

Arabic Online
Handwriting Recognition

Cyrus Bakhtiari-Haftlang

2007

Abstract

There are more than 200 million Arabic speaking people in the world and twice as many use the Arabic script. Today's consumers have high demand for a natural interaction with their consumer electronics. This is many times made possible thanks to smart handwriting recognition systems. Such recognition products exist for interpretation of Latin based characters and Chinese and Japanese signs. The aim of this thesis was to adapt a handwriting engine to recognize Arabic letters. This was accomplished by gathering writing data from Arabs and using present utilities to process the data by clustering it using divisive hierarchical clustering. Several features were added to the existing recognition engine to eliminate ambiguity between problematic character pairs. The achieved recognition rate of the engine and the Arabic database landed on circa 95%. The database from this work could potentially be used in a commercial product in the future.

Chapter 1

Introduction

In the current era of communications, digital handwriting is gaining popularity and is becoming common in fields of applications previously not considered by the pioneers of digital handwriting. Modern game consoles, almost all handheld devices and mobile phones targeting the Asian market support digital handwriting. Such an input device usually consists of a special pen by which pressure is applied on a pressure-sensitive panel. However, besides the hardware these devices need a good handwriting recognition (HWR) algorithm in order to be useful. Many of these algorithms are based on Neural Networks and Hidden Markov Models (see [4]).

A recognition system can be either online or offline. The offline HWR is based on Optical Character Recognition (OCR) and is usually applied on scanned documents, e.g. archiving of articles. On the other hand, online HWR is much more elaborate, being able to recognize in real time where the pressure is applied on the instrument and when/where it ends. In other words it is about recognizing characters by analyzing the temporal sequence of points traced out by the pen hence being able to use additional information such as pen-lift-occurrences and amount of pressure made in each point.

Zi Corporation, a Canadian based company and one of the pioneers in mobile-based online HWR, is continuously developing products to cover most of the existing alphabets in use. Zi has released products such as Decuma Alphabetic, Decuma Japanese and Decuma Chinese. As this thesis is written in cooperation with the company, it aims to contribute to development of a future product covering the Arabic script. The term script will be used instead of alphabet when referring to Arabic characters due to differences in properties of this script compared to the Latin alphabet as explained later in the thesis.

Arabic is the fifth largest language in the world with between 186 and 422 million native speakers. Even though the spoken Arabic varies somewhat across regions, Modern Standard Arabic, the so called *Fuṣḥā*, is the standardized version used for official communication across the Arab world.

Chapter 2

The Arabic Script

2.1 The writing of the script

Arabic script is written from right to left and is a strictly cursive script meaning that almost all letters within a word or "sub word" are connected. A sub word can be described as a word that contains a letter which can not be connected to another letter within the same word. Due to the cursive properties of the script, some words are written in a single stroke i.e. without lifting the pen at all. The standard Arabic script contains 28 letters. Each letter has either two or four different shapes, with shape being dependant of the position of the letter within a word (see Table 2.1). The different positions are defined as: isolated, initial, medial and final form. Letters without initial or medial shapes can not be connected to the subsequent letter and hence constitute a sub word.

2.2 Extension and variations

Along with the spread of Islam and Qur'an, the Arabic script has spread widely and is used today in countries extending from Central Asia to North-Western Africa, see Figure 2.1. Therefore it is also used to write many other languages. Examples of non-Arabic languages written with the Arabic script include Persian, Urdu, Malay and Azerbaijani. In order to accommodate the phonetics of these languages, the script has been expanded with additional letters and symbols.

2.3 Impure Abjad

Historically all scripts belonging to the Semitic family of scripts are categorized as Abjads. In Abjad scripts, the vowels are not included in the written word meaning that only consonants are written. In modern Semitic scripts such as Hebrew, Syriac and Arabic, a lighter version of this rule still applies. In these so called impure Abjads, some consonants can act as long vowels. For example is the Arabic letter Waw pronounced both as a voiced labialized velar approximant (as "w" in the English word "weep") as well as a long close back rounded vowel (as "oo" in "boot"). All short vowels on the other hand are usually omitted in

<i>Unicode value</i>	<i>Unicode Name</i>	<i>Isolated</i>	<i>Final</i>	<i>Medial</i>	<i>Initial</i>
0627	Arabic letter Alef	ا	ا	-	-
0628	Arabic letter Beh	ب	ب	ب	ب
062A	Arabic letter Teh	ت	ت	ت	ت
062B	Arabic letter Theh	ث	ث	ث	ث
062C	Arabic letter Jeem	ج	ج	ج	ج
062D	Arabic letter Hah	ح	ح	ح	ح
062E	Arabic letter Khah	خ	خ	خ	خ
062F	Arabic letter Dal	د	د	-	-
0630	Arabic letter Thal	ذ	ذ	-	-
0631	Arabic letter Reh	ر	ر	-	-
0632	Arabic letter Zain	ز	ز	-	-
0633	Arabic letter Seen	س	س	س	س
0634	Arabic letter Sheen	ش	ش	ش	ش
0635	Arabic letter Sad	ص	ص	ص	ص
0636	Arabic letter Dad	ض	ض	ض	ض
0637	Arabic letter Tah	ط	ط	ط	ط
0638	Arabic letter Zah	ظ	ظ	ظ	ظ
063A	Arabic letter Ain	ع	ع	ع	ع
063B	Arabic letter Ghain	غ	غ	غ	غ
0641	Arabic letter Feh	ف	ف	ف	ف
0642	Arabic letter Qaf	ق	ق	ق	ق
0643	Arabic letter Kaf	ك	ك	ك	ك
0644	Arabic letter Lam	ل	ل	ل	ل
0645	Arabic letter Meem	م	م	م	م
0646	Arabic letter Noon	ن	ن	ن	ن
0647	Arabic letter Heh	ه	ه	ه	ه
0648	Arabic letter Waw	و	و	-	-
064A	Arabic letter Yeh	ي	ي	ي	ي

Table 2.1: The Arabic script

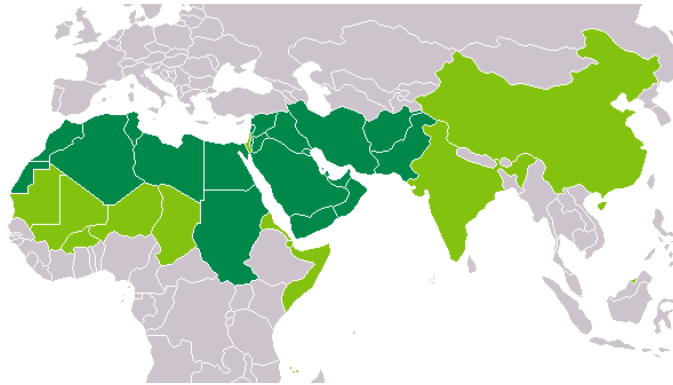


Figure 2.1: Worldwide distribution of the Arabic script showing countries where the Arabic script is the only official orthography (dark green) and countries in which the Arabic script is used alongside other orthographies (light green)

practice and only written in those cases where omission would cause ambiguity in the context. In some non-Semitic languages e.g. the Kurdish dialect of Sorani, which is mainly written using a modified version of the Arabic script, writing of all vowels has been made mandatory.

2.4 Diacritics

Diacritical marks or diacritics are markings which are written either above or below a letter. The diacritics Fatha, Damma, and Kasra indicate short vowels, Sukun indicates a syllable stop, and Fathatan indicates nunation¹ and can accompany Fatha, Damma, or Kasra, see Table 2.2. Diacritics indicating the short vowels are normally omitted from handwriting in most languages which implement the Arabic script.

◌َ	◌ُ	◌ِ	◌ْ	◌ً
<i>Fatha</i>	<i>Damma</i>	<i>Kasra</i>	<i>Sukun</i>	<i>Fathatan</i>

Table 2.2: Diacritical markings

Some other diacritics indicate doubled consonants and special vowels. Examples are Hamza, Shadda, and Madda, see Table 2.3. Madda together with Alef in isolated form is in Persian regarded as a lexically separate letter because of its common occurrence in the language indicating long open back rounded vowel (similar to "o" in "Boston").

¹Nunation is the addition of a final -n to a noun or adjective to indicate that it is fully declinable and syntactically unmarked for definiteness.

ا	ا	=	آ
<i>Alef with Hamza above</i>	<i>Alef with Hamza below</i>	<i>Shadda above</i>	<i>Alef with Madda above</i>

Table 2.3: Other common diacritical markings

2.5 Letter position

In the Arabic script, letters are connected at the same relative height. Thus, the baseline is the position of the height at which letters of the script are connected (see Figure 2.2). This baseline is analogous to the height in which words of the Latin alphabet sit. In handwriting however, the baseline is merely an ideal concept and a simplification of actual writing. In practice, connections occur near, but not necessarily on, such a line.

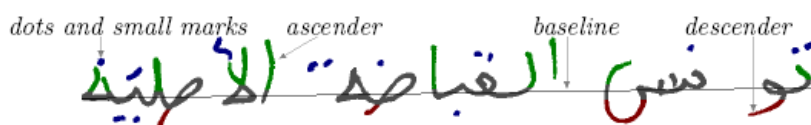


Figure 2.2: Some features of words. Baseline (black), dots and other small marks (blue), ascenders (green), descenders (red).

2.6 Numerals

Arabic-Indic numerals are a direct evolvement from the Brahmi numerals of India and were introduced to Europeans around the 10:th century giving birth to what today is referred to as European numerals see Table 2.4. There are two variants of the Arabic-Indic numerals; the Western and the Eastern, which differ in the shape of three digits. The Eastern Arabic-Indic numerals variant is primarily used in Persian and Urdu. There are several problems associated with Arabic-Indic numerals when working with word processing tools of which all are due to fact that the numerals are written from left-to-right as opposed to the letters.

European	0	1	2	3	4	5	6	7	8	9
Arabic-Indic	٠	١	٢	٣	٤	٥	٦	٧	٨	٩
Eastern Arabic-Indic	٠	١	٢	٣	٤	٥	٦	٧	٨	٩

Table 2.4: European and Arabic-Indic numerals

Chapter 3

Creating an HWR database

An HWR engine requires a database which contains reference information regarding the letter it is to interpret. In order for the engine to be fully generic i.e. be able to interpret as many variations as possible, the reference data should be based on as much diverse handwritings as possible. Data storage may however sometime be a limiting factor for small devices, see [2].

The database which is the focus of this thesis work was build up of handwriting of forty Arabic speaking/writing contributors with various backgrounds. In order to cover the writing variation of each contributor, every letter was asked to be written at least twice. This database was based on all the letters in the standard Arabic script in isolated form as well as some diacritics and punctuation marks. This approach meant that an engine based on this database would not be able to interpret anything cursively written, but would on the other hand require minimum changes to the already existing Decuma Alphabet-engine. The choice of isolated form in particular was to make the character writing analogous to the button tapings on an Arabic keyboard.

3.1 Collecting the data

The collecting of the data from each contributor was made with graphical pens and tablets (Wacom Intuos) connected to a PCs. The contributors were given time to familiarize themselves with the tools. Before the collecting starts the letters of interest were prepared via an XML document. The document was inputted in a tool developed by Zi Corp, named DAB. The tool acts as a recorder of the pen's movement on the tablet and also binds the registered strokes to a certain character. The collecting is a costly and time consuming activity, therefore it is important to register the strokes as accuratly possible using a high sample rate.

3.2 Dividing the material

The material from the contributors had to be divided into two parts. The larger part, based on 2/3 of the material, was used to train the database. The final database was thus constituted by this material. The purpose of the rest of the material, the so called test data, was to introduce the engine with unfamiliar

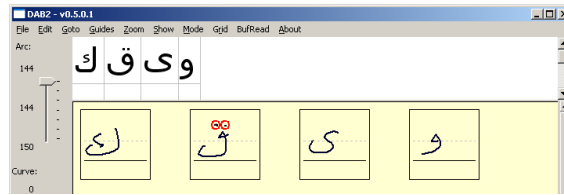


Figure 3.1: User interface of DAB.

data and be able to fine tune the database even further making it more robust. A naïve approach would be to make use of the test data as training data and expect that the database would become more accurate. But then any test result of the engine would become insignificant because then the engine would be able to recognize the test data unrealistically accurate.

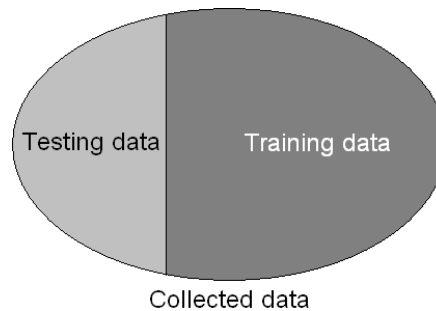


Figure 3.2: Most of the material is used to train the database. The rest is used to test the database.

3.3 Sampling

After all the data had been collected, it had to be cleaned up from unintentional pen strokes and character misplacements. Due to writing variations, the recorded strokes may be of various sizes. In order to create reference characters through clustering (see 3.4), the points of each stroke need to be normalized. The shapes also need to be resampled in order to be easily comparable during clustering. The resampling which used an interpolating technique, resulted in that each character consisted of 32 equidistant points. A higher resample rate would yield a more accurate transformation but in the expense of a larger database file.

3.4 Clustering

3.4.1 Fuzzy C-means

Fuzzy C-means is an algorithm which is commonly used in clustering [3]. The basic idea of this algorithm is that a data point may belong to more than one

cluster with different grades of membership ranging between 0 and 1. It is based on minimization of the following dissimilarity function:

$$J(U, c_1, c_2, \dots, c_c) = \sum_{i=1}^c J_i = \sum_{i=1}^c \sum_{j=1}^n u_{ij}^m d_{ij}^2,$$

where u_{ij} is the degree of membership of the point j in the cluster i ; c_i is the centroid (cluster center) of cluster i ; d_{ij} is the Euclidian distance between the i :th centroid (c_i) and the j :th data point; $m \in [1, \infty]$ is a weighting exponent. The membership matrix U is randomly initialized.

$$\sum_{i=1}^c u_{ij} = 1, \forall j = 1, \dots, n.$$

Fuzzy partitioning is carried out through an iterative optimization of the dissimilarity function taking into consideration the following conditions:

$$c_i = \frac{\sum_{j=1}^n u_{ij}^m x_j}{\sum_{j=1}^n u_{ij}^m}$$

and

$$u_{ij} = \frac{1}{\sum_{k=1}^c \left(\frac{d_{ij}}{d_{kj}}\right)^{2/(m-1)}},$$

where x_j denote the j :th data point. By updating the cluster centers and the relative grades of membership of each data point, the algorithm eventually moves the cluster centers to the "proper" location within that given data set. The iteration will stop when $\max_{ij} \{|u_{ij}^{k+1} - u_{ij}^k|\} < \delta$, where δ is a termination criterion between 0 and 1 and k is numbers of performed iteration steps. This algorithm does not guarantee that it converges to an optimal solution due to the fact that cluster centers are randomly initialized. A remedy would be to run the algorithm several times with different initial centroids or use an algorithm to determine all the centroids by e.g. evaluating the arithmetic means of all data points.

3.4.2 Hierarchical clustering

The slimmed version of the fuzzy c-means algorithm suffers from the fact that it can not ensure global minimum for the dissimilarity function due to random choice of the initial centroids. In the hierarchical clustering algorithm it the data points themselves which constitute the initial centroids. The steps of hierarchical clustering are as follows:

Step 1: Assign each data point $(1, 2, \dots, N - 1, N)$ an own cluster so that there are N clusters, each encompassing one single data point. Also set the cluster's Euclidian distance between one another according to the distance of the data points.

Step 2: Find the shortest distance between a pair of clusters and combine them into a single cluster.

Step 3: Calculate the distances between the new joined cluster and each of the other clusters.

Step 4: Reiterate steps 2 and 3 until all data points are clustered into one single cluster of size N .

Step 5: Cut out the $k - 1$ longest paths in the hierarchical tree to obtain k centroids.

Usually one of these three techniques is used to determine the distance between data points.

Single-linkage: The distance between two clusters is equal to the shortest distance between any member of one cluster to any member of the other cluster.

Complete-linkage: The distance between two clusters is equal to the greatest distance from any member of one cluster to any member of the other cluster.

Average-linkage: The distance between two clusters is equal to the average distance from any member of one cluster to any member of the other cluster.

This kind of hierarchical clustering is called agglomerative because it merges clusters iteratively. The process could reversely start with all data points being part of one cluster which then is subdivided into smaller pieces. Such an approach is usually referred to as divisive hierarchical clustering.

3.4.3 Clustering the material

The clustering algorithm which was used for the Arabic letters is an extension of the divisive hierarchical process. The only input it requires besides the data it self, is a threshold value μ which determines the size of the final clusters for the particular letter. In order to demonstrate the effect of choosing different values of μ . The following example (Figure 3.3) has been adapted from thesis work of David Danowsky [1].

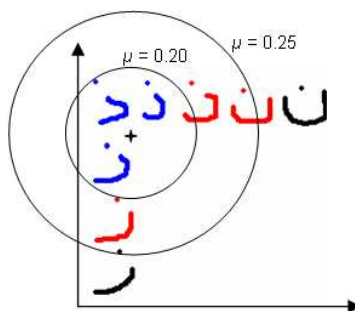


Figure 3.3: A centroid encircles more variations the larger the value of μ

When the clustering is started, all the data of each letter is put into one single cluster in accordance with the divisive hierarchical algorithm. Two randomly chosen letters are set as centroids, each encircling letters similar to them and splitting up the single cluster into two still large clusters. A new cluster center for both of the new clusters is calculated from the mean similarity of the letters

in each cluster. This procedure is iterated until the final clusters reach the size of μ . After clustering of all the letters, the database is essentially complete and ready to be used by the engine.

Chapter 4

Interpretation

Before any interpretation is possible, each written letter must be transformed to the same coordinate system as the reference letters in the database. In our engine the transformation is done using an extended affine transformation algorithm developed by Rikard Berthilsson and Kalle Åström [4] which is independent of choice of coordinates. It also aids in finding point correspondences between curves which makes it possible to differentiate between characters such "b", "d", "q" and "p". The details of this method are beyond the scope of this thesis but are thoroughly described in [4].

4.1 Proximity measures

The most frequently used techniques in handwriting recognition is Hidden Markov Models [5] which is a common technique used in the field of speech recognition. Other popular techniques are neural networks [6], fuzzy logic and key spotting technique. To be able to use any of these solutions, key information about each character needs to be extracted e.g. number of up and down strokes, number of loops, enclosed loop areas etc. The technique used in the Decuma-Alphabet engine is an implementation of an algorithm called proximity measurement [4], in which the entire shape of the character is constructed by treating the entire parameterized curve as a single object rendering feature extraction unnecessary. The definition of a proximity measure is described below.

Let G be a group of transformations:

$$G : L_{\mu}^2(\mathfrak{R}, \mathfrak{S}^n) \rightarrow L_{\mu}^2(\mathfrak{R}, \mathfrak{S}^n).$$

Then a proximity measure p can with help quotient spaces be defined as

$$p : \frac{L_{\mu}^2(\mathfrak{R}, \mathfrak{S}^n)}{G} \times \frac{L_{\mu}^2(\mathfrak{R}, \mathfrak{S}^n)}{G} \rightarrow \mathfrak{R}_+$$

if p is zero on the diagonal.

When the elements in $\frac{L_{\mu}^2(\mathfrak{R}, \mathfrak{S}^n)}{G}$ can be identified by linear subspaces B , the proximity measure can be written as

$$p(B_1, B_2) = \|\mathbf{P}_{B_1} - \mathbf{P}_{B_2}\|_{HS},$$

where $B_1, B_2 \subseteq L_\mu^2(\mathfrak{R}, \mathfrak{S}^n)$ are finite dimensional linear subspaces. The index HS refers to the Hilbert-Schmidt norm which is defined as

$$\sqrt{\sum_{j=1}^{\infty} \|T e_j\|^2},$$

where T is a linear operator in a separable Hilbert space H and $\{e_j\}_{j=1}^{\infty}$ a complete orthonormal set for H . Berthilsson and Åström propose a more convenient way to write the proximity measurement.

$$p(B_1, B_2) = \|(I - \mathbf{P}_{B_1})\mathbf{P}_{B_2}\|_{HS},$$

where I denote the identity matrix. This is possible because of the properties of $\{e_j\}_{j=1}^{\infty}$

$$p(B_1, B_2) = \|(I - \mathbf{P}_{B_1})e_j\|^2,$$

which leaves a finite number of terms.

4.2 Interpretation phases

A normal database which covers one alphabet may have some 250-300 entries each containing 32 points. In the worst case scenario a total number of 8000-10000 points have to be analyzed. Due to CPU-power-limitation on most handheld devices, a technique is used to minimize number of comparisons needed by dividing the interpretation process into several phases. The interpretation is divided into three major phases, see Figure 4.1. Some characters will also undergo a fourth phase which is discussed in section 4.2.4. In each phase the number of possible candidates is reduced which saves processing time for the next phase. The output of the interpretation is a vector of likely candidates in decreasing order of probability.

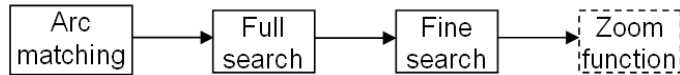


Figure 4.1: The phases of interpration

4.2.1 Arc matching

The first phase is a very simple phase which, in some cases, can easily halve the candidate list. It consists of the following control: If the inputted character consists of x arcs, only retrieve candidates which have x arcs. In the case of the Latin alphabet, consider the letter "l" written with one stroke. In this case the candidate list excludes characters such as "t" and "f" which are usually made of by two strokes.

4.2.2 Full search

The second phase is called the full search which uses the recognition technique of the engine for only one fourth of the points of each remaining character.

4.2.3 Fine search

In the final phase it becomes important to be meticulous in differentiating between the candidates. Therefore the recognition procedure is again run but this time with all 32 points of each remaining character.

4.2.4 Zoom functions

It is very common that two or more different characters in one's handwriting look a lot similar to each other. Compare the handwritten version of "a" and "u" or Arabic letter Dal (د) and Reh (ر) in Figure 4.2. The main reason for this is the hand writers' sloppiness. In fact most readers are able to ignore handwritten ambiguities and even correct spelling errors whilst reading.

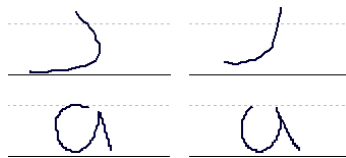


Figure 4.2: Close character pairs: د vs. ر and a vs. u

No matter how well designed the engine may be, there is always a risk that a character like "a" is misinterpreted as "u" and vice versa. The remedy is to set one or several hard coded rules which clarify the distinction between them. Regarding this particular pair, a good trick is e.g. to measure the distance between the top points of the character in relation with its maximum width. If the quotient has a lower value then a predefined ratio then it is more likely the character is an "a". Zoom functions are specifically written functions which are only applied on a set of predefined character pairs. These function are aimed to correct any misinterpretations made by the fine-search-engine. The zoom function for a specific pair is automatically invoked when the pair resides in the first and the second slots of the candidate list. Most zoom functions use constraints as a classification method. Here is another example demonstrating how a u-v-separation-algorithm can be defined.

1. Find the first inputted point and set as point a.
2. Go down until finding a minimum and set as point b.
3. Calculate the area below the arch from a to b.
4. Calculate the area of the triangle which is made up by their relative vertical and horizontal difference.
5. Divide the areas and compare with a predefined ratio.

Figure 4.3 and Figure 4.4 demonstrate how different the areas could be for "u" and "v" respectively.

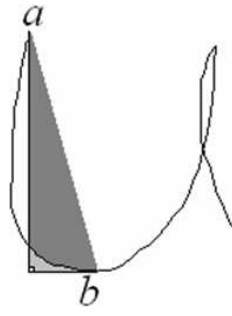


Figure 4.3: When the written letter looks very similar to u

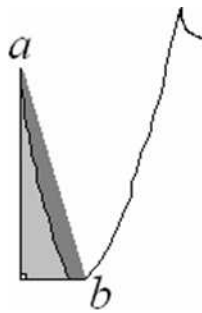


Figure 4.4: When the written letter looks very similar to v

Chapter 5

Result

The database created in this thesis work has support for all 28 standard Arabic letters in isolated form as well as a few commonly used letters which contain some diacritics. It also has support for the standard Arabic-Indic numerals. The clustering was made individually for each character i.e. each character was clustered using a unique μ -value with the goal of minimizing the size of the database as much as possible and on the same time retaining most possible variations. A couple of zoom functions were developed to secure a hit rate above 95%. The results of the hit rate test is shown in Table 5.1.

<i>Character</i>	<i>Occurences</i>	<i>Hit rate</i>
0	23	100.00%
1	23	91.30%
2	22	100.00%
3	22	100.00%
4	23	86.96%
5	23	100.00%
6	23	100.00%
7	22	100.00%
8	23	100.00%
9	23	100.00%
ء	24	83.33%
آ	24	95.83%
أ	24	100.00%
ؤ	24	87.50%
إ	24	87.50%
ئ	24	91.67%
ا	24	100.00%
ب	24	95.83%
ن	24	87.50%
ث	24	91.67%

ح	24	100.00%
ح	25	92.00%
خ	23	86.96%
د	24	100.00%
ذ	24	100.00%
ر	24	95.83%
ز	24	95.83%
س	21	100.00%
ش	18	94.44%
ص	24	100.00%
ض	23	100.00%
ط	24	91.67%
ظ	23	86.96%
ع	24	95.83%
غ	23	95.65%
ف	24	100.00%
ق	24	95.83%
ك	24	62.50%
ل	24	100.00%
م	24	100.00%
ن	48	83.33%
ه	48	97.92%
و	48	100.00%
ي	47	100.00%
٠	23	91.30%
١	24	100.00%
٢	26	100.00%
٣	22	100.00%
٤	24	100.00%
٥	24	100.00%
٦	24	100.00%
٧	23	100.00%
٨	24	100.00%
٩	24	100.00%
<i>Total number of characters</i>	<i>Total number of occurrences</i>	<i>Average hit rate</i>
56	1437	95.76%

Table 5.1: Hit rates results

Chapter 6

Discussion

This thesis work was about creating a good quality database and engine with support for the Arabic script simply by augmenting the current Latin-based database and engine. The database was created by collecting writing contribution of forty Arab persons with different origin, age and sex and further processing these contributions by sampling and clustering. Clusters were successfully created by choosing a unique cluster threshold for each character. Although more laborious, this gave a better control of the clustering and the amount of space each character occupied. Moreover, a third of the collected writings were used to test the engine with our database. To improve recognition of this test data, we experimented with one of the existing features of the alphabetic engine, which enabled interpretation of diacritics and letters separately. However this only lowered the average recognition hit rate. Instead, improvements were made by implementation of a few zoom functions which helped distinguishing problematic character pairs. The final average hit rate of the engine became 95.76% which was higher than expected. One of the main limits of this database was that it only interpreted one character at a time which is not a common way of writing Arabic. In order to add support for word recognition another technique must be used since with proximity measurement each word would have to be regarded as one object and thus be compared to all other words in the database. By using Hidden Markov Model, features of each word could be used in the interpretation. To further improve the recognition hit rate, dictionaries could be used to lower the probability of incorrect character constellations. In addition the feature of zoom functions could be extended to allow more than two characters to be compared or become completely generic e.g. by implementing support vector machines, see [1]. Many of the contributors expressed that it was unnatural to write on a Wacom tablet because during writing the pen's trace isn't visible directly on the tablet as on a piece of paper. This could have affected the quality of the collected material and thus the final database. The work accomplished during this thesis is an important step towards a more complete recognition engine of the Arabic script for handheld devices.

Bibliography

- [1] David Danowsky. *Cyrillic Handwriting Recognition using Support Vector Machines*. Lunds Tekniska Högskola, 2006
- [2] Andreas Pålsson. *Language modeling using HMMs*. Lunds Tekniska Högskola, 2006
- [3] Jim C. Bezdek. *Fuzzy Mathematics in Pattern Classification*. Applied Mathematical Center, Cornell University Ithaca, 1973.
- [4] Rikard Berthilsson & Kalle Åström. *Extension of Affine Shape*. Centre for Mathematical Sciences, Lund University, 1999
- [5] Jonas Andersson. *Hidden Markov Model based Handwriting Recognition*. Centre for Mathematical Sciences, Lund University, 2002
- [6] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996