possible or easy, even for machine printed Arabic text (Abdelazim 2006).

Al-Badr and Haralick (Al-Badr and Haralick 1998) Described the design and implementation of an Arabic word recognition system. The system does not segment a word into characters to recognize it. It recognizes the input word by detecting a set of “shape primitives” on the word. It then compares the regions of the word (represented by the detected primitives) with a set of symbol models. The approach depended on image processing techniques in order to categorize the symbol models. They used a single font, and training was on idealized isolated shape samples. These operators are normally very sensitive to noise (Abdelazim 2006).

## 2.6 Classification Phase

Classification involved assigning each point in the feature space with a class label score of the defined (classified) class. This point was originally mapped from the input pattern onto points in the feature space which was done in the feature extraction phase (Cheriet, Kharma et al. 2007). The Classification phase is sometimes embedded within the feature extraction phase or at other times with the character segmentation phase in one recognition phase.

The classification method applied in this research is a cascade of the Ada-Boosting classifiers. The proposed approach is explained in this section with a number of various classification methods that have shown success or have high potential character recognition.

### 2.6.1 Statistical methods

Statistical classification methods are based on Bayes decision theory, which aims to minimize the error of classification using a loss matrix and estimated probabilities (Cheriet, Kharma et al. 2007). This classifier assumes that different classes and feature vectors have an underlying joint probability. The probability density function can be cumulative (Khorsheed 2002). Abandah *et al.* (Abandah, Younis et al. 2008) presented an optical character recognition solution for Arabic handwritten scripts. The principal component analysis (PCA) method is used to select the best subset of features extracted from a large number of features. Parametric and non-parametric statistical classifiers were used. They found that a subset of 25 features is needed in order to get an 84% recognition accuracy using a linear discriminate classifier. They also found that using more features does not significantly enhance accuracy. Classifiers that are parameterized achieve better accuracy than non-parameterized classifiers.

Hidden Markov Models (HMMs) are statistical models considered as the simplest dynamic Bayesian network which have been found to be extremely efficient for a wide spectrum of applications. HMM applications are commonly used in pattern recognition such as speech, handwriting, gesture recognition and bioinformatics. In a Hidden Markov Model, the state internal is not visible, but output, relates to a visible state. Each state has a probability distribution across the possible output tokens. Therefore the series of tokens generated by an HMM gives some information about the series of states (Khorsheed 2002; Cheriet, Kharma et al. 2007).

Alma'adeed *et al.* (Alma'adeed, Higgens et al. 2002) presented a complete scheme for unconstrained Arabic handwritten word recognition based on a HMM. A complete system of Arabic Handwritten words was first proposed. The system removes variations in the images that do not influence the handwritten word. Next, the system codes the word so that key information about the lines in the skeleton is extracted. Then a classification process based on the HMM approach is used. Finally, the output produced is a word in the database.

### 2.6.2 Structural methods

Structural pattern recognition methods are usually used in online character recognition rather than in offline character recognition. Structural methods represent a pattern as a structure (string, tree, or graph) of flexible size. Structural methods face two major problems. These problems are extracting structural primitives (strokes or line segments) from input patterns and learning patterns from samples (Cheriet, Kharma et al. 2007).

Bushofa and Spann (Bushofa and Spann 1997) presented an approach that enables Arabic words to be segmented into characters. Diacritics are removed using the proposed algorithms. This algorithm reduces the number of classes to 32. Information about these diacritics, such as their number, position and type is kept and used in the final recognition stage. Features of the skeletonised character are used for classification using a decision tree.

### 2.6.3 Artificial Neural Networks (ANN)

Artificial Neural Networks were initially developed with great expectations to achieve intelligent perception and cognition machines by simulating the physical structure of human brain (Cheriet, Kharma et al. 2007). OCR is one of the most successful applications that have been proposed for neural networks (Khorsheed 2002).

Harty and Ghaddar (Harty and Ghaddar 2004) built two neural networks to classify the segmented characters of handwritten Arabic text. The two neural networks correctly recognized 73% of the characters. They stated that character classification, especially handwritten Arabic characters, depends largely on appropriate information, not only on topographic features extracted from these characters.

### 2.6.4 Support Vector Machine (SVM)

A support vector machine (SVM) is a concept for supervised learning algorithm that analyzes input data and recognizes patterns. It is used for classification and regression analysis. The standard SVM takes a set of input data and predicts - for each input - two possible member classes. The approach makes the SVM a non-probabilistic binary linear classifier. Given a set of training examples, each are marked as belonging to one of two categories; a SVM training algorithm builds a model that assigns new examples into one category or another (Cheriet, Kharma et al. 2007).

Mehran *et al.* (Mehran, Pirsiavash et al. 2005) presented a front-end OCR for Persian/Arabic cursive documents, which controls an adaptive layout analysis system with a combined MLP-SVM recognition process. The implementation results - on an inclusive database - show a high degree of accuracy which meets the requirements of commercial use.

### 2.6.5 Classification with multiple classifiers

Classification with multiple classifiers has been an active research topic since the 1990s. Many combinations of classifier methods have been proposed, and the applications to practical problems have been proven to be the advantage of grouping over individual classifiers (Cheriet, Kharma et al. 2007).

Rahman and Fairhurst (Rahman and Fairhurst 2003) presented a review study of the multi classifiers. They explicitly review the field of multiple classifier decision combination strategies for character recognition. A new taxonomy for categorizing approaches is defined and explored to clarify the mechanisms by which multi-classifier configurations deliver performance enhancements. They illustrate explicitly how the principles underlying the application of multi-classifier approaches can easily generalise to a wide variety of different task domains.

### 2.6.6 Cascade of boosting classifiers

A classifiers cascade is a decision tree algorithm which depends upon the rejection of non-object regions. A classifier detects the objects of interest and rejects the non-object patterns (see Figure 2.5). Boosting is a machine learning method that is based on this observation. Boosting algorithms use a large set of weak classifiers in order to generate a powerful classifier. Weak classifiers are used in order to discriminate required objects from the non object in a simple and quick way. Weak classifiers use only one featured each stage and depend on a binary threshold decision or small CART for up to four features at a time (Schapire 2002).

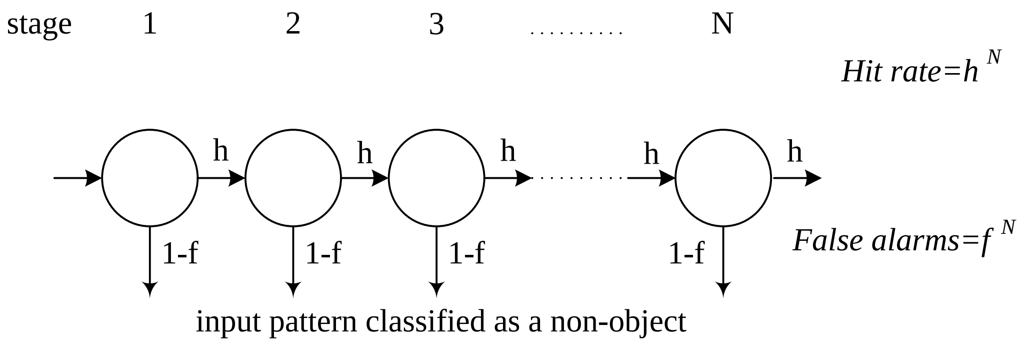


Figure 2.5: Cascade of classifiers with N stages

#### *2.6.6.1 Boosting algorithm*

The Boosting algorithm is a machine learning algorithm required for supervised learning in order to improve the performance of any learning algorithm. It trains an exact learner, extracts its weak suggestions and combines them, in order to finally output a strong classifier which boosts the recognition rate (Bishop 2006). Boosting can be used to reduce the error of any “weak” learning algorithm that consistently generates classifiers which need to have better than random guessing power. Boosting works recursively by running a given weak learning algorithm on various distributions over the training data, and then combining the classifiers produced by the weak learner into a single composite classifier (Freund and Schapire 1996).

#### *2.6.6.2 Ada-Boosting algorithm*

Ada-Boost stands for Adaptive Boosting. Ada-Boost calls up a weak classifier repetitively serially, each time distributing a weight; these weights increase to generate the new classifier (Bradski and Kaehler 2008). It is used to significantly reduce the error margin of any learning algorithm that consistently generates classifiers whose performance is little better than random guessing (Freund and Schapire 1996).

A boosting algorithm is a machine learning algorithm for supervised learning in order to improve the performance of any learning algorithm. Ada-Boost calls up a weak classifier repetitively a series of times, each time distributing a weight. These weights increase to generate the new classifier (Bradski and Kaehler 2008). Gentle Ada-Boost (GAB) reduces the exponential loss function of Ada-Boost. Discrete Ada-Boost (DAB) uses weak assumptions with outputs restricted to the discrete set of classes. Real Ada-Boost (RAB) utilizes confidence rated predictions (Bradski and Kaehler 2008). The default value is Gentle Ada-Boost.

### 2.6.7 Integration of feature extraction and classification

The integration of feature extraction phase with the classification phase is an approache for the object recognition. The Haar Cascade Classifier approach (HCC) is a tool that used the Haar-like feature extraction with the cascade of Ada-boosting classifiers algorithm in one tool.

The Haar Cascade Classifier (HCC) approach was initially presented in 2001. Viola and Jones (Viola and Jones 2001) introduced a rapid object detection algorithm using a boosted cascade of simple features for a face detection application. The Integral Image method of image representation was introduced and the new learning algorithm based on Ada-Boost was proposed. Viola and Jones presented a method to combine more complex classifiers in a cascade. Lienhart and Maydt (Lienhart and Maydt 2002) extended the Haar-like features by adding rotated Haar-like features and introduced a post optimization procedure to improve the false alarm rate. Lienhart *et al.* (Lienhart, Kuranov et al. 2002) presented an empirical analysis of different boosting algorithms which provide better detection performance and lower computational complexity.

The Haar cascade classifier approach was initially created using the Open Computer Vision (OpenCV) library. OpenCV is an open source library of Computer Vision programming functions. It is aimed at real time computer vision applications using C/C++ and Python. OpenCV was founded by the Intel corporation in 1999 (Intel 2001; Bradski and Kaehler 2008). The first release was published in the IEEE Conference on Machine Vision and Pattern Recognition 2000 (Bradski and Pisarevsky 2000). OpenCV receives corporate support from Willow Garage which began from the middle of 2008 (Bradski and Kaehler 2008).

#### *2.6.7.1 Originate of Haar Cascade Classifier approach*

The Haar Cascade Classifier (HCC) is a machine learning approach for visual object detection. It is capable of processing images very fast and in an efficient way. HCC combines the feature extraction and classification phases in one single approach. The Haar Cascade Classifier approach was initially presented for face detection (Viola and Jones 2001; OpenCV 2002; Seo 2008), while (OpenCV 2002) presents a general object detection method not just for faces. Adolf (Adolf 2003) presented an experiment using the method in order to detect a bowl. Kasinski and Schmidt (Kasinski and Schmidt 2010) applied the same approach not only for face detection but also for eye detection. HCC combines three main basic techniques together in order to form a good detection approach. These three techniques are (Kasinski and Schmidt 2010):

1. *Haar-Like Features:* A broad set of visual features that can be computed very quickly and in inexpensive computations using the integral image representation.

2. *Learning Boosting Algorithm:* This selects a small number of essential visual features from a large set of features and applies an efficient classifier.

3. *Cascade Classifiers:* Combining more complex classifiers in a cascade results in a fast and efficient detection method.

#### 2.6.7.2 Advantage of Haar Cascade Classifier approach

The Haar Cascade Classifier (HCC) has the following advantage as a machine learning approach (Lienhart, Kuranov et al. 2002):

1. Dealing with the visual properties of the images and objects. It is very beneficial for detecting objects with clear visual properties.

2. Rotation invariant approach that is powerful and fast in handling rotated images and rotated objects in the images (Lienhart, Kuranov et al. 2002).

3. Scaling invariant approach that detects objects with different scales inside the same image and needs no normalization algorithm for recognition (Lienhart and Maydt 2002).

4. The approach combines the feature extraction phase with the classification phase. These combined phases enable the fast recognition of the objects (Viola and Jones 2001).

5. Copes with grey-scale images that need no binarization algorithm.

6. Fast and reliable machine learning object detection approach.

## 2.7 Post-processing Phase

Post processing involves error correction or resolving ambiguity in OCR results by using appropriate information. The output of OCR is compared to how the system’s dictionary (corpus) and candidates are generated. According to the difference between the output of OCR and the output of corpus look-up, the numbers expressing the belief in the correct classification are modified. The output sequence of suitable candidates is then ordered and the best candidate is selected.

Post-processing approaches deal with the token after the classification stage in order to identify if it is the best matching one. Most of these are heuristic approaches that lead to improvement in the overall recognition accuracy. These approaches contain, in most cases, a powerful data structure, a fast searching algorithm and a smart string correction algorithm (Cheriet, Kharma et al. 2007).

### 2.7.1 Enhancing recognition using a confusion marix

The confusion matrix is a matrix of the predicate and actual classification of the sample. The size of the matrix is defined by the number of classifications defined in the sample. Kohavi and Provost (Kohavi and Provost 1998) introduced a glossary of terms for machine learning. They presented the regular form of the confusion matrix in terms of a relationship between the predicted and actual classifications. The confusion matrix is used extensively in the research topics of OCR, speech recognition, spell checking and natural language processing (Taghva, Borsack et al. 1995). The confusion matrix shown in Table 2.5 is a two class classifier, the confusion matrix in this research is a 61 class classifier, where the 61 is the total number of classifiers that are representing the Arabic language.

Table 2.5: A confusion matrix with two class classifier

| | | Predicted | |
|---|---|---|---|
| | | Negative | Positive |
| Actual | Negative | A: is the number of correct predictions that an instance is negative | B: is the number of incorrect predictions that an instance is positive |
| | Positive | C: is the number of incorrect predictions that an instance negative | D: is the number of correct predictions that an instance is positive |

K. Marukawa *et al.* (Marukawa, Hu et al. 1997) used the confusion matrix in an OCR system in order to process a Japanese document. Their idea was to utilize the characteristics of recognition errors. Two different methods were used. The first one was a confusion matrix in order to generate corresponding query strings that must match incorrectly with the recognized text. The other one searched in a "non-deterministic text" that included multiple candidates for ambiguous recognition results. Experiments applied showed that character recognition and retrieval techniques can be combined effectively.

G. Vamvakas *et al.* (Vamvakas, Gatos et al. 2010) used the confusion matrix in off-line handwritten character recognition. The confusion matrix in this case was used in the classification phase. The features were extracted first at different levels of granularity for all patterns in the training set. The confusion matrices were constructed at each step in the training phases. Classes with high values in the confusion matrix were merged at a certain level and for each group of merged classes.

### 2.7.2 Look-up dictionary

Searching within the look-up dictionary requires a powerful data structure because it includes all the corpus information which usually contains millions of tokens. Researchers usually use either trees or graphs as data structures with the searching algorithm in the post-processing stage. These two types of data structure are tailored to deal with huge datasets like the corpus dataset (Hulden 2009). Since some ambiguity can remain even after dictionary verification, some investigative research is being carried out into the use of higher level context in order to reduce the ambiguity. A dictionary is appended with information about the token which can be used by the higher level contextual system in order to predict or correct the detected token (Ford and Higgins 1990).

Now follows a discussion of various data structures and string correction algorithms that are usually used for the post-processing techniques to enhance the OCR recognition rate.

#### *2.7.2.1 Dictionary tree structure – Trie structure*

The tree is one of the most powerful advanced data structures and consists of nodes organised in a hieratical sequence. The simplest type of index is a sorted listing of the key field. This provides a fast lookup that uses a binary search to locate any item without having to look at each one in turn (Oommen and Badr 2007).

A trie, or prefix tree, is an ordered tree data structure that is used in order to store an associative array of strings. Each string is stored in nodes, each node includes a character. When more than one string shares the same prefix it separates them when a character is changed. All the children nodes of each node have a common parent from the associated string, and the root is associated with the empty string. Values are normally not associated with every node, only with leaves and some inner nodes that match the key of interest (Cheriet, Kharma et al. 2007).

Figure 2.6 shows a sample of a trie structure that includes a simple lexicon of 12 strings of English Gregorian months. The red colour nodes are the nodes with more than one child node, green colour nodes are the start of nodes with only one child, blue colour nodes are the nodes with one parent and one child and the yellow colour nodes are the leaf nodes. The trie contains 9 levels which mean that the maximum string length in the trie is 9 characters. The middle levels of the trie contain more nodes than that on the sides; For the reason that the number of nodes starts in the first level, limited to the number of characters in the language and increase branching while the levels are increase; then decreasing almost after the middle level when string lengths are nearly finished.

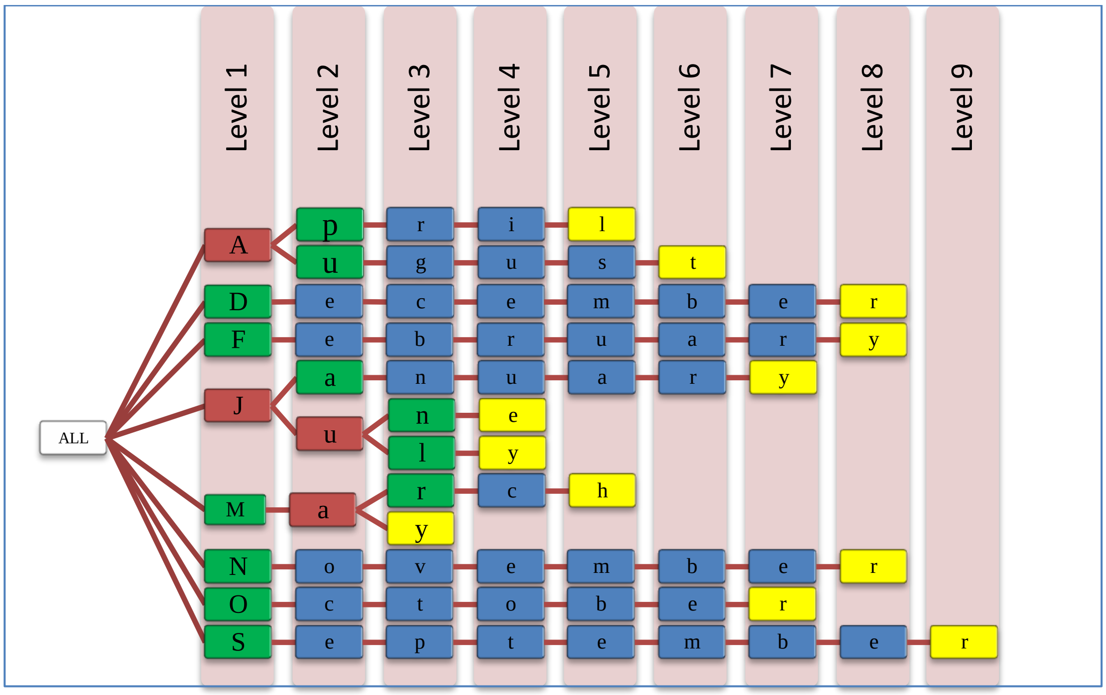


Figure 2.6: Trie structure of a simple lexicon of 12 English months

A tail-end trie is another type of trie structure and is usually used in order to reduce the size of the trie data structure. If the end of a word is unique and is not duplicated with any other word, then the tail-end trie size can be reduced by constructing a special end-of-word (leaf) node which contains the rest of the word as a string. This reduces the need for extra nodes and pointers (Ford and Higgins 1990). Figure 2.7 shows the same sample of Figure 2.6 as a simple lexicon of 12 strings of English Gregorian months as a tail-end trie. The red colour nodes are the nodes with more than one child node and the yellow colour nodes are the leaf nodes. The trie contains only 3 levels which are very small relative to Figure 2.6 and means that the maximum word length must be more than or equal to 3 characters. The number of nodes and branching in this type of trie are limited relative to Figure 2.6.

The advantage of a tail-end trie is that it reduces the size of the data structure and memory size during runtime. It may also increase the speed of search of the trie. The disadvantage is regarding the modification of the dictionary as it is more complex. For example, to add an extra word, it is easier to reconstruct the trie from scratch. Also, searching inside the trie requires different methods with the nodes and leaves (Ford and Higgins 1990).

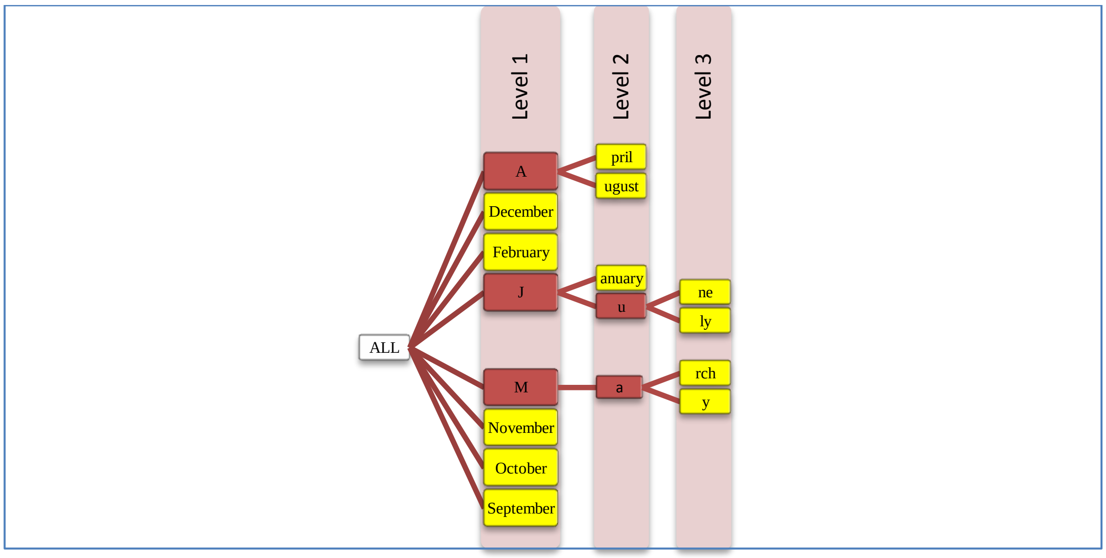


Figure 2.7: Tail-end trie of a simple lexicon of 12 months

One advantage of the trie structure is the fast look-up for entry; if an entry is of length *m* it takes *O (m)* time in a worst case scenario to find it. It also takes less space for saving and for runtime and is also fast because it uses a recursive function (Cheriet, Kharma et al. 2007).

#### 2.7.2.2 Dictionary Graph structure – directed acyclic word graph

A graph is a mathematical data structure consisting of a set of vertexes (nodes) and a set of edges. An edge contains a pair of vertexes which are called the edge endpoints. Graphs are ubiquitous in computer science and are used to model real life applications. A graph may be formed using either directed or undirected edges. The directed edge is a one-way connection, and is typically drawn as an arrow. The undirected edge models a "two-way" or "duplex" connection between its endpoints.

The Directed Acyclic Word Graph (DAWG) is a directed edge graph that represents a set of strings, and allows for a search operation that tests quickly whether a given string belongs to the set. In these respects, a DAWG is very similar to the trie, but it is more space efficient (Inenaga, Hoshino et al. 2005). Using DAWG in forward and backward searching is easier than in the case of a trie. Figure 2.8 shows the previous simple lexicon of 12 strings of the English Gregorian months built using a DAWG. The green colour nodes are the starting nodes and the yellow colour nodes are the leaf nodes.

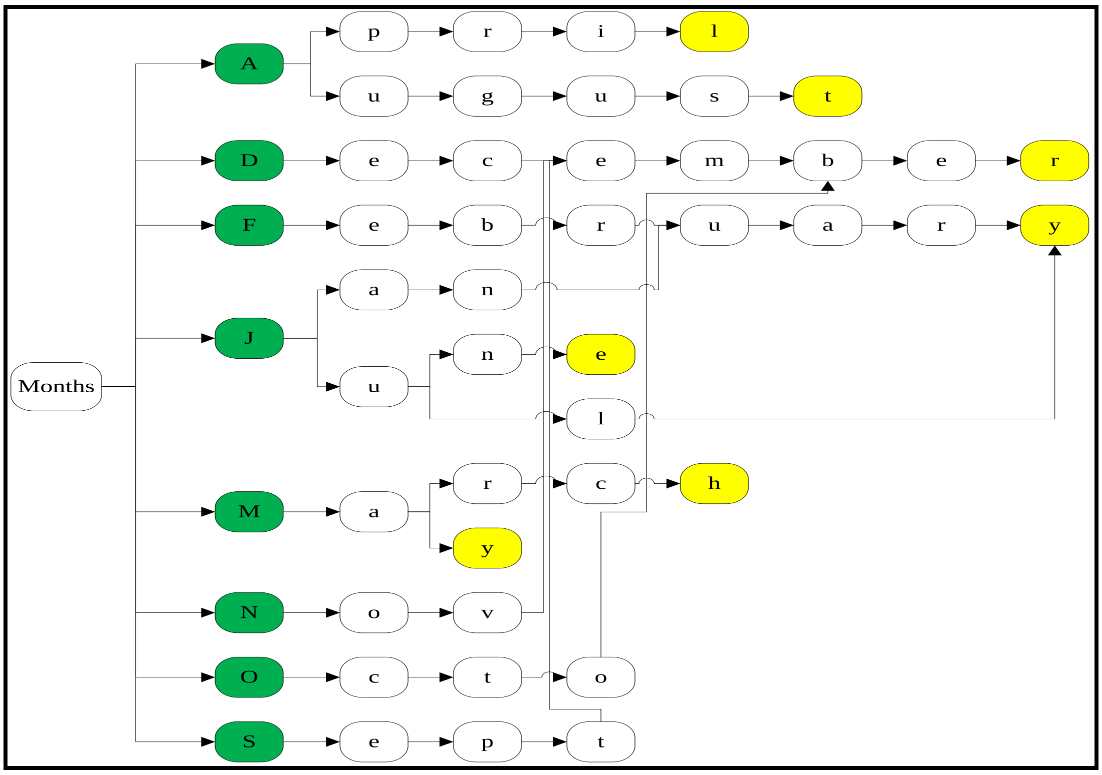


Figure 2.8: DAWG graph structure of a simple lexicon of 12 months

## 2.7.3 String correction

### *2.7.3.1 String correction – Viterbi algorithm*

The Viterbi algorithm is a dynamic programming algorithm designed to find the best sequence of hidden states, while the Viterbi path is the result of a sequence of observed events which could be applied to the problem of text recognition. The Viterbi Algorithm takes the output word from the OCR system and, in applying statistical methods on the sequence of letters in that language and likely errors from the OCR system, calculates the most likely input word (Ford and Higgins 1990).

Gillies *et al.* (Gillies, Erlandson et al. 1999) proposed a complete Arabic OCR application now available as a commercial application by the Novo-Dynamics company (Dynamics 2011). It uses the Viterbi algorithm in the post-processing stage. The Viterbi beam search module uses a dynamic programming algorithm in order to match the array of segments against a model of Arabic text. The model encodes the rules of Arabic typography.

### *2.7.3.2 String correction – Leveneshtien distance algorithm*

The Levenshtein distance algorithm is an algorithm used for measuring the difference between two string sequences. The Levenshtein distance between two strings is the minimum number of simple edits needed in order to transform one string to the other, with the edit operations being insertion, deletion, or substitution of a single character (Levenshtein 1966). This algorithm is used in order to measure the OCR system's accuracy (Tanner, Muñoz et al. 2009) and in post-processing techniques as a string correction algorithm (Schulz and Mihov 2002). The Levenshtein distance is also widely used in other different types of research fields such as: computational biology, signal processing, virus and intrusion detection, image compression, data mining, pattern recognition, file comparison and screen updating (Navarro 2001).

Levenshtein distance is one of the famous algorithms for string comparison, which is based on the notion of a primitive edit operation. In the standard Levenshtein distance, the primitive operations are *substitution* of one character for another character, *deletion* of a character, and *insertion* of a new character. Let $P$ and $W$ be two tokens in the alphabet $\Sigma$. The Levenshtein distance between $P$ and $W$, denoted

*$d_L(P,W)$*, is the minimal number of primitive edit operations (substitutions, deletions, insertions) that are needed in order to transform $P$ into $W$. The Levenshtein distance between two tokens $P$ and $W$ can be computed using the following simple dynamic programming scheme (Levenshtein 1966; Mihov and Schulz 2004):

$$d_L(\varepsilon, W) = |W|$$

$$d_L(P, \varepsilon) = |P|$$

$$d_L(Pa, Wb) = \begin{cases} d_L(P,W) & if\ a = b \\ 1 + \min\big(d_L(P,W), d_L(Pa,W), d_L(P,Wb)\big) & if\ a \neq b \end{cases}$$

for $P, W \in \Sigma^*$and $a, b \in \Sigma$.

Given $P = p1 \cdots\cdots p_m$ and $W = w1 \cdots\cdots w_n (m, n \geq 0)$.

The previous scheme is applied from the top down and from left to right on the cells of a (m + 1) × (n + 1) matrix. Figure 2.9 shows an example of the Levenshtein distance calculation matrix between two English words “Saturday” and “Sunday” which produced *$d_L$(*Saturday, Sunday*)*=3. The yellow colour cells show the equal characters, green colour cells show the deletion editing and the blue colour cell shows the substitution editing.

| | | S | a | t | u | r | d | a | y |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| S | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| u | 2 | 1 | 1 | 2 | 2 | 3 | 4 | 5 | 6 |
| n | 3 | 2 | 2 | 2 | 3 | 3 | 4 | 5 | 6 |
| d | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 4 | 5 |
| a | 5 | 4 | 3 | 4 | 4 | 4 | 4 | 3 | 4 |
| y | 6 | 5 | 4 | 4 | 5 | 5 | 5 | 4 | 3 |

Figure 2.9: Levenshtein distance between the two words ‘Saturday’ and ‘Sunday’

### 2.7.4 Language correction using corpus

A corpus is a large, structured collection of text covering a huge number of words from different domains of a given language. The first modern corpus was the Brown Corpus. It was collected and compiled in 1967 (Kučera and Francis 1967) and contains almost 1 million English words from different disciplines. Currently, the three best known English corpora are: The Corpus of Contemporary American English, which contains 410+ million words, is available via the internet and was created during the period (1990 – 2010). It is updated once or twice a year (Davies 1990-present); The British National Corpus, which contains 100 million words, is also available via the internet, was created during the period (1980 – 1993) and is still being updated (Corpus 2007); and the Time Archive (1923 - to present) from Time magazine. The Time Archive was originally created to archive the magazine's articles and contains more than 275,000 articles with around 100 million words (Time 2008).

Corpora are increasingly valuable in computational linguistics, as well as supporting statistical analysis of languages. They are vital to the development of computational techniques associated with languages such as OCR, speech recognition and natural language processing. Corpora used in these areas may include data beyond simple text, such as audio (speech) or images. These multimodal corpora are also gaining increased attention in mainstream corpus linguistics (Knight, Bayoumi et al. 2006).

Multimodal corpora are of particular value during the development of OCR applications. OCR methods must be trained with, and evaluated against, corpora which capture the structure of the target language, but emphasize its appearance in image data. As there is no central organization for the Arabic language, no standard linguistic or image corpus exists (Hamada 2007).

Stemming is a process of determining the morphological root of a word as well as a tool for data retrieval from a corpus to minimize the mismatching errors (Larkey, Ballesteros et al. 2002). This is done by removing affixes (prefixes, infixes or suffixes) from the word. The word may have many forms while retaining the same meaning, as for example the words (works, working, worker and worked) are all derived from the root word "work" in English. All languages contain nouns and verbs, for example in English the nouns (book, books) are derived from the root word "book" but one is singular and the other is plural. Also verbs like (play, played, playing) have the same root word "play" but in different tenses. A sophisticated lexicon need not keep all these alternative forms of words, but may be optimized to store only the root words and methods to obtain and report the derived words from them.

The Arabic language also has similar rules as English and other Latin languages. Arabic has nouns and verbs. Nouns like (كتاب كتابان كتب) are derived from the same root word (كتب) but the first is singular, the second is for a pair and the third is plural. Verbs like (لعب يلعب سيلعب) are also derived from the root word (لعب) but in different tenses.

#### *2.7.4.1 Collected word corpora*

The Linguistic Data Consortium (LDC) at the University of Pennsylvania has produced the "Arabic Gigaword Fourth Edition" (Parker, Graff et al. 2009). This is a database of 850 million Arabic words collected over several years from news agencies. Though important, it has a number of drawbacks. First, the corpus is collected only from news agencies, limiting the linguistic style of the material captured. Secondly, most of the files come from Lebanese news agencies. The lack of samples from other Arab countries limits the scope of any OCR application which relies on them. Finally, the corpus format is based upon paragraphs and not single words. This makes it less useful for testing and training of OCR methods.

The Institute of Ancient Near Eastern Studies, Charles University Prague (Czech Republic) has compiled the CLARA Corpus. This is an electronic corpus of Modern Standard Arabic. It contains 37 million words. Though the project started in 1997 (Zemanek 2001), the corpus lacks variety in the disciplines and geographical areas sampled. This lack of variety reduces its usefulness to the developers of OCR applications.

The University of Essex, in collaboration with the Open University, has developed a corpus of Al-Hayat newspaper articles. It contains 18.5 million words from 42,591 articles, but covers only 7 main subjects (newspaper 2002). The An-Nahar newspaper has also produced its own an An-Nahar text corpus. This contains 24 million words from 45,000 articles. It was developed between 1995 and 2000 (newspaper 2000). This type of corpus has some drawbacks which limit its value to OCR and, to a certain extent, corpus linguistics in general. First, they represent the home country of the newspaper, and not all Arab countries. Secondly, they sample only a small time period. Finally, they cover only the subjects discussed in a particular newspaper. Lack of variety in a corpus obviously reduces its usefulness to the developers of OCR applications.

The DIVA Group from the University of Fribourg (Switzerland) in collaboration with the REGIM Group from the University of Sfax (Tunisia) and the Software Engineering Unit of the Business Information System Institute (HES-SO //Wallis - Switzerland) recently generated the APTI Arabic Printed Text Image Database. This is a large-scale benchmark of open-vocabulary, multi-font, multi-size and multi-style text recognition systems in Arabic. Using 113,284 words with 10 different Arabic fonts, 10 Arabic font sizes and 4 font styles, APTI generated a total of 45 million words. Its variety of fonts, text sizes and styles; make APTI a valuable resource when testing Arabic OCR applications, but its sources of data lack variety and its image generation process relies upon down-sampling, which is not a good simulation of the real data. APTI can be used with OCR applications that recognize words only, and offers no support to applications working at the PAWs level (Slimane, Ingold et al. 2009).

#### *2.7.4.2 Morphological – created word corpora*

DIINAR.1 is an Arabic lexical corpus produced by the Euro-Mediterranean project. It comprises 119,693 lemmas distributed between nouns, verbs and adverbs. It uses 6,546 roots (Abbes, Dichy et al. 2004).

The Xerox Arabic Morphological Analyzer/Generator was developed by Xerox in 2001. This contains 90,000 Arabic stems, which can create a derived corpus of 72 million words (Beesley 1996). This type of corpora partially solves the problem of not having a definitive Arabic corpus, but it does not include many words used in everyday life.

#### *2.7.4.3 Lexical / stemming*

The Arabic language is rich and has a large variety of grammar rules. Research in Arabic linguistics is varied and can be categorized into four main types:

##### *2.7.4.3.1 Manually constructed dictionary*

A custom Arabic retrieval system is built depending on a list of roots and creates lists of alternative words depending on those roots. This method is limited to the number of roots collected (Al-Kharashi and Evens 1994).

##### *2.7.4.3.2 Morphological analysis*

This is an important topic in natural language processing. It is mainly concerned with roots and stemming identification. It is related more to the grammar of the word and its positioning (Al-Shalabi and Evens 1998; Al-Shalabi and Kanaan 2004; Jomma, Ismail et al. 2006).

##### *2.7.4.3.3 Statistical stemmer*

This does not depend on the language involved but on the similarity of rules among different languages. A stemmer generator has been developed using a parallel corpus which is a collection of sentence pairs with the same meaning in different languages (Rogati, McCarley et al. 2003).

##### *2.7.4.3.4 Light stemmer*

This depends on affix removal. An affix is defined here as one or more letters added to the root word. This type of stemmer needs less knowledge of Arabic grammar (Aljlayl and Frieder 2002; Larkey, Ballesteros et al. 2002).

#### *2.7.4.4 Requirements and limitations*

To adequately represent a given language a corpus must have sufficient capacity of text collection (Hamada 2007) and include words from a wide variety of sources and types (Wynne 2005; Alansary, Nagi et al. 2007; Hamada 2007). It is known that a good corpus should include (Dash and Chaudhuri 2001):

- Language from different disciplines, e.g. engineering, literature, art and medicine.
- Popular language from everyday life.
- Different morphological features of the language. These features are gender (male or female), tense (past, present or future), number (singular, pair or plural),

person (first, second or third), imperative verb and definiteness (Al-Shalabi and Evens 1998).

- Examples created over a long period of time. The Arabic language has been in use for 1,500 years and very old poems, fairy tales, religion and science books are available.
- Examples from varying geographical areas. All the different dialects of the Arabic speaking countries and regions should be covered.
- Images of the contents of the corpus with different image sizes and formats that can be used with OCR applications.

These features are also required of corpora intended to provide good training / testing sets for OCR.

Corpora should provide flexible manipulation and retrieval of data. This is eased by the use of lower level representations; language stored as words, for example, can be accessed in more ways than that held only in paragraphs or full pages. Again, similar criteria apply to multi-modal corpora created to support research in OCR.

The corpora discussed above each fail to meet one or more of these criteria. Moreover, previous Arabic corpora are purely linguistics. A substantial literature review has failed to locate any freely available corpora of images of printed Arabic. Note also that existing corpora operate exclusively at the word level. Though words are clearly important, the structure of the Arabic language and the challenges it presents to OCR suggest that this is not sufficient basis upon which to build a truly effect multi-modal Arabic corpus.

## 2.8 Summary

This chapter introduced a review study of relevant OCR applications. It gave a brief introduction to the OCR applications history and the research topics related to it. It explained the different categories of readable information with the description of the best OCR commercial applications. The established OCR application phases were also introduced. This review study will be referred to in the other chapters of this thesis when selecting appropriate algorithms. The topics discussed in this chapter are:

1. An introduction to OCR with its history. The relationship with other research topics. Different categorizations of readable information. The categorizations are based on printing methods, language and being online or offline. The best OCR commercial applications.

2. An overview of the written Arabic language. The problems occurring when recognizing Arabic written text. Importance of PAWs and NPAWs to the recognition of Arabic text. Description of the Arabic language morphology.

3. The pre-processing stage with its inclusive steps. Steps include Document analysis, binarization, thinning, skew correction, baseline detection and word normalization.

4. The feature extraction phase with samples of common algorithms. The algorithms presented are moment, Hough Transform, Fourier descriptor, linear transform and Haar-like features.

5. The character segmentation phase with the different types of character segmentation used in the OCR. The types discussed are isolated/pre-segmented characters, segmenting a word into characters, segmenting a word into primitives, integration of recognition and character segmentation and finally character segmentation-free systems.

6. Classification phase with samples of common classification methods. The methods presented are statistical methods with Hidden Markov Model, structural methods, Artificial Neural Networks, Support Vector Machine, classification with multi-classifiers, cascade of boosting classifiers and Integration of feature extraction and classification.

7. Post-processing phase with its detailed methods. The methods presented are using confusion matrix, look-up dictionary, string correction and language correction using corpus.

# Chapter 3.

# A Multi-Modal Arabic Corpus

A corpus is a large structured set of texts, electronically stored and processed. Corpora have become very important in the study of languages and have opened new areas of linguistic research which were unknown. Corpora are also considered as a key to the development of OCR applications (Hamada 2007). Access to a corpus of both language and image is essential during OCR development, particularly while training and testing a recognition application. Excellent corpora have been developed for Latin based languages, but few for the Arabic language. This limits the penetration of both corpus linguistics and OCR in Arabic-speaking countries. This chapter describes the construction, and provides a comprehensive study and analysis, of a Multi-Modal Arabic Corpus (MMAC) which is suitable for use in both OCR development and linguistics. MMAC currently contains six million Arabic words and, unlike previous corpora, also includes connected segments or Pieces of Arabic Words (PAWs) as well as Naked Pieces of Arabic Words (NPAWs) and Naked Words (NWords); PAWs and Words without diacritical marks. Multi-modal data is generated from both text, gathered from a wide variety of sources, and images of existing documents. Text-based data is complemented by a set of artificially generated images showing each of the Words, NWords, PAWs and NPAWs involved. Applications are provided to generate a natural-looking degradation to the generated images. A ground truth annotation is offered for each image, while natural images showing small paragraphs and full pages are augmented with representations of the text they depict. A statistical analysis and verification of the corpus has been carried out and is presented. MMAC was also tested using commercial OCR software and is publicly and freely available.

Lexicon and stemming tools are very important in enhancing corpus retrieval and performance in an OCR context. This chapter also presents a lexicon / stemming algorithm based on a new algorithm which uses a light stemmer type for developing the algorithm. This part is considered as a tool for enhancing the development of the MMAC and not as a part of it. Lexicon and stemming lookup is combined to obtain a list of alternatives for uncertain words. This list removes affixes (prefixes, infixes or suffixes) if there are any, if not, adds affixes to the uncertain word. Finally, it tries to verify every word in the list of alternatives by searching the original corpus. A tool is added to continually improve the corpus by adding new words and justifying them using the lexicon / stemming algorithm.

This chapter describes the generation of a corpus containing six million Arabic words and associated image data. Compared to earlier Arabic corpora, this corpus has a number of advantages which make it useful in a wider range of applications areas. The lists and words it contains are in a variety of formats, allowing access and retrieval in diverse ways. It allows access via PAWs (Amin 1997; Lorigo and Govindaraju May, 2006) as well as NPAWs and NWords. A set of artificially generated images showing each of the tokens found in the corpus is provided, with each token shown in three key Arabic fonts. A set of applications is provided which add noise, generating degraded images that simulate real images of Arabic text. Each image is accompanied by annotations in XML format, creating a valuable ground truth resource. The original images and degraded copies are beneficial when testing document image analysis applications, and make MMAC a real benchmark for Arabic OCR projects (AbdelRaouf, Higgins et al. 2008). The corpus also includes real images of the Arabic documents used in its production, a necessary addition for OCR usage. These documents are in full page and single paragraph formats. The three different image formats - single token, paragraph and full page - increase the usability of MMAC. The corpus' value as a benchmark dataset is evaluated using Readiris Pro 10 (IRIS 2004), a well-known commercial software package useful as an exemplar of Arabic OCR applications. Finally, statistical analysis of the corpus' contents is also provided. The corpus is freely available from http://www.ashrafraouf.com/mmac.

This chapter presents a new lexicon/stemming approach based on the idea of the Viterbi path Algorithm to enhance the corpus retrieval. The approach is based on applying the lexicon and stemming at the same time. It generates a list of alternative tokens to the original token. If the token includes an affix it generates a list of alternative tokens after removing the affix. If the token is a root case, it generates a list of alternative tokens by adding affixes.

## 3.1 Building a Multi-Modal Arabic Corpus from textual data

The textual data of the corpus was built in a way to cover the requirements of building a benchmark corpus. It was built to be sure of the validity of its contents and generates output that suits many users' needs.

### 3.1.1 Coverage and Sources

The Multi-Modal Arabic Corpus contains 6 million Arabic words selected from various sources covering old Arabic, religious texts, traditional language, modern language, different specialisations and very modern material from online "chat rooms". The total number of words in the corpus was obtained from the studying of Figure 3.9 which indicates that at 6 million words, the increase of the new unique words to the corpus becomes trivial. The sources used are:

*Topical Arabic websites:* These were obtained via an Arabic search engine. The search engine specifies the web-pages according to topic. Pages allocated to different topics including literature, women, medicine, business, children, religion, sports, programming, design, and entertainment were downloaded and incorporated into MMAC.

*Arabic news websites:* A set of the most widely used Arabic news websites were identified and downloaded. These included the websites of Al Jazeera, Al Ahram, Al Hayat, Al Khaleej, Asharq Al Awsat, Al Arabiya and Al Akhbar. Professional news websites include language which is qualitatively different from that used in the first set of websites, which were created by the general public. They also come from a variety of Arabic countries.

*Arabic chatrooms:* Some Arabic chatrooms were used for areas such as sports, culture and social interaction. These chatrooms represent the popular language part of the corpus.

*Arabic-Arabic dictionaries:* These dictionaries contain the most common Arabic words and word roots, along with their definition.

*Old Arabic books:* These were generally written centuries ago. They include religious books and books using traditional language.

*Arabic research:* A PhD thesis in Law was included to sample the research literature.

*The Holy Quran:* The wording of the Holy Quran was also used.

### 3.1.2 Text data collection

The following steps were taken to overcome the difficulties discussed previously (see section 2.2.2), which are mainly due to the use of different code page encodings and words often containing repeated letters or being formed from connected words. The steps are:

1. An Arabic search engine (arabo.com 2005) was used to search for Arabic websites. WebZIP 7.0 software was then used to download these websites (Ltd 2006) by selecting files that contain text, markup and scripting.

2. The code page used for each part of the file was identified. Sometimes more than one code page is used.

3. A program was developed which removes Latin letters, numbers, and any non-Arabic letters even if the letters are from the Unicode Arabic alphabets like this symbol used in Holy Quran (۝). This program also removes any diacritics.

4. Some common typographical errors were corrected. For example, words containing (ة or ى) inside but not at the end were corrected. Also, words containing those two letters (ىء) at the end were corrected to a single letter (ئ).

5. A text file was created containing one Arabic word per line.

6. A program was written to correct the problem of connected words. It scans the words and delivers a list of words with size more than 7 letters. The list is checked manually and corrected. The program rewrites the corrected words in the list of words.

7. Words containing repeated letters were checked automatically and then manually. A program was written to list all the words that include any repeated letters. The list was checked manually and corrected. The program rewrites the corrected words in the list.

8. A program was written and applied which replaces the final letters (ى) with (ي), (ة) with (ه), and (أ إ آ) with (ا) (Larkey, Ballesteros et al. 2002).

### 3.1.3 Creating NWords, PAWs and NPAWs

The words obtained by the process outlined in section 3.1.2 allow generation of NWords, PAWs and NPAWs data files. This is achieved as follows:

1. A program to create NWord data files was developed. This reads each word sequentially and replaces each letter from the group letters with its joining group letter as shown in Table 2.1, for example it replaces (ج ح خ) with (ح).

2. A program to create PAW data files was developed which reads each word sequentially from the words data file, checks if the word contains one of the six letters (ا د ذ ر ز و) in the middle of the word, and if so puts a carriage return after it. The same is done with the Hamza (ء) but it also put carriage return before it. The carriage return thus separates this PAW onto a new line. Finally it saves the new PAWs file.

3. The program used to convert Words to NWords was used to transfer PAWs to NPAWs. The program is applied to the PAWs data file and generates a NPAWs data file.

### 3.1.4 Creating images from corpus text data

The methods described in sections 3.1.2 and 3.1.3 produce a text-based Arabic corpus. To support the development of Arabic OCR applications, MMAC augments its textual data with images (Kanungo and Resnik 1999) and ground truth information for these images (Slimane, Ingold et al. 2009). Each token (Word, NWord, PAW or NPAW) is shown in the three most common Arabic fonts; Simplified Arabic, Arabic Transparent, and Traditional Arabic (Chang, Chen et al.

2009). Token images are stored in grey-scale Windows bmp file format, with a resolution of 300DPI (Kanoun, Alimi et al. 2005). Figure 3.1 gives an example of a word represented as images showing its appearance in three common Arabic fonts.

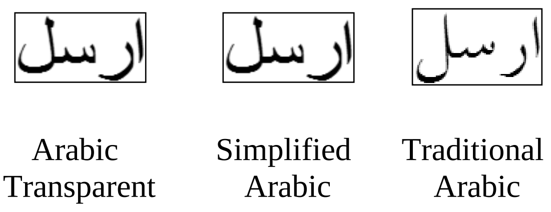


Figure 3.1: A word images of (ارسل) in three different Arabic fonts

The data collection process produced a data file containing the unique Word, NWord, PAW and NPAW tokens, sorted alphabetically. To produce the image files each token was transfer from this text file to a Windows bmp image file. A program was developed to achieve this as follows:

1. The program opens the text file containing the list of unique tokens using the Windows CP1256 character encoding. It reads the file sequentially and writes each token to bitmap memory allocation with 300DPI resolution. The font format is changed to the three most common fonts. The font size is 14 point.

2. The program saves the bitmap memory allocation to disk in Windows bmp grey-scale format. It names the resultant image with a sequential number, for example Word0000045Font1.bmp. Each token generates three different image files, one for each of the three fonts.

3. The program generates a ground truth XML file for each image file from the previous files containing all the information about the token. The information saved contains both text and image information.

The final token image dataset consists of files containing the images of the token. Each token has three font files, Simplified Arabic, Traditional Arabic and Arabic Transparent. To simplify handling of the huge number of files, the program saves every 500 tokens files in a separate sub-folder. Table 3.1 shows the total number of images of the different tokens that are generated from the list of unique tokens.

Table 3.1: Total Number of images for Words, NWords, PAWs and NPAWs

| | Number of Unique Tokens | Total Number of Images |
|---|---|---|
| Words | 282,593 | 847,779 |
| Naked Words | 211,072 | 633,216 |
| PAWs | 66,725 | 200,175 |
| Naked PAWs | 32,804 | 98,412 |

Table 3.2 shows the average letters per token, average width and average height of the Words, NWords, PAWs and NPAWs. The information about the token images shows that the average width of the words and NWords are the same while the height is different due to the existence of dots. This is also recorded in the case of the PAWs and NPAWs. The ratio between the average widths of the Words and NWords on one side and PAWs and NPAWs on the other is almost the same as in the case of the number of letters per token. The average height of words and NWords is greater than that of PAWs and NPAWs.

Table 3.2: The average letters per token, width and height in pixels of tokens

| Token | Average | | |
|---|---|---|---|
| | Letters / token | Width | Height |
| Words | 4.7 | 93.1 | 48.1 |
| Naked Words | 4.7 | 93.1 | 47.0 |
| PAWs | 2.0 | 39.5 | 38.9 |
| Naked PAWs | 2.0 | 39.4 | 38.0 |

### 3.1.5 Degrading the computer-generated token images

The process of degrading the computer generated token images is very important if they are to simulate the characteristics of real document images during OCR development. The degradation stage is composed of two parts: in the first part, the image is skewed to simulate the distortion that may occur during document scanning, in the second, artificial noise is added to simulate scanned image data.

#### *3.1.5.1 Skewing the image*

Rotated images are created from the computer generated image to simulate the rotation that often occurs while scanning documents. A computer program was developed to generate any number of rotated samples from the original image with rotation in both sides. It uses the Box-Muller transform algorithm to convert the uniform distribution random number to normal distribution random number (Box and

Muller 1958). This algorithm is used to simulate the real life rotation angles of scanned documents. A nearest neighbour interpolation algorithm applies the rotation to the images (Sonka, Hlavac et al. 1998). Figure 3.2 (b) shows the skewed token image.

#### *3.1.5.2 Adding noise*

Artificial noise is added to the image to simulate that found in real document images. Three effects are applied; blurring, filter and additive noise (Hartley and Crumpton 1999). A Gaussian blurring effect is first added to simulate the blur that may be occurring during scanning. A high pass filter is then applied to simulate change in the paper colour over time. Gaussian noise is finally added to simulate image acquisition noise. Figure 3.2 (c) shows a token image after applying the Gaussian blurring and noise.

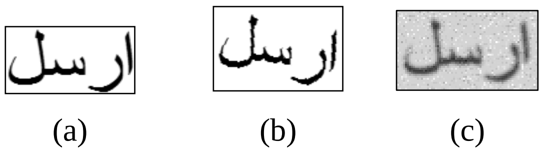


Figure 3.2: (a) word (ارسل) image. (b) Skewed image. (c) Noised image

### 3.1.6 Ground truthing the computer-generated token images

In document image analysis and recognition, *ground truth* refers to various attributes associated with the text on the image such as the size of tokens, characters, font type, font size and so on. Ground truth data is crucial to the systematic training and testing of document image analysis applications (Leea and Kanungob 2003).

Each token image in the MMAC contains ground truth information (Slimane, Ingold et al. 2009). Figure 3.3 shows the ground truth XML file for the word (هذا). MMAC's ground truth XML files contain the following attributes:

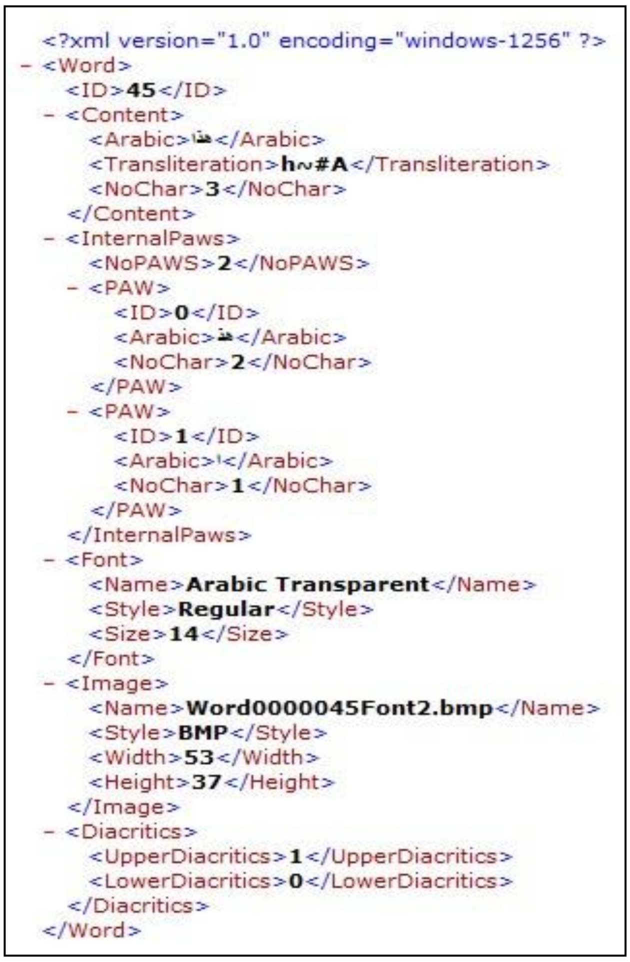
1. *Content:* this contains the token name in Arabic, transliteration of the token text to English – the transliteration method applied will be shown in section 3.7.3 - and the number of characters in the token.

2. *Internal PAWs:* this shows the number of PAWs of the Word or Nword. It contains, as a sub attribute, the content of the PAWs, these contents are the ID, Arabic name and number of characters.

3. *Font:* this contains the name of the font, style and font size in points.

4. *Image:* this contains the image file name, the file format of the image and the size of the image in pixels.

5. *Diacritics (Dots):* this contains the number of letters in the token that have upper diacritics and that have lower diacritics.



```
  <?xml version="1.0" encoding="windows-1256" ?>
- <Word>
    <ID>45</ID>
  - <Content>
      <Arabic>هذا</Arabic>
      <Transliteration>h~#A</Transliteration>
      <NoChar>3</NoChar>
    </Content>
  - <InternalPaws>
      <NoPAWS>2</NoPAWS>
    - <PAW>
        <ID>0</ID>
        <Arabic>هذ</Arabic>
        <NoChar>2</NoChar>
      </PAW>
    - <PAW>
        <ID>1</ID>
        <Arabic>ا</Arabic>
        <NoChar>1</NoChar>
      </PAW>
    </InternalPaws>
  - <Font>
      <Name>Arabic Transparent</Name>
      <Style>Regular</Style>
      <Size>14</Size>
    </Font>
  - <Image>
      <Name>Word0000045Font2.bmp</Name>
      <Style>BMP</Style>
      <Width>53</Width>
      <Height>37</Height>
    </Image>
  - <Diacritics>
      <UpperDiacritics>1</UpperDiacritics>
      <LowerDiacritics>0</LowerDiacritics>
    </Diacritics>
  </Word>
```

Figure 3.3: Ground truth XML file for the word (هذا)

## 3.2 Building a Multi-Modal Corpus: real images

The procedures described in section 3.1 produce a multi-modal Arabic corpus from text data, with images being created artificially from that data. The result is a set of files containing descriptions of key sections of Arabic text, with corresponding image files showing their appearance in a number of common fonts. This dataset provides both a representation of the Arabic language and data needed during development of OCR applications – sample images of text with associated XML ground truth information.

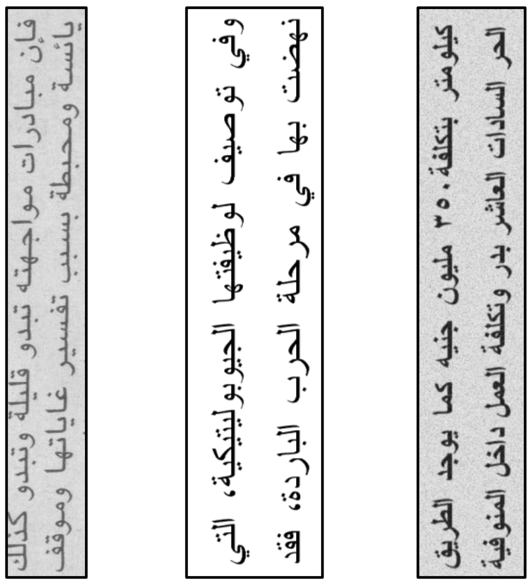
The images generated by the method described above are, however, idealised. As Figure 3.1 showed, the token images are so clear that they can appear as if they were typed. Though the applications described in the previous section allow common distortions and noise processes to be simulated, a complete evaluation should include real images containing the errors, noise and distortions that arise in real life.

To provide the data needed to support OCR, two different datasets are added to MMAC. The first contains images of small paragraphs of Arabic documents with text files providing ground truth. The second dataset is created from real scanned images. These datasets can be used to check the validity of the output of a developing OCR application. They are also valuable during development of pre-processes such as skew detection and correction, noise removal, binarization, thinning and layout analysis.

### 3.2.1 Paragraph image dataset

MMAC's paragraph document dataset contains images of 552 paragraphs. It is generated from 15 full page documents. There are three different categories: real scanned images, computer generated images and computer generated images with artificial noise. Each category contains 5 full pages. Each single paragraph image includes around two lines containing around 10 words. The number of images are 223 real, 141 computer generated and 188 computer generated with noise. Figure 3.4 shows a sample paragraph image from each category. The computer-generated and computer-generated with noise documents include different Arabic font type, sizes and styles (regular, italic and bold). This dataset includes around 8,000 words.

The need for this dataset in the OCR development process is to test the application with a small sample of data which enables tracing any errors. Interestingly, during the testing of MMAC with commercial software, the recognition accuracy of a paragraph image is sometimes different from that of the full page image.



فإن مبادرات مواجهته تبدو قليلة وتبدو كذلك
يائسة ومحبطة بسبب تفسير غاياتها وموقف

وفي توصيف لوظيفتها الجيوبوليتيكية، التي
نهضت بها في مرحلة الحرب الباردة، فقد

كيلومتر بتكلفة ٣٥٠ مليون جنيه كما يوجد الطريق
الحر السادات العاشر بدر وتكلفة العمل داخل المنوفية

(a) Real image | (b) Computer image | (c) Noise image

Figure 3.4: Samples of the three categories of paragraph images

### 3.2.2 Full page real image dataset

MMAC's scanned document dataset contains images of 19 different documents. These comprise books, work documents, medical scripts, faxes, magazines and newspapers. The documents were scanned at 300DPI, in 24bit RGB colour mode and stored in tiff file format. Document quality varies significantly over the dataset, which includes documents at different orientations and some skewed examples. It contains different sizes of documents. Some of the documents are watermarked, others have large page borders. Figure 3.5 shows two examples. The first example is a newspaper image with some skew and images imposed inside the document. The second example is a medicine script with some skew and including tables. This image is rotated 90°. Though the scanned document dataset is currently small compared to the text-based data, the goal is to include examples of most (commonly) occurring real documents.

To create associated text files, the contents of each document were manually typed on Microsoft Word 2003 software. The resulting text files were proofread by two different groups of people, with the writing group different from each of the two proofreading groups. The dataset created from real images currently includes around 11,000 Arabic words.

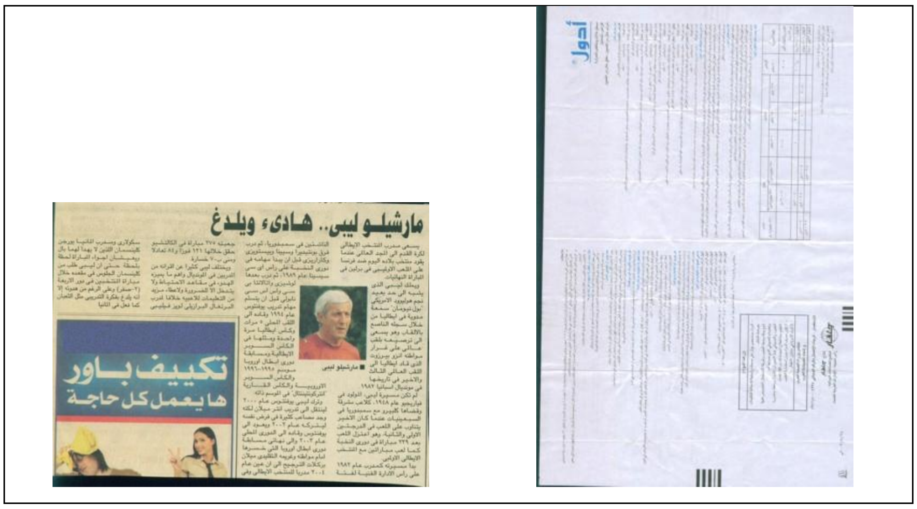

مارشيلو ليبي.. هادىء وبليغ
تكييف باور
هايعمل كل حاجة

Figure 3.5: Two samples of real scanned document images

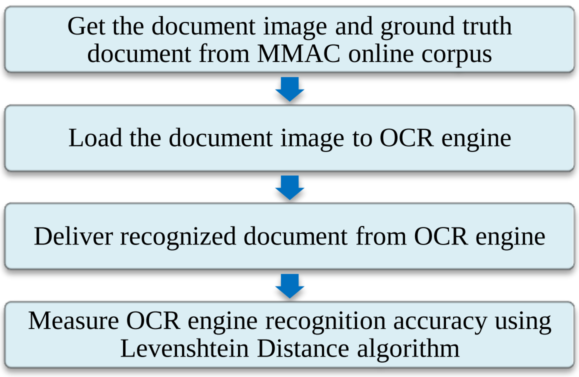
## 3.3 Using MMAC in Arabic OCR development

MMAC was originally generated to support this research in Arabic OCR. The primary motive was to create a benchmark against which Arabic OCR applications could be evaluated. Its inclusion of images of different tokens makes MMAC suitable for training and testing OCR applications that operate in different layers. Some OCR applications deal with the whole word, while others deal with PAWs (Najoua and Noureddine 1995; Mehran, Pirsiavash et al. 2005). In the same manner some applications use diacritics to search for particular words, while others neglect them (Bushofa and Spann 1997). MMAC was generated to be used to evaluate any Arabic OCR application using data at four different levels that start from accurate computer generated images and move to real images of full page scanned documents. These four different levels represent different degrees of difficulty for an OCR application. As the application passes each level successfully and moves to the next, it gains more confidence in the quality of the application.

MMAC was tested with commercial software to check its usefulness in a real world application. The four different levels were tested using the well-known commercial software Readiris pro 10 (IRIS 2004). Recognition results from Readiris were compared to the ground truth documents using the Levenshtein Distance algorithm

(Navarro 2001). The Levenshtein Distance algorithm is an approximate string matching algorithm which gives a tentative accuracy of recognition (Levenshtein 1966). This accuracy is measured based on the percentage of correct characters within the volume of characters covered, defined by the majority of the OCR application vendors (Tanner, Muñoz et al. 2009). Figure 3.6 explains the process applied to test MMAC with the commercial software. This process can be used to test any OCR application with MMAC.

It is important to note that MMAC has been downloaded by both commercial companies and researchers after it was published on the internet. MMAC role was to select the best OCR application between the different applications tested. The variety of data in MMAC was used to test the applications. A clear justification for the best OCR application was produced. MMAC satisfied the needs of the commercial company in selecting the proper OCR application.


Get the document image and ground truth document from MMAC online corpus
Load the document image to OCR engine
Deliver recognized document from OCR engine
Measure OCR engine recognition accuracy using Levenshtein Distance algorithm
Figure 3.6: The typical process of testing OCR application using MMAC

### 3.3.1 Computer generated token images

In the first step MMAC's computer-generated images are used to test recognition accuracy (Figure 3.2). The challenge of MMAC in this step is to maximize the number of token images that are covered by the developing application. In addition to the commonly used PAWs and Words, MMAC can be used to evaluate OCR applications that deal with NWords or NPAWs. As the token images are very clear, it is expected that many applications will be capable of high levels of performance. The importance of this step is to evaluate OCR application with a wide variety of tokens that represents most of the language. This step gives credibility to the OCR application.

In the experiment reported here the 1,274 most commonly used words were selected. These words represent approximately half of the corpus words. A recognition accuracy of 94% was achieved.

### 3.3.2 Computer generated token images with artificial noise

In the second step degraded images are generated from the clean computer generated originals as shown in section 3.1.5. When an OCR application achieves high accuracy at this stage it shows that the OCR application can not only detect most of the tokens in the language but can also detect them in the presence of realistic noise. The ability to control the noise added allows detailed analysis of the effect of different image conditions on any proposed method. Extreme values can also be used to test a given technique “to destruction”.

In the exemplar experiment, three different levels of rotation and noise were applied to the document images. They represent images of good document, a medium document and a bad document. Table 3.3 gives the exact parameters used which were empirically derived after testing many values. The same 1,274 most common words from the previous step were examined. Recognition accuracy was 91% for good documents, 89% for medium documents and 77% for bad documents.

Table 3.3: Artificially added noise to computer generated documents

| | Good document | Medium document | Bad document |
|---|---|---|---|
| Rotation angle | ±2° | ±6° | ±10° |
| Gaussian Blurring | 10% | 13% | 15% |
| High Pass Filter | 35% | 45% | 50% |
| Gaussian Noise | 40% | 65% | 75% |
| Recognition accuracy | 91% | 89% | 77% |

### 3.3.3 Paragraph documents

In the third step, MMAC’s small images of real scanned, computer generated and computer generated with artificial noise documents are used to test the OCR application. Figure 3.4 shows examples of the available images. This step represents a higher level of testing than the previous two. It employs bigger images and different types of images. The recognition accuracy for this step gives a better indication of the likely quality of the OCR application given real-world input.

62 documents were selected from the three different types of documents that represent the majority cases. The recognition accuracy of the real documents ranged from 100% down to 47%. The recognition accuracy of the computer generated documents ranged from 93% down to 33%, while that for computer generated with noise documents ranged from 95% down to 54%. The variety of accuracy values are based on the status of the document images and font properties. It was found that the Italic font style gives bad accuracy results. It was also found that Traditional Arabic font gives comparatively poor accuracy results. MMAC's breadth allows the effect of a wide range of text and image features to be assessed.

### 3.3.4 Real scanned documents

In the final step, MMAC's images of real scanned documents are used to test the OCR application. The real scanned images are ground truthed by the documents that were scanned. These images show a variety of document types from a wide range of sources. Figure 3.5 shows examples of the available images. Passing this step with reasonable recognition accuracy gives the final evaluation of the OCR application. This step tests not only the recognition accuracy but also the overall performance of the OCR application. It probes the effect of page skew, page layout analysis, page orientation and so on. This step is considered to be the final step because the objective of any OCR application is to recognize full page documents with different aspects.

During testing this step, 9 different documents were selected representing different clarity and variety of images. The recognition accuracy achieved ranged from 95% for good documents down to 46% for very poor documents. This wide range of recognition accuracy indicates the usefulness of the large variety of images, documents and added noise provided by MMAC in testing OCR applications.

## 3.4 Enhancing MMAC using a lexicon/stemming approach

The approach prepared to guarantee that the MMAC can be improved using lexicon/stemming algorithm. This approach helps to facilitate an easy and acceptable way to continuously update MMAC with new words. The approach adds new words by presenting the different available linguistics information concerning these words. For every new word, the approach gives the following information:

- The existence of the new word in MMAC. This means that the word can be added to the corpus without any problems.
- The existence of an alternative word in MMAC. This is a less accurate word but it might also be right.
- Finally, the non-existence of neither the word nor its alternatives in MMAC. As a result, this word must be checked carefully to know if it is a right or not.

The approach offers a tool that allows adding a list of new words to MMAC corpus and regenerates MMAC content files automatically.

### 3.4.1 Approach applied in solving the problem

The two main types of stemming are light stemming and morphological stemming. The light stemmer type was applied for the following reasons:

1. Light stemmers give more accurate results (Aljlayl and Frieder 2002).
2. Morphological stemmers depend on regular language rules. This may be applicable in the English language because of the relatively small number of irregular rules, but it is inapplicable in the Arabic language because of the huge number of irregular rules.
3. Morphological stemmers sometimes depend on diacritics to extract the root. Diacritics are not included in this research.
4. A light stemmer does not need good knowledge of the language grammar rules.

### 3.4.2 The proposed light stemmer affixes

In the Arabic language, the root may be prefixed, infixes or suffixed by one or more letters. These letters have no meaning if written separately. Prefixes are letters added to the beginning of the word like (عمل تعمل). Infixes are letters added to the middle of the word like (عمل عمال). Suffixes are letters added at the end of the word like ( عمل عملي). Most of the light stemmers remove prefixes and suffixes only.

The proposed light stemmer is based on the list of Light1, Light2, Light3, Light8 and Light10 (Larkey, Ballesteros et al. 2002) affixes as shown in Table 3.4. Shalabi (Al-Shalabi and Kanaan 2004) created a morphological lexicon that included some affixes in the stemming process using the morphological rules. Shalabi affixes that are not used in other light stemmers are used. During the testing of the proposed stemmer, it had been found that using these affixes is useful. It was clear that more affixes were needed to improve its performance. Table 3.4 lists the different types of light stemmers and Shalabi's affixes. It also shows all affixes used to enable more words to be detected by the corpus.

Table 3.4: Affixes of different light stemming types

| Stemmer | Prefixes | Suffixes |
|---|---|---|
| Shalabi | ي | ا، وا، ت، نا، ن، تما، تن، تم |
| Light 1 | ال، وال، بال، كال، فال | None |
| Light 2 | ال، وال، بال، كال، فال، و | None |
| Light 3 | " | ه، ة |
| Light 8 | " | ها، ان، ات، ون، ين، يه، ية، ه، ة، ي |
| Light 10 | ال، وال، بال، كال، فال،لل ،و | " |
| Proposed Stemmer | ي، ال، وال، بال، كال، فال، لل،و ،يا ،يال | ا، وا، ت، نا، ن، تن، تم، ها، ان، ات، ون، ين، يه، ه، ي، هم، هن، كم |

## 3.4.3 The proposed lexicon / stemmer algorithm

The proposed lexicon / stemmer algorithm is based on the light stemmer algorithm. It creates a list of alternative words instead of the missing word from the corpus. It searches the corpus for the words from the alternative list. The searching algorithm for the missing words inside the corpus is a binary search.

Generating an alternative list of words is the important part in the approach. The alternative list is a list of all the words that might be derived from the original word. This part of the algorithm deals with the words from the testing dataset which are missing in the corpus. It applies the following procedures:

1. It checks the first and last letters of a word. It looks for the prefixes and suffixes letters shown in Table 3.4.

2. It creates a two-dimensional array of characters of size 25*10. 25 characters are used because it is more than the maximum Arabic word length and 10 characters because this is the maximum number of affixes that can be used with a word.

3. It creates a one-dimensional array of 25 characters. This array keeps the path to be followed to generate the alternative word from the other two-dimensional array.

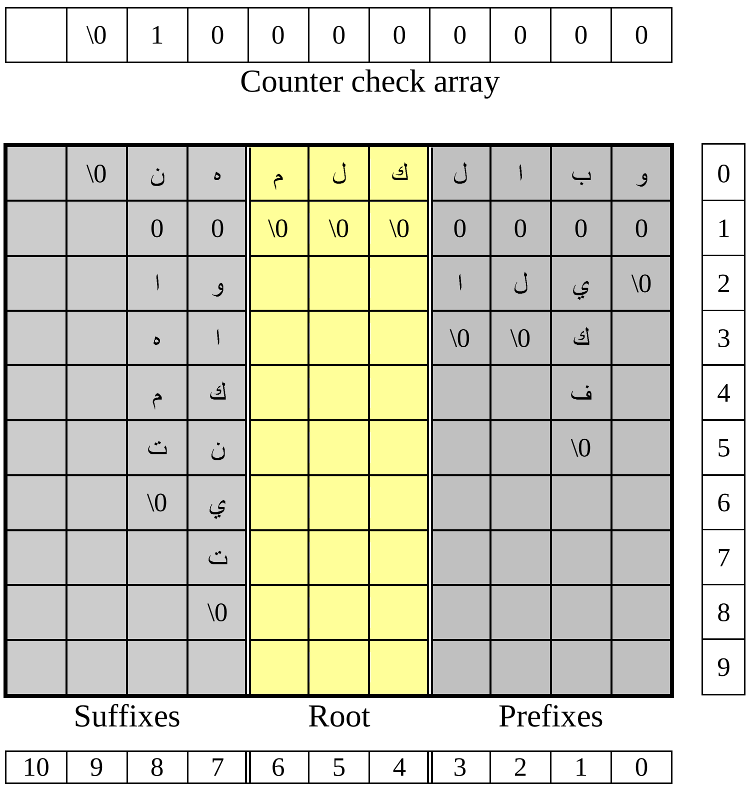
4. The two dimensional array is used to generate all possible alternatives, by adding prefixes or suffixes to the word.

5. In the case of affix with value ‘0’, it runs the search without this character. For example in the sample in Figure 3.7, the first character is (و). The algorithm will add all the paths once with (و) to the alternative list and once more without it. This means going through all the paths once with the affix and once more without it. It will ignore the letter when its value is '0'.

6. It starts getting all the possible paths from the two-dimensional array using the idea of a Viterbi path algorithm. It stops in any column whenever the '\0' character is found.

7. It adds each word created in each possible path to the list of alternative words to the missed word.

Figure 3.7 shows the algorithm used to generate a list of alternative words to a sample root word (كلم). The three letters in the middle are the root word. The letters on the right side show all the possible prefixes. The letters on the left side show all the possible suffixes. The counter check array keeps the location of the path to be followed to generate the alternative words.



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | \0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Counter check array

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | \0 | ن | ه | م | ل | ك | ل | ا | ب | و | 0 |
| | | 0 | 0 | \0 | \0 | \0 | 0 | 0 | 0 | 0 | 1 |
| | | ا | و | | | | ا | ل | ي | \0 | 2 |
| | | ه | ا | | | | \0 | \0 | ك | | 3 |
| | | م | ك | | | | | | ف | | 4 |
| | | ت | ن | | | | | | \0 | | 5 |
| | | \0 | ي | | | | | | | | 6 |
| | | | ت | | | | | | | | 7 |
| | | | \0 | | | | | | | | 8 |
| | | | | | | | | | | | 9 |
| Suffixes | | | | Root | | | Prefixes | | | | |
| 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |

Figure 3.7: The structure of the lexicon / stemming algorithm

## 3.4.4 Implementing the lexicon / stemming approach

A program was implemented to apply the previous algorithm to the corpus words data files. A script programming application was developed using MS DOS batch scripting to generate the required overall application. The script uses the previous program that was developed to apply the lexicon/stemming algorithm. The script also uses the programs developed to generate the corpus data files.

The input here is a list of the new words required to be added to the corpus or to be checked if they are included in the corpus or not. The program generates two lists of words, a list of words found in the corpus and a list of words missed from the corpus.

The list of words missed from the corpus is divided into two different lists. The two lists are a list of trusted words and a list of un-trusted words. The list of trusted words is the list of words that have an alternative word in the corpus. The list of un-trusted words is the list of words that have not got alternative words in the corpus.

The user is required to edit the two lists to double check manually with the words that are going to be added to the corpus. After verifying the list of words to be added to the corpus the application loads the new list of words into the corpus. The application regenerates all the files of the corpus with the new lists of words.

## 3.5 Statistical analysis of the MMAC corpus

Corpus analysis is concerned with the statistical properties of words and other lexical tokens. In this section an investigation was described into the frequency of these entities in Arabic.

Previous studies of Arabic corpora only analyzed words. The other tokens are emphasised also, namely NWords, PAWs and NPAWS. This is achieved by studying the most and least frequently occurring tokens. The analysis of words was compared with previous researches to verify corpus results.

### 3.5.1 Frequency list

A frequency list is a list of number of occurrences of each element (usually word) in a corpus, and is an important part of any corpus (Buckwalter 2002). It is useful for information retrieval, text categorisation, and numerous other purposes (Kilgarriff 1997). Frequency lists are very useful to the OCR application developer as they can be used to validate the accuracy of the recognition phase. Comparison of the frequency of each word in the corpus and in the system output is an important step when assessing the accuracy of a recognition process. The Frequency lists in MMAC are not limited to words, but also include NWords, PAWs, and NPAWs. The variety in the types of token used here gives flexibility to the developers of OCR applications, who can use the MMAC to support a wide range of application evaluations. The file format is a text file; each line contains a word and the number of occurrence of this word sorted alphabetically. There are 4 separate files in this format: for words, NWords, PAWs and NPAWs.

### 3.5.2 Analysis of words, PAWs and characters

Table 3.5 and Table 3.6 show the detailed analysis of the words and PAWs in MMAC. The most common Words are prepositions while the most common PAWs are those that consist a single letter. There are 25% fewer unique NWords than unique Words. The number of NWords that occur once or are repeated twice is less than in the case of Words, while NWords that are repeated heavily are more common than in the case of unique Words. The number of unique PAWs is very limited relative to the total number of PAWs. More PAWs are heavily repeated than Words. There are 50% fewer unique NPAWs than unique PAWs. The number of NPAWs that have few repetitions is less than in the case of PAWs, although the average number of repeated NPAWs is greater than that of PAWs.

Table 3.5: Statistical summary for Words, NWords, PAWs and NPAWs

| | Total Number of Tokens | Number of Unique Tokens | Average Repetition of Tokens |
|---|---|---|---|
| Words | 6,000,000 | 282,593 | 21.23 |
| Naked Words | 6,000,000 | 211,072 | 28.43 |
| PAWs | 14,025,044 | 66,725 | 210.19 |
| Naked PAWs | 14,025,044 | 32,804 | 427.54 |

Table 3.6: The average characters/Word, characters/PAW and PAWs/Word

| | Average |
|---|---|
| Characters / Word | 4.74 |
| Characters / PAW | 2.03 |
| PAWs / Word | 2.33 |

The total number of one letter PAWs is 6,275,167 which represents 44.7% of the total number of PAWs; while there are 3,652,709 two letter PAWs, 26% of the total. The total number of NPAWs in the corpus is 14,025,044, while the total number of unique NPAWs is 32,804 with an average repetition of 427.54 for each NPAW.

Table 3.7 and Table 3.8 show the analysis of the corpus data at the level of characters. The tables show the number of occurrences of each character in the corpus at each different location in the PAW. Recall that the Arabic letter may have one of four different locations in the PAW, isolated, start, middle and end. Table 3.7 shows all the Arabic character and also the Lam Alef ligature. Table 3.8 shows the letters grouped in their joining groups as NPAWs.

Table 3.7: The number of occurrence of letters in their different locations

| Character | Isolated | Start | Middle | End | Total |
|---|---|---|---|---|---|
| ا | 2,921,359 | 0 | 0 | 1,771,139 | 4,692,498 |
| ب | 75,345 | 485,256 | 350,554 | 81,466 | 992,621 |
| ت | 198,275 | 340,989 | 572,644 | 99,135 | 1,211,043 |
| ث | 12,012 | 46,060 | 80,714 | 21,463 | 160,249 |
| ج | 17,306 | 173,873 | 176,861 | 11,391 | 379,431 |
| ح | 20,697 | 203,432 | 276,101 | 30,110 | 530,340 |
| خ | 2,689 | 118,704 | 101,542 | 12,071 | 235,006 |
| د | 271,891 | 0 | 0 | 555,431 | 827,322 |
| ذ | 65,964 | 0 | 0 | 153,830 | 219,794 |
| ر | 445,282 | 0 | 0 | 952,398 | 1,397,680 |
| ز | 66,653 | 0 | 0 | 94,711 | 161,364 |
| س | 28,650 | 296,167 | 293,789 | 61,225 | 679,831 |
| ش | 7,391 | 101,863 | 146,214 | 9,616 | 265,084 |
| ص | 10,352 | 96,163 | 167,961 | 13,985 | 288,461 |
| ض | 23,512 | 72,473 | 99,236 | 18,100 | 213,321 |
| ط | 9,957 | 77,117 | 142,982 | 33,896 | 263,952 |
| ظ | 1,335 | 11,381 | 44,247 | 4,203 | 61,166 |
| ع | 44,763 | 382,756 | 390,729 | 105,718 | 923,966 |
| غ | 1,824 | 48,332 | 57,907 | 4,579 | 112,642 |
| ف | 34,892 | 391,667 | 216,485 | 49,139 | 692,183 |
| ق | 49,168 | 240,580 | 342,488 | 53,715 | 685,951 |
| ك | 26,453 | 266,309 | 213,981 | 60,663 | 567,406 |
| ل | 159,823 | 1,934,453 | 519,953 | 227,527 | 2,841,756 |
| م | 115,682 | 727,719 | 676,678 | 196,923 | 1,717,002 |
| ن | 300,775 | 357,635 | 427,163 | 369,399 | 1,454,972 |
| ه | 245,657 | 234,768 | 269,303 | 857,477 | 1,607,205 |
| و | 835,979 | 0 | 0 | 724,380 | 1,560,359 |
| ي | 164,639 | 644,409 | 955,427 | 613,885 | 2,378,360 |
| ء | 110,216 | 0 | 0 | 0 | 110,216 |
| ئ | 1,812 | 100,543 | 18,054 | 7,834 | 128,243 |
| ؤ | 4,814 | 0 | 0 | 26,145 | 30,959 |
| ة | 0 | 0 | 0 | 0 | 0 |
| لا | 397,210 | 0 | 0 | 131,095 | 528,305 |
| Total | 6,672,377 | 7,352,649 | 6,541,013 | 7,352,649 | 27,918,688 |

Table 3.8: The number of occurrence of naked letters in their different locations

| Characters | Isolated | Start | Middle | End | Total |
|---|---|---|---|---|---|
| ا | 2,921,359 | 0 | 0 | 1,771,139 | 4,692,498 |
| ب - ت – ث | 285,632 | 1,974,892 | 2,404,556 | 202,064 | 4,867,144 |
| ج - ح – خ | 40,692 | 496,009 | 554,504 | 53,572 | 1,144,777 |
| د – ذ | 337,855 | 0 | 0 | 709,261 | 1,047,116 |
| ر – ز | 511,935 | 0 | 0 | 1,047,109 | 1,559,044 |
| س – ش | 36,041 | 398,030 | 440,003 | 70,841 | 944,915 |
| ص – ض | 33,864 | 168,636 | 267,197 | 32,085 | 501,782 |
| ط – ظ | 11,292 | 88,498 | 187,229 | 38,099 | 325,118 |
| ع – غ | 46,587 | 431,088 | 448,636 | 110,297 | 1,036,608 |
| ف | 34,892 | 632,247 | 558,973 | 49,139 | 1,275,251 |
| ق | 49,168 | 0 | 0 | 53,715 | 102,883 |
| ك | 26,453 | 266,309 | 213,981 | 60,663 | 567,406 |
| ل | 159,823 | 1,934,453 | 519,953 | 227,527 | 2,841,756 |
| م | 115,682 | 727,719 | 676,678 | 196,923 | 1,717,002 |
| ن | 300,775 | 0 | 0 | 369,399 | 670,174 |
| ه | 245,657 | 234,768 | 269,303 | 857,477 | 1,607,205 |
| و – ؤ | 840,793 | 0 | 0 | 750,525 | 1,591,318 |
| ى - ي – ئ | 166,451 | 0 | 0 | 621,719 | 788,170 |
| ء | 110,216 | 0 | 0 | 0 | 110,216 |
| لا | 397,210 | 0 | 0 | 131,095 | 528,305 |
| Total | 6,672,377 | 7,352,649 | 6,541,013 | 7,352,649 | 27,918,688 |

### 3.5.3 Discussion

It is important to measure the frequency of occurrence of the Words, NWords, PAWs and NPAWs of the Arabic language. Frequency data aids the development of OCR applications by encouraging the OCR developer to recognize more common elements first. For example, the PAW (ا) represents more than 20% of the written Arabic language. As Figure 3.8 shows, this percentage increases dramatically when measured in NPAWs instead of words. The 25 most repeated NPAWs account for more than half the words used in practice.

Statistical analysis shows that the number of unique NPAWs in the MMAC is very limited. The number of unique NWords is almost six times that of NPAWs. Moreover, this increases rapidly, with no evident asymptote, as new texts are added to the corpus as shown in Figure 3.9. The number of unique NWords is about 80% of the number of unique Words and this ratio remains fairly stable once a reasonable-sized corpus has been established.

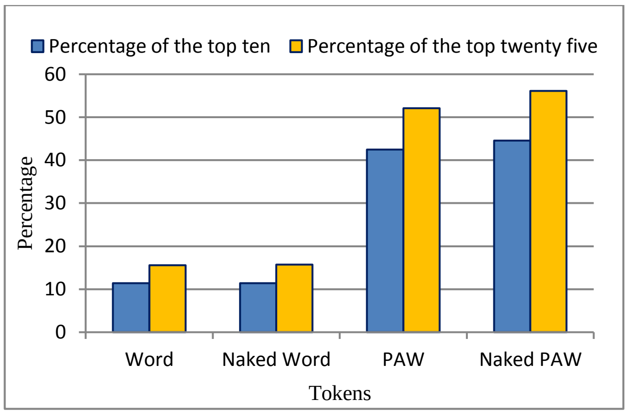


Figure 3.8: The relationship between top 10 and top 25 most repeated tokens

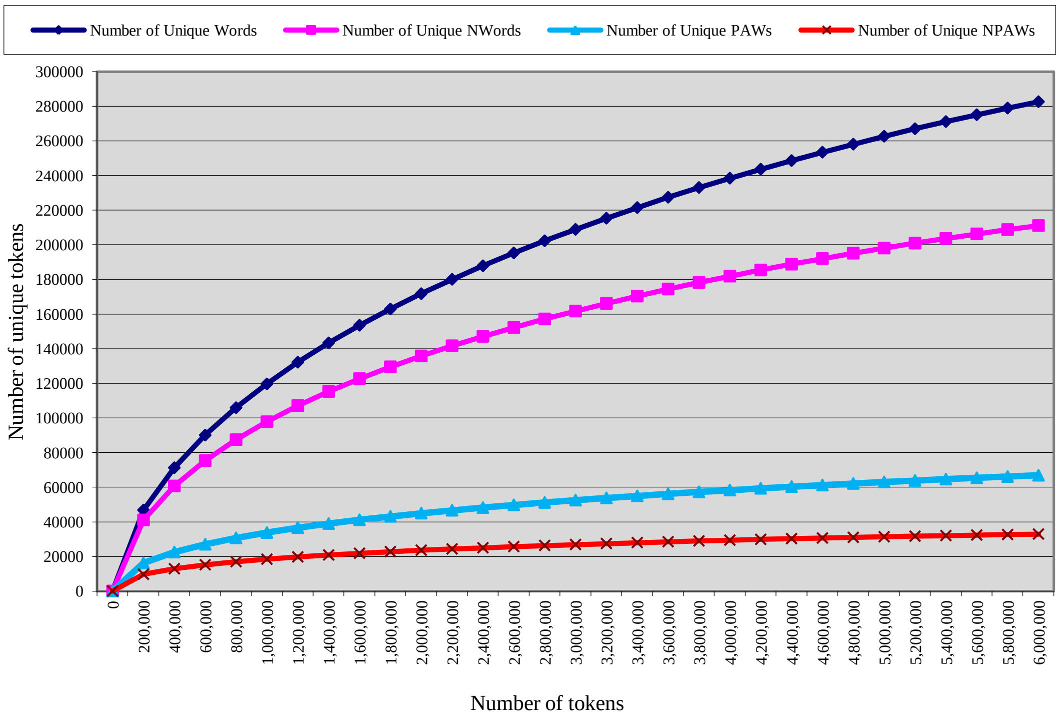


Figure 3.9: Relationship between distinct tokens and words as samples increases

Analysis of MMAC also shows the least repeated Words and PAWs in the language. Table 3.9 shows the relationship between the total number of Words, NWords, PAWs and NPAWs that occur only once; as percentages of the total number of distinct words and the total number of words, NWords, PAWs and NPAWs in the corpus. The twenty five most repeated Words, NWords, PAWs and NPAWs and the percentage of each of them in the corpus are shown in Table 3.10. The analysis of words in Table 3.10 gives almost the same results as that found by others (Al-

Ma'adeed, Elliman et al. 2002; Buckwalter 2002; Pechwitz, Maddouri et al. 2002), which gives credibility to the analysis of this research. The analysis of NWords, PAWs and NPAWs in Table 3.10 is very important to the OCR developer in the sense of giving more emphasis not only to the most repeated words but also the most repeated NWords, PAWs and NPAWs. Tokens that are repeated most should be recognized before those that appear only rarely. According to the previous statistics, advice to OCR developers is to consider recognising the one letter PAWS in their isolated shape first.

Table 3.9: Tokens that occurred once in relation to total unique and total tokens

| | Total Number of Tokens that Occur Once | Percentage to the Total Number of Unique Tokens | Percentage to the Total Number of Tokens |
|---|---|---|---|
| Word | 107,771 | 38.14% | 1.796% |
| Naked Word | 69,506 | 32.93% | 1.158% |
| PAW | 20,239 | 30.27% | 0.144% |
| Naked PAW | 8,201 | 24.92% | 0.058% |

Table 3.10: The 25 most repeated tokens. Number and percentage of repetitions

| Serial | No. of Repetitions | Word | Percentage | No. of Repetitions | Naked Word | Percentage | No. of Repetitions | PAW | Percentage | No. of Repetitions | Naked PAW | Percentage |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 164,165 | في | 2.73% | 164,193 | فى | 2.73% | 2,921,359 | ا | 20.81% | 2,921,359 | ا | 20.84% |
| 2 | 139,380 | من | 2.32% | 139,380 | من | 2.32% | 835,979 | و | 5.96% | 840,793 | و | 5.99% |
| 3 | 82,429 | ان | 1.37% | 82,429 | ان | 1.37% | 445,282 | ر | 3.17% | 511,935 | ر | 3.65% |
| 4 | 78,417 | علي | 1.30% | 78,427 | على | 1.30% | 397,210 | لا | 2.83% | 397,210 | لا | 2.83% |
| 5 | 40,431 | الي | 0.67% | 40,431 | الى | 0.67% | 300,775 | ن | 2.14% | 337,855 | د | 2.41% |
| 6 | 40,091 | هذا | 0.66% | 40,114 | هدا | 0.66% | 271,891 | د | 1.94% | 300,775 | ن | 2.14% |
| 7 | 37,830 | لا | 0.63% | 37,830 | لا | 0.63% | 245,657 | ه | 1.75% | 285,629 | ب | 2.03% |
| 8 | 34,897 | او | 0.58% | 34,897 | او | 0.58% | 198,275 | ت | 1.41% | 245,657 | ه | 1.75% |
| 9 | 34,197 | ما | 0.57% | 34,197 | ما | 0.57% | 176,106 | في | 1.25% | 226,566 | با | 1.61% |
| 10 | 29,896 | عن | 0.49% | 30,315 | البى | 0.50% | 164,317 | ي | 1.17% | 180,173 | فى | 1.28% |
| 11 | 29,793 | التي | 0.49% | 29,900 | عن | 0.49% | 159,823 | ل | 1.14% | 166,129 | ى | 1.18% |
| 12 | 25,759 | اي | 0.42% | 25,759 | اى | 0.42% | 151,009 | من | 1.07% | 159,823 | ل | 1.14% |
| 13 | 21,363 | الذي | 0.35% | 21,413 | الدى | 0.35% | 115,682 | م | 0.82% | 151,009 | من | 1.07% |
| 14 | 20,010 | كل | 0.33% | 20,010 | كل | 0.33% | 115,372 | با | 0.82% | 149,501 | به | 1.06% |
| 15 | 19,576 | هذه | 0.32% | 19,584 | هده | 0.32% | 102,560 | ء | 0.73% | 115,682 | م | 0.82% |
| 16 | 18,566 | هو | 0.30% | 18,566 | هو | 0.30% | 93,731 | ما | 0.66% | 115,436 | بر | 0.82% |
| 17 | 16,738 | كان | 0.27% | 16,738 | كان | 0.27% | 81,895 | علي | 0.58% | 102,560 | ء | 0.73% |
| 18 | 16,467 | مع | 0.27% | 16,468 | مع | 0.27% | 76,162 | يه | 0.54% | 101,839 | بد | 0.72% |
| 19 | 16,317 | لم | 0.27% | 16,317 | لم | 0.27% | 75,344 | ب | 0.53% | 96,635 | لد | 0.68% |
| 20 | 14,065 | اذا | 0.23% | 15,910 | بعد | 0.26% | 69,987 | كا | 0.49% | 93,859 | بو | 0.67% |
| 21 | 13,975 | بعد | 0.23% | 14,602 | ابه | 0.24% | 69,779 | لي | 0.49% | 93,731 | ما | 0.66% |
| 22 | 12,944 | ولا | 0.21% | 14,081 | ادا | 0.23% | 66,653 | ز | 0.47% | 82,003 | على | 0.58% |
| 23 | 12,863 | ذلك | 0.21% | 12,944 | ولا | 0.21% | 65,964 | ذ | 0.47% | 74,079 | فا | 0.52% |
| 24 | 11,750 | بين | 0.19% | 12,873 | دلك | 0.21% | 63,088 | هذ | 0.45% | 69,987 | كا | 0.49% |
| 25 | 11,688 | انه | 0.19% | 12,023 | ببن | 0.20% | 59,907 | لو | 0.42% | 69,791 | لى | 0.49% |

The other important advice to OCR developers is to deal with the two letter PAWs as if they are isolated letters or new glyphs. These new glyphs present just 612 different shapes. If the OCR developers recognize these isolated letters and two letters glyphs they can process 70% of the language.

## 3.6 Testing the corpus' validity

Testing the validity and accuracy of the data is a very important issue in creating any corpus (Mashali, Mahmoud et al. 2005). The testing process started by collecting a testing dataset from unusual sources consisting of scanned images from faxes, documents, books, medicine scripts and well known Arabic news websites. This was done one year after the original MMAC data collection was completed. The testing dataset was collected from sources that intersected minimally with those of the corpus. The total number of Words in the testing dataset is 69,158 and the total number of PAWs is 165,501. Table 3.11 shows the total and the number of distinct Words, NWords, PAWs and NPAWs of the testing dataset.

The lexicon / stemming approach was also tested in two different ways. The first was to test the validity of the algorithm when adding or removing affixes. The second was to test adding the testing dataset to the corpus and to check the new generated corpus files.

Table 3.11: The corpus testing dataset tokens with their unique number

| | Corpus Testing Dataset Total Number of Tokens | Corpus Testing Dataset Unique Number of Tokens |
|---|---|---|
| Word | 69,158 | 17,766 |
| Naked Word | 69,158 | 16,513 |
| PAW | 165,518 | 7,272 |
| Naked PAW | 165,518 | 4,749 |

### 3.6.1 Testing words and PAWs

A program was developed to search in the corpus binary data files using a binary search tree algorithm. The percentage of unique words in the testing dataset found in the corpus data file is 89.8%. Table 3.12 gives the percentage of tokens found from the testing dataset in the corpus data files. This analysis deals with the unique data only. On the other hand, the total testing dataset words missing (that are not unique) in the corpus data files are 2,457 Words with a percentage of 3.5%. Table 3.13 shows this ratio in the corpus tokens.

Table 3.12: Number & percent of unique found tokens of testing data in MMAC

| | Total Number of Unique Tokens Found | Percentage of Total Unique Tokens Found |
|---|---|---|
| Word | 15,967 | 89.8% |
| Naked Word | 15,163 | 91.8% |
| PAW | 6,983 | 96% |
| Naked PAW | 4,629 | 97.5% |

Table 3.13: Number & percentage of missed tokens of testing data from MMAC

| | Total Number of Missed Tokens | Percentage of Total Missed Tokens |
|---|---|---|
| Word | 2,457 | 3.5% |
| Naked Word | 1,843 | 2.66% |
| PAW | 1,600 | 0.96% |
| Naked PAW | 1,358 | 0.82% |

### 3.6.2 Checking corpus accuracy

The testing dataset was also used to check the accuracy of the statistics obtained from the corpus. By extrapolation, it was believed that if the curve between the number of words and the unique number of words (Figure 3.9) is extended according to the number of words that exists in the testing dataset, this will result in an increase in the unique number of words is almost equal to the missing words in the corpus.

The shape of the curve between the total number of words and the unique number of words is a nonlinear regression model. The CurveExpert 1.3 shareware program was used (Hyams 2005) and applied its curve finder process. It was found that the best regression model is the Morgan-Mercer-Flodin (MMF) model, from the Sigmoidal Family (Gu, Hu et al. 2001).

The best fit curve equation is:

$$y = \frac{ab + cx^d}{b + x^d}$$

where:

| | |
|---|---|
| *a = -42.558444* | *b = 40335.007* |
| *c = 753420.21* | *d = 0.64692321* |
| *Standard Error: 156.0703663* | *Correlation Coefficient: 0.9999980* |

Upon applying the previous equation, it was found that the increase in the number of words to 6,069,158 will increase the number of distinct words by 1,676, while the number of missing words is 1,799. This result supports the previous assumption, with an accuracy of 93%.

### 3.6.3 Testing lexicon / stemming approach validity

The lexicon / stemming approach was tested against the MMAC corpus data using the testing dataset mentioned before. The purpose of this testing is to check the validity of the approach. It checks whether the generated list of alternative words includes reasonable words or not.

The total number of the unique words in the corpus is 282,593. The total number of the unique words in the testing dataset is 17,766. The total number of words from the testing dataset found in the corpus is 15,967 with an accuracy of 89.8%. The total number of words from testing dataset missed from the corpus is 1,799 with an error rate of 10.2%.

The total number of words found in the corpus after applying the approach is 17,431 with accuracy of 98.11%. The total number of words missing from the corpus after applying the approach is 335 words with an error of 1.89%.

The total number of words found in the corpus using the approach only is 1,464 words. These words are either relevant to the missing word which wasn't found in the corpus (1,387 words with an accuracy of 94.7%), or irrelevant to the missing word (77 words with an error of 5.3%).

The previous statistics indicate that lexicon / stemming algorithm reaches a high level of accuracy in finding the words (98.1%) with a very minor missing words error factor of 1.89% and also a very minor error in finding irrelevant words of 0.4%.

### 3.6.4 Applying the lexicon / stemming approach

This part tests using the lexicon / stemming approach to add new words to the MMAC corpus and show the effect of these additions upon the performance of the corpus. It also tends to set a rule for the corpus to include more new words regularly.

The testing dataset mentioned before was used as a list of new words to be added to the corpus. Table 3.14 shows the number of words that are added to the corpus using the testing dataset. The numbers in Table 3.14 are for both trusted and un-trusted lists.

Table 3.14: Number and percentage of tokens using trusted and un-trusted lists

| Description | Before lexicon / stemming | After lexicon / stemming | Percentage increase |
|---|---|---|---|
| Total number of Words | 6,000,000 | 6,069,158 | 1.15% |
| Number of Unique words | 282,593 | 284,392 | 0.64% |
| Number of Unique Naked words | 211,072 | 212,422 | 0.64% |
| Number of Unique PAWs | 66,725 | 67,010 | 0.43% |
| Number of Unique Naked PAWs | 32,804 | 32,925 | 0.37% |

Table 3.15 shows the number of words that are added to the corpus using the trusted list of the testing dataset.

Table 3.15: Total number and percentage of tokens using trusted list

| Description | Before lexicon / stemming | After lexicon / stemming | Percentage increase |
|---|---|---|---|
| Total number of Words | 6,000,000 | 6,069,158 | 1.15% |
| Number of Unique words | 282,593 | 284,057 | 0.52% |
| Number of Unique Naked words | 211,072 | 212,142 | 0.51% |
| Number of Unique PAWs | 66,725 | 66,923 | 0.30% |
| Number of Unique Naked PAWs | 32,804 | 32,894 | 0.27% |

Table 3.16 shows the number of words that are added to the corpus using the un-trusted list of the testing dataset.

Table 3.16: Total number and percentage of tokens using un-trusted list

| Description | Before lexicon / stemming | After lexicon / stemming | Percentage increase |
|---|---|---|---|
| Total number of Words | 6,000,000 | 6,069,158 | 1.15% |
| Number of Unique words | 282,593 | 282,928 | 0.12% |
| Number of Unique Naked words | 211,072 | 211,352 | 0.13% |
| Number of Unique PAWs | 66,725 | 66,813 | 0.13% |
| Number of Unique Naked PAWs | 32,804 | 32,839 | 0.11% |

The previous tables show that the approach is working properly with the MMAC corpus. They also showed that the approach can easily increase the total number of valid words in MMAC corpus. It is also apparent also that the process of adding new lists of words to the corpus is very easy using this approach. The complicated procedures that were followed to generate the corpus are no longer needed in the process of adding new lists of words.

## 3.7 The online MMAC corpus

The contents of the MMAC are available from http://www.ashrafraouf.com/mmac. It also includes the method applied to transliterate the tokens text to English tokens name.

### 3.7.1 MMAC corpus contents

The contents of the online version of MMAC are:

1. *Original Data File:* This contains the original sequence of words as collected. It can be used for defining word frequency pairs, for example.

2. *Original Data File After Replacing Alef:* This file is the same as above after substituting Alefs.

3. *Sample Raw Data Collected:* This contains samples of all types of files that are used in the corpus. The files are in their original formats.

4. *Data Files:* This folder contains four sub-folders for the main statistical analysis of the research. These folders are for Words, NWords, PAWs and NPAWs. Each of the previous folders (for example the Word folder) includes a randomized

word list file, a Unique sorted Words list, a Word Repetition Counting list, the Top 50 repeated Words list and a Word Occurred Once list file.

5. *MMAC tokens Images:* This folder contains four sub-folders for the images of the four different types of tokens. Each folder contains samples of the images of 50 tokens and their ground truth XML files.

6. *Paragraph documents Dataset:* This folder contains three sub-folders for the real images, computer images and noisy images. Each folder contains the images and the truth text.

7. *Real Scanned Dataset:* This folder contains two sub-folders for the typed and scanned documents. Each folder contains 19 documents.

8. *Corpus testing data:* This folder contains four sub-folders for the computer tokens tests, noise tokens test, paragraph tests and full page tests.

9. *Frequency List:* This folder contains the frequency lists of the Words, NWords, PAWs, NPAWs. The lists are sorted alphabetically. Each line contains a token with its frequencies in the corpus.

10. *Applications:* This folder contains all the applications used to support the generation of the corpus and degrade images.

11. *All Statistics.xls:* This is an Excel spreadsheet file containing all the data sheets and charts that have been used in the analysis of the corpus data.

### 3.7.2 Transliteration encoding

For flexibility and ease of use it is necessary to use Roman character based letters for the Arabic image text. The Arabic English transliteration encoding is used to name the Arabic text in English. An explanation of the different types of Arabic English transliteration encoding available is now given. Transliteration is a mapping between two different languages, in this case Arabic and English (contributors 2006). There are many types of transliteration between Arabic and English which can be summarized into two main categories: Arabic Transliteration (contributors 2006) and Arabic Chat Alphabets (Palfreyman and Khalil 2003).

Using Arabic transliteration encoding is much better in this case than Arabic Chat Alphabets - although “chat” is more readable to Arabic natives for the following reasons:

1. The Arabic Chat Alphabets neglect letters with diacritics like (ئ ؤ إ أ آ) (contributors 2006).

2. The Arabic Chat Alphabets sometimes use two characters to transliterate a single character.

3. The Arabic Chat Alphabets sometimes use special characters to transliterate (Palfreyman and Khalil 2003).

Table 3.17: Buckwalter Transliteration encoding

| Transliterated Character | Arabic Character | Transliterated Character | Arabic Character | Transliterated Character | Arabic Character | Transliterated Character | Arabic Character |
|---|---|---|---|---|---|---|---|
| ' | ء | T | ت | s | س | f | ف |
| \| | آ | V | ث | $ | ش | q | ق |
| > | أ | J | ج | S | ص | k | ك |
| & | ؤ | H | ح | D | ض | l | ل |
| < | إ | X | خ | T | ط | m | م |
| } | ئ | D | د | Z | ظ | n | ن |
| A | ا | * | ذ | E | ع | h | ه |
| B | ب | R | ر | g | غ | w | و |
| P | ة | Z | ز | - | - | Y | ى |
| | | | | | | y | ي |

The most famous Arabic Transliteration encoding is the Buckwalter Transliteration as shown in Table 3.17 (Buckwalter 2002). Buckwalter Transliteration encoding was used (37 characters) with some modifications for the following reasons:

1. Buckwalter uses a single character for transliteration.

2. Buckwalter includes all written Arabic characters and also some diacritics such as (◌ٍ ◌ٌ ئ ؤ إ أ آ) (Ananthakrishnan, Bangalore et al. 2005).

3. Buckwalter uses fewer capital letters. (it only uses A, H, S, D, T, Z, E, Y).

### 3.7.3 Buckwalter modified transliteration encoding

The purpose of transliteration in this case is different from most other transliteration uses. Transliteration encoding is used for many purposes, including file names; therefore, all the restrictions of the use and also file name restrictions in the most common operating systems (Windows, MAC OS and UNIX) must be followed. These restrictions are:

1. *The file name may not be case sensitive:* Hence changing the eight letters used in Buckwalter (A, H, S, D, T, Z, E, Y) by putting the character '#' before them.

2. *White space is not allowed:* This restriction is not applicable in this case.

3. *Some special characters like (|, >, *) are not allowed:* So changing the transliteration for:

- (آ) from '|' to be 'i'.
- (أ) from '>' to be 'a'.
- (ؤ) from '&' to be '@'.
- (إ) from '<' to be 'e'.
- (ذ) from '*' to be '~'.

The Modified Buckwalter Transliteration is almost the same as the original but with 13 of the original 37 characters modified. Table 3.18 shows the characters and their proposed transliterated characters.

Table 3.18: Buckwalter Modified Transliteration encoding

| Transliterated Character | Arabic Character | Transliterated Character | Arabic Character | Transliterated Character | Arabic Character | Transliterated Character | Arabic Character |
|---|---|---|---|---|---|---|---|
| ' | ء | t | ت | s | س | f | ف |
| I | آ | v | ث | $ | ش | q | ق |
| A | أ | j | ج | #S | ص | K | ك |
| @ | ؤ | #H | ح | #D | ض | l | ل |
| E | إ | x | خ | #T | ط | m | م |
| } | ئ | d | د | #Z | ظ | n | ن |
| #A | ا | ~ | ذ | #E | ع | h | ه |
| B | ب | r | ر | g | غ | w | و |
| P | ة | z | ز | - | - | #Y | ى |
|  |  |  |  |  |  | y | ي |

## 3.8 Summary

This chapter explained the generation of a multi-modal Arabic corpus. It shows the procedures followed to generate and test it. This chapter ended up with a real Arabic corpus. This corpus is used in the next three chapters for the applying of the HCC approach to generate Arabic tokens classifier, generate HCC tokens recognizer and in the post-processing HCC tokens. Here are the issues that were discussed in this chapter:

1. An introduction to the corpus and the need to build an Arabic corpus.
2. The contents, collection and preparation of the textual data of the corpus.
3. The contents and organization of the images data of the corpus and how they were collected and truthed.
4. Developing the OCR application is one of the main benefits of the corpus; this is explained extensively in the chapter.
5. Developing a lexicon /stemming approach to enhance the corpus. The approach guarantee the development of the corpus and the continuity of updating it
6. The statistical analysis of the corpus. Different types of the analysis of the corpus.
7. The corpus information is validated via a testing dataset. The verifications show how the corpus data were trusted through many types of verifications.
8. Presentation of the corpus on the internet.

# Chapter 4.

# Glyph recognition using a Haar Cascade Classifier

This chapter describes the experimental work of applying the Haar Cascade Classifier approach (HCC). The HCC approach uses the cascade of boosted classifiers based on Haar-like features in order to recognize Arabic glyphs. An explanation is given of how the approach was applied to Arabic character recognition. Moreover the steps and experiments that were followed in order to justify that the HCC approach is applicable for the Arabic character recognition application are given. At the end an experiment is introduced to include all the Arabic naked glyphs using the HCC approach and the results shown. It concludes with the usefulness of the approach in Arabic character recognition.

The previous chapter discussed the generating and applying of a Multi-Modal Arabic Corpus (MMAC). This chapter makes use of the MMAC which was mainly used in the training and testing of classifiers. Rather than training and testing, most of the content of the MMAC was used in the experiments applied in this chapter in order to justify the usefulness of the HCC approach in Arabic character recognition.

## 4.1 Using a Haar Cascade Classifier with Arabic character recognition

The Haar Cascade Classifier approach is very useful for problems within printed Arabic character recognition. It can solve many of the traditional problems of Arabic character recognition. Also, it has been found that there are some similarities between the detection of faces and the recognition of Arabic glyphs.

### 4.1.1 Arabic character recognition difficulties

Research studies in Optical Character Recognition (OCR) have common difficulties that need solving no matter the approach. Arabic character recognition has some extra difficulties that are applicable to the Arabic language and languages that use the Arabic alphabet. The properties of the Arabic language are shown in section 2.2. The two types of difficulties are summarized as:

1. Common OCR difficulties include the need to use a binarization algorithm in order to convert the grey-scale image to a binary image (black and white). A skew detection and correction algorithm is needed in order to correct errors that occur during the scanning of the document which causes the document image to become rotated. A normalization algorithm is also required to scale the document in order to make the glyphs the same size as the trained glyphs.

2. The Arabic language is cursive and the letters are frequently connected inside the words. This feature of the Arabic language needs a sophisticated character segmentation algorithm in order to segment the word to its original glyphs. The character segmentation algorithm is one of the bottlenecks of any Arabic character recognition research or application (Abdelazim 2006). This research considers that skipping the character segmentation process will improve Arabic character recognition success rates.

The previous difficulties that have been encountered serve to make the OCR application more sophisticated and take more time in applying the algorithms needed. However, they also reduce the recognition accuracy of Arabic OCR.

### 4.1.2 Benefits of using the Haar Cascade Classifier approach

The advantages of using the HCC approach are particularly beneficial for Arabic character recognition. The Haar-like features technique is ideal for Arabic characters as it copes with the visual properties of an object; and Arabic glyphs have clear visual properties. Combining the feature extraction with the classification stages in HCC facilitates the process of training and testing in the proposed application.

Arabic character recognition can benefit from skipping the pre-processing stages. It benefits from the scaling invariant of the approach in that it removes the need for a normalization algorithm. It benefits from extended, rotated features in order to ignore the skew detection and correction. The ability to use grey-scale images removes the need for a binarization algorithm in order to convert to a binary image.

The nature of the HCC approach (when used with face detection) depends on capturing the image of the face from the testing image. This nature can be applied for Arabic character recognition in order to capture the selected glyph from the document image. Applying the previous nature leads to an Arabic character recognition approach without the character segmentation process which is considered to be the main contribution of this research.

### 4.1.3 Dealing with variations

Although the HCC approach was originally presented for face detection, in this research it has been found to be very useful for the recognition of Arabic characters. There are some similarities with Arabic glyphs:

1. Similar to faces, most of the Arabic glyphs have clear visual features.

2. Arabic characters are connected and it is better to pick out the glyphs from inside the document image without character segmentation; like picking out the face from inside the image.

3. Characters may have many font sizes inside the document image and may also be rotated; this is the similar to faces. Different face sizes may be found inside an image and also at different rotational angles.

4. The facial image may contain different face styles, this may occur in the Arabic glyphs with different font style.

### 4.1.4 How to apply the approach

Although the HCC approach was originally invented for face detection and not recognition, it can be used for Arabic character recognition. The different Arabic glyphs represent different objects. Each glyph can be considered as a different object to be detected, and in this case each glyph has a different classifier. The glyph classifier will detect its glyphs and ignore all other glyphs; in doing so it becomes a glyph recognizer rather than a detector.

This approach needs the preparation of two main sets of images. A positive set contains images which include at least one target object inside every image of the set.

A negative set contains images which do not include any target object inside the images of the set. The approach needs to prepare training data which contains both negative and positive sets. Testing data is then required which contains only a positive set.

The positive set includes a list file containing the image name and the positions of the object inside the image. The position of the object is defined by the top-left X-coordinate, top-left Y-coordinate, width and height. All the units are measured in pixels.

Applying the HCC approach in order to recognize the Arabic characters requires the generation of training and testing sets for each glyph. The glyphs in the Arabic language (as shown in section 2.2) vary for the same letter; the same letter normally has four different locations in the word (isolated and at the start, middle and end). Each letter location has a different shape which leads to a different glyph. This means that almost every letter in the Arabic language has four different glyphs. Thus a total of 100 datasets and classifiers are needed as shown in Table 2.3.

## 4.2 Problem solving approach

The problem solving approach consists of applying experiments in order to check the validity of the approach in Arabic printed character recognition. The aim of the experiment was to run a simple pilot experiment simply to discover whether the approach was applicable or not. When the first pilot experiment was passed successfully, it would move to a more advanced experiment. The extended experiment was the second step in applying the HCC approach. Passing the extended experiment with good recognition accuracy led us to the final experiment which dealt with all the Arabic glyphs. Auxiliary experiments were run in order to get the best training parameters within Arabic printed character recognition.

### 4.2.1 The pilot experiment to detect the isolated letter Ain (ع)

The pilot experiment was applied to test whether the HCC approach is applicable to Arabic character recognition or not. The plan was to make it as simple as possible to investigate the HCC approach with new objects like the Arabic glyphs. For the sake of simplicity a single letter, Ain (ع), was used in its isolated location as an object. The negative and positive images used were that offered from OpenCV. Also, the training and testing parameters used were the default proposed by OpenCV. Finally the output of the Ain classifier was tested with a real Arabic document image in order to make sure that the HCC approach can recognize Arabic characters. The details of the experiment are shown in section 4.3.

### 4.2.2 The extended experiment to detect 15 different glyphs

An extended experiment was planned to extend the first pilot experiment after obtaining good results. The extended experiment was more advanced than the pilot experiment. The number of objects in this case was 15 glyphs which represent most of the Arabic glyph cases. Using actual Arabic document images to generate the positive and negative image datasets, the experiment used different types of images; real scanned, computer generated and computer generated with artificial noise. At the end of the experiment, the results were tested with the testing tool offered by OpenCV and a commercial Arabic OCR application. The details of the experiment are shown in section 4.4.

### 4.2.3 The auxiliary experiments to improve the training parameters

Auxiliary experiments were planned in order to obtain the best parameters that can be used during the training process. These parameters were width, height, number of splits, minimum hit rate and boosting algorithms. The auxiliary experiments were:

1. To test the influence of the training width and height of the glyph on the final recognition accuracy;

2. To test the effect of the number of splits and minimum hit rate used during the training of the glyph on the final recognition accuracy and training duration;

3. To test the influence of using different types of boosting algorithms with a different number of splits on the final recognition accuracy. These boosting types are Gentle Ada-Boost (GAB), Discrete Ada-Boost (DAB) and Real Ada-Boost (RAB).

The details of the experiments are shown in section 4.5.

### 4.2.4 The final experiment to detect naked Arabic glyphs

The final experiment was planned after completing the extended experiment with satisfactory results. The final experiment generated classifiers for all the Arabic characters in their different locations (isolated and at the start, middle and end). The experiment deals with the naked shape of the glyphs to facilitate faster detection and makes use of the results of the auxiliary experiments. The same types of positive and negative datasets used in the extended experiment were used. At the end of the experiment the results were tested using only the testing tool offered by OpenCV in this case, as the testing of the entire system will be discussed later (see Chapter 5 and Chapter 6). The details of the experiment are shown in section 4.6.

## 4.3 The pilot experiment

This experiment was the first step in handling the HCC approach. As it was the first time that the HCC approach has been used with character recognition, particularly Arabic character recognition, it was important to strictly follow the procedure offered by OpenCV in order to generate a classifier. There were three main references to follow in building the classifier (OpenCV 2002; Adolf 2003; Seo 2008). These references explain in detail how to build a face classifier using the HCC approach. The same procedures were followed but replaced the object from a face to the Arabic letter Ain (ع) in its isolated form.

### 4.3.1 Planning the experiment

The experiment was originally conceived in order to discover the applicability of the HCC approach to Arabic character recognition. The experiment was planned as follows:

1. Create a Haar-Cascade Classifier (HCC) for the isolated Arabic letter Ain (ع). The letter Ain is in the binary (black and white) format.
2. Use the negative images that are available from the OpenCV library.

3. Generate positive images using the tool offered by OpenCV and using default parameters in order to generate these positive images.

4. Train the Ain classifier within the default parameters suggested by the HCC approach.

5. Test the Ain classifier with two different methods; one with the default testing tool offered by OpenCV and the other with real Arabic document images.

### 4.3.2 Data Selection and preparation

The instructions provided by OpenCV in order to create the datasets were strictly followed. The datasets required for applying the experiment included: different images of the letter Ain (ع), a negative dataset and a positive dataset.

#### *4.3.2.1 Reasons of selecting letter Ain (ع)*

The letter Ain (ع) is selected in its isolated shape for the following reasons:

1. It is not a simple letter shape and has two curvatures. Also, the top curve ends are very close to each other.

2. Using a letter in its isolated shape is preferable to it being connected with another letters.

3. The letter Ain is Arabic letter that looks a little like a face from its visual features.

4. There is little difference in the shape of the letter among the different font styles.

#### *4.3.2.2 Different images of the letter Ain (ع)*

The training set was generated from twelve different examples and was divided into two equal parts, each with six images. The first part was the computer generated letters with different fonts, sizes and styles (bold and italic). The other six images were extracted from scanned images of real documents. These 12 images are shown in the first three rows of Figure 4.1. The testing set was generated from four images which are shown in the last row of Figure 4.1. The two sets were extracted from different sources. The images used of the letter Ain (ع) were in a binary colour tiff file format with resolution of 200DPI. The binary colour format is used in order to simplify the experiment. They were extracted from grey-scale images with a binarization threshold of 150 out of 255.

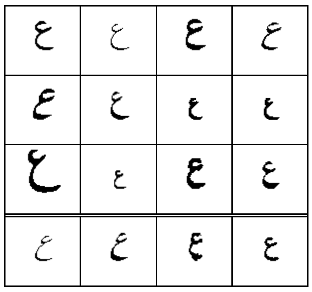


Figure 4.1: The different images of the letter (ع) used in the pilot experiment

### 4.3.2.3 *The negative images (background images)*

These are a set of images that do not include the object (the Ain image) at all. This set is comprised of grey-scale images with sizes starting from 240x240 pixels up to 640x480 pixels. The selected negative images were 3,000 images and were imported from digital libraries of image photos for shareware (http://tutorial-haartraining.googlecode.com/svn/trunk/data/negatives/). The same procedures followed in (Seo 2008) were applied. It wasn't necessary to check whether the letter Ain was included in any of them as they were images taken from countries that do not use the Arabic language. Figure 4.2 shows different samples of the negative images.

Figure 4.2: Samples of the pilot experiment negative images

#### 4.3.2.4 The positive images (foreground images)

These are a set of images that include at least one object (the Ain image) in each. The images were imported from a shareware image library, as were the images in the negative dataset. They are images similar to the negative images but use the tool offered by OpenCV in order to put the object (the letter Ain) in them with variant scaling and rotation parameters.

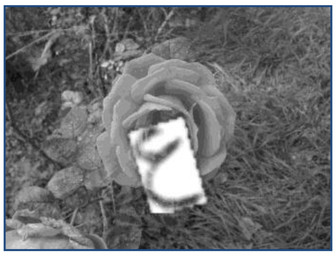
Although this method of creating the positive images is not the best way, but was followed in this experiment because the others who applied the HCC approach advised and followed (OpenCV 2002; Adolf 2003; Seo 2008). Figure 4.3 (a) shows a sample of positive images used in the pilot experiment. Figure 4.3 (b) shows a sample of positive images used for face detection by (Seo 2008).

This set was generated by putting an Ain letter image from the twelve images onto a background image different from the negative images. Each Ain image was put onto 500 different background images. This gave a total of 6,000 positive images. An OpenCV function called *createsamples* was used to generate the positive 6,000 training images and was used with the following parameters:

```
createsamples -img E1.tif -num 500 -bg negatives.dat -vec samples.vec -
maxxangle 0.6 -maxyangle 0 -maxzangle 0.3 -maxidev 100 -bgcolor 0 -
bgthresh 0 -w 20 -h 20
```

The important parameters in the previous function defined the maximum skew angles in the x, y and z axes, in addition to defining the training size of the classifier. Using the previous function generates 500 different positive images with different skew angles. The other important parameters in the previous function are the width and height of the detected object. The recommended size from OpenCV was used.

The positive images are kept in files with a format *.vec*. This file keeps all the information of the 500 images inside it. A function called *mergevec* is used in order to merge all the *.vec* files into one file with 6,000 internal images.


(a)

(b)

Figure 4.3: Sample of positive in image pilot experiment and OpenCV

## 4.3.3 Training the Ain classifier

The training process uses the previous prepared datasets in order to generate the Ain classifier. The output of the training process is an xml file that contains the Haar-Cascade Classifier. The training process takes a long time to complete; around three days continuously (72 hours) on a machine with Intel processor Core 2 Due and 4 GB of RAM running Windows Vista 32 bits. The good thing about this long time training process is that it can be resumed if stopped for any reason. An OpenCV function called *haartraining* was used for the training process with the following parameters:

```
haartraining -data haarcascade -vec positive/E.vec -bg negative/negatives.dat
-nstages 20 -npos 6000 -nneg 3000 -w 20 -h 20 -mem 2000 -mode ALL
```

Using the previous function it was assumed that the default values of hitrate (0.995), maxfalsealarm (0.5), weighttrimming (0.95) and boosting type (GAB, "Gentle Ada-Boost") are good in most cases (Adolf 2003). The least number of stages (-*nstages 20*) was used in this experiment as it is believed that the process for Arabic character recognition is simpler than that of faces. The file *E.vec* is the file that contains all the positive images and was generated using the previous *createsamples* function. The file *negatives.dat* is a text file containing the list of negative images. The parameters (-*npos 6000*, -*nneg 3000*) define the number of positive and negative images. The parameter (-*mem 2000*) defines the size of memory for the function use and it is recorded in Mega byte, so 2,000 means around 2G-Bytes. The Haar-like features used is defined in the parameter (-*mode ALL*); whether just the basic features or the basic and extended features which are defined as ALL.

This stage generates an XML file (*-data haarcascade*) containing the training data of the Ain isolated glyph (Ain classifier). This Ain classifier will be used in the next stage of testing.

### 4.3.4 Testing the Haar-cascade classifier

The testing process is separated into two different parts: the first part is to follow the testing procedures as per the OpenCV library (OpenCV 2002; Adolf 2003; Seo 2008) and the second part is to test real scanned Arabic document images.

#### *4.3.4.1 Testing using the OpenCV performance utility*

The data files prepared in section 4.3.2 were used at this stage, as were a set of positive images. They were generated from a different source of data to that of the training process. A function called *performance* was applied to run the testing process with the following parameters:

```
performance –data haarcascade.xml –info Testing/Positive/E1.dat –w 20 –h 20
–ni
```

The function uses the Ain classifier file (–data haarcascade.xml) generated from the training process. The classifier file tests the list of positive images brought from the list of positive images from the file (–info Testing/Positive/E1.dat). The last parameter (–ni) is used in order to suppress the generation of the resulting file from the detection process.

#### *4.3.4.2 Testing using real scanned Arabic documents*

This part of the testing process was carried out in order to test the Ain classifier using images of real Arabic documents. Table 4.1 shows the different parameters of the document image used in this testing process, knowing that the same Arabic document was used but with different parameters. These parameters are colour type, image scan resolution and binarization threshold.

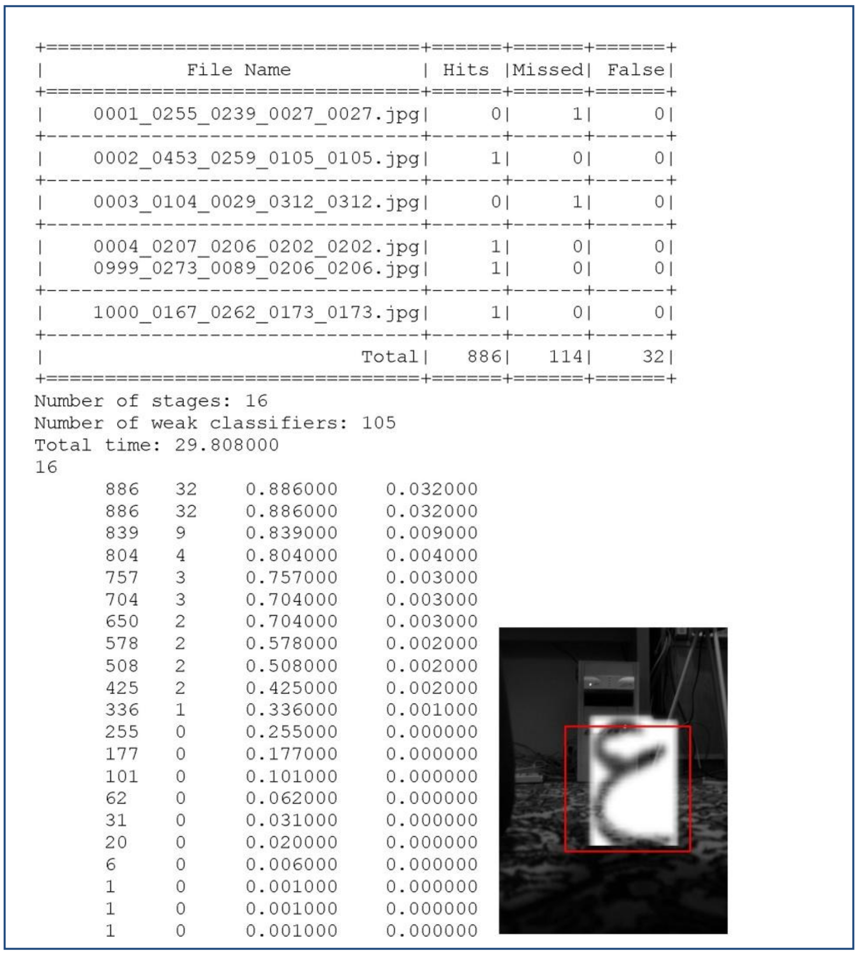
Table 4.1: Different image parameters used in testing the pilot experiment

| Document image | Colour type | Image resolution | Binarization threshold |
|---|---|---|---|
| Doc01 | Grey-scale | 300 | - |
| Doc02 | Grey-scale | 200 | - |
| Doc03 | Binary | 200 | 150 |
| Doc04 | Binary | 200 | 200 |
| Doc05 | Binary | 300 | 150 |
| Doc06 | Binary | 300 | 200 |

### 4.3.5 Results and conclusions

In the first part of testing, the experiment achieved 886 out of 1000 true positive (TP) hits, 114 false negative (FN) hits out of 1000 and 32 were false positive (FP) hits. Figure 4.4 shows a sample of a true positive hit image with the results of the experiment. Thus the accuracy of recognition of the Ain classifier was 88.6%. The testing accuracy of the Ain classifier is higher than that recorded by (Seo 2008) for face detection which was 76%. Figure 4.5 shows the relationship between the true positive rate and the false positive rate of the Ain classifier. It shows also the influence of the true positive rate on the false positive rate.

The second part of the testing process was carried out in order to test the Ain classifier using images of real Arabic documents. The Ain classifier recognized the binary-colour image with a resolution of 300 DPI and a binarization threshold value of 150 (Doc05 in Table 4.1) as shown in Figure 4.6 (a). It missed the correct letter (ع) and detected another incorrect letter (ح) in the case of the binary-colour image, with a resolution of 300 DPI and a binarization threshold of 200 (Doc06 in Table 4.1) as shown in Figure 4.6 (b). It fails to recognize the grey-scale images and the binary-colour images with 200 DPI. It was noticed that the second part of the testing failed with the grey-scale image because the images of letters used in the training process were binary-colour images. It failed in the resolution of 200 DPI because some of the training letters were created with a resolution of 300 DPI. Also the false hitting of letter (ع) and retrieval of letter (ح) instead was because the two letters have a somewhat similar shape.



```
+================================+======+======+======+
|           File Name            | Hits |Missed| False|
+================================+======+======+======+
|    0001_0255_0239_0027_0027.jpg|     0|     1|     0|
+--------------------------------+------+------+------+
|    0002_0453_0259_0105_0105.jpg|     1|     0|     0|
+--------------------------------+------+------+------+
|    0003_0104_0029_0312_0312.jpg|     0|     1|     0|
+--------------------------------+------+------+------+
|    0004_0207_0206_0202_0202.jpg|     1|     0|     0|
|    0999_0273_0089_0206_0206.jpg|     1|     0|     0|
+--------------------------------+------+------+------+
|    1000_0167_0262_0173_0173.jpg|     1|     0|     0|
+--------------------------------+------+------+------+
|                           Total|   886|   114|    32|
+================================+======+======+======+
Number of stages: 16
Number of weak classifiers: 105
Total time: 29.808000
16
      886   32    0.886000    0.032000
      886   32    0.886000    0.032000
      839   9     0.839000    0.009000
      804   4     0.804000    0.004000
      757   3     0.757000    0.003000
      704   3     0.704000    0.003000
      650   2     0.704000    0.003000
      578   2     0.578000    0.002000
      508   2     0.508000    0.002000
      425   2     0.425000    0.002000
      336   1     0.336000    0.001000
      255   0     0.255000    0.000000
      177   0     0.177000    0.000000
      101   0     0.101000    0.000000
      62    0     0.062000    0.000000
      31    0     0.031000    0.000000
      20    0     0.020000    0.000000
      6     0     0.006000    0.000000
      1     0     0.001000    0.000000
      1     0     0.001000    0.000000
      1     0     0.001000    0.000000
```

Figure 4.4: The pilot experiment testing results and detected Ain (ع) image

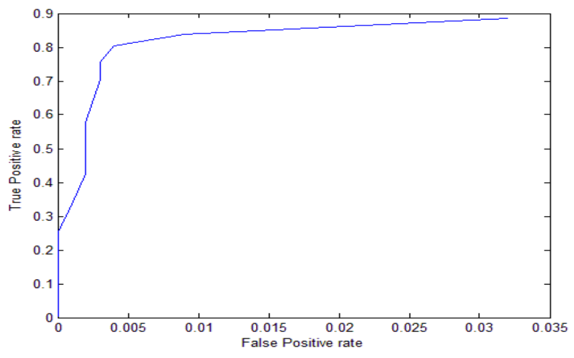
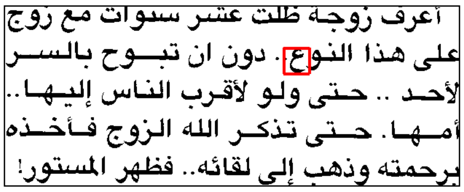


Figure 4.5: The sensitivity ROC of the pilot experiment of the Ain classifier

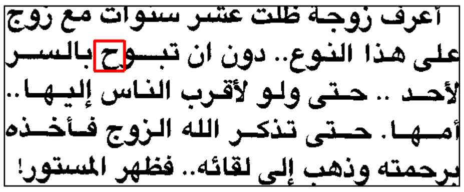
The results of the two testing parts of the experiment showed that the Haar-cascade classifier approach can be applied to Arabic character recognition. It gave higher accuracy than that of (Seo 2008) and showed success in recognizing a real Arabic document image.



أعرف زوجة ظلت عشر سنوات مع زوج
على هذا النوع. دون ان تبـــوح بالســر
لأحـد .. حـتى ولـو لأقـرب الناس إليـهـا..
أمـهـا. حــتـى تذكـر الله الـزوج فـأخـذه
برحمته وذهب إلى لقائه.. فظهر المستور!

(a)

The correct hit of letter (ع)



أعرف زوجة ظلت عشر سنوات مع زوج
على هذا النوع.. دون ان تبـــوح بالســر
لأحـد .. حـتى ولـو لأقـرب الناس إليـهـا..
أمـهـا. حــتـى تذكـر الله الـزوج فـأخـذه
برحمته وذهب إلى لقائه.. فظهر المستور!

(b)

The false hit of letter (ع) and hitting (ح) instead

Figure 4.6: Testing the pilot experiment with a real scanned Arabic document

## 4.4 The extended experiment

The previous pilot experiment examined the Haar-cascade classifier approach and attempted to ascertain whether it is applicable to Arabic character recognition. As this experiment requires the approach to be applied practically, it used real Arabic documents with 15 different glyphs which represent almost all of the Arabic glyphs. The generated classifiers were tested against real commercial software in order to ensure that the achieved accuracy of the approach is comparable to the commercial applications.

### 4.4.1 Planning the experiment

The pilot experiment suggested that the Haar-Cascade Classifier approach may be suitable for Arabic printed character recognition. Planning this experiment requires applying this approach practically and achieves high recognition accuracy. The practical apply of the approach to proof the first assumption regarding the pre-processing and character segmentation stages; it was assumed that the two stages can be skipped partially or totally when applying this approach to Arabic character recognition. This experiment tends to justify this first assumption by applying the approach in the following ways:

1. The binarization and noise removal step is skipped. The approach handles the original grey-scale images. For this reason the grey-scale images were used in the positive and negative document image datasets. Document images used were with a resolution of 300 DPI. This gave the classifier a better chance of good recognition accuracy.

2. The approach deals with the basic and rotated features of the glyphs so there is no need for the skew detection and correction steps. For this reason an application was created in order to generate rotated images from that of the negative and positive images (see section 3.1.5).

3. The text lines detection step is skipped. Each glyph is detected along with its location in the document image, so extracting the text lines is possible using the detected glyphs.

4. The normalization step is not needed because the HCC approach is scale invariant. For that reason, different font sizes are used in the computer generated document images and in those generated with artificial noise.

5. The character segmentation phase of the connected Arabic words or PAWs can be omitted when using the HCC approach. Cancelling the character segmentation step leads to improved recognition accuracy. For this reason the system was trained and tested using real Arabic document images with words and PAWs connected.

The HCC approach produces a different classifier for each Glyph. The 15 selected glyphs generated 15 different classifiers. As the total glyphs that represent the Arabic language are 112 glyphs (see Table 3.7), 15 glyphs were selected to cover almost all the cases of the Arabic glyphs from the following aspects:

1. Use glyphs that share the same joining groups and with different dots like Hah start (حـ) and Khah start (خـ).

2. Use glyphs that have almost the same shape like Reh end (ـر) and Waw end (ـو).

3. Use some middle shape glyphs for proof of ignoring the character segmentation phase, like Heh middle (ـهـ).

4. Use ligature glyphs like Lam Alef in its isolated shape (لا).

Three different types of document images were used; real scanned images, computer generated images and computer generated images with artificial noise. The computer generated types of documents included different font types, sizes and bold and italic fonts. The purposes of these different types of document images were to examine different types of documents, increase the samples of documents and to test the effect of each document type on the recognition accuracy.

### 4.4.2 Data preparation

The data files in this extended experiment are real datasets (i.e. images of Arabic documents). These Arabic document images act as both negative and positive images. The paragraph images datasets shown in section 3.2.1 were used in order to generate the negative and positive images. The negative image is the document image without the tested glyph inside it. The positive image is the document image with the tested glyph inside it. The task of generating the negative and positive images was carried out for the 15 different glyphs that were used in the experiment.

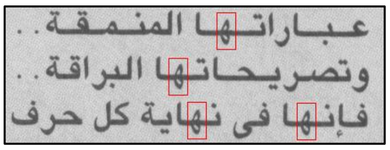
The MMAC corpus shown in Chapter 3 was used in running this experiment. The part of the corpus used in this experiment was the paragraph image dataset shown in section 3.2.1. This part originated from 15 different Arabic document images. Five of these documents were from real scanned documents (Real data), Five documents were computer generated (Computer data), and the last five were computer generated with artificial noise documents (Noise data). The Computer and Noise data used different Arabic fonts, sizes, bold and italic. These varieties in the Computer and Noise data helped in generating robust classifiers.

#### *4.4.2.1 Creating positive and negative images*

The data required for each of the 15 selected glyphs in this part of the experiment are:

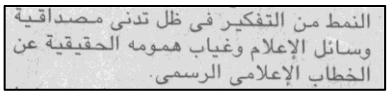
1. Positive images (images that do include the tested glyph on it). This part is separated into two parts; one for the training process and the other for the testing process. The positive images used for testing made up a quarter of the whole positive images, while the positive images used for training were three quarters of the whole positive images (Adolf 2003). Figure 4.7 (a) shows a sample of a positive image for the Heh middle glyph with four glyphs inside the image.

2. Negative images (images that do not include the tested glyph). These are used for the training process of the classifiers. Figure 4.7 (b) shows a sample of a negative image for the Heh middle glyph.



عبـاراتـهـا المنـمـقـة..
وتصـريـحـاتـهـا البراقة..
فإنـهـا فى نهاية كل حرف

(a)
A positive image of Heh middle



النمط من التفكير فى ظل تدنى مصداقية
وسائل الإعلام وغياب همومه الحقيقية عن
الخطاب الإعلامى الرسمى.

(b)
A negative image of Heh middle

Figure 4.7: Sample of positive and negative image for Heh middle glyph

A program was written in order to separate the positive from the negative images for each glyph. The program reads the ground truthed text file of each document image and determines whether or not the tested glyph exists in this document. If the glyph is missing from the document, the program copies the document image into the negative folder of this glyph. If the tested glyph does exist in the document, then the program copies the document's image into the positive folder of the glyph. The previous task is repeated for all 15 glyphs. The program also generates a spreadsheet in order to report the positive and negative images of all the glyphs. The spreadsheet also includes the number of occurrences of the tested glyph in the positive images.

The *Objectmaker* utility offered in OpenCV was used in order to manually define the places of the glyph in each positive document image. The utility shows each image separately to allow the user to define the containing rectangle of the glyph and to do the same for each glyph in the image. It then moves to the next image, and so on until completing all the positive images of the tested glyph. The utility generates a text file with a list of each image name, number of occurrences of the glyph, and containing rectangles. The containing rectangle includes the top left corner coordinates and the width and height. This process was completed manually by two different people; one to make the selection of the glyphs in all of the images and the other to validate that the selection of glyphs and the number of glyphs is correct. This process was a very laborious process and was very time consuming, but produced an excellent research resource.

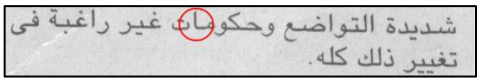
Table 4.2 shows the total number of positive and negative images for each glyph in the three different types of documents; the real scanned, computer generated and computer documents generated with artificial noise.

Table 4.2: The number of original negative and positive images of each glyph

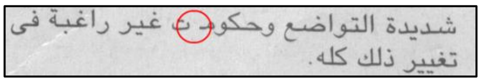
| Glyph Name | Glyph shape | Real data | | Computer data | | Noise data | |
|---|---|---|---|---|---|---|---|
| | | Pos. | Neg. | Pos. | Neg. | Pos. | Neg. |
| Alef Iso | ا | 198 | 25 | 116 | 25 | 163 | 25 |
| Alef End | ـا | 192 | 31 | 115 | 26 | 162 | 26 |
| The Mid | ـتـ | 150 | 73 | 117 | 24 | 152 | 36 |
| Theh Mid | ـثـ | 30 | 193 | 24 | 117 | 47 | 141 |
| Hah Iso | ح | 4 | 219 | 4 | 137 | 11 | 177 |
| Hah Str | حـ | 61 | 162 | 45 | 96 | 83 | 105 |
| Khah Str | خـ | 46 | 177 | 38 | 103 | 35 | 153 |
| Reh End | ـر | 182 | 41 | 131 | 10 | 162 | 26 |
| Seen Mid | ـسـ | 107 | 116 | 63 | 78 | 84 | 104 |
| Sad Mid | ـصـ | 51 | 172 | 37 | 104 | 59 | 129 |
| Ain Iso | ع | 5 | 218 | 14 | 127 | 6 | 182 |
| Lam Str | لـ | 193 | 30 | 116 | 25 | 159 | 29 |
| Heh Mid | ـهـ | 103 | 120 | 59 | 82 | 86 | 102 |
| Waw End | ـو | 168 | 55 | 111 | 30 | 151 | 37 |
| Lam-Alef Iso | لا | 121 | 102 | 84 | 57 | 100 | 88 |

The study of the relationship between the total numbers of positive and negative images for each glyph shows three different categories of the Arabic glyphs. These three Arabic glyph categories are:

1. Glyphs that exist in almost all of the images. These glyphs have a very small number of negative images or sometimes no negative images at all. This problem was solved by editing the images that contained a small number of the glyphs (one or two) and removed the glyphs manually by putting the background of the image over the glyph in the document image. These images were then moved from the positive to the negative images dataset. These glyphs are Alef isolated (ا), Alef end (ـا) and Lam start (لـ). Figure 4.8 (a) shows the positive image containing the glyph Alef end, while Figure 4.8 (b) shows the same document after removing the Alef end glyph and converting it to a negative image.


شديدة التواضع وحكومات غير راغبة فى
تغيير ذلك كله.

(a)
A positive document image of Alef end


شديدة التواضع وحكومـت غير راغبة فى
تغيير ذلك كله.

(b)
Converting positive image to negative

Figure 4.8: Sample of converting a positive image to negative

2. Glyphs that rarely appear in the images which have very small number of positive images. These glyphs were left without any image editing because any editing carried out in order to add a certain glyph would be very artificial. However, good results are not expected from their classifiers. These glyphs are Hah isolated (ح) and Ain isolated (ع).

3. Most of the glyphs have a reasonable ratio between the negative and positive images.

The Lam start glyph sometimes has another ligature glyph when followed by certain letters such as Meem or Jeem, as in some Arabic fonts like Traditional Arabic. For example (لمـ) becomes (لـ) and (لجـ) becomes (لـ). The new ligature glyphs were ignored when using the Lam start glyph as they are considered to be different ligatures. The glyph (ؤ) was included while using the glyph Waw end but the Hamza (ء) was ignored in order to be included during the training process of this glyph.

#### *4.4.2.2 Creating numerous samples of positive and negative images*

Running the experiment requires a huge amount of negative and positive images which are not available for the test data. A computer program was developed in order to generate the required huge amount of positive and negative document images from the available test images. The main purpose of this program was to create as many rotated samples as required from the original image. The inputs are the number of samples, the number of generated rotated samples and a list of images either negative or positive. This program was mentioned above in section 3.1.5.1. The program uses two algorithms; the nearest neighbour interpolation algorithm in order to rotate the images and the Box-Muller transform algorithm in order to generate a normal distribution of random numbers from the computer generated uniform distribution of random numbers.

The nearest neighbour interpolation algorithm (Sonka, Hlavac et al. 1998) is used to rotate the image. This algorithm rotates the image in a reserved way. It scans the destination image pixel by pixel and calculates the value of the equivalent pixel in the source image. If the rotation proceeded in the standard way, which would be to scan each pixel in the source image and then calculate the equivalent in the destination image, then some pixels in the destination image would have no value. The outer areas in the destination images were set to white pixels.

The Box-Muller transform algorithm (Box and Muller 1958) was used to define the rotation angle of the document. It converts the random numbers that are generated from most of the programming language compilers to normally distributed random numbers. The purpose of using this algorithm is in order to simulate the rotation angle that might happen in real life when scanning the document page.

The document rotation angles use standard deviation sigma $\sigma = 5$ and mu $\mu = 0$. Mu $\mu$ as the mean value and it is considered to be angle 0 as the scanning process may be skewed in any direction with the same value. Sigma $\sigma$ is the standard deviation of the angles and when uses 5, this means that 66.6% falls within $\pm 5°$.

The developed program attempted to produce a number of positive and negative images at the end totalling more than 2,000 images in most cases. This number of images was recommended by (OpenCV 2002; Adolf 2003; Seo 2008) for the HCC approach to run properly. The same method of generating numerous positive or negative images from the limited number of images available was by adding rotated samples of images as shown in (Lienhart, Kuranov et al. 2002).

### 4.4.3 Data selection

The selection of glyphs and documents was made in a way that represented a satisfactory quantity, of different quality and varieties in the sizes and rotations. This research seeks to discover whether the documents and glyphs will represent the problem in such a way as to prove or reject the initial assumption.

#### *4.4.3.1 The Selected Glyphs*

The glyph selection process was carried out in order to maximize the representation of all the Arabic glyphs. The selection of the Arabic letters and their locations was done in order to represent the variety of glyphs from simple to difficult. Some samples of glyphs that have similar shapes were also included. It also had glyphs of identical shape but with different dots. It is concluded from this selection that in most cases, the Arabic glyphs are covered in order to discover how applicable the HCC approach would be with the full printed Arabic alphabet. Table 4.3 shows the list of 15 selected glyphs with their names, locations and reasons for selecting them. Table 4.3 also shows the percentage of occurrences of each glyph in the MMAC corpus. The total percentage of occurrences of the 15 selected glyphs represents more than one third of the total occurrence of all the Arabic glyphs. The selected glyphs represent the four glyph locations that exist in the Arabic language. The number of glyphs at the starting location is three, five at the middle location, three at the end location and four at an isolated location.

Table 4.3: Proposed glyphs with occurrences, locations and reasons for selection

| No | English Name | Occurrence | Glyph | Location | Reason of selection |
|---|---|---|---|---|---|
| 1 | ALEF | 10.46% | ا | Isolated | Very common letter and simple |
| 2 | ALEF | 6.34% | ـا | End | opposite shape of No 12 |
| 3 | TEH | 2.05% | ـتـ | Middle | Same letter as No 4 with different dots |
| 4 | THEH | 0.29% | ـثـ | Middle | Same as letter No 3 with different dots |
| 5 | HAH | 0.07% | ح | Isolated | Complicated and conflicts No 11 |
| 6 | HAH | 0.73% | حـ | Start | Same as No 7 with different dots |
| 7 | KHAH | 0.43% | خـ | Start | Same as No 6 with different dots |
| 8 | REH | 3.41% | ـر | End | Conflicts with No 14 |
| 9 | SEEN | 1.05% | ـسـ | Middle | Conflicts with No 10 |
| 10 | SAD | 0.60% | ـصـ | Middle | Conflicts with No 9 |
| 11 | AIN | 0.16% | ع | Isolated | Complicated and Conflicts with No 5 |
| 12 | LAM | 6.93% | لـ | Start | Opposite shape of No 2 |
| 13 | HEH | 0.96% | ـهـ | Middle | Very complicated glyph |
| 14 | WAW | 2.59% | ـو | End | Conflicts with No 8 |
| 15 | Lam Alef | 1.42% | لا | Isolated | Very common sample of ligature |

### 4.4.3.2 *The selected documents*

The documents here selected to cover the varying quality of document images in additional to a variety of font formats. The paragraph type of the MMAC corpus was used as shown in section 3.2.1 because each document contains 10-20 words. This small number of words in the document enables the making of lists of negative and positive document images. If each document is bigger this reduces the existence of negative images. Figure 3.4 shows different samples of the document images. These samples are real scanned, computer generated and computer generated documents with artificial noise.

### 4.4.4 Training the extended experiment classifiers

The training process of the extended experiment was a very lengthy process. It takes a long time to prepare the files and folders for the training process. The training process itself took a long time to be finished. The process occupied 60 PCs in the main computer lab of the School of Computer Science at the University of Nottingham. Each machine has Intel processor Core 2 Due and 2 GB of RAM running Windows XP 32 bits. The duration of the experiment lasted for around two weeks of continuous running. Each machine in the lab hosted the training processes of one of the 15 selected glyphs. Each glyph had four different training processes, for Real data, for Computer data, for Noise data and for All data. The All data training process was the collecting together of the other three positive and negative data in order to generate a bigger dataset.

Defining the training size of the glyph was a very important issue to address. The size was initially defined as the average width and height of the containing rectangles of the glyph in all the glyph positive images. That initial value of the width and height of the training size was changed in order to achieve the best results when testing the resultant classifier. The trials of the experiment in width and height showed that the optimum size is between 35 to 50 pixels of the sum of width and height. The ratio between them is varied upon their average ratio. The same width and height was set for each glyph and was used in all the datasets (Real, Computer, Noise and All).

Table 4.4 shows for each of the selected glyphs the trained width and height. It also shows the total number of negative images, total number of positive images and the total number of glyphs in the positive images for each glyph in every document type. The table also shows that the total number of positive images of the 15 selected glyphs is 63,865 images; the total number of negative images of the 15 selected glyphs is 91,455 images; the total number of the 15 selected glyphs of all the positive images is 145,331 and the average number of glyphs in every positive image is 2.27.

Table 4.4: The training information for each of the 15 selected glyphs

| Glyph Name | Width | Height | Real data | | | Computer data | | | Noise data | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Pos. Img | Neg. Img | Total Glyphs | Pos. Img | Neg. Img | Total Glyphs | Pos. Img | Neg. Img | Total Glyphs |
| Alef Iso | 10 | 23 | 1639 | 1775 | 9592 | 1392 | 1775 | 10528 | 1353 | 1775 | 9482 |
| Alef End | 13 | 29 | 1584 | 2201 | 6270 | 1392 | 1846 | 6752 | 1342 | 1846 | 6259 |
| Teh Mid | 11 | 22 | 2352 | 1898 | 4368 | 1408 | 1704 | 2832 | 1840 | 2016 | 3728 |
| Theh Mid | 18 | 35 | 1562 | 2123 | 1704 | 1349 | 1872 | 1491 | 1476 | 2256 | 1722 |
| Hah Iso | 18 | 23 | 213 | 2409 | 284 | 213 | 2192 | 213 | 568 | 1947 | 568 |
| Hah Str | 23 | 24 | 2116 | 1782 | 2714 | 1564 | 2016 | 1932 | 1612 | 2205 | 2106 |
| Khah Str | 23 | 26 | 1610 | 1947 | 2070 | 1530 | 2163 | 1734 | 1456 | 2448 | 1792 |
| Reh End | 10 | 16 | 1507 | 2091 | 3410 | 1584 | 710 | 3728 | 1353 | 1846 | 3641 |
| Seen Mid | 19 | 17 | 1701 | 2436 | 2121 | 1457 | 2028 | 2046 | 1638 | 2184 | 2262 |
| Sad Mid | 21 | 17 | 1599 | 2752 | 2009 | 1568 | 2184 | 1904 | 1584 | 2064 | 2268 |
| Ain Iso | 17 | 24 | 284 | 2398 | 284 | 781 | 2032 | 781 | 355 | 2002 | 355 |
| Lam Str | 13 | 30 | 1584 | 1830 | 5610 | 1392 | 1775 | 6224 | 1320 | 2059 | 6193 |
| Heh Mid | 18 | 33 | 1638 | 2520 | 2394 | 1584 | 2132 | 2160 | 1690 | 2142 | 2210 |
| Waw End | 14 | 23 | 1397 | 1980 | 2585 | 1743 | 1980 | 3759 | 1840 | 2072 | 3824 |
| Lam-Alef Iso | 16 | 23 | 1456 | 2142 | 2192 | 1664 | 2052 | 2626 | 1575 | 1848 | 2604 |

## 4.4.5 Testing the extended experiment classifiers

The testing process of the extended experiment was separated into two different parts. The first used the *performance* utility offered in OpenCV. The second tested the HCC glyphs classifier against real commercial software in order to compare the results of the selected 15 glyphs classifier.

The main concerns in the testing process were the values of True Positive (TP), False Negative (FN) and False Positive (FP) ratios in all tests. The True Positive (TP) is the number or ratio of the glyphs that were detected correctly and is also known as hits. An example of TP would be when the classifier detects Alef end correctly as Alef end. False Positive (FP) is when glyphs are detected wrongly and is also known as false alarm. An example of FP would be when the classifier detects a glyph as Alef end when it was actually Lam start. Finally, False Negative (FN) is when glyphs are not detected at all even though they exist in the document, also known as a miss. When the classifier misses an Alef end glyph even though it exists in the document image, this would be an example of FN.

### *4.4.5.1 Testing using the OpenCV performance utility*

The process of testing the classifiers using the *performance* utility is the same as the testing that was done in the previous pilot experiment. During testing this experiment tried to investigate the influence of the testing parameters over the classifier detection accuracy. There are two parameters which have a very big effect on the detection accuracy; these two parameters are the scale factor and the minimum neighbours (Lienhart, Kuranov et al. 2002; Seo 2008; Kasinski and Schmidt 2010).

The scale factor affects the detection. The detection phase starts with a sliding window the original size of the width and height and enlarges the window size depending on the scale factor. A scale factor of 1.1 means that the window is enlarges each time by 10% of its size. An increase of the scale factor reduces the detection time as the numbers of sliding windows decreases (Lienhart, Kuranov et al. 2002). This experiment showed that a suitable scale factor for Arabic glyph detection is either 1.01 or 1.02. The minimum neighbour is the number of neighbour regions that are merged together during detection in order to form only one object (Lienhart, Kuranov et al. 2002). The increase in the minimum neighbour reduces the ratio of False Positives (FP). The experiment showed that a suitable minimum neighbour value usually lies between 1 and 3 inclusive.

Table 4.5 shows for each of the selected glyphs the trained width and height. It also shows for each glyph the total number of positive images and the total number of glyphs in the positive images for every document type. The table also shows that the total number of positive images of the 15 selected glyphs is 20,700 images. The total number of the 15 selected glyphs in all the positive images is 47,742. The average number of glyphs in every positive image is 2.30.

Table 4.5: The testing information for each of the 15 selected glyphs

| Glyph Name | Width | Height | Real data | | Computer data | | Noise data | |
|---|---|---|---|---|---|---|---|---|
| | | | Pos. Img. | Total Glyphs | Pos. Img. | Total Glyphs | Pos. Img. | Total Glyphs |
| Alef Iso | 10 | 23 | 539 | 3168 | 464 | 3488 | 440 | 3190 |
| Alef End | 13 | 29 | 528 | 2189 | 448 | 2352 | 440 | 2178 |
| Teh Mid | 11 | 22 | 798 | 1470 | 464 | 1024 | 592 | 1184 |
| Theh Mid | 18 | 35 | 568 | 568 | 355 | 426 | 451 | 451 |
| Hah Iso | 18 | 23 | 71 | 71 | 71 | 71 | 213 | 213 |
| Hah Str | 23 | 24 | 690 | 1058 | 506 | 598 | 546 | 650 |
| Khah Str | 23 | 26 | 506 | 598 | 408 | 510 | 504 | 560 |
| Reh End | 10 | 16 | 495 | 1078 | 512 | 1392 | 429 | 1133 |
| Seen Mid | 19 | 17 | 546 | 651 | 496 | 682 | 546 | 754 |
| Sad Mid | 21 | 17 | 492 | 574 | 504 | 504 | 540 | 828 |
| Ain Iso | 17 | 24 | 71 | 71 | 213 | 213 | 71 | 71 |
| Lam Str | 13 | 30 | 539 | 1848 | 464 | 2128 | 429 | 1991 |
| Heh Mid | 18 | 33 | 525 | 714 | 540 | 792 | 546 | 754 |
| Waw End | 14 | 23 | 451 | 1023 | 588 | 1134 | 576 | 1104 |
| Lam-Alef Iso | 16 | 23 | 480 | 672 | 520 | 858 | 525 | 756 |

#### *4.4.5.2 Testing using a commercial Arabic OCR application*

The objective of this testing process is to compare the HCC approach to Arabic OCR commercial software. This test gives realistic measures of the performance of the HCC approach when comparing it with commercial software. The testing examines just the 15 selected glyphs in both the HCC approach and the commercial software. Well known commercial software was used in order to perform this test; Readiris Pro 10 (IRIS 2004) which was used above in section 3.3. The 62 previously selected paragraph documents were used which represent a variety of document types that occur in most real life cases.

A program was developed in order to calculate the number of recognized glyphs in each of the 62 documents. The commercial software was used in comparison with the total number of glyphs detected in the ground truthed document. A second manual check was done in order to validate the outcome of the program. The values of TP, FN and FP of the glyphs were calculated for each of the selected glyphs with the 62 document images.

Testing the HCC approach was carried out in the same way as in the previous section. It used the *performance* utility in order to test the recognition of each glyph classifier. The 15 XML classifiers were also brought from the previous part of the testing in order to use them in this part. The values of TP, FN and FP of the glyphs were calculated for each of the selected glyphs with the 62 document images (see section 4.4.6.2).

### 4.4.6 Results and conclusions

The extended experiment results showed a real success in recognizing the Arabic documents. Good results were produced in the case of using the OpenCV testing utility as well as when compared with commercial software. The results were encouraging but suggested that some auxiliary experiments needed to be carried out in order to be certain of using the best parameters with the approach before applying the final experiment using all the Arabic language glyphs. The results of the experiment were separated into two different parts, one using the OpenCV *performance* utility and the other when compared to commercial software.

#### 4.4.6.1 *Results of using the OpenCV performance utility*

The extended experiment led to a comparison between three different types of document images. The three types gave different accuracies with each glyph. Table 4.6 shows the TP, FN and FP ratio of each glyph with the three types of document images and all the images. The last row shows the average of percentage accuracy of all the 15 selected glyphs. The FP "percentage" can be greater than 100% as it is calculated using the formula: $\% = \frac{\mathrm{FP}}{\mathrm{TP+FN}}$.

Table 4.6: Percentage testing results of the extended experiment for 15 glyphs

| Glyph Name | Real data | | | Computer data | | | Noise data | | | All data | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP | FN | FP | TP | FN | FP | TP | FN | FP | TP | FN | FP |
| Alef Iso | 78% | 22% | 9% | 84% | 16% | 2% | 96% | 4% | 9% | 74% | 26% | 10% |
| Alef End | 87% | 13% | 16% | 76% | 24% | 4% | 89% | 11% | 24% | 51% | 49% | 22% |
| Teh Mid | 90% | 10% | 17% | 75% | 25% | 22% | 95% | 5% | 68% | 76% | 24% | 17% |
| Theh Mid | 79% | 21% | 12% | 57% | 43% | 21% | 76% | 24% | 18% | 77% | 23% | 16% |
| Hah Iso | 45% | 55% | 0% | 68% | 32% | 24% | 100% | 0% | 112% | 99% | 1% | 2% |
| Hah Str | 86% | 14% | 6% | 95% | 5% | 7% | 91% | 9% | 14% | 92% | 8% | 13% |
| Khah Str | 96% | 4% | 8% | 61% | 39% | 17% | 84% | 16% | 6% | 87% | 13% | 12% |
| Reh End | 79% | 21% | 21% | 91% | 9% | 17% | 92% | 8% | 68% | 86% | 14% | 10% |
| Seen Mid | 90% | 10% | 8% | 83% | 17% | 15% | 70% | 30% | 15% | 82% | 18% | 14% |
| Sad Mid | 96% | 4% | 14% | 77% | 23% | 15% | 93% | 7% | 9% | 95% | 5% | 11% |
| Ain Iso | 100% | 0% | 0% | 61% | 39% | 128% | 89% | 11% | 3% | 97% | 3% | 0% |
| Lam Str | 89% | 11% | 9% | 79% | 21% | 6% | 91% | 9% | 7% | 74% | 26% | 14% |
| Heh Mid | 94% | 6% | 7% | 98% | 2% | 2% | 99% | 1% | 10% | 97% | 3% | 7% |
| Waw End | 94% | 6% | 8% | 81% | 19% | 13% | 88% | 12% | 26% | 87% | 13% | 9% |
| LamAlef Iso | 98% | 2% | 2% | 99% | 1% | 4% | 93% | 7% | 11% | 98% | 2% | 2% |
| Average | 87% | 13% | 9% | 79% | 21% | 20% | 90% | 10% | 27% | 85% | 15% | 11% |

Table 4.7 shows the TP, FN and FP values for each glyph with the three types of document and all images. The last row shows the average accuracy percentage of the 15 selected glyphs calculated according to the total number in each dataset.

Table 4.8 shows the TP, FN and FP ratio for each location of the glyphs with the three types of document and all images. The last row shows the overall percentage for all four locations calculated according to the total number in each location. It is important to note that the overall accuracy in Table 4.7 and Table 4.8 are of the same values.

Table 4.7: Numbers testing results of glyphs using three types of document

| Glyph Name | Real data | | | Computer data | | | Noise data | | | All data | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP | FN | FP | TP | FN | FP | TP | FN | FP | TP | FN | FP |
| Alef Iso | 2478 | 690 | 276 | 2925 | 563 | 82 | 3073 | 117 | 276 | 7264 | 2582 | 977 |
| Alef End | 1900 | 289 | 348 | 1776 | 576 | 90 | 1947 | 231 | 525 | 3419 | 3300 | 1445 |
| Teh Mid | 1330 | 140 | 255 | 768 | 256 | 226 | 1121 | 63 | 808 | 2801 | 877 | 634 |
| Theh Mid | 450 | 118 | 66 | 242 | 184 | 89 | 345 | 106 | 82 | 1119 | 326 | 233 |
| Hah Iso | 32 | 39 | 0 | 48 | 23 | 17 | 213 | 0 | 239 | 353 | 2 | 6 |
| Hah Str | 908 | 150 | 66 | 566 | 32 | 43 | 594 | 56 | 89 | 2116 | 190 | 299 |
| Khah Str | 577 | 21 | 48 | 310 | 200 | 87 | 472 | 88 | 34 | 1450 | 218 | 197 |
| Reh End | 853 | 225 | 224 | 1267 | 125 | 243 | 1046 | 87 | 772 | 3105 | 498 | 372 |
| Seen Mid | 586 | 65 | 52 | 567 | 115 | 102 | 530 | 224 | 110 | 1720 | 367 | 298 |
| Sad Mid | 549 | 25 | 82 | 389 | 115 | 76 | 769 | 59 | 75 | 1816 | 90 | 202 |
| Ain Iso | 71 | 0 | 0 | 130 | 83 | 273 | 63 | 8 | 2 | 344 | 11 | 1 |
| Lam Str | 1645 | 203 | 167 | 1689 | 439 | 128 | 1820 | 171 | 146 | 4409 | 1558 | 853 |
| Heh Mid | 673 | 41 | 47 | 775 | 17 | 19 | 750 | 4 | 77 | 2183 | 77 | 150 |
| Waw End | 962 | 61 | 78 | 922 | 212 | 142 | 970 | 134 | 285 | 2831 | 430 | 298 |
| Lam-Alef Iso | 659 | 13 | 11 | 852 | 6 | 33 | 704 | 52 | 82 | 2235 | 51 | 57 |
| Total | 13673 | 2080 | 1720 | 13226 | 2946 | 1650 | 14417 | 1400 | 3602 | 37165 | 10577 | 6022 |
| Accuracy | 87% | 13% | 11% | 82% | 18% | 10% | 91% | 9% | 23% | 78% | 22% | 13% |

Table 4.8: Testing results in percentages defined by glyph location

| Location | Real data | | | Computer data | | | Noise data | | | All data | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP | FN | FP | TP | FN | FP | TP | FN | FP | TP | FN | FP |
| Isolated | 81% | 19% | 7% | 85% | 15% | 9% | 96% | 4% | 14% | 79% | 21% | 8% |
| Start | 89% | 11% | 8% | 79% | 21% | 8% | 90% | 10% | 8% | 80% | 20% | 14% |
| Middle | 90% | 10% | 13% | 80% | 20% | 15% | 89% | 11% | 29% | 85% | 15% | 13% |
| End | 87% | 13% | 15% | 81% | 19% | 10% | 90% | 10% | 36% | 69% | 31% | 16% |
| All | 87% | 13% | 11% | 82% | 18% | 10% | 91% | 9% | 23% | 78% | 22% | 13% |

From the results of the experiment, the following can be observed:

1. The glyphs with a smaller number of positive or negative samples return unreasonable results. For example, the case of Hah isolated real (ح), Ain isolated computer (ع), Reh end noise (ـر) and Theh middle computer (ـثـ).

2. The glyphs with a balanced number of positive and negative samples usually return good results in all types of documents. For example, Hah start (حـ), Seen middle (ـسـ) and Heh middle (ـهـ).

3. The glyphs with the same shape but different numbers of dots usually present better results when the glyph contains fewer rather than more dots. For example, Teh middle (ـتـ) with Theh middle (ـثـ) and Hah start (حـ) with Khah start (خـ).

4. The glyphs Alef isolated (ا), Alef end (ـا) and Lam start (لـ) have almost the same shape and represent 23.7% of the glyphs in the language as indicated by the MMAC corpus (see Chapter 3). These glyphs require an extra post-processing algorithm in order to improve their recognition or an extra algorithm to aid recognition (AbdelRaouf, Higgins et al. 2010).

5. The higher the number of glyphs samples the better the recognition accuracy.

6. The computer dataset is the least accurate of all the datasets as it includes different types of fonts and styles (italic, bold). Also when the background of the document image is very bright the recognition accuracy is reduced.

7. There are no significant differences between the four different locations of the glyphs in the PAW. The overall accuracy of all of them is almost the same.

#### *4.4.6.2 Results of testing with the commercial OCR application*

Comparing the HCC approach to the commercial software leads to almost the same degree of accuracy. Table 4.9 shows a comparison between the results of the two approaches. The results show that the HCC gets marginally better recognition accuracy (TP) of 92% compared to the commercial software’s 91%. The false alarm rate (FP) for the commercial software at 8% is much better than the HCC approach which was 12%.

The accuracy of the HCC approach in this test is much better than that of the *OpenCV performance* test (see section 4.4.6.1) because the HCC approach contains only un-rotated documents which sometimes lead to loss in the recognition accuracy.

Figure 4.9 shows the relationship in the percentage accuracy of the 15 selected glyphs using the two different approaches. The figure also shows that the accuracy of the two approaches is almost the same except for the glyphs with very small number of examples like Hah isolated (ح) and Ain isolated (ع).

Table 4.9: Testing results between the HCC and commercial software

| Glyph name | Total glyphs | HCC Approach | | | Commercial software | | |
|---|---|---|---|---|---|---|---|
| | | TP | FN | FP | TP | FN | FP |
| Alef Iso | 380 | 371 | 9 | 38 | 368 | 12 | 47 |
| Alef End | 286 | 270 | 16 | 42 | 267 | 19 | 19 |
| Teh Mid | 98 | 86 | 12 | 13 | 77 | 21 | 3 |
| Theh Mid | 8 | 7 | 1 | 2 | 6 | 2 | 7 |
| Hah Iso | 4 | 4 | 0 | 0 | 2 | 2 | 0 |
| Hah Str | 26 | 25 | 1 | 9 | 24 | 2 | 5 |
| Khah Str | 23 | 19 | 4 | 8 | 19 | 4 | 0 |
| Reh End | 134 | 114 | 20 | 18 | 122 | 12 | 14 |
| Seen Mid | 37 | 36 | 1 | 0 | 30 | 7 | 3 |
| Sad Mid | 18 | 18 | 0 | 4 | 14 | 4 | 1 |
| Ain Iso | 3 | 2 | 1 | 0 | 1 | 2 | 0 |
| Lam Str | 250 | 211 | 39 | 34 | 222 | 28 | 13 |
| Heh Mid | 35 | 34 | 1 | 3 | 32 | 3 | 3 |
| Waw End | 96 | 91 | 5 | 8 | 88 | 8 | 3 |
| Lam-Alef Iso | 60 | 59 | 1 | 1 | 57 | 3 | 2 |
| Total | 1458 | 1347 | 111 | 180 | 1329 | 129 | 120 |
| Accuracy | | 92% | 8% | 12% | 91% | 9% | 8% |

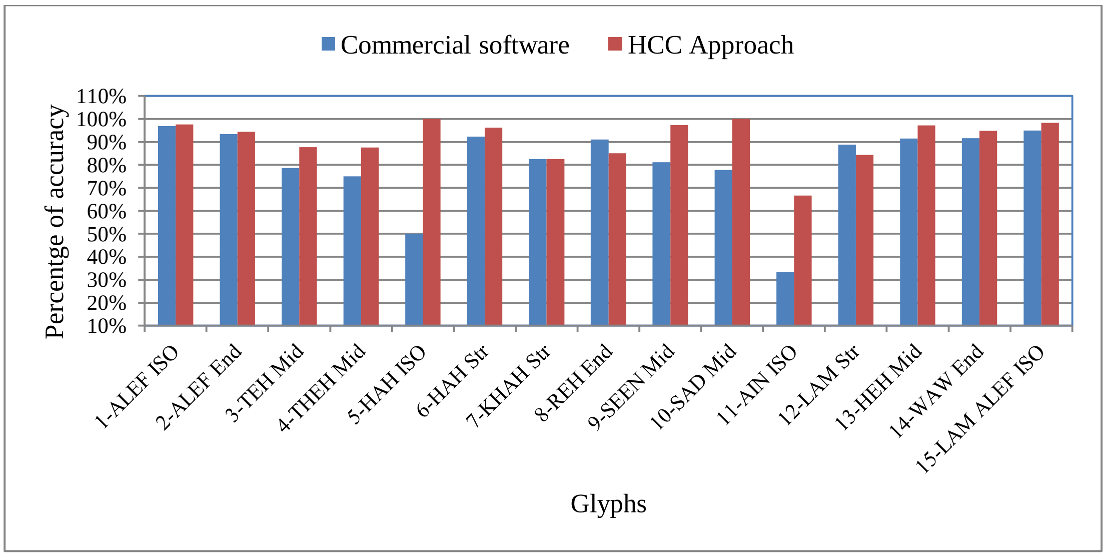


Figure 4.9: Recognition accuracy between HCC and commercial software

#### *4.4.6.3 Extended experiment Conclusions*

The overall conclusions of the extended experiment show that the HCC approach is very suited to the recognition of printed Arabic characters. The HCC approach results and the comparison with the commercial software show that it can reach a level of accuracy competing with commercial software developed, tested and enhanced over time. When the HCC approach reaches the same accuracy level as the commercial software without any post-processing stages, it can be considered as a contribution not only in terms of research but also to the market.

Due to the previous results in the extended experiment, the following recommendations are concluded for the forthcoming final experiment:

1. There is no need to have different types of datasets (Real, Computer and Noise). Using only the real dataset is enough and returns good classifiers.

2. Minimize the glyphs with a limited number of negative or positive images. This can be achieved by getting more samples for these glyphs from the computer and noise datasets.

3. There is no need to generate different classifiers for the glyphs with the same joining groups and different dots. It is preferable to generate only one classifier for the glyphs sharing the same joining group. This leads to naked glyph classifiers.

4. Increase the precision of the manual work in defining the containing rectangles of the glyphs when generating the positive datasets.

5. Undertake more trials in defining the best width and length for each glyph in order to produce results with better recognition accuracy. The other parameters must also be considered.

It is now very clear that the HCC approach can recognize printed Arabic glyphs without segmenting them.

## 4.5 The auxiliary experiments

Auxiliary experiments were planned in order to improve the training process by choosing the best parameters for the training process. The auxiliary experiments were planned after completing the extended experiment with satisfactory results. These experiments were done to prepare for the final experiment by trying to improve the training process as much as possible in order to achieve better recognition accuracy.

The parameters that have the most influence on the training process and that will be studied during these experiments are: number of splits; minimum hit rate; the boosting algorithm and the mode of the Haar-like features. The width and height of the sample are also important parameters.

### 4.5.1 Testing the training width and height

This experiment was performed in order to obtain the best training values for the width and height of the glyphs. The training width and height of a glyph represents the size of the glyph that the classifier will use in order to detect the glyph and are measured in pixels. Six different glyphs were examined that represent different styles of glyphs. Five different values were tested for the width and height of each glyph in order to get the best values.

#### *4.5.1.1 Running the experiment*

The experiment used the same procedures of training and testing the glyphs that were used in the extended experiment. The experiment used the same techniques of data preparation, training and testing. Six different glyphs were selected that represent different shapes, sizes and ratios between width and height. The glyphs were Alef isolated (ا), Theh middle (ـثـ), Khah start (خـ), Seen middle (ـسـ), Heh middle (ـهـ) and Lam start (لـ). The selected glyphs had already been used in the extended experiment, so the data was already prepared. Each glyph was trained with the five different values of widths and heights and then tested. The values of widths and heights were divided approximately equally starting from the minimum width and height of the glyph and ending with the maximum width and height of the glyph.

#### *4.5.1.2 Results and conclusions*

The results of the experiment indicate that the middle values of the widths and heights are the best values and produce a classifier with a higher precision. Table 4.10 shows the results of the experiment. Trial 1 is the smallest size of widths and heights increasing to Trial 5, which provides the maximum size. Each trial includes width in pixels (W), height in pixels (H), true positive percentage (TP) and false positive percentage (FP).

The smallest sizes of width and height in Trial 1 generate classifiers that usually detect incorrect shapes in the glyphs. These classifiers detect part of the glyph and consider it as being the glyph itself. On the other hand, the large size of widths and heights in Trial 5 generates classifiers which can miss detecting the smaller sized glyphs.

As a final conclusion from this experiment, different sizes of the widths and heights must be inspected in the training process in order to produce the best classifier. It is clear from the results of the experiment that the difference in the width and height of the training produces classifiers with a high variety of accuracy. It is concluded also that the middle size (Trial 3) usually offers the best values and generates better recognition classifiers.

Ideally plotting 3D graph to explain the relationship between the width and height is more convenient; however this will lead to have many trials for every single glyph and also because of the different ratios between width and height of the glyphs. It was found that an approximation value based on experience of the enormous previous trials would be enough to get the optimum width and height.

Table 4.10: Testing different training values of widths and heights of glyphs

| | Trial 1 | | | | Trial 2 | | | | Trial 3 | | | | Trial 4 | | | | Trial 5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | W | H | TP | FP | W | H | TP | FP | W | H | TP | FP | W | H | TP | FP | W | H | TP | FP |
| Alef Iso | 10 | 23 | 64 | 8 | 12 | 28 | 41 | 3 | 15 | 34 | 67 | 8 | 17 | 40 | 33 | 3 | 20 | 46 | 19 | 7 |
| Theh Mid | 11 | 22 | 50 | 3 | 15 | 30 | 54 | 7 | 18 | 35 | 71 | 3 | 21 | 40 | 69 | 4 | 24 | 48 | 19 | 2 |
| Khah Str | 20 | 20 | 1 | 0 | 23 | 26 | 80 | 1 | 26 | 31 | 73 | 1 | 29 | 36 | 85 | 3 | 33 | 44 | 78 | 1 |
| Seen Mid | 19 | 17 | 81 | 15 | 24 | 22 | 75 | 8 | 28 | 26 | 80 | 6 | 32 | 30 | 83 | 6 | 38 | 36 | 71 | 6 |
| Heh Mid | 12 | 23 | 83 | 10 | 15 | 29 | 86 | 7 | 18 | 33 | 91 | 6 | 22 | 40 | 88 | 10 | 25 | 45 | 76 | 7 |
| Lam Str | 8 | 18 | 78 | 23 | 10 | 22 | 69 | 25 | 13 | 30 | 77 | 10 | 16 | 37 | 73 | 5 | 19 | 49 | 29 | 5 |

## 4.5.2 Testing the number of splits and minimum hit rate

This experiment was performed in order to obtain the best values for the number of splits and the minimum hit rate of the glyph classifiers. The number of splits used is one of two values; either 1 for stumps or 2 for the single node CART. The number of splits and minimum hit rate both fall into the domain of the boosting stage classifier. Boosting is the basic classifier for the HCC approach. A boosting classifier combines many weak classifiers in order to produce a powerful classifier (Freund and Schapire 1996). The number of splits defines which weak classifier will be used in the stage classifier. The Alef isolated character (ا) was used as an example in this experiment with a width of 10 and height of 23. Four different values were tried for the number of splits and minimum hit rate (two for each) in order to deduce the best values.

#### 4.5.2.1 Running the experiment

This experiment is an extension to the previous auxiliary experiment. It uses the positive results of the training of the width and height of a glyph classifier and applies a different number of splits and a minimum hit rate.

The number of splits defines which weak classifier will be used in the stage classifier. If the number of splits used is 1 (stump), this does not allow learning dependencies between features. A Classification And Regression Tree (CART) is obtained if the number of splits is more than 1, with the number of nodes smaller than the number of splits by one. A CART is a machine learning data structure which generates a prediction model based on the input data (Barros, Basgalupp et al. 2011) and allows dependencies between features (Lienhart, Kuranov et al. 2002). The default value of number of splits is 1. Here the number of splits was tested with values of 1 (stump) and 2 (a single node CART).

In order for the training process to move to the next stage, a minimum value of hit rate (TP) needs to be obtained (Viola and Jones 2001; Lienhart, Kuranov et al. 2002). As the minimum hit rate increases it slows down the training process. The default value is 0.995. Lienhart *et al.* (Lienhart, Kuranov et al. 2002) reported a value of 0.999 in their experimental work which proved remarkably slow so, the two final values used in this experiment were 0.995 and 0.997.

Alef isolated (ا) was used for this experiment as its data was already prepared and was tested with different values of widths and heights. The width and height values used were those that gave good results in the previous auxiliary experiment.

#### 4.5.2.2 Results and conclusions

The results of the experiment proved that, in the case of Arabic letters, using stumps in the splits is much better than using the single node CART (Table 4.11). This means that the Haar-like features of the Arabic characters do not improve with the dependencies between them. It also means that a single node CART may not be enough to build a good classifier and may need more nodes.

On the other hand, increasing the minimum hit rate improves the accuracy of the classifier, although it slows down the time of the training process. It is recommended to use higher minimum hit rate only in the case of glyphs that get a low recognition accuracy.

Table 4.11: Testing different values of number of splits and minimum hit rate

| | Trial 1 | Trial 2 | Trial 3 | Trial 4 |
|---|---|---|---|---|
| Number of splits | 1 | 2 | 1 | 2 |
| Minimum hit rate | 0.995 | 0.995 | 0.997 | 0.997 |
| True Positive (TP) | 64% | 52% | 72% | 65% |
| False Positive (FP) | 8% | 8% | 11% | 12% |

## 4.5.3 Test the boosting type and number of splits

This experiment was done in order to get the best values of the boosting type (algorithm) with the number of splits of the glyph classifiers. Three different boosting algorithms were tested: Gentle Ada-Boost (GAB), Discrete Ada-Boost (DAB) and Real Ada-Boost (RAB). Four different values were used for the number of splits. These four values are 1 for stumps, 2 for a single node CART, 3 for the double node CART and finally 4 for the triple node CART. The glyph Heh Mid (ـهـ) was used in the experiment with a training width and height of 18 and 33 respectively.

### *4.5.3.1 Running the experiment*

This experiment is an extension to the previous auxiliary experiment. It measures the relationship between the boosting type and the number of splits.

The number of splits defines the type of splitting either from stumps with no relation between Haar features if it is 1, or a tree with defined nodes and dependencies between features. The default value is 1.

The main reason for selecting the glyph Heh middle (ـهـ) in this experiment is that the glyph takes less time in the training process, and it was reported by Lienhart *et al.* (Lienhart, Kuranov et al. 2002) that DAB and RAB takes more time in training than GAB.

### *4.5.3.2 Results and conclusions*

The experiment showed that the GAB boosting type is the best algorithm for Arabic glyph recognition and is similar to face detection as reported by Lienhart *et al.*

(Lienhart, Kuranov et al. 2002). It not only gives the best results but also takes less training time. Table 4.12 shows that the stumps give a large number of false positive results when used with DAB or RAB. The best results are achieved when using the GAB boosting type with the double node CART. The false positive (FP) rate can be further reduced via, for example post-processing.

Table 4.12: Testing different number of splits with different boosting types

| Boosting Type | Gentle Ada-Boost (GAB) | | | | Discrete Ada-Boost (DAB) | | | | Real Ada-Boost (RAB) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| No. of Splits | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| True Positive (TP) | 94% | 94% | 97% | 93% | 85% | 95% | 95% | 93% | 92% | 92% | 93% | 93% |
| False Positive (FP) | 7% | 6% | 8% | 7% | 78% | 6% | 6% | 11% | 35% | 6% | 6% | 6% |

The results of the previous and current auxiliary experiments show that the Haar-like features of the Arabic characters achieve good results when using the stump and double node CART which can model second order dependencies and builds a good classifier.

## 4.6 The final experiments

The final experiment was performed in order to create the general classifier that will be used as a tool for Arabic printed character recognition. The previous experiments justified that the Haar Cascade Classifier (HCC) approach can be confidently used as a method for Arabic character recognition. The results of auxiliary experiments were used in order to enhance the performance of the training process and to improve the recognition accuracy. The final experiment uses all the glyphs in their four locations (isolated, start, middle and end) and deals only with naked glyphs.

### 4.6.1 Planning the experiment

The final experiment was planned to generate all classifiers for Arabic printed character recognition. The HCC approach produces a classifier for each single glyph. The following aspects were addressed in the experiment:

1. Naked glyphs were used in the experiment in order to reduce the number of classifiers used in the recognition stage. Reducing the number of classifiers

minimizes the recognition duration. Also, finding the place and number of dots above or below the recognized glyph is an easy process (Abdelazim 2006).

2. The data used in the final experiment is the same as used in the extended experiment but extended further to all the glyphs instead of the original 15 selected glyphs. The Real data is used but when it lacks positive images, computer data or sometimes computer and noise data is added. Using the real data for training and testing the classifiers is the most suitable as, in the end, the application is going to be used to detect real data and not computer generated data and was shown to be good (see Table 4.7).

3. The total number of naked Arabic letters, as indicated in Unicode are 18 (Unicode 1991-2006), but it was found in this research that adding Hamza (ء) and Lam Alef (لا) is essential (see section 2.2). Table 4.13 shows all the naked Arabic glyphs in their different locations that were used in the final experiment. The total number of glyphs is 61. This means that the total number of classifiers that will run on each document image will be 61.

Table 4.13: The Arabic naked glyphs as used in the final experiment

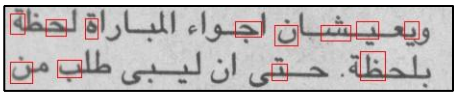
| Letter English name | Arabic Characters | Isolated glyphs | Start glyphs | Middle glyphs | End glyphs |
|---|---|---|---|---|---|
| ALEF | ا | ا | | | ـا |
| BEH | ب ت ث | ب ت ث | بـ تـ ثـ نـ يـ ئـ | ـبـ ـتـ ـثـ ـنـ ـيـ ـئـ | ـب ـت ـث |
| HAH | ج ح خ | ج ح خ | جـ حـ خـ | ـجـ ـحـ ـخـ | ـج ـح ـخ |
| DAL | د ذ | د ذ | | | ـد ـذ |
| REH | ر ز | ر ز | | | ـر ـز |
| SEEN | س ش | س ش | سـ شـ | ـسـ ـشـ | ـس ـش |
| SAD | ص ض | ص ض | صـ ضـ | ـصـ ـضـ | ـص ـض |
| TAH | ط ظ | ط ظ | طـ ظـ | ـطـ ـظـ | ـط ـظ |
| AIN | ع غ | ع غ | عـ غـ | ـعـ ـغـ | ـع ـغ |
| FEH | ف | ف | فـ قـ | ـفـ ـقـ | ـف |
| QAF | ق | ق | | | ـق |
| KAF | ك | ك | كـ | ـكـ | ـك |
| LAM | ل | ل | لـ | ـلـ | ـل |
| MEEM | م | م | مـ | ـمـ | ـم |
| NOON | ن | ن | | | ـن |
| HEH | ه | ه ة | هـ | ـهـ | ـه ـة |
| WAW | و ؤ | و | | | ـو ـؤ |
| YEH | ى ي ئ | ى | | | ـى ـى ـئ |
| HAMZA | ء | ء | | | |
| LAM ALEF | لا | لا | | | ـلا |

Planning for the experiment was done in the same way as the extended experiment. The original assumption regarding the pre-processing and character segmentation steps are now true and tested. It has been shown that when using the HCC approach these two stages can be completely skipped when being applied to Arabic character recognition.

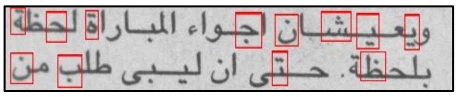
### 4.6.2 Data preparation

The data files in the final experiment were prepared identically as in the extended experiment (section 4.4.2). The same procedures were followed in order to generate the training and testing datasets for each glyph in each location. The same procedures were also followed in order to generate the positive and negative files. The difference being the larger number of glyphs prepared for the final experiment, where 61 glyphs were prepared instead of 15.

Selecting the rectangles that represented the glyphs in the positive files was a little bit tricky regarding the dots. Using the glyphs without the dots was tried first, as shown in Figure 4.10 (a), but this method gave bad results because it usually missed parts of the glyphs when trying to avoid the dots. Finally the dots were included in selecting the glyphs as in Figure 4.10 (b) which gave better results. Using the glyphs with dots in the training process let the glyph classifier train in the different shapes of that naked glyph.


ويعيشان اجواء المباراة لحظة
بلحظة. حتى ان ليبى طلب من

(a)
Glyphs defined without dots


ويعيشان اجواء المباراة لحظة
بلحظة. حتى ان ليبى طلب من

(b)
Glyphs defined with dots

Figure 4.10: Glyphs definition for the final experiment

The total number of original negative and positive images for all the naked glyphs in their four different locations is shown in Table 4.14. The total number of positive images is 6,657 while the total number of negative images is 10,941. The way numerous samples of positive and negative images were generated was the same that was used before in the extended experiment. The same programs and utilities were used for generating the samples as shown in section 4.4.2.2.

Table 4.14: Number of negative and positive images of glyphs in four locations

| Glyph Name | Isolated | | Start | | Middle | | End | |
|---|---|---|---|---|---|---|---|---|
| | Pos. | Neg. | Pos. | Neg. | Pos. | Neg. | Pos. | Neg. |
| ALEF | 198 | 25 | | | | | 207 | 16 |
| BEH | 178 | 186 | 203 | 20 | 190 | 33 | 91 | 132 |
| HAH | 52 | 500 | 133 | 90 | 133 | 90 | 47 | 317 |
| DAL | 182 | 182 | | | | | 170 | 53 |
| REH | 242 | 122 | | | | | 194 | 29 |
| SEEN | 24 | 340 | 101 | 122 | 132 | 91 | 39 | 325 |
| SAD | 26 | 338 | 40 | 183 | 84 | 139 | 25 | 339 |
| TAH | 9 | 355 | 61 | 303 | 68 | 155 | 34 | 330 |
| AIN | 23 | 341 | 146 | 77 | 131 | 92 | 40 | 183 |
| FEH | 29 | 335 | 178 | 45 | 141 | 82 | 34 | 330 |
| QAF | 34 | 330 | | | | | 19 | 345 |
| KAF | 21 | 343 | 88 | 135 | 90 | 133 | 29 | 335 |
| LAM | 92 | 272 | 203 | 20 | 137 | 86 | 68 | 155 |
| MEEM | 104 | 260 | 172 | 51 | 114 | 109 | 58 | 165 |
| NOON | 145 | 219 | | | | | 90 | 133 |
| HEH | 152 | 212 | 88 | 135 | 108 | 115 | 188 | 35 |
| WAW | 314 | 50 | | | | | 170 | 53 |
| YEH | 101 | 263 | | | | | 187 | 36 |
| HAMZA | 40 | 324 | | | | | | |
| LAM ALEF | 206 | 158 | | | | | 54 | 169 |

### 4.6.3 Training the final experiment classifiers

The training process of the final experiment was a very lengthy process compared to the previous extended experiment. It took a long time to prepare the files and folders for the training process. The training process itself took a long time to be accomplished which was around a year. This long time was due to the fact that the single training process takes two days in average; each glyph has 3 different trials on average and there are 61 glyphs, so this gives around a year of continues work on a single dedicated computer. Defining the parameters of the training process of the glyphs was a very important issue to prepare before the experiment. The parameters used were the width, height, number of splits, minimum hit rate and boosting type. These parameters were used following the results and conclusions of the auxiliary experiments. It is clear that the training sizes of a glyph in its different locations are usually different.

Table 4.15 and Table 4.16 show the glyphs in each of their four different locations, the trained width and height, the total number of negative images, total number of positive images and the total number of glyphs in the positive images. The total number of positive images for all glyphs is 106,324 while the total number of negative images is 168,241. The total number of all glyphs in all positive images is 181,874. The average number of glyphs in each positive image is 1.71. The average width of all glyphs is 18.9 while the average height is 24.9. The average number of positive images is 1743.0 and for negative 2758.0.

Table 4.15: Training information of glyphs in isolated and start location

| Glyph Name | Isolated location glyphs | | | | | Start location glyphs | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | W | H | Pos. Img | Neg. Img | Total Glyphs | W | H | Pos. Img | Neg. Img | Total Glyphs |
| ALEF | 10 | 23 | 1639 | 1775 | 9592 | | | | | |
| BEH | 23 | 22 | 2144 | 2976 | 3312 | 12 | 29 | 1683 | 1420 | 7678 |
| HAH | 22 | 28 | 1599 | 5500 | 1848 | 22 | 30 | 2100 | 2340 | 3507 |
| DAL | 14 | 20 | 2192 | 2912 | 2992 | | | | | |
| REH | 16 | 23 | 2002 | 2562 | 4004 | | | | | |
| SEEN | 31 | 25 | 1278 | 3740 | 1420 | 26 | 23 | 1596 | 2562 | 2163 |
| SAD | 34 | 27 | 1420 | 3718 | 1420 | 30 | 24 | 1680 | 2928 | 1960 |
| TAH | 26 | 30 | 497 | 3905 | 497 | 20 | 28 | 1656 | 3333 | 1800 |
| AIN | 17 | 24 | 1278 | 3751 | 1278 | 14 | 21 | 1760 | 2387 | 2624 |
| FEH | 26 | 26 | 1562 | 3685 | 1633 | 14 | 22 | 2144 | 2070 | 3968 |
| QAF | 27 | 40 | 1586 | 3630 | 1769 | | | | | |
| KAF | 23 | 27 | 1136 | 3773 | 1278 | 17 | 25 | 1716 | 2160 | 2002 |
| LAM | 17 | 30 | 1794 | 2992 | 2028 | 13 | 30 | 1683 | 1420 | 5907 |
| MEEM | 15 | 27 | 1638 | 2860 | 1848 | 14 | 22 | 2064 | 2091 | 3952 |
| NOON | 18 | 26 | 1744 | 2409 | 2240 | | | | | |
| HEH | 16 | 24 | 1824 | 2332 | 2368 | 15 | 17 | 1716 | 2160 | 2392 |
| WAW | 14 | 21 | 2596 | 2300 | 6193 | | | | | |
| YEH | 22 | 22 | 1596 | 2893 | 1785 | | | | | |
| HAMZA | 14 | 16 | 1680 | 3564 | 1904 | | | | | |
| LAM ALEF | 16 | 23 | 1705 | 2528 | 2662 | | | | | |

Table 4.16: Training information of glyphs in middle and end location

| Glyph Name | Middle location glyphs | | | | | End location glyphs | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | W | H | Pos. Img | Neg. Img | Total Glyphs | W | H | Pos. Img | Neg. Img | Total Glyphs |
| ALEF | | | | | | 10 | 22 | 1716 | 1136 | 7194 |
| BEH | 12 | 33 | 2288 | 2178 | 12512 | 25 | 30 | 1794 | 2772 | 2080 |
| HAH | 25 | 20 | 2100 | 2340 | 3129 | 20 | 32 | 1656 | 3487 | 1794 |
| DAL | | | | | | 14 | 20 | 2048 | 2173 | 3792 |
| REH | | | | | | 16 | 29 | 2336 | 2059 | 5632 |
| SEEN | 26 | 24 | 2079 | 2366 | 2961 | 27 | 26 | 1680 | 3575 | 1904 |
| SAD | 23 | 20 | 1638 | 2224 | 2288 | 31 | 24 | 1349 | 3729 | 1491 |
| TAH | 18 | 27 | 1581 | 2480 | 1984 | 20 | 24 | 1586 | 3630 | 1647 |
| AIN | 14 | 25 | 2079 | 2392 | 3045 | 18 | 31 | 1680 | 2928 | 1906 |
| FEH | 14 | 25 | 1696 | 2132 | 2752 | 20 | 15 | 1586 | 3630 | 1647 |
| QAF | | | | | | 20 | 28 | 1065 | 3795 | 1065 |
| KAF | 15 | 25 | 1768 | 2793 | 2418 | 20 | 20 | 1562 | 3685 | 1633 |
| LAM | 10 | 26 | 1648 | 2236 | 2544 | 17 | 25 | 1581 | 2480 | 1922 |
| MEEM | 15 | 20 | 1806 | 2289 | 2520 | 15 | 26 | 1584 | 2640 | 1872 |
| NOON | | | | | | 18 | 22 | 1768 | 2793 | 2522 |
| HEH | 18 | 33 | 1701 | 2415 | 2352 | 17 | 25 | 2256 | 2135 | 5184 |
| WAW | | | | | | 17 | 28 | 2048 | 2173 | 4064 |
| YEH | | | | | | 22 | 17 | 2256 | 2196 | 4192 |
| HAMZA | | | | | | | | | | |
| LAM ALEF | | | | | | 18 | 25 | 1681 | 2704 | 1804 |

### 4.6.4 Testing the final experiment classifiers

The testing process of the final experiment was separated into two different parts. The first used the *performance* utility offered in OpenCV. The second tested the HCC glyphs classifier against real commercial software in order to compare the results of the Arabic glyphs classifiers.

#### *4.6.4.1 Testing using the OpenCV performance utility*

This testing process of the final experiment used the testing utility offered by the OpenCV. The testing result values of True Positive (TP), False Negative (FN) and False Positive (FP) ratios were tracked in all tests. The testing process followed the same procedure that was followed previously in the extended experiment (see section 4.4.5).

Table 4.17 shows the total number of testing positive images and the total number of glyphs in the positive images for each of the glyphs in every location. The total number of positive images for all the glyphs is 34,543. The total number of all the glyphs in all the positive images is 59,498. The average number of glyphs in each positive image is 1.72; this value is almost the same as the training phase.

Table 4.17: The testing information for all the glyphs in all locations

| Glyph Name | Isolated location glyphs | | Start location glyphs | | Middle location glyphs | | End location glyphs | |
|---|---|---|---|---|---|---|---|---|
| | Pos. Img | Total Glyphs | Pos. Img | Total Glyphs | Pos. Img | Total Glyphs | Pos. Img | Total Glyphs |
| ALEF | 539 | 3168 | | | | | 561 | 2211 |
| BEH | 704 | 992 | 550 | 2376 | 752 | 4192 | 572 | 780 |
| HAH | 533 | 615 | 693 | 1239 | 693 | 966 | 506 | 284 |
| DAL | 720 | 1120 | | | | | 672 | 1328 |
| REH | 660 | 1331 | | | | | 768 | 1648 |
| SEEN | 426 | 497 | 525 | 987 | 693 | 1155 | 504 | 504 |
| SAD | 426 | 426 | 560 | 560 | 546 | 624 | 426 | 426 |
| TAH | 142 | 142 | 540 | 648 | 527 | 589 | 488 | 549 |
| AIN | 355 | 426 | 576 | 944 | 672 | 861 | 560 | 560 |
| FEH | 497 | 497 | 704 | 1248 | 560 | 896 | 488 | 549 |
| QAF | 488 | 488 | | | | | 284 | 284 |
| KAF | 355 | 355 | 572 | 676 | 572 | 650 | 497 | 497 |
| LAM | 598 | 754 | 550 | 1980 | 544 | 784 | 527 | 744 |
| MEEM | 546 | 546 | 688 | 1520 | 588 | 882 | 504 | 720 |
| NOON | 576 | 720 | | | | | 572 | 754 |
| HEH | 608 | 832 | 572 | 806 | 567 | 903 | 752 | 1632 |
| WAW | 858 | 2068 | | | | | 672 | 1280 |
| YEH | 525 | 588 | | | | | 736 | 1536 |
| HAMZA | 560 | 784 | | | | | | |
| LAM ALEF | 561 | 803 | | | | | 533 | 574 |

#### *4.6.4.2 Testing using a commercial Arabic OCR application*

The objective of this testing process is to compare the HCC approach against Arabic OCR commercial software. This test gives realistic measures of the performance of the HCC approach when comparing it with commercial software. The testing examines all the glyphs classifiers of the HCC approach against the commercial software. Well known commercial software was used in order to perform this test; Readiris Pro 10 (IRIS 2004) which was used above in section 3.3.

A small sample of Arabic paragraph documents that represents a variety of document types that occur in most real life cases were selected to apply this testing process. The sample includes 37 Arabic documents which contain 568 words and 2,798 letters. This sample was not used in the training and testing of the final experiment to produce the final 61 classifiers.

The Levenshtein distance algorithm explained in section 2.7.3.2 and used previously in section 3.3 was used to calculate the commercial software accuracy. The accuracy of the HCC approach was calculated after running the *performance* tool offered by OpenCV to detect the glyphs using the 61 generated classifiers. A manual check was applied to calculate the accuracy of recognition of all the classifiers for all the documents.

### 4.6.5 Results and conclusions

The final experiment results showed a real success of the HCC approach in recognizing the Arabic documents. The results of the auxiliary experiments led to some changes in the training parameters which enhanced the results of the final experiment. High accuracy was produced in the case of using the OpenCV testing utility as well as when compared with commercial software. The results of the experiment were separated into two different parts, one using the OpenCV *performance* utility and the other when compared to commercial software.

#### *4.6.5.1 Results of using the OpenCV performance utility*

Testing the final experiment using the *performance* utility led to a comparison between four different locations of glyphs in the Arabic document images. The four types give different accuracy for each glyph. Table 4.18 shows the TP, FN and FP values for each glyph in its four different locations. The last row shows the average percentage of accuracy of all the Arabic glyphs.

Table 4.18: Testing results as a percentage for all glyphs in different locations

| Glyph Name | Isolated location glyphs | | | Start location glyphs | | | Middle location glyphs | | | End location glyphs | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP | FN | FP | TP | FN | FP | TP | FN | FP | TP | FN | FP |
| ALEF | 94% | 6% | 11% | | | | | | | 93% | 7% | 6% |
| BEH | 84% | 16% | 14% | 81% | 19% | 17% | 74% | 26% | 4% | 93% | 7% | 1% |
| HAH | 76% | 24% | 0% | 94% | 6% | 2% | 86% | 14% | 18% | 100% | 0% | 0% |
| DAL | 85% | 15% | 5% | | | | | | | 93% | 7% | 2% |
| REH | 73% | 27% | 8% | | | | | | | 74% | 26% | 3% |
| SEEN | 85% | 15% | 1% | 92% | 8% | 4% | 93% | 7% | 6% | 82% | 18% | 12% |
| SAD | 83% | 17% | 28% | 97% | 3% | 7% | 89% | 11% | 13% | 88% | 12% | 4% |
| TAH | 95% | 5% | 4% | 82% | 18% | 8% | 93% | 7% | 24% | 88% | 12% | 9% |
| AIN | 100% | 0% | 2% | 92% | 8% | 0% | 92% | 8% | 2% | 99% | 1% | 0% |
| FEH | 92% | 8% | 29% | 94% | 6% | 9% | 83% | 17% | 9% | 92% | 8% | 16% |
| QAF | 75% | 25% | 0% | | | | | | | 100% | 0% | 1% |
| KAF | 97% | 3% | 4% | 95% | 5% | 3% | 93% | 7% | 6% | 93% | 7% | 8% |
| LAM | 94% | 6% | 2% | 85% | 15% | 9% | 92% | 8% | 12% | 92% | 8% | 0% |
| MEEM | 92% | 8% | 5% | 91% | 9% | 13% | 97% | 3% | 7% | 90% | 10% | 4% |
| NOON | 89% | 11% | 9% | | | | | | | 97% | 3% | 3% |
| HEH | 86% | 14% | 13% | 94% | 6% | 1% | 92% | 8% | 0% | 86% | 14% | 3% |
| WAW | 83% | 17% | 16% | | | | | | | 98% | 2% | 4% |
| YEH | 97% | 3% | 5% | | | | | | | 98% | 2% | 3% |
| HAMZA | 78% | 22% | 4% | | | | | | | | | |
| LAM ALEF | 98% | 2% | 3% | | | | | | | 96% | 4% | 1% |
| Average | 88% | 12% | 8% | 91% | 9% | 7% | 89% | 11% | 9% | 92% | 8% | 4% |

Table 4.19: Testing results in numbers for all glyphs in different locations

| Glyph Name | Isolated location glyphs | | | Start location glyphs | | | Middle location glyphs | | | End location glyphs | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TP | FN | FP | TP | FN | FP | TP | FN | FP | TP | FN | FP |
| ALEF | 2969 | 199 | 347 | | | | | | | 2053 | 158 | 126 |
| BEH | 836 | 156 | 140 | 1919 | 457 | 401 | 3084 | 1108 | 159 | 726 | 54 | 4 |
| HAH | 465 | 150 | 3 | 1167 | 72 | 27 | 827 | 139 | 174 | 284 | 0 | 0 |
| DAL | 949 | 171 | 59 | | | | | | | 1240 | 88 | 29 |
| REH | 975 | 356 | 112 | | | | | | | 1223 | 425 | 54 |
| SEEN | 423 | 74 | 6 | 910 | 77 | 43 | 1070 | 85 | 70 | 415 | 89 | 58 |
| SAD | 354 | 72 | 121 | 542 | 18 | 41 | 558 | 66 | 84 | 375 | 51 | 17 |
| TAH | 135 | 7 | 5 | 531 | 117 | 49 | 545 | 44 | 144 | 483 | 66 | 49 |
| AIN | 426 | 0 | 7 | 867 | 77 | 3 | 794 | 67 | 15 | 553 | 7 | 1 |
| FEH | 459 | 38 | 146 | 1177 | 71 | 114 | 744 | 152 | 84 | 504 | 45 | 86 |
| QAF | 364 | 124 | 0 | | | | | | | 284 | 0 | 4 |
| KAF | 345 | 10 | 15 | 640 | 36 | 20 | 603 | 47 | 37 | 460 | 37 | 42 |
| LAM | 709 | 45 | 13 | 1687 | 293 | 177 | 720 | 64 | 92 | 684 | 60 | 0 |
| MEEM | 500 | 46 | 29 | 1389 | 131 | 195 | 856 | 26 | 61 | 651 | 69 | 26 |
| NOON | 644 | 76 | 65 | | | | | | | 733 | 21 | 24 |
| HEH | 712 | 120 | 108 | 760 | 46 | 9 | 834 | 69 | 4 | 1408 | 224 | 57 |
| WAW | 1725 | 343 | 327 | | | | | | | 1249 | 31 | 46 |
| YEH | 570 | 18 | 29 | | | | | | | 1504 | 32 | 49 |
| HAMZA | 614 | 170 | 33 | | | | | | | | | |
| LAM ALEF | 785 | 18 | 22 | | | | | | | 549 | 25 | 3 |
| Total | 14959 | 2193 | 1587 | 11589 | 1395 | 1079 | 10635 | 1867 | 924 | 15378 | 1482 | 675 |
| Accuracy | 87% | 13% | 9% | 89% | 11% | 8% | 85% | 15% | 7% | 91% | 9% | 4% |

Table 4.19 shows the total number of TP, FN and FP objects for each glyph for each of the four different locations for Arabic glyphs. The last row shows the average percentage for all glyphs relative to the total number in each dataset.

#### *4.6.5.2 Results of testing with the commercial OCR application*

Comparing the HCC approach to the commercial software showed that the HCC approach achieved a slightly better accuracy than that achieved by the commercial software. The accuracy of the HCC approach in this test achieved 87% while the commercial software achieved 85%.

The conclusions of the auxiliary experiments led to a change in the training and testing parameters used in the final experiment. This was the reason of a slight enhancement in the results of the final experiment when compared to the extended experiment.

The results of the final experiment show the following:

1. The HCC approach glyph recognition accuracy is 87%. The accuracy achieved can be considered as a high level of recognition at this stage and proves the validity of the approach for the Arabic character recognition.

2. The glyphs with a small number of positive or negative samples return unreasonable results, for instance most of the isolated location glyphs. This justifies the use of Real and Computer data or sometimes Noise data for some glyphs.

3. The glyphs with a balanced number of positive and negative samples usually return good results, for example Beh end (ـب), Seen start (سـ) and Kaf start (كـ).

4. The letters with complicated visual features usually get better recognition accuracy, for instance Hah (ح), Sad (ص), Heh (هـ) and Lam Alef (لا).

5. Using naked glyphs did not reduce the recognition rate but actually improved it and reduced the number of classifiers that were needed.

6. It is now very clear that the HCC approach can recognize printed Arabic glyphs without segmenting them.

## 4.7 Summary

This chapter presented experimental work using a cascade of boosted classifiers based on Haar-like features for Arabic printed character recognition. This experiment made use of the multi-modal Arabic corpus generated and explained above in Chapter 3. The current chapter ended up with 61 classifiers. These classifiers are used to generate a general Arabic classifier as explained in Chapter 5. The issues addressed and conclusions reached in this chapter are:

1. The main research assumption of using the HCC approach with printed Arabic characters and alter it to avoid some steps of the pre-processing and recognition stages and the character segmentation phase.

2. How the HCC approach is tackled and applied in order to solve the problem of printed Arabic character recognition.

3. A preliminary pilot experiment to check the validity of the HCC approach with printed Arabic characters.

4. An extended experiment, justifying the use of the HCC approach with printed Arabic characters.

5. Three auxiliary experiments are applied to enhance the use of the parameters of the training process which led to the enhancement of the recognition accuracy.

6. The final experiment is applied on the Arabic naked glyphs in their locations. It generated classifiers for every Arabic naked glyph.

# Chapter 5.

# HCC tokens recognition

This chapter explains the generation of an Arabic naked tokens recognizer and tests its recognition accuracy. The recognizer uses the 61 naked glyphs classifiers that were previously produced in Chapter 4 in order to generate the single classifier. In addition to this, the training and testing results of the classifiers will be used. The real image dataset introduced in section 3.2.2 was used to prepare a new tokens testing dataset for this recognizer. The work described in this chapter will output the recognized tokens of the input document image using the single classifier.

The purpose of this chapter is to show that the HCC approach can lead to a full Arabic recognition system. For that reason, the algorithms followed here are simple in order to show the credibility of the HCC approach as an OCR approach. It was clear from the results of the previous chapter that the HCC approach detects glyphs with an accuracy of 87% which is slightly better than the commercial software 85%, so it is evident that the glyphs are well detected.

## 5.1 Introduction

This chapter reports on an Arabic naked tokens recognizer based on the 61 classifiers. The classifiers in Chapter 4 were generated based on the HCC approach. The generated recognizer in this chapter uses solely the visual features of the glyphs without acquiring statistical information to this stage.

Figure 5.1 shows the general flowchart of the HCC tokens recognizer application. Each of the main components on the figure includes the provided sections in this thesis.

### 5.1.1 HCC tokens recognizer overview

New tokens testing dataset was prepared to test the HCC token recognizer. The testing dataset is used in order to test the recognizer against the commercial software that is used in all the phases of this research. The testing procedures test three phases; one for the recognizer, after applying the confusion matrix (Tanner, Muñoz et al. 2009) and after applying post-processing in Chapter 6.

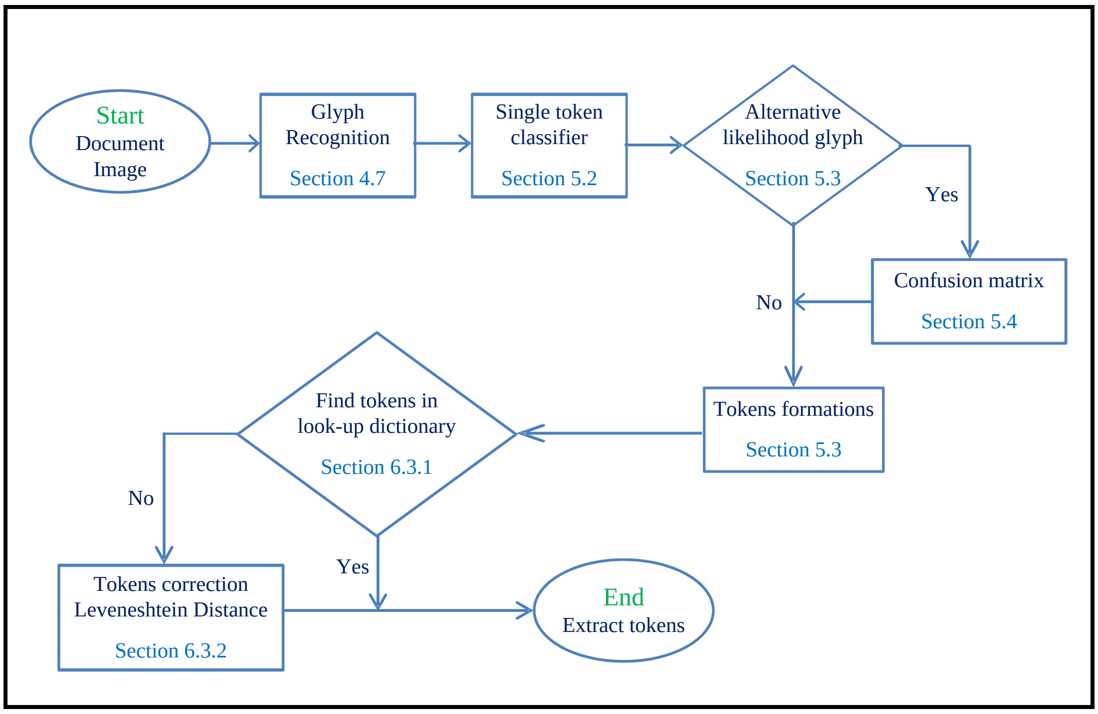


Figure 5.1: Flowchart showing HCC tokens recognition process

A single tokens classifier is generated by loading the classifiers with their parameters. The 61 classifiers are then run over the document image in order to generate the list of detected glyphs. Statistical analysis information of the document and the detected glyphs are then generated in order to reject wrongly detected glyphs (see Figure 5.1). The “baselines” of the words in the document are defined using the centre Y-axis of the detected glyphs within the horizontal histogram of the document.

A tokens recognizer is generated in order to parse the detected glyphs to text. The detected glyphs are sorted vertically into text lines, then horizontally in Arabic order (from right to left). The glyphs are then analyzed and arranged in order to generate Arabic tokens. The detected glyphs information is kept in a list and a file. The generated recognizer is tested using the tokens testing dataset against the previously used commercial software.

A confusion matrix for the 61 classifiers is generated. For each classifier, the most likelihood alternative seven glyphs are stored with the original one. A confusion matrix is used when a glyph is detected by more than one classifier (see Figure 5.1). It runs over all detected glyphs results from the recognizer and keeps the list of the best seven glyphs for each detected one. The new list of detected glyphs, after running the confusion matrix, is also kept in a list and a file. The modified extracted tokens after applying the confusion matrix are tested against the tokens generated from the recognizer.

The post processing stage in Chapter 6 is responsible for improving the detected glyphs by using the statistical information which was collected and arranged in the MMAC defined in Chapter 3. The statistical algorithms and techniques were used to enhance the recognitions rate of the tokens (see Figure 5.1). In this chapter, the missed glyphs are replaced by a (؟) question mark to allow the post processing stage to suggest the best glyph in place of the one that was missed.

### 5.1.2 Preparing the tokens testing dataset

A new token testing dataset was prepared in order to test the HCC application. The procedures followed in generating MMAC (see section 3.2) were followed in order to prepare this dataset. The contents of this dataset were not used previously in the final experiment of section 4.6 which led to the generation of the 61 classifiers.

The main reason of generating the tokens testing dataset is to test the remaining three steps of the HCC application. The tokens testing dataset tests the output of the HCC recognizer (see section 5.3) which is the first step that output recognized tokens. The output after applying the confusion matrix is also tested (see section 5.4). Finally it is used to test the output of the post-processing step (see Chapter 6). In all the testing steps, the output is compared with the commercial software and with the previous steps to get the improvement in the application recognition rate using this step.

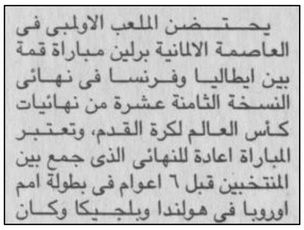
The main source of the tokens classifier testing dataset was the full page real image dataset that was generated in section 3.2.2. The contents of the testing dataset used a variety of document types, including office documents, faxes, books, magazines and newspapers. The documents were ground truthed as text files and were reviewed by an independent group of people. The ground truth files are presented in the form of words, naked words, PAWs and naked PAWs.

The tokens testing dataset was generated in order to test the approach presented in this thesis against well-known commercial software that was used previously in section 3.3. The documents in this dataset were extracted from full page images, which contained text paragraphs of the full pages, with no images or graphics inside. It was prepared in order to test character recognition as page layout analysis is not included in the testing requirements and included 23 different document image files with 1,053 words inside them.

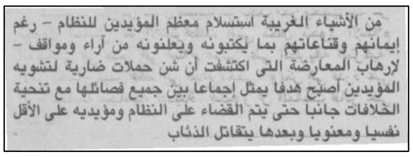
Table 5.1 shows the contents of the tokens testing dataset. It shows the type of documents and number of text lines, words, PAWs and characters of each one. The documents include a variety of document quality from good to poor. Figure 5.2 shows different samples of the tokens testing dataset which indicates the variety of document quality.

Table 5.1: Tokens testing dataset files description

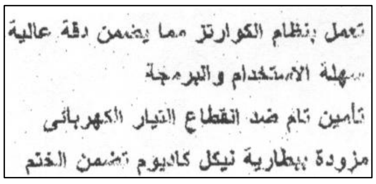
| Document image file | Document Type | No. Of Text Lines | No. Of Words | No. Of PAWs | No. Of Characters |
|---|---|---|---|---|---|
| FinalTest001.bmp | Book | 5 | 34 | 78 | 216 |
| FinalTest002.bmp | Book | 5 | 41 | 89 | 254 |
| FinalTest003.bmp | Office Document | 4 | 21 | 53 | 124 |
| FinalTest004.bmp | Office Document | 4 | 24 | 58 | 146 |
| FinalTest005.bmp | Fax | 3 | 34 | 68 | 189 |
| FinalTest006.bmp | Fax | 3 | 34 | 54 | 161 |
| FinalTest007.bmp | Fax | 4 | 18 | 43 | 107 |
| FinalTest008.bmp | Fax | 4 | 22 | 54 | 136 |
| FinalTest009.bmp | Magazine | 7 | 68 | 141 | 361 |
| FinalTest010.bmp | Magazine | 7 | 63 | 140 | 357 |
| FinalTest011.bmp | Magazine | 6 | 38 | 79 | 197 |
| FinalTest012.bmp | Magazine | 6 | 29 | 70 | 175 |
| FinalTest013.bmp | Magazine | 7 | 68 | 121 | 331 |
| FinalTest014.bmp | Magazine | 8 | 75 | 147 | 386 |
| FinalTest015.bmp | Magazine | 10 | 79 | 179 | 458 |
| FinalTest016.bmp | Magazine | 4 | 23 | 62 | 139 |
| FinalTest017.bmp | Magazine | 7 | 50 | 108 | 290 |
| FinalTest018.bmp | Magazine | 7 | 69 | 132 | 360 |
| FinalTest019.bmp | Magazine | 6 | 56 | 118 | 320 |
| FinalTest020.bmp | News paper | 8 | 58 | 120 | 295 |
| FinalTest021.bmp | News paper | 9 | 59 | 136 | 329 |
| FinalTest022.bmp | News paper | 8 | 43 | 97 | 250 |
| FinalTest023.bmp | News paper | 9 | 47 | 104 | 274 |
| Total |  | 141 | 1053 | 2251 | 5855 |



يحتضن الملعب الاولمبى فى
العاصمة الالمانية برلين مباراة قمة
بين ايطاليا وفرنسا فى نهائى
النسخة الثامنة عشرة من نهائيات
كأس العالم لكرة القدم، وتعتبر
المباراة اعادة للنهائى الذى جمع بين
المنتخبين قبل ٦ اعوام فى بطولة امم
اوروبا فى هولندا وبلجيكا وكان



من الأشياء الغريبة استسلام معظم المؤيدين للنظام – رغم
إيمانهم وقناعاتهم بما يكتبونه ويعلنونه من آراء ومواقف –
لإرهاب المعارضة التى اكتشفت أن شن حملات ضارية لتشويه
المؤيدين أصبح هدفا يمثل إجماعا بين جميع فصائلها مع تنحية
الخلافات جانبا حتى يتم القضاء على النظام ومؤيديه على الأقل
نفسيا ومعنويا وبعدها يتقاتل الذئاب



تعمل بنظام الكوارتز مما يضمن دقة عالية
سهلة الاستخدام والبرمجة
تأمين تام ضد انقطاع التيار الكهربائى
مزودة ببطارية نيكل كاديوم تضمن الختم

Figure 5.2: Different samples of Tokens testing dataset

### 5.1.3 Document statistical information

Keeping statistical information regarding the detected document is important in order to enhance the recognition accuracy of the document. The statistical information includes information about the document image, text lines, overall detected glyphs and each detected glyph.

The whole document image provides information about the properties of the document which includes information about the image quality and image contents. The width and height of the document is used in all the detection processes. The grey scale value of each pixel and each row of pixels is also very important in order to gain information regarding each glyph and text lines as shown in section 5.3.1. Average grey scaling of the document enables the rejection of the empty glyphs as shown in section 5.2.3 and gives information about the image; if it is dark or bright.

The average grey scale of the baselines gives information about the brightness of the glyphs in the document and is mainly used for the parsing of the detected glyphs as shown in section 5.3.2.

The overall detected glyphs information is important for rejecting wrongly detected glyphs and also tells us about the averages of the glyphs which are considered a good comparing base for each detected glyph. The average width, height and area helps in detecting text lines as shown in section 5.3.1, rejecting big detected glyphs as shown in section 5.2.3, and in rejecting empty detected glyphs as shown in section 5.2.3. The average grey scale value is used in order to parse the detected glyphs and to discover the relationship between every contiguous glyph as shown in section 5.3.2.

Each detected glyph holds information about the width, height and area of the glyph which helps to reject wrongly detected glyphs as shown in section 5.2.3. The average grey scale of the detected glyph allows for the rejection of wrongly, empty, detected glyphs as shown in section 5.2.3.

## 5.2 The single token classifier

This part of the research is responsible for extracting a list of detected glyphs and the important information from the document image.

### 5.2.1 Loading the classifiers list

The classifiers list includes information of the 61 classifiers generated previously in section 4.6 and is used in the testing process of each classifier (see section 4.6.4). This information is important to be passed as parameters to each classifier when run over the document image. The classifiers list is kept in a file for further enhancements through reloading the list at any time with new information. Table 5.2 shows the classifiers list that is used in the research for the 61 classifiers. The associated information for the classifiers is:

1. The classifier ID; this is used in order to know which classifier to associate with the detected glyph and is used in further processes.

2. The XML classifier file name is used to load the classifier file. This file is generated from the training process developed in section 4.6.3.

3. The width and height which were used before in the training and testing processes. See section 4.6.3 and 4.6.4. The testing parameters that give better detection accuracy during the previous testing process showed in section 4.6.4.2. These parameters are scale factor and minimum neighbour.

4. The True Positive (TP) percentage which was achieved when applying the classifier in the testing process as explained in section 4.6.4.2.

5. Classifier Average Area defines the average area of the classifier relative to the overall classifiers average area. This value will be used to check whether the detected glyph is correct or not based on detected area.

Table 5.2: Classifiers list of the HCC approach

| Classifier ID | Classifier file | Scale Factor | Minimum Neighbour | Width | Height | TP | Classifier Average Area |
|---|---|---|---|---|---|---|---|
| 1 | xml/_01AlefISO.xml | 1.02 | 3 | 10 | 23 | 0.937 | 0.478 |
| 2 | xml/_02AlefEND.xml | 1.01 | 2 | 10 | 22 | 0.929 | 0.457 |
| 3 | xml/_03BehISO.xml | 1.01 | 1 | 23 | 22 | 0.843 | 1.052 |
| 4 | xml/_04BehSTR.xml | 1.01 | 1 | 12 | 29 | 0.808 | 0.724 |
| 5 | xml/_05BehMID.xml | 1.01 | 1 | 16 | 36 | 0.736 | 1.198 |
| 6 | xml/_06BehEND.xml | 1.01 | 1 | 25 | 30 | 0.931 | 1.560 |
| 7 | xml/_07HahISO.xml | 1.02 | 2 | 22 | 28 | 0.756 | 1.281 |
| 8 | xml/_08HahSTR.xml | 1.01 | 1 | 22 | 30 | 0.942 | 1.372 |
| 9 | xml/_09HahMID.xml | 1.01 | 1 | 24 | 40 | 0.856 | 1.996 |
| 10 | xml/_10HahEND.xml | 1.02 | 3 | 20 | 32 | 1.000 | 1.331 |
| 11 | xml/_11DalISO.xml | 1.01 | 1 | 14 | 20 | 0.847 | 0.582 |
| 12 | xml/_12DalEND.xml | 1.01 | 1 | 14 | 23 | 0.934 | 0.670 |
| 13 | xml/_13RehISO.xml | 1.01 | 1 | 16 | 23 | 0.733 | 0.765 |
| 14 | xml/_14RehEND.xml | 1.01 | 1 | 16 | 29 | 0.742 | 0.965 |
| 15 | xml/_15SeenISO.xml | 1.01 | 1 | 31 | 25 | 0.851 | 1.612 |
| 16 | xml/_16SeenSTR.xml | 1.01 | 1 | 26 | 23 | 0.922 | 1.243 |
| 17 | xml/_17SeenMID.xml | 1.01 | 2 | 26 | 24 | 0.926 | 1.298 |
| 18 | xml/_18SeenEND.xml | 1.01 | 1 | 27 | 26 | 0.823 | 1.460 |
| 19 | xml/_19SadISO.xml | 1.01 | 1 | 34 | 27 | 0.831 | 1.909 |
| 20 | xml/_20SadSTR.xml | 1.01 | 2 | 30 | 24 | 0.968 | 1.497 |
| 21 | xml/_21SadMID.xml | 1.01 | 5 | 23 | 20 | 0.894 | 0.957 |
| 22 | xml/_22SadEND.xml | 1.01 | 1 | 31 | 24 | 0.880 | 1.547 |
| 23 | xml/_23TahISO.xml | 1.01 | 1 | 26 | 30 | 0.951 | 1.622 |
| 24 | xml/_24TahSTR.xml | 1.01 | 1 | 20 | 28 | 0.819 | 1.164 |
| 25 | xml/_25TahMID.xml | 1.01 | 1 | 18 | 27 | 0.925 | 1.011 |
| 26 | xml/_26TahEND.xml | 1.01 | 2 | 20 | 24 | 0.880 | 0.998 |
| 27 | xml/_27AinISO.xml | 1.01 | 1 | 17 | 24 | 1.000 | 0.848 |
| 28 | xml/_28AinSTR.xml | 1.01 | 1 | 14 | 21 | 0.918 | 0.611 |
| 29 | xml/_29AinMID.xml | 1.01 | 1 | 14 | 25 | 0.922 | 0.728 |
| 30 | xml/_30AinEND.xml | 1.02 | 3 | 18 | 31 | 0.988 | 1.160 |
| 31 | xml/_31FehISO.xml | 1.01 | 1 | 26 | 26 | 0.924 | 1.406 |
| 32 | xml/_32FehSTR.xml | 1.01 | 1 | 14 | 22 | 0.943 | 0.640 |
| 33 | xml/_33FehMID.xml | 1.01 | 1 | 14 | 25 | 0.830 | 0.728 |
| 34 | xml/_34FehEND.xml | 1.02 | 1 | 20 | 15 | 0.918 | 0.624 |
| 35 | xml/_35QafISO.xml | 1.01 | 1 | 27 | 40 | 0.746 | 2.246 |
| 36 | xml/_36QafEND.xml | 1.01 | 3 | 20 | 28 | 1.000 | 1.164 |
| 37 | xml/_37KafISO.xml | 1.01 | 1 | 23 | 27 | 0.972 | 1.291 |
| 38 | xml/_38KafSTR.xml | 1.01 | 4 | 17 | 25 | 0.947 | 0.884 |
| 39 | xml/_39KafMID.xml | 1.01 | 1 | 15 | 25 | 0.928 | 0.780 |
| 40 | xml/_40KafEND.xml | 1.01 | 2 | 20 | 20 | 0.926 | 0.832 |
| 41 | xml/_41LamISO.xml | 1.01 | 2 | 17 | 30 | 0.940 | 1.060 |
| 42 | xml/_42LamSTR.xml | 1.02 | 4 | 13 | 30 | 0.852 | 0.811 |
| 43 | xml/_43LamMID.xml | 1.01 | 1 | 10 | 26 | 0.918 | 0.541 |
| 44 | xml/_44LamEND.xml | 1.01 | 1 | 17 | 25 | 0.919 | 0.884 |
| 45 | xml/_45MeemISO.xml | 1.01 | 2 | 15 | 27 | 0.916 | 0.842 |
| 46 | xml/_46MeemSTR.xml | 1.01 | 1 | 14 | 22 | 0.914 | 0.640 |
| 47 | xml/_47MeemMID.xml | 1.01 | 1 | 15 | 20 | 0.971 | 0.624 |
| 48 | xml/_48MeemEND.xml | 1.01 | 1 | 15 | 26 | 0.904 | 0.811 |
| 49 | xml/_49NoonISO.xml | 1.01 | 2 | 18 | 26 | 0.894 | 0.973 |
| 50 | xml/_50NoonEND.xml | 1.01 | 3 | 18 | 22 | 0.972 | 0.823 |
| 51 | xml/_51HehISO.xml | 1.01 | 1 | 16 | 24 | 0.856 | 0.798 |
| 52 | xml/_52HehSTR.xml | 1.02 | 1 | 15 | 17 | 0.943 | 0.530 |
| 53 | xml/_53HehMID.xml | 1.02 | 1 | 18 | 33 | 0.924 | 1.235 |
| 54 | xml/_54HehEND.xml | 1.01 | 1 | 17 | 25 | 0.863 | 0.884 |
| 55 | xml/_55WawISO.xml | 1.01 | 1 | 14 | 21 | 0.834 | 0.611 |
| 56 | xml/_56WawEND.xml | 1.01 | 1 | 17 | 28 | 0.976 | 0.990 |
| 57 | xml/_57YehISO.xml | 1.01 | 1 | 22 | 22 | 0.969 | 1.006 |
| 58 | xml/_58YehEND.xml | 1.01 | 2 | 22 | 17 | 0.979 | 0.778 |
| 59 | xml/_59HamzaISO.xml | 1.01 | 1 | 14 | 16 | 0.783 | 0.466 |
| 60 | xml/_60LamAlefISO.xml | 1.01 | 2 | 16 | 23 | 0.978 | 0.765 |
| 61 | xml/_61LamAlefEND.xml | 1.02 | 3 | 18 | 25 | 0.956 | 0.936 |
| Average | | 1.011 | 1.541 | 18.951 | 25.377 | 0.900 | 1.011 |

### 5.2.2 Running the classifiers

This section of the research is responsible for running all the classifiers that were generated from the training process in section 4.6.3 over the document image and collecting the detected glyphs. The appropriate parameters loaded in the previous section 5.2.1 are associated with each classifier when run over the document.

Each classifier returns a list of the detected glyphs in the form of rectangles. Figure 5.3 shows a document image after running ten classifiers over it. Each classifier's rectangle is in a different colour. Only ten classifiers were used in Figure 5.3 in order to show an example of the output, as figure showing all 61 classifiers will be cluttered and difficult to understand.

The following were observed after running the classifiers over the document images:

1. The classifier usually detects only one glyph at a time rather than a bigger area that might include more than one glyph. Some classifiers of a larger size can sometimes detect more than one glyph, for example; Tah start (طـ) sometimes detects Ain start and Lam middle (علـ). This issue is considered when applying the confusion matrix (see section 5.4).

2. Some classifiers from different letters detect one another, for example; Seen start (سـ) and Sad start (صـ).

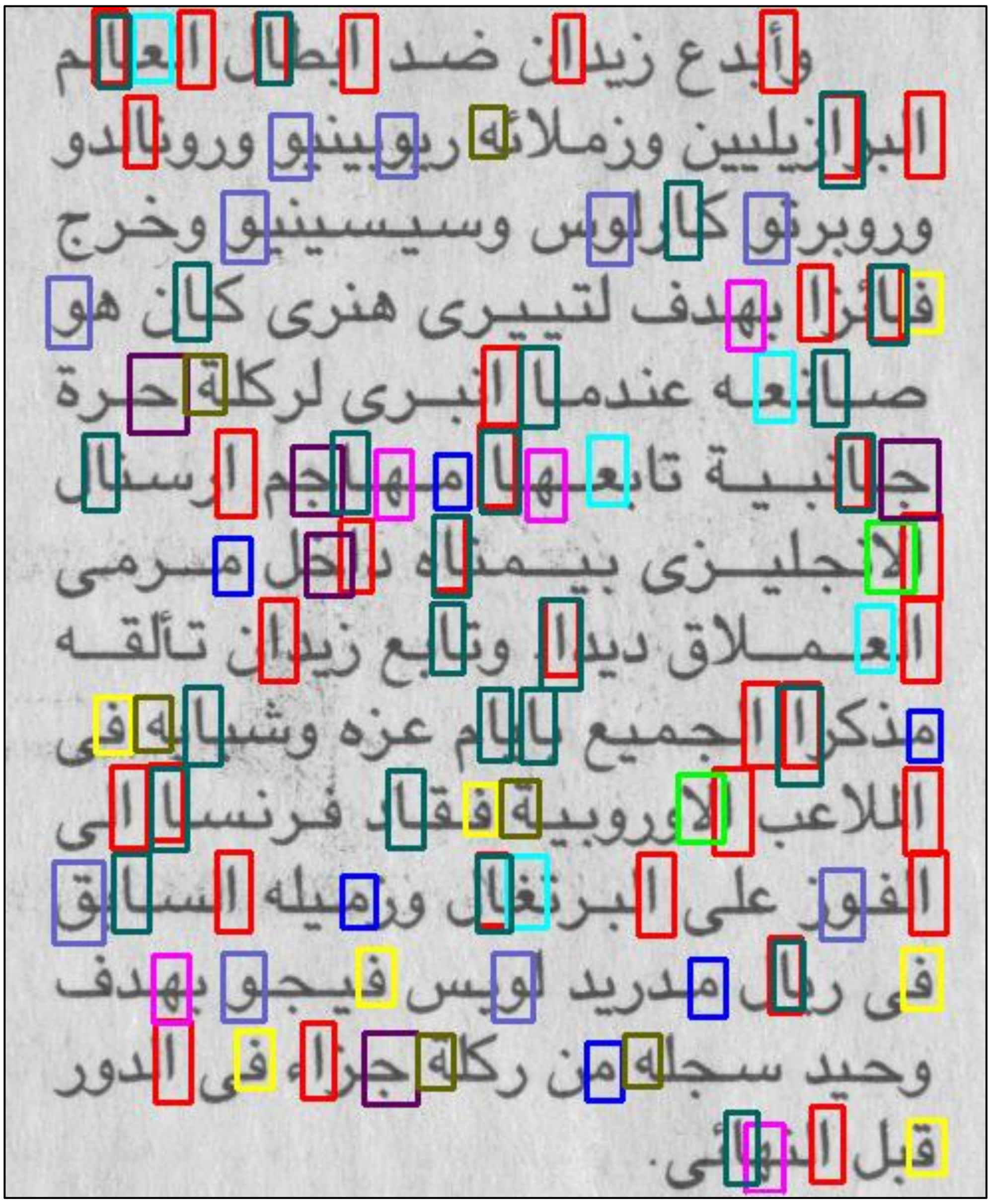
3. The extension letter or Tatweel is usually detected by mistake and must be given special treatment in an OCR application. This issue is considered during parsing process (see section 5.3.2).

4. The same classifier sometimes, by mistake, detects the same glyph again. It is ignored in order to be sure that the classifier always detects any glyph just once.

5. Sometimes more than one classifier detects the same glyph; most of these cases are for classifiers found to be from the same letter but different locations like Alef end (ـا) and Alef isolated (ا). This issue is considered when applying the confusion matrix (see section 5.4).

6. The space between words is clear in most cases and can be detected, although the space between PAWs is not that clear.

7. Applying the text lines to the detected glyphs is clear and easy, as there are no overlaps between lines.



وأبدع زيدان ضد أبطال العالم
البرازيليين وزملائه رونالدينيو ورونالدو
وروبرتو كارلوس وسيسينيو وخرج
فائزاً بهدف لتييري هنري كان هو
صانعه عندما انبرى لركلة حرة
جانبية تابعها مهاجم ارسنال
الانجليزي بيمناه داخل مرمى
العملاق ديدا وتابع زيدان تألقه
مذكراً الجميع بأيام عزه وشبابه في
الملاعب الاوروبية فقاد فرنسا الى
الفوز على البرتغال وزميله السابق
في ريال مدريد لويس فيجو بهدف
وحيد سجله من ركلة جزاء في الدور
قبل النهائي.

Figure 5.3: A document image after running ten classifiers over it

### 5.2.3 Keeping a list of glyphs

The detected glyphs information from all the classifiers that were run over the document images in the previous section (5.2.2) are kept in a list and a file. The following information is kept for each detected glyph:

1. The glyph ID, this is used for further identification of the glyph in the other processes.

2. The ID of the classifier which detected this glyph.

3. The rectangle which defines the detected glyph that was returned from applying the classifier.

4. The centre point of the detected glyph. This point is used for defining the text line and glyphs parsing.

5. The area of the detected glyph. The area is used to decide whether the detected glyph area is reasonable or not (which means it is wrongly detected).

6. The average grey scale value of the detected glyph. This is used to decide whether the detected glyph is a physical or empty glyph.

7. The true positive (TP) percentage which resulted when applying the detecting classifier as shown in section 4.6.4. This value is used to define the best likelihood of alternative glyphs.

8. The line number which contains the glyph. The letter number of the detected glyph in the line is also considered in this case. These two parameters are calculated in the next section.

Table 5.3 shows a sample list of glyphs when running a tokens classifier over a document image. This list is considered as the main source of data for the next steps in the creation of the OCR application.

Table 5.3: Sample of applying a Tokens classifier over a document image

| Glyph ID | Classifier ID | Top left X | Top left Y | Width | Height | Centre X | Centre Y | TP | Glyph Area | Grey Value |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 445 | 4 | 14 | 33 | 452 | 20 | 0.937 | 462 | 180 |
| 20 | 2 | 128 | 3 | 13 | 29 | 134 | 17 | 0.929 | 377 | 187 |
| 35 | 4 | 184 | 10 | 12 | 29 | 190 | 24 | 0.808 | 348 | 183 |
| 62 | 5 | 30 | 47 | 17 | 39 | 38 | 66 | 0.736 | 663 | 191 |
| 79 | 6 | 358 | 348 | 28 | 33 | 372 | 364 | 0.931 | 924 | 191 |
| 85 | 11 | 197 | 53 | 17 | 24 | 205 | 65 | 0.847 | 408 | 182 |
| 90 | 12 | 308 | 133 | 18 | 30 | 317 | 148 | 0.934 | 540 | 185 |
| 101 | 13 | 150 | 312 | 18 | 25 | 159 | 324 | 0.733 | 450 | 182 |
| 106 | 14 | 190 | 5 | 21 | 37 | 200 | 23 | 0.742 | 777s | 192 |
| 108 | 16 | 185 | 174 | 33 | 29 | 201 | 188 | 0.922 | 957 | 193 |
| 111 | 17 | 150 | 4 | 33 | 31 | 166 | 19 | 0.926 | 1023 | 192 |
| 113 | 21 | 272 | 260 | 33 | 29 | 288 | 274 | 0.894 | 957 | 181 |
| 115 | 23 | 196 | 84 | 31 | 36 | 211 | 102 | 0.951 | 1116 | 180 |
| 116 | 25 | 291 | 338 | 24 | 37 | 303 | 356 | 0.925 | 888 | 179 |
| 120 | 28 | 209 | 89 | 21 | 31 | 219 | 104 | 0.918 | 651 | 185 |
| 122 | 29 | 77 | 214 | 18 | 33 | 86 | 230 | 0.922 | 594 | 184 |

The detected glyphs average area is shown in section 5.1.3, the classifier average area shown in Table 5.2 and the detected glyph area shown in Table 5.3 are used in order to decide whether to accept a detected glyph or to reject it. It is rejected when a detected glyph area is greater than triple the value of the detected glyphs' average multiplied by the classifier average area. This value (triple) was obtained while testing the application and concluded that the glyphs detected with bigger value than the triple is wrong detection.

The average grey scale of the document image is shown in section 5.1.3 and the average grey scale of the detected glyph shown in Table 5.3 is also used to decide whether to accept the detected glyph or to reject it. It is rejected when the grey scale of a detected glyph is more than the average document grey scale value.

## 5.3 The HCC tokens recognizer

The tokens recognizer uses the list of detected glyphs generated in section 5.2 in order to extract the document's tokens. It separates the detected glyphs into text lines and then into tokens. The resulted tokens are then tested against the commercial software using the tokens testing dataset generated in section 5.1.2. The mismatched glyphs are collected together in order to get the best choice of them through applying the confusion matrix as shown in section 5.4.

## 5.3.1 Text line extraction

The location of a baseline in this research is obtained from the centre Y-axis of the detected glyphs and a horizontal histogram. Text line extraction is the first step in generating a robust tokens recognizer. It is used to define the text line of each detected glyph from the list of glyphs. The glyphs of each text line are then sorted in an Arabic order with the start X value as a key. This sorting process defines the letter number for each detected glyph.

The text line is extracted based on the detected glyphs of the document image. The detected glyphs are sorted based on the Y-axis of the centre point of the detected glyph. The text line is defined by calculating the difference between every two contiguous values, and when the difference exceeds the average height of the detected glyphs, it considers it as a new text line. The baseline of the text line is obtained based on the horizontal histogram algorithm and work by calculating the sum of the grey-scale values of each row of pixels in the document image (O'Gorman 1993). The average Y-axis of the centre points of the detected glyphs that share the same text line is calculated which represents an approximate value of the baseline. A range of values in the horizontal histogram above and below the previous calculated approximate baseline is investigated in order to get the minimum value which represents the most accurate baseline. The top and bottom line is also defined for each baseline using the maximum difference algorithm. Figure 5.4 shows the application of the algorithm in order to define the document image baselines on one of the tokens testing datasets.

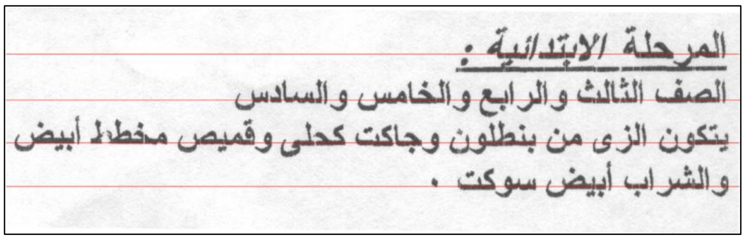

المرحلة الابتدائية :
الصف الثالث والرابع والخامس والسادس
يتكون الزى من بنطلون وجاكت كحلى وقميص مخطط أبيض
والشراب أبيض سوكت .

Figure 5.4: Baseline locations for a document image (shown in red)

The previous algorithm was applied to the testing dataset which contains 23 different document images varying in quality and contains 141 text lines overall. Although some of the document quality was bad, the results were 100% correct in detecting the text lines. The algorithm neither failed to detect any text line nor detected any extra text lines. The algorithm also succeeded in obtaining the baselines in all the testing datasets correctly and precisely as shown in Figure 5.4. The rejection of empty glyphs and big glyphs shown in section 5.2.3 supported the algorithm in order to achieve these results in text line detection.

Table 5.4 shows the same information of the document image that was shown in Table 5.3 after defining the line number of each glyph and letter number. It is clear from the table that the detected glyphs in each text line have almost the same Y centre.

Table 5.4: Sample of detected glyphs with text lines and letters sorted

| Glyph ID | Classifier ID | Top left X | Top left Y | Width | Height | Start X | Centre Y | TP | Grey Val. | Line No. | Letter No. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 445 | 4 | 14 | 33 | 459 | 20 | 0.94 | 180 | 1 | 1 |
| 106 | 14 | 190 | 5 | 21 | 37 | 211 | 23 | 0.74 | 192 | 1 | 11 |
| 35 | 4 | 184 | 10 | 12 | 29 | 196 | 24 | 0.81 | 183 | 1 | 12 |
| 111 | 17 | 150 | 4 | 33 | 31 | 183 | 19 | 0.93 | 192 | 1 | 13 |
| 20 | 2 | 128 | 3 | 13 | 29 | 141 | 17 | 0.93 | 187 | 1 | 14 |
| 85 | 11 | 197 | 53 | 17 | 24 | 214 | 65 | 0.85 | 182 | 2 | 12 |
| 62 | 5 | 30 | 47 | 17 | 39 | 47 | 66 | 0.74 | 191 | 2 | 21 |
| 120 | 28 | 209 | 89 | 21 | 31 | 230 | 104 | 0.92 | 185 | 3 | 12 |
| 115 | 23 | 196 | 84 | 31 | 36 | 227 | 102 | 0.95 | 180 | 3 | 14 |
| 90 | 12 | 308 | 133 | 18 | 30 | 326 | 148 | 0.93 | 185 | 4 | 6 |
| 108 | 16 | 185 | 174 | 33 | 29 | 218 | 188 | 0.92 | 193 | 5 | 15 |
| 122 | 29 | 77 | 214 | 18 | 33 | 95 | 230 | 0.92 | 184 | 6 | 26 |
| 113 | 21 | 272 | 260 | 33 | 29 | 305 | 274 | 0.89 | 181 | 7 | 13 |
| 101 | 13 | 150 | 312 | 18 | 25 | 168 | 324 | 0.73 | 182 | 8 | 16 |
| 79 | 6 | 358 | 348 | 28 | 33 | 386 | 364 | 0.93 | 191 | 9 | 4 |
| 116 | 25 | 291 | 338 | 24 | 37 | 315 | 356 | 0.93 | 179 | 9 | 8 |

### 5.3.2 Parsing the detected glyphs

Parsing the detected glyphs is the most important part in generating a tokens recognizer. It decides the relationship between every two contiguous detected glyphs. This relationship has many variants, especially in the Arabic language. It also deals with the vacant spaces between glyphs inside the text lines. The analysis results have a substantial influence on the recognition accuracy of any OCR application, but are largely beyond the scope of the current research.

Any two contiguous detected glyphs might be a glyph detected by two classifiers, two glyphs with white space in between, two glyphs with the extension (Tatweel) character in-between, two glyphs with an undetected (missed) glyph in-between and finally two actual contiguous glyphs. Studying the relationship between the contiguous glyphs is not a major aspect of this research, so the algorithm used in solving this problem is simple simply to show the problem can be solved. Overall recognition rates could be improved if a more sophisticated algorithm were developed.

Table 5.4 shows the detected glyphs sorted in each line; which is the ideal situation, but is not expected to be the case all the time. The reality is that some glyphs are missed without detection and sometimes more than one classifier detects the same glyph. For these reasons, a data structure for the glyphs has been developed in order to accommodate all scenarios. This structure keeps the line number and character number for each glyph and a list of the classifiers which detected this glyph. Each classifier also keeps a classifier ID and the TP of this classifier when tested.

The algorithm deals with all the cases concerning the relationship between every contiguous glyph as shown hereafter.

#### *5.3.2.1 Two classifiers detect the same glyph*

This case occurs when two or more classifiers detect the same glyph. Figure 5.5 (a) shows two samples each of two classifiers detecting the same glyph. In the first sample Alef isolated glyph (ا) is detected by Alef isolated (ا) and Alef end (ـا) classifiers. In the second sample Alef end glyph (ـا) is detected by Alef isolated (ا) and Alef end (ـا).

This case is checked through investigating the shared width of the two glyphs. If most of their shared width is common then it is assumed they have detected the same glyph.

In this case, both classifiers are kept as only one detected glyph with two records inside in the list of classifiers. This case is managed through sorting of the array of classifiers with the TP percentage. So if the TP of the first classifier is 88% and for the second one is 90%, it will be sorted. The first is the one with 90% TP and the second is the one with 88% TP. In the process of extracting that glyph it will extract the one with higher TP percentage (90%).

Using the classifier with higher TP is convenient as it is the only available information at this stage. This result will be checked again in section 5.4 when applying a confusion matrix in order to enhance the recognition accuracy. It will be checked once more in Chapter 6 with a post-processing algorithm.

#### *5.3.2.2 Two classifiers detect two contiguous glyphs*

This is the ideal case which occurs when the two classifiers detect two contiguous glyphs or detect two glyphs with an extension (Tatweel) glyph between them. Figure 5.5 (b) shows two samples of this case. In the first sample, two contiguous glyphs, Alef isolated (ا) and Lam Alef isolated (لا) are detected correctly. In the second sample, Ain middle (ـعـ) and Heh middle (ـهـ) are detected with an extension glyph between them.

This case is checked through investigating the shared width or the space between the two glyphs. If the space is small relative to the average width of the detected glyphs or if the space between them has no ink above or below the baseline, they detect two different glyphs.

In this case, the two classifiers are kept as two detected glyphs with only records inside each of them in the list of classifiers.

#### *5.3.2.3 Two classifiers detect two glyphs with space between*

This case occurs when the two classifiers detect two glyphs with white space between them. Figure 5.5 (c) shows two samples of this case. In the first sample, two glyphs; Heh end (ـه) and Feh start (فـ), are detected with white space between them. In the second sample, Heh end (ـه) and Meem start (مـ) are detected with white space between them.

This case is checked through investigating the space between the two glyphs. If the space is large relative to the average width of the detected glyphs and the space between them has no ink above or below or on the baseline then there is a white space between the two detected glyphs.

In this case, it keeps the two classifiers as two detected glyphs with only records inside each of them in the list of classifiers. A new glyph is generated in-between with a glyph ID of 200 which represents white space.

#### 5.3.2.4 Two classifiers detect two glyphs with a missed glyph between

This case occurs when the two classifiers detect two glyphs with a missed glyph(s) between them. Figure 5.5 (d) shows two samples of this case. In the first sample, two glyphs; Feh start (ﻓ) and Alef end (ﺎ), are detected with Feh middle (ﻔ) missed between them. In the second sample, Alef end (ﺎ) and Ain middle (ﻌ) are detected with Beh start (ﺑ) missed between them.

This case is checked through investigating the space between the two glyphs. If the space is large relative to the average width of the detected glyphs and the space between them has ink above or below the baseline and on the baseline so there is a missed glyph(s) between the two detected glyphs.

In this case, it keeps the two classifiers as two detected glyphs with only records inside each of them in the list of classifiers. A new glyph (؟) is generated in-between with a glyph ID of 300 which represents a missed glyph.

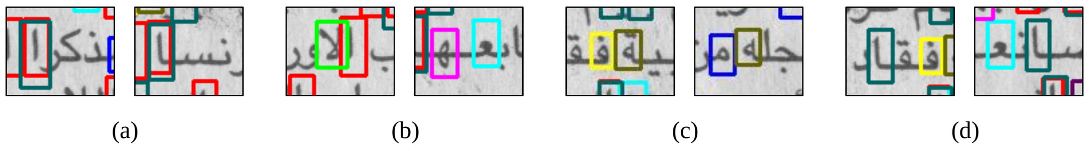


Figure 5.5: Different cases of relationship between each contiguous glyph

The start and end of the text lines are defined with the first and last detected glyph in the text line from right to left.

### 5.3.3 Testing the Arabic tokens recognizer

Testing the Arabic tokens recognizer was done using the tokens testing dataset that was prepared in section 5.1.2. This dataset contains of 1,053 words, 2,251 PAWs and 5,855 characters. The dataset includes a variety of document types, and was ground truthed as text files. The ground truth files are in different forms of tokens which include words, naked words, PAWs and naked PAWs. The HCC tokens recognizer output is in the form of naked words as the classifiers generated in section 4.6 were in the form of naked glyphs; that is why the tokens used in this testing process were in the form of naked words tokens.

The Arabic tokens recognizer was tested against the commercial software in order to check its usefulness as a real OCR application. The well-known commercial software Readiris pro 10 (IRIS 2004) which was used in section 3.3 was used for testing the tokens recognizer. Recognition results from the commercial software and the tokens recognizer were compared to the ground truthed documents using the Levenshtein Distance algorithm (Navarro 2001). The Levenshtein Distance algorithm is an approximate string matching algorithm (Levenshtein 1966) which gives a tentative accuracy of recognition (AbdelRaouf, Higgins et al. 2010). This accuracy is measured based on the percentage of correct characters within the volume of characters covered, defined by the majority of OCR application vendors (Tanner, Muñoz et al. 2009).

Table 5.5 shows the recognition results between the commercial software and the HCC tokens recognizer using Levenshtein Distance algorithm. The characters error shows the number of incorrect characters in the whole document used with the total number of characters in the document in order to calculate the recognition accuracy.

The results of the commercial software showed that the testing dataset had documents of varying quality, as the recognition rate was variable for each document and the overall accuracy was 84%. It is important to mention that the characters' error average is different from that of the recognition rate average. The characters' error average is calculated as the average of the total errors in characters relative to the total number of characters in the testing dataset. The recognition rate average is calculated as the average of the recognition rate of all the documents in the testing dataset.

Table 5.5 shows the recognition time in seconds for every test file on the commercial software and the HCC approach. The calculation of the recognition time is tentative as it is not calculating the accurate duration of recognizing the document images. It is used to justifying that the recognition time of the HCC approach is reasonable. The table also explains that some files takes a long time as recognition time depends on the quality and size of the document. The total time to recognise all the files in the commercial software are 364 seconds with average of 15.8 seconds for each. The total time to recognise all the test files in the HCC approach are 339 seconds with average of 14.7 seconds for each. The recognition time for both the applications is almost the same but slightly less in the case of HCC approach.

Table 5.5: Recognition results of commercial software and HCC recognizer

| Document image file | Original Documents | | Commercial software | | | HCC Tokens parser | | |
|---|---|---|---|---|---|---|---|---|
| | Characters | Words | Time /Sec. | Characters Error | Accuracy | Time /Sec | Characters Error | Accuracy |
| FinalTest001 | 216 | 34 | 10 | 16 | 92.6% | 17 | 55 | 74.5% |
| FinalTest002 | 254 | 41 | 11 | 19 | 92.5% | 17 | 86 | 66.1% |
| FinalTest003 | 124 | 21 | 16 | 36 | 71.0% | 20 | 56 | 54.8% |
| FinalTest004 | 146 | 24 | 15 | 33 | 77.4% | 21 | 61 | 58.2% |
| FinalTest005 | 189 | 34 | 37 | 136 | 28.0% | 48 | 105 | 44.4% |
| FinalTest006 | 161 | 34 | 28 | 82 | 49.1% | 36 | 97 | 39.8% |
| FinalTest007 | 107 | 18 | 10 | 49 | 54.2% | 8 | 48 | 55.1% |
| FinalTest008 | 136 | 22 | 12 | 67 | 50.7% | 11 | 47 | 65.4% |
| FinalTest009 | 361 | 68 | 23 | 34 | 90.6% | 19 | 96 | 73.4% |
| FinalTest010 | 357 | 63 | 25 | 46 | 87.1% | 19 | 89 | 75.1% |
| FinalTest011 | 197 | 38 | 15 | 50 | 74.6% | 9 | 28 | 85.8% |
| FinalTest012 | 175 | 29 | 14 | 12 | 93.1% | 8 | 33 | 81.1% |
| FinalTest013 | 331 | 68 | 16 | 35 | 89.4% | 11 | 102 | 69.2% |
| FinalTest014 | 386 | 75 | 21 | 52 | 86.5% | 13 | 104 | 73.1% |
| FinalTest015 | 458 | 79 | 18 | 62 | 86.5% | 15 | 220 | 52.0% |
| FinalTest016 | 139 | 23 | 10 | 10 | 92.8% | 4 | 56 | 59.7% |
| FinalTest017 | 290 | 50 | 13 | 23 | 92.1% | 9 | 120 | 58.6% |
| FinalTest018 | 360 | 69 | 12 | 33 | 90.8% | 12 | 108 | 70.0% |
| FinalTest019 | 320 | 56 | 12 | 37 | 88.4% | 10 | 87 | 72.8% |
| FinalTest020 | 295 | 58 | 9 | 41 | 86.1% | 8 | 100 | 66.1% |
| FinalTest021 | 329 | 59 | 12 | 49 | 85.1% | 9 | 137 | 58.4% |
| FinalTest022 | 250 | 43 | 10 | 2 | 99.2% | 7 | 49 | 80.4% |
| FinalTest023 | 274 | 47 | 15 | 7 | 97.4% | 8 | 56 | 79.6% |
| Total | 5855 | 1053 | 364 | 931 | | 339 | 1940 | |
| Average | 254.6 | 87.75 | 15.8 | 84% | 81% | 14.7 | 67% | 66% |

The results of the HCC tokens recognizer shows that the algorithm that was used in order to parse the detected glyphs into tokens works and that it extracts tokens with a good accuracy although that is not the main focus of this research. The results also show that the algorithm used in the HCC parsing process is not the best as the overall HCC tokens recognition accuracy is 67% is much less than the HCC glyphs recognition accuracy of 88% shown in section 4.6. The commercial software recognition accuracy is 84% which is much more that of the HCC recognizer 67%, although in other comparison with glyphs recognition results shown in section 4.4.6.2; they get almost the same results.

## 5.4 Enhancing recognition using a confusion matrix

A confusion matrix aims to enhance the recognition accuracy which results from the tokens recognizer generated in section 5.3 with one of the supervised machine learning algorithms. The confusion matrix is generated from the data prepared in section 4.6.2 which is used in order to train and test the 61 classifiers shown in section 4.6.

A confusion matrix is a matrix showing the actual and predicate classification of the samples (Kohavi and Provost 1998). The confusion matrix in this research deals with the 61 classifiers that were generated in section 4.6 and is squared with size L equal to 61. The confusion matrix is generated for each actual classifier by running all the 61 classifiers over all the glyphs in order to get the number of the glyphs that are detected by each of the 61 classifiers (Vamvakas, Gatos et al. 2010). The confusion matrix is built using a list of likelihood alternative classifiers. For example, in order to build the Beh start classifier alternative list, all classifiers must be run and the percentage for each classifier calculated in order to detect Beh start and then build the confusion matrix from the 61 classifiers (Taghva and Stofsky 2001). This task used the list of positive images that were prepared in section 4.6 in order to train and test the glyph classifiers.

### 5.4.1 Generating the confusion matrix

The confusion matrix was applied in order to handle the misrecognized glyphs that occurred most frequently. The confusion matrix contains an original classifier and its associated non-original (alternative) classifiers, i.e., classifiers which are likelihood to misrecognize a glyph as the original classifier (Ohta, Takasu et al. 1997). The contents of the confusion matrix are the TP (True Positive) of the original classifier and the FP (False Positive) when applying the non-original classifiers to the original one. The occurrence of TP is shown in Table 2.5 as D, while occurrence of FP is shown as B.

The confusion matrix values are saved as a percentage because the number of samples in the 61 classifiers is varying and not the same. Using the percentage is considered to be a normal way of comparing the 61 different classifiers (Ohta, Takasu et al. 1997). A separate program was developed in order to generate the confusion matrix for the continuous enhancement of the confusion matrix in order to improve the recognition rate of the HCC approach. This method enables anyone to generate a confusion matrix using this application, for instance when using an enhanced training and testing set of data.

The application uses the 61 XML files which represent the 61 classifiers generated in section 4.6. The application also uses the positive training and testing files of each of the 61 classifiers. The application runs through a loop on each original classifier sequentially. Inside the loop process, it runs each of the 61 classifiers using the parameters of the original one. For example it begins with using classifier 1 as the original classifier then runs the classifiers from 1 to 61 on the positive files of classifier 1. The application uses the testing parameters of classifier 1 when it runs all the classifiers from 1 to 61. These parameters are width, height, scale factor and minimum neighbours. The application stores the number of glyphs detected by each non-original (alternative) classifier as an original classifier 1. In the case of running classifier 1 with itself as original classifier, the number is defined as the TP of classifier 1. In the case of running classifiers from 2 to 61 as non-original classifiers with classifier 1, it is defined as the FP of classifier 1 with the other classifiers.

Table 5.6: Part of confusion matrix with a sample of six classifiers

| | | Predicted | | | | | |
|---|---|---|---|---|---|---|---|
| | | Alef isolated | Alef end | Beh isolated | Beh start | Beh middle | Beh end |
| Actual | Alef isolated | 0.955172 | 0.006034 | 0 | 0 | 0 | 0 |
| | Alef end | 0.016374 | 0.97193 | 0 | 0 | 0 | 0 |
| | Beh isolated | 0 | 0 | 0.929368 | 0.011152 | 0 | 0 |
| | Beh start | 0 | 0 | 0.002188 | 0.847921 | 0.008753 | 0 |
| | Beh middle | 0 | 0 | 0.003831 | 0 | 0.773946 | 0 |
| | Beh end | 0 | 0 | 0.163636 | 0 | 0 | 0.945455 |

The output of the application is a classifier confusion matrix of size 61. The element of each cell in the matrix is the percentage of occurrence of these two classifiers together. A text file for the confusion matrix is generated at the end and is separated by a tab to facilitate easy importing into any suitable application. Table 5.6 shows part of the generated confusion matrix which includes the first six classifiers as an example. Each row in the matrix represents the original (testing) classifier that is used with the others non-original classifiers that are on the columns. For example the first row Alef isolated gets 95% TP. When applying the Alef end classifier with the Alef isolated files it gets 0.60% FP, and so on.

Applying the confusion matrix showed the following comments:

1. The output values of the confusion matrix are different from that obtained from testing the 61 classifiers (see section 4.6.4.2) as the files used here are the positive files that were used in the training and testing process; the rotated samples are not used.

2. The same letter in different locations frequently being alternatives. For example Beh isolated (ب) and Beh start (بـ).

3. The letters with unique shapes usually do not have alternatives. For example; Heh mid (ـهـ) and Lam Alef isolated (لا).

4. Some symbols are detected randomly. For example (») is sometimes detected as (ـد) and at other times it is detected as (ا).

### 5.4.2 Applying the confusion matrix

The confusion matrix in this research was generated in section 5.4.1 from data files prepared in section 4.6.2 and using the classifiers generated in section 4.6.3. The accuracy and efficiency of this matrix are totally dependent on the data collected, trained and tested in this research. How the confusion matrix is used differs from research to research but in most cases it is used in order to enhance the recognition accuracy. The confusion matrix in this research is used in order to enhance the recognition accuracy at the glyphs level, not at the level of PAWs or words (Ohta, Takasu et al. 1997). Chapter 6 presents another technique to improve the recognition accuracy in the words and PAWs level.

Table 5.7: Confusion matrix likelihood alternatives glyphs list

| First (Same) classifier | | | Second | | Third | | Fourth | | Fifth | | Sixth | | Seventh | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | AlefISO | .955 | 42 | .049 | 2 | .006 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 2 | AlefEND | .972 | 1 | .016 | 44 | .025 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 3 | BehISO | .929 | 4 | .011 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 4 | BehSTR | .848 | 5 | .009 | 49 | .008 | 32 | .003 | 3 | .002 | 20 | .002 | 55 | .001 |
| 5 | BehMID | .774 | 3 | .004 | 9 | .002 | 49 | .002 | 40 | .002 | 0 | .000 | 0 | .000 |
| 6 | BehEND | .945 | 3 | .164 | 34 | .010 | 9 | .009 | 0 | .000 | 0 | .000 | 0 | .000 |
| 7 | HahISO | .759 | 27 | .103 | 51 | .034 | 23 | .017 | 8 | .017 | 37 | .017 | 0 | .000 |
| 8 | HahSTR | .947 | 9 | .208 | 51 | .009 | 23 | .004 | 25 | .004 | 0 | .000 | 0 | .000 |
| 9 | HahMID | .867 | 8 | .010 | 47 | .010 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 10 | HahEND | 1.00 | 5 | .053 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 11 | DalISO | .903 | 42 | .047 | 4 | .016 | 49 | .016 | 13 | .008 | 40 | .008 | 5 | .004 |
| 12 | DalEND | .947 | 5 | .016 | 13 | .013 | 40 | .003 | 0 | .000 | 0 | .000 | 0 | .000 |
| 13 | RehISO | .869 | 55 | .027 | 50 | .019 | 14 | .016 | 12 | .016 | 54 | .013 | 4 | .006 |
| 14 | RehEND | .776 | 13 | .051 | 15 | .011 | 56 | .007 | 19 | .004 | 50 | .002 | 4 | .002 |
| 15 | SeenISO | .815 | 18 | .704 | 19 | .296 | 22 | .037 | 0 | .000 | 0 | .000 | 0 | .000 |
| 16 | SeenSTR | .946 | 46 | .047 | 17 | .027 | 20 | .013 | 49 | .007 | 0 | .000 | 0 | .000 |
| 17 | SeenMID | .959 | 20 | .005 | 21 | .005 | 18 | .005 | 0 | .000 | 0 | .000 | 0 | .000 |
| 18 | SeenEND | .907 | 58 | .116 | 15 | .116 | 19 | .047 | 17 | .023 | 0 | .000 | 0 | .000 |
| 19 | SadISO | .923 | 22 | .308 | 15 | .192 | 18 | .192 | 58 | .115 | 0 | .000 | 0 | .000 |
| 20 | SadSTR | .933 | 21 | .311 | 8 | .044 | 4 | .022 | 16 | .022 | 0 | .000 | 0 | .000 |
| 21 | SadMID | .991 | 9 | .071 | 20 | .018 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 22 | SadEND | .926 | 19 | .704 | 18 | .370 | 21 | .259 | 58 | .037 | 0 | .000 | 0 | .000 |
| 23 | TahISO | .889 | 24 | .556 | 26 | .333 | 25 | .111 | 3 | .111 | 51 | .041 | 0 | .000 |
| 24 | TahSTR | .824 | 25 | .324 | 23 | .088 | 43 | .029 | 26 | .015 | 20 | .015 | 0 | .000 |
| 25 | TahMID | .843 | 21 | .024 | 24 | .012 | 43 | .012 | 0 | .000 | 0 | .000 | 0 | .000 |
| 26 | TahEND | .929 | 25 | .786 | 23 | .214 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 27 | AinISO | .917 | 59 | .083 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 28 | AinSTR | .937 | 59 | .117 | 8 | .027 | 25 | .022 | 23 | .013 | 0 | .000 | 0 | .000 |
| 29 | AinMID | .919 | 46 | .022 | 25 | .011 | 33 | .011 | 31 | .005 | 0 | .000 | 0 | .000 |
| 30 | AinEND | .932 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 31 | FehISO | .933 | 34 | .633 | 32 | .233 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 32 | FehSTR | .972 | 55 | .018 | 31 | .015 | 54 | .009 | 33 | .006 | 45 | .003 | 51 | .003 |
| 33 | FehMID | .307 | 40 | .013 | 54 | .004 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 34 | FehEND | 1.00 | 31 | .167 | 6 | .128 | 33 | .028 | 0 | .000 | 0 | .000 | 0 | .000 |
| 35 | QafISO | .730 | 55 | .270 | 36 | .216 | 32 | .162 | 11 | .135 | 51 | .081 | 23 | .054 |
| 36 | QafEND | .778 | 56 | .444 | 54 | .111 | 35 | .111 | 0 | .000 | 0 | .000 | 0 | .000 |
| 37 | KafISO | .913 | 40 | .304 | 42 | .174 | 54 | .043 | 0 | .000 | 0 | .000 | 0 | .000 |
| 38 | KafSTR | .961 | 20 | .019 | 55 | .010 | 57 | .010 | 60 | .010 | 0 | .000 | 0 | .000 |
| 39 | KafMID | .975 | 5 | .034 | 38 | .017 | 40 | .008 | 9 | .008 | 0 | .000 | 0 | .000 |
| 40 | KafEND | .967 | 37 | .033 | 12 | .033 | 25 | .033 | 0 | .000 | 0 | .000 | 0 | .000 |
| 41 | LamISO | .879 | 42 | .262 | 43 | .037 | 11 | .028 | 37 | .009 | 35 | .009 | 49 | .009 |
| 42 | LamSTR | .938 | 1 | .010 | 41 | .004 | 43 | .001 | 0 | .000 | 0 | .000 | 0 | .000 |
| 43 | LamMID | .923 | 25 | .019 | 33 | .005 | 12 | .005 | 0 | .000 | 0 | .000 | 0 | .000 |
| 44 | LamEND | .953 | 41 | .047 | 42 | .012 | 23 | .012 | 19 | .012 | 0 | .000 | 0 | .000 |
| 45 | MeemISO | .798 | 48 | .263 | 46 | .026 | 1 | .009 | 0 | .000 | 0 | .000 | 0 | .000 |
| 46 | MeemSTR | .924 | 47 | .038 | 29 | .006 | 20 | .006 | 31 | .003 | 52 | .003 | 14 | .003 |
| 47 | MeemMID | 1.00 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 48 | MeemEND | .764 | 45 | .125 | 53 | .028 | 37 | .014 | 9 | .014 | 0 | .000 | 0 | .000 |
| 49 | NoonISO | .881 | 4 | .027 | 50 | .022 | 13 | .102 | 11 | .011 | 32 | .005 | 3 | .005 |
| 50 | NoonEND | .929 | 49 | .095 | 15 | .024 | 19 | .016 | 4 | .016 | 0 | .000 | 0 | .000 |
| 51 | HehISO | .875 | 32 | .035 | 54 | .020 | 23 | .020 | 49 | .005 | 37 | .005 | 24 | .005 |
| 52 | HehSTR | .984 | 46 | .056 | 16 | .008 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 53 | HehMID | .942 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 54 | HehEND | .920 | 51 | .023 | 33 | .002 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 55 | WawISO | .860 | 35 | .432 | 13 | .025 | 56 | .023 | 54 | .003 | 29 | .001 | 50 | .001 |
| 56 | WawEND | .976 | 14 | .210 | 55 | .174 | 32 | .006 | 36 | .003 | 51 | .003 | 0 | .000 |
| 57 | YehISO | .965 | 58 | .062 | 38 | .009 | 53 | .009 | 37 | .009 | 0 | .000 | 0 | .000 |
| 58 | YehEND | .950 | 4 | .008 | 55 | .006 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 59 | HamzaISO | .604 | 51 | .042 | 38 | .021 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 60 | LamAlefISO | 1.00 | 37 | .003 | 35 | .003 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |
| 61 | LamAlefEND | .983 | 26 | .017 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 | 0 | .000 |

The confusion matrix is loaded into the main application of the Arabic tokens recognition system. For each glyph, the likelihood alternatives glyphs are sorted in descending order, in order to keep the higher value first. Normally the confusion matrix is a square matrix but in this case it is not convenient to store the whole of that square matrix. Studying the matrix showed that the maximum number of likelihood alternatives glyphs for any original glyph is seven. Percentages other than the top seven are negligible. Table 5.7 shows the likelihood alternatives glyphs for each original glyph from the 61 that were generated. This study showed that there are eleven glyphs that have seven likelihood alternatives glyphs, but only one glyph; Qaf isolated (ق), that has a significant value in the seventh position (likelihood alternative glyph Tah isolated (ط) at 5.4%). For that reason, the application keeps the top seven likelihood alternatives glyphs for every original glyph and at the top is the original glyph itself.

The confusion matrix is applied to a glyph that is detected by more than one classifier. When a glyph is detected by more than one classifier, a new list is generated in order to keep all the likelihood alternatives glyphs to the original detected glyph. For example, if a glyph is detected by two different classifiers then a list including 14 alternatives glyphs is generated (seven for each glyph). When two or more alternative glyphs are the same then the likelihood is added together and the list is sorted again with these new values. The glyph with the highest likelihood is chosen. The confusion matrix has no effect on the white space or missed glyphs, but deals with them as a separate glyph. In this case, the confusion matrix has the ability to change the original glyph by a likelihood alternative glyph when is detected by more than one classifier. The resultant glyph is the one with higher percentage of TP which can enhance the recognition rate of the HCC approach.

### 5.4.3 Testing the confusion matrix enhancement results

Testing the confusion matrix used the same dataset that was used before in section 5.3.3 in order to test the tokens recognizer. The testing was done using the tokens testing dataset that were prepared in section 5.1.2. This dataset contained 1,053 words, 2,251 PAWs and 5,855 characters. The dataset included a variety of document types, and were ground truthed as text files. The testing process followed was the same process as in section 5.3.3 in order to test the tokens recognizer. The results of the testing were compared to the results of the tokens recognizer.

The confusion matrix was applied to the glyphs that were detected by more than one classifier and not all the detected glyphs. This led to a comparison of the enhancement in the recognition rate, which was relative to the number of likelihood alternatives glyphs, rather than all the detected glyphs or all the characters in the dataset.

Table 5.8 shows the results of the confusion matrix enhancements compared to the results of the commercial software and HCC tokens recognizer. It shows the number of likelihood alternatives glyphs in each file in the dataset, and also shows the enhancements in the character errors and shows the percentage of enhancement in every single file from the testing dataset. The likelihood alternatives glyphs represent 8% of the total characters in the testing dataset which varies according to the accuracy of the glyph detection and tokens recognizer of the HCC approach.

Table 5.8: Confusion matrix results in comparison to the tokens recognizer

| Document image file | Original Documents | | Commercial software Characters error | Characters Errors | | Enhancement results | |
|---|---|---|---|---|---|---|---|
| | Characters | likelihood alternatives glyphs | | HCC Tokens recognizer | Confusion matrix | Characters Error | Recognition Rate |
| FinalTest001.bmp | 216 | 23 | 16 | 55 | 51 | 4 | 17.39% |
| FinalTest002.bmp | 254 | 22 | 19 | 86 | 87 | -1 | -4.55% |
| FinalTest003.bmp | 124 | 32 | 36 | 56 | 56 | 0 | 0.00% |
| FinalTest004.bmp | 146 | 34 | 33 | 61 | 56 | 5 | 14.71% |
| FinalTest005.bmp | 189 | 27 | 136 | 105 | 103 | 2 | 7.41% |
| FinalTest006.bmp | 161 | 21 | 82 | 97 | 97 | 0 | 0.00% |
| FinalTest007.bmp | 107 | 16 | 49 | 48 | 49 | -1 | -6.25% |
| FinalTest008.bmp | 136 | 17 | 67 | 47 | 46 | 1 | 5.88% |
| FinalTest009.bmp | 361 | 29 | 34 | 96 | 86 | 10 | 34.48% |
| FinalTest010.bmp | 357 | 35 | 46 | 89 | 80 | 9 | 25.71% |
| FinalTest011.bmp | 197 | 14 | 50 | 28 | 27 | 1 | 7.14% |
| FinalTest012.bmp | 175 | 10 | 12 | 33 | 31 | 2 | 20.00% |
| FinalTest013.bmp | 331 | 34 | 35 | 102 | 98 | 4 | 11.76% |
| FinalTest014.bmp | 386 | 24 | 52 | 104 | 108 | -4 | -16.67% |
| FinalTest015.bmp | 458 | 17 | 62 | 220 | 217 | 3 | 17.65% |
| FinalTest016.bmp | 139 | 5 | 10 | 56 | 56 | 0 | 0.00% |
| FinalTest017.bmp | 290 | 6 | 23 | 120 | 121 | -1 | -16.67% |
| FinalTest018.bmp | 360 | 17 | 33 | 108 | 105 | 3 | 17.65% |
| FinalTest019.bmp | 320 | 20 | 37 | 87 | 81 | 6 | 30.00% |
| FinalTest020.bmp | 295 | 15 | 41 | 100 | 104 | -4 | -26.67% |
| FinalTest021.bmp | 329 | 20 | 49 | 137 | 139 | -2 | -10.00% |
| FinalTest022.bmp | 250 | 9 | 2 | 49 | 49 | 0 | 0.00% |
| FinalTest023.bmp | 274 | 17 | 7 | 56 | 54 | 2 | 11.76% |
| Total | 5855 | 464 | 931 | 1940 | 1901 | 39 | 8.4% |

The result of the confusion matrix show that the algorithm that was used in order to enhance the likelihood alternatives glyphs is working well even through this is not the main focus of this research. The results showed an overall enhancement in the recognition of likelihood alternatives glyphs of 8.4% which is good. Note that in some files in the testing dataset the enhancement is negative which indicates a decrease in the recognition rate.

## 5.5 Summary

This chapter explains the generation and implementation of the HCC tokens recognizer. It uses the 61 classifiers that were generated, trained and tested in Chapter 4 in order to collect them into a single recognition application. This application includes the detected glyphs based on the 61 classifiers and is enhanced using the confusion matrix in order to be ready for use in the next chapter. In the next chapter, statistical post-processing techniques are used to enhance the recognition accuracy. The issues explained in this chapter included the following:

1. The main topics in this chapter and include preparing the testing dataset and statistical information from the document image.

2. The creation and use of an Arabic single token classifier. This merges the previously generated 61 classifiers into one.

3. The generation and implementation of the Arabic tokens recognizer. This extracts text lines from the detected glyphs, parses the detected glyphs, using the text lines, into tokens and extracts the detected tokens data. The HCC generated recognizer is tested against commercial software using the tokens testing dataset. HCC achieved recognition accuracy of 67%.

4. Applying the confusion matrix to enhance recognition accuracy. It explained how the confusion matrix is generated, applied to the HCC token recognizer and showed how a confusion matrix can improve recognition rates. The HCC application after applying the confusion matrix was tested against the token recognizer using the tokens testing dataset. The result overall enhancement of likelihood alternatives glyphs is 8.4%.

# Chapter 6.

# Post-processing HCC tokens

This chapter presents a post-processing technique to enhance the HCC Arabic naked tokens recognizer discussed in Chapter 5 using the multi-modal Arabic corpus from Chapter 3. The HCC Arabic naked tokens recognizer stems from the single tokens classifier that was previously produced in section 5.2; was then assembled in order to generate an HCC tokens recognizer in section 5.3 and finally enhanced its recognition rate using a confusion matrix applied in section 5.4. The multi-modal Arabic corpus textual data that was prepared in section 3.1 was used with statistical information generated in section 3.5 as were naked words and naked PAWs dictionaries for the post-processing technique.

The post-processing results were tested using the tokens testing dataset generated in section 5.1.2 and in comparison with the HCC recognizer shown in section 5.3.3. The results of applying the confusion matrix were then shown in section 5.4.3. The testing procedures were applied using two different types of look-up dictionaries; naked words and naked PAWs. The naked PAWs dictionary was used in order to justify the usability and usefulness of this type of token for the Arabic language and in order to prove its efficiency over the naked words.

## 6.1 Introduction

The post-processing stage is responsible for the correction of errors or resolving ambiguities in OCR results by using contextual information. A dictionary look-up method is the most commonly used post-processing technique (Hulden 2009). The output of the OCR is compared to the contents of the look-up dictionary and appropriate candidates are generated. According to the difference between the output of OCR and the output of the dictionary look-up, some recognized tokens are modified based on the string correction algorithm used (Lehal1, Singh et al. 2001).

If there is an available look-up dictionary that covers all possible input words, a simple procedure may be used for detecting and correcting errors. The input token is first checked in order to assess whether the token is in the dictionary. If the token is missing from the dictionary, the most similar tokens to the input one are suggested as correction candidates. If necessary, appropriate statistical data can be used for refinement of a token's ranking. Similarity between two tokens can be measured in several ways. Most useful are similarity measures based on variants of the Levenshtein distance (Schulz and Mihov 2002).

### 6.1.1 Naked words versus naked PAWs

The look-up dictionary has an important role in the post-processing phase. It is used to compare the OCR systems detected token with its components and is also used to extract the best similar token to the one that has been detected (Reffle 2011). The multi-modal Arabic corpus (MMAC) prepared in Chapter 3 was generated in order to serve this purpose (among to others) and was justified and verified in order to fulfil the standard requirements in generating a corpus. MMAC includes 6 million Arabic words which produced 14 million PAWs, with 211,072 unique NWords and 32,804 unique NPAWs (see section 3.5). MMAC was tested through this research (see section 3.6) and by external parties from research and industry backgrounds.

The HCC tokens recognizer generated in Chapter 5 detects naked glyphs, which indicates that it is necessary to use a look-up dictionary with naked tokens as generated in section 3.1. The HCC recognizer extracts the detected tokens in the forms of NPAWs and NWords, ready for testing with NPAWs and NWords look-up dictionaries.

This research places emphasis on the efficiency of the PAW as an important token in the Arabic language (AbdelRaouf, Higgins et al. 2010). For that reason it was important to test the post-processing technique once with NPAWs and once more with the NWords. This testing technique is a good indicator of the importance of the PAWs as a token in the Arabic language. The PAWs were used before in some limited research such as (Najoua and Noureddine 1995; Mehran, Pirsiavash et al. 2005) but were not reported as being applied as a look-up dictionary, as is the case in this research.

## 6.2 Applying a post-processing technique

The post-processing technique in this research depends on the examination of each recognized token from the HCC tokens recognizer after having enhanced its accuracy by applying a confusion matrix. The examined token is first inspected by the look-up dictionary that is implemented in the form of a tail-end trie structure. If the examined token is missing from the look-up dictionary, it uses the Levenshtein distance algorithm in order to obtain the best similar token to the examined token based on the number of occurrences of similar sized tokens (Schulz and Mihov 2002).

The trie data structure is used because it takes advantage of the redundancy of common prefixes which is very common in this research. The storage method allows fast searches in both acceptance and rejection of the input string (Elliman and Lancaster 1990). The Levenshtein distance is used because it is a very fast method for correcting corrupted input strings of unrestricted text using large look-up dictionary. The Levenshtein distance is very common in use with the character recognition problems (Schulz and Mihov 2002). Schulz and Mihov (Schulz and Mihov 2002) applied the two algorithm together for the post-processing with good results.

### 6.2.1 Searching using a trie structure

The trie structure is the most frequently used structure for searching through the look-up dictionaries (Elliman and Lancaster 1990). The trie structure in this research uses the look-up dictionary from naked words and naked PAWs. The tail-end type of trie structure is implemented in this research, which is explained in section 2.7.2.1, and is shown graphically in Figure 2.7. The tail-end trie is used because it is fast and uses less memory in order to find the examined token, although it is rebuilt from the textual data every runtime.

The trie structure has been implemented using the list of naked words generated in section 3.1 which contains 6,000,000 naked words. It is loaded and generates 12 different levels of trie which means that the longest NWord in the corpus is more than or equal to 12 characters. The trie structure was loaded once more with the list of naked PAWs generated in section 3.1 which contains 14,025,044 naked PAWs.

This trie generated 9 levels which means that the longest NPAW's in the corpus is more than or equal to 9 characters. Each trie node is kept in a structure containing the Arabic character(s) of the node, the level of the node and the number of occurrences of the characters' sequences at this level.

While running the tail-end trie structure in this research it was noted that, as the look-up dictionary became bigger, the tail-end trie acted more like the traditional trie type. The more tokens there were in the dictionary the greater the split of tokens into characters in the trie and so more trie levels. For example, the NPAWs dictionary contains 14 million tokens; the trie levels are 9 with the longest NPAW being 11 characters, while NWords dictionary contains 6 million tokens; where the trie levels are 12 with the longest NWord being 16 characters.

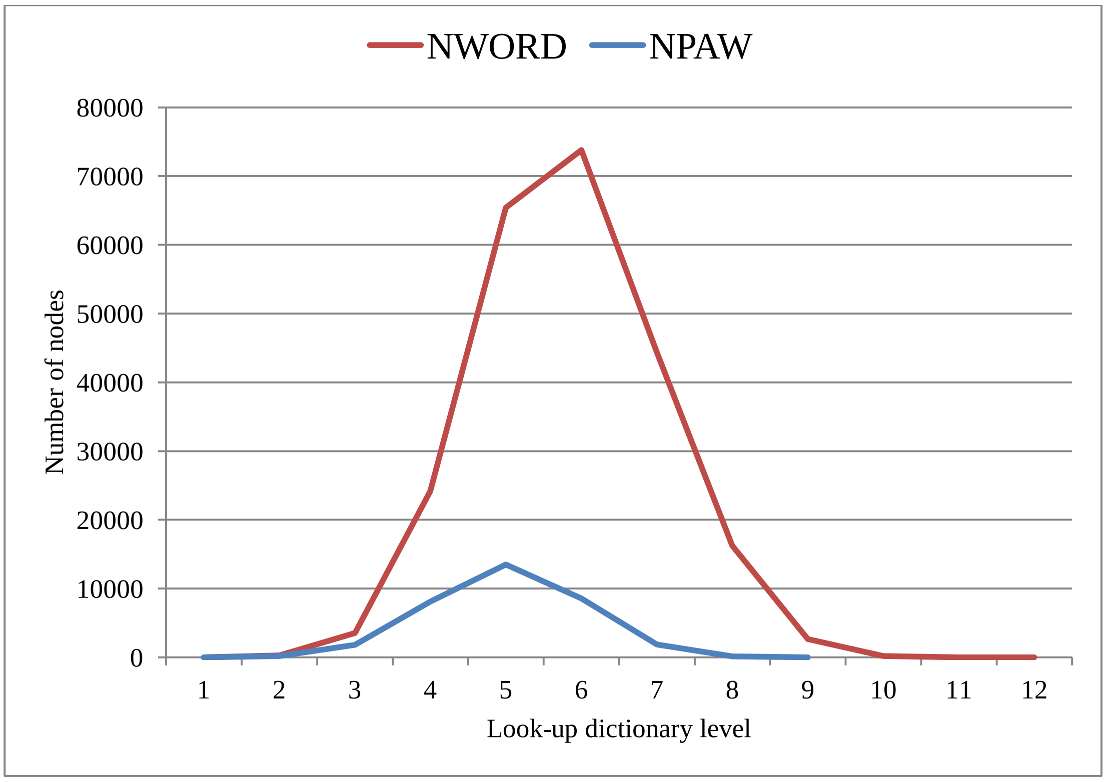


Figure 6.1: Relationship between look-up dictionaries' levels and nodes number

Figure 6.1 shows the relationship between the look-up dictionary levels and the number of nodes at each level, as applied to the naked words and naked PAWs. The most branching level (maximum number of nodes) in the NPAWs trie is at the fifth level, while with the NWords trie is at the sixth level. The figure shows that the NWords trie size is almost 7 times that of the NPAWs trie size which indicates that the NPAWs trie uses less memory and also indicates that the NPAW look-up dictionary is much faster.

### 6.2.2 Token correction using Levenshtein distance

The Levenshtein distance is applied by calculating the distance between the detected token from the HCC token recognizer that is missing from the look-up dictionary and every token in the look-up dictionary. Given the input token $P$ and a bound $n$, the deterministic finite state automaton $A$ accepts exactly all tokens $W$ where length of $P$ and $W$ are equal and the Levenshtein distance between $P$ and $W$ does not exceed $n$. $A$ is called a Levenshtein automaton for $P$. The Levenshtein distance bound $n$ in this research is 2 as reported from (Schulz and Mihov 2002; Mihov and Schulz 2004).

A list of dictionary tokens that are guaranteed to contain all the corrections of the input token $P$ is generated which include the token and number of occurrences of that token in the corpus. A fine search is applied for each candidate where the distance to the input token $P$ is computed, using a fine-graded measure. Candidates are ranked, first according to this distance, then according to the number of occurrences of tokens and the best candidate is then suggested as a correction token.

The Levenshtein distance bound $n$ was examined during the application of the algorithm with the NWords and NPAWs dictionary. The tokens with distance $n$=1 were used first and if there were no tokens with distance $n$=1 then selections were made instead from the tokens with distance $n$=2. This method gave better results than only using tokens with distance $n$=1.

Because at this stage, statistical and image information is not considered dictionary tokens of the same length as the input token gives better results.

## 6.3 Testing the Post-processing techniques

The testing process of the post-processing technique is separated into two parts; one when using the naked words as a look-up dictionary and the other when using the naked PAWs. Each part of the testing process includes testing the finding of the examined token in the trie structure and the other testing the finding of the best similar token to the examined token using the Levenshtein distance. The NWords and NPAWs dictionaries testing results and conclusion are shown in section 6.3.3.

### 6.3.1 Testing the NWords dictionary

The detected tokens are first tested by searching the trie structure of the naked words look-up dictionary. Testing NWords shows that the trie was loaded from the textual data file containing the list of 6 million NWords in 18 seconds on a machine with an Intel processor Core 2 Duo and 4 GB of RAM running Windows Vista 32bit. Table 6.1 shows the number of detected naked words that are missing from the NWords look-up dictionary and separated in each file in the testing dataset. The number of detected words that are missing from the NWords look-up dictionary is 347 NWords which represents 33% of the total detected naked words in the look-up dictionary.

Table 6.1: Post-processing test results of naked words

| Document image file | No. Of Words | Missed NWords from Dictionary | Characters error | | | | |
|---|---|---|---|---|---|---|---|
| | | | Commercial software | HCC Recognizer | Confusion Matrix | Post Processing | Difference |
| FinalTest001.bmp | 34 | 11 | 16 | 55 | 51 | 48 | 3 |
| FinalTest002.bmp | 41 | 15 | 19 | 86 | 87 | 86 | 1 |
| FinalTest003.bmp | 21 | 11 | 36 | 56 | 56 | 53 | 3 |
| FinalTest004.bmp | 24 | 13 | 33 | 61 | 56 | 58 | -2 |
| FinalTest005.bmp | 34 | 20 | 136 | 105 | 103 | 100 | 3 |
| FinalTest006.bmp | 34 | 15 | 82 | 97 | 97 | 91 | 6 |
| FinalTest007.bmp | 18 | 8 | 49 | 48 | 49 | 48 | 1 |
| FinalTest008.bmp | 22 | 14 | 67 | 47 | 46 | 45 | 1 |
| FinalTest009.bmp | 68 | 15 | 34 | 96 | 86 | 91 | -5 |
| FinalTest010.bmp | 63 | 24 | 46 | 89 | 80 | 93 | -13 |
| FinalTest011.bmp | 38 | 8 | 50 | 28 | 27 | 33 | -6 |
| FinalTest012.bmp | 29 | 7 | 12 | 33 | 31 | 33 | -2 |
| FinalTest013.bmp | 68 | 15 | 35 | 102 | 98 | 100 | -2 |
| FinalTest014.bmp | 75 | 20 | 52 | 104 | 108 | 111 | -3 |
| FinalTest015.bmp | 79 | 28 | 62 | 220 | 217 | 219 | -2 |
| FinalTest016.bmp | 23 | 10 | 10 | 56 | 56 | 56 | 0 |
| FinalTest017.bmp | 50 | 21 | 23 | 120 | 121 | 117 | 4 |
| FinalTest018.bmp | 69 | 19 | 33 | 108 | 105 | 107 | -2 |
| FinalTest019.bmp | 56 | 17 | 37 | 87 | 81 | 80 | 1 |
| FinalTest020.bmp | 58 | 16 | 41 | 100 | 104 | 104 | 0 |
| FinalTest021.bmp | 59 | 17 | 49 | 137 | 139 | 143 | -4 |
| FinalTest022.bmp | 43 | 12 | 2 | 49 | 49 | 51 | -2 |
| FinalTest023.bmp | 47 | 11 | 7 | 56 | 54 | 53 | 1 |
| Total | 1053 | 347 | 931 | 1940 | 1901 | 1920 | -19 |

The missed NWords from the look-up dictionary, which are 347 NWords, are then moved to the next step of post-processing. Every missed naked word is examined for its Levenshtein distance for every naked word in the look-up dictionary that has the same word length. The naked words from the look-up dictionary that have a distance of 1 or 2 from the examined naked word are then saved in a list of similar tokens. The list of similar tokens is then sorted based on distance and the number of occurrences of each token in the corpus. The similar naked word with the highest rank in the list is selected as a substitute for the examined naked word. Table 6.1 presents the testing results of applying the Levenshtein distance to the look-up dictionary of naked words for each of the tested dataset files.

### 6.3.2 Testing NPAWs dictionary

The detected tokens are first tested by searching the trie structure of the naked PAWs look-up dictionary. Testing the NPAWs shows that the trie was loaded from the textual data file that contains the list of 14 million NPAWs in 24 seconds on the same machine as in section 6.3.1. Table 6.2 shows the number of detected naked PAWs that are missing from the NPAWs look-up dictionary separated into each file in the testing dataset. There are 247 NPAWs missing which represents 11% of the total detected naked PAWs in the look-up dictionary.

These missed 247 NPAWs are then moved to the next step of post-processing which is similar to that applied to the NWords as shown in section 6.3.1, but replacing the NWords tokens and dictionary with the NPAWs tokens and dictionary. Table 6.2 presents the test results of applying the Levenshtein distance to the look-up dictionary of naked PAWs for each of the testing dataset files.

It was noted that when applying the Levenshtein distance algorithm to the testing dataset some specific files have worse results than other files, for example FinalTest009.bmp, FinalTest010.bmp, FinalTest011.bmp, FinalTest020.bmp and FinalTest021.bmp. The reasons for these bad results vary from having a large number of missing words in the similar list from the look-up dictionary to sometimes longer detected tokens which fail to spot the white spaces between two actual tokens.

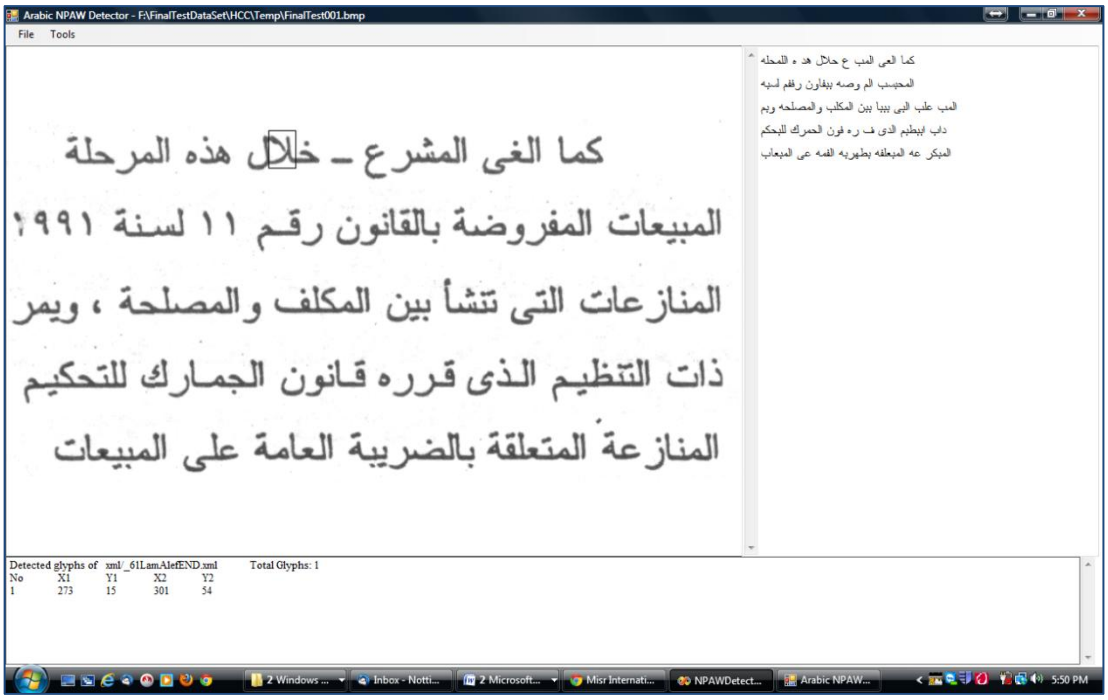
Table 6.2: Post-processing test results of naked PAWs

| Document image file | No. Of PAWs | Missed NPAWs from Dictionary | Characters error | | | | |
|---|---|---|---|---|---|---|---|
| | | | Commercial software | HCC Recognizer | Confusion Matrix | Post Processing | Difference |
| FinalTest001.bmp | 78 | 10 | 16 | 55 | 51 | 49 | 2 |
| FinalTest002.bmp | 89 | 8 | 19 | 86 | 87 | 85 | 2 |
| FinalTest003.bmp | 53 | 10 | 36 | 56 | 56 | 53 | 3 |
| FinalTest004.bmp | 58 | 12 | 33 | 61 | 56 | 55 | 1 |
| FinalTest005.bmp | 68 | 16 | 136 | 105 | 103 | 101 | 2 |
| FinalTest006.bmp | 54 | 15 | 82 | 97 | 97 | 89 | 8 |
| FinalTest007.bmp | 43 | 3 | 49 | 48 | 49 | 46 | 3 |
| FinalTest008.bmp | 54 | 11 | 67 | 47 | 46 | 43 | 3 |
| FinalTest009.bmp | 141 | 2 | 34 | 96 | 86 | 86 | 0 |
| FinalTest010.bmp | 140 | 7 | 46 | 89 | 80 | 82 | -2 |
| FinalTest011.bmp | 79 | 2 | 50 | 28 | 27 | 30 | -3 |
| FinalTest012.bmp | 70 | 1 | 12 | 33 | 31 | 31 | 0 |
| FinalTest013.bmp | 121 | 13 | 35 | 102 | 98 | 97 | 1 |
| FinalTest014.bmp | 147 | 9 | 52 | 104 | 108 | 109 | -1 |
| FinalTest015.bmp | 179 | 32 | 62 | 220 | 217 | 214 | 3 |
| FinalTest016.bmp | 62 | 11 | 10 | 56 | 56 | 55 | 1 |
| FinalTest017.bmp | 108 | 19 | 23 | 120 | 121 | 117 | 4 |
| FinalTest018.bmp | 132 | 12 | 33 | 108 | 105 | 100 | 5 |
| FinalTest019.bmp | 118 | 10 | 37 | 87 | 81 | 76 | 5 |
| FinalTest020.bmp | 120 | 19 | 41 | 100 | 104 | 104 | 0 |
| FinalTest021.bmp | 136 | 15 | 49 | 137 | 139 | 142 | -3 |
| FinalTest022.bmp | 97 | 2 | 2 | 49 | 49 | 51 | -2 |
| FinalTest023.bmp | 104 | 8 | 7 | 56 | 54 | 54 | 0 |
| Total | 2251 | 247 | 931 | 1940 | 1901 | 1869 | 32 |

## 6.3.3 Post-processing test results

The results of applying the post-processing techniques is considered to be the second proof of the assumption in this research regarding the importance of Piece of Arabic Word (PAW) that was firstly explained in section 2.2.3 and tested in section 3.6.1. PAWs are important tokens for the Arabic language and can be a better alternative to using words. The naked token is also an important type of token in the Arabic language, as it shrinks the number of letters from 28 to only 18 (see section 2.2.3).

Figure 6.2 shows the final screen shot of the HCC application result with a document from the testing dataset. The right side shows the output tokens of the HCC application as naked tokens after running the post-processing technique on the detected NPAWs using the NPAWs look-up dictionary.



Arabic NPAW Detector - F:\FinalTestDataSet\HCC\Temp\FinalTest001.bmp

كما الغى المشرع ـ خلال هذه المرحلة
المبيعات المفروضة بالقانون رقم ١١ لسنة ١٩٩١
المنازعات التى تنشأ بين المكلف والمصلحة ، ويمر
ذات التنظيم الذى قرره قانون الجمارك للتحكيم
المنازعة المتعلقة بالضريبة العامة على المبيعات

كما العى المب ع حلال هد ء اللمحله
المحيسب الم وصه بيفاون رفقم لسيه
المب علب البى بيبا بين المكلب والمصلحه ويم
داب ابيطيم الدى فـ ره فون الحمرك للبحكم
المبكر عه المبعلقه بطهربه الفمه عى المبعاب

```
Detected glyphs of   xml/_61LamAlefEND.xml        Total Glyphs: 1
No        X1        Y1        X2        Y2
1         273       15        301       54
```

Figure 6.2: Screenshot of the typical output of the HCC OCR application

The following remarks on applying the post-processing technique support the assumption of the importance of naked PAWs to the Arabic language:

1. The NWords look-up dictionary is faster in loading the trie structure into memory because it is smaller than the NPAWs look-up dictionary. This delay of loading the NPAWs look-up dictionary has little influence on the OCR application as it is loaded once on the start of loading the application.

2. The NWords trie structure size is almost seven times that of the NPAWs trie structure. This guarantees that the NPAWs look up dictionary is more convenient as it occupies less memory and accordingly generates a faster OCR application.

3. The percentage of detected NWords missed from the NWords testing dataset is 33%, while the percentage of detected NPAWs missed from the NPAWs testing dataset is 11%. This means that correcting the tokens recognition error is much smaller in the case of NPAWs rather than that of the NWords.

4. Recognition enhancement using NPAWs is higher than that of NWords. If more advanced post-processing techniques are used, this will justify that NPAWs (as a reference token) are better than NWords.

5. The post-processing stage duration when applying NWords is almost double the time than when applying NPAWs.

## 6.4 Summary

This chapter explains the application of a post-processing technique on recognized tokens. It uses the recognized token and enhances it by applying the confusion matrix in Chapter 5 in order to use them to apply the post-processing technique. The work presented in this chapter also uses the multi-modal Arabic corpus presented in Chapter 3 as a look-up dictionary for the use of post-processing. The testing dataset generated in section 5.1.2 was used in order to test the performance of the post-processing. The main issues explained in this chapter are given below:

1. The use of the naked words and the naked PAWs look-up dictionary with an explanation of why they were used.

2. A full description of the application of the selected post-processing technique. Applying the tail-end trie structure to both the NWords and the NPAWs look-up dictionaries. Use the Levenshtein distance algorithm for string correction with the NWords and NPAWs in order to enhance the recognition rate.

3. Testing the post-processing techniques. Testing the NWords, the NPAWs tokens and look-up dictionaries. The results of testing the NWords, NPAWs and the final findings which indicate that using the NPAWs look-up dictionary enhances the recognition of missed NPAWs by 13%.

# Chapter 7.

# Conclusions and Future work

This chapter explains how the objectives of the research were met and the methods followed in order to meet these objectives. Future work and enhancement of the research is explained in detail for each of the main topics of the research. The conclusions of this research are explained. Closing remarks are introduced in order to summarize the overall contributions of this research.

The objectives and contributions of the research were introduced and explained briefly in Chapter 1 with a definition of how they were applied and met. The multi-modal Arabic corpus generated in Chapter 3 is used to show what improvements and maintenance are needed in order to continually support research studies in the Arabic language. Chapter 4 includes the experimental work of the HCC approach and forms the main contribution of this research. An explanation was provided on how the HCC approach can be further enhanced and improved in order to generate better classifiers. The HCC tokens recognizer in Chapter 5 shows how future enhancements can be made to enhance the accuracy of any Arabic OCR application. The post-processing technique applied in Chapter 6 was discussed regarding its durability and usability for further improving this type of research.

## 7.1 Meeting the objectives

The main concern of this research is printed Arabic character recognition. The aims and objectives of this research were introduced briefly in section 1.2.1 and were defined specifically in section 1.2.2. These aims and objectives were investigated and achieved during the progress of this research. The objectives were presented in the form of research questions. The following are the objectives with an explanation of how they were met:

1. *What are the advantages and disadvantages of the established way of solving the problem of printed Arabic OCR?*

The established way of solving the problem of Arabic character recognition has many advantages and disadvantages as shown in Chapter 2. The main advantages are being able to follow almost the same procedures of solving the OCR problem as for any other language but with some modifications to fit the case of the Arabic language.

The main disadvantage is that the Arabic language has some features that are different from other languages (see section 2.2) which make it different in having a connected script. The relative uniqueness of the language makes the character segmentation algorithm (see section 2.5) almost mandatory, which reduces the recognition time and accuracy (Abdelazim 2006). Also, one of the disadvantages is the low recognition rate compared to Latin alphabet languages. Improving the recognition rate of the Arabic OCR application is a must and can be achieved by increasing the research studies in all topics related to Arabic OCR and to try new non-traditional methods in order to solve it.

The testing results when comparing the proposed HCC approach application against commercial software show that the HCC achieved a glyph recognition accuracy of 87% and the commercial software 85%. Moreover, the average recognition time for HCC is 14.7 Seconds while is 15.8 in the commercial software.

The results of this research reached a conclusion regarding the established way of solving the Arabic OCR problem, which indicates that it is possible to develop a full Arabic OCR application without some steps in the pre-processing and recognition phases, and also without character segmentation stages. This application is slightly more accurate and faster than that used the established way which also indicates that the established way of solving the Arabic OCR problem might not be the optimum one.

2. *To what extent are the steps in the pre-processing and recognition stages essential and can these steps be eliminated totally or partially?*

The pre-processing and recognition stages are usually mandatory stages in any OCR application or research, for each step from the preparation of the document image to feature extraction stage. No Arabic OCR application or research method has been previously developed without these steps of the pre-processing and recognition stages (Binarization & Noise Removal, Thinning, Slant & Skew Correction, Line Finding, Lines & Words Segmentation and Word Normalization). Interestingly no OCR application for any other language has been developed without the previously mentioned steps (Cheriet, Kharma et al. 2007).

During this research, a complete Arabic OCR application was generated without the needs of the previously mentioned steps i.e. the HCC approach in this research totally eliminates steps from the pre-processing and recognition stages. The explanation was given in section 4.2. Importantly these stages are key to increasing or decreasing the recognition accuracy of the OCR application and to affecting the performance of the OCR application, especially when complicated algorithms are used. Note, the HCC approach can be applied to any other language than Arabic and will also eliminate the pre-processing stage.

3. *Can Arabic printed text be recognized without the character segmentation phase?*

The character segmentation phase is usually considered to be a mandatory phase in any Arabic OCR application or research although there are some recorded trials that have reported to have character segmentation free recognition as shown in section 2.5.5. However, these are still limited to a limited number of words and fonts (Abdelazim 2006). No Arabic OCR application has been developed without a character segmentation phase (Abdelazim 2006) until now.

The character segmentation phase also has different approaches available and it usually requires a significant effort from researchers or developers to select or invent the appropriate approaches. Importantly this phase is “bottle neck” for the recognition accuracy of Arabic OCR applications in addition to slowing down the performance of the OCR application.

Along with this research, a complete Arabic OCR application was generated without the need for the character segmentation stage as shown in Chapter 5. The HCC approach is trained to pick out the glyph image from the document image which is considered to be a true character segmentation-free approach. The explanation of how this was applied without the character segmentation was shown in section 4.2.

4. *If some steps from the pre-processing and recognition stages in additional to character segmentation are eliminated, then can a complete Arabic OCR application be developed?*

This objective was achieved during this research and proved that it is both applicable and doable. The HCC approach applied in this research generated glyph classifiers as shown in section 4.6. The approaches applied in section 5.2 and section 5.3 used the classifiers generated in section 4.6 to generate a full HCC Arabic recognizer with a confusion matrix enhancement as shown in section 5.4. A post-processing technique was applied in Chapter 6 to enhance the recognition accuracy of the HCC recognizer. This indicates that the HCC approach can generate a full Arabic OCR application without either steps from pre-processing and recognition phases or a character segmentation stage.

5. *Can PAWs and NPAWs be considered as the smallest Arabic tokens to replace words and naked words in Arabic OCR applications?*

The PAW is considered to be the smallest token in the Arabic language as it is the smallest connected token in the language (AbdelRaouf, Higgins et al. 2010). The statistical analysis information shown in section 3.5 and testing of the corpus shown in section 3.6 shows that the PAW is a convincing token that can be used in many cases instead of the normal small token, which is the word.

The post-processing technique shown in Chapter 6 and the test results in applying that technique shown in section 6.3, confirm that the PAW is better than the word. The reasons for this are, first, the size of the naked word look-up dictionary is almost seven times the size of the naked PAW look-up dictionary. Secondly, the elapsed time in applying the post-processing stage with naked words is almost double that of naked PAWs. Finally, the results of using a naked PAW look-up dictionary is better that that of a naked word look-up dictionary. This proves that the PAW is a much better token to be used with the Arabic language than the word.

6. *Are the available Arabic corpora sufficient to cover Arabic OCR research?*

The available Arabic corpora are still very limited (Hamada 2007) and do not cover the growing need for research studies related to the Arabic language. The available corpora tend not to be continually updated and enhanced by adding new lists of words or more tokens analysis as shown in section 2.7.4. The available corpora are limited to only one type of token which is words.

The MMAC contains 6 million words and is complemented by an application that is kept updated and maintained by adding new words. The MMAC also contains four different types of tokens with their full statistical analysis and validity testing. During and after this research and since the MMAC corpus was published it has been used by many researchers in their research and was also in many commercial companies. This indicates the efficiency of MMAC as a robust Arabic corpus that facilitates the research and commercial requirements.

## 7.2 Future work

Arabic character recognition still needs improvement. During this research many future enhancements were identified which also improved the research results. These future enhancements are separated into two parts; one is general, which included new topics that needed to be covered in order to enhance this research. The other was enhancements related to each of the sections covered in this research. The future enhancements are listed and explained in the following list:

### 7.2.1 General topics

The following topic is required to be included in future enhancements in order to improve the results of this research.

*Page layout analysis* is a document processing technique used to determine the format of a page. The document spectrum approach (O'Gorman 1993), or docstrum method can be adapted to use the list of the detected glyphs in order to calculate the layout structure of the document image. Areas that have very low glyph detection could be considered a neglected area, for example graphics or other languages. The output of this method can define document image areas for the paragraphs, lines, words and PAWs.

### 7.2.2 Arabic corpus

The MMAC future enhancements should be unlimited and the corpus must be kept up-to-date. It is important for the MMAC to be a bench mark corpus for testing all OCR research studies and applications. MMAC can be enhanced in the following ways:

- Continually adding lists of words to the corpus to increase its size and variety. These lists must be truthed first as shown in section 3.4. The other tokens (PAWs, NPAWs and NWords) are generated from the list of words using the application provided. The provided application then modifies the analysis and size of the tokens.

- Including morphological information of the words to their corpus information (section 2.7.4.4). The ground truth XML file shown in section 3.1.6 must also include this morphological information.

- It is important to increase the number of real scanned images in the corpus that were presented in section 3.3.4; increasing these document images will give a greater variety to the corpus.

- A new type of statistical analysis (see section 3.5) is recommended; the word frequency pairs. The word frequency pairs store, for each token, the list of the next tokens that usually occur with this token along with frequency data. This type of analysis is used in the post-processing techniques.

### 7.2.3 HCC glyphs recognition

Future enhancements for the glyphs recognition using Haar Cascade Classifier can be made by keeping the glyphs classifiers continually updated and enhanced by use more glyphs training dataset which will lead to an increase in the recognition rate. Glyphs recognition can be enhanced by using the following sequence:

- Improve the application of the HCC approach that was generated by OpenCV and is an open source application. This enhancement makes it best fit the character recognition or to have a new version of the application for the character recognition. It was very clear from the theoretical background of the approach shown in section 2.6.7 that it can be modified to better suit character recognition.

- Enhance the training of the glyphs that have a small number of positive or negative images by adding new document images in order to increase this small number. This enhancement of a selective number of glyphs will influence the overall recognition accuracy of the HCC approach.

- Generate classifiers for both the Hindu and Arabic numerals for example (٣ ٢ ١) and (1 2 3) and for the Arabic special characters such as (’ ؛ ‘ ؟ ،). These glyphs are usually used in most of the Arabic documents.

### 7.2.4 HCC tokens recognizer

The HCC tokens recognition future enhancements are numerous as the algorithms used in this part of the research are very simple and serve only to confirm the approach is viable. Enhancements here can significantly improve the recognition rate of the HCC tokens recognizer.

- Although the HCC glyphs classifier can detect skewed glyph images, these were not met in the tokens testing dataset shown in section 5.1.2. Document skew angle can be detected using the same method as for extracting text lines which can be defined with the skew angle and calculates the horizontal histogram based on the skew angle.

- The tokens recognition can be significantly enhanced through analyzing the baseline of the text line in order to define the locations of tokens in the text line which enables the enhancement of parsing the text lines' contents into tokens. The gap detector described in (Gillies, Erlandson et al. 1999) is a better algorithm for gap analysis in order to better parse the detected glyphs into tokens and uses an artificial intelligence technique. This approach serves to extract the gaps from the document images. These gaps can be white spaces or gaps between lines. The gap detector depends upon analyzing the list of detected glyphs from the glyph detector.

- Repeated enhancement of the confusion matrix. This enhancement can be achieved by regenerating the confusion matrix with any and every modification occurrence to MMAC.

- Apply the confusion matrix technique at the level of NPAWs. The overall number of NPAWs in MMAC is 32,804. This requires heavy manual work in defining the rectangles containing the NPAWs, but when this part of work is completed it can significantly enhance the recognition accuracy of the HCC approach.

### 7.2.5 Post-processing

Although a big effort was made to prepare the corpus, the post-processing algorithms were not a major area of research for this work so they are simple and limited. Many improvements could be made.

- The look-up dictionary could be applied in a more intelligent way to reject the extracted tokens that rarely occur in the corpus.

- NPAWs frequency pairs could be applied as another measure to enhance the selection of the proper "similar" tokens to the examined tokens shown in section 2.7.3.2. The NPAWs' frequency pairs study the relation between the NPAWs as explains how often the two contiguous NPAWs occur. It can also study the position of the NPAW in the word. The definition of this position is whether the NPAW is at the start, middle or end of the word.

- Levenshtein distance algorithm can be enhanced by including a merge and split editing feature (Schulz and Mihov 2002). Recognized tokens that are long, which may be two tokens concatenated during parsing the detected tokens will benefit from this enhancement.

## 7.3 Conclusions

The main conclusion of this research is that the Haar Cascade Classifier (HCC) approach is applicable to Arabic printed character recognition. This approach eliminates steps of the pre-processing and recognition stages in additional to character segmentation phase. It can generate a full Arabic OCR application that is fast when compared to commercial software, as the average recognition time for HCC is 14.7 Seconds while is 15.8 in the commercial software.

The languages that use Arabic alphabets but not the Arabic language, such as Pashto, can also use this approach for printed character recognition. These languages can, not only apply the approach, but can also benefit from this research, especially the glyph classifiers generated in section 4.6. The HCC approach could also be applied to cursive, handwritten characters. Although the glyphs training process of the HCC approach for handwritten characters is difficult, but it may bring an improvement to the handwritten recognition accuracy.

The ideogram languages such as Chinese and Japanese, which are totally symbolic languages could also benefit from the HCC approach. In this case it might be the only approach for the detection, or a supplementary approach for first level of classification of the characters. The HCC approach could be applied to non-cursive languages such as English. The same methodology is applied by focusing on the features of a definite glyph and then detecting that glyph. The HCC approach is beneficial in this case for the sake of eliminating steps of the pre-processing and recognition stages.

MMAC is beneficial for researchers and developers in the field of Arabic character recognition. It includes a guide for Arabic OCR development (see section 3.3) to be followed by researchers and developers during the testing of any Arabic OCR application. MMAC is supported by an application that guarantees that this corpus will be continually updated by adding new lists of words. MMAC makes innovative use of new tokens (PAW, NPAW and NWord) that have rarely been used before and whose importance was not recognized. These new tokens have significance for linguistics as well as OCR research and applications.

The main focus of the research was the HCC work at the beginning. That is processing the document images so that steps of the pre-processing and recognition stages in additional to the character segmentation stage were eliminated. The other main interest was the linguistic research at the end. The system which connects these two aspects for extracting PAWs and words from the text was not a major part of the work so was done simply to give a complete system. The percentage recognition of individual glyphs is 87% while the commercial software percentage recognition is 85%. This recognition rate was achieved by simple optimisation the parameters of the system. A move through optimization which would require no additional insight, only time and effort might enhance the system. The post-processing which would require some thought to apply the linguistics rules better could easily achieve more improvement. The connecting work which actually gives a reduction in recognition rate at the moment, (as it introduces errors where we get the PAWs spacing and words spacing wrong and so on) could be vastly improved so possibly give up accuracy gain compared to a current 15% loss. This could give a new Arabic OCR application with higher accuracy when compared with the commercial software.

## 7.4 Closing remarks

Character recognition is a wide research field which includes an extensive variety of important topics. Although character recognition began five decades ago, there is still much work to be done in this area and there are always new approaches and algorithms that can used in order to enhance character recognition accuracy. Arabic character recognition started many years after the beginnings of Latin alphabet character recognition and, therefore, much more effort needs to be invested in investigating better techniques. More research to be carried out in the area of Arabic character recognition in order to achieve better accuracy.

This thesis is the outcome of a research study in Arabic printed character recognition. The research applies a novel approach (HCC) to Arabic character recognition which has not been used before in any character recognition, and especially in the Arabic language. The proposed HCC approach implements a full Arabic OCR application without the retarding steps in the pre-processing and recognition stages or the character segmentation phase. A novel corpus was presented which included new types of tokens which provided evidence of significant usefulness when applied in conjunction with post-processing techniques.

# References

Abandah, G. A., K. S. Younis, et al. (2008). Handwritten Arabic character recognition using multiple classifiers based on letter form. the Fifth IASTED International Conference on Signal Processing, Pattern Recognition & Applications (SPPRA 2008), Innsbruck, Austria.

Abbes, R., J. Dichy, et al. (2004). The Architecture of a Standard Arabic Lexical Database. Some Figures, Ratios and Categories from the DIINAR.1 Source Program. Workshop of Computational Approaches to Arabic Script-based Languages, Geneva, Switzerland, COLING.

ABBYY. (2012). "ABBYY FineReader 11." Retrieved 4/05/2012, from http://finereader.abbyy.com/key_features/#-Section_2;-Section_3;Section_9;.

Abdelazim, H. Y. (2006). Recent Trends in Arabic Character Recognition. The sixth Conference on Language Engineering, Cairo - Egypt, The Egyptian Society of Language Engineering.

AbdelRaouf, A., C. Higgins, et al. (2008). A Database for Arabic printed character recognition. The International Conference on Image Analysis and Recognition-ICIAR2008, Póvoa de Varzim, Portugal, Springer Lecture Notes in Computer Science (LNCS) series.

AbdelRaouf, A., C. Higgins, et al. (2010). "Building a Multi-Modal Arabic Corpus (MMAC)." The International Journal of Document Analysis and Recognition (IJDAR) **13**(4): 285-302.

Adolf, F. (2003) "How-to build a cascade of boosted classifiers based on Haar-like features." **Volume**, DOI:

Al-Badr, B. and R. M. Haralick (1998). "A segmentation-free approach to text recognition with application to Arabic text." International Journal on Document Analysis and Recognition **1**(3): 147-166.

Al-Kharashi, I. A. and M. W. Evens (1994). "Comparing words, stems, and roots as index terms in an Arabic Information Retrieval System." Journal of the American Society for Information Science **45**(8): 548 - 560.

Al-Ma'adeed, S., D. Elliman, et al. (2002). A data base for Arabic handwritten text recognition research. Eighth International Workshop on Frontiers in Handwriting Recognition, Ontario, Canada.

Al-Shalabi, R. and M. Evens (1998). A Computational Morphology System for Arabic. Workshop on Computational Approaches to Semitic Languages COLING-ACL98, Montreal.

Al-Shalabi, R. and G. Kanaan (2004). "Constructing An Automatic Lexicon for Arabic Language." International Journal of Computing & Information Sciences **2**(2): 114-128.

Alansary, S., M. Nagi, et al. (2007). Building an International Corpus of Arabic (ICA): Progress of Compilation Stage. The Seventh Conference on Language Engineering, Cairo - Egypt, The Egyptian Society of Language Engineering.

Aljlayl, M. and O. Frieder (2002). On Arabic search: improving the retrieval effectiveness via a light stemming approach. The Eleventh International Conference on Information and knowledge Management, McLean, Virginia, USA, Conference on Information and Knowledge Management archive.

Alma'adeed, S., C. Higgens, et al. (2002). Recognition of Off-Line Handwritten Arabic Words Using Hidden Markov Model Approach. the 16 th International Conference on Pattern Recognition (ICPR'02), Quebec, Canada.

Amin, A. (1997). Off line Arabic character recognition - a survey. the Fourth International Conference on Document Analysis and Recognition, Ulm, Germany.

Ananthakrishnan, S., S. Bangalore, et al. (2005). Automatic diacritization of arabic transcripts for automatic speech recognition. International Conference On Natural Language Processing, Kanpur, India.

arabo.com. (2005). "Arabo Arab Search Engine & Dictionary." Retrieved 12-01-07, 2006, from http://www.arabo.com/.

Baird, H. S. (1987). The skew angle of printed documents. Conference of the Society of Photographic Scientists and Engineers.

Barros, R. C., M. a. P. Basgalupp, et al. (2011). "A Survey of Evolutionary Algorithms for Decision-Tree Induction." IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews **pp**(99): 1-22.

Bates, M. (1995). "Models of natural language understanding." PNAS **92**(22): 9977-9982.

Beebe, N. H. F. (1990). "Character Set Encoding." TUGboat **11**(2): 171-175.

Beesley, K. R. (1996). Arabic Finite-State Morphological Analysis and Generation. 16th International Conference on Computational Linguistics COLING, Copenhagen.

Bishop, C. M. (2006). Pattern recognition and machine learning, New York : Springer, 2006.

Bowyer, K. W., K. Chang, et al. (2005). "A survey of approaches and challenges in 3D and multi-modal 3D + 2D face recognition." Computer Vision and Image Understanding **101**(1): 1-15.

Bowyer, K. W., K. Hollingsworth, et al. (2008). "Image understanding for iris biometrics: A survey." Computer Vision and Image Understanding **110**(2): 281–307.

Box, G. E. P. and M. E. Muller (1958). "A Note on the Generation of Random Normal Deviates." The Annals of Mathematical Statistics **29**(2): 610-611.

Bradski, G. and A. Kaehler (2008). Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, Inc.

Bradski, G. R. and V. Pisarevsky (2000). Intel's Computer Vision Library: Applications in Calibration, Stereo, Segmentation, Tracking, Gesture, Face and Object Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00), SC USA.

Breuel, T. M. (2002). Robust Least Square Baseline Finding using a Branch and Bound Algorithm. Document Recognition and Retrieval VIII, SPIE.

Breuel, T. M. (2002). Two Geometric Algorithms for Layout Analysis. In Workshop on Document Analysis Systems.

Buckwalter, T. (2002). "Arabic Word Frequency Counts." from http://www.qamus.org/wordlist.htm.

Buckwalter, T. (2002). "Buckwalter Arabic Transliteration." Retrieved 28 January 2007, 2007, from http://www.qamus.org/transliteration.htm.

Buckwalter, T. (2004). Issues in Arabic Orthography and Morphology Analysis. The 20th International Conference on Computational Linguistics, COLING 2004, Geneva, Switzerland.

Bushofa, B. and M. Spann (1997). "Segmentation and recognition of Arabic characters by structural classification " Image and Vision Computing **15**(3): 167-179.

Chang, Y., D. Chen, et al. (2009). "An image-based automatic Arabic translation system." Pattern Recognition **42**(9): 2127-2134.

Cheriet, M., N. Kharma, et al. (2007). Character Recognition Systems: A Guide for Students and Practitioners, Wiley.

Consortium, T. U. (2003). The Unicode Consortium. The Unicode Standard, Version 4.1.0, Boston, MA, Addison-Wesley**:** 195-206.

contributors, W. (2006, 20 December 2006). "Arabic chat alphabet. From Wikipedia, the free encyclopedia." Retrieved 27 January 2007, 2007, from http://en.wikipedia.org/wiki/Arabic_Chat_Alphabet.

contributors, W. (2006, 11 November 2006 20:02 UTC ). "Code page. From Wikipedia, the free encyclopedia." Retrieved 22 January 2007 17:45 UTC 2007, from http://en.wikipedia.org/w/index.php?title=Code_page&oldid=87192444.

contributors, W. (2006, 23 January 2007). "Romanization of Arabic. From Wikipedia, the free encyclopedia." Retrieved 27 January 2007, 2007, from http://en.wikipedia.org/wiki/Arabic_transliteration.

Corpus, T. B. N. (2007). The British National Corpus (XML Edition).

Crow, F. C. (1984). "Summed-Area Tables for Texture Mapping." SIGGRAPH Computer Graphics **18**(3): 207-212.

D.J.Ittner and H. S. Baird (1993). Language-free layout analysis. The Second International Conference on Document Analysis and Recognition (ICDAR'93).

Dash, N. S. and B. B. Chaudhuri (2001). Why Do We Need To Develop Corpora In Indian Languages? International Working Conference on Sharing Capability in Localisation and Human Language Technologies (SCALLA-2001), Bangalore.

Davies, M. (1990-present). "The Corpus Of Contemporary American English (COCA), 410+ million words." from http://www.americancorpus.org.

Dynamics, N. (2011). "VERUS™ Professional." Retrieved 27/07/2011, from http://www.novodynamics.com/verus_pro.htm.

Dynamics, N. (2012). "NovoVerus." Retrieved 04/05/2012, from http://www.novodynamics.com/novoverus/.

Education, P. (2009). "Most Popular Language in the world by Number of Speakers." Retrieved 14/08/2011, from http://www.infoplease.com/ipa/A0775272.html.

Elliman, D. (2001). TIF2VEC, An Algorithm for Arc Segmentation in Engineering Drawings. Fourth International Workshop on Graphics Recognition Algorithms and Applications (GREC '01).

Elliman, D. and I. Lancaster (1990). "A review of segmentation and contextual analysis techniques for text recognition." Pattern Recognition **23**(3-4): 337-346.

Fahmy, M. M. M. and S. A. Ali (2001). "Automatic Recognition Of Handwritten Arabic Characters Using Their Geometrical Features." Journal of Studies in Informatics and Control with Emphasis on Useful Applications of Advanced Technology **10**(2): 81-98.

Ford, D. M. and C. A. Higgins (1990). "A tree-based dictionary search technique and comparison with N-GRAM letter graph reduction." Computer Processing of Handwriting: 291-312.

Freund, Y. and R. E. Schapire (1996). Experiments with a New Boosting Algorithm. The thirteenth International Conference on Machine Learning, San Francisco, USA.

Gillies, A., E. Erlandson, et al. (1999). Arabic Text Recognition System. Proceedings of the Symposium on Document Image Understanding Technology.

Gonzalez, R. C. and R. E. Woods (2007). Digital Image Processing, Prentice Hall.

Govindan, V. K. and A. P. Shivaprasad (1990). "Character Recognition - A Review." Pattern Recognition **23**(7): 671-683.

Gu, B., F. Hu, et al. (2001). Modelling Classification Performance for Large Data Sets, An Empirical Study. Advances in Web-Age Information Management: Second International Conference, WAIM 2001, Xi'an, China.

Haar, A. (1910). "Zur Theorie der orthogonalen Funktionensysteme." Mathematische Annalen **69** (3): 331-371.

Hamada, S. (2007). نحو منهج مقترح لصناعه المدونات اللغويه. The Seventh Conference on Language Engineering, Cairo - Egypt, The Egyptian Society of Language Engineering.

Hartley, R. T. and K. Crumpton (1999). Quality of OCR for degraded text images. The fourth ACM conference on Digital libraries, California, United States.

Harty, R. and C. Ghaddar (2004). "Arabic Text Recognition." The International Arab Journal of Information Technology **1**(2): 156-163.

He, Z., T. Tan, et al. (2009). "Toward Accurate and Fast Iris Segmentation for Iris Biometrics." IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(9): 1670 - 1684.

Hulden, M. (2009). "Fast approximate string matching with finite automata." Procesamiento del Lenguaje Natural **43**: 57-64.

Hull, J. J. (1998). Document image skew detection: Survey and annotated bibliography. Document Analysis Systems II. Word Scientific**:** 40-64.

Hyams, D. G. (2005). CurveExpert 1.3, A comprehensive curve fitting system for Windows.

Inenaga, S., H. Hoshino, et al. (2005). "On-line construction of compact directed acyclic word graphs." Discrete Applied Mathematics **146**: 156 – 179.

Intel (2001). Open Source Computer Vision Library - OpenCV.

IRIS (2004). Readiris Pro 10.

IRIS. (2011). "Readiris 12 Pro." Retrieved 27/07.2011, from http://www.irislink.com/c2-1684-225/Readiris-12-for-Windows.aspx.

Jafari, R., H. Noshadi, et al. (2006). "Adaptive Electrocardiogram Feature Extraction on Distributed Embedded Systems." IEEE Transactions on Parallel and Distributed Systems **17**(8): 797-807.

Jianzhuang, L., L. Wenqing, et al. (1991). Automatic thresholding of gray-level pictures using two-dimension Otsu method. International Conference on Circuits and Systems, China.

Jomma, H. D., M. A. Ismail, et al. (2006). An Efficient Arabic Morphology Analysis Algorithm. The Sixth Conference on Language Engineering, Cairo, Egypt, The Egyptian Society of Language Engineering.

Kanoun, S., A. M. Alimi, et al. (2005). Affixal Approach for Arabic Decomposable Vocabulary Recognition: A Validation on Printed Word in Only One Font. The Eight International Conference on Document Analysis and Recognition (ICDAR'05), Seoul, Korea.

Kanungo, T. and P. Resnik (1999). The Bible, Truth, and Multilingual OCR Evaluation. SPIE Conference on Document Recognition and Retrieval VI, San Jose, CA.

Kasinski, A. and A. Schmidt (2010). "The architecture and performance of the face and eyes detection system based on the Haar cascade classifiers." Pattern Analysis and Applications **13**(2): 197-211.

Khorsheed, M. S. (2002). "Off-Line Arabic Character Recognition – A Review." Pattern Analysis & Applications **5**(1): 31-45.

Kilgarriff, A. (1997). Using word frequency lists to measure corpus homogeneity and similarity between corpora. 5th ACL workshop on very large corpora, Beijing and Hong Kong.

Knight, D., S. Bayoumi, et al. (2006). Beyond the text: building and analysing multi-modal corpora. 2nd International Conference on E-Social Science, Manchester, UK.

Kohavi, R. and F. Provost (1998). "Glossary of Terms. Special Issue on Applications of Machine Learning and the Knowledge Discovery Process." Machine Learning **30**: 271-274.

Kučera, H. and W. N. Francis (1967). "Computational Analysis of Present-Day American English." International Journal of American Linguistics **35**(1): 71-75.

Kumar, A. and A. Passi (2010). "Comparison and combination of iris matchers for reliable personal authentication." Pattern Recognition **43**(3): 1016-1026.

Kurt, Z., H. I. Turkmen, et al. (2009). Linear Discriminant Analysis in Ottoman Alphabet Character Recognition. The European Computing Conference, Tbilisi, Georgia.

Larkey, L. S., L. Ballesteros, et al. (2002). Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis. 25th International Conference on Research and Development in Information Retrieval (SIGIR).

Leea, C. H. and T. Kanungob (2003). "The architecture of TrueViz: a groundTRUth=metadata editing and VIsualiZing ToolKit." Pattern Recognition **36**(3): 811 – 825.

Lehal1, G. S., C. Singh, et al. (2001). A shape based post processor for Gurmukhi OCR. the Sixth International Conference on Document Analysis and Recognition (ICDAR’01), Seattle, WA , USA.

Levenshtein, V. I. (1966). "Binary codes capable of correcting deletions, insertions, and reversals." Soviet Physics Doklady **10**(8): 707-710.

Lienhart, R., A. Kuranov, et al. (2002). Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. 25th Pattern Recognition Symposium (DAGM03), Madgeburg, Germany.

Lienhart, R. and J. Maydt (2002). An Extended Set of Haar-like Features for Rapid Object Detection. IEEE International Conference of Image Processing (ICIP 2002), New York, USA.

Lorigo, L. M. and V. Govindaraju (May, 2006). "Offline Arabic Handwriting Recognition: A Survey." IEEE Transactions on Pattern Analysis and Machine Intelligence **28**(5): 712-724.

Ltd, S. P. (2006). WebZIP 7.0.

Lu, Z., I. Bazzi, et al. (1999). A Robust, Language-Independent OCR System. 27th AIPR Workshop: Advances in Computer-Assisted Recognition (SPIE 3584), Washington DC, USA.

Maddouri, S. S., H. Amiri, et al. (2000). Local Normalization Towards Global Recognition of Arabic Handwritten Script. DAS 2000.

Maltoni, D. and R. Cappelli (2009). "Advances in fingerprint modeling." Image and Vision Computing **27**(3): 258-268.

Marukawa, K., T. Hu, et al. (1997). "Document retrieval tolerating character recognition errors—evaluation and application." Pattern Recognition **30**(8): 1361-1371.

Mashali, S., A. Mahmoud, et al. (2005). Arabic OCR Database Development. The Fifth Conference on Language Engineering, Cairo, Egypt.

Maynard, D., V. Tablan, et al. (2002). "Architectural Elements of Language Engineering Robustness." Journal of Natural Language Engineering – Special Issue on Robust Methods in Analysis of Natural Language Data: 1-20.

Mehran, R., H. Pirsiavash, et al. (2005). A Front-end OCR for Omni-font Persian/Arabic Cursive Printed Documents. Digital Image Computing: Techniques and Applications (DICTA'05), Cairns, Australia.

Messom, C. and A. Barczak (2006). Fast and Efficient Rotated Haar-like Features using Rotated Integral Images. Australian Conference on Robotics and Automation (ACRA2006).

Mian, A. S., M. Bennamoun, et al. (2007). "An Efficient Multimodal 2D-3D Hybrid Approach to Automatic Face Recognition." IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(11): 1927 - 1943.

Mihov, S. and K. U. Schulz (2004). "Fast Approximate Search in Large Dictionaries." Computational Linguistics journal **30**(4): 451-477.

Miyoshi, T., T. Nagasaki, et al. (2009). Character Normalization Methods Using Moments of Gradient Features and Normalization Cooperated Feature Extraction. Chinese Conference on Pattern Recognition, 2009 (CCPR 2009) Nanjing.

Mohan, b. A., C. Papageorgiou, et al. (2001). "Example-Based Object Detection in Images by Components." IEEE Transactions on Pattern Analysis and Machine Intelligence **23**(4): 349-361.

Najoua, B. A. and E. Noureddine (1995). A Robust Approach for Arabic Printed Character Segmentation. Third International Conference on Document Analysis and Recognition (ICDAR'95), Montreal, Canada

Nations, U. (2007). "INDEXES : United Nations Documentation." 2007, from http://www.un.org/Depts/dhl/resguide/itp.htm.

Navarro, G. (2001). "A Guided Tour to Approximate String Matching." ACM Computing Surveys (CSUR) **33**(1): 31-88.

Network, T. M. (2012). "2012 Best OCR Software Comparisons and Reviews." from http://ocr-software-review.toptenreviews.com/.

newspaper, A.-H. (2002). Al-Hayat Arabic data set, Al-Hayat newspaper, University of Essex, in collaboration with the Open University.

newspaper, A.-N. (2000). An-Nahar text corpus, An-Nahar newspaper.

O'Gorman, L. (1993). "The Document Spectrum for Page Layout Analysis." IEEE Transactions on Pattern Analysis and Machine Intelligence **15**(11): 1162-1173.

O'Gorman, L. (1998). "An overview of fingerprint verification technologies." Information Security Technical Report **3**(1): 21-32.

O'Gorman, L. (1993). "The Document Spectrum for Page Layout Analysis." IEEE TRANSACTIONS ON PA'ITERN ANALYSIS AND MACHINE INTELLIGENCE **15**(11): 1162-1173.

Ohta, M., A. Takasu, et al. (1997). Retrieval Methods for English-Text with Missrecognized OCR Characters. Fourth International Conference on Document Analysis and Recognition.

Oommen, B. J. and G. Badr (2007). "Breadth-first search strategies for trie-based syntactic pattern recognition." Pattern Analysis & Applications **10**(1): 1-13.

OpenCV (2002) "Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features." OpenCV haartraining Tutorial **Volume**, DOI:

Palfreyman, D. and M. a. Khalil (2003). ""A Funky Language for Teenzz to Use": Representing Gulf Arabic in Instant Messaging " Journal of Computer-Mediated Communication **9**(1).

Papageorgiou, C. P., M. Oren, et al. (1998). A General Framework for Object Detection. International Conference on Computer Vision.

Parker, R., D. Graff, et al. (2009). Arabic Gigaword Fourth Edition. Philadelphia, Linguistic Data Consortium, University of Pennsylvania.

Pechwitz, M., S. S. Maddouri, et al. (2002). IFN/ENIT - Database of handwritten Arabic words. the 7th Colloque International Francophone sur l'Ecrit et le Document, CIFED 2002, Hammamet, Tunisia.

Plamondon, R. and S. N. Srihari (2000). "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey." IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(1): 63-84.

Rahman, A. F. R. and M. C. Fairhurst (2003). "Multiple classifier decision combination strategies for character recognition: A review." International Journal on Document Analysis and Recognition **5**(4): 166–194.

Reffle, U. (2011). "Efficiently generating correction suggestions for garbled tokens of historical language." Natural Language Engineering **17**(2): 265–282.

Rogati, M., S. McCarley, et al. (2003). Unsupervised learning of Arabic stemming using a parallel corpus. The 41st Annual Meeting of the Association for Computational Linguistics (ACL), Sapporo, Japan.

Sarfraz, M., S. N. Nawaz, et al. (2003). Offline Arabic text recognition system. International Conference on Geometric Modeling and Graphics (GMAG'03).

ScanStore, N.-. (2011). "OmniPage Professional 18." Retrieved 27/07/2011, from http://www.scanstore.com/Scanning_Software/default.asp?ITEM_ID=16239.

Schapire, R. E. (2002). The Boosting Approach to Machine Learning, An Overview. MSRI Workshop on Nonlinear Estimation and Classification, 2002, Berkeley, CA, USA.

Schulz, K. U. and S. Mihov (2002). "Fast string correction with Levenshtein automata." International Journal on Document Analysis and Recognition **5**(1): 67-85.

Senior, A. (1992). Off-line handwriting recognition: A review and experiments, Cambridge University, Engineering Department.

Seo, N. (2008) "Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features)." **Volume**, DOI:

Shafait, F., Adnan-ul-Hasan, et al. (2006). Layout Analysis of Urdu Document Images. 10th IEEE Int. Multi-topic Conference, INMIC'06, Islamabad, Pakistan.

Shanthi, N. and K. Duraiswamy (2010). "A novel SVM-based handwritten Tamil character recognition system." Pattern Analysis & Applications **13**(2): 173–180.

Simon, A., J.-C. Pret, et al. (1997). "A fast algorithm for bottom-up document layout analysis." IEEE Transactions on Pattern Analysis and Machine Intelligence **19**(3): 273-277.

Simple Software, s. s. f. d. m. (2012). "OCR Software Guide." from http://www.simpleocr.com/OCR_Software_Guide.asp.

Slavik, P. and V. Govindaraju (2001). "Equivalence of Different Methods for Slant and Skew Corrections in Word Recognition Applications." IEEE Transactions on Pattern Analysis and Machine Intelligence **23**(3).

Slimane, F., R. Ingold, et al. (2009). A New Arabic Printed Text Image Database and Evaluation Protocols. 10th International Conference on Document Analysis and Recognition, Barcelona, Spain.

Software, S. (2011). "OCR." Retrieved 27/07/2011, from http://www.sakhr.com/ocr.aspx.

Sonka, M., V. Hlavac, et al. (1998). Image Processing: Analysis and Machine Vision, Thomson Learning Vocational.

Souza, A., M. Cheriet, et al. (2003). Automatic Filter Selection Using Image Quality Assessment. the Seventh International Conference on Document Analysis and Recognition (ICDAR'03), Edinburgh, Scotland.

Statistics, I. W. (2010). "INTERNET WORLD USERS BY LANGUAGE, Top Ten Languages Used in the Web." from http://www.internetworldstats.com/stats7.htm http://www.internetworldstats.com/stats19.htm.

Stollnitz, E. J., T. D. DeRose, et al. (1995). "Wavelets for Computer Graphics: A Primer Part 1." IEEE Computer Graphics and Applications **15**(3): 76–84.

Taghva, K., J. Borsack, et al. (1995). "Post-Editing through Approximation and Global Correction." International Journal of Pattern Recognition and Artificial Intelligence **9**(6): 911–924.

Taghva, K. and E. Stofsky (2001). "OCRSpell: an interactive spelling correction system for OCR errors in text." International Journal on Document Analysis and Recognition **3**: 125-137.

Tan, X., S. Chen, et al. (2006). "Face Recognition from a Single Image per Person: A Survey." Pattern Recognition **39**(9): 1725-1745.

Tanner, S., T. Muñoz, et al. (2009). "Measuring Mass Text Digitization Quality and Usefulness." D-Lib Magazine **15**(7/8).

Teh, C.-H. and R. T. Chin (1988). "On Image Analysis by the Methods of Moments." IEEE Transactions on Pattern Analysis and Machine Intelligence **10**(4): 496 - 513.

Time. (2008). "Time Archive 1923 to present." from http://www.time.com/time/archive/.

Touj, S., N. E. B. Amara, et al. (2003). Generalized Hough Transform for Arabic Optical Character Recognition. Seventh International Conference on Document Analysis and Recognition (ICDAR 2003), Edinburgh, Scotland.

Trenkle, J., A. Gillies, et al. (2001). Advances In Arabic Text Recognition. Symposium on Document Image Understanding Technology, Maryland, USA.

Trier, Ø. D., A. K. Jain, et al. (1996). "Feature Extraction Methods For Character Recognition - A Survey." Pattern Recognition **29**(4): 641-662.

Trier, O. D. and T. Taxt (1995). "Evaluation of binarization methods for document images." IEEE Transactions on Pattern Analysis and Machine Intelligence **17**(3): 312 - 315.

UCLA, T. U. o. C., Los Angeles. (2006). "Arabic." 2007, from http://www.lmp.ucla.edu/Profile.aspx?LangID=210&menu=004.

Unicode (1991-2006) "Arabic Shaping " Unicode 5.0.0 **Volume**, DOI:

Unicode (2007). "Arabic, Range: 0600-06FF." The Unicode Standard, Version 5.

Vamvakas, G., B. Gatos, et al. (2010). "Handwritten character recognition through two-stage foreground sub-sampling." Pattern Recognition **43**: 2807–2816.

Viola, P. and M. Jones (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE Conference on Computer Vision and Pattern Recognition (CVPR01), Kauai, Hawaii.

Woodford, C. (2010, 02/12/2010). "OCR (Optical character recognition)." Retrieved 23/07/2011
from http://www.explainthatstuff.com/how-ocr-works.html.

World, O. (2010). "Top OCR software." from http://ocrworld.com/software/5-in-depth/149-top-ocr-software.html.

worldwide, M. p. (2009). "World Muslim Population." from http://www.islamicpopulation.com/world_general.html.

Wynne, M. (2005). Corpus and Text — Basic Principles. Developing Linguistic Corpora: a Guide to Good Practice. Oxford, Oxbow Books.

Zemanek, P. (2001). CLARA (Corpus Linguae Arabicae), Charles University, Prague.

Zhang, D. and G. Lu (2001). A comparative study on shape retrieval using Fourier descriptors with different shape signatures. Proceedings of the International Conference on Multimedia and Distance Education, Fargo, ND, USA.

Zidouri, A. (2007). PCA-based Arabic Character feature extraction. 9th International Symposium on Signal Processing and Its Applications (ISSPA 2007 ), Sharjah, United Arab Emirates.

# Appendix: Published papers

AbdelRaouf, A., C. Higgins, et al. (2008). <u>A Database for Arabic printed character recognition</u>. The International Conference on Image Analysis and Recognition-ICIAR2008, Póvoa de Varzim, Portugal, Springer Lecture Notes in Computer Science (LNCS) series.

AbdelRaouf, A., C. Higgins, et al. (2010). "Building a Multi-Modal Arabic Corpus (MMAC)." <u>The International Journal of Document Analysis and Recognition (IJDAR)</u> **13**(4): 285-302.

# A Database for Arabic Printed Character Recognition

Ashraf AbdelRaouf[1,2], Colin A Higgins[1], and Mahmoud Khalil[3]

[1] School of Computer Science, The University of Nottingham, Nottingham, UK
[2] Faculty of Computer Science, Misr International University, Cairo, Egypt
[3] Faculty of Engineering, Ain Shams University, Cairo, Egypt
ara@cs.nott.ac.uk, cah@cs.nott.ac.uk, khalil_mik@yahoo.com

**Abstract.** Electronic Document Management (EDM) technology is being widely adopted as it makes for the efficient routing and retrieval of documents. Optical Character Recognition (OCR) is an important front end for such technology. Excellent OCR now exists for Latin based languages, but there are few systems that read Arabic, which limits the penetration of EDM into Arabic-speaking countries. In developing an OCR system for Arabic it is necessary to create a database of Arabic words. Such a database has many uses as well as in training and testing a recognition system. This paper provides a comprehensive study and analysis of Arabic words and explains how such a database was constructed. Unlike earlier studies, this paper describes a database developed using a large number of collected Arabic words (6 million). It also considers connected segments or Pieces of Arabic Words (PAWs) as well as Naked Pieces of Arabic Word (NPAWs); PAWS without diacritics. Background information concerning the Arabic language is also presented.

**Keywords:** Arabic, Database, Pattern Recognition, OCR, Dictionaries.

## 1 Introduction

A substantial training/testing database in an optical character recognition system is important as it provides a priori contextual information which is crucial in achieving successful recognition rates. With Arabic no central organization is concerned with generating an Arabic corpus, so there is no standard reference list of Arabic words, hence the motivation for creating our own database. We describe the development of a database containing a list of six million Arabic words. This database may be queried in many ways which prove useful in different contexts. The data may be accessed as an array of sorted words or as a sorted list of Pieces of Arabic Word (PAWs) or Naked Pieces of Arabic Word (NPAWs) – see below.

The generation of this database forms part of ongoing research into off-line Arabic character recognition systems. It can be used during training, to validate samples by checking the existence of a given word, or in the recognition phase to eliminate erroneous possibilities. The database is freely available on the Internet.

This paper is organised as follows. Section one is an introduction to the paper and describes our motivation in studying the Arabic language and describes other work that has contributed to our strategies. Section two describes features of the Arabic language and specific recognition difficulties. Section three describes the sources of data, the process by which the data was collected and the difficulties experienced. Section four presents statistical analysis of the database with the algorithms used. Section five explains how the database was validated and tested. Finally, Section six describes the planned future development and use of the database and presents our conclusions.

### 1.1 Motivation

The Arabic language is widely spoken and has been used since the 5th century when written forms were stimulated by the emergence of Islam. It is the native language of more than 230 million speakers and is one of the six official languages of the United Nations (along with Chinese, English, French, Russian and Spanish) [1]. It has been estimated as the tenth language in the world according to the number of Internet users [2] and is the official language of fifteen countries in the world, mainly in the geographical area of the Middle East [3]. There are other languages that use the Arabic alphabet but are not considered as an Arabic language, for example Pashto, Persian, Sindhi, and Urdu. These languages are beyond the scope of our research.

### 1.2 Related Work

There has been a great deal of previous research in three areas related to our topic.

1. **Collected printed word databases:** The Linguistic Data Consortium (LDC) at the University of Pennsylvania produced “Arabic Gigaword Second Edition” [4]. This is a huge database of 1,500 million Arabic words. It has been collected over a period of years from news agencies, but has a number of drawbacks for our purposes. First, the database is collected only from news agencies, whereas a set of more varied sources would be advantageous. Secondly, most of the files come from Lebanese news agencies while it would be better to collect samples from many Arab countries. Thirdly, the database format is in paragraphs and not in single words for testing and training which makes it less immediately useful.

   The Environmental Research Institute of Michigan (ERIM) has created a printed database of 750 pages collected from Arabic books and magazines. This database contains different text qualities saved in an appropriate file formats. However this database has two drawbacks; it is small and hard to access [5].
2. **Creating a lexical database of printed Arabic:** DIINAR.1 is an Arabic lexical database produced by the Euro-Mediterranean project. It comprises 119,693 lemmas distributed between nouns, verbs and adverbials and uses 6,546 roots [6]. The Xerox Arabic Morphological Analyzer/Generator was developed by Xerox in 2001. This contains 90,000 Arabic stems, which can create a derived database of 72 million words [7]. This type of databases partially solves the problem of not having a trusted Arabic corpus, but it misses many of the words used in practice.
3. **Collected handwritten databases:** In 2002 Al-Ma'adeed et al introduced AHDB. A database of 100 different writers which contains Arabic text and words. It contains the most common Arabic words that are used in writing cheques and some handwritten pages [8]. In 2002 another handwritten database of town/village names was created by the Institute for Communications Technology (IFN), Technical

University Braunschweig, Germany and Ecole Nationale d'Ingénieur de Tunis (ENIT). It was completed by 411 writers. They entered about 26,400 names [9]. This database has been used recently in a number of other research projects. The handwritten database has different characteristics than the typed one.

## 2 Overview

In this section the features of the written Arabic language, in additional to the character recognition problems that are peculiar to this language, are discussed. We used the International Unicode Standard as defined by the Unicode Consortium as a reference for character encoding. The Unicode naming convention has been adopted in our research although we mentioned the other schemes in common use [10, 11].

### 2.1 Features of the Arabic Language

1. The Arabic language consists of 28 letters and is written from right to left. Arabic script is cursive even when printed and Arabic letters are connected from the baseline of the word.
2. The Arabic language makes no distinction between capital and lower-case letters; it contains only one case.
3. The digits used in the Arabic language are called Arabic-Indic Digits and were originally invented in India. They were adapted by the Arabic language [11].
4. The widths of letters are variable (for example س and ا).
5. The connecting letter known as Tatweel, or Kashida is used to adjust the left and right alignments; this letter has no meaning in the language, in fact it does not exist at all in any semantic sense.
6. Arabic alphabets depend on dots to differentiate between letters. There are 19 joining groups [12]. Each joining group contains more than one similar letter which are different in the number and place of the dots, as for example (خ ح ج) which have the same joining group (ح) but with differences in the number of dots. Table 1 shows the list of the joining groups with their schematic names.

**Table 1.** Different Arabic joining groups and group letters

| Schematic Name | Joining Group | Group Letters | Schematic Name | Joining Group | Group Letters | Schematic Name | Joining Group | Group Letters |
|---|---|---|---|---|---|---|---|---|
| ALEF | ا | ا إ أ آ | BEH | ب | ب ت ث | HAH | ح | ج ح خ |
| DAL | د | د ذ | REH | ر | ر ز | SEEN | س | س ش |
| SAD | ص | ص ض | TAH | ط | ط ظ | AIN | ع | ع غ |
| FEH | ف | ف | QAF | ق | ق | KAF | ك | ك |
| LAM | ل | ل | MEEM | م | م | NOON | ن | ن |
| HEH | هـ | هـ | WAW | و | و ؤ | YEH | ى | ئ ى ي |
| TEH MARBUTA | ه | ه ة | | | | | | |

7. Arabic letters have four different shapes according to their location in the word [13]; Start, Middle, End and Isolated. For the six letters (ا د ذ ر ز و) there is no Start or Middle location shape. The letter following these six letters must be used in its Start location shape. In the joining type defined by the Unicode Standard all the Arabic letters are Dual Joining, except the previous six letters which are joined from the right side only. Table 2 shows the list of Arabic letters in their different shapes in different locations and their English names.
8. Three letters only (ع غ هـ) have four different glyphs according to their location in the word, while the rest of the Arabic letters have two different glyphs in different locations inside the word. As shown in Table 2.

**Table 2.** List of Arabic letters with their different locations in the word

| Name in English | Arabic Letter | Isolated | Start | Middle | End |
|---|---|---|---|---|---|
| ALEF | ا | ا | | | ـا |
| BEH | ب | ب | بـ | ـبـ | ـب |
| THE | ت | ت | تـ | ـتـ | ـت |
| THEH | ث | ث | ثـ | ـثـ | ـث |
| JEEM | ج | ج | جـ | ـجـ | ـج |
| HAH | ح | ح | حـ | ـحـ | ـح |
| KHAH | خ | خ | خـ | ـخـ | ـخ |
| DAL | د | د | | | ـد |
| THAL | ذ | ذ | | | ـذ |
| REH | ر | ر | | | ـر |
| ZAIN | ز | ز | | | ـز |
| SEEN | س | س | سـ | ـسـ | ـس |
| SHEEN | ش | ش | شـ | ـشـ | ـش |
| SAD | ص | ص | صـ | ـصـ | ـص |
| DAD | ض | ض | ضـ | ـضـ | ـض |
| TAH | ط | ط | طـ | ـطـ | ـط |
| ZAH | ظ | ظ | ظـ | ـظـ | ـظ |
| AIN | ع | ع | عـ | ـعـ | ـع |
| GHAIN | غ | غ | غـ | ـغـ | ـغ |
| FEH | ف | ف | فـ | ـفـ | ـف |
| QAF | ق | ق | قـ | ـقـ | ـق |
| KAF | ك | ك | كـ | ـكـ | ـك |
| LAM | ل | ل | لـ | ـلـ | ـل |
| MEEM | م | م | مـ | ـمـ | ـم |
| NOON | ن | ن | نـ | ـنـ | ـن |
| HEH | هـ | ه | هـ | ـهـ | ـه |
| WAW | و | و | | | ـو |
| YEH | ي | ي | يـ | ـيـ | ـي |

9. The Arabic language incorporates some ligatures such as (Lam Alef لا) which actually consists of two letters (ا ل) but when connected produce another glyph. In some fonts like Traditional Arabic there are some ligatures like (ﲒ) which come from two characters (يمـ).
10. Arabic script can use diacritical marking above and below the letters such as (عِلم عُلم) termed Harakat [11] to help in pronouncing the words and in indicating their meaning [14]. These diacritical markings are not considered in our research.
11. An Arabic word may consist of one or more sub-words. We have termed these disconnected sub-words PAWs (Pieces of Arabic Word) [13, 15]. For example (رسول) is an Arabic word with three PAWs (ل) (سو) (ر). The first and inner PAWs must end with one of the six letters (ا د ذ ر ز و) as these are not left connected.

## 2.2 Arabic Language Recognition Difficulties

The Arabic language is not an easy language for automatic recognition. Some of the particular difficulties are:

1. Characters are cursive and not separated as is the case with Latin script. Hence recognition requires a sophisticated segmentation algorithm.
2. Characters change shape depending on their position in the word, and much of the distinction between isolated characters is lost when they appear in the middle of a word.
3. Sometimes Arabic writers neglect to include whitespaces between words when the word ends with one of the six letters (ا د ذ ر ز و).
4. Repeated characters are sometimes used, even if this breaks Arabic word rules; especially in online “chat” sites, for example (مووووووت) while it is actually(موت).
5. There are two ending letters (ي ى) which sometimes indicate the same meaning but are different characters. For example (الذي) and (الذى) have the same meaning, the first is correct but the second form is often encountered. The same problem exists with the character pair (ه ة).
6. There is often misuse of the letter ALEF (ا) in its different shapes(أ إ آ).
7. The individual letter (و) which means "and" in English is often misused. In this case it is a word and should have whitespace after it, but most of the time Arabic writers neglect to include the whitespace.
8. The Arabic language contains a number of similar letters like ALEF (ا) and the number 1 (١), and also the full stop (.) and the Arabic number 0 (٠) [16].
9. The presence of Arabic font files which define character shapes that are similar to the old form of Arabic writing. These fonts are totally different from the popular fonts. For example a statement with an Arabic Transparent font like (احمد يلعب في الحديقة) when is written in the old shape font like Andalus it becomes (احمد يلعب في الحديقة).
10. It is common to find a transliteration of English based words, especially proper names, medical terms, and Latin-based words.
11. The Arabic language is not based on the Latin alphabet and so requires a different encoding for computer use. This is a purely practical difficulty, but a source of confusion as several incompatible alternatives may be used. A code page is a se-

quence of bits that represent a certain character [17]. There are three main code pages which are used: Arabic Windows 1256, Arabic DOS 720, and ISO 8859-6 Arabic. When selecting any files we must first check the code page used. Most Arabic files use Arabic Windows 1256 encoding, but more recently files have often been encoded using the Unicode UTF-8 code page. The standard code page for Arabic uses Unicode UTF-16 and also the Unicode UTF-32 Standard. The use of Unicode may be regarded as best current practice.

## 3 The Database of Arabic Words

The database contains 6 million Arabic words from a wide variety of selected sources covering old Arabic, religious texts, traditional language, modern language, different specializations and very modern material from "chat rooms." These sources were:

- **Different topical Arabic websites:** We used an Arabic search engine. The search engine specifies the pages according to topic. We downloaded pages allocated to different topics including literature, women, children, religion, sports, programming, design, chatting and entertainment.
- **Common Arabic news websites:** We used the most common Arabic news websites. These included the websites of ElGezira, AlAhram, AlHayat, AlKhalig, Al-Shark AlAwsat, AlArabeya and AlAkhbar. These websites include data which is qualitatively different from that used in the topics above and also comes from a variety of Arabic countries.
- **Arabic-Arabic dictionaries:** These dictionaries contain all the most common Arabic words and word roots with their meanings.
- **Old Arabic books:** These books were generally written centuries ago. They include religious books and books using traditional language.
- **Arabic research:** We used a PhD research thesis in Law.
- **The Holy Quran:** We also used the wording of the Holy Quran.

### 3.1 Data Collection Steps

These steps were taken to overcome the difficulties listed above. They were mainly due to the use of different code page encodings and erroneous words often containing repeated letters or being formed from two concatenated words without white space between. The following steps were followed:

- An Arabic search engine was used to search for Arabic websites [18]. WebZIP 7.0 software was then used to download these websites [19] by selecting files that contain the text, markup and scripting.
- We identified the code page used for each part of the file containing text. Sometimes more than one code page is used.
- A program was developed to remove Latin letters, numbers, and any non Arabic letters even if the letters are from the Arabic alphabets.

- Any typing mistakes were corrected such as those that are common where the word contains (ة or ى) at the end. Also any typing mistakes where the word ends with the two letters (ىء) typed in this order were corrected to (ئ).
- A text file was created containing one Arabic word on each line.
- A program was written to correct the problem of connected words.
- Words which have repeated letters were checked automatically and manually.
- A procedure was written to replace the final letter (ى) with (ي), to replace the final letter (ة) with (ه), and to replace the letters (آ إ أ) with (ا) [20].

## 4 Database Statistics and Analysis

The analysis of the database was concerned with the statistical properties of both words and PAWs. In this section we describe an investigation into the frequency of these entities in Arabic.

### 4.1 Words and PAWs Analysis

Tables (3), (4) and (5) show the detailed analysis of the words and PAWs. The most common words are prepositions while the most common PAWs are those that consist of a single letter. The percentage reduction in considering naked words (without taking account of the dots) rather than unique words is 25%. The number of naked words that are repeated once or twice is less than is the case for words, while naked words that are repeated many times are more than is the case for unique (or decorated) words. The number of unique PAWs is very limited relative to the total number of PAWs. The number of PAWs that are heavily repeated is greater than is the case with words. The percentage of reduction between the unique PAW and the unique naked PAW is 50%. The number of naked PAWs that have few repetitions is less than is the case for PAWs, although the number of much repeated naked PAWs is greater than for unique PAWs.

**Table 3.** The total number of words, naked words, PAWs and naked PAWs with their unique number and average repetition of each of them in additional to the total number of characters

| | Total Number | Number of Unique | Average Repetition |
|---|---|---|---|
| Words | 6,000,000 | 282,593 | 21.23 |
| Naked Words | 6,000,000 | 211,072 | 28.43 |
| PAWs | 14,017,370 | 66,858 | 209.66 |
| Naked PAWs | 14,017,370 | 32,917 | 425.84 |
| Characters | 28,446,993 | | |

**Table 4.** The average number of characters per word, characters per PAW and PAWs per word

| | Average |
|---|---|
| Characters / Word | 4.74 |
| Characters / PAW | 2.03 |
| PAWs / Word | 2.33 |

**Table 5.** The percentage and description of the top ten words, naked words, PAWs and naked PAWs

| | Percentage of all | Description |
|---|---|---|
| Word | 11.36% | Most of these are prepositions |
| Naked Word | 11.37% | Most of these are prepositions |
| PAW | 42.49% | 8 letters, 1 ligature and 1 word |
| Naked PAW | 44.57% | 7 letters, 1 ligature, 1 word and 1 PAW |

## 4.2 General Information

The statistical analysis shows that the number of unique naked PAWs in the whole database is very limited. The number of unique naked words is almost six times that of naked PAWs and is increasing much more rapidly as new text continues to be added to the corpus as shown in figure 1. The number of unique naked words is about 80% of the number of unique words and this ratio remains fairly stable once a reasonably-sized corpus has been established.

The database analysis shows the most repeated words and PAWs in the language. The twenty five most repeated words, naked words, PAWs and naked PAWs and the percentage of each of them in the database is shown in Table 6. This analysis gives almost the same results as that found by others [8, 9, 21]. Table 7 shows the relationship between the total number of words, naked words, PAWs and naked PAWs to the totals and percentage totals in the dataset.

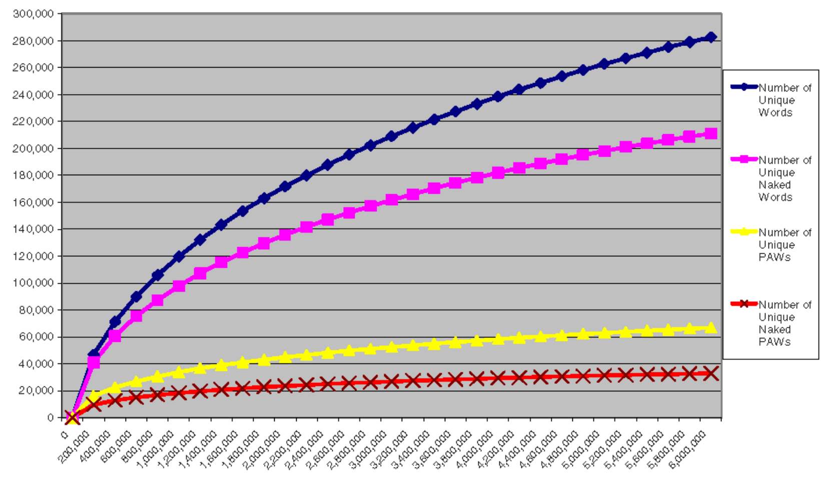


**Fig. 1.** The relationship between the number of unique words, naked words, PAWs and naked PAWs, and the number of words in the database file

**Table 6.** A list of the twenty five most used Arabic words, naked words, PAWs and naked PAWs. Number and percentage of repetitions.

| No. | No. of repetitions | Word | Percentage | No. of repetitions | Naked Word | Percentage | No. of repetitions | PAW | Percentage | No. of repetitions | Naked PAW | Percentage |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 164,165 | في | 2.73% | 164,193 | فى | 2.73% | 2,921,359 | ا | 20.81% | 2,921,359 | ا | 20.84% |
| 2 | 139,380 | من | 2.32% | 139,380 | من | 2.32% | 835,979 | و | 5.96% | 840,793 | و | 5.99% |
| 3 | 82,429 | ان | 1.37% | 82,429 | ان | 1.37% | 445,282 | ر | 3.17% | 511,935 | ر | 3.65% |
| 4 | 78,417 | علي | 1.30% | 78,427 | على | 1.30% | 397,210 | لا | 2.83% | 397,210 | لا | 2.83% |
| 5 | 40,431 | الي | 0.67% | 40,431 | الى | 0.67% | 300,775 | ن | 2.14% | 337,855 | د | 2.41% |
| 6 | 40,091 | هذا | 0.66% | 40,114 | هدا | 0.66% | 271,891 | د | 1.94% | 300,775 | ن | 2.14% |
| 7 | 37,830 | لا | 0.63% | 37,830 | لا | 0.63% | 245,657 | ه | 1.75% | 285,629 | ب | 2.03% |
| 8 | 34,897 | او | 0.58% | 34,897 | او | 0.58% | 198,275 | ت | 1.41% | 245,657 | ه | 1.75% |
| 9 | 34,197 | ما | 0.57% | 34,197 | ما | 0.57% | 176,106 | في | 1.25% | 226,566 | با | 1.61% |
| 10 | 29,896 | عن | 0.49% | 30,315 | البى | 0.50% | 164,317 | ي | 1.17% | 180,173 | فى | 1.28% |
| 11 | 29,793 | التي | 0.49% | 29,900 | عن | 0.49% | 159,823 | ل | 1.14% | 166,129 | ى | 1.18% |
| 12 | 25,759 | اي | 0.42% | 25,759 | اى | 0.42% | 151,009 | من | 1.07% | 159,823 | ل | 1.14% |
| 13 | 21,363 | الذي | 0.35% | 21,413 | الدى | 0.35% | 115,682 | م | 0.82% | 151,009 | من | 1.07% |
| 14 | 20,010 | كل | 0.33% | 20,010 | كل | 0.33% | 115,372 | با | 0.82% | 149,501 | به | 1.06% |
| 15 | 19,576 | هذه | 0.32% | 19,584 | هده | 0.32% | 102,560 | ء | 0.73% | 115,682 | م | 0.82% |
| 16 | 18,566 | هو | 0.30% | 18,566 | هو | 0.30% | 93,731 | ما | 0.66% | 115,436 | بر | 0.82% |
| 17 | 16,738 | كان | 0.27% | 16,738 | كان | 0.27% | 81,895 | علي | 0.58% | 102,560 | ء | 0.73% |
| 18 | 16,467 | مع | 0.27% | 16,468 | مع | 0.27% | 76,162 | يه | 0.54% | 101,839 | بد | 0.72% |
| 19 | 16,317 | لم | 0.27% | 16,317 | لم | 0.27% | 75,344 | ب | 0.53% | 96,635 | لد | 0.68% |
| 20 | 14,065 | اذا | 0.23% | 15,910 | بعد | 0.26% | 69,987 | كا | 0.49% | 93,859 | بو | 0.67% |
| 21 | 13,975 | بعد | 0.23% | 14,602 | ابه | 0.24% | 69,779 | لي | 0.49% | 93,731 | ما | 0.66% |
| 22 | 12,944 | ولا | 0.21% | 14,081 | ادا | 0.23% | 66,653 | ز | 0.47% | 82,003 | على | 0.58% |
| 23 | 12,863 | ذلك | 0.21% | 12,944 | ولا | 0.21% | 65,964 | ذ | 0.47% | 74,079 | فا | 0.52% |
| 24 | 11,750 | بين | 0.19% | 12,873 | دلك | 0.21% | 63,088 | هذ | 0.45% | 69,987 | كا | 0.49% |
| 25 | 11,688 | انه | 0.19% | 12,023 | بين | 0.20% | 59,907 | لو | 0.42% | 69,791 | لى | 0.49% |

**Table 7.** Total number of words, naked words, PAWs and naked PAWs that are repeated once in relation to the total unique number and the total number

| | Total Number | Percentage of the total number of unique | Percentage of the total number of words |
|---|---|---|---|
| Word | 107,771 | 38.14% | 1.796% |
| Naked Word | 69,506 | 32.93% | 1.158% |
| PAW | 20,239 | 30.27% | 0.144% |
| Naked PAW | 8,201 | 24.92% | 0.058% |

## 4.3 Database Analysis Steps

- Dealing with a data file that contains 6,000,000 words as a sequential structure is an inefficient and clumsy process. Also creating statistics based on words from the same domain produces bounds on the chart (Figure 1). Hence we developed a program to convert this to a binary file with a fixed field width of 25 characters. The words in the data file are randomized to obtain meaningful statistical analysis. It also creates 30 different text files with 200,000 words in each.

- A program to create naked word files was also developed. It reads each word sequentially and replaces each letter from the group letters with the joining group letter (Table 1), for example it replaces (خ and ح and ج) with (ح).
- A program to create PAW files was developed. It reads each word sequentially from the words files. It checks if the word contains one of the six letters ( ا، د، ذ، ر، ز، و) in the middle of the word and if so puts a carriage return after it. It saves the new PAW files.

## 5 Testing Database Validity

Testing the validity and accuracy of the database is a very important issue in creating any database [22]. Our testing process started by collecting data from unusual sources. The testing data were collected from scanned images from faxes, documents, books and medicine scripts. Another source was well known Arabic news websites. Some of the testing data files were collected again after six months; while some others were collected after twelve months. Testing data was collected from sources that intersected minimally with those of the database.

### 5.1 Testing Words and PAWs

A program was developed to search in the binary database file using a Binary search tree algorithm. The total number of words in the testing data is 69,158 and the total number of PAWs is 165,501. Table 8 shows the total number of words, naked words, PAWs and naked PAWs of the testing data set in relation to the number of words found in the database.

**Table 8.** Total number of words, naked words, PAWs and naked PAWs of the testing data set in relation to the data base

| | Test data set Total Number | Test data set Unique Number | Number of Unique found | Percentage of Unique found | Number of Unique Missed | Percentage of Unique Missed | Total number of Missed | Percentage of total Missed |
|---|---|---|---|---|---|---|---|---|
| Word | 69,158 | 17,766 | 15,967 | 89.8% | 1,799 | 10.2% | 2,457 | 3.5% |
| Naked Word | 69,158 | 16,513 | 15,163 | 91.8% | 1,350 | 8.2% | 1,843 | 2.66% |
| PAW | 165,501 | 7,275 | 6,989 | 96% | 286 | 4% | 1,583 | 0.95% |
| Naked PAW | 165,501 | 4,754 | 4,636 | 97.5% | 118 | 2.5% | 1,341 | 0.81% |

### 5.2 Checking Database Accuracy

The testing data file is used to check the accuracy of the statistics obtained from the database. By extrapolation, we believe that if the curve between the number of words and the unique number of words is extended according to the number of words that exists in the testing data file, we will find that the increase in the unique number of words is almost equal to the missing words in the database.

The shape of the curve between the total number of words and the unique number of words is a nonlinear regression model. We used a shareware program called CurveExpert 1.3 [23] and applied its curve finder process. We found that the best regression model is the Morgan-Mercer-Flodin (MMF) model, from the Sigmoidal Family [24]. The best fit curve equation is:

$$y = \frac{ab + cx^d}{b + x^d} \tag{1}$$

Where:

| | |
|---|---|
| *a = -42.558444* | *b = 40335.007* |
| *c = 753420.21* | *d = 0.64692321* |
| *Standard Error: 156.0703663* | *Correlation Coefficient: 0.9999980* |

Upon applying equation 1, we found that the increase in the number of words to 6,069,158 words will increase the unique words by 1,676 words, while the number of missing words is 1,799. This means that the accuracy is a respectable 93%.

## 6 Future Work and Conclusions

Future work on the database includes a continuous process of adding new words. It must be flexible enough to include the documents' scanned images. Computer generated PAWs will be provided to test the recognized PAW. It also has to include data structures to contain the entire images database. The Arabic database must cover different specializations, different periods of time and different regions in the Arabic world. Using Unicode as a standard code page is very important as unification among a variety of languages is one of the big problems in automating the processing of non-Latin languages. Statistical analysis applications show the importance of PAWs and naked PAWs in the Arabic language. However, it becomes clear that a powerful stemming algorithm must be applied to the database to enhance data retrieval for improved accuracy.

## References

1. INDEXES : United Nations Documentation, the Department of Public Information (DPI), Dag Hammarskjöld Library (DHL) (2007), `http://www.un.org/Depts/dhl/resguide/itp.htm`
2. INTERNET WORLD USERS BY LANGUAGE, Top Ten Languages Used in the Web,Internet World Stats, Usage and Population Statistics (2007)
3. T. U. o. C. UCLA, Los Angeles, "Arabic", International Institute, Center for World Languages, Language Materials Project (2006)
4. David Graff, K.C., Kong, J., Maeda, K.: Arabic Gigaword Second Edition. Philadelphia: Linguistic Data Consortium, University of Pennsylvania (2006)

5. Schlosser, S.: ERIM Arabic Document Database, Environmental Research Institute of Michigan
6. Ramzi Abbes, J.D., Hassoun, M.: The Architecture of a Standard Arabic Lexical Database. Some Figures, Ratios and Categories from the DIINAR.1 Source Program. In: Workshop of Computational Approaches to Arabic Script-based Languages, Geneva, Switzerland (2004)
7. Beesley, K.R.: Arabic Finite-State Morphological Analysis and Generation. In: COLING, Copenhagen (1996)
8. Al-Ma'adeed, S., Elliman, D., Higgins, C.A.: A data base for Arabic handwritten text recognition research. In: Eighth International Workshop on Frontiers in Handwriting Recognition (2002)
9. Pechwitz, M., Maddouri, S.S., Märgner, V., Ellouze, N., Amiri, H.: IFN/ENIT - DATABASE OF HANDWRITTEN ARABIC WORDS. In: 7th Colloque International Francophone sur l'Ecrit et le Document, CIFED 2002, Tunisia (2002)
10. Unicode, Arabic, Range: 0600-06FF, The Unicode Standard, Version 5 (2007), http://www.unicode.org/charts/PDF/U0600.pdf
11. The Unicode Consortium. The Unicode Standard, Version 4.1.0, Boston, MA, pp. 195–206. Addison-Wesley, Reading (2003)
12. Unicode, "Arabic Shaping" in Unicode 5.0.0 (1991-2006), http://unicode.org/Public/UNIDATA/ArabicShaping.txt
13. Liana, V.G., Lorigo, M.: Offline Arabic Handwriting Recognition: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 712–724 (2006)
14. Fahmy, M.M.M., Ali, S.A.: Automatic Recognition Of Handwritten Arabic Characters Using Their Geometrical Features. Journal of Studies in Informatics and Control with Emphasis on Useful Applications of Advanced Technology 10 (2001)
15. Amin, A.: Off line Arabic character recognition - a survey. In: Fourth International Conference on Document Analysis and Recognition, Germany (1997)
16. Harty, R., Ghaddar, C.: Arabic Text Recognition. The International Arab Journal of Information Technology 1, 156–163 (2004)
17. W. contributors, Code page. From Wikipedia, the free encyclopedia, Wikipedia, The Free Encyclopedia (2006)
18. arabo.com, Arabo Arab Search Engine & Dictionary (2005)
19. S. P. Ltd, WebZIP 7.0, 7.0 ed (2006)
20. Larkey, L.S., Ballesteros, L., Connell, M.E.: Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis. In: 25th International Conference on Research and Development in Information Retrieval (SIGIR) (2002)
21. Buckwalter, T.: ARABIC WORD FREQUENCY COUNTS (2002), http://www.qamus.org/transliteration.htm
22. Mashali, S., Mahmoud, A., Elnemr, H., Ahmed, G., Osama, S.: Arabic OCR Database Development. In: Fifth Conference on Language Engineering, Egypt (2005)
23. Hyams, D.G.: CurveExpert 1.3, A comprehensive curve fitting system for Windows, 1.3 ed (2005)
24. Gu, B., Hu, F., Liu, H.: Modelling Classification Performance for Large Data Sets. In: Wang, X.S., Yu, G., Lu, H. (eds.) WAIM 2001. LNCS, vol. 2118. Springer, Heidelberg (2001)

IJDAR (2010) 13:285–302
DOI 10.1007/s10032-010-0128-2

ORIGINAL PAPER

# Building a multi-modal Arabic corpus (MMAC)

**Ashraf AbdelRaouf · Colin A. Higgins ·
Tony Pridmore · Mahmoud Khalil**

Received: 3 June 2009 / Revised: 3 September 2010 / Accepted: 9 September 2010 / Published online: 29 September 2010

**Abstract** Traditionally, a corpus is a large structured set of text, electronically stored and processed. Corpora have become very important in the study of languages. They have opened new areas of linguistic research, which were unknown until recently. Corpora are also key to the development of optical character recognition (OCR) applications. Access to a corpus of both language and images is essential during OCR development, particularly while training and testing a recognition application. Excellent corpora have been developed for Latin-based languages, but few relate to the Arabic language. This limits the penetration of both corpus linguistics and OCR in Arabic-speaking countries. This paper describes the construction and provides a comprehensive study and analysis of a multi-modal Arabic corpus (MMAC) that is suitable for use in both OCR development and linguistics. MMAC currently contains six million Arabic words and, unlike previous corpora, also includes connected segments or pieces of Arabic words (PAWs) as well as naked pieces of Arabic words (NPAWs) and naked words (NWords); PAWs and Words without diacritical marks. Multi-modal data is generated from both text, gathered from a wide variety of sources, and images of existing documents. Text-based data is complemented by a set of artificially generated images showing each of the Words, NWords, PAWs and NPAWs involved. Applications are provided to generate a natural-looking degradation to the generated images. A ground truth annotation is offered for each such image, while natural images showing small paragraphs and full pages are augmented with representations of the text they depict. A statistical analysis and verification of the dataset has been carried out and is presented. MMAC was also tested using commercial OCR software and is publicly and freely available.

A. AbdelRaouf (✉) · C. A. Higgins · T. Pridmore
School of Computer Science, The University of Nottingham,
Nottingham, UK
e-mail: ara@cs.nott.ac.uk

C. A. Higgins
e-mail: cah@cs.nott.ac.uk

T. Pridmore
e-mail: tpp@cs.nott.ac.uk

A. AbdelRaouf
Faculty of Computer Science, Misr International University,
Cairo, Egypt

M. Khalil
Computer and Systems Engineering Department,
Faculty of Engineering, Ain Shams University,
Cairo, Egypt
e-mail: mahmoud.khalil@eng.asu.edu.eg

**Keywords** Corpora · Arabic · Linguistics · Pattern recognition · OCR

## 1 Introduction

Traditionally, a corpus is a large, structured collection of text covering a huge number of words from different domains of a given language. The first modern corpus was the Brown Corpus. It was collected and compiled in 1967 [1] and contains almost 1 million English words from different disciplines. Currently, the three best-known English corpora are as follows: The Corpus of Contemporary American English, which contains 410+ million words, is available via the internet and was created during the period (1990–2008). It is updated once or twice a year [2]; The British National Corpus, which contains 100 million words, is also available via the internet, was created during the period (1980–1993) and still being updated [3]; and the Time Archive (1923—to present) from Time magazine. The Time Archive was originally created to archive the magazine's articles and contains more than 275,000 articles with around 100 million words [4].

Corpora are increasingly valuable in computational linguistics, as well as supporting statistical analysis of languages. They are central to the development of computational techniques associated with language; such as optical character recognition (OCR), speech recognition and natural language processing. Corpora intended for use in these areas may include data beyond simple text, such as audio (speech) or images. These multi-modal corpora are also gaining increased attention in mainstream corpus linguistics [5].

Multi-modal corpora are of particular value during development of optical character recognition (OCR) applications. OCR methods must be trained with and evaluated against datasets that capture the structure of the target language, but emphasize its appearance in image data.

The Arabic language is widely spoken and has been used since the fifth century when written forms were stimulated by Islam. It is the native language of more than 340 million speakers and is one of the six official languages of the United Nations (along with Chinese, English, French, Russian and Spanish) [6]. It has been estimated to be one of the ten most used languages on the Internet [7] and is the official language of 22 countries located in the geographical area of the Middle East [8]. As there is no central organization for the Arabic language, no standard linguistic or image corpus exists [9].

This paper describes the generation of a corpus containing six million Arabic words and associated image data. Compared to earlier Arabic corpora, our corpus has a number of advantages that make it useful in a wider range of application areas. The lists and words it contains are in a variety of formats, allowing access and retrieval in diverse ways. It allows access via pieces of Arabic words (PAWs) [10,11] as well as naked pieces of Arabic words (NPAWs) and Naked Words (NWords); PAWs and words without diacritical marks. A set of artificially generated images showing each of the tokens found in the corpus is provided, with each token shown in three key Arabic fonts. A set of applications is provided that add noise, generating degraded images that simulate real images of Arabic text. Each image is accompanied by annotations in XML format, creating a valuable ground truth resource. The original images and degraded copies are beneficial when testing document image analysis applications and make MMAC a real benchmark for Arabic OCR projects [12]. The corpus also includes real images of the Arabic documents used in its production, a necessary addition for OCR usage. These documents are in full-page and single-paragraph formats. The three different image formats—single token, paragraph and full page—increase the usability of MMAC. The corpus' value as a benchmark dataset is evaluated using Readiris Pro 10 [13], a well-known commercial software package useful as an exemplar of Arabic OCR applications. Finally, statistical analysis of the corpus' contents is also provided. The corpus is freely available from http://www.ashrafraouf.com/mmac.

OCR methods often rely upon knowledge of the structure of the target language, and it can be beneficial to integrate a corpus into a developing OCR method or system. A corpus may, for example, be used in the recognition phase to eliminate erroneous hypotheses. We would stress, however, that is not the intention here. Though it may be used in this way at a later date, MMAC is primarily conceived as a benchmark dataset to be used during training and testing and not as a system component. Other languages also use the Arabic alphabet, for example Pashto, Persian, Sindhi and Urdu. Though study of these languages may benefit from access to high-quality, multi-modal Arabic corpora, they again are not explicitly considered here.

The remainder of the paper is organized as follows. Section 2 briefly reviews existing Arabic corpora, while Sect. 3 describes key features of the written Arabic language and the specific recognition difficulties they present. Section 4 describes the textual data sources used, and the processes by which the data was collected and processed to produce a combined text and image-based corpus. Further multi-modal data was obtained from real images, by scanning a number of documents and generating associated text data. This process and the resulting data are described in Sect. 5. Section 6 considers the use of MMAC for testing OCR applications describes a real experiment. Section 7 presents a statistical analysis of the corpus. Section 8 explains how the corpus was validated and tested. Section 9 provides a description of the freely available MMAC corpus and defines the transliteration encoding used with the Arabic tokens. Finally, Sect. 10 describes the planned future development and usage of the corpus.

## 2 Related works

Considerable previous research has been directed toward the creation of corpora describing printed Arabic. The emphasis, however, has been on text-based corpora.

### 2.1 Collected word corpora

The Linguistic Data Consortium (LDC) at the University of Pennsylvania has produced the "Arabic Gigaword Fourth Edition" [14]. This is a database of 850 million Arabic words collected over several years from news agencies. Though important, it has a number of drawbacks for our purposes. First, the corpus is collected only from news agencies, limiting the linguistic style of the material captured. Secondly, most of the files come from Lebanese news agencies. The lack of samples from other Arab countries limits the scope of any OCR application that relies on them. Finally, the corpus format is based upon paragraphs and not single words. This makes it less useful for testing and training of OCR methods.

The Institute of Ancient Near Eastern Studies, Charles University Prague (Czech Republic) has compiled the CLARA Corpus. This is an electronic corpus of Modern Standard Arabic. It contains 37 million words. Though the project started in 1997 [15], the corpus lacks variety in the disciplines and geographical areas sampled. This lack of variety reduces its usefulness to the developers of OCR applications.

The University of Essex, in collaboration with the Open University, has developed a corpus of Al-Hayat newspaper articles. It contains 18.5 million words from 42,591 articles, but covers only 7 main subjects [16]. The An-Nahar newspaper has also produced its own an An-Nahar text corpus. This contains 24 million words from 45,000 articles. It was developed between 1995 and 2000 [17]. This type of corpus has some drawbacks that limit its value to OCR and, to a certain extent, corpus linguistics in general. First, they represent the home country of the newspaper, and not all Arab countries. Secondly, they sample only a small time period. Finally, they cover only the subjects discussed in a particular newspaper. Lack of variety in a corpus obviously reduces its usefulness to the developers of OCR applications.

The DIVA Group from the University of Fribourg (Switzerland) in collaboration with REGIM Group from the University of Sfax (Tunisia) and the Software Engineering Unit of the Business Information System Institute (HES-SO //Wallis—Switzerland) recently generated the APTI Arabic Printed Text Image Database. This is a large-scale benchmark of open-vocabulary, multi-font, multi-size and multi-style text recognition systems in Arabic. Using 113,284 words with 10 different Arabic fonts, 10 Arabic font sizes and 4 font styles, APTI generated a total of 45,313,600 words. Its variety of fonts, text sizes and styles makes APTI a valuable resource when testing Arabic OCR applications, but its sources of data lack variety, and its image generation process relies upon down-sampling, which is not a good simulation of the real data. APTI can be used with OCR applications that recognize words only, and offers no support to applications working at the PAWs level [18].

### 2.2 Morphologically created word corpora

DIINAR.1 is an Arabic lexical corpus produced by the Euro-Mediterranean project. It comprises 119,693 lemmas distributed between nouns, verbs and adverbs. It uses 6,546 roots [19].

The Xerox Arabic Morphological Analyzer/Generator was developed by Xerox in 2001. This contains 90,000 Arabic stems, which can create a derived corpus of 72 million words [20]. This type of corpora partially solves the problem of not having a definitive Arabic corpus, but it does not include many words used in everyday life.

### 2.3 Requirements and limitations

To adequately represent a given language, a corpus must have sufficient capacity [9] and include words from a wide variety of sources [9,21,22]. We believe that a good corpus should include the following [23]:

- Language from different disciplines, e.g. engineering, literature, art and medicine.
- Popular language from everyday life.
- Different morphological features of the language. These features are gender (male or female), tense (past, present or future), number (singular, pair or plural), person (first, second or third), imperative verb and definiteness [24].
- Examples created over a long period of time. The Arabic language has been in use for 1,500 years, very old poems, fairy tales, religion and science books are available.
- Examples from varying geographical areas. All the different accents of the Arabic-speaking countries and regions should be covered.
- Images of the contents of the corpus with different image sizes and formats that can be used with OCR applications.

These features are also required of corpora intended to provide good training/testing sets for OCR.

Corpora should provide flexible manipulation and retrieval of data. This is eased by the use of lower level representations; language stored as words, for example, can be accessed in more ways than that held only in paragraphs or full pages. Again, similar criteria apply to multi-modal corpora created to support research in OCR.

The corpora discussed earlier each fail to meet one or more of these criteria. Moreover, previous Arabic corpora are purely linguistic. A substantial literature review by the authors has failed to locate any freely available corpora of images of printed Arabic. Note also that existing corpora operate exclusively at the word level. Though words are clearly important, the structure of the Arabic language and the challenges it presents to OCR suggest that this is not sufficient basis upon which to build a truly effect multi-modal Arabic corpus.

## 3 The written Arabic language

In this section, we provide an overview of the written Arabic languages and discuss the problems these create for the developer of an OCR application. These factors impact upon the design of successful corpora. The Unicode naming convention has been adopted here, though other schemes are in use [25, 26].

## 3.1 Key features of written Arabic

Written Arabic is both rich and complex, features of note are as follows:

- The Arabic language consists of 28 Arabic letters [26] and is written from right to left. Arabic script is cursive even when printed and Arabic letters are connected by the baseline of the word. The Arabic language makes no distinction between capital and lower-case letters; it contains only one case.
- The widths of letters in Arabic are variable (for example س and ا).
- The connecting letter known as Tatweel or Kashida is used to adjust the left and right alignments; this letter has no meaning in the language, it does not exist at all in any semantic sense.
- Arabic alphabets depend on dots to differentiate between letters. There are 19 "joining groups" [27]. Each joining group contains more than one similar letter that differs in the number and place of the dots, as for example (خ ح ج) that have the same joining group (ح) but with differences in the place of dots. Table 1 lists the joining groups, their schematic names and their group letters. We use the Hamza "ء" as a joining group, although it is not included in Unicode.
- Arabic letters have four different shapes according to their location in the word [10]; start, middle, end and isolated. For the six letters (ا د ذ ر ز و) there is no start or middle location shape. The letter following these six letters must be used in its start or isolated location shape. In the joining type defined by the Unicode standard, all the Arabic letters are dual joining, except the previous six letters and the TEH MARBOTA "ة" which is joined from the right side only. The Hamza is not a joining letter. Table 2 shows the different shapes Arabic letters adopt in different locations and gives their English names.
- Three letters only (هـ غ ع) have four different glyphs according to their location in the word, while the rest of the Arabic letters have two different glyphs in different locations inside the word. This is shown in Table 2.
- Fifteen Arabic letters have dots: 12 letters have dots above the baseline; while three have dots below it. Ten of the

**Table 1** Arabic joining groups and group letters, with their English schematic name

| Schematic Name | Joining Group | Group Letters |
|---|---|---|
| ALEF | ا | ا إ أ آ |
| BEH | ب | ب ت ث |
| HAH | ح | ج ح خ |
| DAL | د | د ذ |
| REH | ر | ر ز |
| SEEN | س | س ش |
| SAD | ص | ص ض |
| TAH | ط | ط ظ |
| AIN | ع | ع غ |
| FEH | ف | ف |
| QAF | ق | ق |
| KAF | ك | ك |
| LAM | ل | ل |
| MEEM | م | م |
| NOON | ن | ن |
| HEH | هـ | هـ |
| WAW | و | و ؤ |
| YEH | ى | ى ئ ي |
| TEH MARBUTA | ه | ة ه |
| HAMZA | ء | ء |

**Table 2** Arabic letters and their shapes at different locations within words

| English Name | Arabic Letter | Isolated | Start | Middle | End |
|---|---|---|---|---|---|
| ALEF | ا | ا | | | ـا |
| BEH | ب | ب | بـ | ـبـ | ـب |
| THE | ت | ت | تـ | ـتـ | ـت |
| THEH | ث | ث | ثـ | ـثـ | ـث |
| JEEM | ج | ج | جـ | ـجـ | ـج |
| HAH | ح | ح | حـ | ـحـ | ـح |
| KHAH | خ | خ | خـ | ـخـ | ـخ |
| DAL | د | د | | | ـد |
| THAL | ذ | ذ | | | ـذ |
| REH | ر | ر | | | ـر |
| ZAIN | ز | ز | | | ـز |
| SEEN | س | س | سـ | ـسـ | ـس |
| SHEEN | ش | ش | شـ | ـشـ | ـش |
| SAD | ص | ص | صـ | ـصـ | ـص |
| DAD | ض | ض | ضـ | ـضـ | ـض |
| TAH | ط | ط | طـ | ـطـ | ـط |
| ZAH | ظ | ظ | ظـ | ـظـ | ـظ |
| AIN | ع | ع | عـ | ـعـ | ـع |
| GHAIN | غ | غ | غـ | ـغـ | ـغ |
| FEH | ف | ف | فـ | ـفـ | ـف |
| QAF | ق | ق | قـ | ـقـ | ـق |
| KAF | ك | ك | كـ | ـكـ | ـك |
| LAM | ل | ل | لـ | ـلـ | ـل |
| MEEM | م | م | مـ | ـمـ | ـم |
| NOON | ن | ن | نـ | ـنـ | ـن |
| HEH | هـ | ه | هـ | ـهـ | ـه |
| WAW | و | و | | | ـو |
| YEH | ي | ي | يـ | ـيـ | ـي |

Table 3 Arabic letters with dots

| Description | Arabic Letters |
|---|---|
| Letters with dots above baseline | ت ث خ ذ ز ش ض ظ غ ف ق ن |
| Letters with dots below baseline | ب ج ي |
| One dot Letters | ب ج خ ذ ز ض ظ غ ف ن |
| Two dots letters | ت ق ي |
| Three dots letters | ث ش |

fifteen letters use one dot, three use two dots and two use three dots [28]. Table 3 shows these letters.

– The Arabic language incorporates ligatures such as (Lam Alef لا) that actually consist of two letters (ل ا) but when connected produce another glyph. In some fonts like Traditional Arabic, there are ligatures like (ﳝ) that come from two characters (يمـ).
– Arabic script can use diacritical marking above and below the letters such as (عِلم عُلم) termed Harakat [26] to help in pronouncing the words and in indicating their meaning [29]. These diacritical markings are not considered here.
– An Arabic word may consist of one or more sub-words. We have termed these disconnected sub-words PAWs (pieces of Arabic word) [10, 11]. For example, (رسول) is an Arabic word with three PAWs (ل) (سو) (ر). The first and inner PAWs of the word must end with one of the six letters (ا د ذ ر ز و) as these are not left connected. Hamza is a separate PAW.

## 3.2 Recognizing written Arabic

The Arabic language is not an easy language for automatic recognition. Some of the particular difficulties that must be faced by the developers of OCR applications are as follows:

– Characters are cursive and not separated, as in the case of Latin script. Hence, recognition requires a sophisticated segmentation algorithm.
– Characters change shape depending on their position in the word. The distinction between isolated characters is lost when they appear in the middle of a word.
– Sometimes Arabic writers neglect to include whitespace between words when the word ends with one of the six letters (ا د ذ ر ز و). This is known as the connected words problem [30]. For example, "الأستاذاحمد" should be "الأستاذ احمد". On the other hand, they sometimes separate a single word into two words when the word is long or is pronounced in two parts; "سبع مائه" is actually "سبعمائه" [30].
– Repeated characters are sometimes used, even if this breaks Arabic word rules. This is especially common in online "chat" sites; for example (موووووت) is actually (موت).
– There are two ending letters (ي ى) that sometimes indicate the same meaning but are different characters. For example, (الذي) and (الذى) have the same meaning, the first is correct but the second form is often encountered. The same problem exists with the character pair (ه ة).
– There is often misuse of the letter ALEF (ا) in its different shapes (أ إ آ).
– The individual letter (و) that means "and" in English is often misused. It is a separate word and should have whitespace after it, but most of the time Arabic writers do not use the whitespace. This is a particularly common instance of the connected word problem.
– The Arabic language contains a number of similar letters like ALEF (ا) and the number 1 (١), and also the full stop (.) and the Arabic number 0 (٠) [31].
– Arabic font files exist that define character shapes similar to the old form of Arabic writing. These fonts are totally different from the popular fonts. For example, a statement with an Arabic Transparent font like (احمد يلعب في الحديقة) when written in the old shape font like Andalus becomes (احمد يلعب في الحديقة).
– It is common to find transliterations of English-based words, especially proper names, medical terms and Latin-based words.
– The Arabic language is not based on the Latin alphabet and therefore requires different encoding for computer use. This is a practical problem, and a source of confusion, as several incompatible alternatives may be used. A code page is a sequence of bits representing a certain character [32, 33]. There are three main code pages used: Arabic Windows 1256, Arabic DOS 720 and ISO 8859-6 Arabic. Upon selecting any files, we must first check the code page used. Most Arabic files use Arabic Windows 1256 encoding. Recently, files have been encoded using the Unicode UTF-8 code page. The standard code page for Arabic is Unicode UTF-16 and also the Unicode UTF-32 Standard. The use of Unicode may be regarded as best current practice.

## 3.3 PAWs and NPAWs

Arabic is an intricate language. It is therefore crucial that an equally rich and complex corpus is available to support development of Arabic OCR applications. In particular, the importance of PAWS and NPAWS is such that they must be included in any realistic text/image corpus. The addition of PAWs and naked PAWs is a novel extension of the previous word-based approach to Arabic corpora that specifically facilitates Arabic OCR. We include PAWs and NPAWs in our multi-modal Arabic corpus (MMAC) for the following reasons:

- PAWs are the smallest token in the Arabic language.
- Each PAW is one glyph, making written Arabic a disconnected sequence of PAWs. Using PAWs is a reasonable way to overcome the problem of connected words.
- Naked Words and naked PAWs reduce the process of detecting sub-word ink from 28 letters to 19 glyphs.
- The total number of PAWs representing the language in our corpus is 66,725, while the total number of NPAWs is comparatively small (32,804). Using NPAWs reduces the combinatorics of the problem.
- PAWs consisting of one and two digits can be extracted first, as if they are disconnected letters. These small PAWs represent a significant proportion of the language and are the most commonly used. Their recognition can provide a firm foundation for later processes.

## 4 Building a multi-modal corpus from textual data

### 4.1 Coverage and sources

The multi-modal Arabic corpus contains 6 million Arabic words selected from various sources covering old Arabic, religious texts, traditional language, modern language, different specialisations and very modern material from online “chat rooms.” The sources used are as follows:

*Topical Arabic websites*

These were obtained via an Arabic search engine. The search engine specifies the web-pages according to topic. Pages allocated to different topics including literature, women, medicine, business, children, religion, sports, programming, design and entertainment were downloaded and incorporated into MMAC.

*Arabic news websites*

A set of the most widely used Arabic news websites were identified and downloaded. These included the websites of Al Jazeera, Al Ahram, Al Hayat, Al Khaleej, Asharq Al Awsat, Al Arabiya and Al Akhbar. Professional news websites include language that is qualitatively different from that used in the first set of websites, which were created by the general public. They also come from a variety of Arabic countries.

*Arabic chatrooms*

Some Arabic chatrooms were used for areas such as sports, cultural and social.

*Arabic-Arabic dictionaries*

These dictionaries contain all the most common Arabic words and word roots, along with their definition.

*Old Arabic books*

These were generally written centuries ago. They include religious books and books using traditional language.

*Arabic research*

A PhD thesis in Law was included to sample the research literature.

*The Holy Quran*

The wording of the Holy Quran was also used.

### 4.2 Text data collection

The following steps were taken to overcome the difficulties discussed in Sect. 3.2, which are mainly due to the use of different code page encodings and words often containing repeated letters or being formed from connected words. The steps are as follws:

1. An Arabic search engine [34] was used to search for Arabic websites. WebZIP 7.0 software was then used to download these websites [35] by selecting files that contain text, markup and scripting.
2. The code page used for each part of the file was identified. Sometimes more than one code page is used.
3. A program was developed to removes Latin letters, numbers and any non-Arabic letters even if the letters are from the Unicode Arabic alphabets like this symbol used in Holy Quran (۝). This program also removes any diacritics.
4. Some common typographical errors were corrected. For example, words containing (ة or ى) inside but not at the end were corrected. Also, words containing those two letters (ىء) at the end were corrected to a single letter (ئ).
5. A text file was created containing one Arabic word per line.
6. A program was written to correct the problem of connected words. It scans the words and delivers a list of words with size more than 7 letters. The list is checked manually and corrected. The program rewrites the corrected words in the list of words.
7. Words containing repeated letters were checked automatically and then manually. A program was written to

list all the words that include any repeated letters. The list was checked manually and corrected. The program rewrites the corrected words in the list.

8. A program was written and applied that replaces the final letters (ى) with (ي), (ة) with (ه), and (أ إ آ) with (ا) [36].

## 4.3 Creating NWords, PAWs and NPAWs

The words obtained by the process outlined earlier allow generation of NWords, PAWs and NPAWs data files. This is achieved as follows:

1. A program to create NWord data files was developed. This reads each word sequentially and replaces each letter from the group letters with its joining group letter (Table 1), for example it replaces (ج ح خ) with (ح).
2. A program to create PAW data files was developed which reads each word sequentially from the words data file, checks if the word contains one of the six letters (ا د ذ ر ز و) in the middle of the word, and if so puts a carriage return after it. The same is done with the Hamza "ء" but it also puts carriage return before it. The carriage return thus separates this PAW onto a new line. Finally, it saves the new PAWs file.
3. The program used to convert Words to NWords was used to transfer PAWs to NPAWs. The program is applied to the PAWs data file and generates a NPAWs data file.

## 4.4 Creating images from corpus text data

The methods described in Sects. 4.2 and 4.3 produce a text-based Arabic corpus. To support the development of Arabic OCR applications, MMAC augments its textual data with images [37] and ground truth information for these images [18]. Each token (Word, NWord, PAW or NPAW) is shown in the three most common Arabic fonts; Simplified Arabic, Arabic Transparent, and Traditional Arabic [38]. Token images are stored in greyscale Windows bmp file format, with a resolution of 300DPI [39]. Figure 1 gives an example of a word represented as images showing its appearance in three common Arabic fonts.

The data collection process produced a data file containing the unique Word, NWord, PAW and NPAW tokens, sorted alphabetically. To produce the image files, we transfer each token from this text file to a Windows bmp image file. A program was developed to achieve this as follows:

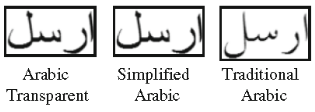


**Fig. 1** A word images showing a word "ارسل" in the three different Arabic fonts

1. The program opens the text file containing the list of unique tokens using the Windows CP1256 character encoding. It reads the file line by line and writes each token to bitmap memory allocation with 300DPI resolution. The font format is changed to the three most common fonts. The font size is 14 point.
2. The program saves the bitmap memory allocation to disk in Windows bmp greyscale format. It names the resultant image with a sequential number, for example Word0000045Font1.bmp. Each token generates three different image files, one for each of the three fonts.
3. The program generates a ground truth XML file for each image file from the previous files containing all the information about the token. The information saved contains both text and image information.

The final token image dataset consists of files containing the images of the token. Each token has three font files, Simplified Arabic, Traditional Arabic and Arabic Transparent. To simplify handling of the huge number of files, the program saves every 500 tokens files in a separate sub-folder. Table 4 shows the total number of images of the different tokens that are generated from the list of unique tokens.

Table 5 shows the average letters per token, average width and average height of the Words, NWords, PAWs and NPAWs. The information about the token images shows that the average width of the words and NWords are the same, while the height is different due to the existence of dots. This

**Table 4** Total number of images for Words, NWords, PAWs and NPAWs

| | Number of unique tokens | Total number of images |
|---|---|---|
| Words | 282,593 | 847,779 |
| Naked Words | 211,072 | 633,216 |
| PAWs | 66,725 | 200,175 |
| Naked PAWs | 32,804 | 98,412 |

**Table 5** The average letters per token, width and height in pixels of Words, NWords, PAWs and NPAWs

| Token | Average | | |
|---|---|---|---|
| | Letters/token | Width | Height |
| Words | 4.7 | 93.1 | 48.1 |
| Naked Words | 4.7 | 93.1 | 47.0 |
| PAWs | 2.0 | 39.5 | 38.9 |
| Naked PAWs | 2.0 | 39.4 | 38.0 |

is also recorded in the case of the PAWs and NPAWs. The ratio between the average widths of the Words and NWords on one side and PAWs and NPAWs on the other is almost the same as in the case of the number of letters per token. The average height of words and NWords is greater than that of PAWs and NPAWs.

### 4.5 Degrading the computer-generated token images

The process of degrading the computer-generated token images is very important if they are to simulate the characteristics of real document images during OCR development. The degradation stage is composed of two parts: in the first, the image is skewed to simulate the distortion that may occur during document scanning; in the second, artificial noise is added to simulate scanned image data.

#### *4.5.1 Skewing the image*

Rotated images are created from the computer-generated image to simulate the rotation that often occurs while scanning documents. The application provided can generate any number of rotated samples of the original image rotated in both sides. It uses the Box–Muller transform algorithm to convert the uniform distribution random number to normal distribution random number [40]. This algorithm is used to simulate the real-life rotation angles of scanned documents. Nearest neighbor interpolation completes the rotated image [41]. Figure 2b shows the skewed token image.

#### *4.5.2 Adding noise*

Artificial noise is added to the image to simulate that found in real document images. Three effects are applied; blurring, filter and additive noise [42]. A Gaussian blurring effect is first added to simulate the blur that may be introduced during scanning. A high-pass filter is then applied to simulate change in the paper color over time. Gaussian noise is finally added to simulate image acquisition noise. Figure 2c shows a token image after applying the Gaussian blurring and noise.

### 4.6 Ground truth

In document image analysis and recognition, *ground truth* refers to various attributes associated with the text on the

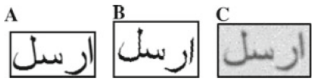
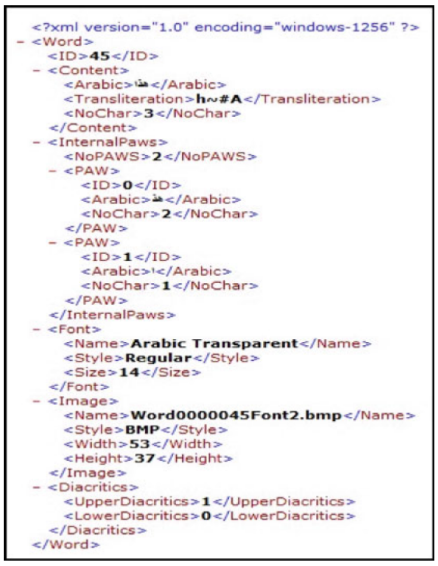


**Fig. 2** **a** an original token "ارسل" image, **b** the token image after skewing and **c** the image after adding Gaussian blurring and Gaussian noise



```
  <?xml version="1.0" encoding="windows-1256" ?>
- <Word>
    <ID>45</ID>
  - <Content>
      <Arabic>هذا</Arabic>
      <Transliteration>h~#A</Transliteration>
      <NoChar>3</NoChar>
    </Content>
  - <InternalPaws>
      <NoPAWS>2</NoPAWS>
    - <PAW>
        <ID>0</ID>
        <Arabic>هذ</Arabic>
        <NoChar>2</NoChar>
      </PAW>
    - <PAW>
        <ID>1</ID>
        <Arabic>ا</Arabic>
        <NoChar>1</NoChar>
      </PAW>
    </InternalPaws>
  - <Font>
      <Name>Arabic Transparent</Name>
      <Style>Regular</Style>
      <Size>14</Size>
    </Font>
  - <Image>
      <Name>Word0000045Font2.bmp</Name>
      <Style>BMP</Style>
      <Width>53</Width>
      <Height>37</Height>
    </Image>
  - <Diacritics>
      <UpperDiacritics>1</UpperDiacritics>
      <LowerDiacritics>0</LowerDiacritics>
    </Diacritics>
  </Word>
```

**Fig. 3** Ground truth XML file for the word "هذا"

image such as the size of tokens, characters, font type, font size, etc. Ground truth data is crucial to the systematic training and testing of document image analysis applications [43].

Each token image in the MMAC contains ground truth information [18]. Figure 3 shows the ground truth XML file for the word "هذا". MMAC's ground truth XML files contain the following attributes:

1. *Content:* this contains the token name in Arabic, transliteration of the token text to English—the transliteration method is shown below—and the number of characters in the token.
2. *Internal PAWs:* this shows the number of PAWs of the Word or Nword. It contains, as a sub-attribute, the content of the PAWs, these contents are the ID, Arabic name and number of characters.
3. *Font:* this contains the name of the font, style and font size in points.
4. *Image:* this contains the image file name, the file format of the image and the size of the image in pixels.
5. *Diacritics:* this contains the number of letters in the token that have upper diacritics and that have lower diacritics.

## 5 Building a multi-modal corpus: real images

The procedures described in Sect. 4 produce a multi-modal Arabic corpus from text data, with images being created artificially from that data. The result is a set of files containing descriptions of key sections of Arabic text, with corresponding image files showing its appearance in a number of common fonts. This dataset provides both a representation of the Arabic language and data needed during development of OCR applications—sample images of text with associated XML ground truth information.

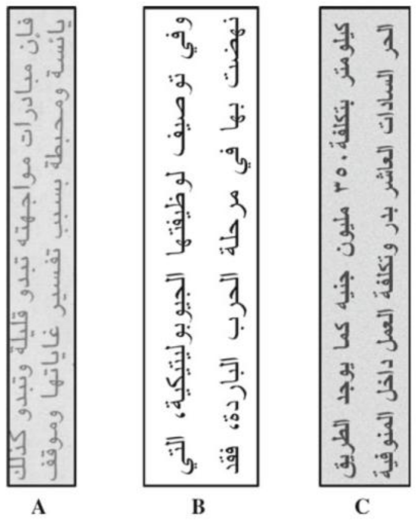
The images generated by the method described earlier are, however, idealized. As Fig. 1 showed, the token images are so clear that they can appear as if they were typed. Though the applications described in Sect. 4.5 allow common distortions and noise processes to be simulated, a complete evaluation should include real images containing the errors, noise and distortions that arise in real life.

To provide the data needed to support OCR, two different datasets are added to MMAC. The first contains images of small paragraphs of Arabic documents with text files providing ground truth. The second dataset is created from real scanned images. These datasets can be used to check the validity of the output of a developing OCR application. They are also valuable during development of preprocesses such as skew detection and correction, noise removal, binarization, thinning and layout analysis.

### 5.1 Paragraph image dataset

MMAC's paragraph document dataset contains images of 552 paragraphs. It is generated from 15 full-page documents. There are three different categories: real scanned images, computer-generated images and computer-generated images with artificial noise. Each category contains 5 full pages. Each single paragraph image includes around two lines containing around 10 words. The number of images are 223 real, 141 computer generated and 188 computer generated with noise. Figure 4 shows a sample paragraph image from each category. The computer-generated and computer-generated with noise documents include different Arabic font type, sizes and styles (regular, italic and bold). This dataset includes around 8,000 words.

The need for this dataset in the OCR development process is to test the application with a small sample of data enabling us to trace any errors. Interestingly during the testing of MMAC with commercial software, the recognition accuracy of a paragraph image is sometimes different from that of the full-page image.



A

فإن مبادرات مواجهته تبدو قليلة وتبدو كذلك
بائسة ومحبطة بسبب تفسير غاياتها وموقف

B

وفي توصيف لوظيفتها الجيوبوليتيكية، التي
نهضت بها في مرحلة الحرب الباردة، فقد

C

كيلومتر بتكلفة ٣٥٠ مليون جنيه كما يوجد الطريق
الحر السادات العاشر بدر وتكلفة العمل داخل المنوفية

**Fig. 4** Samples of the three categories of paragraph images. **a** Real image; **b** Computer image; **c** Noise image

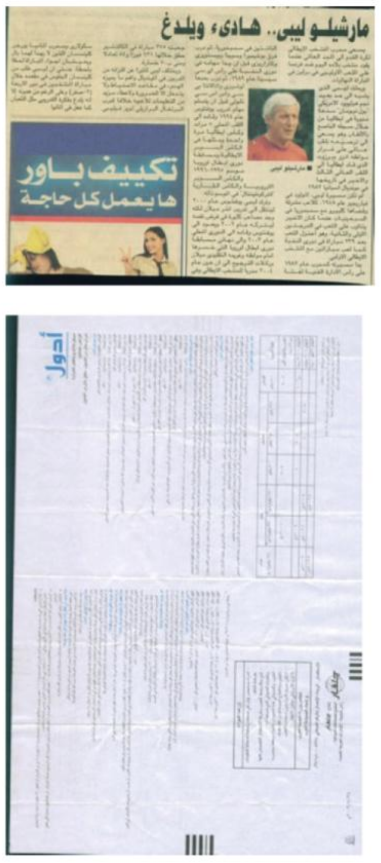

مارشيلو ليبي.. هادىء ويلدغ
تكييف باور
هايعمل كل حاجة

**Fig. 5** Two samples of real document images. The first is a newspaper image with some skew and images imposed inside the document. The second is a medicine script with some skew and includes tables. This image is *rotated* 90 degrees

### 5.2 Full-page real image dataset

MMAC's scanned document dataset contains images of 19 different documents. These comprise books, work documents, medical scripts, faxes, magazines and newspapers. The documents were scanned at 300DPI, in 24bit RGB color mode and stored in tiff file format. Document quality varies significantly over the dataset, which includes documents at different orientations and some skewed examples. It contains different sizes of documents. Some of the documents are watermarked, others have large page borders. Figure 5 shows two examples. Though the scanned document data set is currently small compared to the text-based data, the goal is to include examples of most (commonly) occurring real documents.

To create associated text files, the contents of each document were manually typed on Microsoft Word 2003 software. The resulting text files were proof read by two different groups of people, with the writing group different from each of the two proof-reading groups. The data set created from real images currently includes around 11,000 Arabic words.

## 6 Using the MMAC corpus in Arabic OCR development

MMAC was originally generated to support research in Arabic OCR. The primary motivation was to create a benchmark against which Arabic OCR applications could be evaluated. Its inclusion of images of different tokens makes MMAC suitable for training and testing OCR applications that operate in different layers. Some OCR applications deal with the whole word, while others deal with PAWs [44,45]. In the same manner, some applications use diacritics to search for particular words, while others neglect them [46]. We believe that MMAC can be used to evaluate any Arabic OCR application using data at four different levels that start from accurate computer-generated images and move to real images of full-page scanned documents. These four different levels represent different degrees of difficulty for an OCR application. As the application passes each level successfully and moves to the next, we gain more confidence in the quality of the application.

MMAC was tested with commercial software to check its usefulness in a real world application. The four different levels were tested using the well-known commercial software Readiris pro 10 [13]. Recognition results from Readiris were compared to the ground truth documents using the Levenshtein Distance algorithm [47]. The Levenshtein Distance algorithm is an approximate string matching algorithm that gives a tentative accuracy of recognition. This accuracy is measured based on the percentage of correct characters within the volume of characters covered, defined by the majority of the OCR application vendors [48]. Figure 6 explains the process applied to test MMAC with the commercial software. This process can be used to test any OCR engine with MMAC.

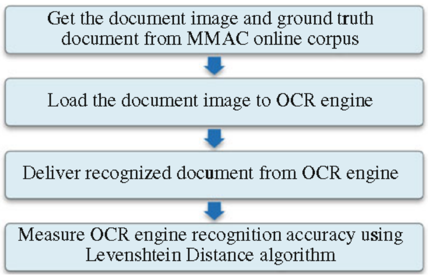

Get the document image and ground truth document from MMAC online corpus
Load the document image to OCR engine
Deliver recognized document from OCR engine
Measure OCR engine recognition accuracy using Levenshtein Distance algorithm

**Fig. 6** The typical process of testing any OCR engine using MMAC corpus

### 6.1 Computer-generated token images

In the first step, MMAC's computer-generated images are used to test recognition accuracy (Fig. 2a). The challenge of MMAC in this step is to maximize the number of token images that are covered by the developing application. In addition to the commonly used PAWs and Words, MMAC can be used to evaluate OCR applications that deal with NWords or NPAWs. As the token images are very clear, it is expected that many applications will be capable of high levels of performance. The importance of this step is to evaluate OCR application with a wide variety of tokens that represents most of the language. This step gives credibility to the OCR application. In the experiment reported here, the 1,274 most commonly used words were selected. These words represent approximately half of our corpus words. A recognition accuracy of 94% was achieved.

### 6.2 Computer-generated token images with artificial noise

In the second step, degraded images are generated from the clean computer-generated originals (Sect. 4.5). When an OCR application achieves high accuracy at this stage, it shows that the OCR application not only can detect most of the tokens in the language but also can detect them in the presence of realistic noise. The ability to control the noise added allows detailed analysis of the effect of different image conditions on any proposed method. Extreme values can also be used to test a given technique "to destruction".

In the exemplar experiment, three different levels of noise were used. These represent a good document, a medium document and a bad document. Table 6 gives the exact parameters used. The same 1,274 most common words from the previous step were examined. Recognition accuracy was 91%

**Table 6** Artificially added noise to computer-generated documents (Good, medium and bad documents)

| | Good document | Medium document | Bad document |
|---|---|---|---|
| Rotation angle | ±2° | ±6° | ±10° |
| Gaussian blurring | 10% | 13% | 15% |
| High pass filter | 35% | 45% | 50% |
| Gaussian noise | 40% | 65% | 75% |

for good documents, 89% for medium documents and 77% for bad documents.

### 6.3 Paragraph documents

In the third step, MMAC's small images of real scanned, computer-generated and computer-generated with artificial noise documents are used to test the OCR application. Figure 4 shows examples of the available images. We believe this step represents a higher level of testing than the previous two. It employs bigger images and different types of image. The recognition accuracy for this step gives a better indication of the likely quality of the OCR application given real-world input.

Sixty-two documents were selected from the three different types of documents that represent the majority cases. The recognition accuracy of the real documents ranged from 100% down to 47%. The recognition accuracy of the computer-generated documents ranged from 93% down to 33%, while that for computer generated with noise documents ranged from 95% down to 54%. The variety of accuracy values are based on the status of the document images and font properties. We found that the Italic font style gives bad accuracy results. We also found that Traditional Arabic font gives comparatively poor accuracy results. MMAC's breadth allows the effect of a wide range of text and image features to be assessed.

### 6.4 Real scanned documents

In the final step, MMAC's images of real scanned documents are used to test the OCR application. The real scanned images are ground trusted by the documents that were scanned. These images show a variety of document types from a wide range of sources. Figure 5 shows examples of the available images. Passing this step with reasonable recognition accuracy gives the final evaluation of the OCR application. This step tests not only the recognition accuracy but also the overall performance of the OCR application. It probes the effect of page skew, page layout analysis, page orientation, etc. We consider this step as the final step because the objective of any OCR application is to recognize full-page documents with different aspects.

During the testing of this step, we selected 9 different documents representing different clarity of images. The recognition accuracy achieved ranged from 95% for good documents down to 46% for very poor documents. This wide range of recognition accuracy indicates the usefulness of the large variety of images, documents and added noise provided by MMAC in testing OCR applications.

## 7 Statistical analysis of the MMAC corpus

Corpus analysis is concerned with the statistical properties of words and other lexical tokens. In this section, we describe an investigation into the frequency of these entities in Arabic.

Previous studies of Arabic corpora only analyzed words. We emphasize the analysis of other tokens, namely NWords, PAWs and NPAWS. This is achieved by studying the most and least frequently occurring tokens. We compared our analysis of words with previous research to verify our results.

### 7.1 Frequency list

A frequency list gives the number of occurrences of each element (usually word) in a corpus and is an important part of any corpus [49]. It is useful for information retrieval, text categorization and numerous other purposes [50]. Frequency lists are very useful to the OCR application developer as they can be used to validate the accuracy of the recognition phase. Comparison of the frequency of each word in the corpus and in the system output is an important step when assessing the accuracy of a recognition process. The frequency lists in

**Table 7** Statistical summary for Words, NWords, PAWs and NPAWs

| | Total number of tokens | Number of unique tokens | Average repetition of tokens |
|---|---|---|---|
| Words | 6,000,000 | 282,593 | 21.23 |
| Naked Words | 6,000,000 | 211,072 | 28.43 |
| PAWs | 14,025,044 | 66,725 | 210.19 |
| Naked PAWs | 14,025,044 | 32,804 | 427.54 |

**Table 8** The average number of characters per Word, characters per PAW and PAWs per Word

| | Average |
|---|---|
| Characters/Word | 4.74 |
| Characters/PAW | 2.03 |
| PAWs/Word | 2.33 |

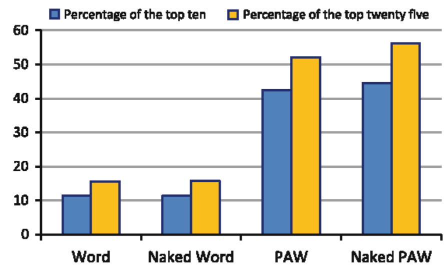


**Fig. 7** The percentage relationship between the top ten and top twenty-five most repeated Words, NWords, PAWs and NPAWs

**Table 9** Total number of Words, NWords, PAWs and NPAWs that occur only once in relation to the total unique number and the total number of tokens

| | Total number of tokens that occur once | Percentage to the total number of unique tokens (%) | Percentage to the total number of tokens (%) |
|---|---|---|---|
| Word | 107,771 | 38.14 | 1.796 |
| Naked Word | 69,506 | 32.93 | 1.158 |
| PAW | 20,239 | 30.27 | 0.144 |
| Naked PAW | 8,201 | 24.92 | 0.058 |

MMAC are not limited to words, but also include NWords, PAWs and NPAWs. The variety in the types of token used here gives flexibility to the developers of OCR applications, who can use the MMAC to support a wide range of application evaluations. The file format is a text file; each line contains a word and the number of occurrence of this word sorted alphabetically. We have 4 separate files in this format: for words, NWords, PAWs and NPAWs.

## 7.2 Analysis of words and PAWs

Tables 7 and 8 show the detailed analysis of the words and PAWs in MMAC. The most common Words are prepositions, while the most common PAWs are those that consist of a single letter. There are 25% fewer unique NWords than unique Words. The number of NWords that occur once or are repeated twice is less than in the case of Words, while NWords that are repeated heavily are more common than in the case of unique Words. The number of unique PAWs is very limited relative to the total number of PAWs. More PAWs are heavily repeated than Words. There are 50% fewer unique NPAWs than unique PAWs. The number of NPAWs that have few repetitions is less than in the case of PAWs, although the average number of repeated NPAWs is greater than that of PAWs.

The total number of one-letter PAWs is 6,275,167 that represents 44.7% of the total number of PAWs; while there are 3,652,709 two-letter PAWs, 26% of the total. The total number of NPAWs in the corpus is 14,025,044, while the total number of unique NPAWs is 32,804 with an average repetition of 427.54 for each NPAW.

## 7.3 Discussion

It is important to measure the frequency of occurrence of the Words, NWords, PAWs and NPAWs making up the Arabic language. Frequency data aids the development of

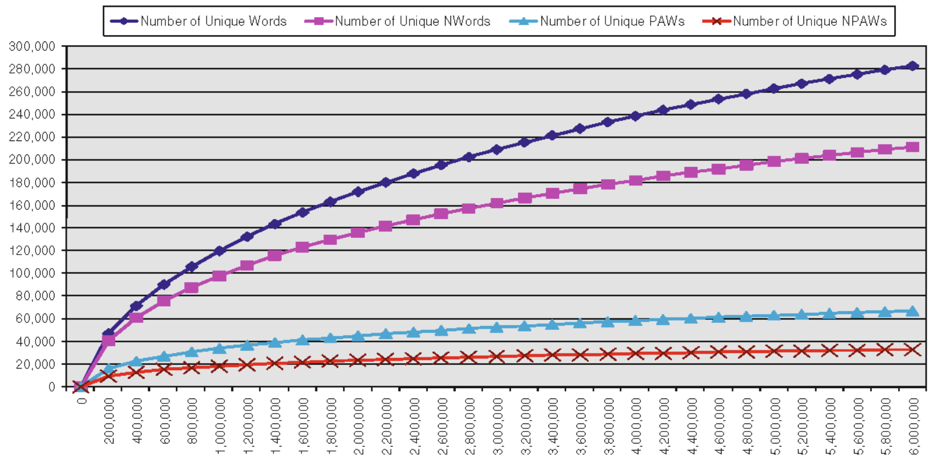


**Fig. 8** The relationship between the number of distinct Words, NWords, PAWs and NPAWs, and the number of words in the corpus as the total number of samples increases

**Table 10** The 25 most repeated Arabic Words, NWords, PAWs and NPAWs. Number and percentage of repetitions

| Serial | No. of Repetitions | Word | Percentage | No. of Repetitions | Naked Word | Percentage | No. of Repetitions | PAW | Percentage | No. of Repetitions | Naked PAW | Percentage |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 164,165 | في | 2.73% | 164,193 | فى | 2.73% | 2,921,359 | ا | 20.81% | 2,921,359 | ا | 20.84% |
| 2 | 139,380 | من | 2.32% | 139,380 | من | 2.32% | 835,979 | و | 5.96% | 840,793 | و | 5.99% |
| 3 | 82,429 | ان | 1.37% | 82,429 | ان | 1.37% | 445,282 | ر | 3.17% | 511,935 | ر | 3.65% |
| 4 | 78,417 | علي | 1.30% | 78,427 | على | 1.30% | 397,210 | لا | 2.83% | 397,210 | لا | 2.83% |
| 5 | 40,431 | الي | 0.67% | 40,431 | الى | 0.67% | 300,775 | ن | 2.14% | 337,855 | د | 2.41% |
| 6 | 40,091 | هذا | 0.66% | 40,114 | هدا | 0.66% | 271,891 | د | 1.94% | 300,775 | ن | 2.14% |
| 7 | 37,830 | لا | 0.63% | 37,830 | لا | 0.63% | 245,657 | ه | 1.75% | 285,629 | ب | 2.03% |
| 8 | 34,897 | او | 0.58% | 34,897 | او | 0.58% | 198,275 | ت | 1.41% | 245,657 | ه | 1.75% |
| 9 | 34,197 | ما | 0.57% | 34,197 | ما | 0.57% | 176,106 | في | 1.25% | 226,566 | با | 1.61% |
| 10 | 29,896 | عن | 0.49% | 30,315 | البى | 0.50% | 164,317 | ي | 1.17% | 180,173 | فى | 1.28% |
| 11 | 29,793 | التي | 0.49% | 29,900 | عن | 0.49% | 159,823 | ل | 1.14% | 166,129 | ى | 1.18% |
| 12 | 25,759 | اي | 0.42% | 25,759 | اى | 0.42% | 151,009 | من | 1.07% | 159,823 | ل | 1.14% |
| 13 | 21,363 | الذي | 0.35% | 21,413 | الدى | 0.35% | 115,682 | م | 0.82% | 151,009 | من | 1.07% |
| 14 | 20,010 | كل | 0.33% | 20,010 | كل | 0.33% | 115,372 | با | 0.82% | 149,501 | به | 1.06% |
| 15 | 19,576 | هذه | 0.32% | 19,584 | هده | 0.32% | 102,560 | ء | 0.73% | 115,682 | م | 0.82% |
| 16 | 18,566 | هو | 0.30% | 18,566 | هو | 0.30% | 93,731 | ما | 0.66% | 115,436 | بر | 0.82% |
| 17 | 16,738 | كان | 0.27% | 16,738 | كان | 0.27% | 81,895 | علي | 0.58% | 102,560 | ء | 0.73% |
| 18 | 16,467 | مع | 0.27% | 16,468 | مع | 0.27% | 76,162 | يه | 0.54% | 101,839 | بد | 0.72% |
| 19 | 16,317 | لم | 0.27% | 16,317 | لم | 0.27% | 75,344 | ب | 0.53% | 96,635 | لد | 0.68% |
| 20 | 14,065 | اذا | 0.23% | 15,910 | بعد | 0.26% | 69,987 | كا | 0.49% | 93,859 | بو | 0.67% |
| 21 | 13,975 | بعد | 0.23% | 14,602 | ابه | 0.24% | 69,779 | لي | 0.49% | 93,731 | ما | 0.66% |
| 22 | 12,944 | ولا | 0.21% | 14,081 | ادا | 0.23% | 66,653 | ز | 0.47% | 82,003 | على | 0.58% |
| 23 | 12,863 | ذلك | 0.21% | 12,944 | ولا | 0.21% | 65,964 | ذ | 0.47% | 74,079 | فا | 0.52% |
| 24 | 11,750 | بين | 0.19% | 12,873 | دلك | 0.21% | 63,088 | هذ | 0.45% | 69,987 | كا | 0.49% |
| 25 | 11,688 | انه | 0.19% | 12,023 | بين | 0.20% | 59,907 | لو | 0.42% | 69,791 | لى | 0.49% |

OCR applications by encouraging the OCR developer to recognize more common elements first. For example, the PAW "أ" represents more than 20% of the written Arabic language. As Fig. 7 shows, this percentage increases dramatically when measured in NPAWs instead of words. The 25 most repeated NPAWs account for more than half the words used in practice.

Statistical analysis shows that the number of unique NPAWs in the MMAC corpus is very limited. The number of unique NWords is almost six times that of NPAWs. Moreover, this increases rapidly, with no evident asymptote, as new texts are added to the corpus as shown in Fig. 8. The number of unique NWords is about 80% of the number of unique Words, and this ratio remains fairly stable once a reasonably sized corpus has been established.

Analysis of MMAC also shows the least repeated Words and PAWs in the language. Table 9 shows the relationship between the total number of Words, NWords, PAWs and NPAWs that occur only once; as percentages of the total number of distinct words and the total number of words, NWords, PAWs and NPAWs in the corpus. The twenty-five most repeated Words, NWords, PAWs and NPAWs and the percentage of each of them in the corpus are shown in Table 10. The analysis of words in Table 10 gives almost the same results as that found by others [49,51,52], which gives credibility to our analysis. The analysis of NWords, PAWs and NPAWs in Table 10 is very important to the OCR developer in the sense of giving more emphasis not only to the most repeated words but also the most repeated NWords, PAWs and NPAWs. Tokens that are repeated most should be recognized before those that appear only rarely. According to the previous statistics, we can advise the OCR developers to consider recognizing the one-letter PAWS in their isolated shape first.

The other important advice to the OCR developer is to deal with the two-letters PAW as if they are isolated letters or new glyphs. These new glyphs present just 612 different shapes. If the OCR developers recognize these isolated letters and two-letter glyphs, they can process 70% of the language.

## 8 Testing the corpus' validity

Testing the validity and accuracy of the data is a very important issue in creating any corpus [53]. Our testing process started by collecting a testing dataset from unusual sources consisting of scanned images (as mentioned earlier in Sect. 5) from faxes, documents, books, medicine scripts and well-known Arabic news websites. This was done one year after the original MMAC data collection was completed. The testing dataset was collected from sources that intersected minimally with those of the corpus. The total number of Words in the testing dataset is 69,158 and the total number of PAWs is 165,501. Table 11 shows the total and the number of distinct Words, NWords, PAWs and NPAWs of the testing dataset.

**Table 11** The total number of testing dataset Words, NWords, PAWs and NPAWs with their unique number

| | Testing dataset total number of tokens | Testing dataset unique number of tokens |
|---|---|---|
| Word | 69,158 | 17,766 |
| Naked Word | 69,158 | 16,513 |
| PAW | 165,518 | 7,272 |
| Naked PAW | 165,518 | 4,749 |

**Table 12** Total number and percentage of unique found Words, NWords, PAWs and NPAWs of the testing dataset from the corpus

| | Total number of unique tokens found | Percentage of total unique tokens found (%) |
|---|---|---|
| Word | 15,967 | 89.8 |
| Naked Word | 15,163 | 91.8 |
| PAW | 6,983 | 96 |
| Naked PAW | 4,629 | 97.5 |

**Table 13** Total number and percentage of missed Words, NWords, PAWs and NPAWs of the testing dataset from the corpus

| | Total number of missed tokens | Percentage of total missed tokens (%) |
|---|---|---|
| Word | 2,457 | 3.5 |
| Naked word | 1,843 | 2.66 |
| PAW | 1,600 | 0.96 |
| Naked PAW | 1,358 | 0.82 |

### 8.1 Testing words and PAWs

A program was developed to search in the corpus binary data files using a binary search algorithm. The percentage of unique words in the testing dataset found in the corpus data file is 89.8%. Table 12 gives the percentage of tokens found from the testing dataset in the corpus data files. This analysis deals with the unique data only. On the other hand, the total testing dataset words missing (that are not unique) in the corpus data files are 2,457 Words with a percentage of 3.5%. Table 13 shows this ratio in the corpus tokens.

### 8.2 Checking corpus accuracy

The testing dataset was also used to check the accuracy of the statistics obtained from the corpus. By extrapolation, we believe that if the curve between the number of words and the unique number of words (Fig. 8) is extended according to the number of words that exists in the testing dataset, we will find that the increase in the unique number of words is almost equal to the missing words in the corpus.

The shape of the curve between the total number of words and the unique number of words is a nonlinear regression model. We used the CurveExpert 1.3 shareware program [54] and applied its curve finder process. We found that the best regression model is the Morgan–Mercer–Flodin (MMF) model, from the Sigmoidal Family [55]. The best-fit curve equation is

$$y = \frac{ab + cx^d}{b + x^d} \tag{1}$$

where:

$a = -42.558444$ $b = 40335.007$

$c = 753420.21$ $d = 0.64692321$

$StandardError:$ $CorrelationCoefficient:$

$156.0703663$ $0.9999980$

Upon applying Eq. 1, we found that the increase in the number of words to 6,069,158 will increase the number of distinct words by 1,676, while the number of missing words is 1,799. This result supports our previous suggestion, with an accuracy of 93%.

## 9 The online MMAC corpus

In this section, we describe the contents of the MMAC corpus available via the project website. It also includes the method applied to transliterate the tokens text to English tokens name.

### 9.1 MMAC corpus contents

The contents of the online version of MMAC are as follows:

1. **Original Data File:** This contains the original sequence of words as collected. It can be used for defining word frequency pairs and is explained in Sect. 4.1 of this paper.
2. **Original Data File After Replacing Alef:** This file is the same as above after replacing Alef as explained in Sect. 4.2.
3. **Sample Raw Data Collected:** This contains samples of all types of files that are used in the corpus. The files are in their original formats (see Sect. 4.1).
4. **Data Files:** This folder contains four sub-folders for the main statistical analysis of the research. These folders are for Words, NWords, PAWs and NPAWs. Each of the previous folders (for example the Word folder) includes a randomized word list file, a Unique sorted Words list, a Word Repetition Counting list, the Top 50 repeated Words list and a Word Occurred Once list file. This part is explained in Sect. 7.
5. **MMAC tokens Images:** This folder contains four sub-folders for the images of the four different types of tokens as explained in Sects. 4.3 and 4.4. Each folder contains samples of the images of 50 tokens and their ground truth XML files.
6. **Paragraph documents Dataset:** This folder contains three sub-folders for the real images, computer images and noisy images. Each folder contains the images and the truth text (see Sect. 5.1).
7. **Real Scanned Dataset:** This folder contains two sub-folders for the typed and scanned documents. Each folder contains 19 documents (see Sect. 5.2).
8. **Corpus testing data:** This folder contains four sub-folders for the computer tokens tests, noise tokens test, paragraph tests and full-page tests. (see Sect. 6).
9. **Frequency List:** This folder contains the frequency lists of the Words, NWords, PAWs, NPAWs. The lists are sorted alphabetically. Each line contains a word with its frequencies (Sect.7.1).
10. **Applications:** This folder contains all the applications used to support the generation of the corpus and degrade images (Sect. 4).
11. **All Statistics.xls:** This is an Excel spreadsheet file containing all the data sheets and charts that have been used in the analysis of the corpus data.

### 9.2 Transliteration encoding

For flexibility and ease of use, we need to use Roman character-based letters for the Arabic image text. We used the Arabic English transliteration encoding to name the Arabic text in English. We explain here the different types of Arabic English transliteration encoding available. Transliteration is a mapping between two different languages, in our case Arabic and English [56]. There are many types of transliteration between Arabic and English which can be summarized into two main categories: Arabic Transliteration [57] and Arabic Chat Alphabets [58].

We believe that using Arabic transliteration encoding is much better in our case than Arabic Chat Alphabets—although "chat" is more readable to Arabic natives for the following reasons:

1. The Arabic Chat Alphabets neglect letters with diacritics like (آ، أ، إ، ؤ، ئ) [57].
2. The Arabic Chat Alphabets sometimes use two characters to transliterate one character.
3. The Arabic Chat Alphabets sometimes use special characters to transliterate [58].

The most famous Arabic Transliteration encoding is the Buckwalter Transliteration as shown in Table 14 [59].

**Table 14** Buckwalter transliteration encoding

| Transliterated Character | Arabic Character | Transliterated Character | Arabic Character | Transliterated Character | Arabic Character | Transliterated Character | Arabic Character |
|---|---|---|---|---|---|---|---|
| ' | ء | t | ت | s | س | f | ف |
| \| | آ | v | ث | $ | ش | q | ق |
| > | أ | j | ج | S | ص | k | ك |
| & | ؤ | H | ح | D | ض | l | ل |
| < | إ | x | خ | T | ط | m | م |
| } | ئ | d | د | Z | ظ | n | ن |
| A | ا | * | ذ | E | ع | h | ه |
| b | ب | r | ر | g | غ | w | و |
| p | ة | z | ز | - | ـ | Y | ى |
|  |  |  |  |  |  | y | ي |

**Table 15** Buckwalter modified transliteration encoding

| Transliterated Character | Arabic Character | Transliterated Character | Arabic Character | Transliterated Character | Arabic Character | Transliterated Character | Arabic Character |
|---|---|---|---|---|---|---|---|
| ' | ء | t | ت | s | س | f | ف |
| i | آ | v | ث | $ | ش | q | ق |
| a | أ | j | ج | #S | ص | K | ك |
| @ | ؤ | #H | ح | #D | ض | l | ل |
| e | إ | x | خ | #T | ط | m | م |
| } | ئ | d | د | #Z | ظ | n | ن |
| #A | ا | ~ | ذ | #E | ع | h | ه |
| b | ب | r | ر | g | غ | w | و |
| p | ة | z | ز | - | ـ | #Y | ى |
|  |  |  |  |  |  | y | ي |

We decided to use this encoding (37 characters) with modifications for the following reasons:

1. Buckwalter uses a single character for transliteration.
2. Buckwalter includes all written Arabic characters and also diacritics like (ـً ئ ؤ إ أ آ) [60].
3. Buckwalter uses fewer capital letters (it only uses A, H, S, D, T, Z, E, Y).

## 9.3 Buckwalter modified transliteration encoding

The purpose of transliteration in our case is different from most other transliteration. We use transliteration encoding for many purposes, including file names, so we have to deal with all the restrictions of the use and also file name restrictions in the most common operating systems (Windows, Macintosh and UNIX). These restrictions are as follows:

*1.* *The file name may be case insensitive*. Hence, we changed the eight letters used in Buckwalter (A, H, S, D, T, Z, E, Y) by putting the character '#' before them.
*2.* *White space is not allowed*. This restriction is not applicable in our case.
*3.* Some special characters like (|, >, *) are not allowed. So we changed the transliteration for the following:

   1. 'آ' from '|' to be 'i'.
   2. 'أ' from '>' to be 'a'.
   3. 'ؤ' from '&' to be '@'.
   4. 'إ' from '<' to be 'e'.
   5. 'ذ' from '*' to be '~'.

The Modified Buckwalter Transliteration is almost the same as the original but with 13 of the original 37 characters modified. Table 15 shows the characters and their proposed transliteration characters.

## 10 Future works and conclusion

Creating a corpus is a large and complex task. While currently extremely useful, the MMAC corpus would benefit from additional work. First, saving the data files in a hypertext mark-up formatting language and adding other information for each word, such as the morphological information explained in Sect. 2.3. Secondly, adding statistical analysis regarding the words' frequency pair lists. Thirdly, a powerful stemming/lexical algorithm must be applied to the corpus to enhance data retrieval for improved accuracy. Fourthly, the process of adding new words should continue. Fifthly, it would be advantageous if MMAC could be extended to include other languages that use the Arabic alphabet. Finally, it must be capable of including words with diacritics.

After considering the extensions proposed earlier, we conclude that an organization dealing with Arabic corpora is urgently required. Also, using Unicode as a standard code page is very important because unification among various languages is a significant problem in automating the processing of non-Latin languages. Statistical analysis shows the importance of PAWs and NPAWs in the Arabic language; these should be included in future corpora.

To conclude, the MMAC corpus was designed to meet the needs of users of traditional linguistic corpora, but at the same time to be beneficial to OCR applications developers. It can be used in testing and training each phase of an Arabic OCR application.

## References

1. Kučera, H., Francis, W.N.: Computational analysis of present-day American English. Int. J. Am. Linguist. **35**(1), 71–75 (1967)
2. Davies, M.: (1990-present) The corpus of contemporary American english (COCA), 410+ Million words. http://www.americancorpus.org (2008)
3. The British National Corpus: Oxford University. http://www.natcorp.ox.ac.uk (2005)
4. Time: Time archive 1923 to present. http://www.time.com/time/archive/ (2008)
5. Knight, D., Bayoumi, S., Mills, S., Crabtree, A., Adolphs, S., Pridmore, T., Carter, R.: Beyond the text: building and analysing multi-modal corpora. In: 2nd International Conference on E-Social Science. Manchester, UK (2006)
6. Indexes : United nations documentation.: the Department of Public Information (DPI), Dag Hammarskjöld Library (DHL). http://www.un.org/Depts/dhl/resguide/itp.htm (2007)
7. Internet world users by language: Top ten languages used in the web. Internet World Stats, Usage and Population Statistics. http://www.internetworldstats.com/stats7.htm. Accessed 22-01-07 (2007)
8. UCLA TUoC, Los Angeles: Arabic. International Institute, Center for World Languages, Language Materials Project. http://www.lmp.ucla.edu/Profile.aspx?LangID=210&menu=004 (2006)
9. Hamada, S.: نحو منهج مقترح لصناعه المدونات اللغويه . In: The Seventh Conference on Language Engineering. Cairo, Egypt (2007)
10. Lorigo, L.M., Govindaraju, V.: Offline Arabic handwriting recognition: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **28**(5), 712–724 (2006)
11. Amin, A.: Off line Arabic character recognition—a survey. In: The Fourth International Conference on Document Analysis and Recognition, pp. 596–599. Ulm, Germany (1997)
12. AbdelRaouf, A., Higgins, C., Khalil, M.: A database for Arabic printed character recognition. In: The International Conference on Image Analysis and Recognition-ICIAR 2008, Póvoa de Varzim, Portugal, pp. 567–578 (2008)
13. IRIS: Readiris pro 10 (2004)
14. Parker, R., Graff, D., Chen, K., Kong, J., Maeda, K. : Arabic Gigaword. Linguistic Data Consortium, University of Pennsylvania, Philadelphia (2009)
15. CLARA (Corpus Linguae Arabicae): Charles University, Prague (2001)
16. Al-Hayat newspaper, Al-Hayat Arabic data set, University of Essex, in collaboration with the Open University
17. An-Nahar newspaper: An-Nahar text corpus (2000)
18. Slimane, F., Ingold, R., Kanoun, S., Alimi, A.M., Hennebert, J.: A new Arabic printed text image database and evaluation protocols. In: 10th International Conference on Document Analysis and Recognition, pp. 946–950. Barcelona, Spain (2009)
19. Abbes, R., Dichy, J., Hassoun, M.: The architecture of a standard Arabic lexical database. Some figures, ratios and categories from the DIINAR.1 source program. In: Workshop of Computational Approaches to Arabic Script-based Languages, pp. 15–22. Geneva, Switzerland (2004)
20. Beesley, K.R.: Arabic finite-state morphological analysis and generation. In: 16th International Conference on Computational Linguistics, pp. 89–94. Copenhagen (1996)
21. Alansary, S., Nagi, M., Adly, N.: Building an International Corpus of Arabic (ICA): progress of compilation stage. In: The Seventh Conference on Language Engineering. Cairo, Egypt (2007)
22. Wynne, M.: Corpus and text—basic principles. In: Developing Linguistic Corpora: A Guide to Good Practice. Oxbow Books, Oxford. Available online from http://ahds.ac.uk/linguistic-corpora/ (2005)
23. Dash, N.S., Chaudhuri, B.B.: Why do we need to develop corpora in Indian languages? In: the International Working Conference on Sharing Capability in Localisation and Human Language Technologies SCALLA-2001. Bangalore (2001)
24. Al-Shalabi, R., Evens, M.: A computational morphology system for Arabic. In: Workshop on Computational Approaches to Semitic Languages COLING-ACL98, pp. 66–72. Montreal (1998)
25. The Unicode consortium: Arabic, range: 0600-06ff. The Unicode Standard, Version 5 (2007)
26. The Unicode consortium: The Unicode standard, version 4.1.0. In: pp. 195–206. Boston, MA, Addison-Wesley (2003)
27. The Unicode consortium: Arabic shaping The Unicode Standard, Version 5 (2006)
28. Khorsheed, M.S.: Off-line Arabic character recognition—a review. Pattern Anal. Appl. **5**(1), 31–45 (2002)
29. Fahmy, M.M.M., Ali, S.A.: Automatic recognition of handwritten Arabic characters using their geometrical features. J. Stud. Inform. Control Emphasis Useful Appl. Adv. Technol. **10**(2), 81–98 (2001)
30. Buckwalter, T.: Issues in Arabic orthography and morphology analysis. In: The 20th International Conference on Computational Linguistics, COLING 2004, pp. 31–34. Geneva, Switzerland (2004)
31. Harty, R., Ghaddar, C.: Arabic text recognition. Int. Arab. J. Inf. Technol. **1**(2), 156–163 (2004)
32. Contributors, W.: Code page. From wikipedia, the free encyclopaedia. http://en.wikipedia.org/w/index.php?title=Code_page&oldid=87192444. Accessed 22 /01/07 (2006)
33. Beebe, N.H.F.: Character set encoding. TUGboat **11**(2), 171–175 (1990)
34. arabo.com: Arabo Arab search engine and dictionary. http://www.arabo.com/. Accessed 12-01-07 (2005)
35. Ltd SP: Webzip 7.0. 7.0 edn (2006)
36. Larkey, L.S., Ballesteros, L., Connell, M.E.: Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis. In: 25th International Conference on Research and Development in Information Retrieval (SIGIR), pp. 269–274 (2002)
37. Kanungo, T., Resnik, P.: The bible, truth, and multilingual OCR evaluation. In: the SPIE Conference on Document Recognition and Retrieval VI, pp. 86–96. San Jose, CA (1999)
38. Chang, Y., Chen, D., Zhang, Y., Yang, J.: An image-based automatic Arabic translation system. Pattern Recognit. **42**(9), 2127–2134 (2009)
39. Kanoun, S., Alimi, A.M., Lecourtier, Y.: Affixal approach for Arabic decomposable vocabulary recognition: A validation on printed word in only one font. In: The Eight International Conference on Document Analysis and Recognition (ICDAR'05), pp. 1025–1029. Seoul, Korea (2005)
40. Box, G.E.P., Muller, M.E.: A note on the generation of random normal deviates. Ann. Math. Stat. **29**(2), 610–611 (1958)
41. Sonka, M., Hlavac, V., Boyle, R.: Image Processing: Analysis and Machine Vision, 2nd edition edn. Thomson Learning Vocational (1998)
42. Hartley, R.T., Crumpton, K.: Quality of OCR for degraded text images. In: The Fourth ACM Conference on Digital Libraries, pp. 228–229 Berkeley, California, United States (1999)
43. Leea, C.H., Kanungob, T.: The architecture of trueViz: a groundTRUth=metadata editing and vIsualiZing toolKit. Pattern Recognit. **36**(3), 811–825 (2003)
44. Mehran, R., Pirsiavash, H., Razzazi, F.: A front-end OCR for omni-font Persian/Arabic cursive printed documents. In: Digital Image Computing: Techniques and Applications (DICTA'05), pp 56–64. Cairns, Australia (2005)
45. Najoua, B.A., Noureddine, E.: A robust approach for Arabic printed character segmentation. In: Third International Conference on Document Analysis and Recognition (ICDAR'95), pp. 865–868. Montreal, Canada, (1995)

46. Bushofa, B.M.F., Spann, M.: Segmentation and recognition of Arabic characters by structural classification. Image Vis Comput. **15**(3), 167–179 (1997)
47. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. (CSUR) **33**(1), 31–88 (2001)
48. Tanner, S., Muñoz, T., Ros, P.H.: Measuring mass text digitization quality and usefulness. D-Lib Mag. **15**(7/8), (2009)
49. Buckwalter, T.: Arabic word frequency counts. http://www.qamus.org/wordlist.htm. Accessed 28/01/07 (2002)
50. Kilgarriff, A.: Using word frequency lists to measure corpus homogeneity and similarity between corpora. In: The 5th ACL Workshop on Very Large Corpora, pp. 231–245. Beijing and Hong Kong (1997)
51. AL-Ma'adeed, S., Elliman, D., Higgins, C.A.: A data base for Arabic handwritten text recognition research. In: Eighth International Workshop on Frontiers in Handwriting Recognition, pp. 485–489 Ontario, Canada (2002)
52. Pechwitz, M., Maddouri, S.S., Märgner, V., Ellouze, N., Amiri, H.: IFN/ENIT—Database of handwritten Arabic words. In: The 7th Colloque International Francophone sur l'Ecrit et le Document, CIFED 2002, pp. 129–136. Hammamet, Tunisia (2002)
53. Mashali, S., Mahmoud, A., Elnemr, H., Ahmed, G., Osama, S.: Arabic OCR database development. In: The Fifth Conference on Language Engineering, pp. 250–283. Cairo, Egypt (2005)
54. Hyams, D.G.: CurveExpert 1.3, a comprehensive curve fitting system for windows (2005)
55. Gu, B., Hu, F., Liu, H.: Modelling classification performance for large data sets, an empirical study. In: Advances in web-age information management: second international conference, waim 2001, pp. 317–328. xi'an, china (2001)
56. contributors, W.: Romanization of Arabic. From wikipedia, the free encyclopaedia. http://en.wikipedia.org/wiki/Arabic_transliteration. Accessed 27/01/07 (2006)
57. contributors, W.: Arabic chat alphabet. From wikipedia, the free encyclopaedia. http://en.wikipedia.org/wiki/Arabic_Chat_Alphabet. Accessed 27/01/07 (2006)
58. Palfreyman, D., Khalil, M.a.: A funky language for teenzz to use: representing gulf Arabic in instant messaging. J. Comput. Mediat. Commun. **9**(1) (2003)
59. Buckwalter, T.: Buckwalter Arabic transliteration. http://www.qamus.org/transliteration.htm. Accessed 28/01/07 (2002)
60. Ananthakrishnan, S., Bangalore, S., Narayanan, S.: Automatic diacritization of Arabic transcripts for automatic speech recognition. In: International Conference on Natural Language Processing. Kanpur, India (2005)