

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

**Handwritten word recognition – Application to
Arabic cheque processing**

Yousef Al Ohali

A Thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montreal, Quebec, Canada

February 2002

© Yousef Al-Ohali, 2002



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-73361-0

Canada

ABSTRACT

Handwritten word recognition –Application to Arabic cheque processing

Yousef Al-Ohali, Ph.D.

Concordia University, 2002

This thesis presents a study to process Arabic handwritten cheques. It includes the development of a unique set of databases that constitute a solid base for research in this domain. The databases are unique in terms of their source, domain and tags validation process. First, they are originated from real-world bank cheques, which, to the best of our knowledge, has never been reached in a university setting. Second, it constitutes the only databases in the domain of Arabic handwritten cheques so far. To the best of our knowledge, there is no database that provides training and testing samples for Arabic cheques. Third, it involved a unique tagging validation process that takes advantage of the embedded redundancy in the format of the cheque to verify the tagging process. A grammar to validate Arabic legal amounts and translate them to numerical values is also included.

This work includes an efficient method to derive one-dimensional feature sequence that preserves the dynamics of the original two-dimensional images. This is very important to accommodate small variations while modeling two-dimensional signals. The thesis provides a detailed description of an improved graph representation of sub-word images, a more efficient method to extract dynamic information from two-

dimensional images and a clear positioning of the applicability of other curve-ordering criteria, e.g. vision rules.

In addition, this thesis includes a significant improvement in the discrimination power of HMM which allows the differentiation between short sub-words and longer ones that share significant initial observation sequences. It also allows the HMM to properly classify incomplete observation sequences. The improvement is achieved by introducing a new parameter to the HMM called the termination probability.

Included in this work are tests that prove the applicability and efficiency of the above contributions. At the time of this dissertation, our survey indicates that this work is the only research in the literature which handles images of handwritten sub-words extracted from Arabic cheques. The results of this study show a 94.36% sub-word recognition rate on the top 10 choices. Error analysis indicates some errors caused by the pre-processing (48%), feature extraction (28%) and classification (24%) modules.

Acknowledgments

I would like to express my thanks and appreciation to all people who helped me toward completing my degree. My sincere gratitude is due to my supervisors Prof. Ching Y. Suen and Prof. Mohamed Cheriet for their continuous support and informative feedback in a wonderful scientific environment. I am grateful also to Prof. Tonis Kasvand for the critical yet friendly discussions we had throughout this work. The rest of the exam committee deserves my special thanks also for their critical contribution to the quality of this work including Prof. Majid Ahmadi and Prof. O. Moselhi.

My thanks are also due to all friends in CENPAMRI who provided and preserved a very friendly and cooperative research environment including Karim, Nicola, Christine, Andrea, Boulos, Nedjem-Eddine, Alessandro, Jianxiong, Jun, Qizhi, Liqiang, Javad, Beverley, Frederic, Nancy, Marisa and Luiz.

The Muslim community here in Montreal deserves my special thanks also for the social support my family and I have received from them.

Last but not least, my great thanks and appreciation are due to all my family members. I am particularly thankful to my father back home for his patience and lifelong encouragement, and to my wife, Asma, for her patience, scarifies, encouragement, and for taking care of the three young lights in my home: Sura, Jana and Salma.

Yousef Al-Ohali

May 1, 2002

Montreal- Canada

Table of contents

LIST OF FIGURES.....	ix
LIST OF TABLES.....	xii
CHAPTER 1: INTRODUCTION.....	1
1.1. PREVIOUS WORK.....	2
1.1.1. Databases.....	3
1.1.2. Pre-processing.....	4
1.1.3. Segmentation.....	5
1.1.4. Feature Extraction.....	6
1.1.5. Classification.....	8
1.1.6. Combination of multi-classifiers.....	10
1.1.7. Integration of legal amounts.....	11
1.2. BRIEF DESCRIPTION OF LEGAL AND COURTESY AMOUNTS WRITTEN IN ARABIC	12
1.3. PROPOSAL	15
1.4. CONTRIBUTIONS	16
1.5. THESIS ORGANIZATION	17
CHAPTER 2: DATABASE DEVELOPMENT.....	19
2.1. DATA COLLECTION	19
2.2. PRE-PROCESSING	20
2.3. TAGGING.....	21
2.4. VALIDATION	23
2.4.1. Automatic validation procedure.....	23
2.4.2. Manual validation procedure.....	29
2.5. STRENGTH OF THE VALIDATION PROCESS.....	30
2.6. STRUCTURE AND ORGANIZATION OF THE DATABASES.....	31

CHAPTER 3: SUB-WORD RECOGNITION SYSTEM	36
3.1. DESIGN AND METHODOLOGY	37
3.2. GLOBAL MODEL.....	40
3.2.1. Feature extraction	40
3.2.2. Classification.....	42
3.3. ANALYTICAL MODEL	45
3.3.1. Pre-processing.....	46
3.3.2. Feature extraction	46
3.3.3. Estimation of pen trajectory of hand-written sub-words	49
3.3.4. Vector quantization.....	54
3.3.5. Clustering	55
3.3.6. Classification.....	56
3.3.7. Effect of the termination probability on the training procedure.....	59
3.3.8. Effect of the termination probability on the testing procedure	61
3.4. COMBINATION OF THE GLOBAL AND ANALYTICAL MODELS	62
CHAPTER 4: EXPERIMENTAL RESULTS	64
4.1. GLOBAL MODEL.....	64
4.1.1. Number of competitive neurons	64
4.1.2. Performance of the global model.....	66
4.2. ANALYTICAL MODEL	68
4.2.1. Size of the code book.....	68
4.2.2. Re-estimation of the initial probabilities	70
4.2.3. Termination probabilities	70
4.2.4. Performance of the analytical model.....	71
4.2.5. Error analysis.....	72
4.3. COMBINED CLASSIFIERS	76
CHAPTER 5: LEGAL AMOUNT PROCESSING	80
5.1. PRE-PROCESSING	80
5.2. LEGAL AMOUNT INTERPRETATION	82
5.3. DISCUSSION AND CONCLUSION.....	85

CHAPTER 6: CONCLUSION	89
6.1. SUMMARY OF CONTRIBUTIONS.....	90
6.2. FUTURE PERSPECTIVES	91
REFERENCES	93
APPENDIX A: ARTIFICIAL NEURAL NETWORKS	102
A.1. INTRODUCTION	102
A.2. APPLICATIONS OF ANN	103
A.3. ANN STRUCTURE.....	104
A.4. CATEGORIES OF ANN	105
<i>A.4.1. Static vs. Dynamic ANN</i>	<i>106</i>
<i>A.4.2. Supervised vs. unsupervised training.....</i>	<i>106</i>
A.5. MAJOR ANN TYPES.....	106
<i>A.5.1. Multi-layer perceptron (MLP).....</i>	<i>106</i>
<i>A.5.2. Competitive ANN.....</i>	<i>107</i>
<i>A.5.3. Radial Basis Function (RBF) Networks.....</i>	<i>108</i>
<i>A.5.3. Recurrent (Hopfield) network.....</i>	<i>109</i>
A.6. LIMITATIONS OF ANN	109
APPENDIX B: GRAPH THEORY	111
B.1. BASIC DEFINITIONS	111
B.2. THEOREMS.....	113
APPENDIX C: HMM.....	119
C.1. INTRODUCTION	120
C.2. TRAINING AN HMM	122
C.3. TYPES OF HMM.....	127
C.4. EXTENSIONS TO HMM- LEVEL BUILDING ALGORITHM	129

APPENDIX D: SAMPLES FROM THE DATABASES	131
D.1. COMPLETE CHEQUES	131
D.2. LEGAL AMOUNTS	134
D.3. COURTESY AMOUNTS	137
D.4. SUB-WORDS	140
D.5. INDIAN DIGITS	148

List of Figures

FIGURE 1-1: SINGULAR, DOUBLE AND PLURAL FORMS FOR THE WORD "THOUSAND"	13
FIGURE 1-2: FOUR GRAMMATICAL FORMS OF THE WORD (TWO THOUSAND).....	13
FIGURE 1-3: FEMININE AND MASCULINE FORMS OF THE WORD "THREE"	14
FIGURE 1-4: TWO COMMON FORMS FOR THE WORD "HUNDRED"	14
FIGURE 1-5: SECONDARY COMPONENTS OF THE LAST LETTER MAY BE IGNORED, A COMMON MISTAKE	14
FIGURE 1-6: ARABIC DIGITS AND THEIR CORRESPONDING INDIAN DIGITS.	15
FIGURE 1-7: THESIS STRUCTURE	18
FIGURE 2-1: A SAMPLE OF THE ARABIC CHEQUE DATABASE	21
FIGURE 2-2: SEGMENTED LEGAL AMOUNT (FROM FIGURE 2-1).....	21
FIGURE 2-3: SEGMENTED COURTESY AMOUNT (FROM FIGURE 2-1)	21
FIGURE 2-4: THE WORD TEN (RIGHT) APPEARS AS A SUB-WORD IN THE WORD TWENTY	24
FIGURE 2-5: ONE ARABIC AMOUNT WITH TWO POSSIBLE INTERPRETATIONS.....	26
FIGURE 2-6: ALGORITHM USED TO TRANSLATE ARABIC LEGAL AMOUNT INTO A NUMERICAL VALUE.....	29
FIGURE 2-7: SAMPLE OF THE REJECTED CHEQUES DUE TO SPELLING MISTAKE	31
FIGURE 2-8: THE VALIDATION PROCESS	32
FIGURE 2-9: STRUCTURE OF THE COURTESY AMOUNT AND DIGITS DATABASES.....	35
FIGURE 3-1: OVERVIEW OF THE SUB-WORD RECOGNITION SYSTEM	39
FIGURE 3-2: SUB-WORD IMAGE BEFORE AND AFTER SIZE NORMALIZATION	40
FIGURE 3-3: FEATURES EXTRACTED FROM FIGURE 3-2 ABOVE, IN THE GLOBAL MODEL	41
FIGURE 3-4: NN CLASSIFIER.....	42
FIGURE 3-5: A SAMPLE SUB-WORD IMAGE AND ITS SKELETON	46
FIGURE 3-6: THE CHAIN CODE SEQUENCE EXTRACTED FROM FIGURE 3-5 ABOVE.....	48
FIGURE 3-7: THE FREEMAN CHAIN CODE.....	48
FIGURE 3-8: LINE APPROXIMATION OF THE CHAIN CODE SEQUENCE SHOWN IN FIGURE 3-6	49
FIGURE 3-9: GRAPHICAL REPRESENTATION OF FIGURE 3-8	49

FIGURE 3-10: THREE HANDWRITTEN ARABIC WORDS (ONLY, SEVEN AND FIFTY) WITH THEIR SKELETONS AND THE ESTIMATED SEQUENCES OF SEGMENTS	51
FIGURE 3-11: PARTIAL APPLICATION OF ALGORITHM 2 TO A GENERAL GRAPH	55
FIGURE 3-12: TWO DIFFERENT SUB-WORDS THAT SHARE THEIR INITIAL PORTIONS (STARTING FROM RIGHT TO LEFT)	56
FIGURE 3-13: PATTERN (B) FROM FIGURE 3-12 ABOVE, PASSED THROUGH BOTH MODELS (A) AND (B)	57
FIGURE 3-14: TWO DIFFERENT SUB-WORDS THAT SHARE THEIR INITIAL PORTION (STARTING FROM RIGHT TO LEFT)	58
FIGURE 4-1: PERFORMANCE OF THE GLOBAL MODEL VERSUS THE NUMBER OF CLUSTERS ..	65
FIGURE 4-2: SIGNIFICANCE OF THE GLOBAL MODEL VERSUS THE NUMBER OF CLUSTERS ...	66
FIGURE 4-3: A SAMPLE THAT HAS BEEN MIS-CATEGORIZED BY THE NN CLASSIFIER (CLASS NUMBER 5, CODE 1-9)	67
FIGURE 4-4: SAMPLES OF ERRORS CAUSED BY DIFFICULT HANDWRITING. THE CORRECT CLASS NUMBER (CODE) IS SHOWN BELOW EACH IMAGE	73
FIGURE 4-5: SAMPLES OF ERRORS CAUSED BY THE EXTRACTION AND BINARIZATION MODULE. THE CORRECT CLASS NUMBER (CODE) IS SHOWN BELOW EACH IMAGE	74
FIGURE 4-6: AN ERROR CAUSED BY THE SKELETONIZATION MODULE	74
FIGURE 4-7: AN ERROR CAUSED BY THE FEATURE EXTRACTION MODULE	75
FIGURE 4-8: SAMPLES OF ERRORS CAUSED BY THE APPROXIMATION PROCEDURE	76
FIGURE 4-9: AN ERROR CAUSED BY THE CLASSIFICATION MODULE. FIRST CHOICE IS CLASS NUMBER 5 (CODE 1-09)	76
FIGURE 5-1: SEGMENTATION AND BINARIZATION OF THE LEGAL AMOUNT	81
FIGURE 5-2: GROUPING OF CANDIDATE SUB-WORDS TO PRODUCE THE TOP FIVE POSSIBLE VALUES IN TABLE 5-4	88
FIGURE A-1: A SIMPLE EXAMPLE OF ANN	105
FIGURE A-2: MLP WITH SIGMOID TRANSFER FUNCTION IN THE HIDDEN LAYER	107
FIGURE A-3: COMPETITIVE NETWORK	108
FIGURE A-4: THE RADIAL BASIS FUNCTION NETWORK	109
FIGURE A-5: HOPFIELD NETWORK	110

FIGURE B-1: A GRAPH WITH FOUR VERTICES (A,B,C,D) AND FOUR EDGES ((A,B),(A,C),(A,D),(B,C))	111
FIGURE B-2: A STAR	113
FIGURE B-3: (A) FIGURE B-3: (A) THE TREE WITH SINGLE VERTEX HAS 0 EDGES, (B) THE TREE WITH 2-VERTICES HAS 1 EDGE	115
FIGURE B-4: THE MINIMUM COST TO VISIT THE TWO VERTICES A AND B IS X	117
FIGURE C-1: EXAMPLE OF A STOCHASTIC NETWORK WITH 5 STATES. TRANSITION PROBABILITIES ARE SHOWN NEAR THEIR ORIGIN.....	119
FIGURE C-2: EXAMPLE OF A STATISTICAL NETWORK WITH 5 STATES AND THEIR OBSERVATION PROBABILITIES.....	120
FIGURE C-3: GRAPHICAL REPRESENTATION OF THE COMPUTATION OF THE FORWARD VARIABLE AT TIME STEP $T+1$ AT STATE S_i	124
FIGURE C-4: GRAPHICAL REPRESENTATION OF THE COMPUTATION OF THE BACKWARD VARIABLE AT TIME STEP T AT STATE S_i	125

List of Tables

TABLE 2-1: DISTRIBUTION OF THE DIGITS DATA SET	32
TABLE 2-2: DISTRIBUTION OF THE SUB-WORDS DATA SET.....	33
TABLE 2-3: DISTRIBUTION OF DATABASES BETWEEN TRAINING AND TESTING SETS	35
TABLE 4-1: PERFORMANCE OF THE GLOBAL MODEL.....	67
TABLE 4-2: PERFORMANCE OF THE HMM CLASSIFIER VERSUS THE CODEBOOK SIZE. THIS EXPERIMENT WAS PERFORMED ON THE VALIDATION SET, WITH REESTIMATION OF π AND USING ϕ	69
TABLE 4-3: EFFECT OF THE RE-ESTIMATION OF π ON THE PERFORMANCE OF THE HMM CLASSIFIER. THIS EXPERIMENT WAS PERFORMED ON THE TRAINING SET USING ϕ	70
TABLE 4-4: EFFECT OF THE INTRODUCTION OF ϕ ON THE PERFORMANCE OF THE HMM CLASSIFIER. THIS EXPERIMENT WAS PERFORMED ON THE TRAINING SET	71
TABLE 4-5: PERFORMANCE OF THE ANALYTICAL MODEL	71
TABLE 4-6: SUB-WORD IMAGES COVERED BY THE RECOGNITION SYSTEM.....	72
TABLE 4-7: PERFORMANCE AND CONFIDENCE OF EACH CLASS ON THE TRAINING DATA.....	78
TABLE 4-8: PERFORMANCE OF THE COMBINED RECOGNITION SYSTEM.....	79
TABLE 4-9: RECOGNITION RESULTS COMPARED WITH OTHER SYSTEMS IN THE LITERATURE.....	79
TABLE 5-1: THE SEQUENCE OF SUB-WORDS SEGMENTED FROM FIGURE 5-1	83
TABLE 5-2: RESULTS FROM THE SUB-WORD CLASSIFIER – TOP 10 CHOICES FOR EACH SUB-WORD IN TABLE 5-1 (THE CORRECT CHOICES ARE SHADED)	84
TABLE 5-3: WORD RECONSTRUCTION (SOLID LINES SEPARATE WORDS, DASHED LINES SEPARATE SUB-WORDS WITHIN A WORD)	86
TABLE 5-4: TOP 10 NUMERICAL RESULTS WITH THEIR PROBABILITIES (THE CORRECT CHOICE IS SHADED).....	87

CHAPTER 1

Introduction

Pattern Recognition is a vast field of research that intends to train the computer to simulate some of the human abilities (e.g. hearing, vision) using artificial intelligence. Although these abilities seem very trivial to humans, yet they have claimed many years of research and development in the field of artificial intelligence. Nowadays it is unlikely to build a unique system that simulates all human abilities. Therefore, different human abilities have generated different sub-areas of research.

Character Recognition is one of the challenging areas in pattern recognition that have been decomposed into smaller sub-areas seeking for a human equivalent performance. It involves On-line Character Recognition, where the user needs to use an electronic pen to write on a paper-like electronic plate. It also involves Off-line Character Recognition, where the user has more freedom in selecting the writing material. Due to the high complexity of some specific types of documents, special research works have been initiated in collaboration with other research fields of study. Form processing has helped the area of cheque processing among other applications of handwriting recognition.

From the administrative point of view, cheque processing involves all the tasks a bank officer may perform to process an incoming cheque for a client. This includes: accessing account numbers, verifying names and signatures on the cheque, verifying the date of the cheque, matching the legal amount with the courtesy amount and verifying the credit of the cheque writer. However, from the technical point of view, cheque processing

could involve capturing the cheque image, separating the foreground of the cheque from its background, extracting fields of interest, and recognizing each of them.

The goal of this study is to establish the research in recognition of legal amounts extracted from Arabic cheques and to investigate the applicability of some of the classification methods to this problem.

This work employs theories and methodologies from various fields in order to achieve its objectives. First, Natural language processing is used to validate, translate and interpret legal amounts. Second, theories and corollaries from the field of Graph Theory are integrated to derive an efficient transformation of the feature map into 1-D domain. Finally, Statistical classification methods are employed to achieve an improved discrimination between various sub-word patterns.

The motivation of this work is not less than the motivation of the entire research in artificial intelligence, which aims to carry out tedious routine processes, freeing time and space for humans to perform tasks that require higher levels of intelligence. A major advantage of this study is that it can be easily adjusted to serve more than 20 different countries (all of them use Arabic as their first language). In addition, legal amounts are widely found in documents other than bank cheques (e.g. business sell/purchase forms). Therefore, this study will be applicable to a wide range of applications. Moreover, similar languages (e.g. Urdu, Farisi) which use the same alphabet can benefit from this study.

1.1. Previous work

This section gives a general review of research in areas related to this work including cheque processing, handwritten word recognition and Arabic character recognition. Detailed review is not the goal of this section.

1.1.1. Databases

Due to strict banking rules to protect their customers, it is extremely difficult to gain access to real cheques. This led researchers to perform their research within financial institutions [GL93], or to build artificial databases [GS98]. In [GS98], about 2600 English cheques, written by 800 writers with pre-set legal amounts, have been collected. Another set of 1900 French cheques from 600 different writers has been collected too. The legal amounts were set for the writers to reflect balanced word distribution.

In a different type of applications, a database of addresses has been developed in [Hu94]. The database contains 5000 city names, 10,000 ZIP codes and 50,000 alphanumeric characters. The database was collected from real mail pieces. [MC98] has collected an artificial set of handwritten Arabic city names. The database contains about 5900 city names, comprising a lexicon of 232 different cities.

It is noted that the accuracy of the labeling process highly depends on the clarity/ease of the tool used for labelling, and on the person who uses the tool. None of the above databases mention the use of redundant information to verify the accuracy of the labelling process.

In this thesis we develop new databases for research in Arabic legal amount and courtesy amount recognition. A major characteristic of this work is that the underlying data is extracted from real bank cheques, exposing real-life problems to research focus points. Another significant characteristic of this work is that the labeling procedure is validated by logical use of redundant information.

1.1.2. Pre-processing

Few research works were targeting a complete cheque processing system. In such systems, pre-processing includes the following tasks:

1. **Foreground/background separation process.** This task could be difficult in case of complex background patterns. For this task, average filtering has been applied and shown acceptable results [SL96].
2. **Segmentation and extraction of useful fields from the cheque form.** Locating fields of interest is normally based on preliminary assumptions about their approximate locations. Guidelines could then be detected and used to extract such fields [SL96] [LLG97]. Minor skews are taken into consideration during the guideline detection process.
3. **Removal of pre-printed lines that may overlap with useful handwritten data.** Such overlaps could be processed locally through heuristic rules [LLG97], or through topological operations [SL96]. This task becomes more challenging in cases where the background is darker than the guidelines.
4. **Slant normalization.** This process is generally applied to the overall legal amount [SL96], [LLG97], [KB00a], [GA99]. Methods used for slant computation are based on the histogram of contour pixels [KB00a], Hough transform [Am00], vertical projections, and the average angle of near vertical strokes or chain codes [KF00].
5. **Baseline detection.** This process is frequently applied due to its noticeable importance for further pre-processing steps and feature extraction modules [EG99] [CL98].

6. Segmentation of legal amounts into words. This is mostly done based on horizontal spacing [LLG97]. To avoid the effect of artefacts, computation of horizontal spacing could be constrained within the middle base line area [YS98].
7. Size normalization. This is an important step for some types of features. While the object's size could be set to pre-defined values for character sets of the same width [AV00], relational size normalisation is more efficient for handwritten words. In such a case, height and width should be dealt with independently. Width is normalized based on approximation of number of letters [KB00a], while height could be normalized based on the middle base line area [EG99] [KB00a]. Stroke width normalisation is useful for hand-printed characters [GW98].

In this thesis, we apply most of the above-mentioned steps, using suitable adjustments of previously designed algorithms.

1.1.3. Segmentation

Segmentation takes place again here, and is meant to partition a word image into smaller segments, possibly letters. This is essentially noticeable in handwritten words and in cursive languages (e.g. Arabic). Segmentation allows word composition from basic components, which enables the same lexicon domain as the underlying language. However, segmentation errors are hardly recoverable in subsequent stages. Researchers have taken different stands toward this process. Following are the main approaches considered in this regard:

1. Explicit segmentation uses some explicit measures to produce segmentation hypotheses. Such measures could be structural or statistical measures extracted from the input image, and are usually selected not to overlook any segmentation

points, resulting in what is called over-segmentation. The classifier builds its decision based on features extracted from each segment, with the possibility of joining feature vectors of successive segments. Ordering of successive segments is important in joining feature vectors [LLG97] [YS98] [EG99] [ZM99] [KA98] [CB99] [BS99] [AK01].

2. Implicit segmentation delays the segmentation process until the classification phase. Recognition confidence is used by the classifier to detect proper segmentation points [KB00a] [KK97]. In this approach, the classifier works under the assumption that features have been extracted in a sequential manner that preserves the sequence of symbols. Overlaps between features of successive symbols could mislead the classifier.
3. Segmentation free approach avoids word segmentation at all. Classification is normally done based on global features extracted from the input word image as a whole [Am00] [MG96]. This is also known as a wholistic approach. It is applicable where the lexicon contains a limited (small) number of entries. Avoiding explicit segmentation hypothesis that could form a basis for misclassification is the main advantage of this approach.

In this thesis, we apply the third method in dealing with Arabic sub-words.

1.1.4. Feature Extraction

Both statistical and structural features were used to describe input word/character images. Structural features are more commonly used to describe handwritten data due to their low sensitivity to minor shape variations. Global features (e.g. ascender, descenders, loops ... etc.) were used in [SL96] [LLG97] [CL98] [EG99] [PS97]. In [GS98] number and

position of these features were used to compose a weighted feature vector. Such features are sensitive to slant and to handwriting artefacts that are frequently introduced at the beginning and/or at the end of the writing. Geometrical features (e.g. horizontal and vertical bars) were used in [HS97] [SL96] [PS97] [RP94] [LL97] [Ab98]. These features are also sensitive to slant. Contour features were used in [GS98] [EG99]. Contour based features are more stable than skeleton based ones, but are severely affected by touching and broken parts. Topological features were used in [HS97] [RP94] [LL97] [AY97]. These features are sensitive to artifacts and small noise that could affect the input image. Junctions that involve more than three segments could mistakenly be regarded as two or more junction segments. Graphemes were used as features in [GA99] [KA98]. Characterization of graphemes is normally done by feeding a simpler set of features into an NN or a KNN classifier. Other structural features include projection based features [HS97] [CK98], edge maps [CK98], linguistic expressions [LM00] and pen trajectory elements [DF98].

Statistical features, on the other hand, describe the input image in terms of relational statistics that are computed from the input image. They are generally easier and faster to extract, but are more sensitive to small variations and require more effort on the pre-processing stage. Statistical features have been used as stand alone descriptions for handwritten images, or in complement to other structural features. They have been used also with various types of classifiers including NN, HMM, KNN. Statistical features could be based on pixel counts [KB00a] [LLG97], run-lengths [MG96] [AV00], eigenvalues and eigenvectors [GW98] or Fourier descriptors [Ma94] [AN97]. Combinations of structural and statistical features are found in [EG99] [HP98].

In this thesis, we use both statistical and structural features to describe each input image. The statistical features are defined to provide global information about the input word, while the structural features provide analytical information. A major contribution of this thesis is the design of an efficient algorithm to retrieve an approximation of the pen trajectory.

1.1.5. Classification

Classification module in a pattern recognition system maps descriptions of input images into object identities. Researchers have used various types of classifiers while seeking an intelligent mapping process. Nearest neighbor classifier employs a direct comparison between input feature vectors and pre-registered models. Different distance measurements have been used to compare feature vectors. Special filtering process need to be applied during the training phase to exclude any misrepresenting patterns. This could be a tedious task in cases of huge training data. A KNN classifier overcomes this problem by defining a set of clusters. Each cluster is represented by the mean of all its vectors, which minimizes the effect of extreme or misrepresenting samples [ZM99] [SL96] [GS98] [LM00].

Neural Networks (NN) use small computational engines, called neurons, to arrive at a complex decision. Different types of NN differ in their structure (number and connectivity of neurons), basic functionality of neurons and the decision making function at each layer. NN is widely used because of its easy learning process and proven generalization ability on some types of problems. However, it is used as a black box that makes inference of the decision making process very difficult. In addition, the output values of NN do not reflect confidence or probability. Among the most common types of

NN are the multi-layer perceptron (MLP) [CL98] [GW98] [Alt95] and Radial Basis Function (RBF) [CK98].

Hidden Markov Model (HMM) is a statistical classifier that employs both values and order of feature vector elements to deduce a probabilistic recognition measure. Features are presented to each model in a sequential manner, and the recognition is based on model probabilities. Evaluation of the probability of a given model could be based on one of the following criteria:

1. The probability of the best path travelled by the feature vector,
2. The probability of being in the last state of the model (by all possible paths) at the end of the feature vector, or
3. The probability of being at any state in the model (by all possible paths) at the end of the observation vector.

After its successful application to speech recognition, HMM has been widely applied in the last few years to handwritten word recognition [LLG97] [KB00a] [GA99] [EG99] [PL98] [MS98] [CK94] [MG96] [KA98] [AV00] [GM93] [GS98] [MC98] [BS99] [AY97]. The most common HMM structure used in the literature is left-to-right, which enforces forward transitions throughout the states of each model. Discrete HMM expects discrete observation vectors. In cases where the feature vectors are not discrete, a quantization process has to be performed, which usually introduces some amount of error. A possible alternative is to use semi-continuous HMM, which works on some assumptions regarding the distribution of the input feature vectors. While such assumptions are normally not completely accurate, the amount of error introduced is less

than those introduced by the quantization process. However, semi-continuous HMM requires more resources in terms of memory and processing time.

Tree classifier is popular for its structural stage-wise nature. Its weakness arises in cases of early erroneous decisions [Am00a]. Other types of classifiers that are found in the literature include matching algorithm [RP94] [DF98], machine learning algorithm [Am00b] and hash table [HS97].

In this thesis, we design an NN classifier based on global features and an HMM classifier based on analytical features. A new criterion to evaluate the probability of a given HMM model represents one of the contributions of this thesis.

1.1.6. Combination of multi-classifiers

Multi-classifiers are used by many researchers in the field of pattern recognition to improve system performance and/or reliability. In general, combination methods could be divided into two categories: sequential and parallel. Sequential methods apply one classifier at a time, feeding the output of each classifier to the next one. This can be used for lexicon reduction in stage-wise recognition approaches or to achieve higher accuracy rates. In parallel methods, all classifiers are employed in parallel and the overall decision is based on the classification results of each classifier.

The authors in [SM98] proposed three combination functions for multiple classifiers. The combination methods intend to improve word recognition results out of character classifiers. The inputs of each function are sorted lists of character classes and confidence values produced by each classifier. The parameters of each function are optimized using gradient descent method. In a vocabulary of 100 words, the optimized

sigmoid combination function improved the recognition results to 84.47%, about 10% more than the better of the two classifiers used.

In [RK98], the authors proposed an approach to combine the advantages of discrete and continuous Markov models. In their approach, the probability density functions of the continuous HMMs are estimated using a neural network.

Authors in [KA98] used an NN-HMM hybrid approach to recognize cursive words. Input words are first segmented into graphemes. Features are then extracted from each grapheme and fed to a neural network to estimate the observation probabilities. Transition probabilities are estimated by the HMM. Training becomes more complicated. Given a trained HMM, they applied Viterbi backtracking algorithm to provide the target values for the NN. These values are then used to perform supervised training to the NN. Then the HMM is re-trained using the observation probabilities from the NN. This process iterates for a number of times to ensure the stability of the system.

In this thesis, we apply a sequential combination method of the two classifiers used in the classification phase. The NN classifier is first used for lexicon reduction purposes. The reduced lexicon is then fed, along with the unknown pattern, to the HMM classifier for identification purposes.

1.1.7. Integration of legal amounts

In addition to word recognition, legal amount processing involves the task of finding the most probable sequence of words that leads to a proper legal amount. Usually, syntax and semantic rules of any language disallow certain combination of words. In [Gu96], an English parser is defined and used to pick up sequences of words that lead to syntactically and semantically correct legal amounts. Another parser is used for French

legal amount. In [KB98], a syntax-directed translation graph has been designed to parse and translate German legal amounts into digit strings. The resulting digit strings are then compared to the recognized courtesy amounts, and mismatches are localised for further processing.

In this thesis, we have designed and implemented a parser for Arabic legal amounts. The parser is used to validate a given sequence of words and to derive numerical values from correct sequences.

1.2. Brief description of legal and courtesy amounts written in Arabic

Unlike Latin, Arabic is written from right to left in cursive script. Out of the 28 basic Arabic letters, 22 are cursive letters while 6 are non-cursive. Within one word, a cursive letter should be connected to the succeeding letter, while a non-cursive letter can not be connected to any succeeding letter. Thus, an Arabic word may be decomposed into more than one sub-word, each represents one or more connected letters with their corresponding secondary components [AM95][Am98].

In addition, Arabic defines two types of secondary components: dot and Hamzah (a zigzag-like shape). The number and position of secondary components play a role in the identification of the letters of the alphabet. Moreover, Arabic allows the presence of diacritics that control the pronunciation of words and possibly their meanings. However, such diacritics are only used in highly formal documents or in cases of contextual ambiguity [AM95][Am98].

The shape of an Arabic letter may change significantly depending on its position within a sub-word, identity of neighbouring letters, the writing font, and the way the

writer connects successive letters [AM95][Am98]. Arabic handwritten letters differ in height and width.

The vocabulary of Arabic legal amounts is larger than those found in Latin languages. This is due to three major factors. First, Arabic has three different forms: singular, double, and plural (Figure 1-1). This could affect both the number and the

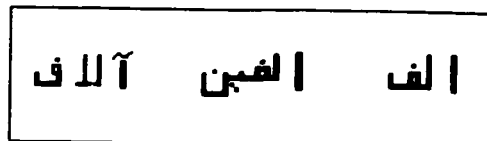


Figure 1-1: Singular, double and plural forms for the word "thousand"

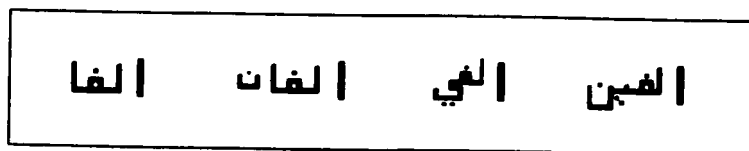


Figure 1-2: Four grammatical forms of the word (two thousand)

counted item (currency words in the context of legal amounts). Second, double and plural nouns have up to four different forms according to their grammatical positions (Figure 1-2). Third, most numbers define two forms, one for feminine and one for masculine countable things (Figure 1-3). Different forms of the same word share most letters and possibly sub-words.

In addition, a few common spelling mistakes and/or colloquial occur in writing some Arabic numbers (Figures 1-4,1-5). These factors affect the identity of letters and the number of sub-words composing a word.

We found more different words than sub-words in the Arabic legal amount lexicon. That was one of the reasons to consider sub-word as the basic unit of Arabic legal amounts.

In principle, Arabic allows legal amounts to be written in any order, i.e. starting from the most significant digit, from the least significant digit or even from the middle. However, eloquence measurements and personal habits excluded most permutations.



Figure 1-3: Feminine and masculine forms of the word "three"

Arabic imposes certain constraints on numerical values. For instance, the word hundred does not appear with numbers more than 9. Values larger than 999 should be written in the forms of the word thousand instead of cases like twelve hundred in English. In addition, since Arabic defines three different categories for single, double and plurals, there are normally different constraints for the numbers one and two of each numerical category (1, 2, 100, 200... etc).

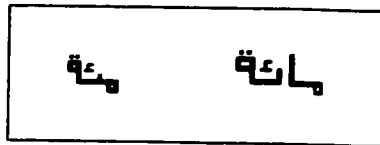


Figure 1-4: Two common forms for the word "hundred"

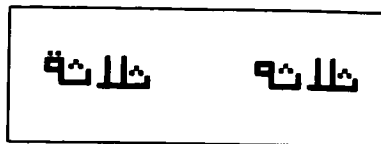


Figure 1-5: Secondary components of the last letter may be ignored, a common mistake.

With respect to courtesy amounts, there are no language constraints. Courtesy amounts are written in either Arabic or Indian digits. Indian digits are more widely used in eastern Arabian countries. Figure 1-6 shows the ten Arabic digits with the corresponding Indian ones. Note that the zero is written as dot. The decimal point is written as a large comma. There is no standard symbol to represent delimiters of courtesy amounts (dollar signs), which are optional Arabic cheques. Whether Arabic or Indian digits are used, courtesy amounts are written in the same order, putting the least significant digit in the rightmost position.

Arabic digits	Corresponding Indian digits
0	.
1	١
2	٢
3	٣
4	٤
5	٥
6	٦
7	٧
8	٨
9	٩

Figure 1-6: Arabic digits and their corresponding Indian digits.

1.3. Proposal

We aim to establish the field of Arabic legal amount processing. To this aim, we propose to investigate the applicability of global and analytical classification methods to recognize Arabic sub-words. In addition, combination of the two methods will be evaluated. Moreover, all necessary grammar and parsing algorithms will be developed.

As it is a basic infrastructure that is necessary to develop and evaluate different methodologies, we also propose to develop a database for Arabic legal amounts that could be used to train and test recognition systems. Such a database should be realistic and should be labelled in an accurate manner.

1.4. Contributions

Contributions of this thesis fall into three categories: databases development, recognition system and natural language processing to translate and interpret legal amount system.

1. This work establishes the research in Arabic legal and courtesy amount recognition by providing databases for training and testing purposes. Major characteristics of these databases are summarized below:
 - a. They were extracted from real bank cheques.
 - b. They went through a solid validation procedure to ensure the accuracy of the tagging process. Redundant information (which are widely used in various forms) have been employed to verify the correctness of the tags of each of the legal and courtesy amounts.
 - c. To the best of our knowledge, they include the first and only database for Arabic legal amounts and the first and only database for Indian digits.
2. This thesis makes initiative by investigating the difficulties and challenges of Arabic legal amount recognition. Major contributions of the recognition development part in this work include:
 - a. An efficient method to derive one-dimensional feature sequences from two-dimensional sub-word images. This involves an improved graph representation of sub-word images, a more efficient transformation to traverse all edges in the

- graph and a clear identification of traversal alternatives where other curve-ordering criteria (e.g. vision rules) can be applied.
- b. An improved discrimination power of HMM has been achieved by introducing a new parameter to the model, the termination state probability. This allows for the discrimination between short sub-words and longer ones that share significant initial observation sequences. It also allows proper classification of incomplete observation sequences. The introduction of the termination state probability does not require additional inputs to the training system, but uses some of the available (but unused data) to derive useful information.
 3. This work includes the design and implementation of a parser to validate and translate legal amounts into numerical values. The parser could be re-used for future research/applications in the field of Arabic legal amount recognition with minor modifications.

1.5. Thesis organization

After this introduction, we provide detailed description of the database development process, including data collection, labelling, and validation. Chapter three gives details of the sub-word recognition systems and the decision making process. Chapter four provides experimental results on sub-word recognition levels. Chapter five shows by example how the sub-word recognition system can be employed to arrive at a legal amount processing system. Finally, chapter six presents the conclusion of this thesis. Figure 1-7 shows a graphical presentation of the thesis structure.

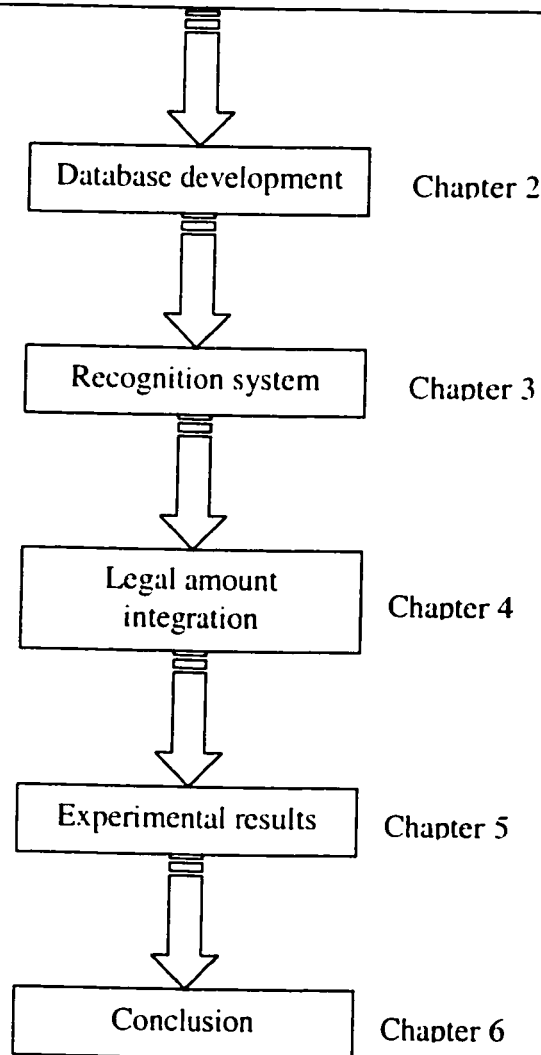
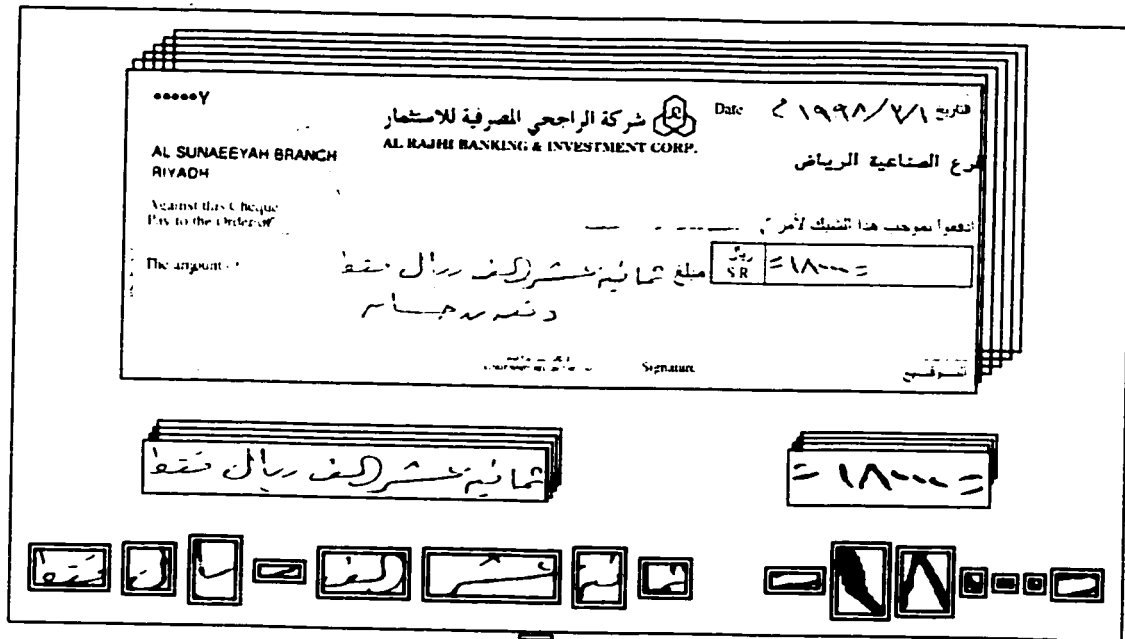


Figure 1-7: Thesis organization

Chapter 2

Database Development

Databases are essential infrastructure for the development of Arabic legal-amount recognition systems. They could be used to train, validate and test legal amount recognition systems. In addition, common databases could provide a basis for quantitative comparisons of different methodologies. This chapter describes the process of constructing databases of Arabic legal amounts and Indian courtesy amounts. These databases outperform other published databases for Latin based languages in terms of reality and validation procedures with comparable sizes.

Using real data is very important to eliminate the bias involved in laboratory environments [Hu94]. However, such data has its own drawbacks. This is observed from the uneven distribution of sub-word classes in our sub-word database that may cause some training problems. This is particularly true for those classes that are rarely used. Training and testing of such classes become more difficult and may impose restrictions on the types of classifiers.

Validation intends to verify the correctness of the labeling procedure, and is very important in building a solid ground truth. The sizes of the databases make them good candidates to be standard databases for a number of Arabic hand-written cheque processing research topics.

2.1. Data collection

The first step toward the development of a database is to find suitable sources of data. Finding a source for real data becomes a problem when dealing with applications that

carry sensitive or private information like bank cheques. Through collaboration with Al Rajhi Banking and Investment Corporation (one of the largest banking corporations in Saudi Arabia), we were able to collect about 7000 real world grey-level cheque images. The gathering process involved scanning the real cheques at the bank's centre, and removing all personal (private) information including names, account numbers, and signatures. The cheques were scanned in grey level at 300 dpi, and were used as the core of other databases throughout this project.

Cheques differ in the amount of external noise added by bank officers. Some cheques contained a stamp covering part of the legal amount, making the automated extraction and cleaning processes more difficult or impossible. In this thesis, we decided to collect two sets of data: random data and filtered data. The first set contains cheques that are randomly taken from whatever was available to us at the bank center. The second set contains only those cheques that have no stamps covering any part of the legal amount. For database development, we decided to start with the filtered data set, which is the focus of this thesis, leaving processing of the random data set for future work.

2.2. Pre-processing

The next step is to extract fields of interest from cheque images. We concentrated on the legal and courtesy amounts, leaving the date field for future work. That was achieved by localising the target fields on all kinds of cheque forms. In Saudi Arabia, there are only two types of cheques, which share the same format (structure) but have different sizes, and inter-field distances.

The next pre-processing step is to binarize and reduce existing noise in the segmented fields. Such noise could result from the digitisation process, from the

extraction process, by bank officers or by the client. Examples of such noise include lines, borders, and pre-printed texts that may appear along with the extracted fields. This step has been achieved by adapting the tools available at CENPARMI (which were designed for Canadian cheques) [CY99a] [CY99b]. Figures 2-1, 2-2 and 2-3 show an Arabic cheque and its corresponding segmented legal and courtesy amounts.

AL SUNAEEYAH BRANCH
RIYADH

AL RAJHI BANKING & INVESTMENT CORP.

Date ١٩٩٨/٧/١

فرع الصناعية الرياض

Against this Cheque
Pay to the Order of

ادفعوا بموجب هذا الشيك لأمر

The amount of

مبلغ ثمانية عشر ألف ريال فقط
دفعه حساب

ريال S.R. = ١٨٠٠٠ =

DO NOT WRITE BELOW THIS LINE

Signature

التوقيع

Figure 2-1: A sample of the Arabic cheque database.

ثمانية عشر ألف ريال فقط

Figure 2-2: Segmented legal amount (from figure 2-1).

= ١٨٠٠٠ =

Figure 2-3: Segmented courtesy amount (from figure 2-1).

2.3. Tagging

Tagging intends to label each object in the cleaned legal and courtesy amounts. A tagging tool has been prepared to tag Arabic sub-words and numerals. The tagging operator is

required to click at any point on each connected component of the target object and then select a tag from a pre-defined vocabulary. The tool stores the Cartesian coordinates of each point, adds a delimiter and stores the tag. It allows tagging of touching objects and permits reverse action (undo).

While defining our vocabulary, we have accounted for most differences, even small ones. For instance, two different tags were used to label objects that differ only in their secondary components (Figure 1-5). This gives more choices for future analysis since there is no cost to merge similar sub-classes (if such discrimination is not useful for a particular method or application). We limited the vocabulary to amounts that are less than one million. Larger amounts will mostly require manual manipulation by bank officer(s). We also included in the lexicon other words that are frequently used in the collected data (e.g. currency words).

For each cheque, the tagging tool produces four sets of tagged files:

1. Courtesy amount file: contains the image of the courtesy amount extracted from the input cheque, along with a sequence of Cartesian coordinates and tags of every object (component) in the courtesy amount. Objects may include Indian digits, delimiters, commas, decimal points or noise. The coordinates provide an unambiguous pointer to the object intended by each tag.
2. Indian digit files: each of these files contains the image of a single connected component (digit) extracted from the courtesy amount of the cheque and its tag(s). Each file contains also a reference to the originating cheque, followed by the tag of the digit.

3. **Legal amount file:** contains the image of the legal amount extracted from the input cheque, along with a sequence of Cartesian coordinates and tags of every sub-word (connected component). The coordinates provide an unambiguous pointer to the sub-word intended by each tag.
4. **Arabic sub-word files:** each of these files contains image of a single connected component (sub-word) extracted from the cheque and its tag(s). Each file contains also a reference to the originating cheque.

File naming of both the courtesy and legal amounts was made to refer to the original cheque number from which these amounts were extracted. Sub-word and digit file names were chosen in a sequential manner.

Tagging of legal and courtesy amounts were done independently. This was intentionally made to minimise chances of complex human errors that may happen to both the legal and courtesy amounts of the same cheque.

2.4. Validation

Although the tagging tool was designed to prevent or warn for possible errors in the tagging process, yet there are still some traces of mistakes. This is particularly true when dealing with large amounts of data. Therefore, a procedure to verify the truthfulness of the tagging process is needed. In the following two sections, we describe two procedures that were developed for this purpose.

2.4.1. Automatic validation procedure

We took an early advantage of the redundancy available in bank cheques by comparing numerical values of the tags of legal and courtesy amounts of each cheque. A tagged

cheque is approved if the numerical values obtained from the tags of its legal and courtesy amounts match. Otherwise, further steps need to be taken to validate or correct the tagged legal and/or courtesy amounts.

Comparing the two amounts requires translation and interpretation of each sequence of tags into its numerical value. While this looks trivial for courtesy amounts, it involves a complex process in the case of legal amounts. First, each tag should be translated into the appropriate sub-word. Second, each proper sequence of sub-words needs to be translated into a correct word. This requires special attention since some words may appear as sub-words of a larger word (Figure 2-4). This was achieved by means of a context sensitive grammar developed for this purpose. Third, the sequence of words should be converted into a numerical value. Again, this requires special manipulation since there are various orders to write an amount in Arabic (e.g. from high order to low order).

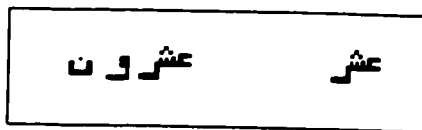


Figure 2-4: The word ten (right) appears as a sub-word in the word twenty.

As will be explained in the next section, the language of Arabic legal amounts is ambiguous. Thus, a given legal amount may produce more than one possible numerical value. In the automatic validation procedure, a tagged cheque is approved if any of the numerical values generated from its legal amount matches the numerical value generated from its courtesy amount.

In the following section, we discuss the need for ambiguous grammar to define Arabic legal amounts, then we provide an overview of the grammar used to parse and translate Arabic legal amounts followed by a section on the structure of the grammar.

2.4.1.1 The language of Arabic legal amounts

Definition 2-1: A language is a group of meaningful sentences that share similar structures.

Definition 2-2: A grammar is a formal specification of structures allowable by a given language. A grammar G is completely specified by four components: $\{N, T, S, P\}$ where:

N: the set of non-terminal symbols,

T: the set of terminal symbols,

S: the starting symbol ($S \in N$), and

P: a set of production rules $A \rightarrow B$, where $A \cap N \neq \emptyset$, and $B \in T \cup N$.

Definition 2-3: A language L is said to be ambiguous if it allows a sentence $S \in L$ to have more than one interpretation.

Lemma 2-1: The language of Arabic legal amounts is ambiguous.

Proof of lemma 2-1: To prove Lemma 1, we need to present a single example of such ambiguity. Figure 2-5 shows a legal amount with two grammatically correct interpretations. ♦

Definition 2-4: A grammar G for a language L is said to be general if and only if

1. G accepts all possible sentences of L , and
2. G provides all possible interpretations of S , $\forall S \in L$.

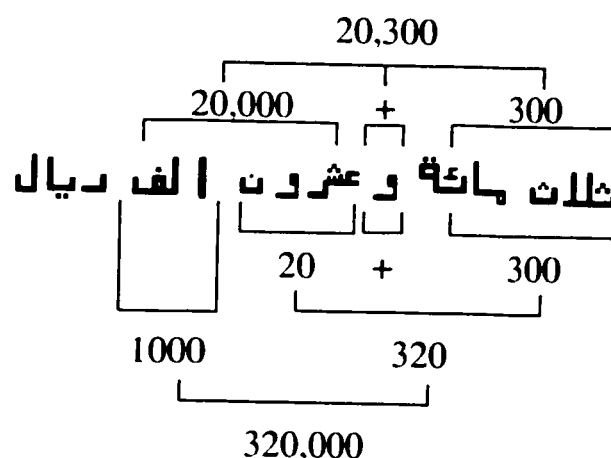


Figure 2-5: One Arabic amount with two possible interpretations.

Grammars provide a formal way to specify and interpret languages. When defining a grammar for a given language, it is essential to define a general grammar. At the same time, it is important to define a selective grammar, one that rejects most ill formed sentences.

Definition 2-5: An ambiguous grammar is one that allows more than one parse tree for one or more sentences.

Lemma 2-2: To generally define an ambiguous language, an ambiguous grammar is required.

Proof of lemma 2-2:

1. Assume that L is an ambiguous language, $S \in L$, and S has two different interpretations: $S \Rightarrow I_1$ and $S \Rightarrow I_2$.
2. Assume that G is a grammar that provides a general specification for L .
3. Since $S \Rightarrow I_1$, and since G is general, there is a parse tree T_1 derived by G , that leads to I_1 .

4. Since $S \Rightarrow I_2$, and since G is general, there is a parse tree T_2 derived by G , that leads to I_2 .
5. From 3&4 above, we see that G provides two parse trees T_1 & T_2 for the same sentence S , which implies that G is an ambiguous grammar. ♦

2.4.1.2 Grammar overview

In this section we illustrate the grammar used to parse Arabic legal amounts. The grammar rules are based on both the syntactic and grammatical rules of the Arabic legal amount language. The set of terminals T is clearly chosen as the words that compose the Arabic legal amount vocabulary, e.g. words that represent numbers and currency words. In Arabic, as in some other languages, language rules insist different constraints for various groups of numerical words. For instance, the number 200 should be written as a single word, rather than being composed of the number two and the number hundred. This is not the case with numbers from 300-900. Due to this reason, different non-terminal symbols are used to produce each group of terminals that share some characteristics. Production rules P_1 to P_3 below show examples of non-terminal symbols used to overcome some of the aforementioned difficulties.

P_1 : NumWord \rightarrow any word that has a numerical value above 0.

P_2 : FewNumWord \rightarrow words that have values between 3 and 9

P_3 : LessThanTenWord \rightarrow words with values between 1 and 9.

Unit words (e.g. currency word) have various grammatical constraints too. P_4 to P_8 are defined to produce each group of unit words.

P ₄ : Unit_1 -> 1	ريالات ريال
P ₅ : Unit_2 -> 10	عشر
P ₆ : Unit_3 -> 100	مائة
P ₇ : Unit_4 -> 1000	آلاف ألف
P ₈ : unit_5 -> 0.01	عشرات علة

2.4.1.3 Grammar structure

An Arabic legal amount could be seen as a list of terms connected to each other by a connector (and, و) as shown in P₉ below. Each term contains a number phrase followed by a unit, or simply a unit (P₁₀, P₁₁, P₁₂, and P₁₃). A number phrase could be a simple number, a complete term, or two numbers connected together (P₁₄).

P₉: S -> Term | Term Connector Term

P₁₀: Term -> Term3 | Term4

P₁₁: Term3 -> Unit1 | Unit5 | Term5

P₁₂: Term4 -> NumPhrase Unit1 | Numphrase Unit4

P₁₃: Term5 -> FewNumWord Unit3 | LessThanTenNumWord Unit2

P₁₄: Numphrase -> NumWord | Term5 | NumPhrase Connector NumPhrase

To facilitate the translation of legal amounts into numerical amounts, each grammatical symbol is assigned a numerical value. Figure 2-6 shows the algorithm used to translate Arabic legal amounts into numerical values.

1. Extract the sequence of sub-words' tags.
2. Form all correct sequences of words (using the context sensitive grammar).
3. Remove words that have no numerical value (e.g. only)
4. Find all possible parse trees for the legal amount.
5. For each term (a term contains value-word(s) followed by a unit)
6. Look for embedded terms within the current term. If any, then perform steps 5-6 to each of them.
7. Replace words (terms) by values, and evaluate the term by multiplying the term value by the term unit.
8. Sum up values of all terms.

Figure 2-6: Algorithm used to translate Arabic legal amount into a numerical value

2.4.2. Manual validation procedure

For the set of cheques that could not be validated automatically (i.e. rejected by the grammar or produced inconsistent numerical values), we have designed an interface to facilitate the manual validation process. For each legal amount, the operator may take one of the following two decisions:

1. Mark the legal amount to be re-tagged.
2. Reject the legal amount.

The same procedure is performed for rejected courtesy amounts. Marked amounts are fed back to the tagging tool, and then to the validation module.

2.5. Strength of the validation process

Theoretically speaking, our validation process may fail to detect some tagging mistakes. However, this is extremely rare in practice. We may make a point by describing what it takes to approve incorrect tagged cheque C:

1. An error would occur while tagging the legal amount of C.
2. This error should create a different, yet correct sequence of sub-words that makes a word.
3. The new sequence of words should be grammatically correct to generate a corresponding numerical value.
4. Another error should occur while tagging the corresponding courtesy amount.
5. The two independent errors in the legal and courtesy amounts should produce equivalent numerical values.

It is clear that such a sequence of events is hardly expected of a single cheque. Note that legal and courtesy amounts are tagged independently at different instances.

A question may arise about the reasons that could prevent the approval of tags of a particular cheque. Following are some of the reasons that led most unapproved-tagged cheques to this category:

1. The extraction tool may have cut the legal amount (or the courtesy amount), providing incomplete or incorrect data to the tagging tool.
2. The legal amount may have contained an unexpected spelling mistake that left the relevant sub-word untagged (tagged as OTHER symbol), leaving a gap in the legal amount.

3. The legal amount may contain a word that is out of the range covered by this study (e.g. million).
4. There may be missing sub-words (mainly letters) in the original legal amount.
5. The tagging operator may have produced some error.

Figure 2-7 shows an example of rejected cheques.

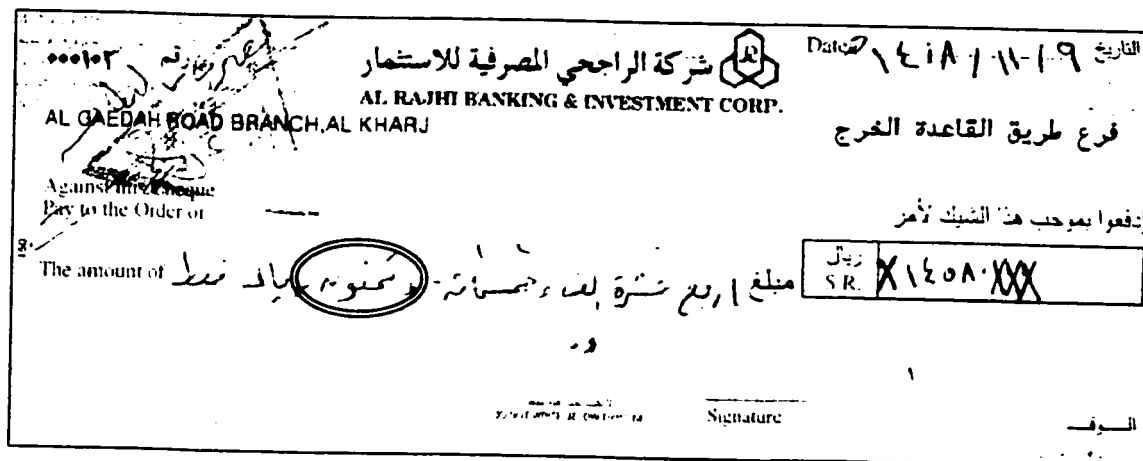


Figure 2-7: Sample of the rejected cheques due to spelling mistake.

2.6. Structure and organization of the databases

The complete validation process approved about 83% of the 3000 tagged cheques, which provided about 29,498 sub-words and 15,175 digits. Tables 2-1 and 2-2 show the distribution of the validated sub-word/digit classes (excluding touching sub-words and touching digits). Some classes are very rare, though they do exist in the lexicon of handwritten Arabic legal amounts. Such classes should remain in the lexicon although they are not very well represented. It is important to note that this validation process guarantees the correctness of the tagged legal/courtesy amounts, and all Indian digits.

Figure 2-8 shows the structure of the validation module.

Table 2-1: Distribution of the digits data set

digit	# of samples
0	5367
1	1086
2	770
3	506
4	440
5	912
6	390
7	342
8	344
9	268
Delimiter	4088
Comma	347
Total	14860

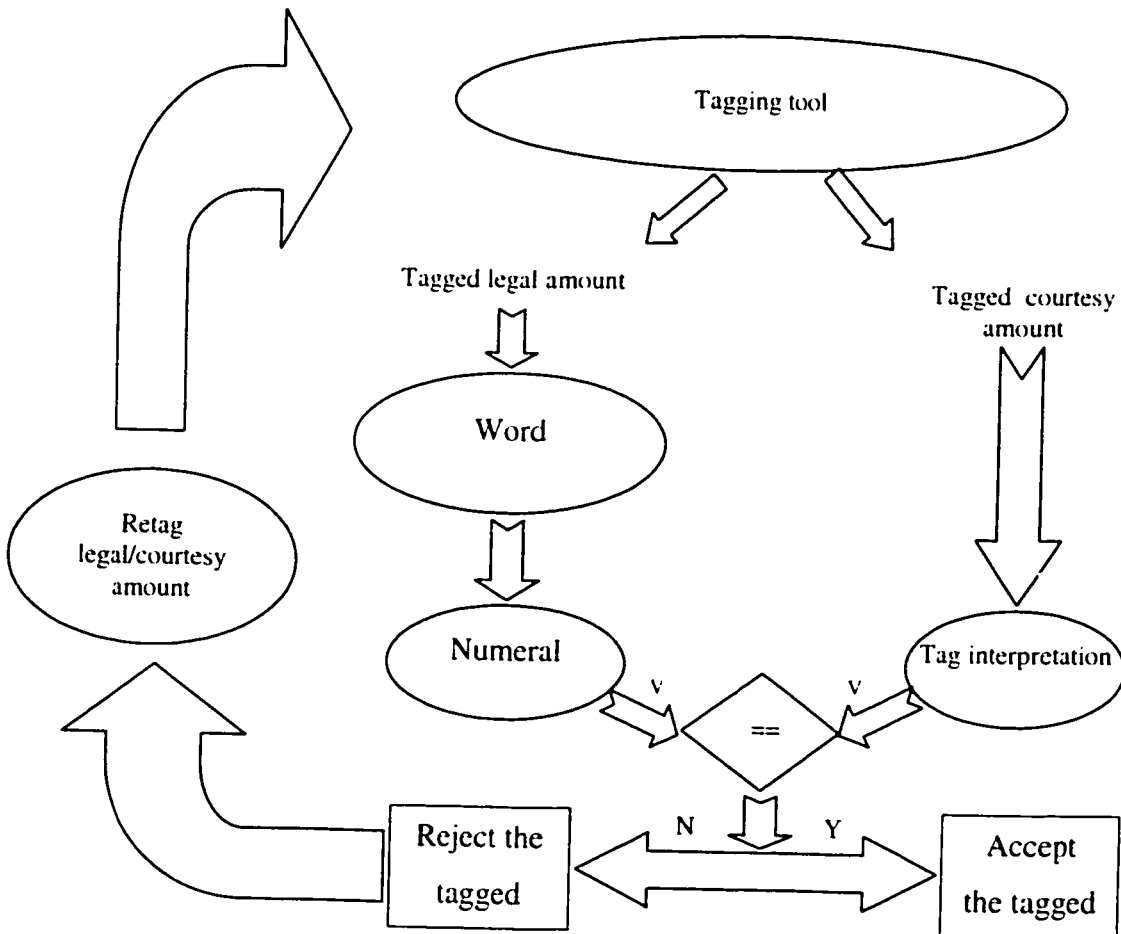


Figure 2-8: The validation process

Table 2-2: Distribution of the sub-words data set

code	Image	Count	code	Image	Count	code	Image	Count	code	Image	Count	code	Image	Count
1-00	ا	2822	2-09	لا	2450	3-16	ستا	1	6-06	وللين	0	4-23	وطني	0
1-05	ت	6	2-10	لفا	1049	3-17	سنة	160	4-01	بعيا	118	4-24	نهضة	1
1-06	ث	14	2-11	را	299	3-19	ستو	89	4-02	بعين	8	4-26	وللا	0
1-07	د	32	2-12	نور	94	3-20	سعو	34	4-03	تسعا	0	4-27	وللة	6
1-08	ر	2920	2-13	يا	1830	3-21	عشر	671	4-04	تسعة	90	5-00	بعية	1
1-09	ف	811	2-14	ين	16	3-22	عبري	1519	4-06	تسعو	58	5-02	تسعا	116
1-10	ل	1665	3-00	عنا	71	3-23	فقط	1415	4-07	ثقة	0	5-03	تسعين	1
1-11	ن	1357	3-02	عندي	4	3-24	لفا	179	4-09	تسعين	18	5-04	تسعا	263
1-12	ة	345	3-04	بعقة	222	3-25	لفا	17	4-10	تسعا	1	5-05	تسعين	14
1-14	و	2677	3-06	بمور	117	3-27	لغي	3	4-11	تسعة	392	5-06	تسعا	118
1-16	ي	53	3-07	تسعم	5	3-28	لبن	1	4-13	تسعو	213	5-07	تسعين	1
2-00	عق	1152	3-08	ظا	509	3-29	رعة	14	4-15	سبعة	121	5-08	سبعة	1
2-02	م	9	3-09	شا	440	3-31	نها	99	4-17	سعو	102	5-10	وكتين	4
2-03	شع	269	3-10	عنا	77	3-33	نية	132	4-18	ستها	132	5-11	حالات	0
2-05	شور	128	3-12	شني	20	3-35	نبن	2	4-19	ستين	5	6-00	تسعة	1
2-06	كا	93	3-13	تسبن	6	3-36	والك	2	4-20	لغبن	89	6-02	تسعة	1
2-07	ست	3	3-14	تسبن	15	4-00	تسبن	31	4-21	رعا	48	6-04	تسعة	1
2-08	قاب	111	3-15	سب	2									

This research effort has produced a number of databases that can help researchers in various topics. These databases include Arabic legal-amounts database (2,499 legal amounts), Courtesy amounts database (2,499 courtesy amounts written in Indian digits), Arabic sub-words database (29,498 sub-words within the domain of legal amount), and Indian digits database (15,175 digits). In addition, this work produced a database of complete (original) gray level cheques, which can be used for other research purposes

(c.g. date processing). Moreover, it is not difficult to derive a database of Arabic words. This is achievable using the sub-words database. It is also possible to generate a database of Arabic dates from the Arabic cheques database.

Each database mentioned above is divided into training and testing sets. The training set includes 66% to 75% of the available data. That is true for legal amounts, courtesy amounts, Indian digit classes and most sub-word classes. In few sub-word classes, this condition could not be satisfied due to insufficient samples. This ratio was chosen to provide enough training samples on one hand, and to give enough measurement of the generality of recognition systems on the other hand. The division between training and testing data was done randomly with restrictions to ensure the ratio mentioned above.

Training and testing data sets are further divided into two sets: touching amounts and non-touching amounts. A courtesy amount is located in the touching set if it contains at least one pair of touching components. The same can be said about the database of legal amounts.

Data sets of the non-touching Indian digits and the sub-words are further divided based on their class (i.e. each class is located in a separate directory). The number of classes defined for Indian digits and Arabic sub-words were 11 and 87 respectively.

Training data of each of the legal amounts, courtesy amounts, Indian digits and Arabic sub-words databases are extracted from the same set of cheques. Table 2-3 shows the sizes of training and testing sets in each of the above four databases. The above mentioned databases are all available in tiff format. Figure 2-9 shows structures and inter-relation between the courtesy amount database and the digits database. Similar relation ship applies to the legal amounts database and the sub-words database.

Table 2-3: Distribution of databases between training and testing sets.

	Sub-words (# of samples)	Digits (# of samples)	Legal amounts (# of cheques)	Courtesy amount (# of cheques)
Training				
Touching	1066	243	838	266
Not touching	19813	10536	941	1513
Total	20879	10779	1779	1779
Testing				
Touching	447	72	321	94
Not touching	8172	4324	399	626
Total	8619	4396	720	720

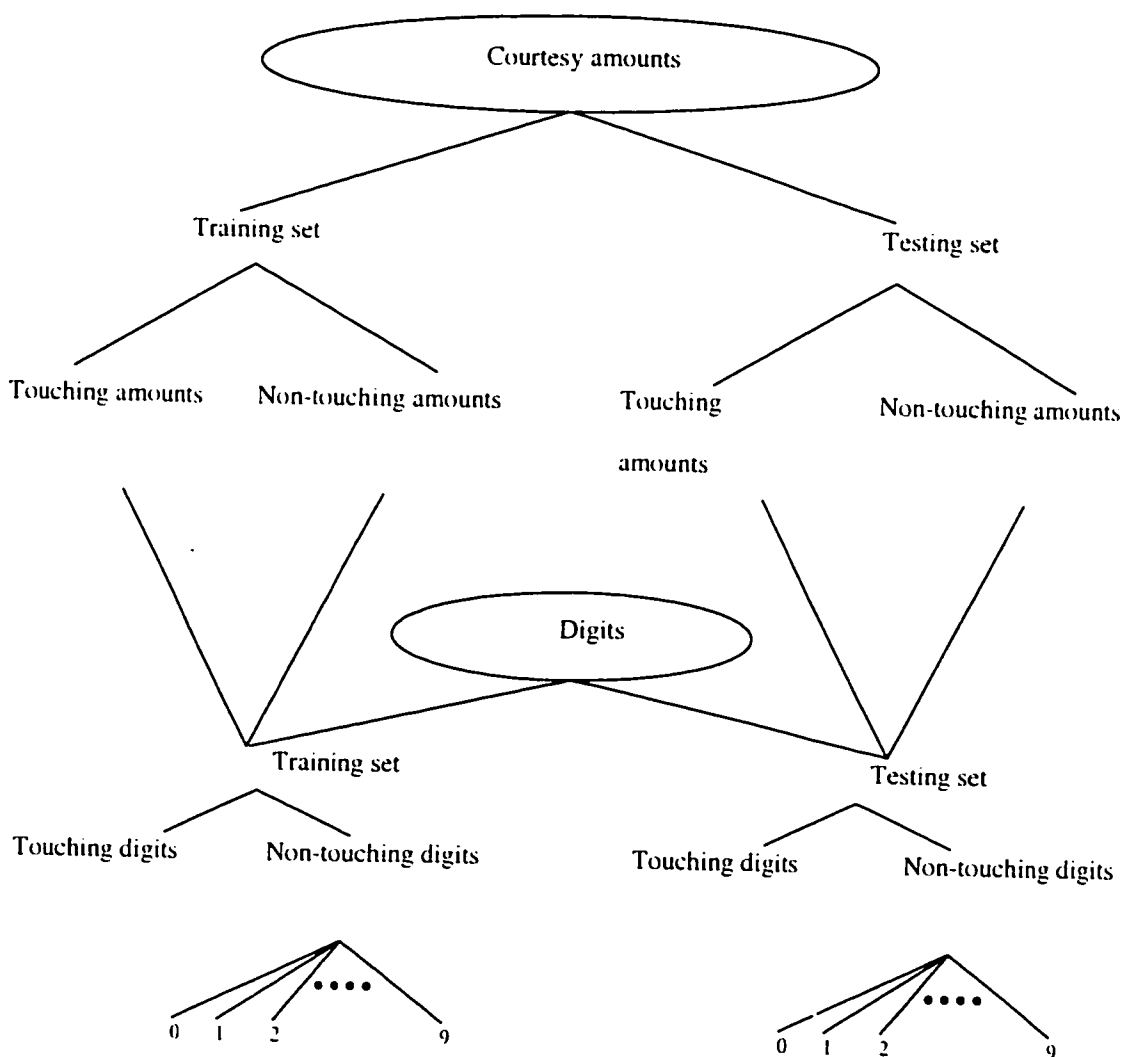


Figure 2-9: Structure of the courtesy amount and digits databases

Chapter 3

Sub-word Recognition System

Recognizing connected components (letters, words or sub-words) is a major step towards processing legal amounts. Significant research work has been recorded toward word recognition in the context of legal amounts in Latin cheques. Guillevic and Suen [GS98] used a combination of global feature scheme with an HMM to recognize Latin words. Kaufmann and Bunke [KB00a] developed a complete system to read legal amounts extracted from German cheques. Statistical features are extracted from each component and are fed to an HMM classifier. Contextual information is used to post process the classification results.

A number of attempts have already been made to process Arabic handwriting. Miled, Cheriet and Olivier [MC98] applied global and analytical features in two HMM levels to recognize Arabic city names. At one stage, HMM recognizes characters composing the input word, while the second stage integrates sequential HMMs to recognize a complete word. Abuhaiba et al. [AM94] converted the skeleton of Arabic characters into a tree structure, and matched it with fuzzy constrained character graph models. In Abuhaiba [Ab98], a thinned image is converted into straight-line approximations. Tokens are then recognized using fuzzy sequential machines.

In this chapter, we give a detailed description of the system that has been designed and implemented in this study to process Arabic sub-words within the context of legal amounts. The system is based on global and analytical models. Two main contributions are emphasized in this chapter: (1) a sequential algorithm to estimate the

pen trajectory from off-line handwritten data, and (2) the introduction of termination state probability in the design of our HMM models.

3.1. Design and Methodology

Recognition aims to map each input image into its proper code (tag). Note that the current study is uniquely complex in a number of aspects. First, it is complex in terms of the shapes of the patterns involved. Cursive writing (as the case of Arabic) is obviously more complex than non-cursive writing. In addition, Handwriting is more complex than printed text. Moreover, Arabic allows various shapes for characters based on their location within words. Second, the current study involves real-world data captured from a noisy environment. Significant amounts of noise increases the difficulty level of the recognition problem. Third, the number of classes involved in the current study exceeds those involved in similar studies for English and French cheques. This increases the ambiguity of the current recognition problem as well.

The aforementioned complexity of recognition problem of Arabic sub-words imposes the use of a highly discriminative classification scheme. Such a scheme should employ all useful information to determine the identity of the input sub-word. One way to solve the recognition problem is to apply a stage-wise classification, where each stage provides a partial solution of the recognition problem. It would be advantageous to have different classifiers, employing different types of features, in the stage-wise classification process.

In this thesis, we used multiple classifiers with different sets of features. A global-analytical reading model is designed to recognize sub-word patterns. First, a general impression is drawn from the global shape of the input image. This step reduces the

lexicon size. Second, an analytical model is used to detect and use analytical characteristics to determine the exact identify of the input image.

The global model is designed to provide fast and reliable results. Pixel distribution of the input image is used to define a broad basis for categorization. A neural network is used at this level for its fast decision making process. A Kohonen neural network is employed for its robustness in defining proper grouping (clustering) among the input data.

The analytical model uses analytical features to identify the input image. Due to its robust and transparent decision making process, HMM based classifier is used at this level. The input image is described by the sequence of strokes that were used to draw it. Though very expensive to estimate, this description provides an efficient sequential description of the 2-D image.

Figure 3-1 shows a general view of the recognition model used in this study. The input sub-word image is first passed to the global model to give a prediction of all possible classes. These possible classes are then used by the analytical model, along with the original input image, to achieve a sorted list of possibilities about the identity of the input image. Thus, the global model reduces the ambiguity of the problem, and increases the reliability of the analytical model. In the following few sections, we give a detailed description of each of the recognition models.

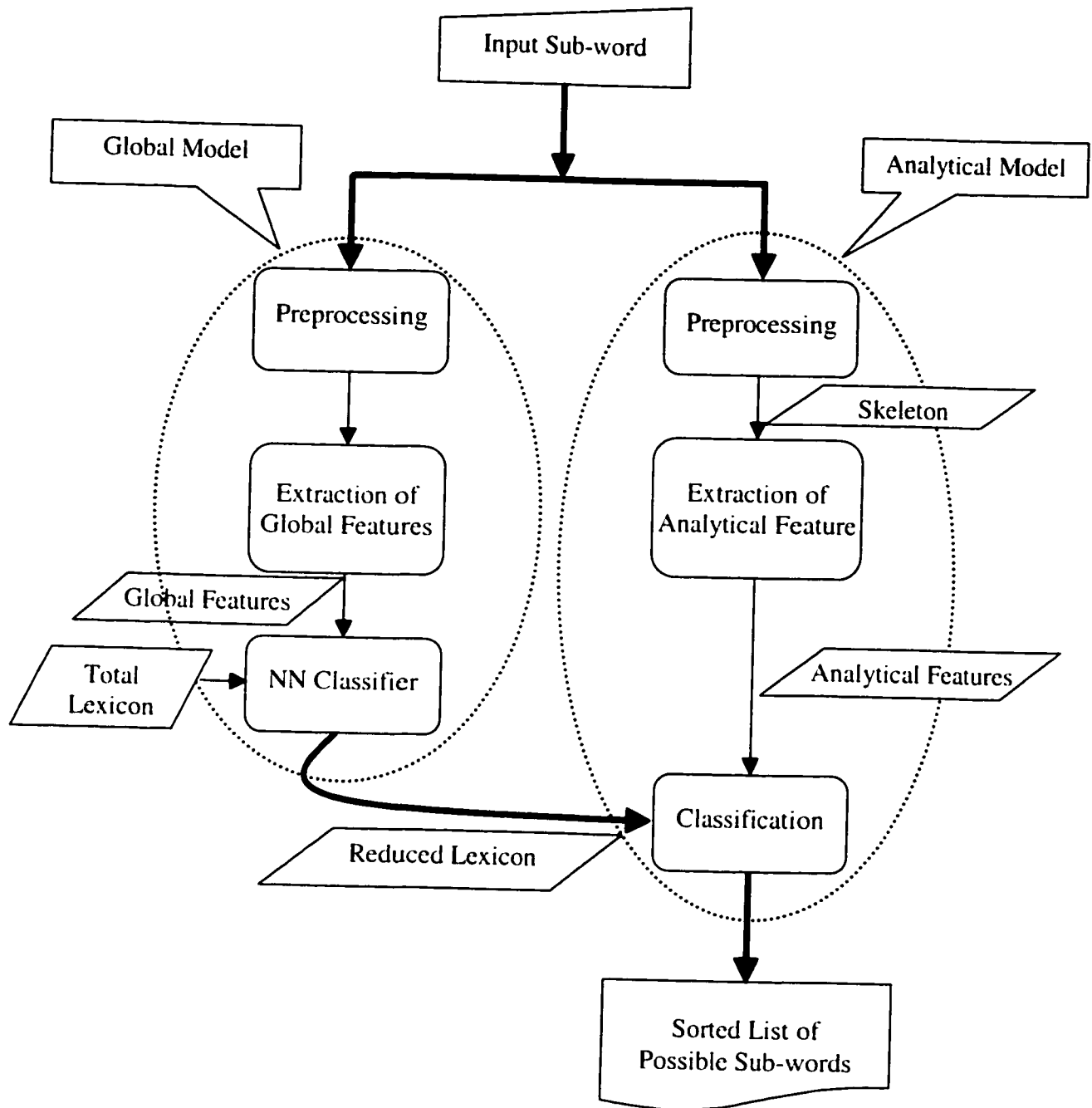


Figure 3-1: Overview of the sub-word recognition system

3.2. Global model

The global recognition model intends to provide fast and reliable lexicon reduction based on global vision of the input sub-word. Images are first normalized to a fixed size (120x200 pixels). Figure 3-2 shows a sample image before and after this operation. Features are based on the global distribution of pixels within the input image. An NN based classifier is used at this stage for its efficiency and simplicity. Due to the large number of classes, it is impractical to expect direct recognition result from the NN. Therefore, we designed an NN to provide a reduced lexicon, which yields a faster and more accurate recognition at the second stage. This was achieved using a Kohonen neural network.

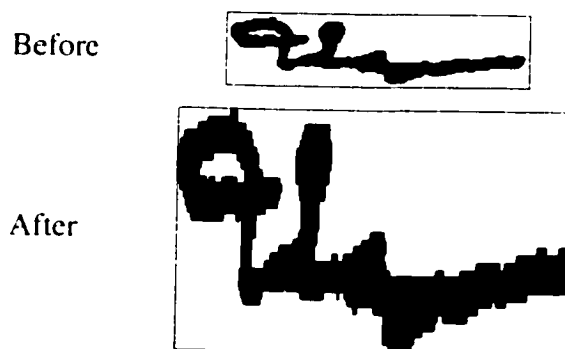


Figure 3-2: Sub-word image before and after size normalization

3.2.1. Feature extraction

We used a set of features that reflect a general view of the input word in terms of pixel distribution. The input image is partitioned into grids of size 10 x 10 (pixels). Pixel densities are computed from each portion, producing feature vectors of dimension 240.

Figure 3-3 shows the partitioning of the sample shown in Figure 3-2 above, and the corresponding feature values.

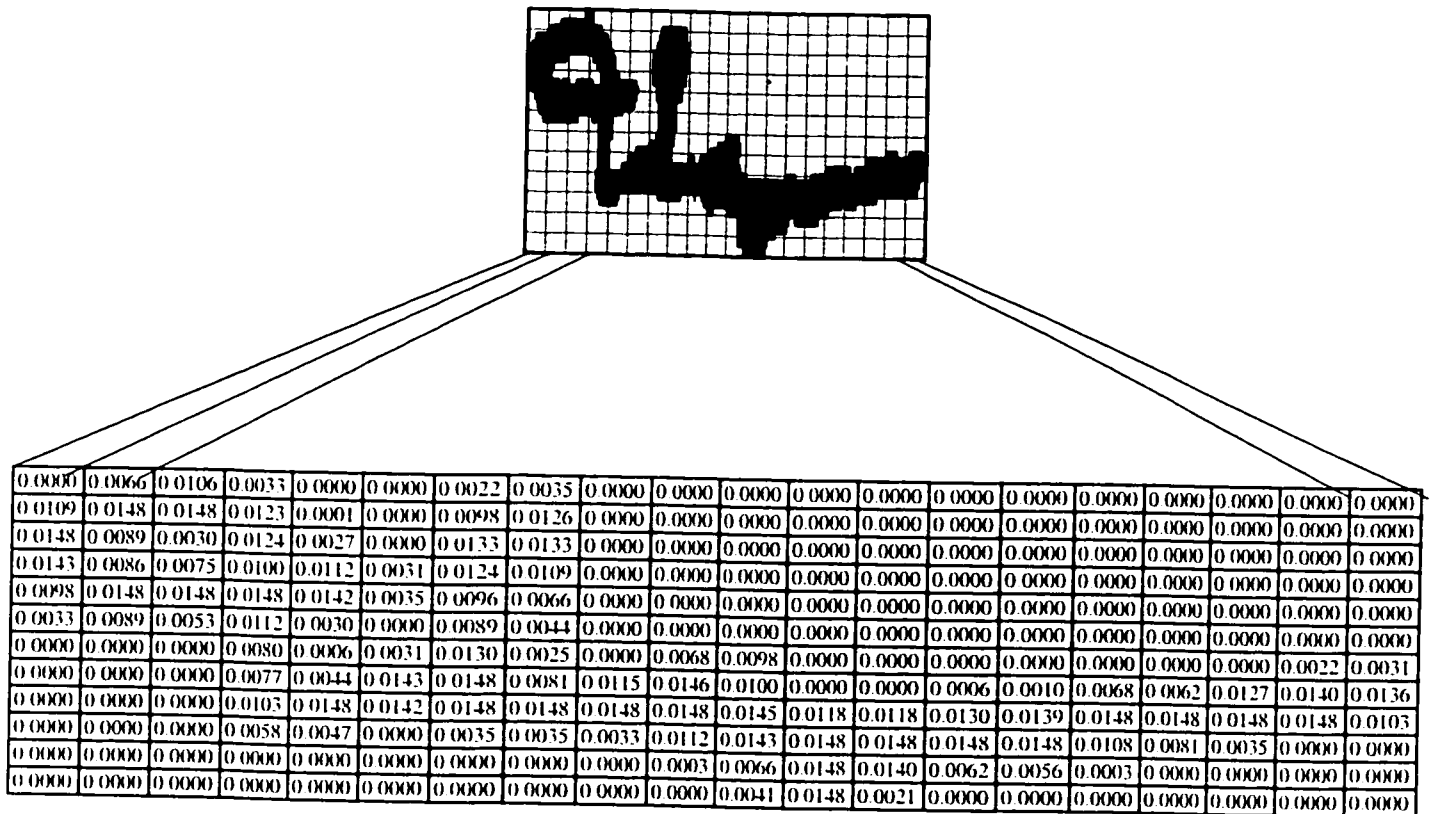


Figure 3-3: Features extracted from Figure 3-2 above, in the global model

Possible correlation among vectors in the feature space is eliminated using the principal component analysis transform. The covariance matrix is first computed, to determine the eigenvectors. Input vectors are then multiplied by the matrix of eigenvectors to produce un-correlated vectors.

Reduction of the feature space dimensions is very important for time and memory concerns. Out of the un-correlated vectors, we have eliminated those with small variance. This process resulted in feature vector of size 80.

3.2.2. Classification

We used a 3-layers Kohonen neural network to perform rough clustering of the input data. The input layer consists of 80 neurons corresponding to the dimension of the feature vectors. The second layer contains 20 competitive neurons (corresponding to 20 target clusters). The number of clusters was determined experimentally to allow efficient and reliable grouping of the training feature vectors. The output of the network is an integer between 1 and 20 inclusively that corresponds to the winning competitive neuron. A single neuron is used for that (Figure 3-4).

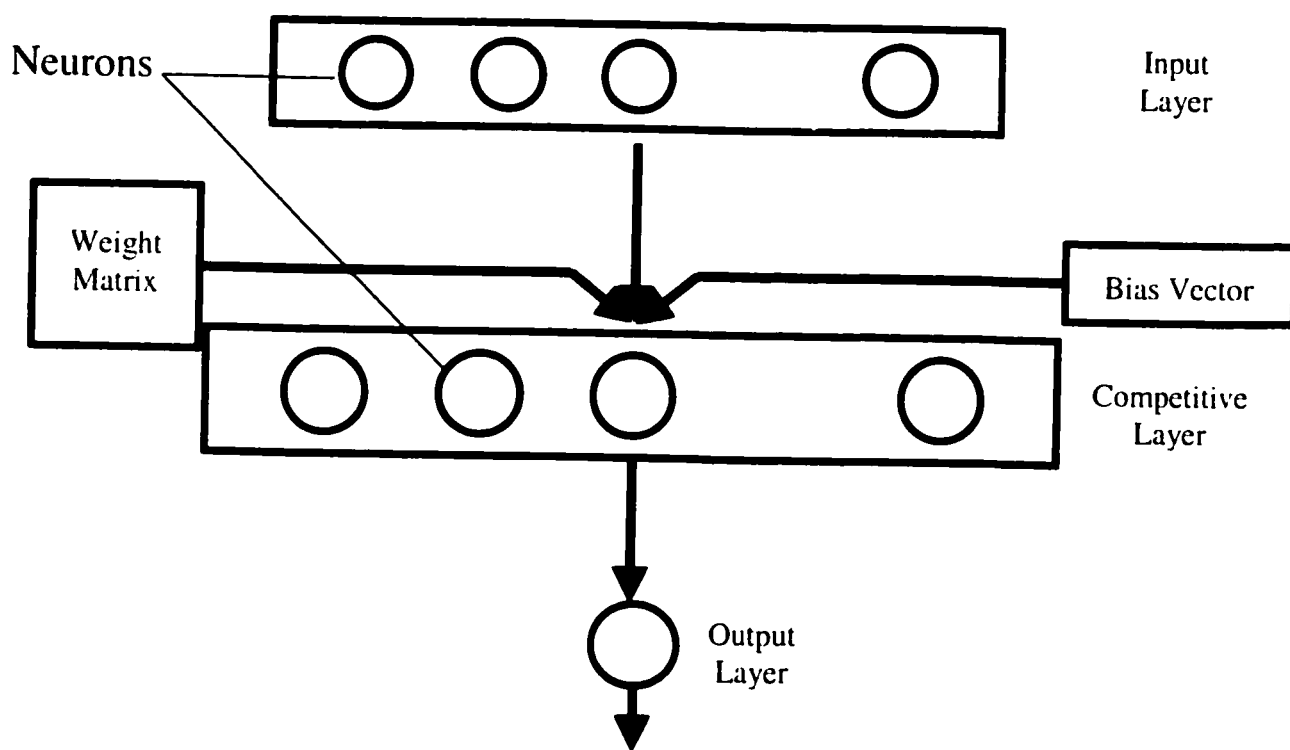


Figure 3-4: NN Classifier

3.2.2.1 The Learning phase

Unsupervised learning is applied to produce suitable clustering of the training feature vectors. Input vectors are fed to the input layer in a sequential manner. Distances between an input vector and weights of neurons in the competitive layer are computed according to equation 3-1 below.

$$D^i = \sqrt{\sum (\omega^i - X)^2} \quad (3-1)$$

where D^i is the distance between the input feature vector and the i^{th} neuron, ω^i is the weight vector for the i^{th} neuron and X is the input feature vector.

A similarity measure is computed based on the distance vector D and the bias vector (Eq 3-2) and is then fed to a competitive transfer function (Eq 3-3).

$$S^i = -D^i + \beta^i \quad (3-2)$$

$$V = \arg \max_{\forall i} (S^i) \quad (3-3)$$

where S^i is the degree of similarity between the input vector and the i^{th} cluster, β^i is the bias associated with the i^{th} neuron and V is the index of the winning neuron.

Weights of the winning neuron are then adjusted according to the Kohonen learning rule (Eq 3-4).

$$\omega_{new}^v = \omega_{old}^v + \alpha(X - \omega_{old}^v) \quad (3-4)$$

where, ω^v is the weights of the winning neuron and α is the learning rate. Only weights of the winning neuron are allowed to adjust toward the input vector. This adjustment

makes the winning neuron more likely to win a future input vector that is similar to the current input, and less likely to win a feature vector that is completely different from the current one. As more input vectors are presented to the network, each neuron that is closest to a group of inputs adjusts its weights vector toward this group.

The learning rate affects the learning speed and its accuracy. Higher learning rate implies larger steps toward the optimal weight vector, which means faster learning. However, larger steps may cause more adjustment than what is necessary, making the learning process more vulnerable to miss the ideal weight values. The learning rate in this study is chosen large enough to allow fast learning at the beginning of the learning phase, and is adjusted to a smaller value toward the end of the training process to allow more accurate converging steps [FS92].

The bias vector is used to balance the attraction power for all neurons during the training phase. After each training pattern, the bias vector is adjusted to reflect a running average of neurons' outputs as shown in equations 3-5 and 3-6 below.

$$C_{new}^i = (1 - \alpha_b)C_{old}^i + \alpha_b a \quad (3-5)$$

$$\beta^i = \exp(1 - \log(C_{new}^i)) \quad (3-6)$$

where C is the conscience vector, α_b is the bias learning rate and a is a vector of zeros for the all neurons but the winning one. Equation 3-5 above computes the running average of each neuron giving a constant extra weight (α_b , where $\alpha_b \in [0,1]$) to the winning neuron. The increase of the conscience value of the winning neuron in Eq3-5 causes a decrease in its bias value in Eq3-6, which in turn causes less support for the winning neuron in Eq3-2

above. Neurons that responded to very few patterns on the other hand have their conscience values reduced in equation 3-5, which gives them larger bias values in Eq 3-6, allowing such neurons to come closer to some set of samples. By allowing all neurons to participate in the clustering process, the bias vector encourages even distribution among all neurons by assigning more neurons to dense areas of the feature space [Mw99].

The result from the training session of the Kohonen network is used as a basis to define basic clustering of the problem based on global features. Input vectors are surveyed to detect the classes contained in each cluster. The number of clusters was chosen experimentally to reflect low error ratio on one hand, and to provide a significant simplification to the recognition problem on the other hand. The average number of classes per cluster was found to be 24.

3.2.2.2 The Testing phase

During the testing phase, feature vectors are fed to the input layer. The winning neuron at the output layer, which corresponds to the nearest cluster, is determined by the NN. The mapping is considered correct if the class of the input feature vector is found to be one of the classes composing the nearest cluster. Detailed results are discussed in the next chapter.

3.3. Analytical model

The analytical recognition model complements the global model with the purpose of providing a sorted list of probable classes that match the input image, based on analytical features. HMM is used to provide probabilistic similarity measures based on sequential

observations. Sequential observations are detected from the input 2-D image based on an estimation of the original writing sequence of strokes.

3.3.1. Pre-processing

To facilitate easier extraction of analytical features, skeletonization is applied to the input sub-word image. A sequential iterative thinning algorithm is used for this purpose Figure 3-5 shows a sample image from the training data, and its skeleton [Mw99].



Figure 3-5: A sample sub-word image and its skeleton

3.3.2. Feature extraction

The aim of this process is to generate 1-D descriptions out 2-D skeletons. A graph representation of the skeleton is first built and then transformed to a tree. Tree transformation is simpler and faster than Eulerian and Hamiltonian graph transformations used in [AH94], [Ja96] and [KY00]. Pen-trajectory is then estimated by finding the most efficient traversal of all edges in the transformed tree. In the following, we give more details about each of these steps.

Graph representation is achieved by labeling the skeleton into feature points and curve segments. Feature points include end points and junction segments. A junction segment is a cluster of adjacent junction points. The labeled image is then transformed into a weighted graph $G=(V,E)$ where the set of vertices V includes all feature points and the set of edges E includes all curve segments in the skeleton. The weight of each edge is set equal to the length (in pixels) of its corresponding curve segment. This representation requires much fewer vertices and therefore less amount of resources, than the one applied in [AH94]. Pen trajectory can then be estimated by a proper traversal of all edges in G .

To reduce the time complexity of the estimation process, the graph G is transformed into a tree by eliminating all cycles. Cycles do not affect the sequence of vertices in the most efficient traversal walk. There are two types of cycles that need to be removed. Simple cycles include only one vertex, and their removal is quite an obvious task. Complex cycles involve more than one vertex and are a direct result of unnecessary feature points in the skeleton of handwritten words. Removal of such a cycle can be achieved by mapping all vertices involved in the cycle into a single vertex. This reduces the cycle into a simple one that can be removed easily by removing the single edge that connects the newly formed vertex to itself. The removed vertices and edges can be retracted after we have determined the most efficient walk to traverse the resulting tree.

In a tree, the most efficient walk to traverse all edges should start and end at the furthest two pendant vertices. This ensures the most savings in double-traced edges. However, applications may incur additional rules to determine the start and end vertices. Once these vertices are determined, re-ordering all other vertices does not affect the

overall cost of the traversal process (as long as no edge is traversed unnecessarily). Details of the traversal algorithm are given in the next section.

After determining the best walk in the tree, cycles can be restored based on other factors, e.g. vision rules. The inclusion of such vertices and edges does not affect the efficiency of the best walk. This is one of the major advantages of our transformation algorithm over a general minimum spanning tree algorithm. The algorithm was designed for handwritten letters/words in the domain of Arabic legal amounts. However, it is applicable to other domains with suitable adjustments.

Each segment is then represented by a sequence of Freeman chain codes (Figure 3-6). Linear approximation is applied next to extract a shorter description of the chain code list. The approximation process may result in breaking some segments into several strokes based on direction variations. It may also allow points from consecutive segments to be merged into one stroke. This minimizes the effect of unnecessary junctions. Each stroke is then represented by two values: length and direction. Stroke length is normalized to the length of the original skeleton. Figures 3-8 and 3-9 show tabulated result of this operation and its graphical presentation.

56567677770777665545444434444444444344444444
5344444454434444444444444444444444534544544544
4554554555545555555545445554445444455444444

Figure 3-6: The chain code sequence (from left to right) extracted from Figure 3-5

3	2	1
4	<i>P</i>	0
5	6	7

Figure 3-7: The Freeman chain code

Direction	Length	Normalized Length
5.33	3	0.0214
6.76	14	0.1000
4.24	123	0.8786

Figure 3-8: Linear approximation of the chain code sequence shown in Figure 3-6

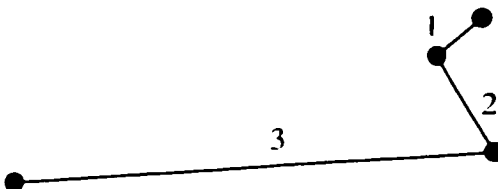


Figure 3-9: Graphical presentation of Figure 3-8

3.3.3. Estimation of pen trajectory of hand-written sub-words

As mentioned earlier, the pen-trajectory is estimated by minimizing the traversal cost of the input image. Curve segments of the labeled skeleton are represented by a weighted graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. In the current application, the vertices represent end points and junction segments, while the edges represent curve segments. The weight of a given edge is defined as the length of its corresponding curve segments. With such a representation, some of the good findings of the graph theory become applicable. The pen trajectory can then be estimated by minimum-cost path taken to traverse all edges in the graph G . Adjacency matrix is built to reflect the neighborhood of each feature point. Basic definitions about the graph theory can be found in [ST80], and are summarized in Appendix B.

Loops are temporarily removed from the graph. This ensures that the composed graph becomes a tree, which makes subsequent processing steps more efficient.

The minimum and maximum costs for traversing all edges in a graph G are given by C_{\min} and C_{\max} as shown below:

$$C_{\min} = \sum_i E_i \quad (3-7)$$

$$C_{\max} = 2\sum_i E_i \quad (3-8)$$

where C is the cost of traversal. The maximum cost is encountered by traversing each edge twice (e.g. traversing a star starting from and ending at its center). The cost could be made more efficient by minimizing the total length of edges that need to be traced twice (double traces). It has been shown that, except for the first and last vertices, all vertices of odd degree require some edges to be traced twice. In fact, it can be shown that, in a tree, each internal vertex v would cause either $d(v)$ or $|d(v)-2|$ double traces, where $d(v)$ is the degree of the vertex v (see Theorem B-5 in Appendix B).

Selection of the starting vertex affects the overall traversal cost since it could impose unnecessary double traces. Therefore, the starting vertex is chosen based on its degree and length of its edges. Starting with a vertex of odd degree saves a single double trace. The longer the first edge is, the more we save on the overall cost. In addition, practical considerations impose a generally right-left traversal of Arabic sub-words. Therefore, vertices that represent the rightmost feature points are more likely to be selected as starting points.

Having the starting vertex determined, Algorithm 3-1 is applied to find the most efficient traversal path of the composed graph. Since we now deal with a tree, visiting all vertices ensures traversing all edges and vice versa.

Relocation of the removed loops is performed in an obvious manner. Traversal of a loop edge at a given vertex v can be performed upon arrival to the vertex v . If the

degree of v is more than 4, traversal could be delayed, but not to the end. Such a delay could be necessary to account for other practical writing constraints. Delaying the loop traversal does not affect the cost efficiency as long as it is not left as the last edge for the vertex v (unless v is the last vertex to be traversed in the graph). Figure 3-10 shows the result of applying Algorithm 3-1 to three different words.

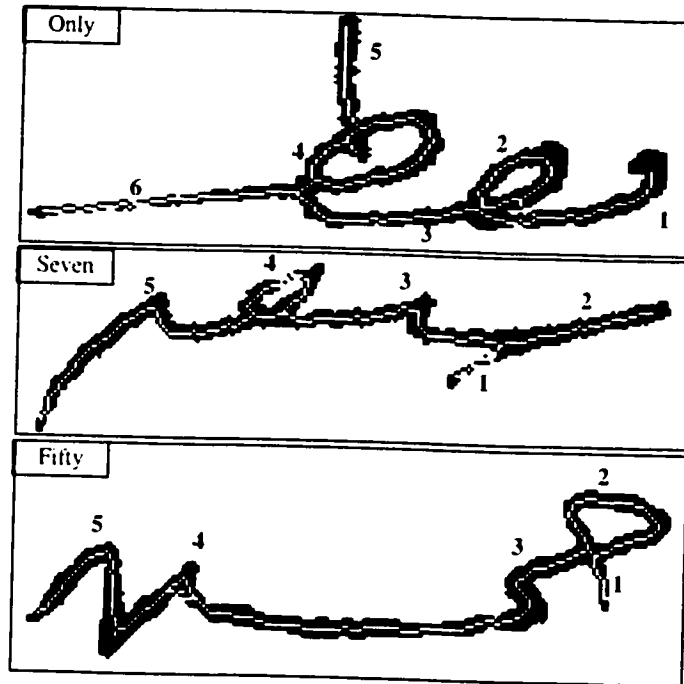


Figure 3-10: Three handwritten Arabic words (only, seven and fifty) with their skeletons and the estimated sequences of segments

Algorithm 3-1:

Objective: Find the shortest path to visit all the vertices in a given weighted graph $G = (V, E)$ starting from vertex v_s . G must be a tree.

1. Find the furthest pendant vertex from v_s , call it v_e (the ending vertex).
2. Apply algorithm 3-2 below to find the best path in G from v_s to v_e .

Algorithm 3-2:

Objective: Find the shortest path that visits all the vertices in a weighted graph $G = (V, E)$, starting from a given vertex v_s , and ending at a given vertex v_e . G must be a tree.

1. Pick up an adjacent vertex to v_e , v_i , where $e_i(v_i, v_e) \in E$.
2. Eliminate e_i and focus on the connected component $G_i = (V_i, E_i)$ that does not contain v_e . Find the shortest path R_i that traverses all vertices in G_i and ends at v_i . Traversal should start from v_i unless $v_s \in V_i$, in which case traversal should start from v_s .
3. Append v_e to R_i so that the new path will terminate in v_e .
4. If $v_s \notin V_i$ and v_i is the first adjacent vertex considered for v_e , then precede R_i by v_e , as the complete route would start from v_s and pass through v_e to R_i .
5. Repeat steps 1-3 for all adjacent neighbors of v_e .

Pseudo code for Algorithm 3-2

Function find_best_path(G, v_s, v_e)

1. *InitialPath* = ();
2. For $i = 1 : \text{degree}(v_e)$
 - Find $e_i = (v_e, v_i)$.
 - Split G into two connected components by removing e_i .
 - Construct sub graph $G_i = (V_i, E_i)$, the connected component that does not contain v_e .
 - If $v_s \in G_i$
 - $P_i = \text{find_best_path}(G, v_s, v_i)$;
 - $P_i = (P_i, v_e)$;
 - InitialPath* = P_i ;
 - $P_i = ()$;
 - Else
 - $P_i = \text{find_best_path}(G, v_i, v_i)$,

```

     $P_i = (P_i, v_r);$ 
    If  $i=1$ 
         $P_i = (v_r, P_i);$ 
    Endif
Endif

```

Endfor

3. Return (*InitialPath*, P_1 , P_2 , ..., P_k) Where $k = \text{degree}(v_r)$.

Notes on algorithm 3-2

a) Correctness of the algorithm: Correctness of the above algorithm can be easily proved by mathematical induction as follows:

1. The algorithm works for a tree of 2 vertices, starting from any of the two vertices and ending at any of them.
2. Assume that the algorithm works for all trees of n vertices for any starting vertex v_s and ending at any vertex v_r .
3. The following is to prove that the algorithm works correctly for a tree T of $n+1$ vertices for a given starting vertex v_s :
 - Find the pendant vertex v_i that is furthest from v_s (note that a tree of more than 1 vertex has at least two pendant vertices, see Appendix B).
 - Name v_j as the vertex adjacent to v_i .
 - Now, eliminate v_i from T , to get another tree T_2 of n vertices.
 - From (2) above, we know that the algorithm works for T_2 since it contains n vertices. Thus we can use the algorithm to find the most efficient path P_2 to traverse all vertices in T_2 starting from v_s and ending at v_j .
 - Construct P_1 by adding a single edge (v_j, v_i) to P_2 . P_1 traverses all vertices in T starting from v_s and ending at v_i .

Note that the minimum cost to traverse any tree can not go below the sum of all its edges. By adding the cost of (v_j, v_i) only once, we are adding the minimum cost for traversing an additional vertex and this guarantees the minimum cost of P_1 . ♦

b) Complexity of the algorithm: Consider a tree of n vertices. Step 1 could be repeated at most $(n-1)$ times, as this is the maximum number of edges in such a graph. However, in this case each sub-graph contains an isolated vertex that requires constant time to be traversed. That is $O(n)$. In the opposite case, step 1 could be repeated only once, to process a tree of $(n-1)$ vertices. On the average, step 1 would be repeated \sqrt{n} times each of which may process a tree of \sqrt{n} vertices. That gives an $O(n)$ complexity.

c) Applicability of the good continuity rule: Internal paths do not affect the overall cost of the most efficient route. This includes any path where the traversal of a sub-graph starts and ends at the same intermediate vertex. Therefore, the good continuity rule can be applied there without any compromise. Examples of such a case include v_5 in Figure 3-11.

3.3.4. Vector quantization

As we intend to use discrete HMM, observation vectors should be mapped into a discrete domain. A codebook is used to map each pair of direction and length values to a single observation. The optimum size of the codebook was computed experimentally, and found to be 170, as will be explained in the next chapter.

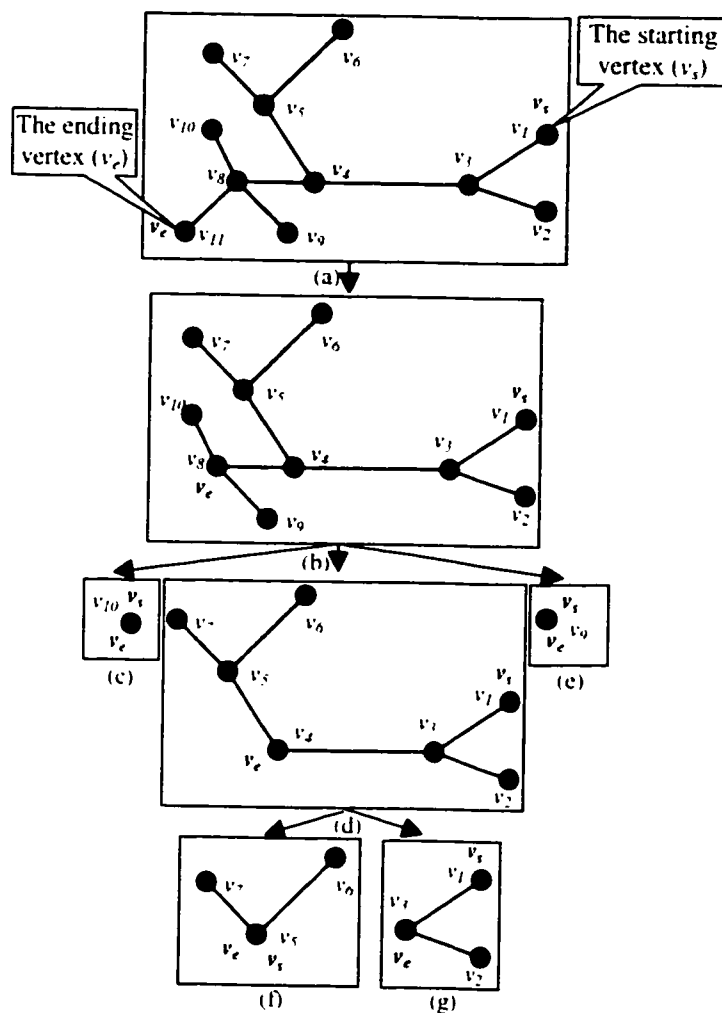


Figure 3-11: Partial application of Algorithm 2 to a general graph

3.3.5. Clustering

Clustering is needed to overcome the within-class variations in human handwriting. The goal of this process is to partition each class into clusters that share similar feature vectors. Clustering is performed on the extracted sequence of features.

The number of clusters per class is computed in two steps. First, ISODATA algorithm is used to approximate the proper number of clusters for each class using unified threshold values. After that, the number of clusters for each class is manually estimated, based on ISODATA approximation and on the prior knowledge of the

variations of each sub-word. The total number of clusters used was 150, representing 67 different classes.

3.3.6. Classification

A left-to-right HMM is used to model each cluster. The model of a cluster ω_i , is noted λ_i (N_i, π_i, A_i, B_i) where N_i represents the number of states, π_i is the initial probability at each state, A_i is the matrix of transition probabilities and B_i is the emission probabilities (the priori probabilities). The number of possible observations, M , is shared for all models. The number of states in each model is defined in relation to the number of letters within each sub-word, and therefore is unique for all models of the same class. This is used to account for the variability in width and structural complexities of different sub-words. Models are trained using the Baum-Welch algorithm. Each model is trained only to feature vectors that belong to it. Therefore, each model learns to produce high probability to similar inputs, but does not learn to produce low probabilities to different inputs. This is known as the Maximum Likelihood (ML) learning criterion.

Unlike the standard left-right HMM model, we perform re-estimation of π , allowing $\pi(1)$ to be less than 1. In such a case, $\pi(i)$, for any $i > 1$ could be above zero as the summation of π should always be 1. This is intended to account for missing observation at the beginning of the observation sequence.

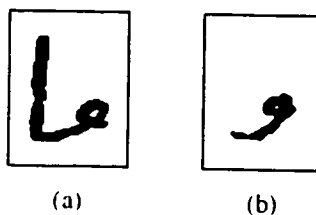


Figure 3-12: Two different sub-words that share their initial portions (starting from right to left)

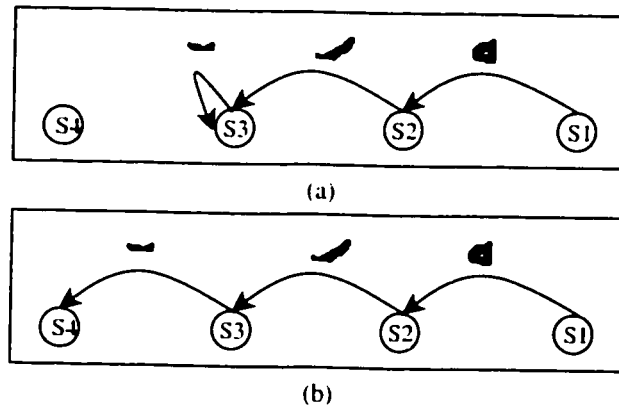


Figure 3-13: Pattern (b) from Figure 3-12 above, passed through both models (a) and (b)

Researchers used different criteria to evaluate the overall response P_k of a given model λ_k . Some of which are as follows:

1. By adding the probabilities of being in each state of the model λ_k after presenting the last observation of the input sequence. That is $\sum \alpha_T(i)$, \forall states i , where $\alpha_T(i)$ is the probability of being in state S_i at time T , and T is the length of observation vector O .
2. By considering the probability of being in the last state at the end of observation vector O . That is $\alpha_T(N)$, where N is the last state in the model λ_k .
3. By adding the probability of being in each state of the model, considering only the best path taken by the observation vector, $\delta_T(i)$, \forall states S_i .

The first and third criteria accounts for any partial similarity of the input feature vector to the model λ_k , while the second criterion accounts for complete similarities only. The advantage of the second criterion is to discriminate between longer observation vectors and short ones that share a significant part. However, the models become very sensitive to small variations.

HMM may show significant weakness in recognizing such patterns in applications where these cases happen frequently. An example of partial similarity is shown in Figure 3-12 below. The initial part of pattern (b), the right most part of the pattern, is similar to the initial part of pattern (a). Thus, pattern (b) will pass through the states of both model (a) and (b) with high probabilities. If the system allows its models to terminate in any state without penalizing improper termination states, pattern (b) may fall in the category of model (a). This possibility is shown in Figure 3-13. On the other hand, a noisy sample of pattern (a) that has lost the left most part of its shape can easily be mapped to class (b) if the system allows the termination only in one particular state (the last state). Figure 3-14 shows another similar situation with two other patterns.

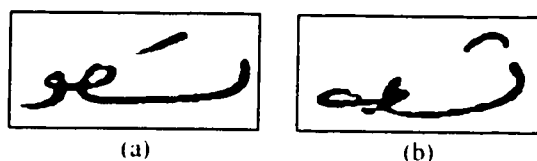


Figure 3-14: Two different sub-words that share their initial portion (starting from right to left)

In this thesis, we proposed a new criterion that combines the above mentioned advantages. To that aim, we introduced a new parameter ϕ that controls the termination probability at each state. Thus, a model will be allowed to terminate at any state, but will be evaluated based on where it terminates the observation sequence. The effects of the new parameter on each of the training and testing processes are given in the following two sections. Details of the training procedure of general HMM models can be found in [Ra89] and is described briefly in Appendix C.

3.3.7. Effect of the termination probability on the training procedure

First, the following is a brief description of the terminology used in this section:

λ : the HMM model, which includes model parameters (N , a , b , π , and φ),

O : the observation sequence,

N : number of states,

T : length of the observation vector,

a_{ij} : the transition probability from state S_i to state S_j ,

$b_i(o)$: the probability of observation o at state S_i ,

π_i : the initial state probability of state S_i ,

$\alpha_t(i)$: the probability of the partial observation sequence O_1 to O_t and state S_i at time t , given the observation sequence O and the model λ ,

$\beta_t(i)$: the probability of the partial observation sequence O_{t+1} to O_T and state S_i at time t , given the observation sequence O and the model λ ,

$\gamma_t(i)$: the probability of being in state S_i at time t , given the observation sequence O and the model λ ,

$\xi_t(i,j)$: the probability of being in state S_i at time t and state S_j at time $t+1$, given the observation sequence O and the model λ , and

$\varphi(i)$: the probability of being in state S_i at the end of the observation sequence, given the model λ .

In this section, a hat (^) on top of a variable name indicates the newly estimated variable (according to the modified training method).

The first effect of the newly introduced termination probability φ will apply to the backward procedure. In its original form the probability of being at state S_i after the end

of the observation sequence, $\beta_T(i)$, is initialized to 1 for all states, which assumes that the observation sequence could terminate at each state equally. It would be more logical to use the termination probabilities ϕ as initial values for β_T . Thus, we get:

$$\hat{\beta}_T(i) = \phi(i), \quad 1 \leq i \leq N \quad (3-10)$$

This change is analogous to the initialization of α_t in the forward phase (α_t is initialized to values that consider the initial state probabilities, π).

The remaining of the backward procedure will be affected in the following manner:

$$\hat{\beta}_t(i) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \hat{\beta}_{t+1}(j), \quad t = T-1, T-2, \dots, 1, \quad 1 \leq i \leq N \quad (3-11)$$

The above change should also be reflected on the computation of the overall probability of the model λ given an observation sequence, as follows:

$$\hat{P}(O | \lambda) = \sum_{i=1}^N \alpha_T(i) \hat{\beta}_T(i) = \sum_{i=1}^N \alpha_T(i) \phi(i) \quad (3-12)$$

The computation of the forward variable α is not affected by the introduction of the termination probability. However, γ computation differs in the following way:

$$\hat{\gamma}_t(i) = \frac{\alpha_t(i) \hat{\beta}_t(i)}{\hat{P}_t(O | \lambda)} = \frac{\alpha_t(i) \hat{\beta}_t(i)}{\sum_{j=1}^N \alpha_t(j) \hat{\beta}_t(j)} \quad (3-13)$$

Note that

$$\sum_{i=1}^N \hat{\gamma}_t(i) = \sum_{i=1}^N \frac{\alpha_t(i) \hat{\beta}_t(i)}{\sum_{j=1}^N \alpha_t(j) \hat{\beta}_t(j)} = \frac{1}{\sum_{j=1}^N \alpha_t(j) \hat{\beta}_t(j)} \times \sum_{i=1}^N \alpha_t(i) \hat{\beta}_t(i) = 1 \quad (3-14)$$

Because the newly estimated value of β appears equivalently in both the numerator and denominator of equation 3-14 above, γ_t maintains its probabilistic property.

Similarly, $\zeta_t(i,j)$, the probability of being in state S_i at time t and in state S_j at $t+1$ given the model and the observation sequence is affected as follows:

$$\hat{\zeta}_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\hat{\beta}_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i)a_{ij}b_j(O_{t+1})\hat{\beta}_{t+1}(j)} \quad (3-15)$$

Again, the newly estimated value of β as show in the above equation appears equivalently in the numerator and denominator of the equation, maintaining the probabilistic property ζ .

The termination probability φ can be estimated analogously to the estimation of π as follows:

$$\pi_i = \text{expected frequency is state } S_i \text{ at time } (t=1) = \gamma_1(i) \quad (3-16)$$

$$\varphi_i = \text{expected frequency is state } S_i \text{ at time } (t=T) = \gamma_T(i) \quad (3-17)$$

Note that the probabilistic property of γ at all times t ensures the probabilistic property for φ .

3.3.8. Effect of the termination probability on the testing procedure

The evaluation of HMM response to a particular sequence of observations is augmented by the termination probability. At the end of the observation sequence, the computed forward probability is multiplied by the termination probability of each state. This is

applied implicitly in the computation of γ as shown in Equation 3-18. Substituting T for t in Equation 3-13 above we get:

$$\begin{aligned}\hat{\gamma}_T(i) &= \frac{\alpha_T(i)\hat{\beta}_T(i)}{\sum_{j=1}^N \alpha_T(j)\hat{\beta}_T(j)} \\ &= \frac{\alpha_T(i)\beta_T(i)\varphi(i)}{\sum_{j=1}^N \alpha_T(j)\beta_T(j)\varphi(i)}\end{aligned}\tag{3-18}$$

This increases the sensitivity of each of the evaluation methods mentioned in section 2 using additional information from the training data by employing φ . Note that the original computations of $P(O|\lambda)$ uses α_T instead of γ_T . This is due to the definition of β_T to be one in traditional models, which makes the values of γ_T identical to those of α_T . This is not the case with $\hat{\gamma}_T$.

3.4. Combination of the global and analytical models

The two models mentioned above are combined in a sequential manner. First the global model is applied for the sake of lexicon reduction. The reduced lexicon size is then fed to the analytical model to provide an ordered list of possible sub-word classes. The global model has two major contributions to the system: (1) reduced lexicon increases the speed of the analytical model by reducing the number of models to be evaluated, and (2) reduced lexicon implies less ambiguity fed to the analytical model, which means more reliability. However, a mistake in the global model is severe enough to imply wrong decision by the sub-word recognition system. Therefore, decisions of the global model

should be supported by high confidence. This was achieved by defining a relatively small number of clusters, and by allowing each class to be scattered between clusters.

Chapter 4

Experimental Results

In this chapter, we present various evaluation results of each of the models involved in the recognition system. We start by describing experiments on the global model in section 4.1, followed by the analytical model in section 4.2, and finally we describe experiments on the combined sub-word recognition system in section 4.3.

4.1. *Global model*

As mentioned in Chapter 3, the global model uses a Kohonen network to perform a rough clustering of the data. In a Kohonen network, the number of neurons in the competitive layer represents the number of clusters. Section 4.1.1 describes the experiments and factors that were considered while determining the network structure. Section 4.1.2 shows the performance of the global model using the various data sets.

4.1.1. **Number of competitive neurons**

A number of experiments were conducted to select an optimum number of clusters to be used at the global level. Optimality is defined by two terms:

1. Accuracy of the clustering process, which is measured by the number of correct decisions made by the NN. Feature vectors of the validation set are fed to the NN, which maps each of the input vectors to one cluster. A correct decision about a sample s would map the input feature vector of s to a cluster that contains the correct class of the sample s .

2. Significance of the lexicon reduction process, which is defined as the ratio of reduced lexicon size to the original one. The average lexicon size is determined as the weighted average of number of classes in each cluster.

Theoretically, the minimum number of clusters is one, in which case, the system will make no errors, but makes no reduction in the lexicon. As the number of clusters increases, so does the ambiguity of the grouping process. Thus, the accuracy is expected to decrease as we increase the number of clusters. The maximum number of clusters would be as much as the number of training samples, in which case the system reduces the lexicon to one (which means identification of the input). However, the error rate is expected to be high in this case. Figure 4-1 shows the accuracy rate for NNs versus numbers of competitive neurons. The irregularities in the figure are due to the randomness of both the initialization step and the order of sample presentation during the training phase of each network.

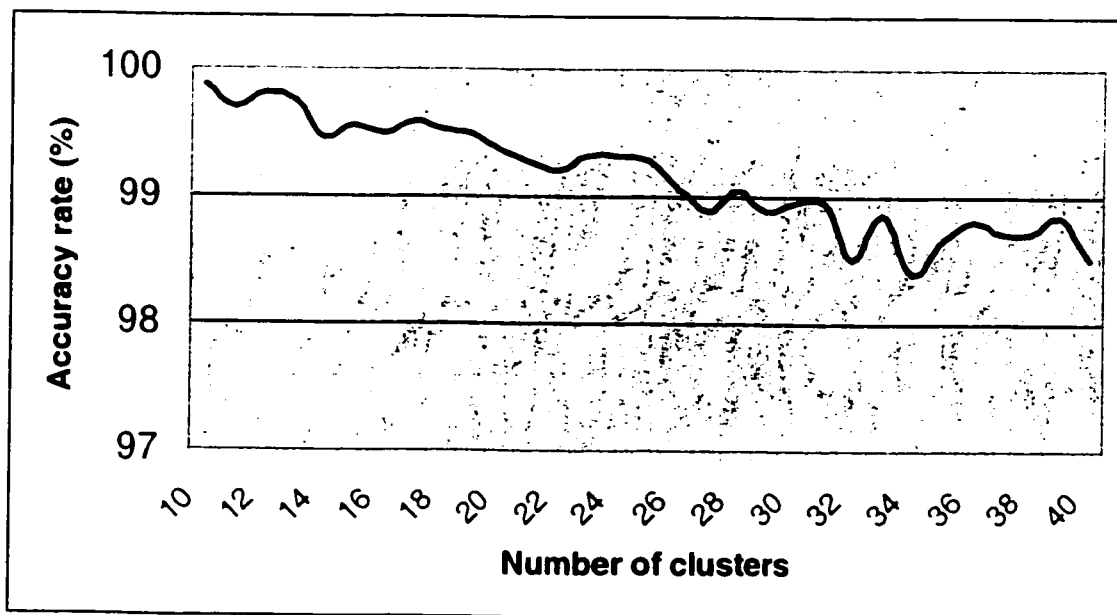


Figure 4-1: Performance of the global model versus the number of clusters

The significance of the lexicon reduction process is zero if we use only one cluster, since such a cluster will contain all classes, making no lexicon reduction at all. On the other hand, the significance would be optimal if the resulting cluster contains only one class. In practice, the more reduction achieved at this level, the easier becomes the identification process at the next level. Figure 4-2 shows the ratio of reduction achieved against the number of clusters used.

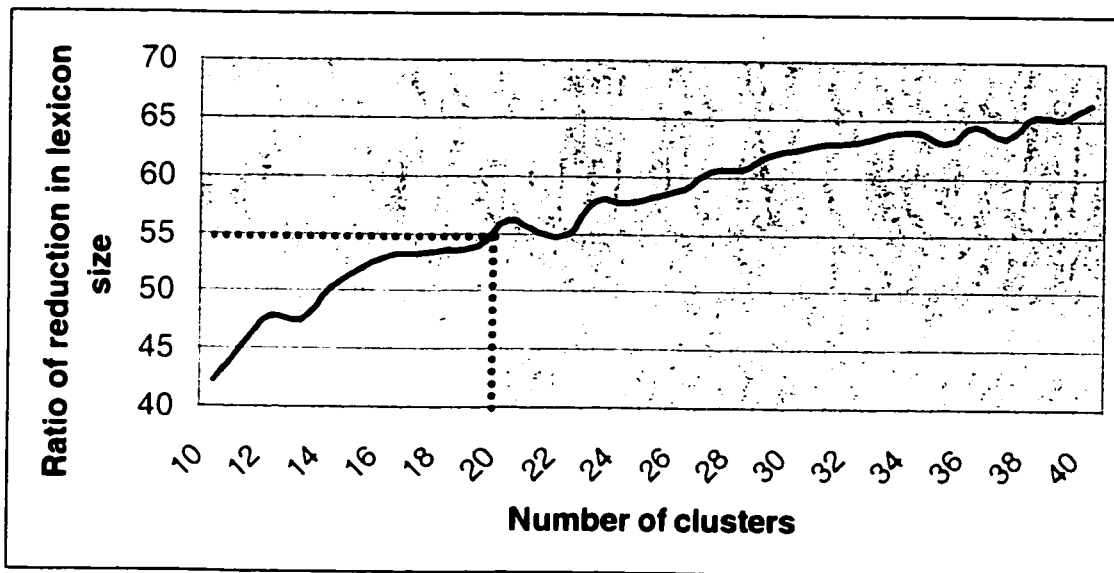


Figure 4-2: Significance of the global model versus the number of clusters

4.1.2. Performance of the global model

Based on the above two factors, and on the experiments mentioned above, we have used 20 competitive neurons and hence the reduction in lexicon size is above 50%. Training is allowed to iterate for a larger number of epochs than that of the experiment described in section 4.1.1. Moreover, training is allowed to start from various initialization points to reduce the effect of local equilibrium points. Table 4-1 shows the performance of the global model on the validation and on the testing sets. The validation

set was used during the training phase for verification purposes only, while the testing set was used to evaluate the final model. Less than 1% of errors were reported on both the validation and testing sets. While this can be referred to the performance of artificial neural networks, it is partially due to the fuzziness allowed in this model. A cluster may contain a number of classes. In average, there are 24 classes per cluster in this model. The network does not provide any ordering of classes within each cluster.

Most errors at the global level are due to severe noise existing in the sub-image. Figure 4-3 shows one of such problematic sub-words. Significant parts of the sub-word shown in Figure 4-3 were eliminated during the binarization step.

Table 4-1: Performance of the global model

Data Set	Accuracy (%)
Training	100
Validation	99.36
Testing	99.04



Figure 4-3: A sample that has been mis-categorized by the NN classifier (Class number 5, code 1-9)

4.2. Analytical model

As mentioned earlier, due to the complexity of the decision space, we have allowed more than one model for classes that have shown considerable variations. During the evaluation process, classification of a sample s from class c is considered correct if it is mapped to any of the models that represent the class c . In sections 4.2.1 to 4.2.3, we describe the effect of various parameters on the analytical model. Section 4.2.4 describes the performance of the analytical model.

4.2.1. Size of the code book

The vector quantization process affects the identity of the observations extracted, and therefore the sensitivity of the classifier to feature variations. A large number of categories implies higher sensitivity to variations, which could cause similar features to be mapped into different categories, making it harder for the classifier to detect their similarity. This would also require more resources as it increases the number of different observations in the system. On the other hand, a small number of categories could map different observations into the same category, making accurate classification impossible.

Table 4-2 shows the performance of the HMM classifier in relation to the size of the codebook used in the quantization process. The results shown in this experiment are

Table 4-2: Performance of the HMM classifier versus the codebook size. This experiment was performed on the validation set, with reestimation of π and using φ

Number of categories	Top 1 (%)	Top 3 (%)	Top 5 (%)	Top 10 (%)
30	62.90	74.55	82.07	92.64
45	65.79	78.48	86.52	94.40
60	66.97	77.71	86.30	94.48
75	67.87	79.15	86.68	94.59
90	69.68	79.84	87.17	94.92
105	69.32	80.56	87.87	94.74
110	67.82	78.28	86.85	94.05
120	70.93	80.40	87.25	94.64
130	68.74	78.62	87.08	94.14
140	68.71	80.47	87.73	94.64
150	70.61	79.35	87.42	94.33
160	71.25	79.68	86.93	94.19
170	72.03	79.84	86.65	94.04
180	70.43	80.07	87.14	94.64
190	69.77	79.67	86.70	94.31
205	71.31	79.61	86.56	94.22
220	70.78	79.18	86.81	94.60
235	71.17	78.21	86.07	93.50
250	70.17	78.46	85.85	93.96
265	69.66	79.32	85.92	93.59
280	69.91	78.01	85.18	93.47

derived from the validation set. The training set results may show improvement while the classifier overfit the training data, which affects the ability to generalize the classifier.

Based on this experiment, we have set the size of the codebook to 170 elements.

4.2.2. Re-estimation of the initial probabilities

Lengths of observation vectors in the analytical model are based on the curvature of the input image (see Chapter 3). Short observation vectors over-weigh small variations, causing more rigid decisions. To avoid this effect, we tuned the linear approximation module to allow longer observation vectors. However, this did not help much with short sub-words that could carry no curvature at all. In such cases the observation vector may contain only one observation, making the HMM decision based on the initial probability vector (π) and on the emission probabilities B . Without re-estimation of π , decision on such cases will be solely based on the emission probabilities B , which makes the decision more vulnerable to errors. This led us to allow the initial probability to be re-estimated. Table 4-3 shows the effect of allowing the re-estimation of π on the performance of the HMM classifier.

Table 4-3: Effect of the re-estimation of π on the performance of the HMM classifier. This experiment was performed on the training set using φ

	Top 1 (%)	Top 3 (%)	Top 5 (%)	Top 10 (%)
Fixed π	62.41	76.06	84.84	94.42
Reestimated π	81.13	87.46	93.26	98.68

4.2.3. Termination probabilities

Partial similarity among different classes was the source of some errors. Smaller sub-words could be mapped to models of longer sub-words if they share similar initial parts of their shapes. The termination probability was introduced to reduce such chances. Table 4-4 shows the effect of the introduction of the termination probability on the performance of HMM.

Table 4-4: Effect of the introduction of φ on the performance of the HMM classifier.
This experiment was performed on the training set

	Top 1 (%)	Top 3 (%)	Top 5 (%)	Top 10 (%)
Without φ	70.18	79.78	87.67	96.17
With φ	81.13	87.46	93.26	98.68

4.2.4. Performance of the analytical model

The overall performance of the analytical model is shown in Table 4-5. Images of classes under consideration are shown in Table 4-6. Detailed recognition performance (on the training data, first choice of the classifier) and confidence for each class are shown in Table 4-7. The confidence of a given class c is defined as the ratio of correct appearance of class c as the first choice to the total number of times the class c has appeared as the first choice by the classifier.

Table 4-5: Performance of the analytical model

Data Set	Top 1 (%)	Top 3 (%)	Top 5 (%)	Top 10 (%)
Training	81.13	87.46	93.26	98.68
Validation	72.03	79.84	86.65	94.05
Testing	69.85	78.60	85.89	93.28

Table 4-6: Sub-word images covered by the recognition system

Class No.	Image	Class No.	Image	Class No.	Image	Class No.	Image	Class No.	Image
1	ا	14	فد	27	ثبا نبا	40	تتين	54	هتتا
2	ت	15	ست	28	ثين	41	بعبا	55	فلا
3	د	16	لا	29	فيس	42	بعين	56	بععة
4	ر	17	لظا	30	سقة	43	تسعة	57	صصا
5	قا	18	را	31	ستو	44	تسعو	58	تصين
6	ك	19	يا	32	سعو	46	فيسا	59	فصبا
7	ن	20	ينا	33	عشر	47	فصا	60	فصين
8	ت	21	تتا	34	غبير	48	فيسو	61	صصبا
9	و	22	تتبا ثتبا	35	فقط	49	سبعة	62	صصين
10	يا	23	بععة	36	لظا	50	سعو	63	صصعة
11	تتبا ثتبا	24	بغو	37	لغي	51	ستا	64	هتتبن
12	ج	25	تصم	38	هتعة	52	ستين	65	تصعة
13	تتبا ثتبا	26	تظا	39	نبا	53	لصين	66	تصعة
								67	صصعة

4.2.5. Error analysis

Throughout the evaluation process, we encountered various sources of errors in the analytical model. Following is a chronologically sorted list of these sources, with some examples.

1. Difficult handwriting or severe noise on the cheque document (Figure 4-4). We estimate that this type of error caused 20% of all errors encountered in the validation set. This type of error is the most difficult to overcome, because it affects the source of the image, which makes correct classification merely remotely visible. As a result, the correct class is mostly far away from the top

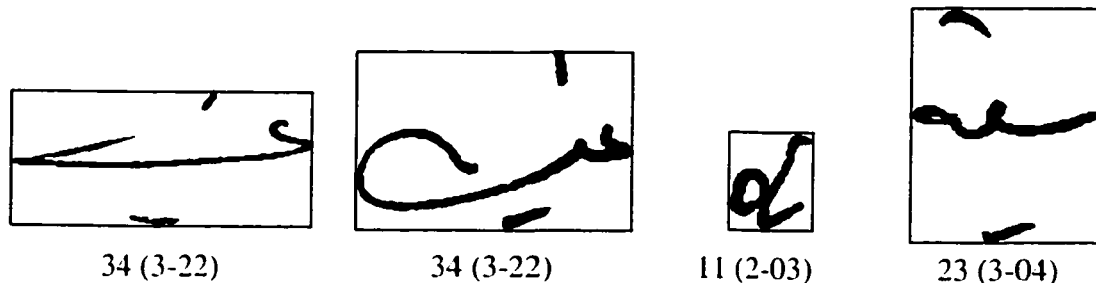


Figure 4-4: Samples of errors caused by difficult handwriting. The correct class number (code) is shown below each image

choice of the classifier. On average, the correct class is estimated to be the top 11th choice of the classifier in such cases. Rejection control could be used to detect and avoid this type of error.

2. The extraction and/or binarization steps (Figure 4-5). Extraction of the legal amount from the cheque image could leave out significant parts of some sub-words. The same could result from the binarization process. In addition, the binarization step could cut a sub-word image into two or more pieces. These errors represent about 20% of all errors in the validation set. The effect on the classifier depends on the significance of the damage caused to the sub-word. On the average, the classifier puts images affected by this type of error at the top 10th choice.

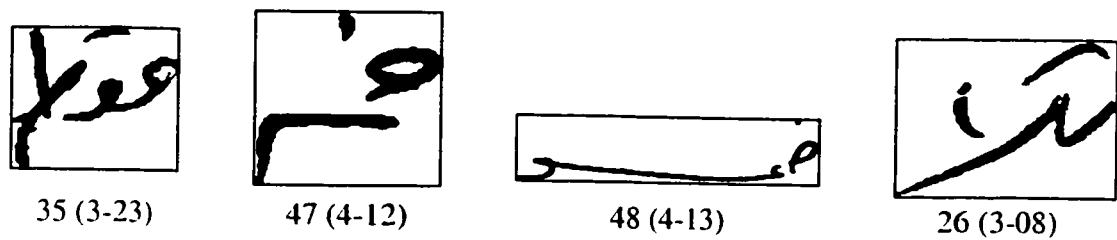


Figure 4-5: Samples of errors caused by the extraction and binarization module. The correct class number (code) is shown below each image

3. Imperfect skeletonization (Figure 4-6). As perfect skeletonization is not visible, imperfect skeletons could affect the sequence of strokes used to characterize the input image. This type represents around 8% of the total errors on the validation set. In average, the classifier puts images affected by this type of error at the top 9th choice.

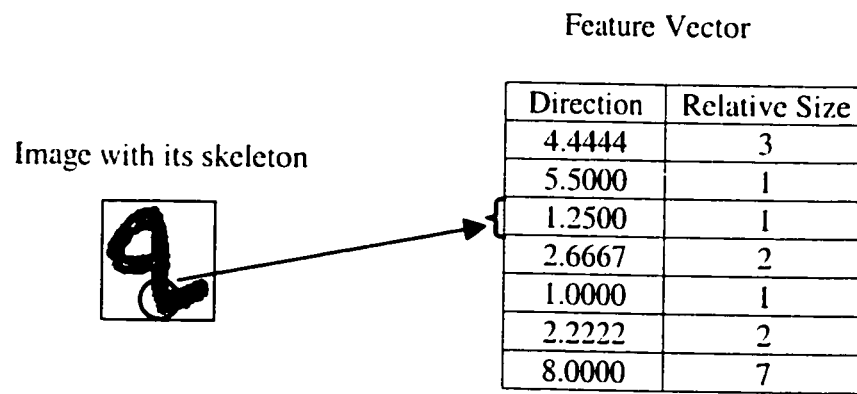


Figure 4-6: An error caused by the skeletonization module

4. The feature extraction module (Figure 4-7). This type of error is not very frequent, and is mainly due to errors in selecting the starting point. Interchangeable feature points (end points & junction points) could be traced in the wrong direction, creating a reversed portion in the feature vector. This error

represents about 4% of the total errors on the validation set. On the average, the classifier puts images affected by this type of error at the top 12th choice.

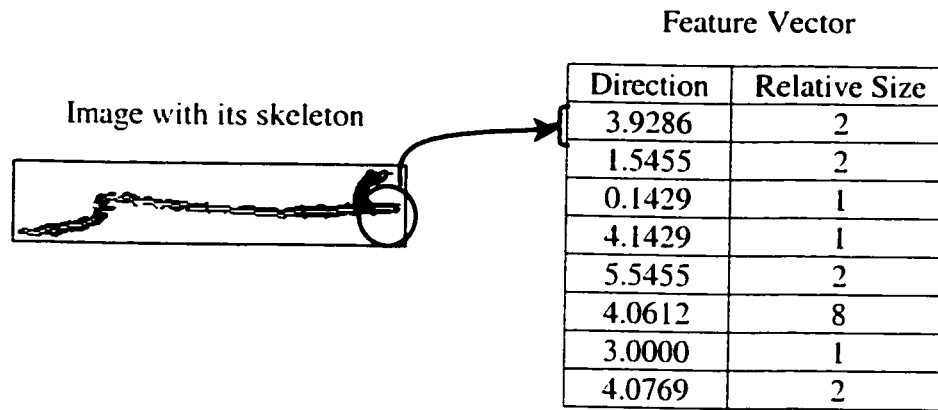


Figure 4-7: An error caused by the Feature Extraction module

5. The linear approximation process (Figure 4-8). This process is used to map the sequence of pixels into line segments. Due to the static thresholds used for this purpose, some significant shape variations could be overlooked causing a loss of important characteristic. On the other hand, some minor variations could be over emphasized causing unnecessary change in the feature vector. This error represents about 24% of the total errors on the validation set. On the average, the classifier puts images affected by this type of error at the top 6th choice.
6. The classification module (Figure 4-9). This type of error could be a result of the small size of training data of particular classes, large similarity between classes or large variations within the affected classes. This error represents about 24% of the total errors on the validation set. On the average, the classifier puts images affected by this type of error at the top 3rd choice.

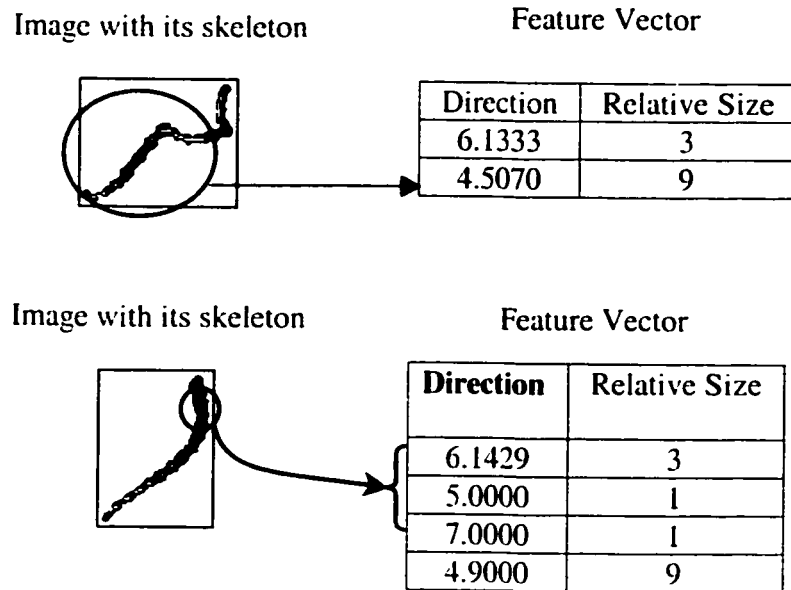


Figure 4-8: Samples of errors caused by the approximation procedure

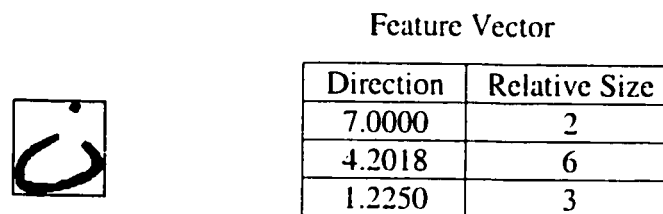


Figure 4-9: An error caused by the classification module.

4.3. Combined Classifiers

As noticed from sections 4.1 and 4.2 above, the global model provides more accurate but less precise results than the analytical model. The error involved in the global model does not exceed 1%. On the other hand, the average number of classes per cluster is 24. Moreover, the model provides no ordering between classes of a given cluster.

The sequential combination method improved the recognition rate of the analytical model by 3.68%. Detail results of the combined classifiers are shown in Table

4-8. While the above results are not ideal for a practical industrial system, they highlight the strengths of both models in solving this problem, and expose them to further improvements.

The results shown in this chapter reflect the complexity of the recognition problem under consideration, and the difficulty of the samples used in this study. Appendix D shows some samples from the training data sets used in this study. Table 4-9 shows the results of the current study as compared to other studies found in the literature. The table shows impressive results from [MC98] and [KG97] considering the lexicon sizes involved in their studies. However, it is important to consider the type of data (domain and quality) used to test each of the presented systems. While the domain of each study is normally stated, the quality of the data is not always reported in detail.

Table 4-7: Performance and confidence of each class on the training data

Class number	Number of samples	Perf. (%)
1	1226	97.80
2	12	75.83
3	17	97.06
4	1358	87.30
5	369	77.39
6	775	87.23
7	622	77.66
8	158	88.10
9	1248	79.97
10	27	77.78
11	624	83.64
12	5	100.00
13	108	83.33
14	95	77.05
15	2	70.00
16	1139	65.74
17	487	67.27
18	139	72.52
19	813	89.54
20	9	100.00
21	70	83.57
22	13	100.00
23	103	74.27
24	54	82.22
25	5	100.00
26	237	70.88
27	251	91.79
28	5	100.00
29	8	100.00
30	74	73.78
31	40	70.00
32	16	87.50
33	309	85.04
34	699	72.30

Class number	Number of samples	Perf. (%)
35	623	80.38
36	95	88.95
37	2	100.00
38	9	100.00
39	60	69.33
40	22	76.37
41	55	66.18
42	5	96.00
43	45	64.00
44	28	74.28
45	1	100.00
46	1	100.00
47	183	71.92
48	102	63.73
49	55	71.27
50	45	59.11
51	59	67.12
52	4	100.00
53	37	94.06
54	22	70.91
55	4	90.00
56	1	100.00
57	59	85.43
58	1	100.00
59	119	81.93
60	8	60.00
61	51	76.86
62	1	100.00
63	1	100.00
64	3	100.00
65	1	100.00
66	1	100.00
67	1	100.00
Average		81.13

Table 4-8: Performance of the combined recognition system

Data set	Top 1 (%)	Top 3 (%)	Top 5 (%)	Top 10 (%)
Training	85.74	90.69	95.52	99.87
Validation	76.36	82.74	89.33	95.37
Testing	73.53	81.50	88.19	94.36

Table 4-9: Recognition results compared with other systems in the literature

Reference	Lexicon size	Top 1 (%)	Top 10 (%)	Top 20 (%)
Current study	67	73.53	94.36	97.70
[DF01]	198	65.05	90.83	95.00
[MC98]	232	81.60	94.90	
[GS98]	30	86.70	99.90	
[KG97]	100	84.60		99.00
[Am00b]	28	89.65		
[WF01]		80.9		

Chapter 5

Legal amount processing

Processing legal amounts is an important step to achieve automated cheque processing systems. Among the attempts to develop a complete legal amount processing system we cite Kaufmann and Bunke [KB00a], who developed a complete system to read legal amounts extracted from German cheques. Statistical features are extracted from each component and are fed to an HMM classifier. Contextual information is used to post process the classification results. The correct legal amount was among the top 10 choices in 88.8% of the cheques.

In this chapter, we describe how the previously built grammar and sub-word recognition system could be employed to interpret the legal amount into a numerical value. A detailed example is shown for the various steps in the interpretation process. Further enhancements of the interpretation method are required to make it practical for real world applications.

5.1. Pre-processing

The input is assumed to be a gray level digitized image of a cheque. Figure 5-1 shows a sample cheque and all the pre-processing steps applied to it. First, the legal amount is statically segmented from the cheque form. Dynamic thresholding is then applied to binarize the extracted legal amount. Basic filling and thinning operations are applied next to enhance the image of the legal amount.

Baseline is an important factor that affects object orientation and noise removal decisions. We determine the location of the baseline and its thickness using horizontal

٠٠١٩٩٠

شركة الراجحي المصرفية للاستثمار
AL RAJHI BANKING & INVESTMENT CORP.

Date ١٤١٧/١١/١٤ التاريخ

SULTANAH BRANCH فرع سلطانة الرياض

RIYADH

Against this Cheque
Pay to the Order of

الذخيرة بموجب هذا الشيك لأمر

The amount of

المبلغ فقط عشرة آلاف ريال لا يد

١٠٠٠٠ /

رفع

توقيع

فقط عشرة آلاف ريال لا يد

فقط عشرة آلاف ريال لا يد

Figure 5-1: Segmentation and binarization of the legal amount

projections. Slant correction is then applied by computing the density of the baseline in various angles. The minimum slant of the image will produce the maximum baseline density.

It is very common that undesirable objects occur in the extracted legal amount. Such objects include external noise and intrusions (introduced by the cheque writer or bank officers), portions of the upper (or lower) handwritten lines, or portions of the pre-printed text. It is important to remove such objects as they may affect the recognition results. Due to the limited lexicon entries involved in the current application, secondary components (e.g. dots) are not critical in sub-word identification. Noise and secondary components are removed based on the following factors:

- 1) Inter-component distances: small horizontal gaps are normally used to separate handwritten words. Larger gaps are indications of noise or unrelated objects. Dynamic threshold is used to define the term “small gap”.
- 2) Vertical position of components: relevant components are normally written near the baseline. Having an object above or below the baseline with significant distance is an indication of unrelated components. Dynamic threshold is used to define the term “significant distance”.
- 3) Size of each component: very small or very large objects are clear signs of noisy components. Extreme aspect ratio is another indicator that is used to detect undesirable objects.
- 4) Slant: slant of each component should be close enough to the overall slant of the legal amount.

Different writers have different habits in terms of inter-component distances and letter sizes. Dynamic thresholds are determined based on the mean and standard deviation of the overall components in the legal amount. Location of an object is described in terms of the bounding box and its center of gravity.












Each connected component is then extracted as a sub-word. Slant correction is applied to each object to correct small slant variations between sub-words. Table 5-1 shows segmented sub-words from the binarized legal amount in Figure 5-1. Each sub-word image is assigned a unique position number (index).

5.2. Legal amount interpretation

Given an ordered list of sub-word images, the sub-word recognizer is invoked to provide a list of candidate sub-word codes for each sub-word image. Table 5-2 shows the top ten

choices for each sub-word in Table 5-1. Legal amount (LA) interpretation intends to translate proper sequences of sub-word codes into their equivalent numerical values. LA interpretation is achieved at two levels: local and global. At the local level, we aim to detect possible word from successive sub-word combinations. A word is encountered in a given position p if all its composing sub-words are found in a sequence that ends at p . This is easily performed using a dictionary lookup procedure. Due to the fact that some words share sub-words, the sequence of words may not be unique (e.g. at a given position p , sub-word w_1 could stand alone as a complete word, or could be concatenated with the preceding sub-word at $p-1$ to form another word w_2). In such cases, all possible words are passed to the global level. Table 5-3 shows the list of constructed words at each position of the legal amount (deduced from Table 5-2). For each word, starting and ending positions, as well as the probability, are passed to the global interpretation level to determine the best combination of words.

Table 5-1: The sequence of sub-words segmented from Figure 5-1

11	10	9	8	7	6	5	4	3	2	1	Position ←
											Segmented sub-word

The search complexity in the first level is fairly limited due to the fact that the maximum number of consecutive sub-words that may compose a single word is found to be 4 (for the current vocabulary). This search procedure would then be performed $4kn$, where k is a constant defined in relation to the dictionary size, and n is the number of sub-words in the extracted legal amount. Thus the complexity of this level is of $O(n)$.

The global level tries to find a complete and correct legal amount out of the list of all possible words. A correct legal amount should satisfy two conditions:

- 1) It should make use of each sub-word exactly once. That is not to ignore any sub-word in the legal amount, and not to use any sub-word more than once.
- 2) A correct legal amount should follow the grammatical rules of the Arabic language.

Table 5-2: Results from the sub-word classifier – Top 10 choices for each sub-word in Table 5-1 (The correct choices are shaded)

11	10	9	8	7	6	5	4	3	2	1	Position ←
لا	و	لا	ف	ر	ي	شا	ل	ة	عشر	فقط	1
ن	لا	ل	يا	ت	ل	لف	ا	نية	ف	بما	2
عق	ف	ن	عنا	لا	ن	عق	لا	عق	لا	لف	3
غير	غير	ي	عق	و	ف	لغا	و	لغا	فسو	لغا	4
خلا	ن	عق	لا	ة	عق	ن	لغا	ف	فئة	بما	5
ك	عق	غير	و	غير	غير	ل	غير	فئة	خلا	لا	6
ف	ل	يا	ر	ن	عنا	لا	يا	ل	لف	ن	7
يا	ر	لف	غير	لغا	يا	فقط	ر	تسع	عق	فئة	8
ما	عشر	بعا	ا	ك	ثو	غير	ن	عشر	و	بما	9
ر	ما	خلا	ن	عق	لف	ك	ت	و	غير	شا	10

These two conditions form the evaluation criteria of the global level. Details of the grammar rules have already been described in Chapter 2.

To simplify the application of grammatical rules, all words that do not change the numerical value of LA (e.g. the word “only”) are removed from the word list. If such

words appear in the first position of the legal amount, LA is considered correct even if it ignores the first position. The same can be said about the last position.

Depth first search is applied next to find all correct syntactic trees that satisfy the above two conditions. Numerical values for each complete tree is computed based on predefined correspondence between grammatical terms and mathematical operations ('+' and '*' operations). Table 5-4 shows the induced numerical values with corresponding probabilities. Only the top 10 choices are shown out of 34 produced values. Figure 5-2 provides parsing trees that were used to achieve the top five choices in Table 5-4.

5.3. Discussion and Conclusion

The goal of this chapter is to show how to interpret Arabic legal amounts, and not to develop this part of the system. As a result, concept validation and implementation efficiency considerations were avoided at this level. However, these two points constitute a very important part of the future work.

Table 5-4: Top 10 numerical results with their probabilities
(The correct one is the top choice)

Value	Probability (x 1.0e-06)
10000	0.3453
18	0.3013
2010	0.0006
3000	0.0003
11	0.0003
1010	0.0001
2005	0.0000
15000	0.0000
23	0.0000
1005	0.0000

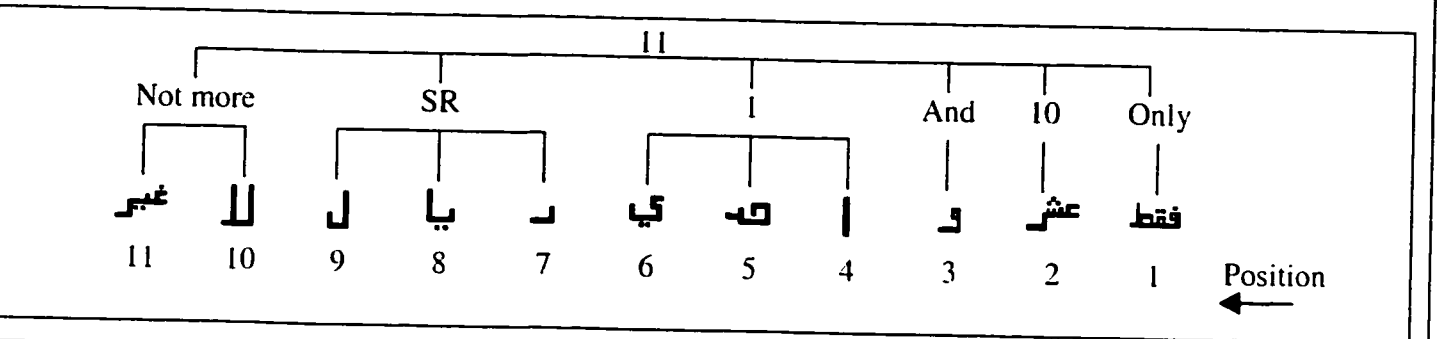
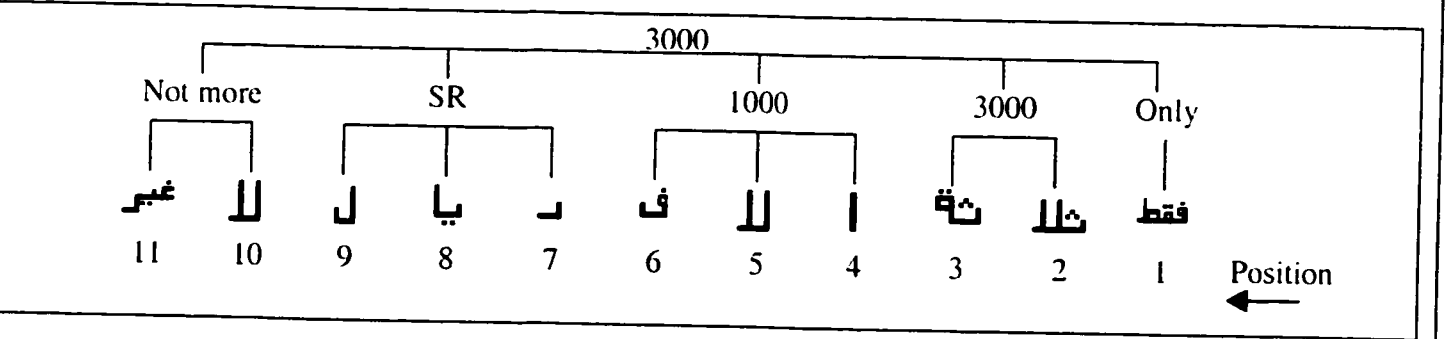
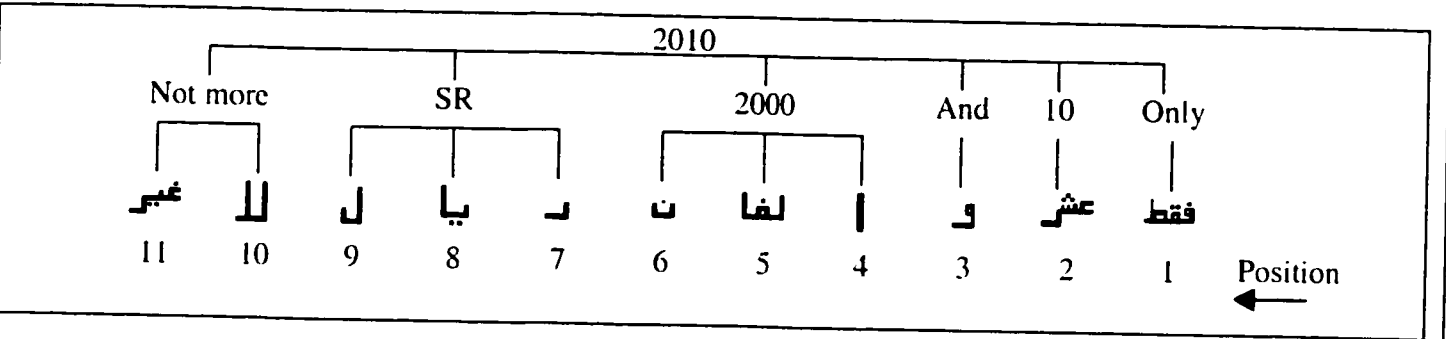
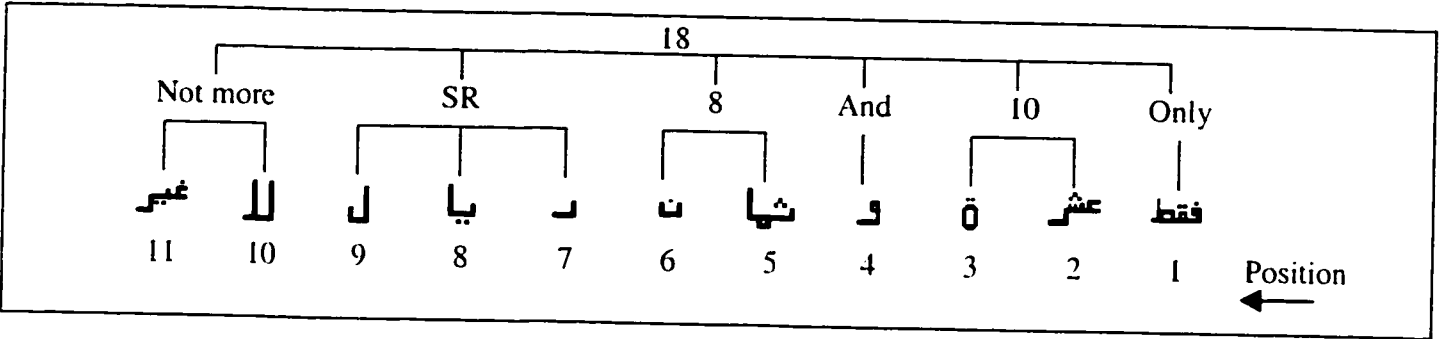
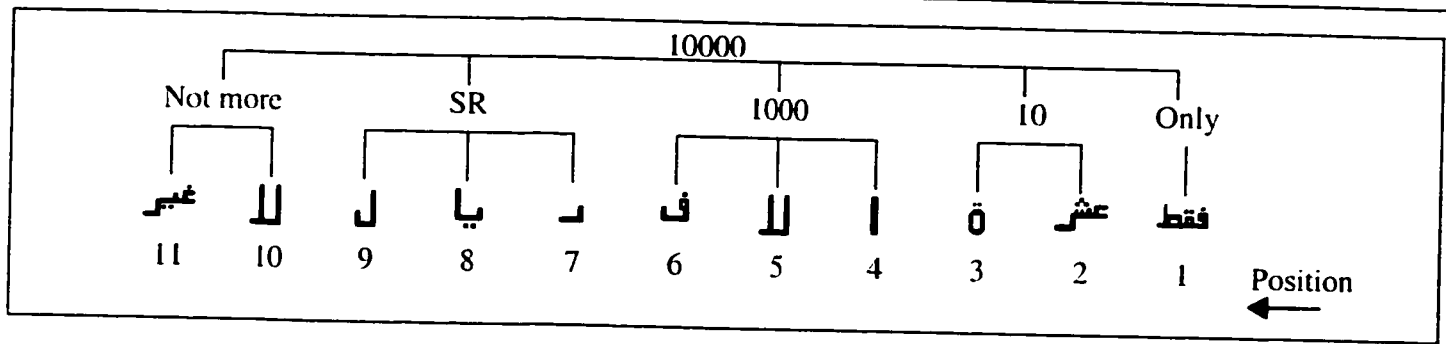


Figure 5-2: Grouping of candidate sub-words to produce the top five possible values in Table 5-4

Chapter 6

Conclusion

This work has produced a significant infrastructure for research in Arabic Cheque Processing by providing Training and Testing sets that can be used as a consistent evaluation medium among various recognition methods. A parsing and translating grammar for Arabic legal amounts is also included to verify and generate a numerical value out of the list of words. The database has the potential to be standardized among researchers in this field because it came from real-world source and because it provides sufficiently large training and testing data.

Pen-trajectory has long been used for various objectives. Here, we propose a more efficient method for the estimation of pen-trajectory from off-line images. We have applied this method to sub-word images from Arabic legal amounts. However, the method is applicable as well to similar types of images. We also propose to use the pen-trajectory to feed HMM with meaningful sequence of observations that resemble the original occurrence of events.

In this work, we have developed a recognition system to recognize Arabic sub-words based on global and analytical models. The global model is used to reduce the lexicon based on global features, while the analytical model is used to produce a sorted list of classes based on the pen-trajectory information. We have used the above mentioned databases to train and test our system. The sub-word recognition rate recorded was 73.53% on the top 1st choice and 94.36% on the top 10 choices of the testing data set. It is to be noted that the database used to test this system came from real-life source, which includes a greater difficulty in terms of style variations and noise.

In the remaining of this chapter we highlight the major contributions of this work and list our future perspectives.

6.1. Summary of contributions

The first major contribution of this thesis is the development of a set of databases to assist researchers in the field of Arabic cheque processing. This set is unique of its kind as it has been created using real-world bank cheques. It is also uniquely characterized as it is the only database (to the best of our knowledge) for Arabic handwritten cheques. The validation algorithm introduced in this work is another unique characteristic of this work. In the proposed validation algorithm, redundant information is employed to ensure the accuracy of the tagging operation. In addition, this work introduced a parser for Arabic legal amounts. This parser can be used to validate and translate legal amounts into numerical values. Such translation is essential to the validation algorithm mentioned above and to the processing of complete legal amounts as well.

Another important contribution of this work is the introduction of a new method to estimate the pen-trajectory from handwritten data. Such estimation is an efficient way to transform 2-D patterns to a 1-D sequence of observations. The input image is skeletonized and then efficiently mapped to a graph representation. Rather than transforming the resulting graph to an Eulerian or Hamiltonian graph, the proposed method applies a simpler transformation into a tree. It is noticeable that the application of this method can be extended to various types of problems where 1-D sequences need to be extracted from 2-D images or the pen trajectory has to be estimated.

Last but not least, the discriminatory power of the HMM classifier is improved by the introduction of a new parameter. The new parameter, the termination probability,

makes use of an additional piece of information to estimate the most probable state in the model to terminate the observation process. It is to be noted that the parameter does not require extra inputs to the system. Rather, the system uses extra information that is normally created during the training phase. Such information is not used in traditional HMM. The new parameter has improved recognition results by about 10% on the top 1st choice.

6.2. Future perspectives

This work has instantiated a solid basis for the research in Arabic cheque processing. However, the research work in this topic is far from over. Future work will concentrate on the following items:

1. System improvements

- In our system, we used a simple linear approximation to map the sequence of pixels into a list of strokes. We see a chance of improvement to the recognition system in this direction.
- In our system, we used a code-book quantization scheme. Using a more robust quantization method could lead to improved results.
- Improving the sub-word classification rate using multiple classifiers.

2. System integration

- Implement the complete legal amount recognition system given an improved sub-word classification rate.
- Define and implement a criterion to combine the results of the legal and courtesy amount recognition systems.

- Develop a complete system to extract and process both the legal and the courtesy amounts.
3. Public research society
 - Get the databases approved by all parties and ready for publications.
 - Find ways to increase the training and testing data.
 4. Industry
 - Collaborate with the industry to perform on-line test of the complete system.

References

- [Ab98] I. Abouhaibah, Recognition of off-line cursive handwriting, *Computer Vision and Image Understanding*, 71(1), 1998, pp. 19-38.
- [AC00] Y. Alohalı, M. Cheriet and C. Y. Suen, Databases for Recognition of Handwritten Arabic Cheques, *Proc. 7th IWFHR*, Sept. 2000, Amsterdam, the Netherlands, pp. 601-606
- [AC02-a] Y. Alohalı, M. Cheriet and C. Y. Suen, Databases for Recognition of Handwritten Arabic Cheques, *Pattern Recognition*, (Accepted).
- [AC02-b] Y. Alohalı, M. Cheriet and C. Y. Suen, Efficient Estimation of Pen Trajectory from Off-Line Handwritten Words, *ICPR*, Quebec 2002, (Accepted).
- [AC02-c] Y. Alohalı, M. Cheriet and C. Y. Suen, Introducing Termination Probabilities to HMM, *ICPR*, Quebec 2002, (Accepted).
- [AC02-d] Y. Alohalı, M. Cheriet and C. Y. Suen, Dynamic Observations and dynamic state Termination For Off-line Handwritten Word recognition using HMM, *IWFHR*, 2002, (Accepted).
- [AE99] K. Aas, L. Eikvil and R. Huseby, Applications of hidden Markov chains in image analysis, *Pattern recognition*, 32, 1999, pp. 703-713.
- [AH98] B. Al-Badr and R. Haralick, A segmentation-free approach to text recognition with application to Arabic text, *IJDAR*, 1, 1998, pp. 147-166.
- [AK01] R. Azmi and E. Kabir, A new segmentation technique for omnifont Farsi text, *Pattern Recognition Letters*, 22, 2001, pp. 97-104.

- [Alt95] M. Altuwaijri, *A Parallel Recognition System for Arabic Cursive Words with Neural Learning Capabilities*, Ph.D. Thesis, University of Southwestern Louisiana, 1995.
- [All95] J. Allen, *Natural Language Understanding*, The Benjamin/Cummings Publishing Company, Inc., 1995.
- [AM94] I. Abouhaibah, S. Mahmoud and R. Green, recognition of handwritten cursive Arabic characters, *IEEE Trans. on PAMI*, 16(6), 1994, pp. 664-672.
- [AM95] B. Al-Badr and S. Mahmoud, Survey and bibliography of Arabic Optical text recognition, *Signal Processing*, 41, pp. 49-77, 1995.
- [Am98] Adnan Amin, Off-line Arabic Character Recognition: The State of the Art, *Pattern Recognition*, 31 (5), pp. 517-529, 1998.
- [Am00a] A. Amin, Recognition of Printed Arabic Text Based on Global Features and Decision Tree Learning Techniques, *Pattern Recognition*, 33, 2000, pp. 1309-1323.
- [Am00b] A. Amin, Prototyping structural description using an inductive learning program *International Journal of Intelligent Systems*, vol. 15, no. 12, pp. 1103-1123, Dec 2000.
- [AN97] S. Alshebeili, A. Nabawi, and S. Mohmoud, Arabic character recognition using 1-D slices of the character spectrum, *Signal Processing*, 56, 1997, pp. 59-75.
- [AU90] H. Al-Yousefi and S. Upda, Recognition of Handwritten Arabic Characters via Segmentation, *Arab Gulf Journal for Scientific Research*, 8, pp. 49-59, 1990.

- [AV00] N. Arica and Y. Vural, One-dimensional representation of two-dimensional information for HMM based handwriting recognition, *Pattern Recognition Letters*, 21, 2000, pp. 583-592.
- [AY97] A. Aticic and F. Yarman-Vural, A heuristic algorithm for optical character recognition of Arabic script, *Signal Processing*, 62, 1997, pp. 87-99.
- [Bo99] F. Bouslama, Structural and fuzzy techniques in the recognition of online Arabic characters, *International Journal of Pattern Recognition and Artificial Intelligenc (IJPRAI)*, 13(7), 1999, pp. 1027-1040.
- [BS99] I. Bazzi, R. Schwartz and J. Makhoul, An omnifont open-vocabulary OCR system for English and Arabic, *IEEE Trans. on PAMI*, 21(6), 1999, pp. 495-504.
- [CB99] A. Cheung, M. Bennamoun, N. Bergmann, An Arabic optical character recognition system using recognition-based segmentation, *Pattern Recognition*, 34, 2001, pp. 215-233.
- [CK94] M. Chen, A. Kundu and J. Zhou, Off-line handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network, *IEEE Trans. on PAMI*, 16(5), 1994, pp. 481-496.
- [CK98] Y. Chim, A. Kassim and Y. Ibrahim, Dual Classifier system for handprinted alphanumeric character recognition, *Pattern Analysis & Applications*, 1998, 1, 155-162.
- [CL98] M. Cote, E. Lecolinet, M. Cheriet, C. Y. Suen, Automatic reading of cursive scripts using a reading model and perceptual concepts, *IJDAR*, 1, pp. 3-17, 1998.

- [DF98] G. Dzuba, A. Filatov, D. Gershuny and I. Kil, Handwritten Word Recognition-The approach proved by Practice, *Proc. International Workshop of Frontiers in Handwriting Recognition*, Taejon, Korea, pp. 99-111, August 1998.
- [DF01] M. Dehghan, K. Faez, M. Ahmadi and M. Shridhar, Handwritten Farsi (Arabic) word recognition: a holistic approach using discrete HMM, *Pattern Recognition*, 34, pp. 1057-1065, May 2001.
- [EG99] A. El-Yacoubi, M. Gilloux, R. Sabourin and C. Suen, An HMM-based approach for off-line unconstrained handwritten word modeling and recognition, *IEEE Trans. on PAMI*, 21, 8, 1999, pp. 752-760.
- [FS92] J. Freeman and D. Skapura, *Neural Networks Algorithms, Applications and Programming Techniques*, Addison Wesley, New York, 1992.
- [GA99] N. Gorski, V. Animismov, E. Augustin, O. Baret, D. Price and J. Simon, A2iA check reader: a family of bank check recognition system, *Proc. of the 5th Int. Conf. on Document Analysis and Recognition*, Bangalore, India, 1999.
- [Gi85] A. Gibbons, *Algorithmic Graph Theory*, Cambridge University Press, 1985
- [GL93] M. Gilloux and M. Leroux, Recognition of Cursive Script amounts on Postal Cheques, *European Conf. Dedicated to Postal Technologies*, Nantes, France, pp. 705-712, June 1993.
- [GM93] Michel Gilloux and Manuel Leroux, Recognition of Cursive Script amounts on Postal Cheques, *European Conf. Dedicated to Postal Technologies*, Nantes, France, pp. 705-712, June 1993.
- [GS98] D. Guillevic and C. Y. Suen, Recognition of Legal Amounts on Bank Cheques, *Pattern Analysis and Applic.*, 1, pp. 28-41, 1998.

- [Gu96] D. Guillevic, *Unconstrained Handwriting Recognition Applied to the Processing of Bank Cheques*, PhD. Thesis, Concordia University, Montreal, Quebec, Canada, 1995.
- [GW98] M. Garris, C. Wilson and J. Blue, Neural network-based systems for handprinted OCR applications, *IEEE Trans. On Image Processing*, 7(8), 1998, pp. 1097-1112.
- [HP98] L. Heutte, T. Paquet, J. Moreau, Y. Lecourtier and C. Olivier, A structural/statistical feature based vector for handwritten character recognition, *Pattern Recognition Letters*, 19, 1998, pp. 629-641.
- [HS97] K. Han and I. Sethi, An off-line cursive handwritten word recognition systems and its application to legal amount interpretation, *Inter. Jour. of Pattern Recognition and Artificial Intelligence*, 11(5), 1997, pp. 757-550.
- [Hu94] J. Hull, A Database for Handwritten Text Recognition Research, *IEEE Trans. PAMI*, 16(5), 1994, pp. 550-554.
- [KA98] S. Knerr and E. Augustin, A Neural Network-Hidden Markov Model Hybrid for Cursive Word Recognition, *Proc. 14th International Conference on Pattern Recognition*, Brisbane, Australia, pp. 1518-1520, August, 1998.
- [KB00a] G. Kaufmann and H. Bunke, Automatic Reading of Cheque Amounts, *Pattern Analysis & Applications*, 2000, 3, pp. 132-141.
- [KB00b] G. Kaufmann and H. Bunke, *IJDAR*, 2000, 2, pp. 211-221.

- [KB98] G. Kaufmann and H. Bunke, Amount Transilation and Error Localization in Check Processing Using Syntax-Directed Translation, *Proc. 14th International Conference on Pattern Recognition*, Brisbane, Australia, pp. 1530-1534, August 1998.
- [KF00] E. Kavallieratou, N. Fakotakis and G. Kokkinakiss, A slant removal algorithm, *Pattern Recognition*, *Pattern Recognition*, 33, 2000, pp. 1261-1262.
- [KG97] G. Kim and V. Goveindaraju,, A lexicon driven approach to handwritten word recognition for real time applications, *IEEE Trans. PAMI*, 19(4), 1997, pp.366-379.
- [KH98] A. Kundu, Y. He and M. Chen, Alternatives to variable duration HMM in handwritten recognition, *IEEE Trans. on PAMI*, 20, 11, 1998, pp. 1275-1280.
- [KK97] H. Kim, S. Kim, K. Kim and J. Lee, An HMM-Based Character Recognition Network Using Level Building, *Pattern Recognition*, 30 (3), 1997, pp. 491-502.
- [KY00] Y. Kato and M. Yasuhara, Recovery of Drawing Order from Single-Stroke Handwritten Images, *IEEE Trans. on PAMI*, Vol. 22, Num. 9, Sept. 2000, pp. 938-949.
- [LL97] H. Lee and H. Liu, Structural stroke extraction from off-line handwritten Chinese characters, *Progress in handwriting recognition*, editors A. Downton and S. Impedovo, *World Scientific*, 1997.
- [LLG97] M. Leroux, E. Lethelier, M. Giloux and B. Lemarie, Automatic Reading of Handwritten Amounts on French Checks, *Inter. Jour. Of Pattern Recognition and Artificial Intelligence*, 11(4), 1997, pp. 619-638.

- [LM00] B. Lazzerini and F. Marcelloni, A linguistic fuzzy recognizer of off-line handwritten characters, *Pattern Recognition Letters*, 21, 2000, pp.319-327.
- [Ma94] S. Mahmoud, Arabic character recognition using Fourier descriptors and character contour encoding, *Pattern Recognition*, 27, 6, 1994, pp. 815-824.
- [MC98] H. Miled, M. Cheriet, and C. Olivier, Multi-level Arabic Handwritten Words Recognition, *Proc. Advances in Pattern Recognition*, Sydney, Australia, pp. 944-951, August 1998.
- [MG96] M. Mohamed and P. Gader, Handwritten Word Recognition Using Segmentation-Free Hidden Markov Modeling and Segmentation-Based Dynamic Programming Techniques, *IEEE Trans. on PAMI*, 18(5), pp. 548-554, 1996.
- [MO97] H. Miled, C. Olivier, M. Cheriet and Y. Lecourtier, Coupling Observation/Letter for a Markovian Modelisation Applied to the Recognition of Arabic Handwriting, *ICDAR*, pp. 580-583, 1997.
- [MR81] C. Myers and L. Rabiner, A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition, *IEEE Trans. On Acoustics, Speech and Signal Processing*, Vol. ASSP-29, No. 2, April 1981.
- [MS98] J. Mao, O. Sinha and K. Mohiuddin, A system for cursive handwritten address recognition, *Proc. 14th Inter. Conf. on Pattern recognition*, August 1998, Brisbane, Australia, pp. 1285-1287.
- [Mw99] The Mathworks Inc., Matlab- Neural Networks Tool Box, 1999.
- [PL98] H. Park, and S. Lee, A truly 2-d hidden markov model for off-line handwritten character recognition, *Pattern Recognition*, 31(12), 1998, pp. 1849-1864.

- [PS97] R. Powalka, N. Sherkat and R. Whitrow, Word shape analysis for a hybrid recognition system, *Pattern Recognition*, 30, 3, 1997, pp. 421-445.
- [Ra89] L. Rabiner, A tutorial on Hidden Markov Models and selected applications in speech recognition, *Proceedings of the IEEE*, 77 (2), 1989, pp. 257-285.
- [RK98] G. Rigoll, A. Kosmala and D. Willett, A new hybrid approach to large vocabulary cursive handwritten recognition, *Proc. 14th Intern. Conf. on Pattern Recognition*, Brisbane, Australia, August, 1998, pp. 1512-1514.
- [RP94] J. Rocha and T. Pavlidis, A shape analysis model with applications to a character recognition system, *IEEE Trans. on PAMI*, 16(4), 1994, pp. 393-404.
- [Sc97] R. Schalkoff, *Artificial Neural Networks*, The McGraw-Hill Company, Inc., Singapore, 1997.
- [SL96] C. Suen, L. Lam, D. Guillevic, N. Strathy, M. Cheriet, J. Said and R. Fan, Bank Check Processing System, *International Journal of Imaging Systems and Technology*, Vol. 7, pp. 392-403, 1996.
- [SM98] P. Sinha and J. Mao, Combining multiple OCRs for optimizing word recognition, *Proc. 14th Intern. Conf. on Pattern Recognition*, Brisbane, Australia, August 1998, pp. 436-438.
- [ST80] M. Swamy and K. Thulasiraman, *Graphs, Networks and Algorithms*, Wiley Interscience, 1980.
- [WF01] A. Wang and K. Fan, Optical recognition of handwritten Chinese characters by hierarchical radical matching method, *Pattern Recognition*, vol. 34, no. 1, pp. 15-35, Jan 2001.

- [YC99a] X. Ye, M. Cheriet, C. Y. Suen and K. Liu, Extraction of bankcheck items by mathematical morphology, *Intl. J. Document Analysis and Recognition*, 2, (2/3), 1999, pp 53-66.
- [YC99b] X. Ye, M. Cheriet, and C. Y. Suen, Model-Based Character Extraction from Complex Backgrounds, *Proc. ICDAR99*, 1999, pp. 511-514.
- [YS98] B. Yanikoglu and P. Sandon, Segmentation of off-line cursive handwriting using linear programming, *Pattern Recognition*, 31(12), 1998, pp. 1825-1833.
- [ZM99] M. Zimmermann and J. Mao, Lexicon reduction using key characters in cursive handwritten words, *Pattern Recognition Letters*, 20, 1999, pp. 1297-1304.

Appendix A

Artificial Neural Networks

A.1. Introduction

Artificial Neural Network (ANN) is a collection of simple processing elements (neurons, units) that are connected together in the form of a directed graph, and organized to learn and perform a more complex task. Units' functionality does not exceed performing simple operations, e.g. addition and multiplication. A unit's output value is determined by its input, its interconnection with other units in the same layer, some optional external inputs (e.g. bias and momentum), and its activation function. ANN functionality is determined by its structure, individual neuron functionality, its training method, and data set used in the training phase.

ANN is characterized by its massively distributed and parallel computations. Ideally, each unit in a given layer could be dedicated to perform a single useful task toward the overall functionality of the ANN. All units in a given layer perform their computations in parallel. ANN is also characterized by its learning abilities that could replace any priori programmed knowledge. Thus, the same ANN could be used to perform two different tasks provided that proper training data is used for each task. These two characteristics represent the core computation innovation of ANN over the traditional procedural programmed computing.

In the following sections of this appendix we provide a brief description of the structure of ANN, main applications of ANN, categories of ANN, the major types of ANN, and the limitations of ANN.

A.2. Applications of ANN

ANN has been applied to numerous types of applications with significant success. Some of the applications where ANN has been used include: Image processing, pattern recognition, implementation of Boolean logic functions, solving classification problems, functional approximation, non-linear transformation, control, filtering applications, medicine and financial systems. In general, ANN is more applicable to problems that pose high dimensionality, large complexity, and/or mathematically intractable interaction between problem variables.

In addition to the application of ANN as a stand alone system, it also has been used in combination with other modeling techniques. Hidden Markov Model (HMM) was combined with ANN in numerous ways to improve accuracy and/or reduce risk. Following are some of the methods used to combine ANN with HMM:

- Sequential combination method, where the output of the ANN is used as a sequence of observations for the HMM.
- ANN could be used to perform the quantization process in discrete HMM.
- ANN could be used to estimate the observation probabilities during the training session of HMM.

In a similar way, the integration of ANN with fuzzy-systems yielded the emerging of fuzzy neural networks. Such integration has been defined in a number of ways, including:

- ANN were used to implement fuzzy systems and fuzzy logic.
- Fuzzy approaches may be used in the design and training of ANN.
- ANN could be used to train and tune fuzzy systems.

A.3. ANN Structure

ANN structure is defined by the number of layers, interconnection within and between layers, unit activation functions, and the number of neurons in each layer. At least two layers exist in any ANN: an input layer and an output layer. The input layer represents a holding site of the characteristics (features) that are fed to the ANN. These inputs can be homogeneous or heterogeneous list of values. In the latter case, it is essential to normalize the inputs to ensure a fair chance to all inputs to participate in the decision process. The output layer could use a single unit for each possible output, or may contain a single unit that outputs the identity (id number) of the winning unit. In addition to the input and output layers, ANN may contain a number of hidden layers composed of the core processing stage to give ANN the ability to perform non-linear functionality. Figure A-1 shows an example for a simple, yet useful ANN. Although ANN may use more than one hidden layer, yet it is not proven that this extends the functionality of ANN.

A connection between two units refers to a directed edge from one unit to another, in which the output of the unit at the beginning of the edge is fed to the unit at the end of the edge. Successive layers are mostly fully connected. Partial connection is yet an alternative that has been considered to impose certain constraints on the system or to reduce its complexity. Connections between units within a single layer is a characteristic of competitive networks.

The activation function refers to the operation or function that a unit performs to produce its output. Typically, a unit accepts an input, compares it with the corresponding weight, adds up external bias or momentum inputs and then performs the activation

function. Activation functions of hidden units are always chosen to be differentiable to ensure gradient descent training that leads to a convergence in the training procedure to local minima. The most commonly used activation functions for hidden units are *sigmoid*, *tan* and *Gaussian* functions. The output layer, on the other hand, may define a binary or a continuous function depending on the desirable ANN output.

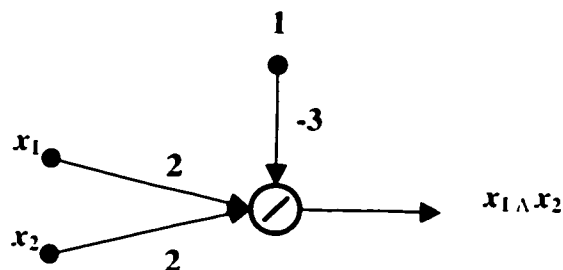


Figure A-1: A simple example of ANN of two layers. The input layer has two neurons and functions to hold and pass the input to the succeeding layer. The output layer contains a single neuron that accepts three inputs (two from the input layer and one is called the bias value) along with a weight for each input. The output unit basically multiplies each weight with the value of its corresponding input and add them together. The output unit then applies a hard-limit function to its net input. The above ANN implements a logical “and” operation between the two inputs. The straight line shown inside the output unit indicates the use of hard limit activation function at that unit.

A.4. Categories of ANN

ANN may be categorized based on a number of factors. Based on their memorization power, ANN may be categorized as either static or dynamic. ANN can be categorized based on the training strategy into supervised training and unsupervised training. Description of each of these categories is given below.

A.4.1. Static vs. Dynamic ANN

Static ANN employs no memory about the past inputs or outputs. Thus, the ANN reaches a static state upon completion of its training phase. Examples of this type of ANN include the multi-layer perceptron (MLP), the radial basis function (RBF) and self organizing maps (SOM). Dynamic ANN on the other side, maintains memory about past inputs and/or outputs. Examples of this type of ANN include time delay ANN (TDNN) and Hopfield network.

A.4.2. Supervised vs. unsupervised training

The nature of training data used to train ANN plays a role in the training strategy to be employed. In a supervised trained ANN, the training procedure expects pairs of inputs and their corresponding outputs. The ANN creates a mapping that matches the example pairs as closely as possible. Examples of this type of ANN include the aforementioned MLP, RBF and Hopfield network. On the other side, unsupervised trained ANN accepts only input examples without any expectations of their corresponding output values. ANN employs inhibitory connections between units to group the input patterns into a number of clusters. Examples of this type of ANN include Kohonen's self-organizing feature map (SOM) and the Adaptive Resonance Theory (ART).

A.5. Major ANN types

A.5.1. Multi-layer perceptron (MLP)

MLP represents one of the earliest attempts to extend the functionality of ANN beyond linear operations. The introduction of the hidden layer(s) in MLP was regarded a breakthrough in ANN as it proved (by example) the practicality of training hidden layers.

One of the major shortcomings of MLP however, is the limited availability of training algorithms, which reduces its applicability to small problems. Figure A-2 shows an MLP with one hidden layer.

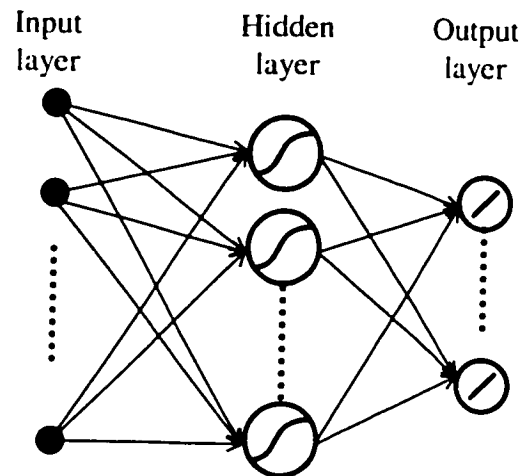


Figure A-2: MLP with sigmoid transfer function in the hidden layer

The net input of each hidden neuron is given in Eq. A-1 below.

$$net_i = f(\sum W_{ji} U_{ji}) \quad (A-1)$$

where U_{ji} is the input element number j to neuron number i , and W_j is the weight associated with that input. The most activation function used for the hidden layer is the sigmoid function. In Most cases, additional parameters are used to improve the performance of MLP (e.g. bias vector and momentum).

A.5.2. Competitive ANN

Competitiveness constitute the core of unsupervised learning strategy. Units in competitive ANN exhibit a competitive form of behavior. During the training process, the winning unit contributes negative activation to the non-winners to reduce their sensitivity

to particular pattern of inputs. During the testing phase, each unit matches its weights to the incoming signal, and sends negative contribution to other units according to some similarity measure. The winning unit will be the one that produces maximum similarity to the incoming signal and receives the minimum negative contribution from other units. Among the most commonly used competitive ANN are the self organizing maps (SOM) and the adaptive resonance network (ART). Figure A-3 shows the interconnections among hidden neurons in a competitive network.

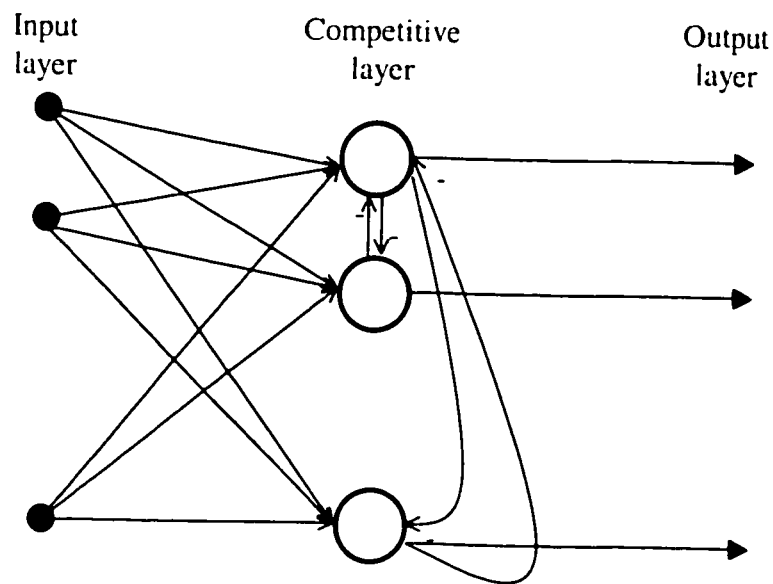


Figure A-3: Competitive network

A.5.3. Radial Basis Function (RBF) Networks

RBF can be seen as an extension of MLP with a special activation function of the hidden layer. The structure of RBF consists of three layers, input, hidden and output layers. The hidden layer consists of locally sensitive units that are governed by radial basis (commonly a Gaussian) activation function. The output layer consists of linear units. RBF has been used in, control, speech processing, image processing, pattern recognition and classification. Figure A-4 shows the basic structure of RBF.

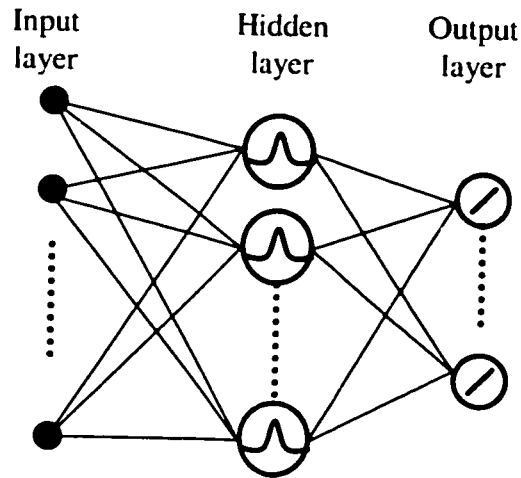


Figure A-4: The Radial Basis Function network

A.5.4. Recurrent (Hopfield) network

The major topological property of recurrent networks is the connection from the output layer to the input layer. Such ANN is trained via storage prescription that forces local minima points in the decision space to correspond to the stable representations (stages) of the input templates. The testing phase starts by exposing the external signal to the input layer. The network then enters into a recurrent loop where the output of the each stage becomes input to the next one. The iteration process continues until the system reaches a stable stage, which mostly correspond to one of the stored templates. This type of network could be used to recall template image patterns, or to complete partially specified patterns. Figure A-5 shows the architecture of the Hopfield network.

A.6. Limitations of ANN

Like any other model, ANN has its own difficulties and limitations. First, achieving the global minimum error during the training phase of ANN is not visible. Various methods were invented to reduce the effect of local minima, but no algorithm can eliminate such

effect yet. Second, the time complexity of the training procedure makes it a relatively expensive option. A third important consideration related to ANN is to ensure the generality of the ANN after the training procedure. Various training and stopping criteria were used to avoid over fitting the training data. A fourth issue of concern is how to choose the proper network structure for a particular application. Finally, the inference about the ANN decision making process is not visible. This is due to the use of ANN as a black-box that does not facilitate such inference.

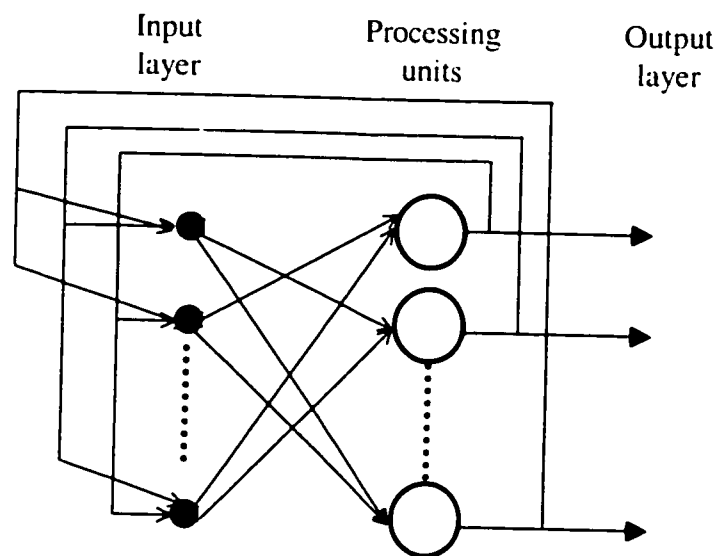


Figure A-5: Hopfield network

Appendix B

Graph Theory

The graph theory was founded by Euler with his solution to the Königsberg bridge problem in the 18th century. However, its first application in physical science came after more than a century of its initial foundation. Over the past four decades, graph theory has evolved into an important mathematical tool in the solution of a wide variety of problems in many areas, e.g. physics, and chemistry. In this appendix, we provide the basic definitions that have been referred to throughout the text of the thesis. We also provide some of the theorems of concern to the thesis along with the proof of each of them.

B.1. Basic Definitions

Graph: A graph $G=(V, E)$ is a set of vertices V connected via a set of edges E (Figure B-1). In cases where there is at most a unique edge between any two vertices, the edges may be labeled by its ending vertices. If there is more than one edge between any two vertices in the graph, the edges have to carry distinct labels.

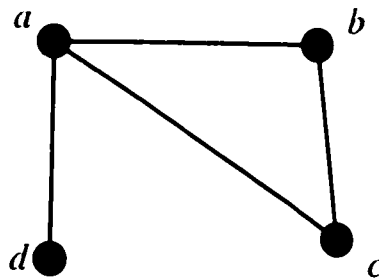


Figure B-1: A graph with four vertices (a,b,c,d) and four edges ((a,b),(a,c),(a,d),(b,c))

- Walk:** A walk in a graph G , is a sequence of vertices and edges. In Figure B-1 above, the sequence $\{a, (a,b), b, (b,c), c, (c,b)\}$ is a walk. For simplicity, we may say that $\{a,b,c,b\}$ is a walk.
- Trail:** A walk is called a trail if all its edges are distinct. In Figure B-1 above, the sequence $\{a,b,c,a\}$ is a trail.
- Path:** A walk is called a path if all its vertices are distinct. In Figure B-1 above, the sequence $\{a,b,c\}$ is a path.
- Closed Trail:** A trail is called *closed* if its end vertices are identical.
- Cycle:** A cycle is a closed trail that has one or more edges. In Figure B-1 above, the trail $\{a,b,c,a\}$ forms a cycle.
- Connected graph:** A graph is connected if there is a walk between every pair of its vertices. The graph shown in Figure B-1 is a connected graph.
- Weighted graph:** A weighted graph is a graph that assigns weight to each edge.
- Degree of a vertex:** The degree of a vertex v , denoted $d(v)$, is the number of edges connected to that vertex. In Figure B-1 above, $d(a) = 3$.
- Pendent vertex:** A pendent vertex is a vertex with degree of one.
- Tree:** A graph is called a tree if it is connected and has no cycles.
- Star:** A star is a connected graph of three or more vertices, in which all vertices are pendent except one. Figure B-2 shows a star of 5 vertices.
- Center of a star:** The center of a star is the vertex that is not pendant. It connects all other vertices to the graph. In Figure B-2, the vertex a is the center of the star.

Double-traced edge: Double-traced edge is an edge that must be traced more than once to traverse a given graph.

Euler trail: An Euler trail in a graph G is a trail that traces all the edges in G exactly once. An Euler trail is said to be *open* if its end vertices are distinct, and is said to be *closed* otherwise. In figure B-1, the trail $\{d,a,b,c,a\}$ is an open Euler trail. A graph that possess an Euler trails is called Eulerian graph.

Neighbor (adjacent): Neighbors of a vertex v are all vertices that share an edge with v .

In Figure B-1 above, the neighbors of a are b , c and d .

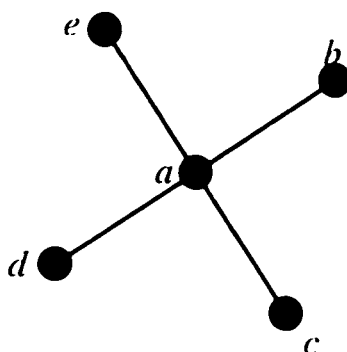


Figure B-2: A star

B.2. Theorems

Theorem B-1

Traversing all vertices in a tree T is equivalent to traversing all edges in T .

Proof:

This theorem can be proven in two parts. First we prove that traversing all edges in a tree implies traversing all vertices. Second we proof that traversing all vertices in a tree implies traversing all edges.

I) Traversing all edges in a tree \Rightarrow traversing all vertices.

1. Assume that $T(V,E)$ is a tree where the above implication does not hold.
2. \Rightarrow We are able to traverse all edges in T without visiting some vertex $v \in V$.
3. \Rightarrow The vertex v was not connected to any of the edges of T .
4. $\Rightarrow T$ is not a connected graph.
5. The consequence achieved in (4) contradicts the assumption made in (1) above.

Note that T is a connected graph by definition. Therefore, such assumption can not hold.

II) Traversing all vertices in a tree \Rightarrow traversing all edges.

1. Assume that $T(V,E)$ is a tree where the above implication does not hold.
2. \Rightarrow We are able to visit all vertices in T without tracing an edge $e(v_i, v_j) \in E$.
3. \Rightarrow There is a walk w between v_i to v_j that does not pass through e .
4. \Rightarrow There is a cycle in T composed of (w, e) .
5. The consequence stated in (4) contradicts the assumption made in (1). Note the T does not contain cycles by definition. Therefore, such assumption can not hold.

From (I) and (II) above, we can conclude that traversing all vertices in a tree is equivalent to traversing all edges in the tree.

Theorem B-2

For any tree of n vertices, there are exactly $n-1$ edges.

Proof:

1. This theorem applies clearly to any tree of 1 or two vertices (Figure B-3).
2. Assume that this theorem applies for all trees of n vertices or less.
3. We need to prove that the theorem applies for all trees $T=(V, E)$ of $n+1$ vertices:

- a. Consider any edge $e \in E$. The edge e constitute the only link between its end vertices.
- b. Remove the edge e from E , to break T into two distinct trees $T_1=(V_1, E_1)$ and $T_2=(V_2, E_2)$. Assume that n_1 is the number of vertices in T_1 and n_2 is the number of vertices in T_2 .
- c. Note that $n_1 \leq n$ and $n_2 \leq n$. Thus, the theorem applies to both T_1 and T_2 , i.e. T_1 has n_1-1 edges and T_2 has n_2-1 edges.
- d. Note also that the number of edges in $T = (n_1 - 1) + (n_2 - 1) + 1$. This counts for all edges T_1 in plus all edges in T_2 plus the removed edge in step (b) above.
- e. From (c) and (d) above, we conclude that T has $(n_1-1) + (n_2-1) + 1 = n_1 + n_2 - 1 = n-1$ edges.
- f. Therefore, the theorem applies for all trees of $n+1$ vertices too.

From (1), (2) and (3) we conclude that the above theorem applies to all trees.



Figure B-3: (a) The tree with single vertex has 0 edges, (b) the tree with 2-vertices has 1 edge

Theorem B-3

For any tree $T (V, E)$, there are at least two pendant vertices.

Proof:

Suppose that T has n vertices.

$\Rightarrow T$ has $n-1$ edges

\Rightarrow The sum of degrees in $T = \sum d = 2(n-1)$. (Each edge is counted for two vertices)

Let us try to assign these degrees to all vertices in T .

The minimum degree that can be assigned to a vertex is 1 (because T is connected)

Therefore, we are left with $(n-2)$ degrees to be assigned to n vertices.

\Rightarrow At least two vertices will not be assigned any of the $(n-2)$ degrees.

Thus, at least two vertices will be pendent vertices in T .

Theorem B-4

The minimum cost for traversing all vertices in a tree $G(V,E)$ is the sum of all its edges.

Proof

The proof can be easily shown using mathematical induction:

1. For tree two vertices, the minimum cost is clearly the cost of its unique edge (Figure B-4). Thus, the theorem holds for a tree of two vertices.
2. Assume that the theorem holds for all trees with n vertices.
3. We need to prove that the theorem holds for any tree $T(V,E)$ of $n+1$ vertices. Consider the following:
 - a. Find a pendent vertex $v \in V$.
 - b. Remove the vertex v and its connecting edge e , to get another tree $T_2 (V_2, E_2)$ of n vertices.
 - c. The theorem certainly applies to T_2 because it has n vertices. Thus, the minimum cost to traverse all vertices in T_2 is the $\sum E_2 = \sum E_1 - e$.
 - d. Having all vertices in T_2 visited, the minimum additional cost to visit v can not be less than the cost of the edge e .

e. \Rightarrow The minimum cost to traverse all vertices in T can not be lower than $\sum E_2 + e$,
which is equal the $\sum E$.

f. \Rightarrow The theorem also applies to any tree of $n+1$ vertices.

From 1.2 and 3, we can conclude that the minimum cost to traverse all vertices in a tree is the sum of its edges.

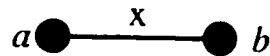


Figure B-4: The minimum cost to visit the two vertices a and b is x

Theorem B-5

In the most efficient traversal walk of a tree T , each vertex of degree d would involve between $|d-2|$ and d (inclusive) double-traced edges.

Proof

1) Let v be an internal vertex in a traversal walk of T , i.e. v is not the starting nor the ending vertex. The degree of v , $d(v)$ can be either one or more than one. Let us consider each of these cases:

1) $d(v) = 1$

In this case, v is a pendant vertex that should be visited during the traversal walk. Since there is only one edge e connecting v to the rest of the tree, e has to be traced twice: one to visit v and the other to connect back to the remainder of the tree to continue the traversal process. Thus, visiting v requires 1 double-traced edge, which is equal to d .

2) $d(v) > 1$

Assume that the vertex v was reached through the edge (v_i, v) . There are two possibilities about the location of the ending vertex of the whole traversal process:

- a) If the ending vertex lays toward v_i then the traversal of v and its neighbors should end at v_i , which implies that all the edges of v are double traced (d double-traced edges).
- b) If the ending vertex does not lay toward v_i but lays toward another neighbor of v , say v_j , then the two edges (v_i, v) and (v, v_j) do not need to be double traced. However, all other edges (v, v_k) $k \neq i$ and $k \neq j$, need to be double traced since after all neighbors k , we will need to take the edge (v, v_j) , which implies taking the edge (v_k, v) first. Thus, exactly $d-2$ edges should be double traced.

From (1) and (2) we conclude that the theorem applies to all internal vertices.

II) Let v_s be the starting vertex and v_e be the ending vertex of a certain traversal of T .

There are only two possibilities about the identities of these two vertices.

- 1) $v_s = v_e$

In this case, all edges of v_s should be double traced to ensure ending at the same vertex v_s . Thus, d edges are double-traced.

- 2) $v_s \neq v_e$

In this case, all but one (the one leading to v_e) of the edges of v_s should be double traced. Thus, $d-1$ edges are double traced. The same can be said about v_e .

From (1) and (2) above, we conclude that the theorem applies to the start and ending vertices.

From (I) and (II) we conclude that the theorem applies to all vertices in the tree T .

Appendix C

HMM

C.1. Introduction

HMM is an extension of finite automaton machines (FAM). FAM tries to predict the future step (state) or observation based on the past steps and observations. For example, in the stochastic network shown in Figure C-1, if the current state is S_1 , then there is 0.6 probability that the next state will be S_2 , and 0.4 probability the next state will be S_3 . Thus, one may conclude that S_2 is most probably the next state.

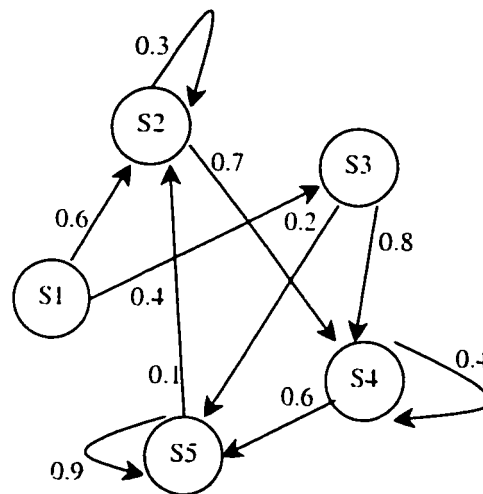


Figure C-1: Example of a stochastic network with 5 states. Transition probabilities are shown near their origin

In real world problems, there are other factors that affect the transition between various states. One other factor that affects the identity of the next state is the observation probabilities. The observation probabilities are shown in Figure C-2, in which it is not sufficient to look into the transition probabilities alone. If the current state is S_1 , and the

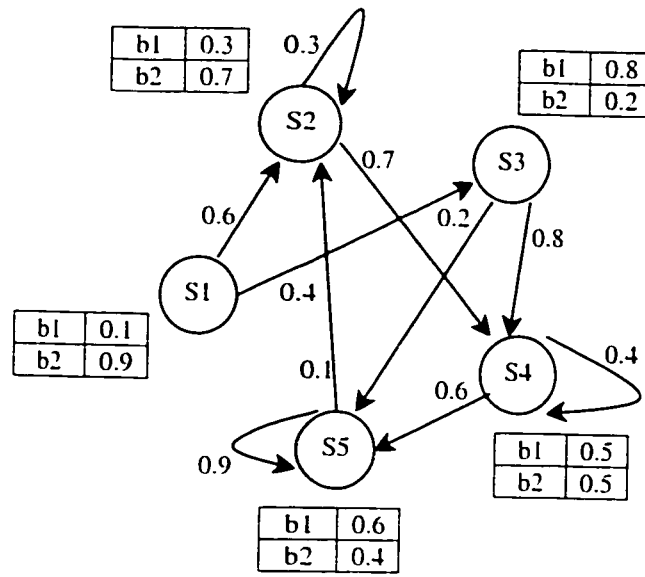


Figure C-2: Example of a statistical network with 5 states and their observation probabilities

next observation is b_1 , then the probability of having S_2 as the next state is 0.18 ($0.6 * 0.3$), and the probability of having S_3 as the next state is 0.32 ($0.4 * 0.8$).

The process of state-prediction can be repeated more than once by multiplying successive state probabilities. However, this will yield accurate results only if the successive events (state transitions and observations) are statistically independent. Statistical independence between any two events A and B means that the occurrence of A does not influence the occurrence or absence of B and vice versa. This can be represented mathematically by equation C-1 below:

$$P(A|B) = P(A) \quad (C-1)$$

where $P(A)$ is the probability of event A , and $P(A|B)$ is the probability of event A given that event B has occurred.

This sequence of operations can be used to predict (infer) the system's behavior. Given a sequence of observations, how probable it is that it would pass (has passed) through a specific sequence of states? This sequence of operations can also be used to test the system ability to experience a particular sequence of events. Given a sequence of observations, how probable is it that a particular system would use it to pass from the initial state to the last state? In other words, how suitable is a given sequence of observations to a particular system? In each of the above cases, the known sequence of observations is used to detect the hidden sequence of states, hence the name Hidden Model.

A Markov process is a stochastic system for which the identity of the next state is assumed to be solely dependant on the immediately preceding state. Markov chain is simply a chain of random Markov processes. Markov chain represents a very important concept that incorporates a minimum amount of memory without being memoryless.

The k^{th} order Hidden Model assumes that the next state is dependent on the past k states. To reduce computation complexity, first order Hidden Model is often used. The first order Hidden Model basically applies Markov assumption that the next state depends only on the last state and on the current observation. This is what is called Hidden Markov Model (HMM).

HMM has been used in various types of applications including voice recognition, and handwriting recognition. The common ground between various applications of HMM to pattern recognition problems is to define a model for each target class. Each model is then trained to produce high probability (response) to the kind of observation sequences that match its target class, and/or to produce a low probability to the kind of observation

sequences that do not match the target class. The overall result of the HMM recognition system will be the model that produces the highest probability for a given sequence of observations.

An HMM is completely specified by the of parameters:

N : the number of states in the model,

M : the number of distinct observations that could be encountered by the system,

A : the transition probabilities (the probability of passing from state S_i to state S_j),

B : the emission probabilities (the probability of having an observation o at state S_i),

π : the initial state probabilities.

C.2. Training an HMM

The objective of the training phase is to optimize the model parameters so that it responds properly to various patterns of observations. This is the most difficult part of HMM as it involves maximization of complex parameters in order to increase the probability of observing a sequence given the model. In fact, there is no known training algorithm that would produce a global maximum of such parameters. However, there are a number of algorithms that aim to find a local maximum of such variables. In the following we describe an iterative procedure that is commonly used to train HMM based on the Baum-Welch method.

The Baum-Welch iterative procedure can be divided into two phases, computation phase and probability adjustment phase. At each iteration, the model probabilities are used to deduce an overall model probability (computation phase). These computed probabilities are then used to adjust the model parameters in order to increase such a probability at the next iteration.

Of the two phases mentioned above, the computation phase is by far more complex. The difficulty in the computation phase arise from the fact that, for a given model $\lambda(N, M, \pi, A, B)$, there are $N(M+N+1)$ different values to be manipulated in order to maximize the overall model probability. These values correspond to π (N items), A (N^2 items) and B ($N*M$ items). Obviously, such manipulation should maintain the stochastic property for all the three parameters π , A and B .

The computation phase computes the probability of being in each state given the training observation sequence as well as the model parameters. In other words, for a given time step t , these probabilities are dependent on the values obtained in the previous time step $t-1$ and on the current values of the model parameters. This introduces the notion of forward variable α , which is used to keep the values obtained in step $t-1$ to avoid repeating these computations. At a certain time step t and a given state S_j , the forward variable α holds the probability of partial observation sequence o_1, o_2, \dots, o_t and state S_j given the model λ , or in mathematical notation:

$$\alpha_t(j) = P(O_1 O_2 \dots O_t, q_t = S_j | \lambda) \quad (C-2)$$

The computation of the forward variable at a given time step t is computed as follows:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(O_{t+1}) \quad (C-3)$$

The above equation can be represented graphically as shown in Figure C-3 below. The interesting thing about the above equation is that the computation at a given time step $t+1$ is induced from a simple manipulation of the computations achieved at time step t . Thus, eliminating the huge computational load involved in re-evaluating each of the previous

steps. The computation of the first step (time step 1) depends solely on the emission and the initial state probabilities as follows:

$$\alpha_1(j) = \pi_j b_j(O_1) \quad (\text{C-4})$$

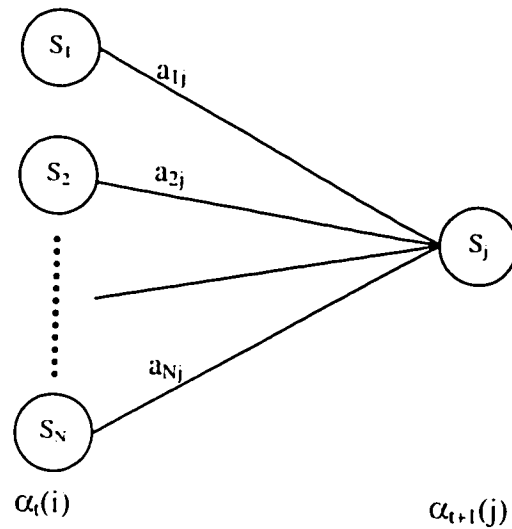


Figure C-3: Graphical representation of the computation of the forward variable at time step $t+1$ at state S_j

In the same way the forward variable is defined, the backward variable, β , is the probability of the partial observation sequence from $t+1$ to the end of the observation sequence, given state S_i at time t and the model λ . The computation of β at time t given state S_i is shown in Figure C-4, and is given by Equation C-5 below.

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \quad (\text{C-5})$$

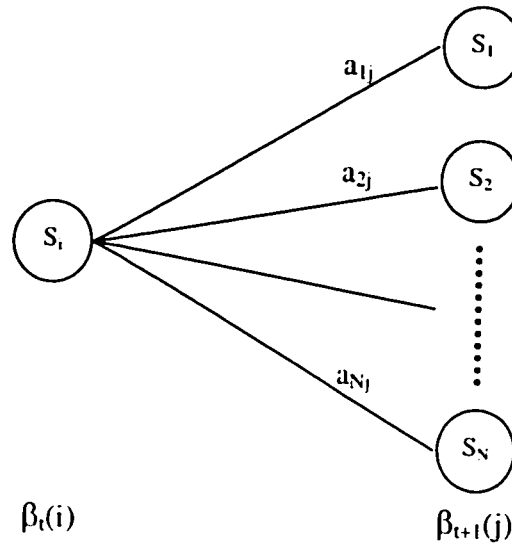


Figure C-4: Graphical representation of the computation of the backward variable at time step t at state S_i

The initialization part of β at time T (the end of the observation vector) is defined arbitrary to be 1 for all states.

The forward and backward variables are then used to compute the expected relative frequencies (or probabilities) necessary to adjust the model parameters. The probability of being in state S_i at a particular time step t given the model and the observation sequence, $\gamma_t(i)$, is computed as follows:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^N \alpha_t(j)\beta_t(j)} \quad (\text{C-6})$$

This translates to the probability of reaching state S_i at time t , times the probability of leaving state S_i right after t , divided by the probability of being in any state in the model at time t . In addition, the probability of being in state S_i at time t and state S_j at time $t+1$ given the model and the observation sequence, $\zeta_t(i,j)$, is computed as:

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(i)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(i)} \quad (C-7)$$

This translates to the probability of reaching state S_i at time t , times the transition probability from state S_i to state S_j , times the probability of observing O_{t+1} at state S_j , times the probability of leaving state S_j right after the next observation, divided by the total transitions between all states in the model at time t .

Having the computation phase completed, the adjustment phase follows the basic probabilistic rules. The initial probability of state S_i , π_i , is readjusted to reflect the relative frequency that the model starts the observation process at state S_i . This is shown in Equation C-8 below.

$$\pi_i = \gamma_i(i) \quad (C-8)$$

The transition probability between state S_i and state S_j , a_{ij} , is defined as the relative frequency of transitions between the two states to the number of transitions initiated from the state S_i . Equation C-9 gives the mathematical notation for this computation.

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (C-9)$$

The emission probability of observation o at state S_i , $b_i(o)$, is the relative frequency of observing o in state S_i relative to the number of times the model be in state S_i . This computation is shown mathematically in Equation C-10 below.

$$b_i(k) = \frac{\sum_{t=1}^T \gamma_t(i)}{\sum_{t=1}^T \gamma_t(i)} \quad (\text{C-10})$$

To achieve a local maxima during the training process, the computation and adjustment processes need to be repeated a number of times. The initial probabilities used at the first computation process can be selected randomly or be defined to impose desirable model properties. Note that a zero initialization of any probability prevents any adjustment on that particular probability as its relative frequency will never exceed zero.

C.3. Types of HMM

As mentioned earlier, HMM can be categorized based on a number of variables. Based on the type of observations, HMM is classified into discrete and continuous models. As indicated by its name, a discrete model expects a discrete list of observations. Such a list comes from a finite set of observable events. In cases where the original observations are continuous in nature, a quantization process is introduced. Quantization could be seen as another layer after the feature extraction layer. It aims to reduce the dimensionality of the classification problem by shrinking the number of characterizing tags that are available to describe a sequence of observations. As any other layer in the recognition system, the quantization process could be a source of some errors. Such error can be reduced by using more efficient quantization methods, increasing the size of the finite set of observable events (the domain of the mapping process).

Continuous HMM is an attempt to eliminate the error involved in the quantization process using continuous density functions to characterize the observation probabilities.

For practical considerations, these density functions are assumed to be known in their general form (normal distribution is assumed in most cases). To permit more complex decision boundaries, a combination of a finite number (M) of density functions (mixtures) is assumed at each state. The emission probabilities are computed as a linear combination of these functions as shown in Equation C-11 below.

$$b_j(O) = \sum_{m=1}^M C_{jm} \Pi(O, \mu_{jm}, U_{jm}) \quad (C-11)$$

where O is the observation vector to be modeled, C_{jm} is the coefficient of the m^{th} mixture in the j^{th} state, and Π is any log-concave or elliptically symmetric density function with mean μ_{jm} and covariance U_{jm} . What need to be estimated in the training session of continuous HMM are the parameters of the each mixture Π in each state (i.e. the mean and covariance) and the coefficients of these mixtures.

Though the amount of error introduced with the assumptions made in continuous HMM is theoretically less than those introduced by the quantization process, continuous models require more resources than those required by the discrete HMM.

Another way to categorize HMM models is based on the structure used to model patterns. The structure of a given model is defined in terms of the number and connectivity of given states. Among the most widely used structures in pattern recognition systems, the left-right model dictates a certain ordering of states in the model. Model states are numbered and the process of presenting the observations start at the pre-determined initial state. During this process, state transitions are only allowed toward the last state in the model.

Other variations in the structure of HMM models include the following:

- Null transitions, where some transitions are allowed without encountering any observation. Such transitions allow the absence of some observations from the characterizing observation sequence.
- Tied state probabilities, where two or more probabilities are set to be equal in value. This helps in modeling symmetrical events in the input patterns.
- Inclusion of state duration, where self-transitions are set to zero, and another density function is defined to characterize the duration of each state. This improves the model response in cases where there is a large number of self transitions in the system.

Optimization criterion refers to the learning method that is used during the training process. Among the most common criteria, Maximum Likelihood (ML) criterion trains each model on its own samples to maximize the samples likelihood. Models are not presented with samples of other classes. An alternative to this method is to maximize the discrimination power of each model using the maximum mutual information (MMI) criterion. In MMI, the average mutual information between the observation vector of the target model and the complete set of models is maximized. Another alternative criterion is to minimize the discrimination information (DI) or the cross entropy between the set of valid signal probability densities (Q) and the set of HMM probability densities.

C.4. Extensions to HMM – level building algorithm

HMM has been successfully used to process complex signals that contain more than one simple pattern. While the viterbi algorithm is used to re-track the optimal sequence of states within a single model, level building algorithm can be used to re-track

the optimal sequence of models contained in the complex signal. The algorithm works on the assumption that each model is connected to every other model in the HMM classifier.

The level building algorithm is used to empower HMM with segmentation capabilities. That is, although the complex signal may contain more than one simple pattern, yet the level building algorithm allows HMM classifiers to detect the sequence of simple patterns that composes the complex signal without pre-segmentation. Other classifiers rely on external segmentation algorithms that make segmentation decisions before the classification module. Thus, segmentation decisions are based on different criteria than the identity of the simple patterns. Level building algorithm allows the segmentation step to be embedded within the classification module and be based on the identity of the simple patterns. This has a number of advantages. First, it avoids premature errors that could apprehend the classifier. Second, it allows the determination of simple patterns based on their identities. Third, it allows the determination of global optimal sequence of simple patterns rather than optimizing the identity of each individual pattern.

On the other hand, level building algorithm employs a number of thresholds that can limit its generality. Implementation considerations normally impose the use of a maximum number of simple patterns that could be included in a single complex signal. Any increase in this number causes an increase in the search time of various pattern combinations. The search time is normally reduced by pruning based on various criteria, e.g. minimum carried out probability, and minimum and maximum lengths of simple patterns. Constraining the possible combinations of simple patterns can also be used to reduce the time complexity of the search process when applications permit.

Appendix D

Samples from the databases

It would be very useful to show the whole databases in this appendix. However, due to the huge sizes of these databases, we can include only some typical samples of each one. Nevertheless, the complete databases are available at CENPARMI.

D.1. Complete Cheques

٠٠٠٤٩٤

شركة الراجحي المصرفية للاستثمار
AL RAJHI BANKING & INVESTMENT CORP.

Date ٢٠١١/٣/٢٢

AL THALATHEEN STREET OLLEYA
BRANCH RIYADH

فرع شارع الثلاثين العليا
الرياض

Against this Cheque
Pay to the Order of

الموسى

١٠٠٠٠

The amount of

المبلغ عشرة آلاف ريال فقط لأخر
دفعة من الحساب

٦

Signature

٠٠٢٨٤٨

شركة الراجحي المصرفية للاستثمار
AL RAJHI BANKING & INVESTMENT CORP.

Date ٢٠١١/٣/١٢

AL GAEDAH ROAD BRANCH
AL-KHARJ

فرع طريق القاعدة العرج

Against this Cheque
Pay to the Order of

١٨٩٥

The amount of

المبلغ اثنى عشر ألفاً وتسعون ريالاً فقط

٧

Signature

٠٠١٤٧١

شركة الراجحي المصرفية للاستثمار
AL RAJHI BANKING & INVESTMENT CORP.

Date ١٤٣١/١١/١٩

AL RABWAH TRADE CENTER, RIYADH

الربوة المركزي الرياض

Against this Cheque
Pay to the Order of

ادفعوا بموجب هذا الشيك لأمر

The amount of

١٠٠٠٠٠٠٠
مليون وخمسة مائة الف ريال سعودي
دفعها باسم شركة الراجحي

مبلغ خمسة مائة الف ريال سعودي
دفعها باسم شركة الراجحي

ريال SR ٥٠٠٠٠٠٠

Signature

التوقيع

٠٠٠٠١٧

شركة الراجحي المصرفية للاستثمار
AL RAJHI BANKING & INVESTMENT CORP.

Date ١٤٣١/١١/١٩

HAI AL WORUD BRANCH
RIYADH

فرع حي الورود الرياض

Against this Cheque
Pay to the Order of

ادفعوا بموجب هذا الشيك لأمر

The amount of

المبلغ نحو اربعة ودمسون الف ريال سعودي

ريال SR ٤٠٠٠٠

Signature

التوقيع

٠٠٣٩٨٦

شركة الراجحي المصرفية للاستثمار
AL RAJHI BANKING & INVESTMENT CORP.

Date ١٤٣١/١٠/٢٤

ASWAG AL SHEMAL BRANCH, RIYADH

اسواق الشمال الرياض

Against this Cheque
Pay to the Order of

ادفعوا بموجب هذا الشيك لأمر

The amount of

مبلغ خمسة مائة الف ريال سعودي
سداد دفتر مصرف

ريال SR ٥٠٠٠٠٠

Signature

التوقيع

رقم ٠٠١٣٥٢

شركة الراجحي المصرفية للاستثمار
AL RAJHI BANKING & INVESTMENT CORP.

Date ١٤١٨/٠٨/٢٢

AL GUDUS BRANCH, RIYADH

فرع القدس الرياض

Against this Cheque
Pay to the Order of

بمعمو بموجب هذا الشيك لأمر

The amount of

مبلغ فقط ألف ريال لا غير
دستمعت اى
١٠٠٠
ريال SR

Signature

رقم ٠٠٠٤٧١

شركة الراجحي المصرفية للاستثمار
AL RAJHI BANKING & INVESTMENT CORP.

Date ١٤١٨/٠٧/١٠

HAI ALSALAM BRANCH

فرع حي السلام الرياض

Against this Cheque
Pay to the Order of

بمعمو بموجب هذا الشيك لأمر

The amount of

مبلغ فقط مئتا ألف ريال لا غير
دستمعت اى
١٠٠٠٠
ريال SR

Signature

رقم ٠٠١٥٦٧

شركة الراجحي المصرفية للاستثمار
AL RAJHI BANKING & INVESTMENT CORP.

Date ١٩٩٨/٣/١٩

MUSHRIFA BRANCH, RIYADH

فرع مشرفة الرياض

Against this Cheque
Pay to the Order of

بمعمو بموجب هذا الشيك لأمر

The amount of

مبلغ فقط أربعمائة وخمسون ريال لا غير
مقباسبت
٤٥٠
ريال SR

Signature

D.2. Legal Amounts

المبلغ ثلاثون ألف ريال فقط شهري ريال
S.R.

عشره الاف وثمانه مائه وسته واربعون ريال فقط

ستة الآف ريالاً فقط واحداً عشر ريال

المبلغ عشرين ألف ريال فقط شهري ريال
S.R.

فقط عشرون ألف ريال فقط

فقط عشره ألف ريال فقط

وقدره ألفان وخمسمائة وعشرون ريال فقط

وقدره العنبر ومائتة ريال

فقط ثمانية آلاف ومبت مائة وواحد

ثلاثة عشر ألف وخمسة وأربعون ريالاً

فقط خمسة آلاف ريال لا غير .

إثنان وعشرون ألفاً وستة وثلاثون ريالاً

عشرة آلاف ريال فقط

وقدره خمسون ألف ريال فقط والمبلغ

ثمانته الألف ريال فقط

عشرة آلاف - ٥٤

أربع عشرة ألفاً وخمسة عشر ريالاً

تسعة آلاف وخمسة مئة ريالاً

فقط ستة آلاف ريالاً لا غير

أربعة آلاف وسبعمائة وستة عشر ريالاً

أربعة آلاف وسبع مائة ريالاً فقط لا غير

D.3. Courtesy Amounts

~~10~~ 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

~~100~~ 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000

~~1000~~ 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000

10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000

100000, 200000, 300000, 400000, 500000, 600000, 700000, 800000, 900000, 1000000

1000000, 2000000, 3000000, 4000000, 5000000, 6000000, 7000000, 8000000, 9000000, 10000000

2 ...

0 ...

5 1 3 0 #

9 0 ...

X 0 ... X

1 0 ... #

// 1 0 ... //

Handwritten scribbles in a box, including a large 'W' shape and horizontal dashes.

Handwritten scribbles in a box, including a 'W' shape and a '#' symbol.

Handwritten scribbles in a box, including a '#' symbol and the number '110'.



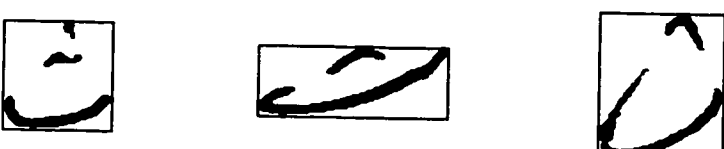



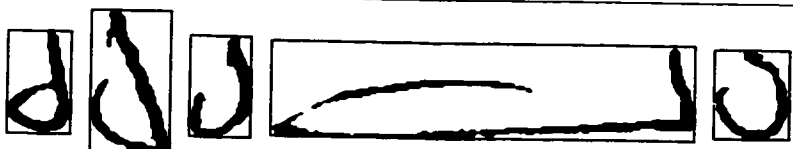
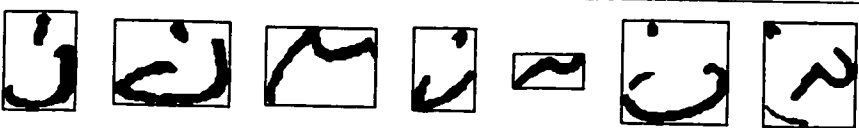

Handwritten scribbles in a box, including a large 'Z' shape, the number '017', and a '#' symbol.




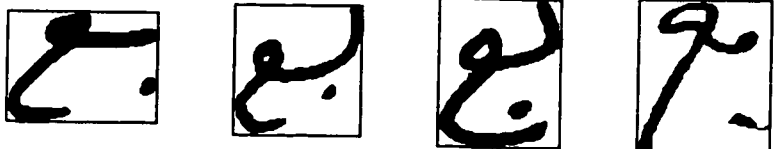

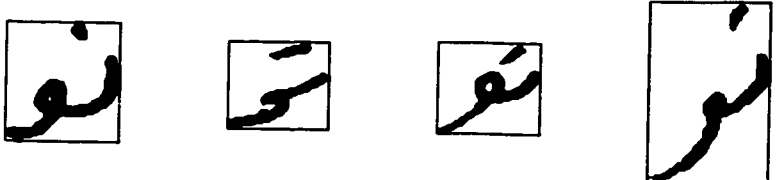
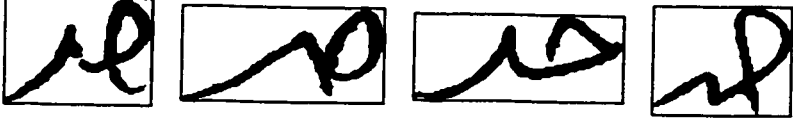

Handwritten scribbles in a box, including a large 'Z' shape, a 'W' shape, and a '#' symbol.






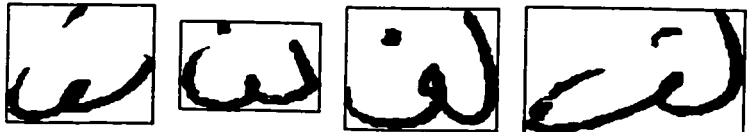



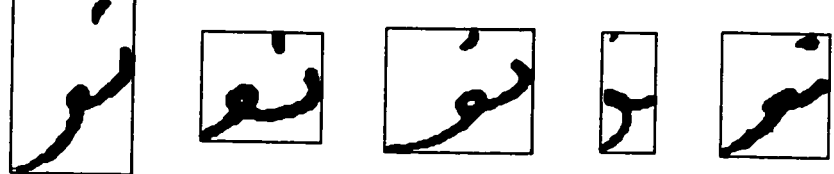

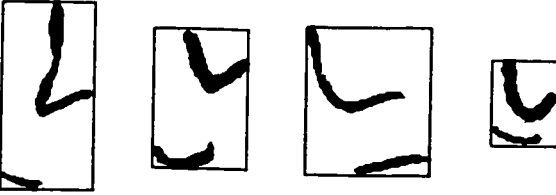




Handwritten scribbles in a box, including a large 'Z' shape, the number '131', and a '#' symbol.




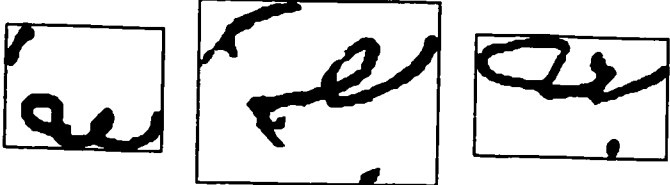



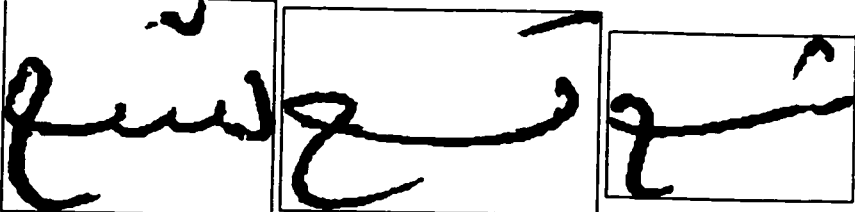

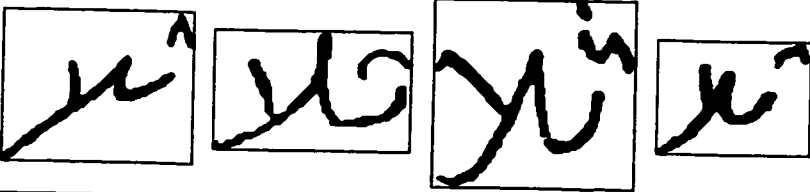




Handwritten scribbles in a box, including a large 'Z' shape, a 'W' shape, and a '#' symbol.

D.4. Sub-words


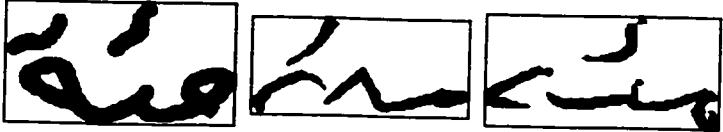
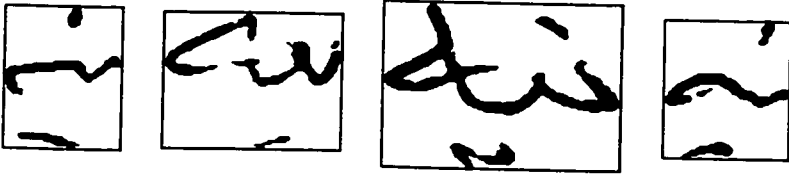
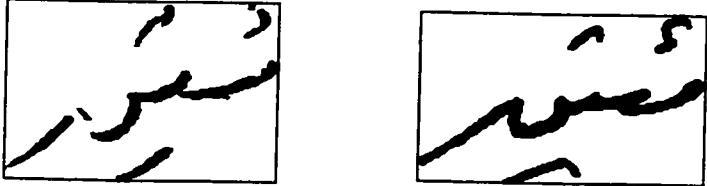
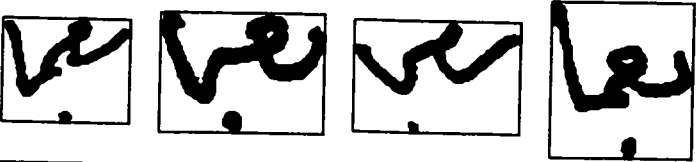
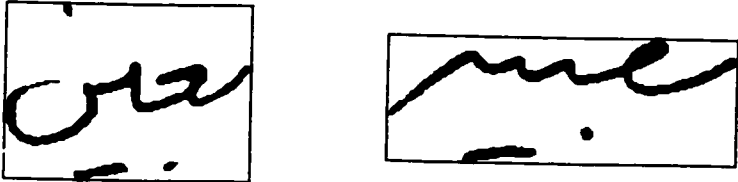


Machine Printed Sub-word	Handwritten Samples
ا	
ت	
ث	
ن	
ر	
ف	
ك	
ذ	
ة	

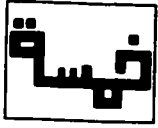
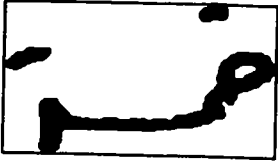
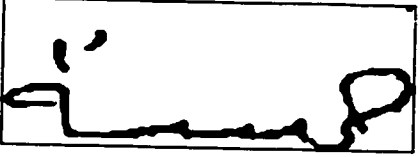
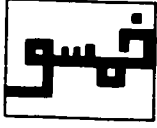
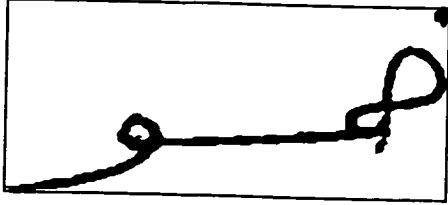

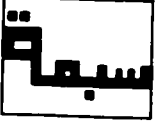
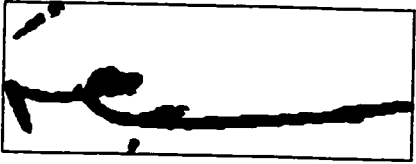
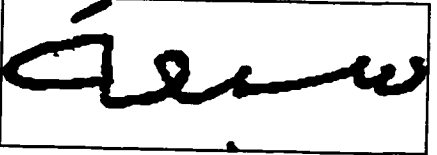
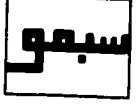
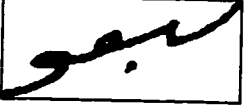



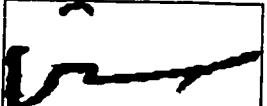

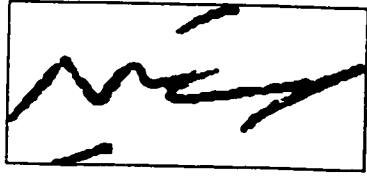
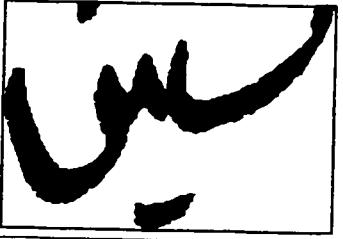

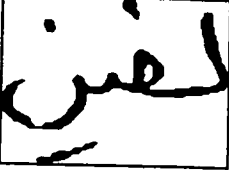

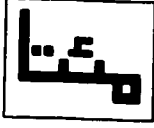

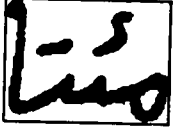
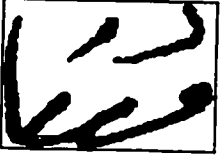
Machine Printed Sub-word	Handwritten Samples
و	
ی	
عق	
شق	
عج	
شع	
عز	
ط	
ست	



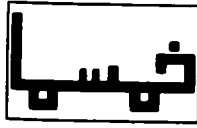

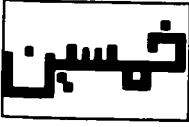
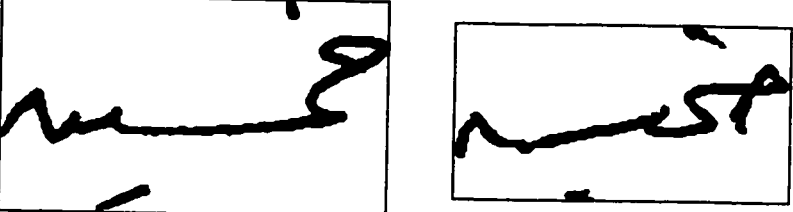
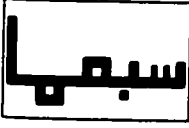
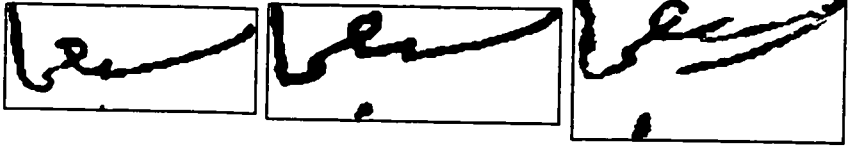


Machine Printed Sub-word	Handwritten Samples
	
	
	
	
	
	
	
	

Machine Printed Sub-word	Handwritten Samples
	
	
	
	
	
	
	




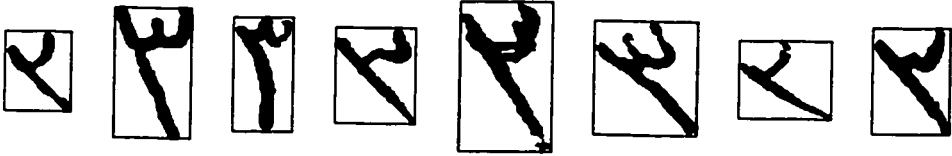



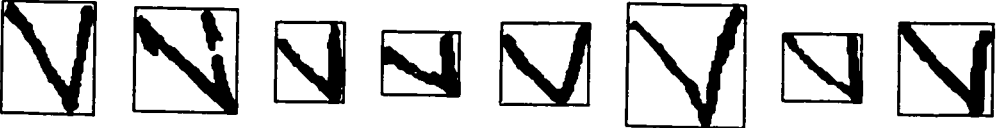
Machine Printed Sub-word	Handwritten Samples		



Machine Printed Sub-word	Handwritten Samples
لفا	
لقتا	
لقتا	
لقتين	
لجا	
لجتا	
لقتا	
لقتو	

Machine Printed Sub-word	Handwritten Samples	
		
		
		
		
		
		
		
		
		

Machine Printed Sub-word	Handwritten Samples
	
	
	
	
	

D.5. Indian Digits

Digit	Samples
0	
1	
2	
3	
4	
5	
6	
7	

Digit	Samples
8	
9	
Delimiter	