



National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

AVIS

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

If pages are missing, contact the university which granted the degree.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

The Role of the Elementary Perceiver and Memorizer (EPAM)
in Optical Character Recognition (OCR)

Siasb Radvar-Zanganeh

A Thesis
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

December 1994

© Siasb Radvar-Zanganeh, 1994



National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

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

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

ISBN 0-612-10888-0

Name SEASB RADVAR-ZANGANEH

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

Computer Science

SUBJECT TERM

0984

SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523

Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language 0679
 General 0289
 Ancient 0290
 Linguistics 0291
 Modern 0291
Literature 0401
 General 0294
 Classical 0295
 Comparative 0297
 Medieval 0298
 Modern 0316
 African 0591
 American 0305
 Asian 0352
 Canadian (English) 0355
 Canadian (French) 0593
 English 0311
 Germanic 0312
 Latin America 0315
 Middle Eastern 0313
 Romance 0314
 Slavic and East European

PHILOSOPHY, RELIGION AND THEOLOGY

Theology 0422
Religion 0318
 General 0321
 Biblical Studies 0319
 Clergy 0320
 History of 0322
 Philosophy of 0469

SOCIAL SCIENCES

American Studies 0323
Anthropology 0324
 Archaeology 0326
 Cultural 0327
 Physical 0310
Business Administration 0272
 General 0770
 Accounting 0454
 Banking 0338
 Management 0385
Canadian Studies 0501
Economics 0503
 General 0505
 Agricultural 0508
 Commerce-Business 0509
 Finance 0510
 History 0511
 Labor 0358
 Theory 0366
Folklore 0351
Geography 0578
Gerontology 0579
History 0578
 General

Ancient 0579
Medieval 0581
Modern 0582
Black 0328
African 0331
Asia, Australia and Oceania 0332
Canadian 0334
European 0335
Latin American 0336
Middle Eastern 0337
United States 0585
History of Science 0398
Law 0615
Political Science 0616
 General 0617
 International Law and Relations 0814
 Public Administration 0452
Recreation 0626
Social Work 0627
Sociology 0938
 General 0631
 Criminology and Penology 0628
 Demography 0629
 Ethnic and Racial Studies 0630
 Individual and Family Studies 0700
 Industrial and Labor Relations 0344
 Public and Social Welfare 0709
 Social Structure and Development 0999
 Theory and Methods 0453
Transportation 0453
Urban and Regional Planning 0453
Women's Studies

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture 0473
 General 0285
 Agronomy 0475
 Animal Culture and Nutrition 0476
 Animal Pathology 0359
 Food Science and Technology 0478
 Forestry and Wildlife 0479
 Plant Culture 0480
 Plant Pathology 0817
 Plant Physiology 0777
 Range Management 0746
 Wood Technology 0306
Biology 0287
 General 0308
 Anatomy 0309
 Biostatistics 0329
 Botany 0353
 Cell 0369
 Ecology 0793
 Entomology 0410
 Genetics 0307
 Limnology 0317
 Microbiology 0416
 Molecular 0433
 Neuroscience 0821
 Oceanography 0778
 Physiology 0472
 Radiation 0786
 Veterinary Science 0760
 Zoology 0760
Biophysics 0786
 General 0760
 Medical

Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences 0566
 General 0300
 Audiology 0992
 Chemotherapy 0567
 Dentistry 0350
 Education 0769
 Hospital Management 0758
 Human Development 0982
 Immunology 0564
 Medicine and Surgery 0347
 Mental Health 0569
 Nursing 0570
 Nutrition 0380
 Obstetrics and Gynecology 0354
 Occupational Health and Therapy 0381
 Ophthalmology 0571
 Pathology 0419
 Pharmacology 0572
 Pharmacy 0382
 Physical Therapy 0573
 Public Health 0574
 Radiology 0575
 Recreation

Speech Pathology 0460
Toxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry 0485
 General 0749
 Agricultural 0486
 Analytical 0487
 Biochemistry 0488
 Inorganic 0738
 Nuclear 0490
 Organic 0491
 Pharmaceutical 0494
 Physical 0495
 Polymer 0754
 Radiation 0405
Mathematics 0605
Physics 0986
 General 0606
 Acoustics 0608
 Astronomy and Astrophysics 0748
 Atmospheric Science 0607
 Atomic 0798
 Electronics and Electricity 0759
 Elementary Particles and High Energy 0609
 Fluid and Plasma 0610
 Molecular 0752
 Nuclear 0756
 Optics 0611
 Radiation 0463
 Solid State 0346
Statistics 0984
Applied Sciences

Applied Mechanics 0984
Computer Science

Engineering 0537
 General 0538
 Aerospace 0539
 Agricultural 0540
 Automotive 0541
 Biomedical 0542
 Chemical 0543
 Civil 0544
 Electronics and Electrical 0348
 Heat and Thermodynamics 0545
 Hydraulic 0546
 Industrial 0547
 Marine 0794
 Materials Science 0548
 Mechanical 0743
 Metallurgy 0551
 Mining 0552
 Nuclear 0549
 Packaging 0765
 Petroleum 0554
 Sanitary and Municipal 0790
 System Science 0428
Geotechnology 0796
Operations Research 0795
Plastics Technology 0994
Textile Technology

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451



CONCORDIA UNIVERSITY
Division of Graduate Studies

This is to certify that the thesis prepared

By: **Siasb Radvar-Zanganeh**

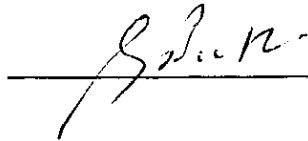
Entitled: **The Role of the Elementary Perceiver and Memorizer (EPAM)
in Optical Character Recognition (OCR)**

and submitted in partial fulfillment of the requirements for the degree of

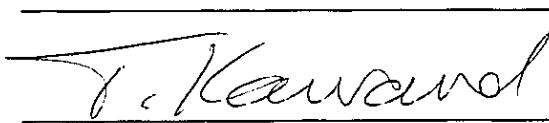
Master of Computer Science

complies with the regulations of this university and meets the accepted
standards with respect to originality and quality.

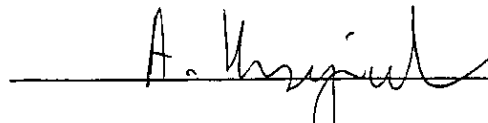
Signed by the final examining committee:



Chair



Examiner




Examiner

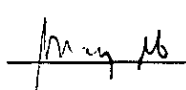



Supervisor

Approved by



~~Chairman of Department of~~ Graduate Program Director

 19 ⁹⁵



Dean of Faculty

ABSTRACT

The Role of the Elementary Perceiver and Memorizer (EPAM) in Optical Character Recognition (OCR)

Siasb Radvar-Zanganeh

Elementary Perceiver And Memorizer (EPAM) is one of the few existing general models of human perception. It has been successful in simulating and describing many human cognition phenomena. In this thesis, the role of EPAM is studied in terms of an OCR machine. The model is described in some detail. Its strengths and shortcomings have been pointed out. Some suggestions are provided for its shortcomings. Two of the suggestions are implemented and tested in this thesis. In EPAM, objects of any complexity level could be described. In this experiment the primitive objects of the EPAM model are defined as characters, and words of the English language are defined as the upper bound on the complexity of the objects. In this way the model is tested under: 1) One level of object complexity. 2) A large base of objects with a variable length feature set. As such, to test the modification's effect on the behaviour of the model, EPAM is used as a postprocess to a character segmentation and recognition subsystem. The experiment is tested with 30 fonts of different typefaces with sizes ranging between 7 points to 12 points. For the segmentation subsystem a line, word, and character segmenter is provided. A Feature extraction methodology and a recognition model are also provided for the character recognition subsystem. EPAM is trained with 23692 words and tested with the results of the character

recognition subsystem that recognizes characters within words written in the testing font set.

Dedicated to my Parents and my brother

Cyrus Radvar-Zanganeh

Farideh Khazaei

Sepher Radvar-Zanganeh

whom I love.

Without your support, patience, and encouragement this was impossible.

Acknowledgements

I cannot put in words the gratitude that I have for my supervisor Professor C. Y. Suen who has helped me throughout my graduate studies, from my acceptance into the program to helping me with this thesis. Without his teachings, guidance, encouragement and support I could not have completed this work. I feel privileged and honoured to have been Dr. Suen's student.

I am grateful to Dr. H. A. Simon of Carnegie-Mellon University for providing me with needed references and clarifying my view on various points regarding EPAM.

I would also like to thank Dr. Pervez Ahmed as my first teacher in pattern recognition, whom without his wonderful teaching I would not have discovered such an exciting field.

I also like to thank my professors Dr. A. Kryzack and Dr. T. Kasvand from whom I learned a lot in pattern recognition and image processing.

Last but not least I like to thank Aileen, Angie, Halina, Pam, and Mary for their wonderful help throughout my studies as a graduate student.

Epigram

Things which we see are not by themselves
what we see ... it remains completely
unknown to us what the objects may be by themselves
and apart from the receptivity of our senses.
We know nothing but our manner of perceiving them ...

Immanuel Kant

Table of Contents

| | |
|--|-----|
| List of Figures | xi |
| List of Tables | xii |
| 1. Introduction | 1 |
| 1.1 Why EPAM | 1 |
| 1.2 Overview | 4 |
| 2. The EPAM model | 5 |
| 2.1 History | 5 |
| 2.2 The model | 7 |
| 2.2.1 Introduction | 7 |
| 2.2.2 Discrimination Net and Recognition Process | 8 |
| 2.2.2.1 knowledge Representation | 8 |
| 2.2.2.2 The Net | 9 |
| 2.2.2.3 The Recognition Process | 13 |
| 2.2.3 Learning Processes | 15 |
| 2.2.3.1 Familiarization | 16 |
| 2.2.3.2 Discrimination | 17 |
| 2.2.3.3 Noticing order | 22 |
| 2.2.3.4 Semantic memory | 22 |
| 2.3 The evidence for and against | 22 |
| 2.3.1 The evidence for EPAM | 23 |
| 2.3.1.1 The serial position effect | 23 |
| 2.3.1.2 Constant fixation time | 23 |
| 2.3.1.3 One-trial versus continuous learning | 24 |
| 2.3.1.4 Oscillation and retroactive inhibition | 24 |
| 2.3.1.5 Effects of familiarity and meaningfulness | 25 |
| 2.3.1.6 Effects of similarity | 25 |
| 2.3.1.7 Expert pattern perception | 26 |
| 2.3.1.8 Other phenomena | 27 |
| 2.3.2 The evidence against EPAM | 28 |
| 2.3.2.1 Sensitivity to discriminativeness of features | 28 |
| 2.3.2.2 Sensitivity to missing or incorrect properties ... | 29 |
| 2.3.2.3 Multiple knowledge domains and network growth | 30 |
| 2.3.2.4 Time latencies | 30 |
| 2.4 Other models of cognition | 31 |
| 2.5 Suggested Extensions | 33 |
| 2.5.1 The redundant paths | 33 |
| 2.5.2 The generalisations and their use in the architecture | 34 |

| | |
|---|----|
| 2.5.3 A physiological look at the network | 35 |
| 3. EPAM in an OCR system | 36 |
| 3.1 An OCR system | 36 |
| 3.1.1 Text segmentation | 38 |
| 3.1.2 Character segmentation | 38 |
| 3.1.3 Word formation | 39 |
| 3.1.4 Character recognition | 39 |
| 3.1.5 Contextual verification | 39 |
| 3.1.5.1) Markov methods | 39 |
| 3.1.5.2) Dictionary look-up techniques | 40 |
| 3.1.5.3) Hybrid methods | 42 |
| 3.2 Use of EPAM as a contextual verifier | 42 |
| 4. An example OCR system | 44 |
| 4.1 The purpose | 44 |
| 4.2 The experiment | 45 |
| 4.3 The generation of data processed | 48 |
| 4.3.1 Fonts used | 48 |
| 4.3.2 Generation of raster images of the fonts | 51 |
| 4.3.3 Segmentation of Lines, words, and characters and generation of an ASCII representation file for each font's raster file | 52 |
| 4.3.4 Generation of recognition map files | 58 |
| 4.3.5 The dictionary | 60 |
| 4.3.6 Generation of mapped dictionary | 60 |
| 4.4 The character recognition subsystem | 62 |
| 4.4.1 The feature extraction | 62 |
| 4.4.2 The training | 64 |
| 4.4.3 The recognition | 64 |
| 4.5 The EPAM model as a contextual verifier | 66 |
| 4.5.1 Implementation | 66 |
| 4.5.2 Training | 68 |
| 4.5.3 EPAM search behaviour | 68 |
| 4.5.4 The modified search algorithm | 70 |
| 4.6 The results | 73 |
| 4.6.1 Results of the character recognition subsystem | 73 |
| 4.6.1.1 Tests scenarios (cases) | 73 |
| 4.6.1.2 Results of the cases where training and testing sets were the same (Using the "mean" statistic) | 75 |
| 4.6.1.3 Results of the cases where training and testing sets were two different sets (Using the "mean" statistic) | 94 |

| | |
|---|-----|
| 4.6.1.4 Results of the cases where training and testing sets were the same (Using the "median or the "standard deviation" statistics) | 107 |
| 4.6.1.5 Analysis of the results of the character recognition subsystem | 112 |
| 4.6.2 Results of the modified EPAM word search | 113 |
| 4.6.2.1 Analysis of the results of the modified search . | 114 |
| 5. Conclusion | 116 |
| References | 119 |
| Appendix A (The fonts used in the experiment) | 129 |
| Glossary | 135 |

List of Figures

| | |
|--|----|
| Fig. 2.1 The EPAM architecture | 7 |
| Fig. 2.2 A node testing a property | 10 |
| Fig. 2.3 A node testing a subobject | 10 |
| Fig. 2.4 A discrimination net for letters and words | 11 |
| Fig. 2.5 Addition of a new branch at a test node (breath growth) | 19 |
| Fig. 2.6 Addition of a new test node (depth growth) | 20 |
| Fig. 2.7 Net growth on the NEC branch | 21 |
| Fig 3.1 A general text recognition flow chart | 37 |
| Fig 4.1 The Character Recognition Subsystem | 46 |
| Fig 4.2 The EPAM Subsystem | 47 |
| Fig. 4.3 Content of a Raster File (Times Font) | 52 |
| Fig. 4.4 Connected Components | 54 |
| Fig. 4.5 A Minmax box | 55 |
| Fig. 4.6 Line determination (Transition between connected components) ... | 55 |
| Fig. 4.7 i's and j's in lines | 56 |
| Fig. 4.8 A portion of the content of an ASCII segmented file | 57 |
| Fig. 4.9 Content of a map file | 59 |
| Fig 4.10 A mapped dictionary's records. See Fig. 4.9 for correlation | 61 |
| Fig. 4.11 A character's features | 62 |
| Fig. 4.12 Simplification of feature space | 63 |
| Fig. 4.13 A flow diagram for the EPAM subsystem | 67 |
| Fig. 4.14 Behaviour of EPAM under concatenated words | 69 |
| Fig. 4.15 The alternate paths from a test node | 71 |

List of Tables

| | |
|--|-----|
| Table 4.1 The fonts used in the experiment | 50 |
| Table 4.2 Distribution of the fonts used | 51 |
| Table 4.3 Test cases for character recognition subsystem | 73 |
| Table 4.4 case_1, cond_1 | 76 |
| Table 4.5 case_1, cond_2 | 77 |
| Table 4.6 case_1, cond_3 | 78 |
| Table 4.7 case_1, cond_4 | 79 |
| Table 4.8 case_1, cond_5 | 80 |
| Table 4.9 case_1, cond_6 | 81 |
| Table 4.10 case_2, cond_1 | 82 |
| Table 4.11 case_2, cond_2 | 83 |
| Table 4.12 case_2, cond_3 | 84 |
| Table 4.13 case_2, cond_4 | 85 |
| Table 4.14 case_2, cond_5 | 86 |
| Table 4.15 case_2, cond_6 | 87 |
| Table 4.16 case_3, cond_1 | 88 |
| Table 4.17 case_3, cond_2 | 89 |
| Table 4.18 case_3, cond_3 | 90 |
| Table 4.19 case_3, cond_4 | 91 |
| Table 4.20 case_3, cond_5 | 92 |
| Table 4.21 case_3, cond_6 | 93 |
| Table 4.22 case_4, cond_1 | 95 |
| Table 4.23 case_4, cond_2 | 96 |
| Table 4.24 case_4, cond_3 | 97 |
| Table 4.25 case_4, cond_4 | 98 |
| Table 4.26 case_4, cond_5 | 99 |
| Table 4.27 case_4, cond_6 | 100 |
| Table 4.28 case_5, cond_1 | 101 |
| Table 4.29 case_5, cond_2 | 102 |
| Table 4.30 case_5, cond_3 | 103 |
| Table 4.31 case_5, cond_4 | 104 |
| Table 4.32 case_5, cond_5 | 105 |
| Table 4.33 case_5, cond_6 | 106 |
| Table 4.34 case_1, cond_1 (Median) | 108 |
| Table 4.35 case_1, cond_5 (Median) | 109 |
| Table 4.36 case_1, cond_1 (Standard deviation) | 110 |
| Table 4.37 case_1, cond_5 (Standard deviation) | 111 |
| Table 4.38 Recognition results for the modified search algorithm | 113 |

1. Introduction

1.1 Why EPAM

During the eight years that I have been studying in the field of pattern recognition, I have seen many different methodologies and endless applications of the methodologies. However, none has interested me more than the *Elementary Perceiver And Memorizer* (EPAM) theory. It has successfully exhibited many human recognition phenomena, from serial position effect in verbal learning to explanation of *intuition* [1]. The most interesting aspect of EPAM that has attracted me to experiment with this theory was its notion of homogeneous knowledge representation. It represents the "external" knowledge, stored "internal" knowledge, and the mechanism of access¹ in the same format. The unit of knowledge is an *object*. Objects are defined as complex objects having an ordered list of subobjects or a set of properties², all of which could be called *features*. This

¹The access is not modeled as such by the originators, although it could very well be, as it is described in the end of the section 2.2.2.2.

²This description is similar but not as expressive to the one given by F. Bancelhon and S. Khoshafian in "A calculus for complex objects" [2].

powerful and generic representation, free from physical constraints³, is a strong candidate to be used in different applications of pattern recognition.

The access mechanism is a self organizing network of objects, which discriminate the stimulus object and recognize it as familiar chunks of objects. The stimulus object is first recognized as whole, if unsuccessful, then the process follows by trying to recognize the stimulus object as chunks of familiar objects. Similar chunking process was also used in a very interesting model: Recognition-by-components (RBC) in [4] for recognition of objects from images by humans.

The current implementation of the access network is similar to a discrimination tree, where local decisions will sort the object to a certain category uniquely describing the stimulus; with the extension, that it might recognize the object as a set of categories, which collectively describe the stimulus object.

A comprehensive account of discriminating decision trees is given in a doctoral thesis by Wang in [5] where decision trees are successfully applied to classify objects in a large set of classes. However, a note must be made about the difference between classification and recognition. People may not recognize an object, however they may be able to classify it. This is the case with new objects that have not been seen before, which may be recognized as a collection of previously known objects, each of which could be a generalization (a class) of a

³This is not the case with neural networks. The physical constraint may limit an explanation of phenomena where otherwise is expressible. An example is given in a book by J. R. Anderson (chapter 4 pp. 144) [3] on the PDP model. More reasons are also included in the section 2.4. of this thesis.

set of objects⁴.

The expressive representation, and a successful explanation of human reading phenomena by EPAM, and the good capabilities of decision trees shown by Wang, interested me to study this model further and experiment with it in an application such as Optical Character Recognition (OCR). EPAM is a theory, and like other theories it is subject to change and evolution. In this thesis, I set out to experiment with this model and bring out its applicability, and provide some possible methods to overcome the shortcomings.

In the recursive architecture of EPAM one can define the objects of the network to be of any complexity. I choose the primitives to be characters and the upper bound on the complexity of the objects to be the words in the English language; primarily because the system is easier to study and modify, and secondly to test its capabilities as a word recognizing assistant in an OCR machine, and thirdly to test it in a large base of possible objects. However, this was purely a choice and nothing inhabited me from defining the primitives to be features of the characters, or the upper bound to be phrases, sentences, or concepts.

⁴Note that the collection could be of one or many objects, which each could be an object representing a class (a generalization) of other objects. For example, if the stimulus object is: (car, porchse, pink, doors), it will most probably be recognized as three objects: (car, porchse), (pink), and (door).

1.2 Overview

Chapter 2 describes the EPAM model, its structure, processes, and evolution. It also gives positive evidence for the model as well as its shortcomings. Some suggestions are given at the end of the chapter for further enhancements to the model and its processes. In chapter 3, a general structure of an OCR system is described along with EPAM's role as part of it in the experiments performed for this thesis. Chapter 4, describes the processes and data involved in the experiments of this thesis. Chapter 5, provides conclusion taken from the experiments and possible future directions for enhancements.

2. The EPAM model

2.1 History

The *Elementary Perceiver And Memorizer* (EPAM) model was first introduced around 1960 by Edward A. Feigenbaum [6, 7]. It later flourished by him and Herbert A. Simon[8, 9, 10]. The original model was based on a binary tree architecture, which did not simulate some aspects of human verbal learning behaviour. Authors introduced EPAM III between 1962-1964 [11, 12] which possessed an n-ary tree architecture. Between the late 1960s and the early 1970s Simon et al. have used EPAM in conjunction with describing perceptual processes in problem solving, and perception in chess and its required memory [13, 14, 15]. In 1984, Simon and Feigenbaum published a paper [16] to shed some light on the misunderstood properties of their model, in which a new branch was introduced as Not Elsewhere Classified (NEC) branch. This branch allowed the objects to be sorted at a test-node, although, the object was either missing the feature that is being tested or has a form of the feature that has not been seen before. In 1984, Howard B. Richman and Simon published a paper [17] comparing EPAMIII to

the connectionist *Interactive Activation Model* (IAM) of McClelland & Rumelhart [18, 19]⁵. In that paper, they have redemonstrated the capabilities of EPAMIIIA in simulating many human perception phenomena in reading of words, which IAM has also simulated. Both systems performed well. The success of IAM was attributed to its top-down feedback of information from larger chunks to smaller ones (*back-propagation*), particularly from words to letters. The success of EPAM was attributed to both perceiving the whole words as chunks and subwords sequence of letters as chunks. Both, the top-down feedback of IAM and the chunking of EPAM, allow the respective models to incorporate subwords into larger words. However, in some aspects, EPAMIIIA data matched much closer to the human data than did IAM⁶. The close fit of data according to the authors was the result of the system trying to recognize the whole word as a chunk first, but failing that, it tried to recognize parts of the word as chunks. EPAM, solely or in conjunction with other subsystems, has been used to describe or show many human cognition phenomena, such as Insight and intuition, "AHA" phenomena, understanding, problem solving, and scientific discovery [1, 17, 21, 22, 23].

⁵This model is more comprehensively explained in the two books "*Parallel Distributed Precessing*" (PDP) [20].

⁶The close fit of a model to human data is not a sufficient or necessarily proper way of comparing theories as explained in the section "Testing theories" in [17].

2.2 The model⁷

2.2.1 Introduction

EPAM was one of the first computer models of human information processing. It regards people as manipulators of chunks of information. EPAM undertakes to model a major portion of the system that allows humans to recognize and to learn to recognize *stimuli*, and thereby to gain access to information related to these stimuli that are stored in the *long-term memory*. The use of the term "portion" is due to EPAM modelling of only a component of the entire system, although a principal one. EPAM is taken as the middle subsystem between the sensory subsystem and the semantic long-term memory (See Fig. 2.1).

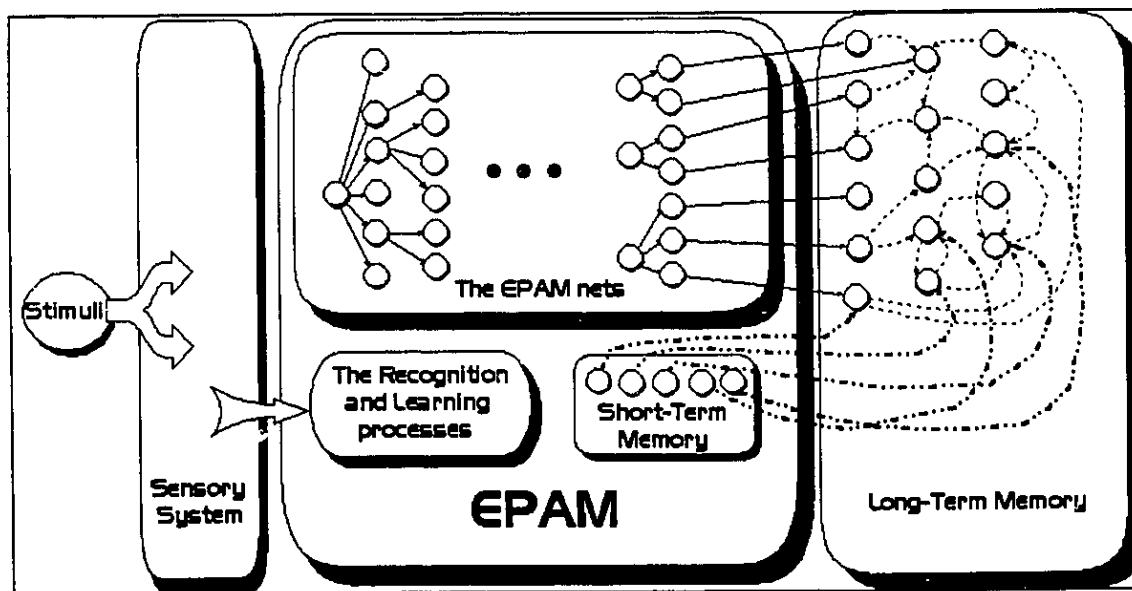


Fig. 2.1 The EPAM architecture.

⁷The information in this section was mostly extracted from the two papers [16, 17]. Some parts may have slight terminology differences with the original papers. This was done to keep a uniform terminology (my view) throughout this thesis.

It assumes a feature set provided by the sensory system (the front end) that is extracted from a stimulus. Then, it attempts to recognize the stimulus using the extracted features. A set of symbols is then put in the *short-term memory* that points to or accesses the relevant information in the long-term memory. The long-term memory is the accumulation of associated information that has been discriminated and have been learned. There has been no attempt by EPAM to model the sensory system, nor the whole structure of the semantic long-term memory and its associations. The model of short-term memory is also elementary. EPAM mostly deals with the discrimination net, stored *images* of learned stimuli, and the access process. The learning component of the model is also restricted to the modification of the discrimination net and the stored images of the stimuli. In short, according to Feigenbaum and Simon the semantic long-term memory is viewed as the "encyclopedia" in which most of the knowledge acquired by the learning system is stored; the discrimination net as the "index" to this "encyclopedia" a recognition memory; and the short-term memory as a small working memory needed by learning processes.

2.2.2 Discrimination Net and Recognition Process

2.2.2.1 knowledge Representation

EPAM uses an object-oriented methodology to represent the knowledge. This representation is homogeneously done throughout the model. Which implies, the knowledge that it learns and access has the same representation, namely the

object. The same object representation is used for external objects, stimuli, and internal objects, images. An object is an ordered list of parts called *subobjects* or a set of *properties*, all of which could be called *features*. The object itself can have properties of its own that are different from the properties of its subobjects. This representation is recursively defined, i.e., a subobject has the same representation. During learning, the image that is built may contain some but not all of the information present in the stimulus object; however, the representation is still kept the same. For example: once people have seen a person for the first time, they may remember few features of the person. However, as they see the person more often they become more familiar with that person's features. Note that even though they may know the presence of the features of the person, they may not know their specificities.

2.2.2.2 The Net

The discrimination net is an n -ary tree. The leaves of the tree are the stored images. All other nodes of the tree are test-nodes; at which are stored tests to be performed by the recognition process on the stimulus object's features. On each testing-node a feature is tested, therefore the tests are subobject tests or property tests. The branches that lead to the next level in the tree are accordingly labelled with either the (internal) name of an object or the value of a property (See Fig. 2.2, 2.3, and 2.4).

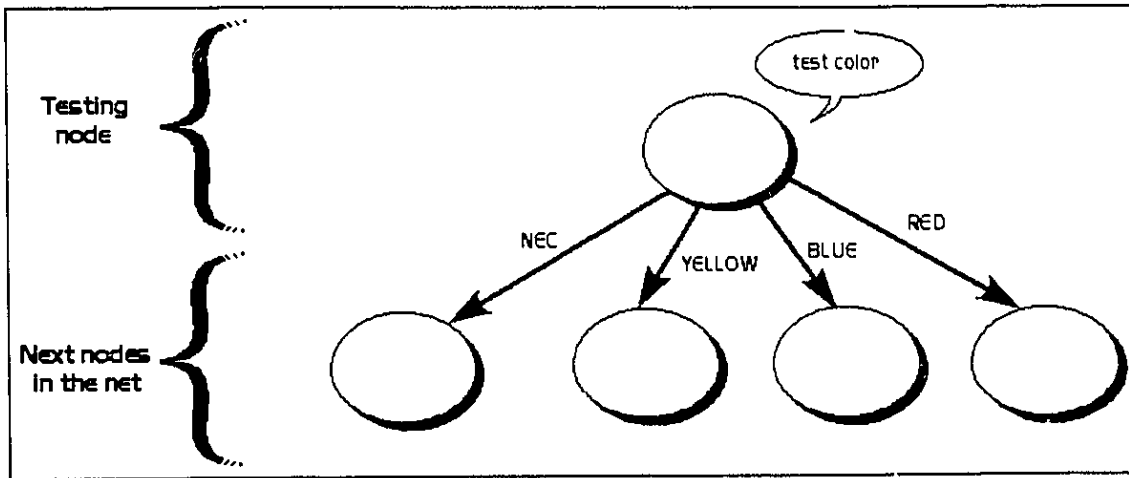


Fig. 2.2 A node testing a property.

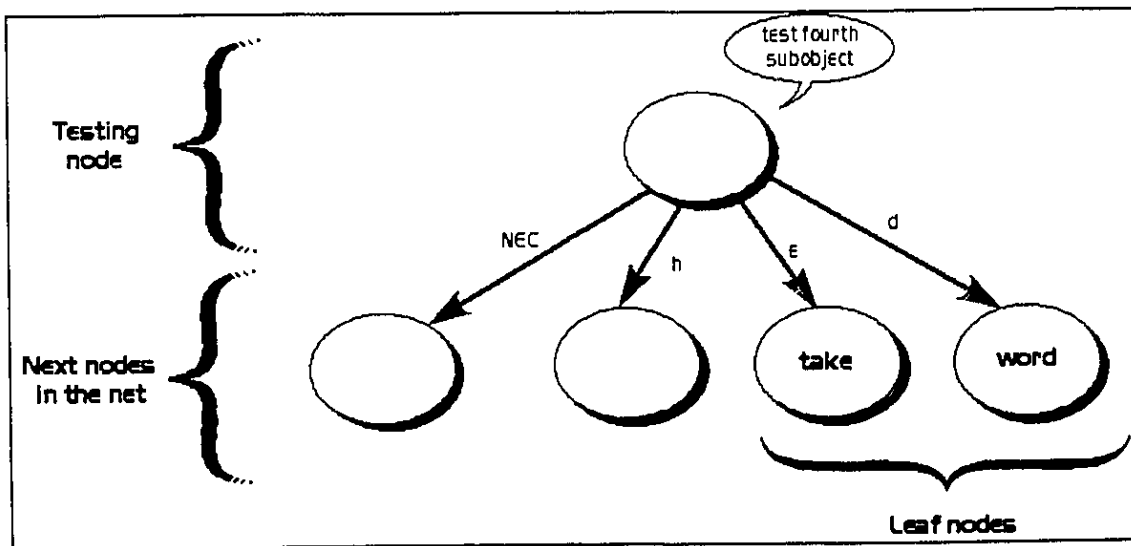


Fig. 2.3 A node testing a subobject.

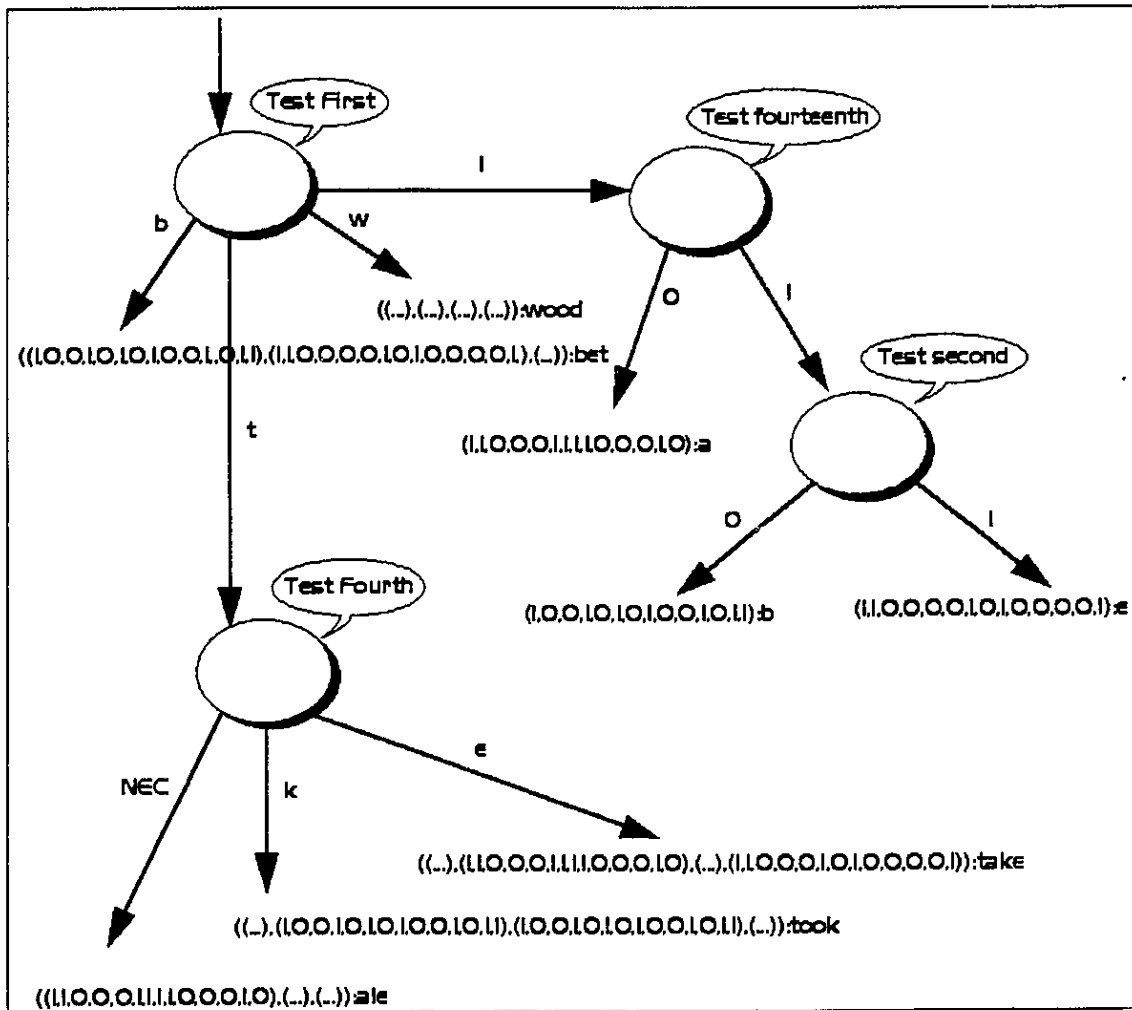


Fig. 2.4 A discrimination net for letters and words.

(The atomic features 0, and 1 are as in [18, 19])

Formally, each test node poses a one-to-one relation, $A \mathfrak{R} B$. Let the tuple (branch) $\langle a, b \rangle \in \mathfrak{R}$ such that:

Each element of the domain of the relation \mathfrak{R} , "a", is:

(1) A value of a property for a property test.

Or

(2) A complex object's internal name (a pointer value) for a subobject test.

Or

(3) An atomic object for a subobject test.

Each element of the range of the relation \mathfrak{R} , "b", is:

(1) An address (a pointer) to the next testing node.

Also included as one of the branches (elements of the relation \mathfrak{R}) at a test node is the tuple $\langle \text{NEC}, b \rangle$. When a feature is not found in the domain of the relation, then the Not Elsewhere Classified (NEC) branch (tuple) is used.

It should be noted that when a feature is searched for in the domain of the relation, if the value of the domain variable is a pointer to a complex object, then the object pointed by that pointer is used for comparison with the feature in question. Alternately, a copy of the object could have been stored at the test node during learning which then a direct comparison would have been done. Evidently

the object pointed to should be completely familiar, where otherwise the previously stored objects via the given branch may not be reachable upon further familiarization (See Section 2.2.3.1) of the pointed object. Of course, this is valid in the current model since the features of the familiar objects are not replaced by further familiarization. I will describe a scheme in section 2.5.2 to compensate for the possibility of feature evolution.

It should be noted that the nodes of the tree are objects (in an object-oriented sense) having a property (behaviour) of testing a feature of the inputted stimuli⁸.

2.2.2.3 The Recognition Process

The recognition process is used to determine if the presented stimulus is already familiar. The process uses the recursive definition of the stimulus object to recognize it. The object *O* is presented. The first test node is the root node of the tree. If the test is a property test, it is applied to the appropriate property being tested and its value determined. Its value is used to branch to the next test node using the relation available at the test node between the values and the next test nodes to be followed (See Section 2.2.2.2). A measure of similarity (distance) is used to follow the branch where the element of the domain is the closest. If a close match is not found (determined by a local and/or a global threshold) the NEC branch is followed. If the test is a subobject test, the subobject is accessed.

⁸This is not modeled by Simon and Feigenbaum as such.

If the subobject is a primitive, then it is used to branch to the next test node using the relation present at the test node. If the subobject is not a primitive, then the recognition of the object is temporarily suspended until the particular subobject is recognized so the process can continue to the next test node. The subobject itself is an object, hence the recognition starts from the root node for the subobject itself until it is recognized. This is a homogeneous process, i.e. the recognition of the subobject may be suspended until its subobjects are recognized. Again, note that the NEC branch could exist at any test node. Eventually the recognition of primitives will terminate all levels of the recursion. If an image of subobject is found, its internal name (a pointer) is used to branch to the next test node using the relation present at the test node (See Section 2.2.2.2).

There is also a tracking list kept (a stack) for the test nodes visited. This tracking list is used when a comparison is done between the object and the stored image. If the image has more subobjects than the stimuli object, then the recognition process continues by backtracking to the last test node visited and continuing by following the NEC branch from that test node. Also, if the process needs to branch via a NEC branch at a test node and there is none, the process continues by backtracking to the previous node visited and tries to branch using the previous node's NEC branch. This continues until an image is found or the tracking list is empty. This backtracking and following the NEC branch, provides for the chunking capability of EPAM.

The recognition process is not just finding some images, but ones that

match mostly to the extent of features available to the stimulus object. The match may succeed if some features of the stored images are unknown, but it will fail if features are different between the stimulus object and the images. This difference is exploited by the learning process to allow for new objects.

In EPAM IIIA, first, the whole object is sorted. If unsuccessful in finding a single image that matches the stimulus object, the process follows by first trying to recognize the left most chunks and then the right most chunks. If unsuccessful it does the reverse by first trying the left most chunks and then the right most chunks. A more comprehensive algorithm is provided in [17].

2.2.3 Learning Processes

EPAM III has two learning processes: FAMILIARIZE, and DISCRIMINATE. FAMILIARIZE is the memorization process by which the images of stored objects are gradually built up by adding features (clarifying them). DISCRIMINATE is a discrimination learning process by which the discrimination net is augmented (test nodes and branches added) to allow recognition as the number of learned items grows.

When the stimulus object becomes present to the EPAM, a search in the net is done to find the image of the stimulus object (to recognize it). Depending upon the results of the recognition, i.e., a single chunk was found or many, and the whole word parameter, and the *anchor-point* assumption the DISCRIMINATE process is activated with a combination of the chunks retrieved. The

FAMILIARIZE process is activated, either when a single chunk is retrieved during initial recognition and needs to be elaborated or through DISCRIMINATE process when an image is not found and a new leaf node needs to be created.

For more details of different cases for activating the DISCRIMINATE process refer to [17].

2.2.3.1 Familiarization

Familiarization starts after an object is searched for in the net (see the parent section).

- If no image is found, an initial image is created which gets a copy of the features of the stimulus object. The features are copied to image governed by a *familiarization parameter*, a measure of attention and learning capability. Going from first, to the last feature, this parameter is used in learning (copying) the features. For each feature, a random number is generated and tested against the familiarization parameter. A feature is copied to image if a random number generated is above this parameter. If a random number is found below this parameter the rest of the features are marked not noticed except the last feature, which is tested against a new random number for it to be learned (copied onto the image).

- If an image is found, the number of features is compared. If they are not the same the process is exited. If they are the same, for each feature that are marked not noticed, a random number is compared against the familiarization

parameter to determine learning (copying the feature onto the image). If a random number is below the familiarization parameter, then the process continues by finding the next familiar feature on the image and processes the previous feature to that one⁹. This continues until no more features are left to process. Note that after many sessions of familiarization, the learning of an object is more likely to complete. Also note that, the completion of familiarity with objects is dependent on the familiarization parameter as well.

2.2.3.2 Discrimination

Discrimination learning is done by first searching for the object through the net¹⁰. This search is different from the search for recognition in that it does not follow the NEC branch unless the object has fewer features than the test is expecting. If the search terminates at a test node, a new leaf node is created. The object is familiarized as an image on a newly created leaf node. The leaf node is then attached to the test node via a new branch (see Fig. 2.5). This branch (tuple) is added to the test node to represent the new value of the feature tested. If the search terminates at a leaf node, the object and the image are compared at that leaf node to modify the discrimination net. If a difference is found between the object and the image, the differentiating feature is used as the testing feature. A

⁹The process looks for an anchor point, either by passing the end of the image or by finding a familiarized subobject on the image. When an anchor point is found, the previous position is taken as the place where the process continues.

¹⁰Note that this object may not be the same as the one presented to EPAM originally but a combination of the chunks extracted during initial recognition.

test node is created which tests that feature. This test node will replace the leaf node. A new leaf node is created, and is familiarized to contain the image of the object. This new leaf node and the one compared are then attached to two different branches from the created test-node. The test-node is then attached to where the compared leaf node was attached previously (See Fig. 2.6). If the compared image and the object did not have the same number of features, then the one with fewer features will be attached to the NEC branch of the new test-node and the one with more features to a regular branch. In this case, the test node will be testing for the last feature of the object attached to the regular branch (See Fig. 2.7). For more details of different cases in comparison of object and image, and action in each case, refer to [17].

Since most differences are due to different values of features, the growth of the net is biased toward breath, rather than depth.

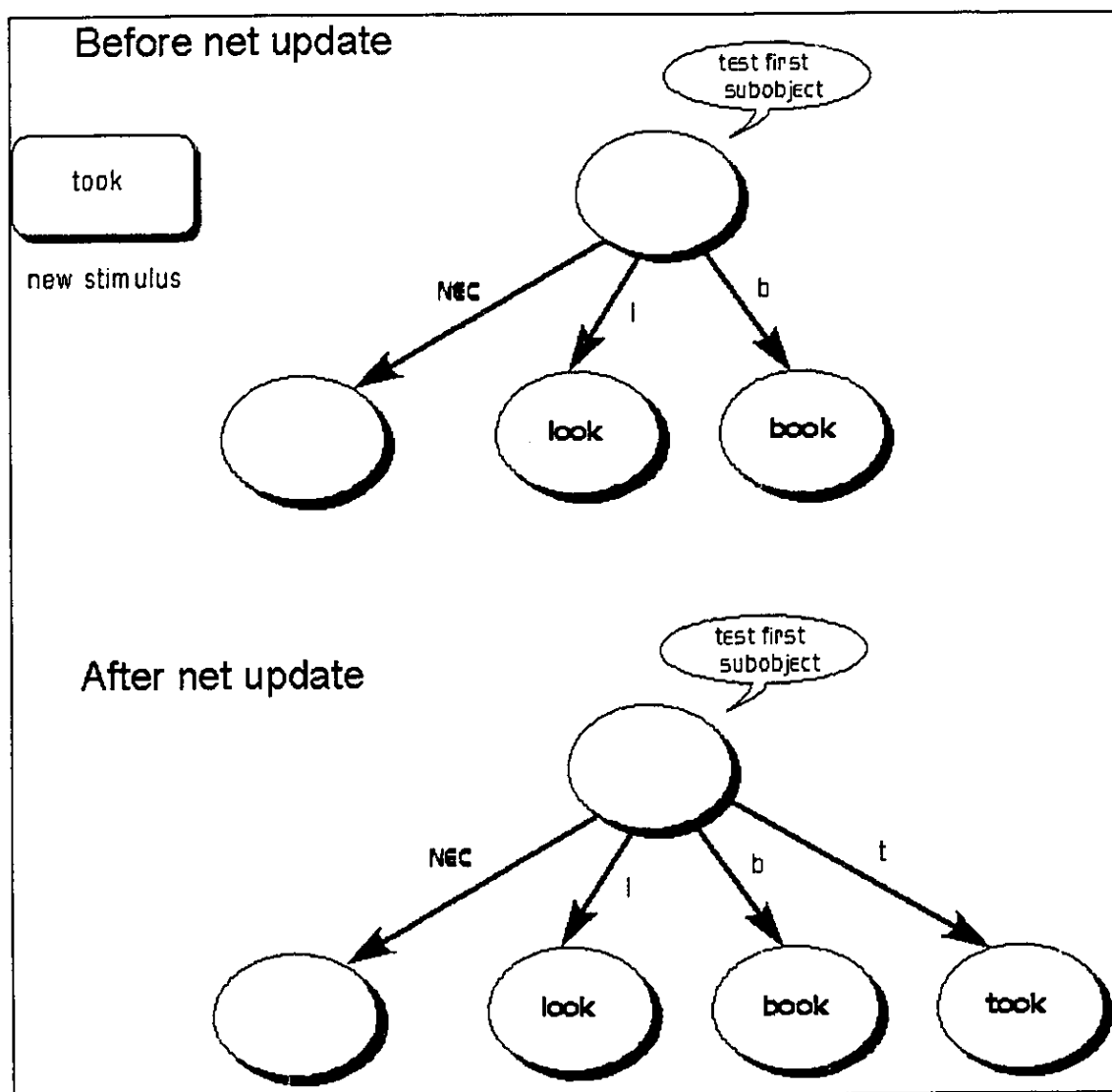


Fig. 2.5 Addition of a new branch at a test node (breath growth).

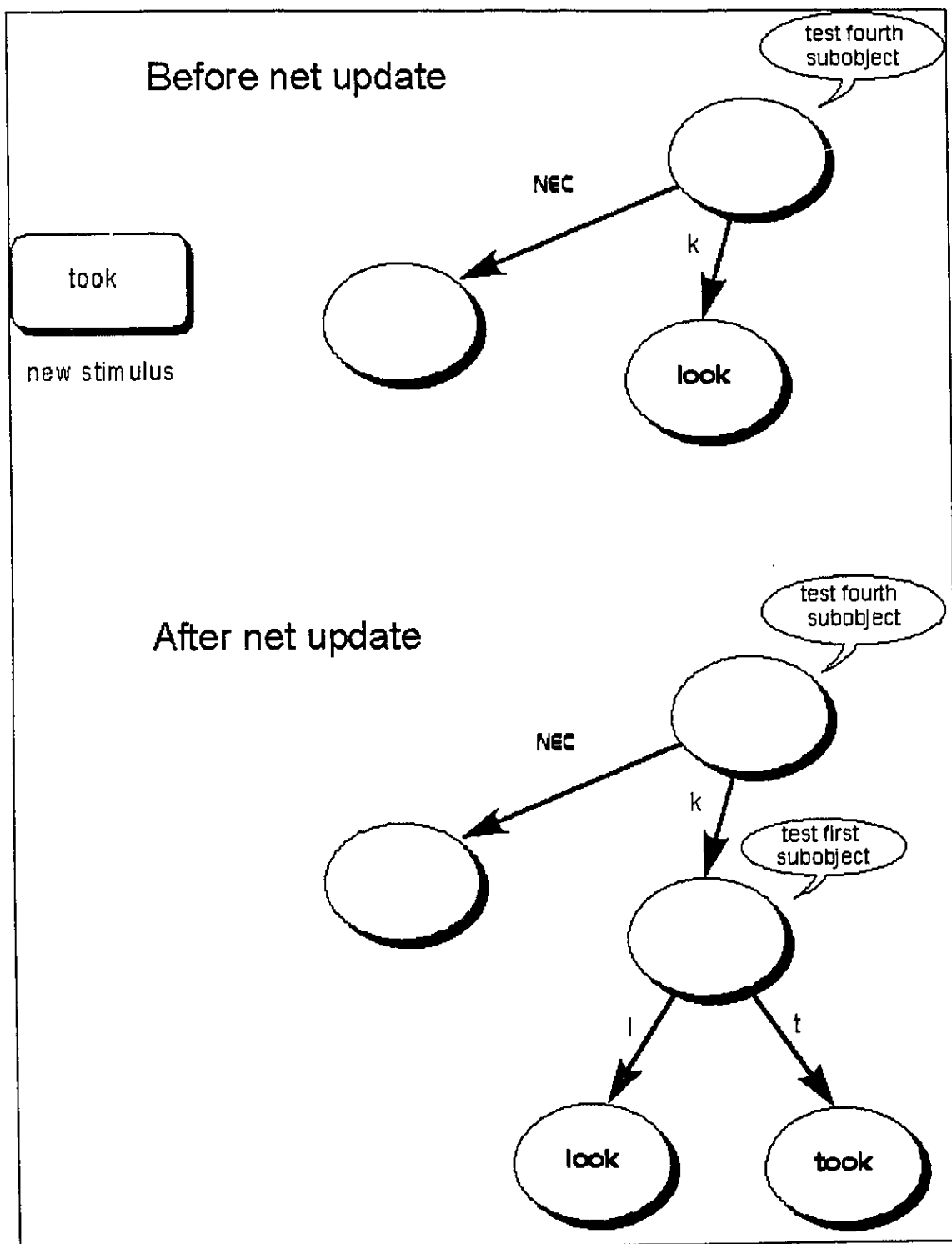


Fig. 2.6 Addition of a new test node (depth growth).

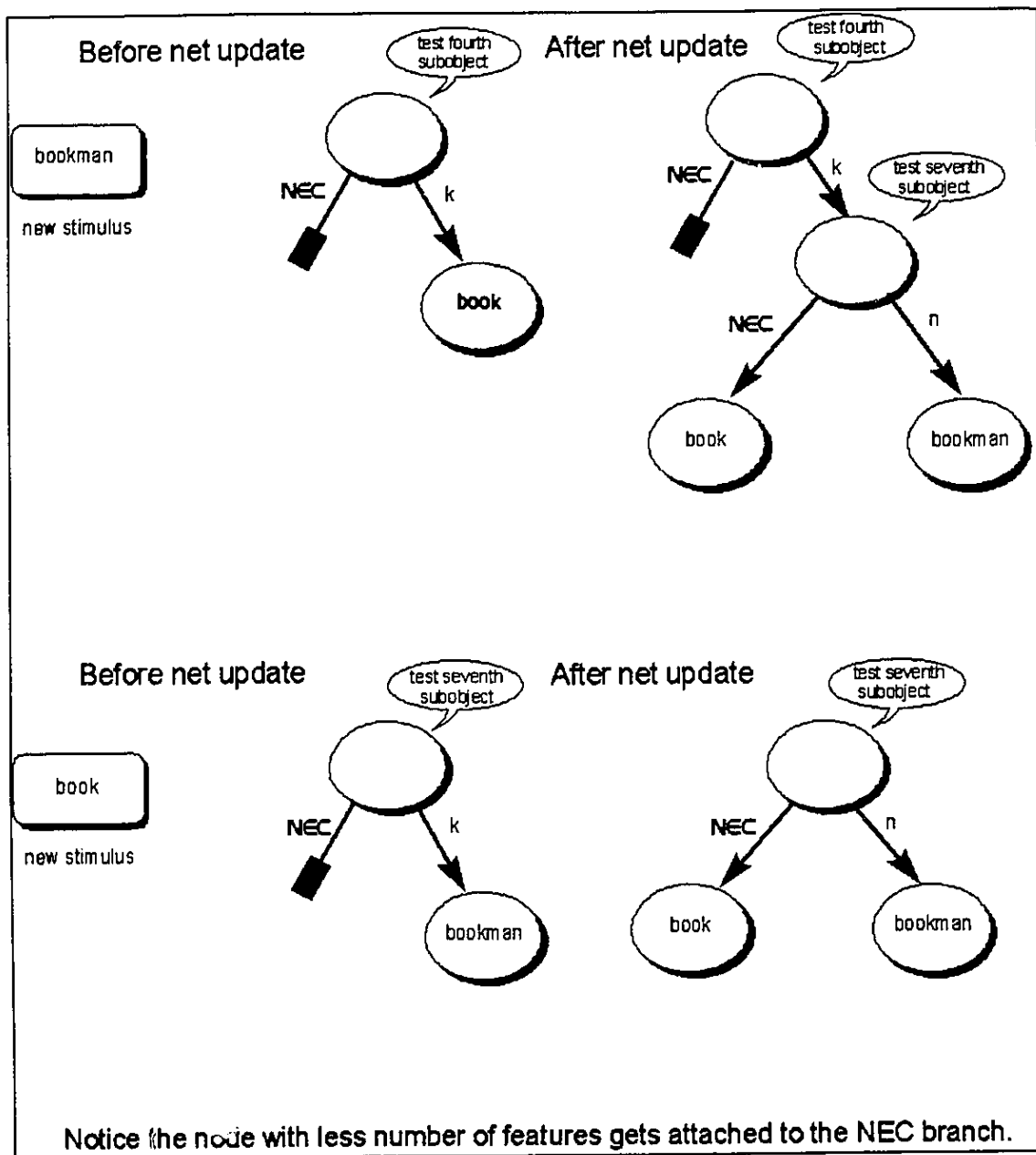


Fig. 2.7 Net growth on the NEC branch.

2.2.3.3 Noticing order

In EPAMIII and this thesis the stimuli being tested are the English words. Therefore, the noticing order is modeled through *anchor-point* assumption. The *serial-position* phenomenon [17, 24] is used as the basis for the anchor point assumption. The serial-position phenomenon institutes that people learn objects from outside points inwards. This behaviour is implemented in different processes of EPAM. The anchor point assumption is used in familiarization process which is used when an image is created or elaborated. Also, it is used as a noticing order when an image and an object are being compared to find a difference. Further, it is exploited during learning, after the initial recognition, to determine the chunk combination that is sent to the discrimination process.

2.2.3.4 Semantic memory

EPAM does not address the issue of associations between images, nor any groups of them. An example was attempted in [3]. EPAM primarily focuses on the memorization and discrimination among images.

2.3 The evidence for and against

There are seven main evidences for EPAM among many others that support its architecture and processes. These evidences are the corner stone of the EPAM theory. The biggest critique of the EPAM theory came in the paper by L. W. Barsalou and G. H. Bower [25]. In the following subsections the two facets of

the model are reiterated. For more information and references see [16, 25].

2.3.1 The evidence for EPAM

The following are the main evidences for the EPAM theory.

2.3.1.1 The serial position effect

The serial position effect was reported by McCrary and Hunter [26] and further elaborated by Feigenbaum and Simon in [21]. It has also been studied by many others including J. C. Johnston in his first of the two experiments [27, 28, 29].

This serial position effect exhibited by EPAM is due to three factors:

- 1) The attention strategy.
- 2) The anchor-point assumption.
- 3) The approximate constancy of the aggregate set of processes to recognize and to learn a new chunk in memory.

In verbal learning, any system that learns one object at a time, works from outer ends of the object (list) toward the middle of it, and requires about the same amount of work to learn an object, would exhibit this property.

2.3.1.2 Constant fixation time

The amount of fixation that is done for each new object is a function of growing the discrimination net and storing the image of the object. Total fixation

time is independent of number of trials or the time of exposure per trial. This constancy was first explicitly addressed and confirmed by B. R. Bugelski [30].

2.3.1.3 One-trial versus continuous learning

The ability of learning an object in one trial is dependent on the object's complexity. This complexity is determined by the organization of the discrimination net and the previously stored images of objects. This is possible in EPAM since the images of objects are built recursively in an organized fashion from the existing images of complex objects. This phenomenon would not be easily reproducible in a system that didn't possess such recursive organization.

2.3.1.4 Oscillation and retroactive inhibition

The oscillation and retroactive inhibition are also exhibited by EPAM. *Oscillation*, the *forgetting* and *remembering* of objects, is exhibited by EPAM due to modification of discrimination net. Objects correctly sorted may be misidentified or not identified due to further learning of objects with high similarity. This object may be remembered again upon representation and relearning. Oscillation is more evident in EPAM with redundant paths, as previously forgotten objects could be remembered if a different path from an inhibited access path is traversed upon noticing of the features in question (See section 2.5.1).

Retractive inhibition the interference of the newly learned object with the previously learned one is also exhibited in EPAM in the same way as the

oscillation is caused.

2.3.1.5 Effects of familiarity and meaningfulness

It has been found that it takes much less time to learn a meaningful object as opposed to a low meaning object. Actually it has been shown that this time is proportional to the number of familiar chunks that are associated in learning [31]. This phenomenon is exhibited in EPAM via its recursive building of complex images from other complex images previously learned. For example, a unicorn could be quickly learned simply as a horse with a horn, or, a mermaid as a woman with a fish tail. This process is dependent on the architecture of the discrimination net. For more discussion on familiarity and meaningfulness see [16, 17, 19].

2.3.1.6 Effects of similarity

It has been shown in experiments [27, 28, 29] that in 15% of the cases where the correct word was not recognized by human subjects, another word was named sharing three letters. In accordance, this statistic was calculated for EPAM as 10% of cases reporting another word sharing three similar letters [17]. Also, EPAM has shown with good qualitative agreement the effects of intralist and interlist stimulus and response similarity [19].

2.3.1.7 Expert pattern perception

Using EPAM and PERCEIVER [13], Simon and K. Gilmartin constructed Memory Aided Pattern Perceiver (MAPP) [15] which was able to simulate as well as explain the performance of a good chess player vs. a novice one. Experts not only have learned much more patterns than novice people, but also these patterns tend to be of complex compound patterns. This is provided in EPAM via its capability to store complex patterns and to store a lot of them while being able to retrieve these patterns very fast. The issue of time latencies of experts vs. novice people is also attributed to:

- 1) Experts possess and can find complex objects in one search, whereas novices have to retrieve the object as a collation of many simpler known objects.
- 2) By providing shortcut redundant-paths (See Section 2.5) to an object, an expert may be able to retrieve an object much faster than a novice not possessing such an access path.
- 3) If the recognition is accomplished via an indirect association in the long-term memory, then experts may possess shorter paths to an object than novices.

2.3.1.8 Other phenomena

Word-Superiority effect

This phenomenon that was first discovered by Cattell [32] basically states that more letters could be identified within words than in isolation. EPAM has exhibited this property very well [17].

Ineffectiveness of Letter Constraints

Johnston showed [27, 28, 29] that contrary to previous belief, people do not recognize the letters in words better when there are fewer possible choices for that letter in the existing words of the target language. For example, the S in SINK with many similar possibilities (wink, rink, link, kink, . . .) was not harder to recognize than the S in SHIP with two other possibilities (whip, chip).

Word-Letter effect

This phenomenon that was discovered by Reicher [33], states that under some conditions letters in words are perceived better than in isolation. This effect was simulated by EPAM and the results matched closely to the Johnstons human data [17].

2.3.2 The evidence against EPAM

The most elaborate critique of EPAM model and similar systems came in the paper by L. W. Barsalou and G. H. Bower [25]. Most of the issues were answered back in the same journal by Feigenbaum and Simon [16]. Most concerning issues were with the older version of EPAM (EPAMII). And the issues did not hold any longer with the new architecture. However, there were few issues that were central or valid which should be mentioned.

2.3.2.1 Sensitivity to discriminativeness of features

It was argued that in the discrimination nets, tests are built not necessarily using the most discriminating features of the stimuli but using some arbitrary order. As a result, in learning, the most discriminating features of an object may not be used for a test. Hence, during the search, the most prominent features of an object may not be tested at all. Further, the organization of these tests does not change upon new evidence (some new features that are more discriminating that were not detected originally or values of some features which upon closer investigation are found different from the original). There are few answers to this issue. 1) The noticing order of reading of words has been found to follow the properties of the serial-position effect (See Section 2.3.1.1). 2) Redundant paths to objects overcome this issue. The best work in this regard has been by Janet L. Kolodner [34]. Redundant paths facilitate different access and test sequence to get to an object. It should be noted that one does not remember an object necessarily

in one way, be it the least efficient or the most efficient way. EPAMIII has redundancy in the form of the NEC paths. This redundancy has shown promising features; however, I believe there can be better results if we use a hybrid of the two models put forward by Ms. Kolodner and EPAM. 3) It is not guaranteed in human (or for that matter anything else in life) things to be optimized. A human may not pick up an "obvious" distinctive feature that would discriminate that object from others in his memory and a recognition path may be built upon the information at hand. The same object may be remembered through this path until a new discriminating feature that has been detected (noticed) and can discriminate that object in another way. This path, a new way of remembering that object may be used if this feature of the object is noticed again. This new path may become more prominent than the others depending upon repeated traversal and attention strategy.

2.3.2.2 Sensitivity to missing or incorrect properties

If a feature is missing or has a wrong value the search may not converge to the object in question. In EPAMIII the effect is that the object is segmented into smaller familiar objects. Also, since tests and decisions are done locally on features and the next testing node is dependent on that test, direction of the search may be altered due to following a wrong path. This is true of any Hill-Climbing search algorithm where moving toward local maxima may not lead to the global maxima. Generally this problem could be avoided with the

introduction of redundant paths. Wang, in his doctoral thesis [5], has put forward a method of search, which is not only driven by local decisions but also global optimization is taken in count to drive the search.

2.3.2.3 Multiple knowledge domains and network growth

It has been argued by Barsalou and Bower that the nodes of the decision tree will grow exponentially as more knowledge domains are introduced. There are two answers to this point. 1) Since EPAM uses a recursive architecture and process to represent and handle knowledge, there are no duplicate redundant tests for any subobject. 2) In an EPAM net the total number of nodes grows linearly with the number of objects discriminated. In the worst case, the binary net, the total number of nodes is twice the total number of objects discriminated. In an n -ary net the total number of nodes is $n/(n-1)$ times the number of objects. If there are k redundant paths to an object, then this total would come to $k*(n/(n-1))*\text{number of objects}$. This equation is certainly of a linear order.

2.3.2.4 Time latencies

It has been argued that as people learn more, it will take longer time to recognize an object using the EPAM net. This comes from the assumption that net will grow in depth, and as there will be more levels, it will take longer time to reach the end of the net. I will quote the example given by Feigenbaum and Simon [16], which goes as this: "Suppose that $n=8$ (there are eight branches, on

average, at each node), and that each test takes 10 ms. Then seven tests (taking a total of 70 ms) will discriminate among two million stimuli ($8^7 = 2,097,152$). In the net 10 tests deep (100 ms), over 1 billion objects can be discriminated. There is no evidence of recognition speeds of experts faster than these, nor any evidence that experts can discriminate among billions of different things. Even a binary net can discriminate among a million stimuli in 200 ms." Of course, the time would be shorter if there are more direct redundant paths to the object, which experts may possess. Also, as mentioned in section 2.3.1.7 indirect recognitions through long-term memory could also be shorter for experts, which may possess shorter paths to objects than novices. I should note that in Wang's thesis [5], he found that the optimal number of branches, in terms of access speed and memory utilization, at each test node in his system is 4. This value was for recognizing Chinese characters.

2.4 Other models of cognition

There are a few other theories of cognition that possess such generality: Connectionist model of McClelland and Rumelhart [18, 19, 20], John Anderson's ACT* [3], and Rosenbloom and Newell's SOAR [35, 36, 37]. The connectionist model describes thinking without using symbols, but as patterns of activation in a network of connected structures. The model does not account for the initial feature detection phase. It has been used in many, simple to complex cognitive tasks. In the connectionist model each object is represented as a node in a

network of interconnected nodes. Basically each node is evaluated against the features of the stimulus for a measure of goodness of fit. This is provided by inhibitory and excitatory connections between the primitive nodes and the more complex nodes, and connections from complex nodes to primitive nodes. There are also connections between nodes at the same complexity level. Features of the stimulus activate the primitive nodes of the network which in turn activate the more complex nodes to which they are connected. This process is generally called propagation. Further, a back-propagation from the more complex nodes to primitive nodes is devised to incorporate the contextual effect of inclusion of the primitive features in the more complex nodes. This forward and back-propagation are generally performed for several cycles until the change in the system has "settled down". The node with the highest activation level is taken as a recognition of the stimulus. The recognition improvement (learning) is done through strengthening of the connections. To allow such system to learn new discriminations, it must distinguish between selecting the wrong node for a stimulus and selecting the same node for two or more distinct classes of stimuli. The implementation of this learning process is still unclear. ACT* has features similar to both EPAM and IAM. It employs a discrimination net that permits the search to be performed in several paths in parallel. Further, it associates weights to these paths so not all the paths are followed during the search. Generally looking at the ACT* it could be looked at as an EPAM-like system with redundant paths. ACT*'s emphasis has been on the structure and operations of the

semantic memory and particularly on the spreading of activation in performance and learning. ACT* does not provide detailed account of perception phenomena. SOAR claims the most general of all the above theories; however, it has been mainly applied in complex cognitive tasks.

2.5 Suggested Extensions

2.5.1 The redundant paths

In section 2.3.2 the need for redundant paths in the architecture of EPAM is shown. The current model of EPAM incorporates redundant paths in the form of the NEC branch. The idea basically is that if an object is not recognizable as is, the search continues by going to a node that describes that object in the more general form. This process is intuitive since if an object is not recognized we tend to recognize it as a set of more generalized objects. This form of redundant path is certainly needed in future architectures. However, the system lacks alternate paths to objects. These alternate paths could be in the form given by Ms. Kolodner in [34]. Instead of one test at a test node, multiple tests with multiple relations could be devised. Each test, would check a different set of features of the stimulus. The features to be tested could be determined dynamically from the commonality of the feature of the stimuli. Each test then, determines the difference within those common features for different stimuli. Alternate nodes as well as the NEC branch are kept in the backtracking list. The next node to follow from a test node depends upon the noticing order and frequency of activation of

a given path between two nodes. Further, instead of a single path determined by a test, multiple paths could be taken as the tests would determine sets of values instead of single results. This behaviour is evident in humans reading of written scripts, as alternate choices similar to an ambiguous character in question is postulated to arrive to recognition. Additionally, In EPAM an image is retrieved without further comparison of other features that did not get tested during the search. If a complete comparison is done between the image and the equivalent part of the stimulus object a more selective response could be made. The latter two modifications are tested together in this thesis and explained in the section 4.5.4.

2.5.2 The generalisations and their use in the architecture

Each node currently possesses tests and a relation specifying the next node to follow. Further modifications explained in the previous section brings the notion of generalization of objects. Each testing node could be looked at as the generalization of the objects that is accessed via that node. General information could be in terms of common features and level of their activation.

Further, upon creation of new nodes not only the subobjects are copied to the image, but also a pointer is kept to the generalization (class) of the subobject being copied. This would allow acknowledgement of inheritance. Further it would allow noticing of contradiction upon evolution of both the instance and the class.

2.5.3 A physiological look at the network

The method of access, as mentioned, is a network of interconnected objects. Such representation could be implemented symbolically or physically as any complex mechanism including neural nets, where an object is either a neuron or a network of them.

3. EPAM in an OCR system

3.1 An OCR system

Document handling applications such as wordprocessors, desktop publishing systems, Computer Aided Design systems (CAD), and databases are now in a wide spread usage by people. To enter the printed or written documents automatically users must resort to an *Optical Character Recognition* (OCR) system. A scanner is used to input a raster image of the document. This image can be entered and stored in most of the mentioned applications. However, to extract the symbolic information in the image (text, lines, curves, . . .) and to manipulate these symbols an OCR system is required. There are many methods of extracting these symbols from the document, such as the ones given in [38, 39, 40]. Text is generally extracted as collinear stripes of small components. Further, these text components could be processed to extract letters and words from them. A review of available methods for extraction and analysis of text could be found in [41]. A general process of text recognition is given as a flow chart in Fig. 3.1.

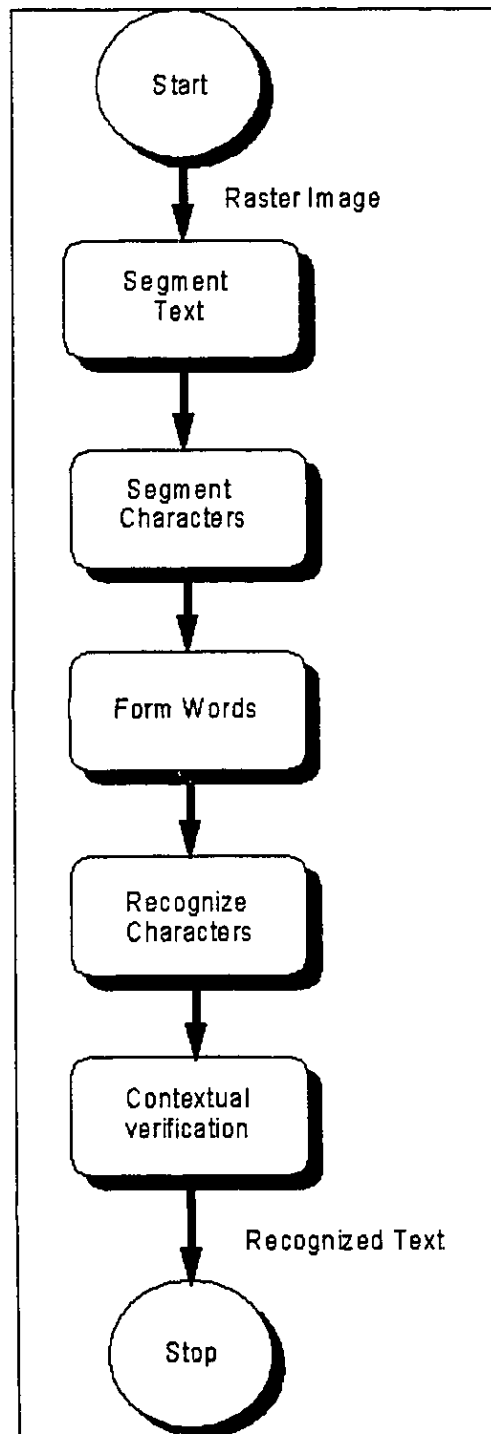


Fig 3.1 A general text recognition flow chart.

Each general process is briefly explained in the following sections. References are provided for further study.

3.1.1 Text segmentation

As mentioned in the parent of this section, text lines must be extracted from the documents in order to be further processed. Some working methods are presented in [38, 39, 40, 41].

3.1.2 Character segmentation

Most OCR systems recognize the words in the text from extracted characters present in the text. Therefore, these characters must be extracted through segmentation. A survey of character segmentation techniques is given in [42]. Segmentation of characters is performed either before recognizing those characters or in a collaboration with the recognition system. Some of the interesting works in segmenting the characters before recognition are given in [43, 44, 45, 46]. An example of interactive segmentation could be found in [47]. A preprocessing to segmentation can be applied to remove noise from the image containing the characters. This preprocessing could be in the form of a filter. Various filters and their effects are reported in [48]. Depending on segmentation technique it may require some preprocessing.

3.1.3 Word formation

Word boundaries need to be determined in order to apply contextual verification. This boundary could be determined from inter-character and inter-word spacing. Such relation can be detected much better after the characters pitch has been determined [49].

3.1.4 Character recognition

Character recognition techniques have been extensively reviewed in [50, 51, 52, 53].

3.1.5 Contextual verification

A word determined by its boundary (See Section 3.1.3), and the recognized characters, forms a string, which can be verified for error incurred during the recognition and segmentation phase. There are three approaches to verification of these generated strings.

3.1.5.1) Markov methods

These methods model the text in a given language (e.g., English) as a Markov process. As such, transition probabilities could be assigned to various letter combinations called *n-grams*. These transition probabilities are calculated from *n-grams* in a valid dictionary or from a collection of journals or books that are diversely variable in their subjects. Alternatively these probabilities could be

extracted dynamically as the text is being processed. A whole account of n-grams statistics is given by Suen [54]. Generally as the order of n-grams increases, the performance of the result is better. These statistics are generally domain dependent. As the context of text changes the results may not be as good. Several methods have been proposed that used these statistics to determine the most likely transition probabilities that would fit the string in hand, by maximizing the probability of the string. Among these methods is the *Viterbi Algorithm* [55, 56, 57, 58], the *Recursive Bayes Algorithm* [59, 60], and the *probabilistic relaxation* [61]. One of the problems of these techniques was the fact that once an error was present it compounded itself. This problem was remedied by using binary n-grams that took only two values, 1 if it's a legal n-gram, 0 otherwise.

3.1.5.2) Dictionary look-up techniques

The dictionary look-up techniques basically check the given string against the available dictionary. Simplest form of this technique checks whether the string exists as a word in the dictionary or not. In the more advanced form, it will try to find the closest possible words in the dictionary to the given string using some kind of similarity measure. The complexity of these methods is not only dependent on their algorithm but also on the storage structure of the dictionary. It was shown by Damerau [62] that most (80%) spelling errors are of the form, one letter wrong, one letter missing, one letter inserted extra, and/or two letters transposed. These are generally known as substitution, deletion, insertion, and

transposition errors respectively. Errors introduced during recognition and segmentation are basically the same as these errors. Most methods proposed assumes a subset of these errors that could have happened in the input string. There are generally two approaches to string verification: one is to measure the number of edits that is required to obtain a dictionary word from the inputted string. The other is to obtain probabilities that the input string is a dictionary word. The most known edit method is the Levenshtein distance method, which defines the similarity of the input string to the dictionary based on the total number of edits required to obtain a dictionary word from the input string [63]. This method was later modified by adding weights to edit operations to allow distances to be varied according to different applications [64, 65, 66, 67, 68]. Probabilistic methods measure the likelihood of a string to be a word in the dictionary. Among the probabilistic methods presented are the works by Hall and Dowling [69] and Kashyap and Oommen [70]. Most of the computations are done recursively which result in a much less order of operation and a faster process. An order of $O(lm + ln)$ has been claimed where l is the length of the dictionary, m the average dictionary word length, and n the length of the input string. One of the big issues in dictionary look-up techniques is the structure of the dictionary used in search. Many methods and architectures have been proposed, which narrow the search space. Among these are trie memory structures [71, 72, 73], frequency ordering [74, 75], and redundant hash addressing [76, 77].

3.1.5.3) Hybrid methods

Hybrid methods [78, 79, 80, 81, 82] use both Markov methods and dictionary look-up techniques' advantages. Usually a form of the Viterbi Algorithm is used to generate several possible hypotheses for the input string. Then these generated hypotheses (strings) are searched for in the dictionary to find them as words. If these words are not found in the dictionary, then Markov techniques are used to obtain the most probable results. These techniques allow exclusions of the high-order Markov dependencies due to verification with the dictionary, which in turn reduces the computational complexity of the process.

3.2 Use of EPAM as a contextual verifier

When I first saw the paper comparing the two theories, EPAM and IAM [17], and the phenomena explained by the two models, I got very interested to find out what makes EPAM tick¹¹. The fact that the two models tried to simulate a word recognition process to show their capabilities provoked me since I already was active in the field of character and text recognition. I set out to find the virtues of the EPAM theory. Maybe, use its feature in other models, or use other models and methods in EPAM for its shortcomings, if any. EPAM as mentioned earlier, possess recursive architecture; therefore, objects of any complexity could be described in this model. However, for me to find the model's virtues and shortcomings, it was necessary to simplify the objects being represented for the tests.

¹¹I already was familiar with the IAM model and theory.

As such, I choose the primitive objects of the system to be characters, and the more complex objects to be the words in the English language¹². With this constraint, the model essentially turned to a contextual verifier for words that have been generated by some character recognition system.

¹²The words in this simulation are chosen as the upper bound. However, there is no restriction in the model for this, as the objects could be phrases or sentences of the English language.

4. An example OCR system

4.1 The purpose

To test the EPAM model as mentioned in the section 3.2, I needed to:

- 1) Implement the EPAM model, preferably in C, where I could have more control over the modification of the structures and the processes¹³. It should be noted that this year I contacted Dr. Simon, who offered a lisp version of the model on the MAC for me to test (I ask for a PC version, which they didn't have ready). However, I already had implemented the model. Furthermore, as I expected, it was better to implement the model in order to gain a deeper understanding of it.
- 2) Implement a scrambled word generator module that would provide data for testing the EPAM model. Since the output of such generator would not represent a real data, I tried to find a commercial OCR system that I could use to generate the data. Most OCR systems that I tried, did very poor on single characters of multi-font nature. Further, I had little control on the format or generation of

¹³I have more experience in C than Lisp that EPAM is implemented in.

statistics that I needed. Alternatively, I should have implemented tools to extract such information from the output of those OCR systems.

Further, I wanted to have a character feature extractor, in case that I would want to test the EPAM net, to recognize characters only or words as a list of character features. As such, I decided to implement my own character recognition module.

4.2 The experiment

The basic processes in this experiment are shown in Fig. 4.1 and 4.2.

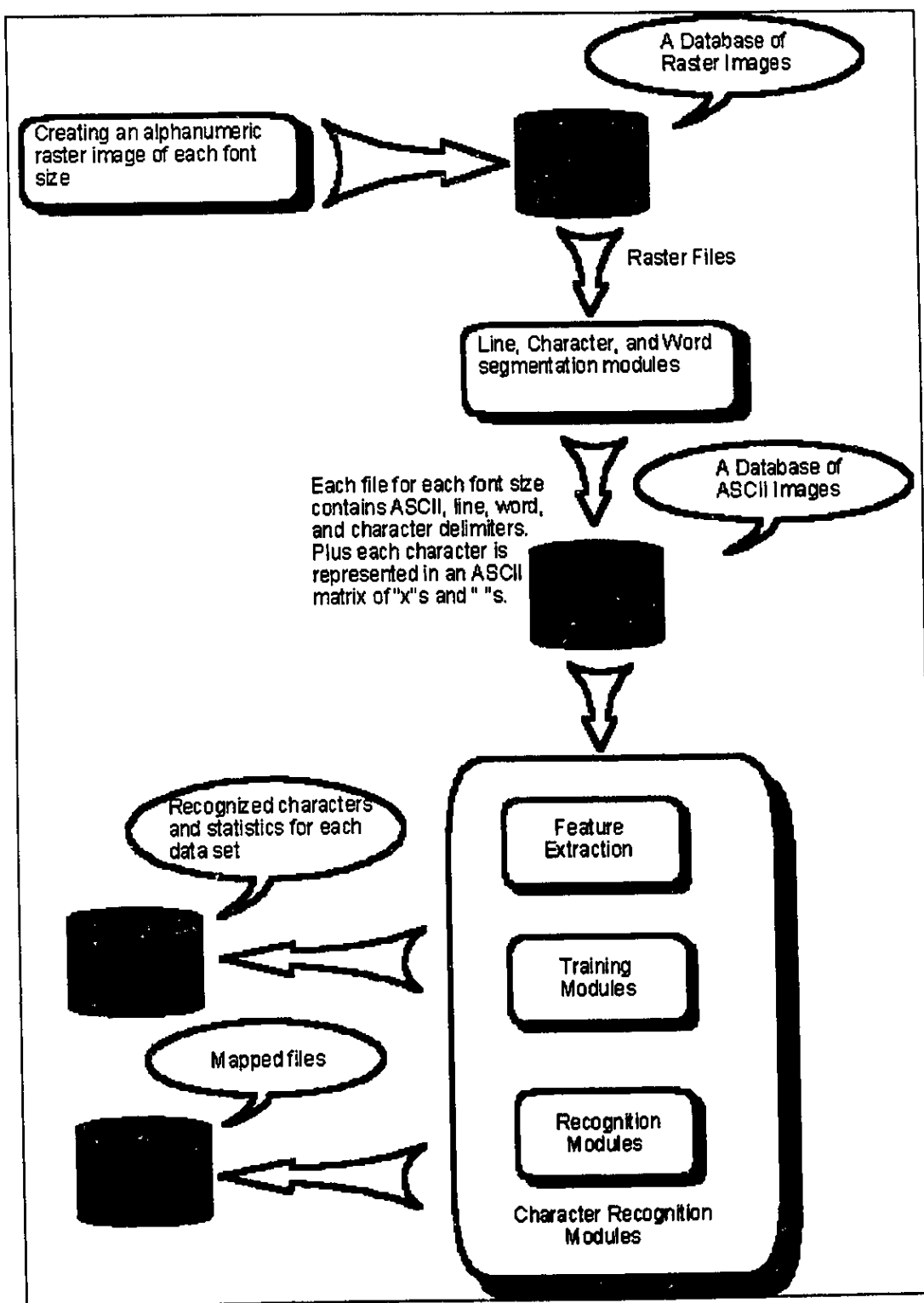


Fig 4.1 The Character Recognition Subsystem.

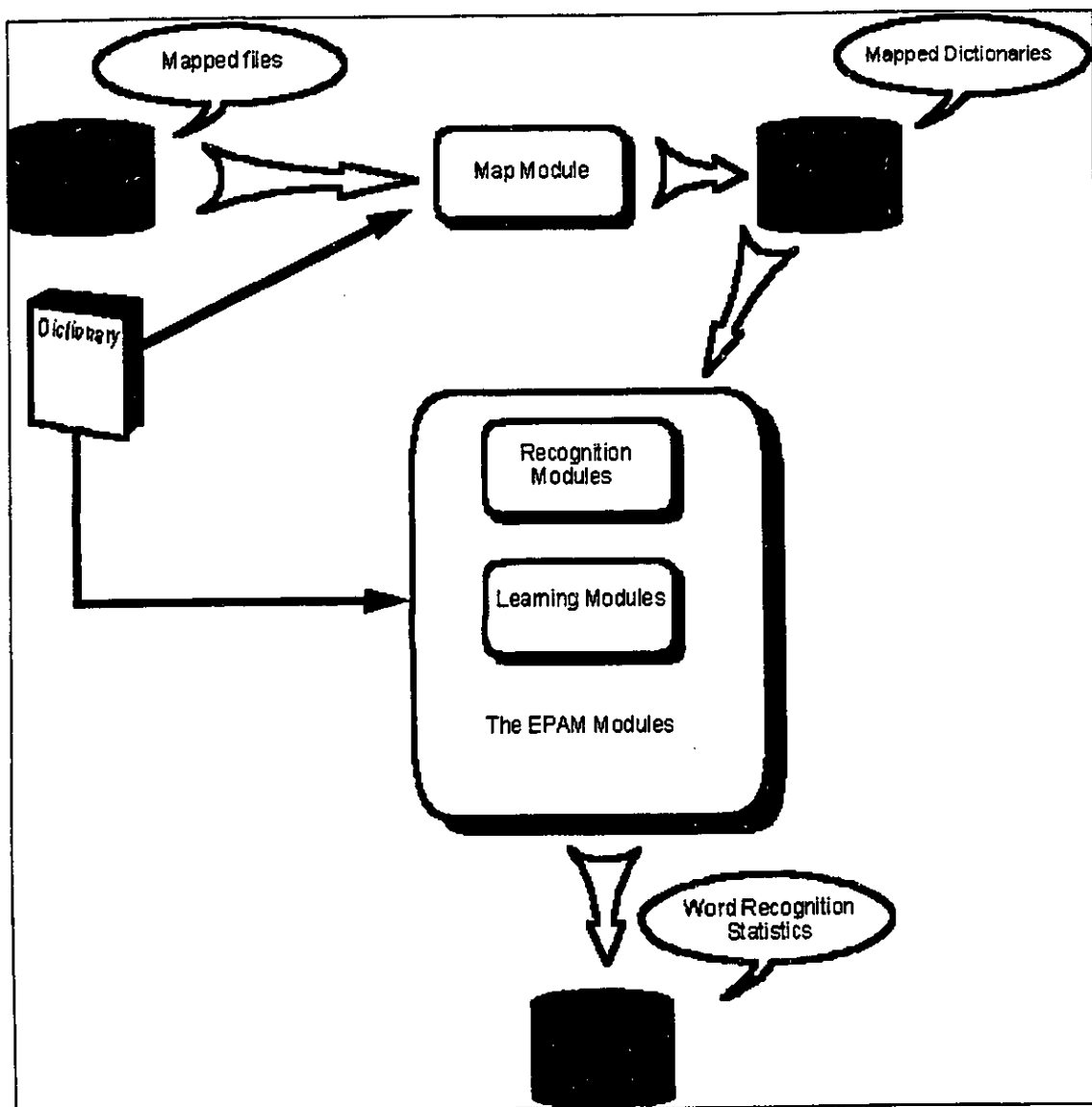


Fig 4.2 The EPAM Subsystem.

- (1) A database of 30 fonts of six different sizes in individual files in a raster format (PCX files) has been created (See the following section for details).
- (2) The data gathered in the step (1) are then segmented to extract the lines, words and characters in each raster file. For each file an ASCII version is generated which contains delimiters for lines and words, and for each character a matrix of Xs and " "s.
- (3) The data generated in step (2) are then processed in different combinations of training and testing by the character recognition subsystem.
- (4) Since it would take a great amount of time and resources to capture a database of words in a dictionary for each font size, a map could be generated by the character recognition subsystem that captures the recognition results for each font under different training conditions. This file, which is small could be used by a mapping module to generate a file that contains a simulated version of recognition of the words in the dictionary by the character recognition subsystem.
- (5) The mapped dictionary for testing and the original dictionary for training is then passed to the EPAM subsystem for word recognition and gathering of statistics.

4.3 The generation of data processed

4.3.1 Fonts used

Table 4.1 gives the list and type of each font. A sample of each font in 12 points is given in appendix A.

| Font # | Font Name | Serif/San-Serif | Font Class |
|--------|---------------------|-----------------|------------|
| 1 | Aero | San-Serif | Decorative |
| 2 | Arial | San-Serif | Text |
| 3 | Avant Guard | San-Serif | Headline |
| 4 | Baskerton | Serif | Text |
| 5 | Buckingham | Serif | Headline |
| 6 | Capelli | Serif | Text |
| 7 | Carnegie | Serif | Text |
| 8 | Century | Serif | Text |
| 9 | Classic Typewriter | Serif | Text |
| 10 | Corporate Condensed | San-Serif | Text |
| 11 | Dateline | Serif | Text |
| 12 | Eterna | San-Serif | Headline |
| 13 | Futuri | San-Serif | Text |
| 14 | Garamand | Serif | Text |
| 15 | Gazette | Serif | Decorative |

| | | | |
|----|---------------|-----------|-------------|
| 16 | Gettysburg | Serif | Text |
| 17 | Gibraltar | San-Serif | Headline |
| 18 | Jewel | Serif | Decorative |
| 19 | Joulliard | Serif | Text |
| 20 | Katrina | Serif | Text |
| 21 | Letter Gothic | San-Serif | Fixed width |
| 22 | Obelisk | San-Serif | Text |
| 23 | Oxford | Serif | Text |
| 24 | Padua | Serif | Text |
| 25 | Pica | Serif | Fixed width |
| 26 | Prestige | Serif | Fixed width |
| 27 | Rockland | Serif | Headline |
| 28 | Souvienne | San-Serif | Text |
| 29 | Times | Serif | Text |
| 30 | Top Hat | Serif | Decorative |

Table 4.1 The fonts used in the experiment.

Table 4.2 shows the distribution of the fonts used, depending on their types.

| | Serif | San-Serif | Text | Decorative | Headline | Fixed width |
|-------------------------|-------|-----------|------|------------|----------|-------------|
| Font numbers (1-15) | 6 | 9 | 10 | 2 | 3 | 0 |
| Font numbers (16-30) | 4 | 11 | 8 | 2 | 2 | 3 |
| Total | 10 | 20 | 18 | 4 | 5 | 3 |

Table 4.2 Distribution of the fonts used.

4.3.2 Generation of raster images of the fonts

The data are generated on a PC compatible using the paintbrush™ program within the Windows® environment. I attained a font package (Typecase™) containing many frequently used text typefaces, which I used within the paintbrush™ program to generate the raster image of the fonts. For each font and font size a PCX file was saved which contained all the alphanumeric characters in that font size. A space was inserted between each character to help further segmentation. This was done to simplify the character segmentation and further recognition of character and words, therefore, limiting the scope of the thesis.

Font sizes of 7 points to 12 points were captured. Each file was captured under a numeric name signifying the font, font size, and font style. An example of a PCX file's content is shown in Fig 4.3

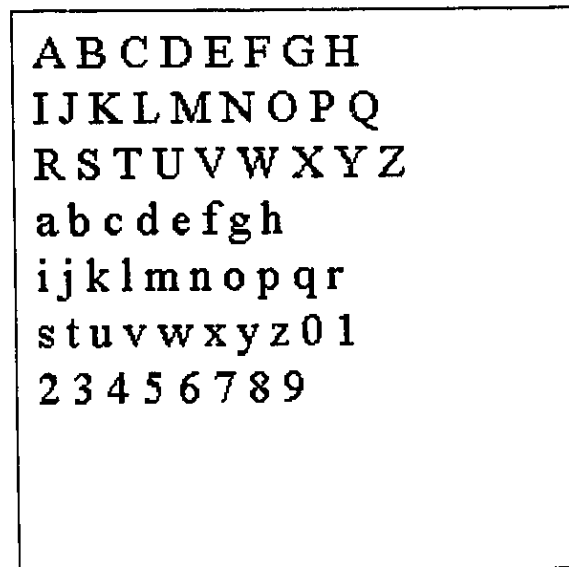


Fig. 4.3 Content of a Raster File (Times Font).

4.3.3 Segmentation of Lines, words, and characters and generation of an ASCII representation file for each font's raster file

A program is written which scans the raster image (a PCX file) from the top to bottom and left to right. The white pixels are taken as the background (0), and the black pixels as the foreground (1). The foreground pixels are assumed to represent lines, words, and characters. Connected components [83] are extracted

in individual lists (See Fig 4.4). The minimax box [84] for each connected component was computed (See Fig. 4.5). Going down from top of the page to the bottom, lines were segmented based on the transitions between connected components (See Fig 4.6). Dots on "i"s and "j"s may form a line if there were no ascender characters or a capital letter in a given line (See Fig 4.7). Inter-line spacing, as well as the size and position of the connected components (dots) relative to other lines connected components (the body of i and j) were used to merge the two lines connected components together. The algorithm written, works also for italic letters; although, to limit the scope of the thesis I did not generate the fonts in an italic format. Further, inter connected component spacing was used to determine the words' boundaries. Characters (connected components) are then saved in an ASCII file in a matrix format consisting of "X"s for foreground pixels and " "s for background pixels. These matrices are separated by string of "\$" as markers, defining the boundaries of characters, words, and lines. For this experiment, since there were no words in the sample data, only characters, the ASCII files contain only markers for the end of characters and lines (See Fig. 4.8).

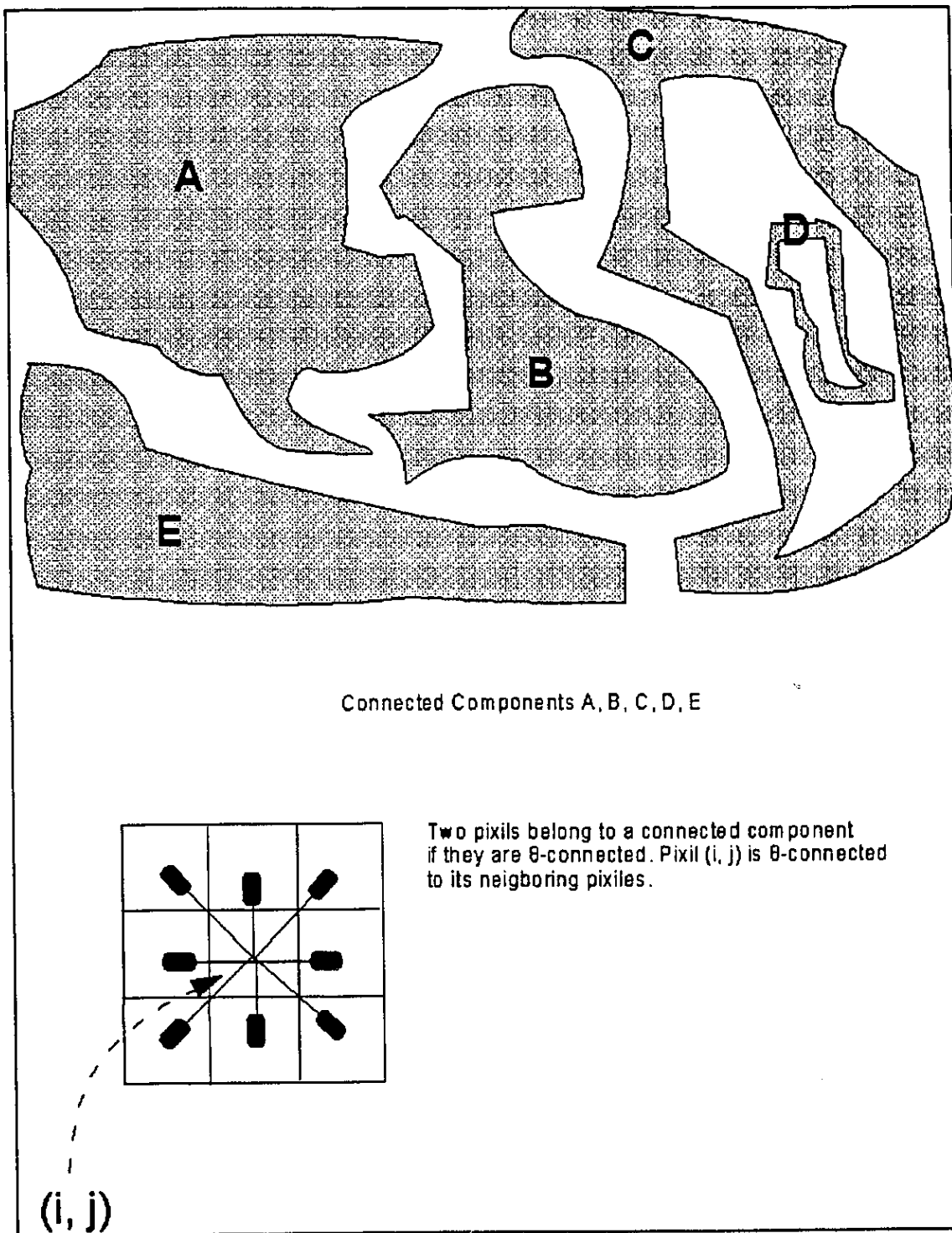


Fig. 4.4 Connected Components.

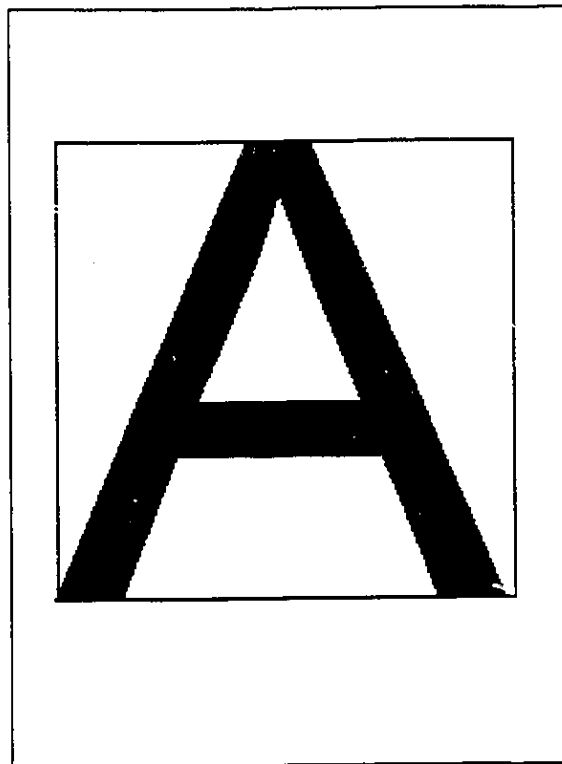


Fig. 4.5 A Minmax box.

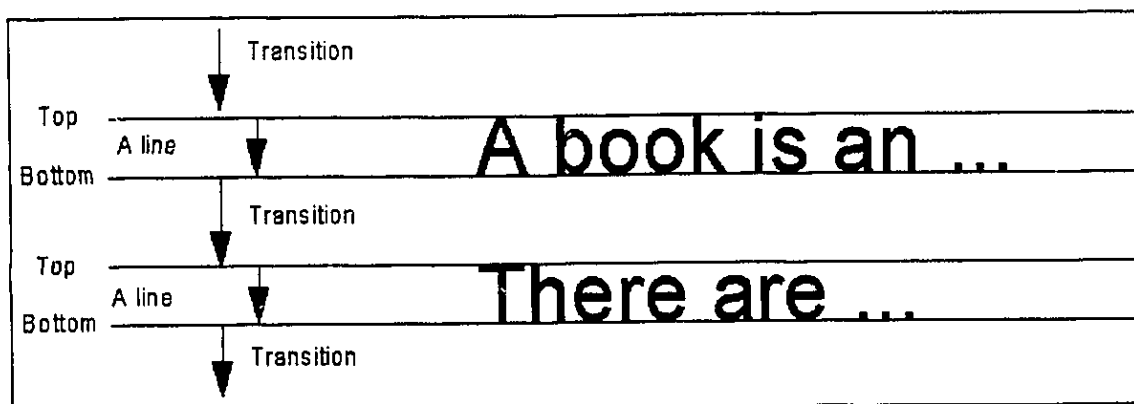


Fig. 4.6 Line determination (Transition between connected components).

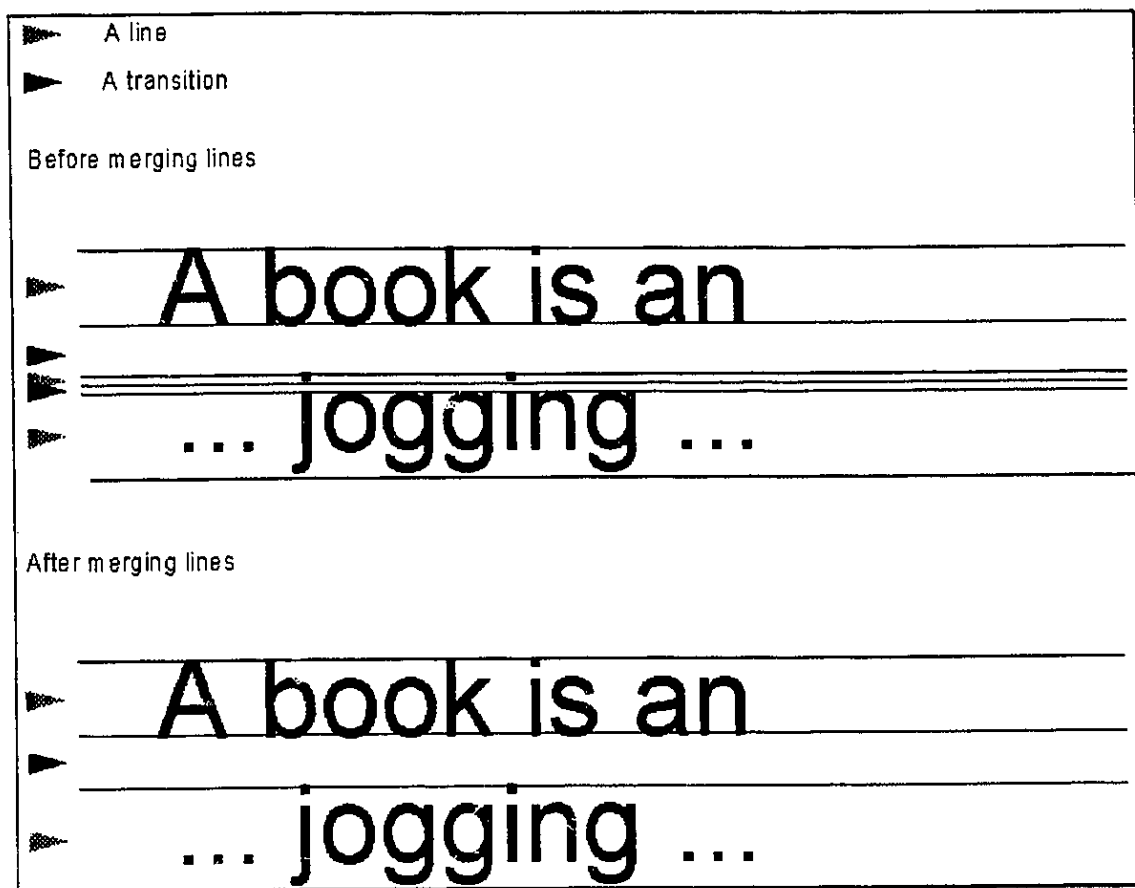


Fig. 4.7 i's and j's in lines.

4.3.4 Generation of recognition map files

In order to test the system properly I should have generated a bitmap of all the words in the dictionary in all fonts and all sizes. This would have taken much resources and time. Further, I did not take in to account in my segmentation and recognition the touching characters nor the disjoint ones. As a result, it was logical to record the recognition of characters for different set of data (training-testing) for each font under a map file. This map file is used later in conjunction with the dictionary to simulate as if the words in the dictionary were passed through the recognition machine (See Section 4.3.6). The structure for each map file contains several lines where each line corresponds to an alphanumeric character class. On each line, recognition results for that character of the given font is put in an order from the most likely to the least likely possibilities for a given character. Next to each possibility, a relative distance to the most likely choice is also included to be used as a potential by the EPAM subsystem (See Fig. 4.9).

```

A 1.0000 4 0.7464 h 0.5130 z 0.5025 b 0.4834 E 0.4717 t 0.4690 x 0.4624 a 0.4482 K 0.4302
P 1.0000 p 0.8588 F 0.8199 s 0.7415 B 0.7061 E 0.6497 R 0.5701 n 0.5517 8 0.5288 2 0.5146
C 1.0000 c 0.6034 G 0.4758 O 0.4197 0 0.3913 o 0.3796 6 0.3450 s 0.3257 q 0.3229 e 0.2987
D 1.0000 n 0.6754 o 0.6147 0 0.5901 u 0.5178 U 0.4671 o 0.4454 b 0.4312 h 0.4169 5 0.3936
F 1.0000 R 0.8381 E 0.7000 K 0.6315 P 0.6151 x 0.5385 p 0.5353 y 0.5056 a 0.4698 z 0.4610
G 1.0000 E 0.2632 P 0.2557 p 0.1839 R 0.1605 x 0.1379 s 0.1346 z 0.1221 B 0.1209 y 0.1159
F 1.0000 o 0.6553 O 0.6316 o 0.5880 6 0.5802 C 0.5287 c 0.4632 a 0.4397 5 0.4269 Q 0.4212
H 1.0000 K 0.9829 x 0.9635 v 0.9251 X 0.8424 R 0.7764 E 0.6937 V 0.6670 a 0.6259 u 0.6234
I 1.0000 l 0.8959 i 0.5632 1 0.3706 t 0.3584 7 0.3583 f 0.3410 J 0.3365 j 0.3327 T 0.3150
J 1.0000 j 0.2361 1 0.1864 1 0.1758 I 0.1738 i 0.1505 3 0.1468 z 0.1426 t 0.1414 7 0.1386
K 1.0000 X 0.5541 x 0.5254 r 0.4474 k 0.4358 R 0.4353 v 0.4343 w 0.4118 E 0.4036 z 0.3873
L 1.0000 r 0.3496 c 0.2794 h 0.2664 k 0.2433 t 0.2408 f 0.2359 l 0.2297 y 0.2279 I 0.2271
M 1.0000 w 0.4938 W 0.4793 H 0.4635 m 0.4597 N 0.4522 u 0.4108 U 0.4056 V 0.3914 v 0.3672
N 1.0000 K 0.7620 X 0.7129 x 0.6909 R 0.6761 v 0.6639 w 0.6606 V 0.6202 H 0.5549 u 0.5547
O 1.0000 o 0.9721 o 0.8601 D 0.4706 G 0.4592 6 0.4194 C 0.4130 n 0.3967 c 0.3879 Q 0.3616
P 1.0000 F 0.3013 p 0.2263 E 0.1454 R 0.1173 s 0.1129 e 0.1066 B 0.1062 r 0.0984 2 0.0983
Q 1.0000 o 0.9156 O 0.8753 o 0.8421 q 0.7168 9 0.7131 e 0.7118 D 0.6996 c 0.6981 p 0.6870
R 1.0000 E 0.7664 K 0.7342 a 0.7239 x 0.7009 F 0.6859 z 0.6642 X 0.6529 P 0.6021 v 0.5743
S 1.0000 S 0.9223 g 0.7719 8 0.7303 3 0.6807 9 0.6260 5 0.6235 2 0.5886 z 0.5776 q 0.5655
T 1.0000 7 0.2073 I 0.1983 r 0.1656 l 0.1637 t 0.1597 y 0.1579 f 0.1579 Y 0.1505 l 0.1483
U 1.0000 u 0.9923 x 0.8615 v 0.7206 X 0.6716 n 0.6574 r 0.6406 K 0.6305 y 0.6249 h 0.5640
V 1.0000 v 0.7610 y 0.4871 X 0.4566 K 0.4234 x 0.4076 Y 0.4068 R 0.3880 E 0.3462 F 0.3171
W 1.0000 w 0.7934 M 0.6445 m 0.5463 N 0.5241 K 0.4849 n 0.4723 q 0.4664 R 0.4596 p 0.4582
X 1.0000 K 0.9738 x 0.9414 R 0.6807 y 0.6746 E 0.6702 F 0.6210 r 0.6173 v 0.6158 z 0.6001
Y 1.0000 Y 0.8990 V 0.6695 v 0.5784 T 0.4357 f 0.3791 x 0.3699 t 0.3526 f 0.3496 r 0.3409
Z 1.0000 z 0.9941 7 0.9174 2 0.6762 r 0.6266 f 0.5245 3 0.5130 t 0.4921 I 0.4573 x 0.4491
a 1.0000 x 0.7880 z 0.7490 5 0.6690 s 0.6596 2 0.6548 t 0.6054 1 0.6039 3 0.5950 R 0.5919
h 1.0000 b 0.6962 k 0.3885 L 0.3622 n 0.3393 u 0.3263 z 0.3046 5 0.3034 U 0.3016 r 0.2878
r 1.0000 c 0.9475 C 0.7566 F 0.6479 f 0.6321 z 0.6212 L 0.6080 s 0.5899 n 0.5587 P 0.5523
4 1.0000 d 0.9262 a 0.8198 A 0.6786 t 0.6036 R 0.5954 v 0.5913 x 0.5876 i 0.5838 q 0.5730
P 1.0000 p 0.8844 r 0.8749 c 0.8727 F 0.8197 z 0.7834 s 0.7768 n 0.7347 2 0.7118 e 0.6925
f 1.0000 t 0.6241 7 0.5866 T 0.5728 y 0.5019 I 0.4883 Y 0.4755 1 0.4638 l 0.4589 r 0.4343
g 1.0000 a 0.8273 q 0.7943 8 0.7917 6 0.7526 S 0.7337 s 0.7306 z 0.7087 K 0.7000 4 0.6853
h 1.0000 k 0.9489 b 0.6976 x 0.4815 L 0.4640 K 0.4613 v 0.4606 a 0.4338 u 0.4267 A 0.4068
i 1.0000 l 0.4290 I 0.3760 f 0.3063 t 0.2831 7 0.2420 J 0.2341 1 0.2297 r 0.2256 j 0.2232
j 1.0000 J 0.1560 1 0.1245 1 0.1211 I 0.1134 i 0.1094 t 0.1054 3 0.0954 7 0.0921 f 0.0879
k 1.0000 b 0.5497 h 0.5191 x 0.4585 L 0.4432 r 0.4171 t 0.4159 K 0.4098 y 0.4010 R 0.3942
l 1.0000 i 0.7799 I 0.7460 t 0.5051 1 0.3780 f 0.3735 j 0.3386 J 0.3141 r 0.2963 7 0.2924
m 1.0000 N 0.7338 n 0.7182 p 0.6214 w 0.6105 M 0.5556 D 0.5482 H 0.5177 u 0.5045 q 0.5007
n 1.0000 x 0.6730 a 0.5737 q 0.5707 h 0.5662 u 0.5461 D 0.5440 r 0.5409 X 0.5299 0 0.5295
n 1.0000 D 0.9618 o 0.8842 O 0.7699 u 0.7593 R 0.7541 U 0.7208 c 0.7086 O 0.6770 p 0.6186
p 1.0000 n 0.6975 P 0.4891 F 0.4606 r 0.4553 D 0.4459 q 0.4369 x 0.4004 s 0.3973 u 0.3948
q 1.0000 a 0.6778 R 0.6591 x 0.6263 g 0.6026 K 0.5900 4 0.5769 Q 0.5586 9 0.5343 v 0.5300
t 1.0000 f 0.7351 I 0.6152 y 0.5792 r 0.5764 1 0.5610 7 0.5522 Y 0.5240 1 0.4784 T 0.4593
s 1.0000 5 0.7043 z 0.5932 F 0.5828 r 0.5773 3 0.5424 S 0.5405 p 0.5184 q 0.5098 c 0.5057
t 1.0000 l 0.7815 I 0.6649 f 0.6110 1 0.5315 i 0.5046 7 0.3732 r 0.3615 y 0.3551 T 0.3485
u 1.0000 U 0.8119 n 0.7129 x 0.6493 h 0.5907 v 0.5646 D 0.5292 k 0.5070 X 0.5063 b 0.5046
V 1.0000 v 0.8488 y 0.6281 Y 0.5829 K 0.4593 x 0.4458 X 0.4283 T 0.4085 F 0.4036 N 0.3970
w 1.0000 X 0.8696 K 0.8587 v 0.8441 x 0.8252 V 0.8005 W 0.7234 y 0.6846 Y 0.6746 N 0.6662
I 1.0000 l 0.8793 1 0.8137 7 0.7540 x 0.7357 T 0.6815 t 0.6336 Y 0.6128 i 0.6054 J 0.5988
Y 1.0000 Y 0.7791 V 0.6960 v 0.6344 F 0.5220 T 0.4531 x 0.4491 f 0.4446 E 0.4362 K 0.4282
7 1.0000 1 0.9786 T 0.9775 t 0.9530 z 0.9491 y 0.8357 I 0.8281 Y 0.7864 f 0.7847 v 0.7243
n 1.0000 D 0.8877 u 0.8489 o 0.8118 U 0.7796 0 0.7442 p 0.7132 r 0.6620 s 0.6177 O 0.6171
l 1.0000 i 0.7799 I 0.7460 t 0.5051 1 0.3780 f 0.3735 j 0.3386 J 0.3141 r 0.2963 7 0.2924
i 1.0000 1 0.9621 1 0.9256 I 0.9065 j 0.8849 J 0.8729 2 0.8603 t 0.8520 7 0.8486 3 0.7226
J 1.0000 3 0.9838 1 0.9756 j 0.9658 1 0.9363 i 0.9273 I 0.9273 7 0.8664 2 0.8436 t 0.8395
d 1.0000 4 0.9371 a 0.8030 x 0.6756 z 0.6278 J 0.6177 s 0.6071 g 0.6051 A 0.5981 2 0.5856
j 1.0000 J 0.8183 3 0.6440 1 0.6336 1 0.6024 I 0.6017 5 0.5742 7 0.5081 t 0.4894 i 0.4887
d 1.0000 z 0.9969 4 0.9188 5 0.9088 s 0.8815 f 0.8419 6 0.8368 Z 0.8360 J 0.8299 E 0.7992
l 1.0000 7 0.9790 J 0.9487 I 0.8394 t 0.7767 1 0.7537 2 0.7351 f 0.7129 r 0.6965 T 0.6879
2 1.0000 s 0.8800 3 0.8329 z 0.7231 s 0.6967 7 0.6591 Z 0.6150 i 0.6023 8 0.5996 S 0.5944
D 1.0000 u 0.9442 p 0.9383 n 0.8741 U 0.8495 q 0.8133 s 0.8133 o 0.7955 9 0.7757 5 0.7595

```

Fig. 4.9 Content of a map file.

4.3.5 The dictionary

The dictionary used in this experiment was taken from the Unix operating systems utility's file (/usr/lib/dict/words) on a SGI workstation. This file contains 23788 words, including words starting with Capital letters. For this experiment, words containing " ' ", and " & " were removed leaving 23692 words. Each word is individually put on a single line in this ASCII file. This dictionary is used to train the EPAM model as well as to generate a mapped dictionary (See Section 4.3.6) in conjunction with the map file generated by the recognition subsystem.

4.3.6 Generation of mapped dictionary

As mentioned in the section 4.3.4 a map file is generated by the recognition subsystem, which is used to generate a simulated recognition of the dictionary in a given font. The dictionary (See Section 4.3.5) is used with each map file to generate a file that contains a record for each word in the dictionary. The first element of the record is the word itself (in a single line) to be compared with the results from EPAM for statistic verification. Each of the subsequent elements (lines) of the record contains recognition results for a single character of a word. The recognition result are taken from the map file. Each line contains a sequence of possible choices for a given character. The choices are from the most likely choice to the least likely with relative distances from the most likely choice (See Fig. 4.10).

```

10th
l 1.0000 i 0.7799 I 0.7460 t 0.5051 l 0.3780 f 0.3735 j 0.3386 J 0.3141 r 0.2963 7 0.2924
n 1.0000 D 0.8877 u 0.8489 o 0.8118 U 0.7796 0 0.7442 p 0.7132 r 0.6620 s 0.6177 O 0.6171
t 1.0000 l 0.7815 I 0.6649 f 0.6110 l 0.5315 i 0.5046 7 0.3732 r 0.3615 y 0.3551 T 0.3485
h 1.0000 k 0.9489 b 0.6976 x 0.4815 L 0.4640 K 0.4613 v 0.4606 a 0.4338 u 0.4267 A 0.4068

1st
l 1.0000 i 0.7799 I 0.7460 t 0.5051 l 0.3780 f 0.3735 j 0.3386 J 0.3141 r 0.2963 7 0.2924
s 1.0000 5 0.7043 z 0.5932 F 0.5828 r 0.5773 3 0.5424 S 0.5405 p 0.5184 q 0.5098 c 0.5057
t 1.0000 l 0.7815 I 0.6649 f 0.6110 l 0.5315 i 0.5046 7 0.3732 r 0.3615 y 0.3551 T 0.3485

2nd
i 1.0000 l 0.9621 l 0.9256 I 0.9065 j 0.8849 J 0.8729 2 0.8603 t 0.8520 7 0.8486 3 0.7226
n 1.0000 x 0.6730 a 0.5737 q 0.5707 h 0.5662 u 0.5461 D 0.5440 r 0.5409 X 0.5299 0 0.5295
4 1.0000 d 0.9262 a 0.8198 A 0.6786 t 0.6036 R 0.5954 v 0.5913 x 0.5876 i 0.5838 q 0.5730

3rd
J 1.0000 3 0.9838 l 0.9756 j 0.9658 l 0.9363 i 0.9273 I 0.9273 7 0.8664 2 0.8436 t 0.8395
t 1.0000 f 0.7351 I 0.6152 y 0.5792 r 0.5764 l 0.5610 7 0.5522 Y 0.5240 l 0.4784 T 0.4593
4 1.0000 d 0.9262 a 0.8198 A 0.6786 t 0.6036 R 0.5954 v 0.5913 x 0.5876 i 0.5838 q 0.5730
.
.
.

```

Fig 4.10 A mapped dictionary's records. See Fig. 4.9 for correlation.

4.4 The character recognition subsystem

4.4.1 The feature extraction

The features are extracted via the character crossings taken at 9 points (zones) on the character matrix (See Fig. 4.11). At each point (zone) the character matrix is scanned in 8 directions, North, North-East, East, South-East, South, South-West, West, North-West. The number of crossings for each direction of each zone is saved as parts of the feature vector. If the count is above 4 the count is set to 4 and a flag is raised for possible further segmentation. The *feature vector* comprises cross counts in all the 8 directions for all the zones.

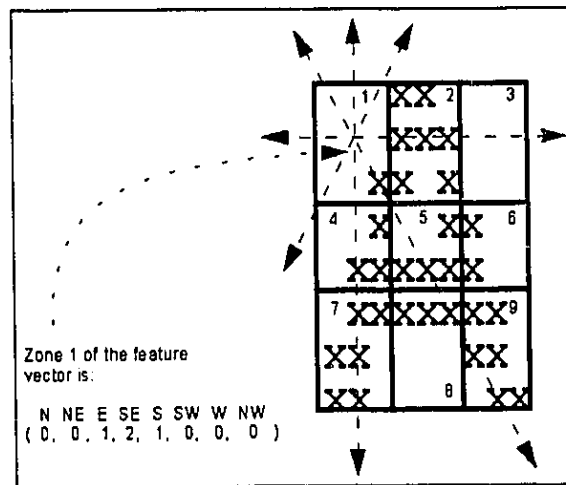


Fig. 4.11 A character's features.

It can be seen that due to symmetry not all the paths need to be scanned. However, first, the computational cost is not that much for a small matrix, and second, multiple representation of a feature reinforces the existence of that

feature. Generally a linear factor does not help in determination of a given class. However, due to a digital grid in the existing images, the path from one zone to another may be different in the other direction, generally different by one pixel. This difference in a small matrix could be a factor in crossing counts. Note that this feature extraction method is insensitive to character size, however, it is sensitive to the rotation of the character. It is possible to transform the feature vector to compensate for rotation of the character, and then compare the transformed vector with the stored model of the existing classes.

One interesting finding that was reported by Miller [31] and elaborated by Simon [85], determines a chunk as having 7 features plus or minus 2. This could suggest that the number of features extracted by humans for an object at their most complex form be around 7. If there were ample resolution and all the redundant crossings were removed, the feature vector described for the characters would contain crossings for 16 paths. (see Fig. 4.12).

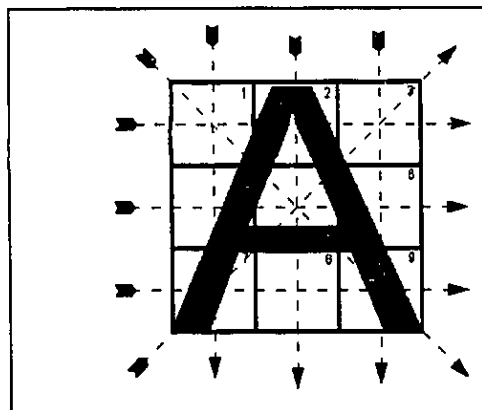


Fig. 4.12 Simplification of feature space.

4.4.2 The training

Several methods were tested to capture a model for each character class. These models were captured over the training sample data as vectors themselves.

- 1) Taking a mean of all the feature vectors in a given class.
- 2) Taking a median of all the feature vectors in a given class.
- 3) Taking a mean and standard deviation of all the feature vectors in a given class.

The model vectors are similar to the feature vectors except that each field of the vector possess the appropriate statistic that is extracted. The unknown feature vectors were then compared against these models for the best match (See Section 4.4.3). In this research project, the best results were found using the first method, i.e., taking the mean of all the feature vectors in a given class.

4.4.3 The recognition

The recognition was performed via finding the smallest distance between the feature vector for the unknown character and the existing models vectors.

- (1) For the case of mean, the distance was computed as the following:

$$\| (X - \mu) \|$$

where X is the unknown feature vector, and μ the mean of the training vectors of a sample space for a given class.

- (2) The same computation was done with the median instead of the mean.
- (3) For the case of standard deviation, a covariance matrix with 0 for the off

diagonal was assumed, i.e., no interdependence between cross counts in different directions. This however, may not be true in reality, although it's a reasonable assumption. Only the variance in cross counts in each direction was computed and stored as the diagonals of the matrix. The distance therefore was computed as the following:

$$\| (X - \mu) \Sigma^{-1} \|$$

where X is the unknown feature vector, and μ the mean of the training vectors of a sample space for a given class, and Σ^{-1} the covariance matrix, a diagonal matrix, where the entries on the diagonal of the matrix are variances (squared standard deviations) that are extracted during training. The aim of the covariance matrix was to normalize the dimensional distances computed between the unknown feature vector and the mean feature vector of a given class.

It was found that using the mean difference (1) and a filter during distance computation much better results could be achieved. This filter did not add inter-vector differences of items if the items' difference was below 0.5. This came from the logic that due to the digital grid's property, a crossing may fall in a path in one sample and it may not in another. The average of the two cases taking 1 for one and 0 for the other is 0.5.

4.5 The EPAM model as a contextual verifier

4.5.1 Implementation

The algorithm for EPAMIII is given in [17]. The algorithm was implemented in C on an IBM® PC compatible. The implementation structure was kept the same as the algorithm in the reference, for correspondence and verification. Modifications and additions of modules were implemented consistent with the original architecture and methodology. Although the semantics of the algorithm looked difficult to be understood at first, by implementing and debugging the system, it proved very simple in nature. The simplicity comes from the recursiveness and local processing. Fig. 4.13 show a flow diagram for components of the EPAM subsystem.

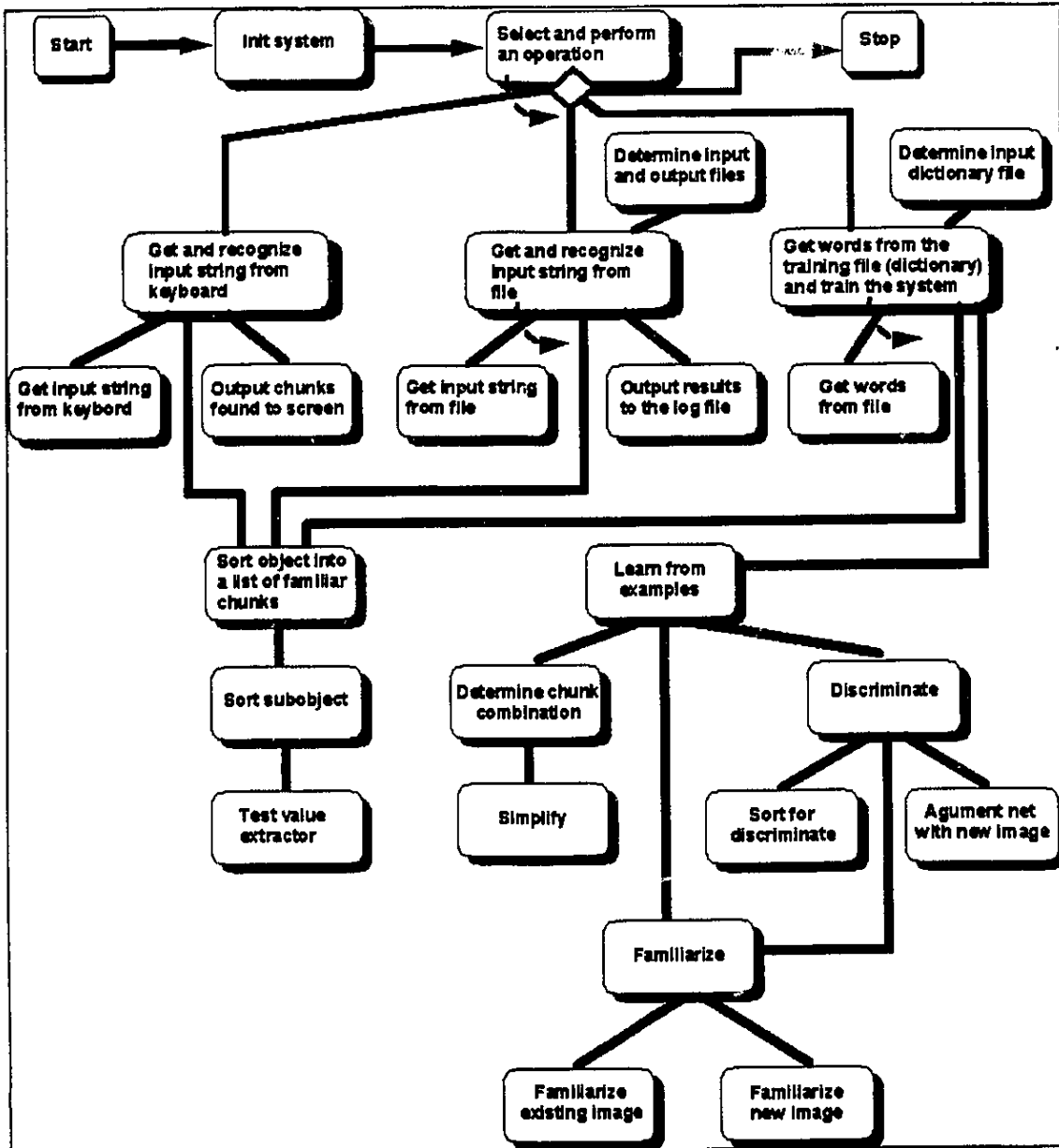


Fig. 4.13 A flow diagram for the EPAM processes.

4.5.2 Training

To accelerate learning, the familiarization parameter was set to 1.0, i.e., every word would be learned in one trial. Further, the whole word parameter was set to 1.0, i.e., words would be learned as a whole or not at all. By doing this, the effect is similar to learning all of the dictionary without any partial learning of any word. Therefore, the system would look like a dictionary look-up system. The dictionary mentioned in section 4.3.5 was used as an input to train the system with the above mentioned settings of parameters.

4.5.3 EPAM search behaviour

EPAM takes the inputted string and performs the search explained in the section 2.2.2.3. The result is:

- 1) If a word (a leaf node) is found through the search that has the same number of features as the inputted stimulus, then that word is outputted as the response of the EPAM. Note that the response is similar to the stimulus to the extent of the tests performed along the search path. Differences with the word retrieved and the stimulus is further exploited in learning processes (See Section 2.2.3).
- 2) If a word (a leaf node) is found through the search that has a fewer number of features than the stimulus, then that word is stored in the short-term memory and other features of the stimulus are searched for in the net to find other terminal nodes that would match those features (the rest of the stimulus). The set of chunks found is the response of the EPAM. This kind of recognition has an

effect of segmenting the stimulus object into many smaller familiar objects. This by itself is a promising model for segmentation by recognition as mentioned in section 3.1.2. To test this property, EPAM was provided with concatenated strings of words as one word (See Fig. 4.14). Most of the time, the effect was segmentation of words that were concatenated (See Fig. 4.14a). The errors generally fell into the categories of over lumping or fragmentation. If two chunks (words) made a familiar meaningful word it would be lumped into one word, although they "meant" to be separate. Also, there could be some chunks left that lost their features due to lumping of those features with other words; and as a result, do not make any single word (See Fig. 4.14b, and Fig. 4.14c).

| | |
|------|--------------------------------------|
| 14.a | 1staboutapple (1st)(about)(apple) |
| 14.b | appleabout1st (adolescent)(1st) |
| 14.c | about1stapple (arbutus)(t)(apple) |

Fig. 4.14 Behaviour of EPAM under concatenated words.

Note also that each chunk found in the response is similar to the stimulus

to the extent of the tests performed along the search path. As a result, most erroneous lumping of chunks, were lumped chunks that were only similar to the extent of the tests performed along the search path. For errors, on average, the concatenation of response chunks were not that similar to the concatenation of the original words. This could have been deducted from the earlier studies of EPAM (See Section 2.3.1.6) showing at least three similar characters in words when a correct one was not responded. This similarity ratio will certainly be more jeopardized as only three characters are similar in a larger chunk. Of course, this dissimilarity could be used during the search to produce response chunks that are closer to the stimulus when concatenated¹⁴. This would solve most errors due to over lumping and fragmentation of chunks. For the cases that can't be solved by this methodology the system needs higher contextual information such as phrase objects, sentence objects, and so on to determine the segmentation of the chunks.

4.5.4 The modified search algorithm

One of the first problems encountered with the EPAM was its sensitivity to the values of the features of the presented stimulus. As mentioned in the section 2.3.2.2, the search may not lead to the "correct" object if the value of a feature has been extracted "wrong". This may also happen if there is noise in the stimulus. One reason for arriving at a wrong response is following only one path

¹⁴In EPAM an image is retrieved without further comparison of other features that did not get tested during the search. If a complete comparison is done between the image and the equivalent part of the stimulus object, a more selective response could be made. This of the course is relative to other possible responses that could be made (See the following section 4.5.4 for more details).

from a test node to another via the branch that its domain item is specified as the closest possible match to the feature in question (See Section 2.3.2.2). Of course, the search would continue by going through the NEC branch if there is not a close match. However, alternate paths having close similarity with the given feature are not explored. Exploring those alternate paths would in effect simulate a form of redundant path access. Those alternate paths could be determined via some form of similarity measure (See Fig. 4.15). In this experiment the alternate paths where characters, therefore, alternate character possibilities with accompanying distances generated from the character recognition subsystem were used to decide these alternate paths. The 10 closest distances were chosen as the basis for alternate paths. Alternatively, the number of alternate paths could be determined by some local and/or global threshold acting on the distances.

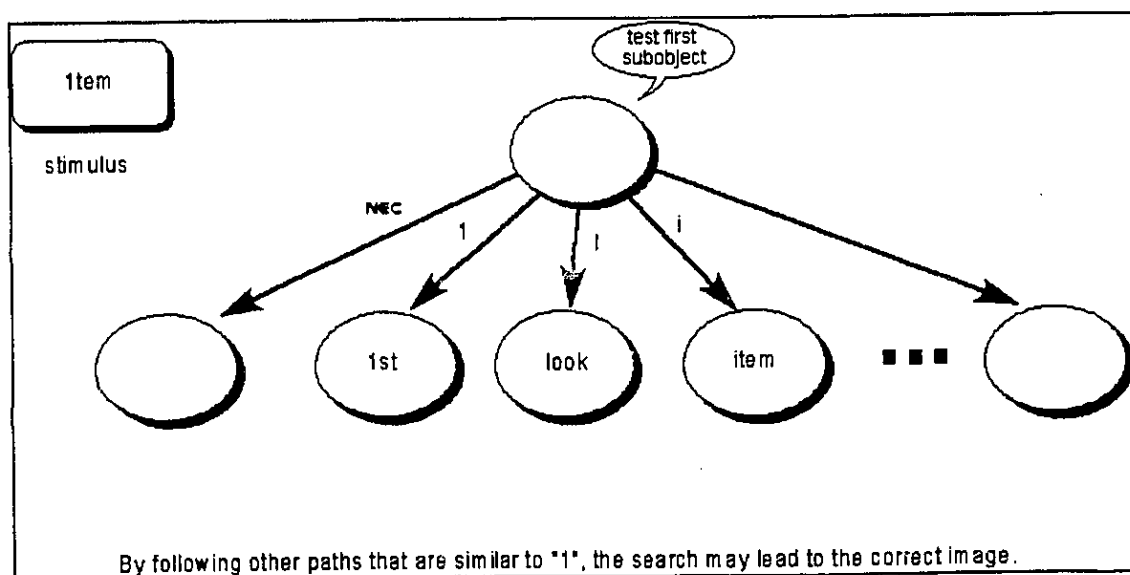


Fig. 4.15 The alternate paths from a test node

Furthermore, all nodes reached via all the paths, are subject to closeness evaluation to the stimulus. The closest image to the stimulus is the response of the system. This closeness could be determined by some form of similarity measure. In this experiment, the provided distances of characters to the closest match were used as potentials to compute the distance between an image and the stimulus. The computation of an image's potential at a leaf node is done as follows:

$$P_T(i) = \prod_{j=1}^m p(j)$$

where $P_T(i)$ is the total potential of an image at a leaf node i , m the length of the image at the leaf node, $p(j)$ the potential of feature j . Each $p(j)$ is determined by searching for that feature (character) of the stimulus in the alternate character list. Corresponding potential of the found character is then applied to computation of the total potential for the image. If the corresponding character is not found in the alternate list, a balanced potential (0.5) is applied to the computation.

In this experiment to test the algorithm described above in a controlled setting, the recall of a single chunk was encouraged, i.e., for each feature that is lacking in the image, a low potential (0.0001) is applied to the total potential of the image, making it very low. The images with potential below a threshold are not considered for competition with other images found throughout the search.

The image with the highest potential is the first immediate response of the

system with other choices as alternate responses depending upon their potential difference to the image with the highest potential.

4.6 The results

4.6.1 Results of the character recognition subsystem

4.6.1.1 Tests scenarios (cases)

The fonts used in this experiment were divided into three sets:

set_1 : { font #1, ...,font #15 }

set_2 : { font #16,...,font #30 }

set_3 : { font #1, ...,font #30 }

Each font as described in the sections 4.2 and 4.3, consists of a set of six different font sizes from 7 to 12 points. Each font size consists of alphanumeric characters including both upper and lower case characters.

Five cases were tested for character recognition subsystem.

| case | Training set | Testing set |
|--------|--------------|-------------|
| case_1 | set_3 | set_3 |
| case_2 | set_1 | set_1 |
| case_3 | set_2 | set_2 |
| case_4 | set_1 | set_2 |
| case_5 | set_2 | set_1 |

Table 4.3 Test cases for character recognition subsystem.

In each case, three types of statistics were taken for six different conditions as follows: (The three types of statistics [mean, median, and standard deviation] are described in the section 4.4.2)

- cond_1 : Recognition results for the first choice.
- cond_2 : Recognition results for the first choice not
differentiating between the upper and the lower case characters.
- cond_3 : Recognition results for the top five choices.
- cond_4 : Recognition results for the top five choices not
differentiating between the upper and the lower case characters.
- cond_5 : Recognition results for the top ten choices.
- cond_6 : Recognition results for the top ten choices not
differentiating between the upper and the lower case characters.

Tables 4.4 - 4.37 present the recognition results for the "mean" statistic. A brief description of each case and condition tested is given under each table. For the "median" and the "standard deviation", only ((case_1, cond_1) and (case_1, cond_5)) is reported for comparison with the "mean", since these two statistics did not do as well as the "mean" did.

4.6.1.2 Results of the cases where training and testing sets were the same (Using the "mean" statistic)

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.4677 | 0.4516 | 0.5000 | 0.5484 | 0.6613 | 0.4839 | | 0.5188 |
| 2 | | 0.5161 | 0.5806 | 0.7097 | 0.6774 | 0.7903 | 0.8065 | | 0.6801 |
| 3 | | 0.4839 | 0.4677 | 0.5968 | 0.5806 | 0.6774 | 0.6290 | | 0.5726 |
| 4 | | 0.5000 | 0.6452 | 0.7419 | 0.7419 | 0.7581 | 0.8226 | | 0.7016 |
| 5 | | 0.5000 | 0.5323 | 0.5645 | 0.6452 | 0.5323 | 0.4677 | | 0.5403 |
| 6 | | 0.4355 | 0.5484 | 0.8548 | 0.8387 | 0.8226 | 0.7903 | | 0.7151 |
| 7 | | 0.4194 | 0.5323 | 0.6774 | 0.8065 | 0.7742 | 0.7419 | | 0.6586 |
| 8 | | 0.7581 | 0.7742 | 0.5645 | 0.7258 | 0.7903 | 0.6935 | | 0.7177 |
| 9 | | 0.7419 | 0.7903 | 0.7742 | 0.6613 | 0.8387 | 0.6774 | | 0.7473 |
| 10 | | 0.3710 | 0.4677 | 0.5806 | 0.6774 | 0.6129 | 0.6129 | | 0.5538 |
| 11 | | 0.4032 | 0.5645 | 0.5484 | 0.5484 | 0.5806 | 0.5645 | | 0.5349 |
| 12 | | 0.4516 | 0.6290 | 0.7097 | 0.7097 | 0.6613 | 0.7742 | | 0.6559 |
| 13 | | 0.6129 | 0.4839 | 0.6290 | 0.6452 | 0.6774 | 0.7097 | | 0.6263 |
| 14 | | 0.6290 | 0.7258 | 0.6129 | 0.7258 | 0.6935 | 0.7097 | | 0.6828 |
| 15 | | 0.7097 | 0.5484 | 0.6290 | 0.8226 | 0.5484 | 0.6452 | | 0.6505 |
| 16 | | 0.5161 | 0.5484 | 0.5484 | 0.5000 | 0.4516 | 0.4194 | | 0.4973 |
| 17 | | 0.5323 | 0.6613 | 0.7581 | 0.5806 | 0.6935 | 0.7903 | | 0.6694 |
| 18 | | 0.4194 | 0.4355 | 0.6613 | 0.6452 | 0.6290 | 0.5968 | | 0.5645 |
| 19 | | 0.5645 | 0.7097 | 0.7903 | 0.7581 | 0.8065 | 0.7419 | | 0.7285 |
| 20 | | 0.5484 | 0.7419 | 0.7581 | 0.7581 | 0.7258 | 0.7903 | | 0.7204 |
| 21 | | 0.3548 | 0.4194 | 0.6613 | 0.6935 | 0.6129 | 0.6290 | | 0.5618 |
| 22 | | 0.7097 | 0.6452 | 0.6774 | 0.6452 | 0.7903 | 0.7742 | | 0.7070 |
| 23 | | 0.5161 | 0.6774 | 0.7581 | 0.8710 | 0.9032 | 0.7903 | | 0.7527 |
| 24 | | 0.6290 | 0.8226 | 0.7903 | 0.7742 | 0.7581 | 0.7742 | | 0.7581 |
| 25 | | 0.3226 | 0.4839 | 0.6290 | 0.7742 | 0.7097 | 0.7097 | | 0.6048 |
| 26 | | 0.5161 | 0.5968 | 0.7581 | 0.7097 | 0.6129 | 0.6129 | | 0.6344 |
| 27 | | 0.6935 | 0.7581 | 0.8226 | 0.6290 | 0.5323 | 0.6129 | | 0.6747 |
| 28 | | 0.6613 | 0.7419 | 0.7903 | 0.8226 | 0.7581 | 0.8548 | | 0.7715 |
| 29 | | 0.5806 | 0.6452 | 0.7581 | 0.7581 | 0.7742 | 0.8226 | | 0.7231 |
| 30 | | 0.5161 | 0.5968 | 0.6290 | 0.6452 | 0.7258 | 0.7097 | | 0.6371 |
| Average | | 0.5360 | 0.6075 | 0.6828 | 0.6973 | 0.6968 | 0.6919 | | 0.6521 |

Table 4.4 case_1, cond_1

Trained with {font #1,...,font #30}.
Tested with {font #1,...,font #30}.
Recognition results for the first choice.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.6290 | 0.5323 | 0.5968 | 0.6452 | 0.7097 | 0.5645 | | 0.6129 |
| 2 | | 0.5968 | 0.6774 | 0.7903 | 0.7581 | 0.8226 | 0.8226 | | 0.7446 |
| 3 | | 0.6290 | 0.6129 | 0.7097 | 0.6935 | 0.7742 | 0.7258 | | 0.6909 |
| 4 | | 0.5806 | 0.7419 | 0.8387 | 0.7742 | 0.7903 | 0.8871 | | 0.7688 |
| 5 | | 0.5806 | 0.6290 | 0.6613 | 0.7258 | 0.5968 | 0.5323 | | 0.6210 |
| 6 | | 0.5161 | 0.5645 | 0.8710 | 0.8871 | 0.8710 | 0.8548 | | 0.7608 |
| 7 | | 0.5161 | 0.5645 | 0.7903 | 0.8226 | 0.8548 | 0.8387 | | 0.7312 |
| 8 | | 0.7903 | 0.8226 | 0.5968 | 0.7258 | 0.7903 | 0.8065 | | 0.7554 |
| 9 | | 0.8226 | 0.8710 | 0.8548 | 0.7258 | 0.8871 | 0.7903 | | 0.8253 |
| 10 | | 0.4839 | 0.5484 | 0.6452 | 0.7581 | 0.7581 | 0.6613 | | 0.6425 |
| 11 | | 0.4677 | 0.5968 | 0.5968 | 0.6129 | 0.6774 | 0.6452 | | 0.5995 |
| 12 | | 0.5968 | 0.7258 | 0.8710 | 0.7903 | 0.7742 | 0.8387 | | 0.7661 |
| 13 | | 0.7097 | 0.5968 | 0.7419 | 0.6935 | 0.7419 | 0.7742 | | 0.7097 |
| 14 | | 0.7258 | 0.8548 | 0.6935 | 0.8226 | 0.7903 | 0.8387 | | 0.7876 |
| 15 | | 0.7581 | 0.6452 | 0.7581 | 0.8710 | 0.6290 | 0.6774 | | 0.7231 |
| 16 | | 0.5484 | 0.5645 | 0.5806 | 0.5806 | 0.5000 | 0.5000 | | 0.5457 |
| 17 | | 0.6613 | 0.7742 | 0.8871 | 0.7097 | 0.7903 | 0.8387 | | 0.7769 |
| 18 | | 0.4839 | 0.4839 | 0.7097 | 0.7258 | 0.6774 | 0.6935 | | 0.6290 |
| 19 | | 0.6452 | 0.7581 | 0.8387 | 0.7903 | 0.8387 | 0.8387 | | 0.7849 |
| 20 | | 0.6452 | 0.8387 | 0.8548 | 0.7903 | 0.7742 | 0.8710 | | 0.7957 |
| 21 | | 0.3871 | 0.4677 | 0.7258 | 0.7419 | 0.6452 | 0.6935 | | 0.6102 |
| 22 | | 0.7903 | 0.7258 | 0.8226 | 0.7258 | 0.8226 | 0.8387 | | 0.7876 |
| 23 | | 0.5645 | 0.7419 | 0.8226 | 0.8871 | 0.9516 | 0.8226 | | 0.7984 |
| 24 | | 0.7097 | 0.9032 | 0.9032 | 0.8548 | 0.8548 | 0.8871 | | 0.8522 |
| 25 | | 0.3710 | 0.5484 | 0.6774 | 0.8065 | 0.7581 | 0.8065 | | 0.6613 |
| 26 | | 0.6452 | 0.6613 | 0.8065 | 0.7742 | 0.6935 | 0.6613 | | 0.7070 |
| 27 | | 0.8065 | 0.8387 | 0.9032 | 0.6613 | 0.5806 | 0.6290 | | 0.7366 |
| 28 | | 0.7581 | 0.8065 | 0.8387 | 0.8226 | 0.7903 | 0.9032 | | 0.8199 |
| 29 | | 0.6129 | 0.6774 | 0.8065 | 0.8710 | 0.8226 | 0.8871 | | 0.7796 |
| 30 | | 0.5645 | 0.6452 | 0.6774 | 0.6935 | 0.7742 | 0.7742 | | 0.6882 |
| Average | | 0.6199 | 0.6806 | 0.7624 | 0.7581 | 0.7581 | 0.7634 | | 0.7237 |

Table 4.5 case_1, cond_2

Trained with (font #1,...,font #30).
Tested with (font #1,...,font #30).
Recognition results for the first choice
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.8387 | 0.8226 | 0.9032 | 0.8871 | 0.9032 | 0.8387 | | 0.8656 |
| 2 | | 0.8871 | 0.9194 | 0.9355 | 0.9355 | 0.9839 | 0.9677 | | 0.9382 |
| 3 | | 0.8710 | 0.9194 | 0.9677 | 0.9355 | 0.9677 | 0.9516 | | 0.9355 |
| 4 | | 0.8871 | 0.9194 | 1.0000 | 0.9839 | 0.9516 | 0.9516 | | 0.9489 |
| 5 | | 0.8548 | 0.8871 | 0.9194 | 0.8871 | 0.8871 | 0.8548 | | 0.8817 |
| 6 | | 0.7419 | 0.8065 | 0.9677 | 0.9839 | 1.0000 | 1.0000 | | 0.9167 |
| 7 | | 0.7258 | 0.8226 | 0.9194 | 0.9677 | 0.9677 | 1.0000 | | 0.9005 |
| 8 | | 0.9355 | 0.9516 | 0.8871 | 0.8871 | 0.9516 | 0.9516 | | 0.9274 |
| 9 | | 0.9355 | 0.9677 | 0.9355 | 0.9839 | 0.9839 | 0.9194 | | 0.9543 |
| 10 | | 0.8710 | 0.8548 | 0.9516 | 0.9516 | 0.9516 | 0.8710 | | 0.9086 |
| 11 | | 0.8226 | 0.9032 | 0.9194 | 0.9516 | 0.8871 | 0.9032 | | 0.8978 |
| 12 | | 0.9032 | 0.9516 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9704 |
| 13 | | 0.9355 | 0.8548 | 0.9194 | 0.9194 | 0.9677 | 0.9516 | | 0.9247 |
| 14 | | 0.9516 | 0.9839 | 0.9355 | 0.9677 | 0.9677 | 0.9839 | | 0.9651 |
| 15 | | 0.9516 | 0.9194 | 0.9355 | 0.9516 | 0.9839 | 0.9839 | | 0.9543 |
| 16 | | 0.8387 | 0.8387 | 0.8871 | 0.8065 | 0.7742 | 0.7903 | | 0.8226 |
| 17 | | 0.9032 | 0.9032 | 0.9677 | 0.9677 | 0.9839 | 1.0000 | | 0.9543 |
| 18 | | 0.9032 | 0.7903 | 0.9032 | 0.8871 | 0.8387 | 0.8871 | | 0.8683 |
| 19 | | 0.8548 | 0.8871 | 0.9677 | 0.9677 | 0.9677 | 0.9677 | | 0.9355 |
| 20 | | 0.8871 | 0.9677 | 0.9677 | 0.9677 | 0.9677 | 0.9839 | | 0.9570 |
| 21 | | 0.6129 | 0.8065 | 0.9355 | 0.9677 | 0.9355 | 0.9194 | | 0.8629 |
| 22 | | 0.9194 | 0.9355 | 0.9516 | 0.9355 | 0.9839 | 0.9839 | | 0.9516 |
| 23 | | 0.8710 | 0.9194 | 0.9839 | 1.0000 | 1.0000 | 0.9516 | | 0.9543 |
| 24 | | 0.9194 | 0.9677 | 0.9677 | 0.9677 | 0.9839 | 0.9677 | | 0.9624 |
| 25 | | 0.7419 | 0.8226 | 0.9194 | 0.9355 | 0.9516 | 0.9516 | | 0.8871 |
| 26 | | 0.8548 | 0.8871 | 0.9677 | 0.9516 | 0.9194 | 0.8710 | | 0.9086 |
| 27 | | 0.9839 | 0.9839 | 0.9677 | 0.9516 | 0.9194 | 0.9355 | | 0.9570 |
| 28 | | 0.9032 | 0.9516 | 0.9839 | 0.9677 | 0.9839 | 1.0000 | | 0.9651 |
| 29 | | 0.8226 | 0.8871 | 0.9194 | 1.0000 | 0.9677 | 0.9839 | | 0.9301 |
| 30 | | 0.8065 | 0.8226 | 0.9194 | 0.9516 | 0.9677 | 0.9839 | | 0.9086 |
| Average | | 0.8645 | 0.8952 | 0.9435 | 0.9468 | 0.9495 | 0.9435 | | 0.9238 |

Table 4.6 case_1, cond_3

Trained with {font #1,...,font #30}.
Tested with {font #1,...,font #30}.
Recognition results for the top five choices.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.8548 | 0.8548 | 0.9032 | 0.9032 | 0.9194 | 0.8387 | | 0.8790 |
| 2 | | 0.8871 | 0.9194 | 0.9355 | 0.9355 | 0.9839 | 0.9677 | | 0.9382 |
| 3 | | 0.8710 | 0.9194 | 0.9839 | 0.9516 | 0.9677 | 0.9516 | | 0.9409 |
| 4 | | 0.8871 | 0.9355 | 1.0000 | 0.9839 | 0.9677 | 0.9516 | | 0.9543 |
| 5 | | 0.8710 | 0.9194 | 0.9355 | 0.8871 | 0.9032 | 0.8871 | | 0.9005 |
| 6 | | 0.7742 | 0.8065 | 0.9677 | 0.9839 | 1.0000 | 1.0000 | | 0.9220 |
| 7 | | 0.7419 | 0.8226 | 0.9194 | 0.9677 | 0.9839 | 1.0000 | | 0.9059 |
| 8 | | 0.9355 | 0.9677 | 0.8871 | 0.8871 | 0.9677 | 1.0000 | | 0.9409 |
| 9 | | 0.9355 | 0.9677 | 0.9355 | 0.9839 | 0.9839 | 0.9194 | | 0.9543 |
| 10 | | 0.8871 | 0.8548 | 0.9516 | 0.9516 | 0.9839 | 0.9032 | | 0.9220 |
| 11 | | 0.8226 | 0.9355 | 0.9355 | 0.9677 | 0.9194 | 0.9355 | | 0.9194 |
| 12 | | 0.9355 | 0.9516 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9758 |
| 13 | | 0.9355 | 0.8548 | 0.9355 | 0.9194 | 0.9677 | 0.9677 | | 0.9301 |
| 14 | | 0.9677 | 0.9839 | 0.9516 | 0.9677 | 0.9677 | 0.9839 | | 0.9704 |
| 15 | | 0.9516 | 0.9194 | 0.9516 | 0.9677 | 1.0000 | 1.0000 | | 0.9651 |
| 16 | | 0.8710 | 0.8387 | 0.8871 | 0.8548 | 0.8226 | 0.8065 | | 0.8468 |
| 17 | | 0.9032 | 0.9194 | 0.9677 | 0.9839 | 0.9839 | 1.0000 | | 0.9597 |
| 18 | | 0.9032 | 0.8065 | 0.9032 | 0.9194 | 0.8871 | 0.9355 | | 0.8925 |
| 19 | | 0.8548 | 0.9032 | 0.9677 | 0.9677 | 0.9839 | 0.9839 | | 0.9435 |
| 20 | | 0.8871 | 0.9677 | 0.9677 | 0.9839 | 0.9839 | 0.9839 | | 0.9624 |
| 21 | | 0.6774 | 0.9032 | 0.9677 | 0.9839 | 0.9355 | 0.9194 | | 0.8978 |
| 22 | | 0.9194 | 0.9355 | 0.9516 | 0.9355 | 0.9839 | 0.9839 | | 0.9516 |
| 23 | | 0.8710 | 0.9355 | 0.9839 | 1.0000 | 1.0000 | 0.9516 | | 0.9570 |
| 24 | | 0.9194 | 0.9677 | 0.9677 | 0.9677 | 0.9839 | 0.9839 | | 0.9651 |
| 25 | | 0.7903 | 0.8387 | 0.9516 | 0.9839 | 0.9839 | 0.9677 | | 0.9194 |
| 26 | | 0.9032 | 0.9194 | 0.9839 | 0.9839 | 0.9355 | 0.8710 | | 0.9328 |
| 27 | | 0.9839 | 0.9839 | 0.9677 | 0.9677 | 0.9355 | 0.9355 | | 0.9624 |
| 28 | | 0.9194 | 0.9516 | 0.9839 | 0.9677 | 0.9839 | 1.0000 | | 0.9677 |
| 29 | | 0.8387 | 0.8871 | 0.9194 | 1.0000 | 0.9677 | 0.9839 | | 0.9328 |
| 30 | | 0.8065 | 0.8387 | 0.9194 | 0.9516 | 0.9677 | 0.9839 | | 0.9113 |
| Average | | 0.8769 | 0.9070 | 0.9495 | 0.9565 | 0.9613 | 0.9532 | | 0.9341 |

Table 4.7 case_1, cond_4

Trained with (font #1,...,font #30).
Tested with (font #1,...,font #30).
Recognition results for the top five choices
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.9516 | 0.8871 | 0.9677 | 0.9677 | 0.9516 | 0.9032 | | 0.9382 |
| 2 | | 0.9677 | 0.9677 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | | 0.9839 |
| 3 | | 0.9839 | 0.9839 | 0.9839 | 0.9677 | 0.9839 | 0.9839 | | 0.9812 |
| 4 | | 0.9516 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9866 |
| 5 | | 0.9516 | 0.9677 | 0.9839 | 0.9355 | 0.9516 | 0.9355 | | 0.9543 |
| 6 | | 0.8710 | 0.9194 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9651 |
| 7 | | 0.8710 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9731 |
| 8 | | 0.9839 | 1.0000 | 0.9516 | 0.9839 | 1.0000 | 1.0000 | | 0.9866 |
| 9 | | 0.9677 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | 0.9516 | | 0.9839 |
| 10 | | 0.9516 | 0.9194 | 0.9677 | 1.0000 | 1.0000 | 0.9677 | | 0.9677 |
| 11 | | 0.9194 | 0.9677 | 1.0000 | 0.9677 | 0.9516 | 0.9839 | | 0.9651 |
| 12 | | 0.9677 | 0.9677 | 1.0000 | 0.9839 | 1.0000 | 1.0000 | | 0.9866 |
| 13 | | 0.9839 | 0.9516 | 0.9194 | 0.9677 | 0.9677 | 0.9677 | | 0.9597 |
| 14 | | 1.0000 | 1.0000 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | | 0.9946 |
| 15 | | 0.9839 | 0.9677 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | | 0.9866 |
| 16 | | 0.9677 | 0.9677 | 0.9677 | 0.9677 | 0.9194 | 0.9032 | | 0.9489 |
| 17 | | 0.9839 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9919 |
| 18 | | 0.9839 | 0.9194 | 0.9516 | 0.9839 | 0.9516 | 0.9516 | | 0.9570 |
| 19 | | 0.9839 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9919 |
| 20 | | 0.9516 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | 1.0000 | | 0.9892 |
| 21 | | 0.7581 | 0.9032 | 0.9839 | 0.9839 | 0.9839 | 0.9677 | | 0.9301 |
| 22 | | 0.9516 | 0.9839 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9839 |
| 23 | | 0.9677 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | | 0.9892 |
| 24 | | 0.9677 | 0.9839 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9866 |
| 25 | | 0.8387 | 0.9194 | 0.9355 | 0.9839 | 0.9839 | 0.9839 | | 0.9409 |
| 26 | | 0.9355 | 0.9355 | 1.0000 | 0.9839 | 0.9677 | 0.9355 | | 0.9597 |
| 27 | | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9677 | 0.9516 | | 0.9866 |
| 28 | | 0.9677 | 0.9839 | 1.0000 | 0.9839 | 1.0000 | 1.0000 | | 0.9892 |
| 29 | | 1.0000 | 1.0000 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9973 |
| 30 | | 0.8871 | 0.8710 | 0.9516 | 0.9677 | 1.0000 | 1.0000 | | 0.9462 |
| Average | | 0.9484 | 0.9618 | 0.9823 | 0.9860 | 0.9839 | 0.9780 | | 0.9734 |

Table 4.8 case_1, cond_5

Trained with (font #1,...,font #30).
Tested with (font #1,...,font #30).
Recognition results for the top ten choices.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.9516 | 0.9032 | 0.9677 | 0.9677 | 0.9516 | 0.9032 | | 0.9409 |
| 2 | | 0.9677 | 0.9677 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | | 0.9839 |
| 3 | | 0.9839 | 0.9839 | 0.9839 | 0.9677 | 0.9839 | 0.9839 | | 0.9812 |
| 4 | | 0.9516 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9866 |
| 5 | | 0.9516 | 0.9677 | 0.9839 | 0.9516 | 0.9516 | 0.9516 | | 0.9597 |
| 6 | | 0.8871 | 0.9194 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9677 |
| 7 | | 0.8710 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9731 |
| 8 | | 0.9839 | 1.0000 | 0.9677 | 0.9839 | 1.0000 | 1.0000 | | 0.9892 |
| 9 | | 0.9677 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | 0.9677 | | 0.9866 |
| 10 | | 0.9677 | 0.9194 | 0.9677 | 1.0000 | 1.0000 | 0.9677 | | 0.9704 |
| 11 | | 0.9355 | 0.9677 | 1.0000 | 0.9839 | 0.9839 | 0.9839 | | 0.9758 |
| 12 | | 0.9677 | 0.9677 | 1.0000 | 0.9839 | 1.0000 | 1.0000 | | 0.9866 |
| 13 | | 0.9839 | 0.9677 | 0.9516 | 0.9839 | 0.9839 | 0.9839 | | 0.9758 |
| 14 | | 1.0000 | 1.0000 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9973 |
| 15 | | 0.9839 | 0.9839 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | | 0.9892 |
| 16 | | 0.9677 | 0.9839 | 0.9677 | 0.9677 | 0.9355 | 0.9194 | | 0.9570 |
| 17 | | 0.9839 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9919 |
| 18 | | 0.9839 | 0.9194 | 0.9516 | 1.0000 | 0.9677 | 0.9839 | | 0.9677 |
| 19 | | 0.9839 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9946 |
| 20 | | 0.9516 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | 1.0000 | | 0.9892 |
| 21 | | 0.8871 | 0.9677 | 0.9839 | 1.0000 | 0.9839 | 0.9839 | | 0.9677 |
| 22 | | 0.9516 | 0.9839 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9839 |
| 23 | | 0.9677 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | | 0.9892 |
| 24 | | 0.9677 | 0.9839 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9866 |
| 25 | | 0.8871 | 0.9355 | 0.9677 | 1.0000 | 0.9839 | 0.9839 | | 0.9597 |
| 26 | | 0.9516 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 0.9516 | | 0.9785 |
| 27 | | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9677 | 0.9516 | | 0.9866 |
| 28 | | 0.9839 | 0.9839 | 1.0000 | 0.9839 | 1.0000 | 1.0000 | | 0.9919 |
| 29 | | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 1.0000 |
| 30 | | 0.8871 | 0.8710 | 0.9516 | 0.9677 | 1.0000 | 1.0000 | | 0.9462 |
| Average | | 0.9570 | 0.9683 | 0.9860 | 0.9898 | 0.9876 | 0.9823 | | 0.9785 |

Table 4.9 case_1, cond_6

Trained with {font #1,...,font #30}.
Tested with {font #1,...,font #30}.
Recognition results for the top ten choices
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.5323 | 0.4355 | 0.5161 | 0.5806 | 0.6452 | 0.5323 | | 0.5403 |
| 2 | | 0.5806 | 0.6452 | 0.6935 | 0.7581 | 0.7903 | 0.7742 | | 0.7070 |
| 3 | | 0.5000 | 0.5484 | 0.6774 | 0.6613 | 0.6935 | 0.6935 | | 0.6290 |
| 4 | | 0.4677 | 0.6129 | 0.7742 | 0.6774 | 0.7581 | 0.8065 | | 0.6828 |
| 5 | | 0.4516 | 0.5000 | 0.6290 | 0.5806 | 0.5645 | 0.4839 | | 0.5349 |
| 6 | | 0.4516 | 0.4839 | 0.8387 | 0.8065 | 0.8065 | 0.7903 | | 0.6962 |
| 7 | | 0.4516 | 0.5806 | 0.6613 | 0.7903 | 0.6774 | 0.7097 | | 0.6452 |
| 8 | | 0.7258 | 0.7581 | 0.5000 | 0.6935 | 0.6935 | 0.7097 | | 0.6801 |
| 9 | | 0.6774 | 0.7581 | 0.7742 | 0.6935 | 0.8065 | 0.6774 | | 0.7312 |
| 10 | | 0.4355 | 0.5484 | 0.6452 | 0.6613 | 0.6935 | 0.6290 | | 0.6022 |
| 11 | | 0.4516 | 0.4516 | 0.5645 | 0.5323 | 0.5806 | 0.5484 | | 0.5215 |
| 12 | | 0.5806 | 0.7097 | 0.7097 | 0.7258 | 0.6774 | 0.6935 | | 0.6828 |
| 13 | | 0.6613 | 0.4516 | 0.5645 | 0.6129 | 0.6452 | 0.6452 | | 0.5968 |
| 14 | | 0.6774 | 0.7258 | 0.5645 | 0.7258 | 0.7258 | 0.7742 | | 0.6989 |
| 15 | | 0.6774 | 0.5161 | 0.6290 | 0.7581 | 0.6452 | 0.6613 | | 0.6478 |
| | | | | | | | | | |
| Average | | 0.5548 | 0.5817 | 0.6495 | 0.6839 | 0.6935 | 0.6753 | | 0.6398 |

Table 4.10 case_2, cond_1

Trained with {font #1,...,font #15}.
Tested with (font #1,...,font #15).
Recognition results for the first choice.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.6774 | 0.5323 | 0.5968 | 0.6613 | 0.7097 | 0.5968 | | 0.6290 |
| 2 | | 0.6452 | 0.7097 | 0.8226 | 0.8387 | 0.8065 | 0.8065 | | 0.7715 |
| 3 | | 0.6290 | 0.6613 | 0.7742 | 0.7581 | 0.8065 | 0.7742 | | 0.7339 |
| 4 | | 0.5323 | 0.6935 | 0.8548 | 0.7097 | 0.7742 | 0.8548 | | 0.7366 |
| 5 | | 0.5806 | 0.6129 | 0.7258 | 0.6774 | 0.6129 | 0.5323 | | 0.6237 |
| 6 | | 0.5161 | 0.5323 | 0.8387 | 0.8710 | 0.8548 | 0.8387 | | 0.7419 |
| 7 | | 0.5000 | 0.6290 | 0.7419 | 0.8226 | 0.7742 | 0.8065 | | 0.7124 |
| 8 | | 0.7419 | 0.8065 | 0.5484 | 0.6935 | 0.7097 | 0.7742 | | 0.7124 |
| 9 | | 0.7903 | 0.8226 | 0.8548 | 0.7419 | 0.8387 | 0.7419 | | 0.7984 |
| 10 | | 0.5484 | 0.5968 | 0.7097 | 0.7419 | 0.7742 | 0.6774 | | 0.6747 |
| 11 | | 0.5161 | 0.5161 | 0.5968 | 0.5968 | 0.6613 | 0.6129 | | 0.5833 |
| 12 | | 0.6774 | 0.7903 | 0.8710 | 0.7903 | 0.8065 | 0.7742 | | 0.7849 |
| 13 | | 0.7581 | 0.5968 | 0.6935 | 0.6774 | 0.7258 | 0.7581 | | 0.7016 |
| 14 | | 0.7581 | 0.8548 | 0.6935 | 0.8065 | 0.7742 | 0.8387 | | 0.7876 |
| 15 | | 0.7419 | 0.6129 | 0.7258 | 0.8065 | 0.7258 | 0.7258 | | 0.7231 |
| | | | | | | | | | |
| Average | | 0.6409 | 0.6645 | 0.7366 | 0.7462 | 0.7570 | 0.7409 | | 0.7143 |

Table 4.11 case_2, cond_2

Trained with {font #1,...,font #15}.

Tested with (font #1,...,font #15).

Recognition results for the first choice
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.8871 | 0.8710 | 0.8871 | 0.9032 | 0.9355 | 0.8226 | | 0.8844 |
| 2 | | 0.9032 | 0.9355 | 0.9516 | 0.9677 | 0.9839 | 0.9677 | | 0.9516 |
| 3 | | 0.9194 | 0.9355 | 0.9839 | 0.9516 | 0.9677 | 0.9677 | | 0.9543 |
| 4 | | 0.8387 | 0.9194 | 1.0000 | 0.9839 | 0.9516 | 0.9677 | | 0.9435 |
| 5 | | 0.8710 | 0.9032 | 0.9194 | 0.9032 | 0.8710 | 0.8871 | | 0.8925 |
| 6 | | 0.7419 | 0.7903 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | | 0.9167 |
| 7 | | 0.7419 | 0.8226 | 0.9194 | 0.9677 | 0.9677 | 1.0000 | | 0.9032 |
| 8 | | 0.9355 | 0.9677 | 0.9194 | 0.8871 | 0.9516 | 0.9355 | | 0.9328 |
| 9 | | 0.9032 | 0.9677 | 0.9194 | 0.9677 | 0.9677 | 0.8710 | | 0.9328 |
| 10 | | 0.8871 | 0.9032 | 0.9516 | 0.9355 | 0.9677 | 0.8871 | | 0.9220 |
| 11 | | 0.7581 | 0.9032 | 0.9516 | 0.9355 | 0.9032 | 0.9194 | | 0.8952 |
| 12 | | 0.8871 | 0.9194 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9624 |
| 13 | | 0.9677 | 0.8387 | 0.9194 | 0.9516 | 0.9677 | 0.9677 | | 0.9355 |
| 14 | | 0.9355 | 0.9839 | 0.9194 | 0.9516 | 0.9677 | 0.9839 | | 0.9570 |
| 15 | | 0.9355 | 0.9032 | 0.9355 | 0.9516 | 0.9677 | 0.9677 | | 0.9435 |
| | | | | | | | | | |
| Average | | 0.8742 | 0.9043 | 0.9441 | 0.9484 | 0.9570 | 0.9430 | | 0.9285 |

Table 4.12 case_2, cond_3

Trained with {font #1,...,font #15}.

Tested with (font #1,...,font #15).

Recognition results for the top five choices.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.9032 | 0.9032 | 0.8871 | 0.9355 | 0.9355 | 0.8548 | | 0.9032 |
| 2 | | 0.9032 | 0.9355 | 0.9516 | 0.9677 | 0.9839 | 0.9677 | | 0.9516 |
| 3 | | 0.9194 | 0.9355 | 0.9839 | 0.9516 | 0.9677 | 0.9677 | | 0.9543 |
| 4 | | 0.8387 | 0.9355 | 1.0000 | 0.9839 | 0.9516 | 0.9677 | | 0.9462 |
| 5 | | 0.8871 | 0.9355 | 0.9194 | 0.9032 | 0.8871 | 0.9032 | | 0.9059 |
| 6 | | 0.7742 | 0.8065 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | | 0.9247 |
| 7 | | 0.7419 | 0.8226 | 0.9194 | 0.9677 | 0.9839 | 1.0000 | | 0.9059 |
| 8 | | 0.9355 | 0.9677 | 0.9194 | 0.8871 | 0.9677 | 0.9677 | | 0.9409 |
| 9 | | 0.9032 | 0.9677 | 0.9194 | 0.9677 | 0.9839 | 0.9032 | | 0.9409 |
| 10 | | 0.9032 | 0.9032 | 0.9516 | 0.9516 | 0.9839 | 0.9194 | | 0.9355 |
| 11 | | 0.7581 | 0.9194 | 0.9677 | 0.9677 | 0.9355 | 0.9355 | | 0.9140 |
| 12 | | 0.9194 | 0.9194 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9677 |
| 13 | | 0.9677 | 0.8387 | 0.9355 | 0.9516 | 0.9677 | 0.9677 | | 0.9382 |
| 14 | | 0.9355 | 0.9839 | 0.9355 | 0.9516 | 0.9677 | 0.9839 | | 0.9597 |
| 15 | | 0.9516 | 0.9032 | 0.9355 | 0.9677 | 0.9839 | 1.0000 | | 0.9570 |
| | | | | | | | | | |
| Average | | 0.8828 | 0.9118 | 0.9473 | 0.9548 | 0.9656 | 0.9559 | | 0.9364 |

Table 4.13 case_2, cond_4

Trained with {font #1,...,font #15}.

Tested with (font #1,...,font #15).

Recognition results for the top five choices
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.9516 | 0.9194 | 0.9839 | 0.9677 | 0.9677 | 0.9194 | | 0.9516 |
| 2 | | 0.9677 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9892 |
| 3 | | 0.9839 | 0.9839 | 0.9839 | 0.9839 | 0.9839 | 0.9839 | | 0.9839 |
| 4 | | 0.9355 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9839 |
| 5 | | 0.9677 | 1.0000 | 0.9677 | 0.9516 | 0.9677 | 0.9516 | | 0.9677 |
| 6 | | 0.8548 | 0.9032 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9597 |
| 7 | | 0.8871 | 0.9516 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9704 |
| 8 | | 0.9677 | 1.0000 | 0.9516 | 0.9677 | 1.0000 | 1.0000 | | 0.9812 |
| 9 | | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9516 | | 0.9866 |
| 10 | | 0.9516 | 0.9516 | 0.9839 | 1.0000 | 1.0000 | 0.9839 | | 0.9785 |
| 11 | | 0.9194 | 0.9677 | 1.0000 | 0.9677 | 0.9677 | 1.0000 | | 0.9704 |
| 12 | | 0.9677 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9892 |
| 13 | | 0.9839 | 0.9355 | 0.9194 | 0.9677 | 0.9677 | 0.9677 | | 0.9570 |
| 14 | | 0.9677 | 0.9839 | 0.9677 | 1.0000 | 0.9677 | 1.0000 | | 0.9812 |
| 15 | | 1.0000 | 0.9355 | 0.9839 | 0.9839 | 0.9839 | 1.0000 | | 0.9812 |
| | | | | | | | | | |
| Average | | 0.9516 | 0.9656 | 0.9806 | 0.9860 | 0.9860 | 0.9828 | | 0.9754 |

Table 4.14 case_2, cond_5

Trained with {font #1,...,font #15}.

Tested with (font #1,...,font #15).

Recognition results for the top ten choices.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.9516 | 0.9194 | 0.9839 | 0.9677 | 0.9677 | 0.9194 | | 0.9516 |
| 2 | | 0.9677 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9892 |
| 3 | | 0.9839 | 0.9839 | 0.9839 | 0.9839 | 0.9839 | 0.9839 | | 0.9839 |
| 4 | | 0.9355 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9839 |
| 5 | | 0.9677 | 1.0000 | 0.9839 | 0.9516 | 0.9677 | 0.9677 | | 0.9731 |
| 6 | | 0.8548 | 0.9032 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9597 |
| 7 | | 0.8871 | 0.9516 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9731 |
| 8 | | 0.9677 | 1.0000 | 0.9677 | 0.9677 | 1.0000 | 1.0000 | | 0.9839 |
| 9 | | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9677 | | 0.9892 |
| 10 | | 0.9677 | 0.9516 | 0.9839 | 1.0000 | 1.0000 | 0.9839 | | 0.9812 |
| 11 | | 0.9194 | 0.9677 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9758 |
| 12 | | 0.9677 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9892 |
| 13 | | 0.9839 | 0.9677 | 0.9516 | 0.9839 | 0.9839 | 0.9839 | | 0.9758 |
| 14 | | 1.0000 | 0.9839 | 0.9839 | 1.0000 | 0.9677 | 1.0000 | | 0.9892 |
| 15 | | 1.0000 | 0.9516 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | | 0.9866 |
| | | | | | | | | | |
| Average | | 0.9548 | 0.9688 | 0.9871 | 0.9882 | 0.9892 | 0.9860 | | 0.9790 |

Table 4.15 case_2, cond_6

Trained with {font #1,...,font #15}.

Tested with {font #1,...,font #15}.

Recognition results for the top ten choices
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.5000 | 0.5484 | 0.5323 | 0.5323 | 0.4839 | 0.4516 | | 0.5081 |
| 17 | | 0.5645 | 0.6774 | 0.6774 | 0.5000 | 0.6935 | 0.7419 | | 0.6425 |
| 18 | | 0.4516 | 0.4677 | 0.6774 | 0.6290 | 0.6129 | 0.6129 | | 0.5753 |
| 19 | | 0.5806 | 0.7097 | 0.7742 | 0.7258 | 0.7903 | 0.7419 | | 0.7204 |
| 20 | | 0.5645 | 0.6774 | 0.6935 | 0.7419 | 0.6935 | 0.7581 | | 0.6882 |
| 21 | | 0.3710 | 0.3710 | 0.6452 | 0.5968 | 0.5968 | 0.6129 | | 0.5323 |
| 22 | | 0.6774 | 0.5645 | 0.5968 | 0.5645 | 0.7097 | 0.7258 | | 0.6398 |
| 23 | | 0.5000 | 0.6774 | 0.7581 | 0.8710 | 0.8871 | 0.8065 | | 0.7500 |
| 24 | | 0.6129 | 0.7742 | 0.7742 | 0.7581 | 0.7742 | 0.7742 | | 0.7446 |
| 25 | | 0.3710 | 0.5000 | 0.7097 | 0.7903 | 0.7742 | 0.7097 | | 0.6425 |
| 26 | | 0.5000 | 0.6452 | 0.7419 | 0.7419 | 0.6774 | 0.6290 | | 0.6559 |
| 27 | | 0.6774 | 0.7742 | 0.8065 | 0.6129 | 0.5484 | 0.6129 | | 0.6720 |
| 28 | | 0.6452 | 0.7258 | 0.8226 | 0.8548 | 0.8226 | 0.8548 | | 0.7876 |
| 29 | | 0.6452 | 0.6452 | 0.7581 | 0.8387 | 0.7903 | 0.7903 | | 0.7446 |
| 30 | | 0.5484 | 0.6129 | 0.6935 | 0.7097 | 0.7419 | 0.7742 | | 0.6801 |
| | | | | | | | | | |
| Average | | 0.5473 | 0.6247 | 0.7108 | 0.6978 | 0.7065 | 0.7065 | | 0.6656 |

Table 4.16 case_3, cond_1

Trained with {font #16,...,font #30}.
Tested with (font #16,...,font #30).
Recognition results for the first choice.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.5484 | 0.5806 | 0.5968 | 0.5806 | 0.5161 | 0.5161 | | 0.5565 |
| 17 | | 0.6935 | 0.7903 | 0.8548 | 0.6452 | 0.7903 | 0.7903 | | 0.7608 |
| 18 | | 0.5161 | 0.5161 | 0.7258 | 0.7258 | 0.6774 | 0.7097 | | 0.6452 |
| 19 | | 0.6613 | 0.7419 | 0.8387 | 0.7742 | 0.8387 | 0.8548 | | 0.7849 |
| 20 | | 0.6613 | 0.7581 | 0.7903 | 0.7742 | 0.7419 | 0.8548 | | 0.7634 |
| 21 | | 0.4194 | 0.4194 | 0.7258 | 0.7258 | 0.6613 | 0.6935 | | 0.6075 |
| 22 | | 0.7742 | 0.6452 | 0.7903 | 0.6452 | 0.7581 | 0.8065 | | 0.7366 |
| 23 | | 0.5161 | 0.7258 | 0.8226 | 0.9032 | 0.9355 | 0.8387 | | 0.7903 |
| 24 | | 0.6935 | 0.8710 | 0.8871 | 0.8387 | 0.9032 | 0.9032 | | 0.8495 |
| 25 | | 0.4355 | 0.5806 | 0.7419 | 0.8065 | 0.8226 | 0.8226 | | 0.7016 |
| 26 | | 0.6452 | 0.6935 | 0.7903 | 0.7903 | 0.7581 | 0.7097 | | 0.7312 |
| 27 | | 0.7903 | 0.8387 | 0.8548 | 0.6613 | 0.5968 | 0.6290 | | 0.7285 |
| 28 | | 0.7419 | 0.8065 | 0.8871 | 0.8548 | 0.8548 | 0.9032 | | 0.8414 |
| 29 | | 0.6613 | 0.6774 | 0.8065 | 0.9032 | 0.8387 | 0.8710 | | 0.7930 |
| 30 | | 0.5806 | 0.6452 | 0.7419 | 0.7581 | 0.8065 | 0.8226 | | 0.7258 |
| | | | | | | | | | |
| Average | | 0.6226 | 0.6860 | 0.7903 | 0.7591 | 0.7667 | 0.7817 | | 0.7344 |

Table 4.17 case_3, cond_2

Trained with {font #16,...,font #30}.

Tested with {font #16,...,font #30}.

Recognition results for the first choice
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.8871 | 0.9032 | 0.9032 | 0.8387 | 0.8387 | 0.7903 | | 0.8602 |
| 17 | | 0.8548 | 0.9032 | 0.9677 | 0.9839 | 1.0000 | 1.0000 | | 0.9516 |
| 18 | | 0.9194 | 0.8548 | 0.9032 | 0.9032 | 0.8710 | 0.9032 | | 0.8925 |
| 19 | | 0.8710 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | | 0.9704 |
| 20 | | 0.9032 | 0.9677 | 0.9677 | 0.9677 | 0.9677 | 1.0000 | | 0.9624 |
| 21 | | 0.6452 | 0.8065 | 0.9516 | 0.9516 | 0.9355 | 0.9516 | | 0.8737 |
| 22 | | 0.9355 | 0.8871 | 0.9516 | 0.9194 | 0.9839 | 0.9677 | | 0.9409 |
| 23 | | 0.8710 | 0.9194 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | | 0.9624 |
| 24 | | 0.9355 | 0.9677 | 0.9839 | 0.9839 | 0.9839 | 0.9677 | | 0.9704 |
| 25 | | 0.7419 | 0.8387 | 0.9194 | 0.9516 | 0.9516 | 0.9516 | | 0.8925 |
| 26 | | 0.8710 | 0.9355 | 0.9839 | 0.9839 | 0.9194 | 0.9194 | | 0.9355 |
| 27 | | 0.9677 | 0.9839 | 0.9839 | 0.9677 | 0.9355 | 0.9355 | | 0.9624 |
| 28 | | 0.9194 | 0.9839 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9785 |
| 29 | | 0.8226 | 0.8871 | 0.9516 | 1.0000 | 0.9839 | 1.0000 | | 0.9409 |
| 30 | | 0.8226 | 0.8065 | 0.9355 | 0.9516 | 0.9839 | 0.9839 | | 0.9140 |
| | | | | | | | | | |
| Average | | 0.8645 | 0.9075 | 0.9602 | 0.9591 | 0.9559 | 0.9559 | | 0.9339 |

Table 4.18 case_3, cond_3

Trained with {font #16,...,font #30}.

Tested with (font #16,...,font #30).

Recognition results for the top five choices.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.9032 | 0.9194 | 0.9032 | 0.8710 | 0.8710 | 0.8065 | | 0.8790 |
| 17 | | 0.8548 | 0.9194 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | | 0.9570 |
| 18 | | 0.9194 | 0.8548 | 0.9032 | 0.9032 | 0.8871 | 0.9194 | | 0.8978 |
| 19 | | 0.8710 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | | 0.9731 |
| 20 | | 0.9032 | 0.9677 | 0.9677 | 0.9839 | 0.9677 | 1.0000 | | 0.9651 |
| 21 | | 0.7097 | 0.9032 | 0.9677 | 0.9677 | 0.9516 | 0.9516 | | 0.9086 |
| 22 | | 0.9355 | 0.9032 | 0.9516 | 0.9355 | 0.9839 | 0.9677 | | 0.9462 |
| 23 | | 0.8710 | 0.9355 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | | 0.9651 |
| 24 | | 0.9355 | 0.9677 | 0.9839 | 0.9839 | 0.9839 | 0.9677 | | 0.9704 |
| 25 | | 0.8226 | 0.8871 | 0.9516 | 0.9839 | 0.9839 | 0.9677 | | 0.9328 |
| 26 | | 0.9194 | 0.9516 | 0.9839 | 1.0000 | 0.9516 | 0.9194 | | 0.9543 |
| 27 | | 0.9677 | 0.9839 | 0.9839 | 0.9677 | 0.9516 | 0.9355 | | 0.9651 |
| 28 | | 0.9355 | 0.9839 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9812 |
| 29 | | 0.8387 | 0.8871 | 0.9516 | 1.0000 | 0.9839 | 1.0000 | | 0.9435 |
| 30 | | 0.8226 | 0.8387 | 0.9355 | 0.9516 | 0.9839 | 0.9839 | | 0.9194 |
| | | | | | | | | | |
| Average | | 0.8806 | 0.9258 | 0.9634 | 0.9688 | 0.9656 | 0.9591 | | 0.9439 |

Table 4.19 case_3, cond_4

Trained with (font #16,...,font #30).

Tested with (font #16,...,font #30).

Recognition results for the top five choices
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.9677 | 1.0000 | 0.9677 | 0.9677 | 0.9355 | 0.8710 | | 0.9516 |
| 17 | | 1.0000 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9973 |
| 18 | | 1.0000 | 0.9194 | 0.9355 | 0.9839 | 0.9516 | 0.9677 | | 0.9597 |
| 19 | | 1.0000 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9973 |
| 20 | | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9946 |
| 21 | | 0.7258 | 0.9194 | 0.9839 | 0.9839 | 0.9839 | 0.9677 | | 0.9274 |
| 22 | | 0.9677 | 1.0000 | 0.9839 | 0.9677 | 0.9839 | 0.9839 | | 0.9812 |
| 23 | | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | | 0.9919 |
| 24 | | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9946 |
| 25 | | 0.8387 | 0.9194 | 0.9516 | 1.0000 | 0.9839 | 1.0000 | | 0.9489 |
| 26 | | 0.9516 | 0.9677 | 1.0000 | 1.0000 | 0.9677 | 0.9516 | | 0.9731 |
| 27 | | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9946 |
| 28 | | 0.9839 | 0.9839 | 1.0000 | 0.9839 | 1.0000 | 1.0000 | | 0.9919 |
| 29 | | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 1.0000 |
| 30 | | 0.8871 | 0.8548 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9543 |
| | | | | | | | | | |
| Average | | 0.9484 | 0.9688 | 0.9871 | 0.9925 | 0.9860 | 0.9806 | | 0.9772 |

Table 4.20 case_3, cond_5

Trained with {font #16,...,font #30).
Tested with (font #16,...,font #30).
Recognition results for the top ten choices.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.9677 | 1.0000 | 0.9839 | 0.9677 | 0.9355 | 0.9355 | | 0.9651 |
| 17 | | 1.0000 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9973 |
| 18 | | 1.0000 | 0.9194 | 0.9355 | 1.0000 | 0.9677 | 0.9839 | | 0.9677 |
| 19 | | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 1.0000 |
| 20 | | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9946 |
| 21 | | 0.8226 | 0.9677 | 0.9839 | 1.0000 | 0.9839 | 0.9677 | | 0.9543 |
| 22 | | 0.9839 | 1.0000 | 0.9839 | 0.9677 | 0.9839 | 0.9839 | | 0.9839 |
| 23 | | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | | 0.9919 |
| 24 | | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9946 |
| 25 | | 0.8710 | 0.9355 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9651 |
| 26 | | 0.9516 | 0.9839 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9839 |
| 27 | | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9946 |
| 28 | | 0.9839 | 0.9839 | 1.0000 | 0.9839 | 1.0000 | 1.0000 | | 0.9919 |
| 29 | | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 1.0000 |
| 30 | | 0.8871 | 0.8548 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9543 |
| | | | | | | | | | |
| Average | | 0.9581 | 0.9753 | 0.9903 | 0.9946 | 0.9892 | 0.9882 | | 0.9826 |

Table 4.21 case_3, cond_6

Trained with {font #16,...,font #30}.

Tested with {font #16,...,font #30}.

Recognition results for the top ten choices
not differentiating between upper and lower case characters.

4.6.1.3 Results of the cases where training and testing sets were two different sets (Using the "mean" statistic)

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.4839 | 0.4677 | 0.4355 | 0.4677 | 0.4516 | 0.4194 | | 0.4543 |
| 17 | | 0.5645 | 0.6452 | 0.7258 | 0.5323 | 0.7581 | 0.7903 | | 0.6694 |
| 18 | | 0.3387 | 0.3871 | 0.5484 | 0.6290 | 0.5968 | 0.5645 | | 0.5108 |
| 19 | | 0.5323 | 0.6452 | 0.7097 | 0.6613 | 0.7742 | 0.7097 | | 0.6720 |
| 20 | | 0.5645 | 0.6613 | 0.7097 | 0.7419 | 0.7258 | 0.7903 | | 0.6989 |
| 21 | | 0.3065 | 0.4355 | 0.6290 | 0.7419 | 0.6774 | 0.5645 | | 0.5591 |
| 22 | | 0.7419 | 0.6613 | 0.7258 | 0.6774 | 0.7419 | 0.7258 | | 0.7124 |
| 23 | | 0.5484 | 0.6290 | 0.8065 | 0.8226 | 0.9194 | 0.7258 | | 0.7419 |
| 24 | | 0.6129 | 0.7581 | 0.7903 | 0.7742 | 0.7097 | 0.6935 | | 0.7231 |
| 25 | | 0.3226 | 0.4032 | 0.6129 | 0.6774 | 0.6774 | 0.6613 | | 0.5591 |
| 26 | | 0.5000 | 0.6129 | 0.7419 | 0.6613 | 0.5968 | 0.5484 | | 0.6102 |
| 27 | | 0.6774 | 0.7419 | 0.8065 | 0.7097 | 0.5161 | 0.5645 | | 0.6694 |
| 28 | | 0.6290 | 0.7258 | 0.7742 | 0.8065 | 0.8226 | 0.8548 | | 0.7688 |
| 29 | | 0.5323 | 0.5968 | 0.7258 | 0.7258 | 0.7258 | 0.7097 | | 0.6694 |
| 30 | | 0.4677 | 0.5323 | 0.5645 | 0.6613 | 0.7097 | 0.7097 | | 0.6075 |
| | | | | | | | | | |
| Average | | 0.5215 | 0.5935 | 0.6871 | 0.6860 | 0.6935 | 0.6688 | | 0.6418 |

Table 4.22 case_4, cond_1

Trained with {font #1,...,font #15).
Tested with (font #16,...,font #30).
Recognition results for the first choice.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.5161 | 0.5000 | 0.5000 | 0.5645 | 0.5323 | 0.4839 | | 0.5161 |
| 17 | | 0.6935 | 0.7742 | 0.8871 | 0.6613 | 0.8226 | 0.8387 | | 0.7796 |
| 18 | | 0.4194 | 0.4677 | 0.5968 | 0.6613 | 0.6452 | 0.6774 | | 0.5780 |
| 19 | | 0.5968 | 0.7097 | 0.7903 | 0.7258 | 0.8065 | 0.8065 | | 0.7392 |
| 20 | | 0.6613 | 0.7419 | 0.8065 | 0.8065 | 0.7742 | 0.8871 | | 0.7796 |
| 21 | | 0.3548 | 0.4677 | 0.6613 | 0.8065 | 0.7419 | 0.6774 | | 0.6183 |
| 22 | | 0.8065 | 0.7419 | 0.8387 | 0.7258 | 0.8065 | 0.8065 | | 0.7876 |
| 23 | | 0.5806 | 0.7097 | 0.8387 | 0.8387 | 0.9677 | 0.7742 | | 0.7849 |
| 24 | | 0.6935 | 0.8387 | 0.8871 | 0.8226 | 0.8226 | 0.7903 | | 0.8091 |
| 25 | | 0.3871 | 0.5161 | 0.6613 | 0.7097 | 0.7258 | 0.7419 | | 0.6237 |
| 26 | | 0.5806 | 0.6935 | 0.8065 | 0.7097 | 0.6935 | 0.6129 | | 0.6828 |
| 27 | | 0.7903 | 0.8548 | 0.8710 | 0.7419 | 0.5645 | 0.5968 | | 0.7366 |
| 28 | | 0.7258 | 0.7742 | 0.8387 | 0.8226 | 0.8548 | 0.9032 | | 0.8199 |
| 29 | | 0.5806 | 0.6613 | 0.8065 | 0.8387 | 0.7742 | 0.7903 | | 0.7419 |
| 30 | | 0.5323 | 0.5806 | 0.6290 | 0.7097 | 0.7419 | 0.7903 | | 0.6640 |
| | | | | | | | | | |
| Average | | 0.5946 | 0.6688 | 0.7613 | 0.7430 | 0.7516 | 0.7452 | | 0.7108 |

Table 4.23 case_4, cond_2

Trained with {font #1,...,font #15}.

Tested with {font #16,...,font #30}.

Recognition results for the first choice
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.7742 | 0.8710 | 0.8710 | 0.7903 | 0.7419 | 0.7742 | | 0.8038 |
| 17 | | 0.9032 | 0.9355 | 0.9677 | 0.9839 | 1.0000 | 1.0000 | | 0.9651 |
| 18 | | 0.8387 | 0.8226 | 0.9032 | 0.8871 | 0.8871 | 0.8387 | | 0.8629 |
| 19 | | 0.8548 | 0.8548 | 0.9839 | 0.9516 | 0.9677 | 0.9516 | | 0.9274 |
| 20 | | 0.8710 | 0.9677 | 0.9839 | 0.9839 | 0.9839 | 0.9839 | | 0.9624 |
| 21 | | 0.5484 | 0.7581 | 0.9194 | 0.9516 | 0.9355 | 0.9032 | | 0.8360 |
| 22 | | 0.9194 | 0.9516 | 0.9839 | 0.9355 | 0.9839 | 0.9839 | | 0.9597 |
| 23 | | 0.8710 | 0.9194 | 0.9677 | 0.9839 | 1.0000 | 0.9355 | | 0.9462 |
| 24 | | 0.9194 | 0.9516 | 0.9677 | 0.9677 | 0.9677 | 0.9516 | | 0.9543 |
| 25 | | 0.6774 | 0.7903 | 0.9032 | 0.9194 | 0.9677 | 0.9516 | | 0.8683 |
| 26 | | 0.8065 | 0.8548 | 0.9677 | 0.9677 | 0.9355 | 0.8548 | | 0.8978 |
| 27 | | 0.9516 | 0.9839 | 0.9677 | 0.9516 | 0.9355 | 0.8871 | | 0.9462 |
| 28 | | 0.9194 | 0.9516 | 0.9677 | 0.9677 | 0.9839 | 1.0000 | | 0.9651 |
| 29 | | 0.8226 | 0.8710 | 0.9194 | 1.0000 | 0.9516 | 0.9677 | | 0.9220 |
| 30 | | 0.7903 | 0.8226 | 0.9194 | 0.9355 | 0.9355 | 0.9516 | | 0.8925 |
| | | | | | | | | | |
| Average | | 0.8312 | 0.8871 | 0.9462 | 0.9452 | 0.9452 | 0.9290 | | 0.9140 |

Table 4.24 case_4, cond_3

Trained with {font #1,...,font #15}.
Tested with (font #16,...,font #30).
Recognition results for the top five choices.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.8065 | 0.8710 | 0.8710 | 0.8065 | 0.7742 | 0.7903 | | 0.8199 |
| 17 | | 0.9032 | 0.9516 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9731 |
| 18 | | 0.8548 | 0.8387 | 0.9032 | 0.9194 | 0.9194 | 0.8871 | | 0.8871 |
| 19 | | 0.8548 | 0.8710 | 0.9839 | 0.9516 | 0.9839 | 0.9677 | | 0.9355 |
| 20 | | 0.8710 | 0.9677 | 0.9839 | 0.9839 | 0.9839 | 0.9839 | | 0.9624 |
| 21 | | 0.6290 | 0.8548 | 0.9516 | 0.9677 | 0.9516 | 0.9194 | | 0.8790 |
| 22 | | 0.9355 | 0.9516 | 0.9839 | 0.9355 | 0.9839 | 0.9839 | | 0.9624 |
| 23 | | 0.8710 | 0.9355 | 0.9677 | 0.9839 | 1.0000 | 0.9355 | | 0.9489 |
| 24 | | 0.9355 | 0.9516 | 0.9677 | 0.9677 | 0.9677 | 0.9839 | | 0.9624 |
| 25 | | 0.7258 | 0.8387 | 0.9355 | 0.9839 | 0.9839 | 0.9677 | | 0.9059 |
| 26 | | 0.8871 | 0.8871 | 0.9839 | 0.9839 | 0.9516 | 0.8548 | | 0.9247 |
| 27 | | 0.9516 | 0.9839 | 0.9839 | 0.9677 | 0.9355 | 0.8871 | | 0.9516 |
| 28 | | 0.9355 | 0.9516 | 0.9677 | 0.9677 | 0.9839 | 1.0000 | | 0.9677 |
| 29 | | 0.8387 | 0.8710 | 0.9194 | 1.0000 | 0.9516 | 0.9677 | | 0.9247 |
| 30 | | 0.7903 | 0.8387 | 0.9194 | 0.9355 | 0.9355 | 0.9677 | | 0.8978 |
| | | | | | | | | | |
| Average | | 0.8527 | 0.9043 | 0.9538 | 0.9570 | 0.9538 | 0.9398 | | 0.9269 |

Table 4.25 case_4, cond_4

Trained with {font #1,...,font #15}.

Tested with (font #16,...,font #30).

Recognition results for the top five choices
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.9677 | 0.9516 | 0.9677 | 0.9355 | 0.9194 | 0.8871 | | 0.9382 |
| 17 | | 0.9839 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9919 |
| 18 | | 0.9516 | 0.9194 | 0.9516 | 0.9839 | 0.9677 | 0.9677 | | 0.9570 |
| 19 | | 0.9032 | 0.9355 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9677 |
| 20 | | 0.9677 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9892 |
| 21 | | 0.7419 | 0.9032 | 0.9839 | 0.9839 | 1.0000 | 0.9516 | | 0.9274 |
| 22 | | 0.9677 | 0.9839 | 1.0000 | 0.9839 | 0.9839 | 0.9839 | | 0.9839 |
| 23 | | 0.9677 | 0.9516 | 1.0000 | 1.0000 | 1.0000 | 0.9677 | | 0.9812 |
| 24 | | 0.9677 | 0.9839 | 1.0000 | 0.9839 | 0.9839 | 0.9839 | | 0.9839 |
| 25 | | 0.8548 | 0.8871 | 0.9032 | 0.9516 | 0.9839 | 0.9839 | | 0.9274 |
| 26 | | 0.9194 | 0.9194 | 1.0000 | 0.9677 | 0.9355 | 0.9355 | | 0.9462 |
| 27 | | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9677 | 0.9355 | | 0.9839 |
| 28 | | 0.9677 | 0.9839 | 1.0000 | 0.9677 | 1.0000 | 1.0000 | | 0.9866 |
| 29 | | 0.9516 | 0.9516 | 0.9839 | 1.0000 | 0.9677 | 0.9839 | | 0.9731 |
| 30 | | 0.8871 | 0.8548 | 0.9355 | 0.9516 | 0.9839 | 0.9839 | | 0.9328 |
| | | | | | | | | | |
| Average | | 0.9333 | 0.9473 | 0.9806 | 0.9796 | 0.9774 | 0.9699 | | 0.9647 |

Table 4.26 case_4, cond_5

Trained with {font #1,...,font #15}.
Tested with (font #16,...,font #30).
Recognition results for the top ten choices.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 16 | | 0.9677 | 0.9677 | 0.9677 | 0.9516 | 0.9355 | 0.9032 | | 0.9489 |
| 17 | | 0.9839 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9946 |
| 18 | | 0.9516 | 0.9194 | 0.9516 | 1.0000 | 0.9839 | 0.9839 | | 0.9651 |
| 19 | | 0.9032 | 0.9516 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9704 |
| 20 | | 0.9677 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9892 |
| 21 | | 0.8387 | 0.9516 | 0.9839 | 1.0000 | 1.0000 | 0.9516 | | 0.9543 |
| 22 | | 0.9677 | 0.9839 | 1.0000 | 0.9839 | 0.9839 | 0.9839 | | 0.9839 |
| 23 | | 0.9677 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | 0.9677 | | 0.9839 |
| 24 | | 0.9677 | 0.9839 | 1.0000 | 0.9839 | 0.9839 | 0.9839 | | 0.9839 |
| 25 | | 0.9032 | 0.8871 | 0.9355 | 0.9839 | 0.9839 | 1.0000 | | 0.9489 |
| 26 | | 0.9516 | 0.9516 | 1.0000 | 0.9839 | 0.9677 | 0.9516 | | 0.9677 |
| 27 | | 1.0000 | 1.0000 | 1.0000 | 1.0000 | 0.9677 | 0.9355 | | 0.9839 |
| 28 | | 0.9839 | 0.9839 | 1.0000 | 0.9677 | 1.0000 | 1.0000 | | 0.9892 |
| 29 | | 0.9516 | 0.9516 | 1.0000 | 1.0000 | 0.9677 | 0.9839 | | 0.9758 |
| 30 | | 0.8871 | 0.8548 | 0.9355 | 0.9516 | 0.9839 | 0.9839 | | 0.9328 |
| | | | | | | | | | |
| Average | | 0.9462 | 0.9559 | 0.9849 | 0.9860 | 0.9817 | 0.9742 | | 0.9715 |

Table 4.27 case_4, cond_6

Trained with {font #1,...,font #15}.

Tested with (font #16,...,font #30).

Recognition results for the top ten choices
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.4677 | 0.4355 | 0.4516 | 0.5000 | 0.5806 | 0.4194 | | 0.4758 |
| 2 | | 0.4677 | 0.5323 | 0.6290 | 0.5968 | 0.6452 | 0.6774 | | 0.5914 |
| 3 | | 0.4355 | 0.4032 | 0.5484 | 0.5323 | 0.6129 | 0.5968 | | 0.5215 |
| 4 | | 0.5000 | 0.6290 | 0.7258 | 0.6935 | 0.6935 | 0.7742 | | 0.6694 |
| 5 | | 0.5161 | 0.5161 | 0.5323 | 0.5645 | 0.5000 | 0.4194 | | 0.5081 |
| 6 | | 0.3871 | 0.5484 | 0.8226 | 0.7903 | 0.8548 | 0.7419 | | 0.6909 |
| 7 | | 0.4194 | 0.5323 | 0.6774 | 0.8226 | 0.7742 | 0.7097 | | 0.6559 |
| 8 | | 0.7258 | 0.6935 | 0.6129 | 0.7258 | 0.8065 | 0.6935 | | 0.7097 |
| 9 | | 0.7419 | 0.8065 | 0.7742 | 0.6452 | 0.8226 | 0.6774 | | 0.7446 |
| 10 | | 0.3387 | 0.4516 | 0.5645 | 0.5968 | 0.5161 | 0.5484 | | 0.5027 |
| 11 | | 0.4355 | 0.5645 | 0.6129 | 0.5161 | 0.5000 | 0.5161 | | 0.5242 |
| 12 | | 0.4355 | 0.5968 | 0.6613 | 0.5968 | 0.5645 | 0.6774 | | 0.5887 |
| 13 | | 0.5806 | 0.4355 | 0.5000 | 0.5323 | 0.5484 | 0.6290 | | 0.5376 |
| 14 | | 0.6774 | 0.7258 | 0.5645 | 0.6613 | 0.6452 | 0.6935 | | 0.6613 |
| 15 | | 0.7097 | 0.5645 | 0.6452 | 0.7097 | 0.5484 | 0.5968 | | 0.6290 |
| | | | | | | | | | |
| Average | | 0.5226 | 0.5624 | 0.6215 | 0.6323 | 0.6409 | 0.6247 | | 0.6007 |

Table 4.28 case_5, cond_1

Trained with {font #16,...,font #30).
Tested with (font #1,...,font #15).
Recognition results for the first choice.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.5968 | 0.5645 | 0.5484 | 0.5968 | 0.6290 | 0.5000 | | 0.5726 |
| 2 | | 0.5968 | 0.6452 | 0.7258 | 0.7097 | 0.6935 | 0.7097 | | 0.6801 |
| 3 | | 0.5484 | 0.5484 | 0.6774 | 0.6935 | 0.7419 | 0.6935 | | 0.6505 |
| 4 | | 0.5806 | 0.7258 | 0.8226 | 0.7581 | 0.7581 | 0.8548 | | 0.7500 |
| 5 | | 0.6129 | 0.5968 | 0.6452 | 0.6774 | 0.5645 | 0.4839 | | 0.5968 |
| 6 | | 0.4677 | 0.5484 | 0.8387 | 0.8387 | 0.8871 | 0.8226 | | 0.7339 |
| 7 | | 0.5161 | 0.5806 | 0.7903 | 0.8226 | 0.8387 | 0.8387 | | 0.7312 |
| 8 | | 0.7419 | 0.7581 | 0.6452 | 0.7419 | 0.8226 | 0.8226 | | 0.7554 |
| 9 | | 0.8065 | 0.8548 | 0.8548 | 0.7419 | 0.9032 | 0.7903 | | 0.8253 |
| 10 | | 0.4677 | 0.5645 | 0.6774 | 0.6935 | 0.6935 | 0.6452 | | 0.6237 |
| 11 | | 0.5000 | 0.6129 | 0.6452 | 0.5968 | 0.5968 | 0.6129 | | 0.5941 |
| 12 | | 0.5806 | 0.7258 | 0.8226 | 0.7419 | 0.7097 | 0.7581 | | 0.7231 |
| 13 | | 0.6935 | 0.5645 | 0.5968 | 0.6129 | 0.6613 | 0.7258 | | 0.6425 |
| 14 | | 0.7581 | 0.8710 | 0.6290 | 0.8065 | 0.7581 | 0.8226 | | 0.7742 |
| 15 | | 0.7581 | 0.6613 | 0.7742 | 0.7742 | 0.6129 | 0.6452 | | 0.7043 |
| | | | | | | | | | |
| Average | | 0.6151 | 0.6548 | 0.7129 | 0.7204 | 0.7247 | 0.7151 | | 0.6905 |

Table 4.29 case_5, cond_2

Trained with {font #16,...,font #30}.

Tested with (font #1,...,font #15).

Recognition results for the first choice
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.8065 | 0.8387 | 0.8871 | 0.8548 | 0.9032 | 0.8065 | | 0.8495 |
| 2 | | 0.8710 | 0.8710 | 0.9355 | 0.9194 | 0.9839 | 0.9677 | | 0.9247 |
| 3 | | 0.8226 | 0.8548 | 0.9355 | 0.8871 | 0.9516 | 0.9355 | | 0.8978 |
| 4 | | 0.8387 | 0.9032 | 1.0000 | 0.9839 | 0.9677 | 0.9839 | | 0.9462 |
| 5 | | 0.8548 | 0.8548 | 0.9032 | 0.8710 | 0.8548 | 0.8226 | | 0.8602 |
| 6 | | 0.7258 | 0.8065 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | | 0.9167 |
| 7 | | 0.7258 | 0.8226 | 0.9194 | 0.9839 | 0.9677 | 1.0000 | | 0.9032 |
| 8 | | 0.9355 | 0.9355 | 0.9355 | 0.9194 | 0.9516 | 0.9677 | | 0.9409 |
| 9 | | 0.9355 | 0.9677 | 0.9516 | 0.9516 | 0.9839 | 0.9194 | | 0.9516 |
| 10 | | 0.8387 | 0.8387 | 0.9032 | 0.9032 | 0.9194 | 0.8710 | | 0.8790 |
| 11 | | 0.8065 | 0.8871 | 0.9194 | 0.8871 | 0.9032 | 0.8548 | | 0.8763 |
| 12 | | 0.9194 | 0.9355 | 1.0000 | 0.9677 | 0.9839 | 1.0000 | | 0.9677 |
| 13 | | 0.9355 | 0.8226 | 0.8871 | 0.9032 | 0.9355 | 0.9516 | | 0.9059 |
| 14 | | 0.9839 | 1.0000 | 0.9194 | 0.9677 | 0.9677 | 1.0000 | | 0.9731 |
| 15 | | 0.9516 | 0.9032 | 0.9516 | 0.9355 | 0.9677 | 0.9839 | | 0.9489 |
| | | | | | | | | | |
| Average | | 0.8634 | 0.8828 | 0.9344 | 0.9290 | 0.9495 | 0.9376 | | 0.9161 |

Table 4.30 case_5, cond_3

Trained with {font #16,...,font #30}.
Tested with (font #1,...,font #15).
Recognition results for the top five choices.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.8387 | 0.8548 | 0.8871 | 0.8710 | 0.9194 | 0.8226 | | 0.8656 |
| 2 | | 0.8710 | 0.8710 | 0.9355 | 0.9194 | 0.9839 | 0.9677 | | 0.9247 |
| 3 | | 0.8387 | 0.8548 | 0.9355 | 0.9032 | 0.9516 | 0.9516 | | 0.9059 |
| 4 | | 0.8387 | 0.9194 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9543 |
| 5 | | 0.8710 | 0.8871 | 0.9194 | 0.8710 | 0.8871 | 0.8387 | | 0.8790 |
| 6 | | 0.7581 | 0.8065 | 0.9677 | 1.0000 | 1.0000 | 1.0000 | | 0.9220 |
| 7 | | 0.7419 | 0.8387 | 0.9194 | 0.9839 | 0.9839 | 1.0000 | | 0.9113 |
| 8 | | 0.9355 | 0.9516 | 0.9516 | 0.9355 | 0.9677 | 1.0000 | | 0.9570 |
| 9 | | 0.9355 | 0.9677 | 0.9516 | 0.9677 | 0.9839 | 0.9194 | | 0.9543 |
| 10 | | 0.8548 | 0.8548 | 0.9194 | 0.9194 | 0.9677 | 0.9032 | | 0.9032 |
| 11 | | 0.8065 | 0.9355 | 0.9516 | 0.9355 | 0.9516 | 0.9355 | | 0.9194 |
| 12 | | 0.9516 | 0.9355 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9758 |
| 13 | | 0.9355 | 0.8387 | 0.9355 | 0.9194 | 0.9677 | 0.9839 | | 0.9301 |
| 14 | | 1.0000 | 1.0000 | 0.9516 | 0.9839 | 0.9839 | 1.0000 | | 0.9866 |
| 15 | | 0.9516 | 0.9032 | 0.9677 | 0.9677 | 0.9839 | 1.0000 | | 0.9624 |
| | | | | | | | | | |
| Average | | 0.8753 | 0.8946 | 0.9462 | 0.9441 | 0.9667 | 0.9538 | | 0.9301 |

Table 4.31 case_5, cond_4

Trained with {font #16,...,font #30}.

Tested with {font #1,...,font #15}.

Recognition results for the top five choices
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.9194 | 0.9032 | 0.9839 | 0.9677 | 0.9355 | 0.9032 | | 0.9355 |
| 2 | | 0.9355 | 0.9516 | 0.9677 | 0.9839 | 1.0000 | 1.0000 | | 0.9731 |
| 3 | | 0.9839 | 0.9677 | 0.9839 | 0.9677 | 0.9839 | 0.9839 | | 0.9785 |
| 4 | | 0.9355 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9839 |
| 5 | | 0.9677 | 0.9516 | 0.9516 | 0.9355 | 0.9839 | 0.9355 | | 0.9543 |
| 6 | | 0.9032 | 0.9032 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9677 |
| 7 | | 0.8871 | 0.9355 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9704 |
| 8 | | 0.9839 | 1.0000 | 0.9677 | 0.9839 | 1.0000 | 1.0000 | | 0.9892 |
| 9 | | 0.9677 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | 0.9516 | | 0.9812 |
| 10 | | 0.9516 | 0.9032 | 0.9677 | 1.0000 | 0.9839 | 0.9677 | | 0.9624 |
| 11 | | 0.9032 | 0.9355 | 1.0000 | 0.9677 | 0.9677 | 0.9677 | | 0.9570 |
| 12 | | 0.9677 | 0.9677 | 1.0000 | 0.9839 | 1.0000 | 1.0000 | | 0.9866 |
| 13 | | 0.9839 | 0.9516 | 0.9516 | 0.9516 | 0.9677 | 0.9516 | | 0.9597 |
| 14 | | 1.0000 | 1.0000 | 0.9516 | 1.0000 | 1.0000 | 1.0000 | | 0.9919 |
| 15 | | 0.9839 | 0.9677 | 0.9677 | 0.9839 | 1.0000 | 1.0000 | | 0.9839 |
| | | | | | | | | | |
| Average | | 0.9516 | 0.9548 | 0.9785 | 0.9817 | 0.9871 | 0.9763 | | 0.9717 |

Table 4.32 case_5, cond_5

Trained with {font #16,...,font #30).
Tested with (font #1,...,font #15).
Recognition results for the top ten choices.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.9194 | 0.9032 | 0.9839 | 0.9677 | 0.9355 | 0.9032 | | 0.9355 |
| 2 | | 0.9355 | 0.9516 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | | 0.9758 |
| 3 | | 0.9839 | 0.9677 | 0.9839 | 0.9677 | 0.9839 | 0.9839 | | 0.9785 |
| 4 | | 0.9516 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9866 |
| 5 | | 0.9677 | 0.9677 | 0.9677 | 0.9355 | 0.9839 | 0.9516 | | 0.9624 |
| 6 | | 0.9032 | 0.9032 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9677 |
| 7 | | 0.8871 | 0.9516 | 1.0000 | 1.0000 | 1.0000 | 1.0000 | | 0.9731 |
| 8 | | 0.9839 | 1.0000 | 0.9677 | 0.9839 | 1.0000 | 1.0000 | | 0.9892 |
| 9 | | 0.9677 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | 0.9677 | | 0.9839 |
| 10 | | 0.9677 | 0.9194 | 0.9677 | 1.0000 | 0.9839 | 0.9677 | | 0.9677 |
| 11 | | 0.9516 | 0.9677 | 1.0000 | 0.9839 | 1.0000 | 0.9839 | | 0.9812 |
| 12 | | 0.9677 | 0.9677 | 1.0000 | 0.9839 | 1.0000 | 1.0000 | | 0.9866 |
| 13 | | 0.9839 | 0.9677 | 0.9677 | 0.9839 | 0.9839 | 0.9839 | | 0.9785 |
| 14 | | 1.0000 | 1.0000 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | | 0.9973 |
| 15 | | 0.9839 | 0.9839 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | | 0.9892 |
| | | | | | | | | | |
| Average | | 0.9570 | 0.9624 | 0.9849 | 0.9849 | 0.9903 | 0.9817 | | 0.9769 |

Table 4.33 case_5, cond_6

Trained with {font #16,...,font #30}.

Tested with {font #1,...,font #15}.

Recognition results for the top ten choices
not differentiating between upper and lower case characters.

4.6.1.4 Results of the cases where training and testing sets were the same (Using the "median or the "standard deviation" statistics)

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.4516 | 0.4355 | 0.4194 | 0.4516 | 0.4355 | 0.4032 | | 0.4328 |
| 2 | | 0.4516 | 0.5161 | 0.5484 | 0.5000 | 0.5968 | 0.6290 | | 0.5403 |
| 3 | | 0.4032 | 0.3710 | 0.5323 | 0.4677 | 0.5968 | 0.5484 | | 0.4866 |
| 4 | | 0.5323 | 0.6129 | 0.7097 | 0.7258 | 0.7258 | 0.7581 | | 0.6774 |
| 5 | | 0.4677 | 0.4839 | 0.5323 | 0.4839 | 0.3226 | 0.3710 | | 0.4435 |
| 6 | | 0.4032 | 0.5161 | 0.8226 | 0.7742 | 0.7581 | 0.6613 | | 0.6559 |
| 7 | | 0.3710 | 0.4194 | 0.5806 | 0.7097 | 0.6774 | 0.6452 | | 0.5672 |
| 8 | | 0.7097 | 0.6935 | 0.4516 | 0.6774 | 0.6452 | 0.6452 | | 0.6371 |
| 9 | | 0.5968 | 0.6774 | 0.7419 | 0.5968 | 0.6613 | 0.6452 | | 0.6532 |
| 10 | | 0.3548 | 0.4839 | 0.5323 | 0.5161 | 0.5000 | 0.4839 | | 0.4785 |
| 11 | | 0.4355 | 0.5000 | 0.4839 | 0.5000 | 0.3871 | 0.3871 | | 0.4489 |
| 12 | | 0.5000 | 0.5323 | 0.6935 | 0.6613 | 0.5323 | 0.5484 | | 0.5780 |
| 13 | | 0.5323 | 0.4032 | 0.5484 | 0.4839 | 0.5161 | 0.5968 | | 0.5134 |
| 14 | | 0.6613 | 0.7903 | 0.5484 | 0.6774 | 0.6452 | 0.6290 | | 0.6586 |
| 15 | | 0.5806 | 0.5161 | 0.6129 | 0.6613 | 0.4355 | 0.3710 | | 0.5296 |
| 16 | | 0.5484 | 0.5645 | 0.4839 | 0.4355 | 0.4355 | 0.4032 | | 0.4785 |
| 17 | | 0.5645 | 0.6129 | 0.5968 | 0.4516 | 0.5000 | 0.6290 | | 0.5591 |
| 18 | | 0.3710 | 0.3871 | 0.5000 | 0.5323 | 0.5484 | 0.4516 | | 0.4651 |
| 19 | | 0.5161 | 0.6452 | 0.7097 | 0.5484 | 0.6613 | 0.6129 | | 0.6156 |
| 20 | | 0.5161 | 0.5484 | 0.7258 | 0.6129 | 0.5806 | 0.6290 | | 0.6022 |
| 21 | | 0.3226 | 0.3871 | 0.6129 | 0.5323 | 0.4355 | 0.5323 | | 0.4704 |
| 22 | | 0.6452 | 0.5000 | 0.6129 | 0.4677 | 0.5968 | 0.6452 | | 0.5780 |
| 23 | | 0.4677 | 0.6290 | 0.7097 | 0.7581 | 0.8226 | 0.6290 | | 0.6694 |
| 24 | | 0.6452 | 0.7419 | 0.7742 | 0.5645 | 0.6774 | 0.7258 | | 0.6882 |
| 25 | | 0.2903 | 0.4839 | 0.5161 | 0.6129 | 0.7097 | 0.5484 | | 0.5269 |
| 26 | | 0.4677 | 0.5484 | 0.6935 | 0.6129 | 0.5161 | 0.5161 | | 0.5591 |
| 27 | | 0.6129 | 0.6129 | 0.7742 | 0.5806 | 0.5000 | 0.4839 | | 0.5941 |
| 28 | | 0.6129 | 0.6935 | 0.7742 | 0.7097 | 0.6613 | 0.6452 | | 0.6828 |
| 29 | | 0.5645 | 0.6452 | 0.8065 | 0.6935 | 0.6774 | 0.6452 | | 0.6720 |
| 30 | | 0.4516 | 0.4839 | 0.5161 | 0.5161 | 0.6129 | 0.5806 | | 0.5269 |
| Average | | 0.5016 | 0.5478 | 0.6188 | 0.5839 | 0.5790 | 0.5667 | | 0.5663 |

Table 4.34 case_1, cond_1 (Median)

Trained with {font #1,...,font #30}.
Tested with {font #1,...,font #30}.
Recognition results for the first choice.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.9355 | 0.8548 | 0.9355 | 0.9677 | 0.9355 | 0.8710 | | 0.9167 |
| 2 | | 0.9516 | 0.9355 | 0.9839 | 0.9516 | 1.0000 | 1.0000 | | 0.9704 |
| 3 | | 0.9839 | 0.9677 | 0.9677 | 0.9677 | 0.9839 | 0.9839 | | 0.9758 |
| 4 | | 0.8871 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | 0.9839 | | 0.9758 |
| 5 | | 0.9355 | 0.9194 | 0.9516 | 0.9194 | 0.8710 | 0.8710 | | 0.9113 |
| 6 | | 0.8226 | 0.8871 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | | 0.9489 |
| 7 | | 0.8548 | 0.9194 | 0.9677 | 0.9839 | 0.9839 | 1.0000 | | 0.9516 |
| 8 | | 0.9677 | 1.0000 | 0.9516 | 0.9677 | 0.9677 | 0.9839 | | 0.9731 |
| 9 | | 0.9677 | 0.9839 | 0.9677 | 1.0000 | 1.0000 | 0.9516 | | 0.9785 |
| 10 | | 0.9516 | 0.8871 | 0.9677 | 1.0000 | 0.9839 | 0.9355 | | 0.9543 |
| 11 | | 0.8548 | 0.9516 | 0.9839 | 0.9355 | 0.9516 | 0.9194 | | 0.9328 |
| 12 | | 0.9516 | 0.9677 | 1.0000 | 0.9677 | 1.0000 | 1.0000 | | 0.9812 |
| 13 | | 0.9677 | 0.9516 | 0.9194 | 0.9516 | 0.9677 | 0.9677 | | 0.9543 |
| 14 | | 0.9839 | 1.0000 | 0.9516 | 0.9839 | 1.0000 | 1.0000 | | 0.9866 |
| 15 | | 0.9677 | 0.9355 | 0.9677 | 0.9677 | 0.9839 | 0.9839 | | 0.9677 |
| 16 | | 0.8710 | 0.9355 | 0.9355 | 0.9355 | 0.8710 | 0.7903 | | 0.8898 |
| 17 | | 0.9355 | 0.9839 | 0.9839 | 0.9839 | 1.0000 | 1.0000 | | 0.9812 |
| 18 | | 0.9194 | 0.8871 | 0.9032 | 0.9516 | 0.9032 | 0.9516 | | 0.9194 |
| 19 | | 0.9355 | 0.9839 | 1.0000 | 1.0000 | 1.0000 | 0.9839 | | 0.9839 |
| 20 | | 0.9516 | 1.0000 | 0.9839 | 0.9516 | 0.9839 | 1.0000 | | 0.9785 |
| 21 | | 0.7419 | 0.8871 | 0.9677 | 0.9516 | 0.9194 | 0.9032 | | 0.8952 |
| 22 | | 0.9677 | 0.9516 | 0.9677 | 0.9839 | 0.9839 | 0.9677 | | 0.9704 |
| 23 | | 0.9194 | 0.9516 | 0.9839 | 1.0000 | 1.0000 | 0.9839 | | 0.9731 |
| 24 | | 0.9677 | 0.9839 | 1.0000 | 0.9677 | 0.9839 | 0.9839 | | 0.9812 |
| 25 | | 0.8226 | 0.8548 | 0.9194 | 0.9516 | 0.9839 | 0.9677 | | 0.9167 |
| 26 | | 0.9194 | 0.9355 | 1.0000 | 0.9839 | 0.9355 | 0.9194 | | 0.9489 |
| 27 | | 1.0000 | 1.0000 | 0.9839 | 0.9516 | 0.9677 | 0.9516 | | 0.9758 |
| 28 | | 0.9516 | 0.9677 | 1.0000 | 0.9839 | 0.9839 | 1.0000 | | 0.9812 |
| 29 | | 0.9677 | 0.9677 | 0.9839 | 1.0000 | 0.9839 | 1.0000 | | 0.9839 |
| 30 | | 0.8548 | 0.8548 | 0.9355 | 0.9516 | 0.9839 | 0.9677 | | 0.9247 |
| Average | | 0.9237 | 0.9435 | 0.9688 | 0.9704 | 0.9699 | 0.9602 | | 0.9561 |

Table 4.35 case_1, cond_5 (Median)

Trained with {font #1,...,font #30}.

Tested with {font #1,...,font #30}.

Recognition results for the top ten choices
not differentiating between upper and lower case characters.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|--------------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.1935 | 0.1935 | 0.2419 | 0.1290 | 0.1774 | 0.1452 | | 0.1801 |
| 2 | | 0.3548 | 0.4032 | 0.2903 | 0.2581 | 0.1452 | 0.2419 | | 0.2823 |
| 3 | | 0.3226 | 0.2581 | 0.3871 | 0.3065 | 0.2903 | 0.2419 | | 0.3011 |
| 4 | | 0.3871 | 0.5000 | 0.4355 | 0.3387 | 0.3065 | 0.4355 | | 0.4005 |
| 5 | | 0.2903 | 0.4032 | 0.2903 | 0.2258 | 0.2097 | 0.1452 | | 0.2608 |
| 6 | | 0.3226 | 0.3710 | 0.5484 | 0.5000 | 0.3710 | 0.3871 | | 0.4167 |
| 7 | | 0.2419 | 0.3710 | 0.4677 | 0.3710 | 0.3548 | 0.3871 | | 0.3656 |
| 8 | | 0.5484 | 0.5323 | 0.3548 | 0.2903 | 0.2581 | 0.3548 | | 0.3898 |
| 9 | | 0.4677 | 0.5323 | 0.4839 | 0.3387 | 0.2581 | 0.2742 | | 0.3925 |
| 10 | | 0.3387 | 0.3387 | 0.3871 | 0.2581 | 0.2258 | 0.2742 | | 0.3038 |
| 11 | | 0.2258 | 0.2581 | 0.2742 | 0.2742 | 0.1774 | 0.1613 | | 0.2285 |
| 12 | | 0.3871 | 0.3548 | 0.3871 | 0.3710 | 0.3065 | 0.3226 | | 0.3548 |
| 13 | | 0.3871 | 0.2581 | 0.2419 | 0.2419 | 0.1935 | 0.3065 | | 0.2715 |
| 14 | | 0.4839 | 0.4677 | 0.2742 | 0.3710 | 0.4516 | 0.4032 | | 0.4086 |
| 15 | | 0.4516 | 0.3387 | 0.4032 | 0.4516 | 0.1774 | 0.1613 | | 0.3306 |
| 16 | | 0.3548 | 0.3871 | 0.2258 | 0.2581 | 0.2097 | 0.1935 | | 0.2715 |
| 17 | | 0.3226 | 0.4032 | 0.4032 | 0.2581 | 0.2581 | 0.3710 | | 0.3360 |
| 18 | | 0.2903 | 0.3065 | 0.3548 | 0.2742 | 0.2903 | 0.2419 | | 0.2930 |
| 19 | | 0.3871 | 0.4516 | 0.3871 | 0.2581 | 0.3065 | 0.3710 | | 0.3602 |
| 20 | | 0.3710 | 0.4032 | 0.4032 | 0.3710 | 0.3387 | 0.3387 | | 0.3710 |
| 21 | | 0.3065 | 0.3387 | 0.3710 | 0.3387 | 0.2419 | 0.3387 | | 0.3226 |
| 22 | | 0.3710 | 0.3710 | 0.4032 | 0.3065 | 0.2742 | 0.3065 | | 0.3387 |
| 23 | | 0.3710 | 0.4194 | 0.3871 | 0.5000 | 0.4355 | 0.3710 | | 0.4140 |
| 24 | | 0.4677 | 0.6129 | 0.4194 | 0.2581 | 0.3226 | 0.3226 | | 0.4005 |
| 25 | | 0.1774 | 0.2742 | 0.3387 | 0.3065 | 0.3387 | 0.1935 | | 0.2715 |
| 26 | | 0.3387 | 0.4032 | 0.4516 | 0.1935 | 0.2419 | 0.2419 | | 0.3118 |
| 27 | | 0.3871 | 0.5000 | 0.4194 | 0.2258 | 0.1774 | 0.2258 | | 0.3226 |
| 28 | | 0.4839 | 0.4516 | 0.4677 | 0.4194 | 0.3065 | 0.3226 | | 0.4086 |
| 29 | | 0.4355 | 0.4677 | 0.5000 | 0.4194 | 0.2903 | 0.4194 | | 0.4220 |
| 30 | | 0.3065 | 0.3548 | 0.2742 | 0.2903 | 0.3065 | 0.3710 | | 0.3172 |
| Average | | 0.3591 | 0.3909 | 0.3758 | 0.3134 | 0.2747 | 0.2957 | | 0.3349 |

Table 4.36 case_1, cond_1 (Standard deviation)

Trained with {font #1,...,font #30}.
Tested with {font #1,...,font #30}.
Recognition results for the first choice.

| | font size | 7 | 8 | 9 | 10 | 11 | 12 | | Average |
|---------|-----------|--------|--------|--------|--------|--------|--------|--|---------|
| font | | | | | | | | | |
| 1 | | 0.7742 | 0.6774 | 0.7097 | 0.7581 | 0.7097 | 0.6935 | | 0.7204 |
| 2 | | 0.8387 | 0.8710 | 0.8548 | 0.8226 | 0.8710 | 0.8226 | | 0.8468 |
| 3 | | 0.8065 | 0.8548 | 0.8548 | 0.8226 | 0.8548 | 0.8226 | | 0.8360 |
| 4 | | 0.8710 | 0.8387 | 0.8226 | 0.8548 | 0.8710 | 0.8548 | | 0.8522 |
| 5 | | 0.7097 | 0.7419 | 0.7258 | 0.7097 | 0.7097 | 0.7419 | | 0.7231 |
| 6 | | 0.8226 | 0.7903 | 0.9355 | 0.9194 | 0.8710 | 0.8387 | | 0.8629 |
| 7 | | 0.7258 | 0.7258 | 0.8226 | 0.8226 | 0.8387 | 0.8065 | | 0.7903 |
| 8 | | 0.8710 | 0.8548 | 0.6935 | 0.7097 | 0.8226 | 0.7742 | | 0.7876 |
| 9 | | 0.8871 | 0.9032 | 0.8871 | 0.8871 | 0.7419 | 0.7742 | | 0.8468 |
| 10 | | 0.8065 | 0.7581 | 0.8871 | 0.8226 | 0.8710 | 0.7419 | | 0.8145 |
| 11 | | 0.6290 | 0.6613 | 0.6452 | 0.6290 | 0.5968 | 0.5968 | | 0.6263 |
| 12 | | 0.8065 | 0.8710 | 0.8871 | 0.8871 | 0.8710 | 0.8226 | | 0.8575 |
| 13 | | 0.7581 | 0.7258 | 0.7903 | 0.7419 | 0.7581 | 0.8065 | | 0.7634 |
| 14 | | 0.8387 | 0.9194 | 0.8226 | 0.7903 | 0.8548 | 0.7903 | | 0.8360 |
| 15 | | 0.8710 | 0.7903 | 0.8065 | 0.8065 | 0.7742 | 0.7742 | | 0.8038 |
| 16 | | 0.6774 | 0.6774 | 0.6774 | 0.6774 | 0.5968 | 0.5968 | | 0.6505 |
| 17 | | 0.8548 | 0.8387 | 0.8710 | 0.8065 | 0.8065 | 0.7903 | | 0.8280 |
| 18 | | 0.7097 | 0.6935 | 0.7581 | 0.6935 | 0.6129 | 0.6774 | | 0.6909 |
| 19 | | 0.8226 | 0.8226 | 0.8387 | 0.8387 | 0.8710 | 0.8548 | | 0.8414 |
| 20 | | 0.8226 | 0.9032 | 0.9194 | 0.8226 | 0.8548 | 0.8387 | | 0.8602 |
| 21 | | 0.6935 | 0.7258 | 0.8710 | 0.8548 | 0.8065 | 0.8387 | | 0.7984 |
| 22 | | 0.8387 | 0.8548 | 0.8387 | 0.8226 | 0.8710 | 0.8387 | | 0.8441 |
| 23 | | 0.8065 | 0.8548 | 0.8226 | 0.9194 | 0.9355 | 0.8710 | | 0.8683 |
| 24 | | 0.8871 | 0.8548 | 0.8387 | 0.8065 | 0.8548 | 0.7903 | | 0.8387 |
| 25 | | 0.6613 | 0.7097 | 0.7903 | 0.8065 | 0.7903 | 0.7419 | | 0.7500 |
| 26 | | 0.7581 | 0.8065 | 0.8065 | 0.7419 | 0.7581 | 0.6452 | | 0.7527 |
| 27 | | 0.9032 | 0.9355 | 0.8710 | 0.7903 | 0.7097 | 0.7419 | | 0.8253 |
| 28 | | 0.8548 | 0.8548 | 0.9032 | 0.8548 | 0.8710 | 0.8548 | | 0.8656 |
| 29 | | 0.8871 | 0.9032 | 0.9194 | 0.8387 | 0.8710 | 0.9032 | | 0.8871 |
| 30 | | 0.8226 | 0.7903 | 0.7258 | 0.6935 | 0.8226 | 0.7903 | | 0.7742 |
| Average | | 0.8005 | 0.8070 | 0.8199 | 0.7984 | 0.8016 | 0.7812 | | 0.8014 |

Table 4.37 case_1, cond_5 (Standard deviation)

Trained with {font #1,...,font #30}.
Tested with {font #1,...,font #30}.
Recognition results for the top ten choices
not differentiating between upper and lower case characters.

4.6.1.5 Analysis of the results of the character recognition subsystem

There are several points as well as many others that can be deduced from the generated results:

- 1) The first choice statistics shows a promising feature extraction methodology.
- 2) The top five and top ten statistic show a promising character recognition methodology that could be coupled with a contextual verification method mentioned in the section 3.1.5 within an application.
- 3) Although generally the recognition results get better as the font size gets bigger, sometimes this was not the case between two consecutive intermediate font sizes. This probably happens due to the property of the digital grid where the features are extracted from.
- 4) The bullring effect inherent in models that average a data is not apparent in these results ((case_1, cond_1), (case_2, cond_1), and (case3, cond_1)). On average, it actually performs better as more data are introduced. This could be the effect of the filter introduced during the distance computation of an unknown sample from a model (See section 4.4.3). New samples strengthen the features of a model and the filter would take out the variances of those features. However, the results are reversed slowly as more possibilities are considered as part of solution (five or ten possibility conditions). This could be the result of introduction of far alternate distances in the process of selecting five or ten best choices.

4.6.2 Results of the modified EPAM word search

It was sufficient to test the modified search algorithm with a sample font under case_1 with cond_5 (training set:{font #1,...,font #30}, testing set:{font #1,..., font #30}, cond_5:Recognition results for the top ten choices). I chose font #30 for this experiment. Under the specified settings the range of the recognition results varies wide enough to observe the behaviour of the modified search algorithm. Total number of words tested was 23692.

| Font size | Character recognition rate | First choice word recognized | Word recognized within alternate choices | Did not recognize at all |
|-----------|----------------------------|------------------------------|--|--------------------------|
| 7 point | 88.7% | 3040 (12.8%) | 2103 (8.9%) | 18549 (78%) |
| 8 point | 87.1% | 3321 (14%) | 1142 (4.8%) | 19229 (81.2%) |
| 9 point | 95.2% | 14865 (62.7%) | 1425 (6%) | 7402 (31.2%) |
| 10 point | 96.8% | 22440 (94.7%) | 900 (3.8%) | 352 (1.5%) |
| 11 point | 100% | 23359 (98.60%) | 333 (1.4%) | 0 |
| 12 point | 100% | 22997 (97%) | 695 (3%) | 0 |

Table 4.38 Recognition results for the modified search algorithm.

(Percentage values are approximate)

4.6.2.1 Analysis of the results of the modified search

The results for the modified search algorithm indicate:

- 1) If the features are detectable to some threshold (in this thesis top ten choices), then there is a high probability the objects consisting of those features will be recognizable.
- 2) Most of the alternate responses of the system on the larger font sizes (two last rows of the table) were correct on the second choice. The difference between the first choice and the second choice usually was only one character, where that character was very similar to the one mistaken. The mistake originally came from the character recognition subsystem that determines the order of hypotheses that are made about the class of a given character. In words that some characters are very similar in shape (features), it is very likely that the two words could be mistaken without any further contextual constraint. This phenomenon that is evident for humans could happen under conditions where the character in question (feature) is degraded enough to be mistaken for another character and therefore the word that contains it is recognized for another similar word.
- 3) As can be seen, the results could be very good if all the leaf nodes similar to the stimulus object are reached and tested as a hypothesis (the last two rows of the table). The completeness of the nodes reached could be determined from the recognition results of the character recognition subsystem. It can be seen that the two samples (two last rows of the table) possess a 100% character recognition rate for the top 10 choices, which implies that the access path to the most optimal leaf

node in the EPAM net is explored. For the other cases (the first three rows of the table) the results diminish very fast, as the possibility of not reaching the most optimal node is increased. This result definitely shows the need for redundant access paths (See section 2.5). A combination of redundant path and alternate path access would make this model a formidable one for recognition.

5. Conclusion

The results obtained in the section 4.6 show a modified EPAM model has a great potential for a recognition system. Coupled with its segmentation capability via the chunking mechanism, this model could be used in many text recognition applications. The modifications proposed or tested are not confined to an OCR machine as the modifications are generic as the model itself is. It would be interesting to implement the redundant paths described in the section 2.5 and determine its effect on the recognition results. Further, the segmentation capability of the modified model should prove very interesting to test. Also, the system should be tried with words where its characters inputted as complex object consist of a list of features themselves. Subsequently, the system should be tested with lines of text inputted which contains a nested list of words, characters, and character features. Alternatively, lines of text consisting of character features could be inputted and the system tested for its integrated segmentation of characters and words from the inputted lines.

The feature extraction described in this thesis has also shown promising

results that could be used with the modified EPAM model or other post processing methods described in section 3.1.5.

I extremely enjoyed the study of the EPAM model and similar systems as their generality reaches many aspects and fields in science. The quest for modelling human perception and intelligence has been very interesting in the past decades as few general models and architectures have been proposed. Most models fall into one of the two groups of symbolic models or connectionist models with few others some kind in between. Most of the arguments have been on the parallelism or serialism of the processes involved in cognition. Although the consensus is that a mix of the two is needed in different "levels" of abstraction in the mind's architecture. In EPAM the process is assumed serial in nature. The objects' features are processed serially in recognition and learning. These processes require attention that is serial in nature. Alternate paths devised in this thesis may be followed in parallel for a single feature, however, the processing of different features is done serially. The redundant paths are explored serially from a test node as following each redundant path would be the result of testing a different set of features.

The modified search algorithm was devised on the "Intelligence" notion where search of all the possible paths is not done but only few possibly optimal ones. This concept has been carried over from the AI field. Indeed, the modified search algorithm is similar to many existing search algorithms in that field.

EPAM IV is currently under construction in Carnegie-Mellon University

where efforts are being made to introduce redundant paths into its architecture.

I shall end with the notion that many nodes have been traversed and there are interesting nodes to go to next in this knowledge space! .

References

- [1] Simon, H. A., *Models of Thought*, (In Chapter 7: *The information-Processing Explanation of Gestalt Phenomena*) New Haven and London, Yale University Press, vol. 2, 1979.
- [2] Bancilhon, Francois, and Khoshafian, Setrag, *A Calculus for Complex Objects*, Proceedings of the ACM PODS Conference, pp. 53-59, 1986.
- [3] Anderson, John R., *The architecture of cognition*, Harvard University Press, 1983.
- [4] Biederman, Irving, *Human Image Understanding: Recent Research and a Theory*, Comp. Vis. Graph. Image Proc., vol. 32, pp. 29-73, 1985.
- [5] Wang, Qing Ren, *Decision Tree Approach to Pattern Recognition Problems in a Large Character Set*, Doctoral thesis, Department of Computer Science, Concordia University, 1984.
- [6] Feigenbaum, E. A., *An information processing theory of verbal learning*, Report P-1817. Santa Monica, Calif.: The RAND Corporation, 1959.
- [7] Feigenbaum, E. A., *The simulation of verbal learning behaviour*, Proceedings of the 1961 Western Joint Computer Conference, vol. 19, pp. 121-129, 1961.

- [8] Feigenbaum, E. A., and Simon, H. A., *Forgetting in an association memory*, Reprints of the 1961 National Conference of the Association for Computing Machinery, vol. 16, pp. 2C2-2C5, 1961.
- [9] Feigenbaum, E. A., and Simon, H. A., *Brief notes on the EPAM theory of verbal learning*, in Charles N. Cofer and Barbara S. Musgrave (eds.) *Verbal Behaviour and Learning*, pp. 333-335, New York, McGraw-Hill, 1963.
- [10] Feigenbaum, E. A., and Simon, H. A., *Elementary Perceiver and Memorizer: Review of Experiments*, in Hoggatt, A. C., and Balderston (Eds.), *Symposium on Simulation Models: Methodology and Application to the Behavioral Sciences*, Cincinnati: South-Western Publishing Company, 1963.
- [11] Feigenbaum, E. A., and Simon, H. A., *Generalization of an Elementary Perceiving and Memorizing Machine*, Information Processing, Proceedings of IFIP Congress 62. Amsterdam: North-Holland Publishing Co., pp. 401-406, 1962.
- [12] Simon H. A., and Feigenbaum, E. A., *An Information-Processing Theory of Some Effects of Similarity, Familiarization, and Meaningfulness in Verbal Learning*, Journal of Verbal Learning and Verbal Behavior, vol. 3, pp. 385-396, 1964.
- [13] Simon, H. A., and Barenfeld, M., *Information Processing analysis of Perceptual Processes in problem Solving*, Psychological Review, vol. 76, pp. 473-483, 1969.
- [14] Chase William G., and Simon H. A., *Perception in Chess*, Cognitive Psychology, vol. 4, no. 1, pp. 55-81, Jan. 1973.
- [15] Simon, H. A., and Gilmartin Kevin, *A Simulation of Memory for Chess Positions*, Cognitive Psychology, vol. 5, pp. 29-46, 1973.

- [16] Feigenbaum, E. A., and Simon H. A., *EPAM-like Models of Recognition and Learning*, Cognitive Science, vol. 8, pp. 305-336, 1984.
- [17] Richman, Howard B., and Simon, H. A., *Context Effects in Letter Perception: Comparison of Two Theories*, Psychological Review, vol. 96, no. 3, pp. 417-432, 1989.
- [18] McClelland, J. L., and Rumelhart, D. E., *An Interactive Activation model of Context Effects in Letter Perception: Part 1. An Account of Basic Findings*, Psychological Review, vol. 88, pp. 375-407, 1981.
- [19] Rumelhart, D. E., and McClelland, J. L., *An Interactive Activation model of Context Effects in Letter Perception: Part 2: The Contextual Enhancement Effect and Some Texts and Extensions of the Model*, Psychological Review, vol. 89, pp. 60-94, 1982.
- [20] Rumelhart, D. E., McClelland, J. L., and the PDP group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, and 2, The MIT Press, Cambridge, MA, 1986.
- [21] Rosenbloom, P. S., and Newell, A., *Learning by Chunking: A Production System model of Practice*, (Tech. Rep No. 82-135), Carnegie-Mellon University, Computer Science Department, 1982.
- [22] Simon, H. A., Hayes, J. R., *Understanding Written Problem Instructions*, in Gregg, L. W. (Ed.), *Knowledge and Cognition* (pp. 167-200). Hillsdale, NJ: Erlbaum, 1974.
- [23] Newell, A., and Simon, H. A., *Human Problem Solving*, Englewood Cliffs, NJ: Prentice-Hall, 1972.

- [24] Feigenbaum, E. A., and Simon, H. A., *A Theory of the Serial Position Effect*, Brit. J. Psychol., vol. 53, no. 3, pp. 307-320, 1962.
- [25] Barsalou L. W., and Bower, G. H., *Discrimination Nets as Psychological Models*, Cognitive Science, vol. 8, pp. 1-26, 1984.
- [26] McCrary, J. W., & Hunter, W. S. *Serial Position Curves in Verbal Learning*, Science, vol. 117, pp. 131-134, 1953.
- [27] Johnston, J. C., and McClelland, J. L., *Visual Factors in Word Perception*, Perception & Psychophysics, vol. 10, pp. 365-370, 1973.
- [28] Johnston, J. C., *The role of Contextual Constraints in the Perception of Letters in Words*, Unpublished Doctoral Dissertation, University of Pennsylvania, 1974.
- [29] Johnston, J. C., *A Test of the Sophisticated Guessing Theory of Word Perception*, Cognitive Psychology, vol. 10, pp. 123-154, 1978.
- [30] Bugelski, B. R., *Presentation Time, Total Time, Mediation in Paired-Associate Learning*, Journal of Experimental Psychology, vol. 63, pp. 409-412, 1962.
- [31] Miller, G. A., *The magical number seven, plus or minus two*, Psychological Review, vol. 63, pp. 81-97, 1956.
- [32] Cattell, J. M., *The Inertia of the Eye and Brain*, In A. T. Poffenberger (Ed.), *James McKean Cattell: Man of Science* (vol. 1:Psychological research, pp. 26-40). Lancaster, PA: Science Press, 1974 (Original work published 1885).
- [33] Reicher, G. M., *Perceptual Recognition as a Function of Meaningfulness of Stimulus Material*, Journal of Experimental Psychology, vol. 81, pp. 274-280, 1969.

- [34] Kolodner, J. L., *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*, Hillsdale, NJ, Lawrence Erlbaum, 1984.
- [35] Rosenbloom, P. S., and Newell, A., *Learning by Chunking: A Production System model of Practice* (Tech. Rep. No. 82-135). Carnegie-Mellon University Computer Science Department.
- [36] Laird, J. E., Newell, A., and Rosenbloom, P. S., *Soar: An architecture for general intelligence*, *Artificial Intelligence*, vol. 33, no. 1, pp.1-64, 1987.
- [37] Rosenbloom, P. S., Laird, J. E., Newell, A., *The SOAR papers: research on integrated intelligence*, The MIT Press, Cambridge, MA, 1993.
- [38] Makino, H., *Representation and Segmentation of Document Images*, Proc. IEEE Conf. Computer Vision Pattern Recognition, pp. 291-296, 1983.
- [39] Fletcher, A. L., and Kasturi, R., *A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images*, *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 10, no. 6, pp. 910-918, November 1988.
- [40] Akiyama, T., and Hagita, N., *Automated Entry System for Printed Documents*, *Pattern Recognition*, vol. 23, no. 11, pp. 1141-1154, 1990.
- [41] Elliman, D. G., *A review of Segmentation and Contextual Analysis Techniques for Text Recognition*, *Pattern Recognition*, vol. 23, no. 3/4, pp. 337-346, 1990.
- [42] Dunn, C. E., and Wang, P. S. P., *Character Segmentation Techniques for Handwritten Text - A survey*, *Proceedings 11th IAPR International Conference on Pattern Recognition*, Vol. II, Conference B: Pattern Recognition Methodology and Systems, 1992.

- [43] Shridhar, M., and Badreldin, A., *Recognition of Isolated and Simply Connected Handwritten Numerals*, Pattern Recognition, vol. 19 , no. 1, pp. 1-12, 1986.
- [44] Shridhar, M., and Badreldin, A., *Context-directed Segmentation Algorithm for Handwritten Numeral Strings*, Image Vision Comput., vol. 5, no. 1, pp. 3-9, 1987.
- [45] Ahmad, P., and Suen, C. Y., *Segmentation of Unconstrained Handwritten Numeric Postal Zip Codes*, Proc. Int. Conf. Pattern Recognition, pp. 545-547, 1982.
- [46] Narasimha, M. S., and Westall, J., M., *Vertex Directed Segmentation of Handwritten Numerals*, Pattern Recognition, vol. 26, no. 10, pp. 1473-1486, 1993.
- [47] Casey, R. G., and Nagy, G., *Recursive Segmentation and Classification of Composite Character Patterns*, Proc. 6th Int. Conf. Pattern Recognition vol. 2, pp. 1023-1025, 1982.
- [48] Gonzalez, R. C., and Wintz, P., *Digital Image Processing*, Second Edition, Addison Wesley, 1987.
- [49] Asai, K., and Tsuji, Y., *Character Image Segmentation*, Proc. Soc. Photo-opt. Instrum. Engrs. vol. 504, pp. 2-9, 1984,
- [50] Suen, C. Y., and Berthod, M., and Mori, S., *Automatic Recognition of Handprinted Characters - The State of the Art*, Proc. IEEE, vol. 68, no. 4, pp. 469-487, 1980.
- [51] Suen, C. Y., *Character Recognition by Computer and Applications*, Handbook of Pattern Recognition and Image Processing, T. Y. Young and K. S. FU (Eds.), pp. 569-586, Academic Press, New York, 1986.

- [52] Mantas, J., *An Overview of Character Recognition Methodologies*, Pattern Recognition, vol. 19, no. 6, pp. 425-430, 1986.
- [53] Mori, S, Suen, C. Y., and Yamamoto, *Historical Review of OCR Research and Development*, Proc. IEEE, vol. 80, no. 7, pp. 1029-1058, 1992.
- [54] Suen, C. Y., *n-Gram statistics for Natural Language Understanding and Text Processing*, IEEE Transactions on Pattern Recognition and Machine Analysis, vol. PAMI-1, no. 2, April 1979.
- [55] Viterbi, A. J., *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm*, IEEE Trans. Information Theory, vol. 13, no. 2, pp. 260-269, 1979.
- [56] Forney, G. D., *The Viterbi Algorithm*, Proc. IEEE, vol. 61, no. 3, pp. 268-278, 1973.
- [57] Neuhoff, D. L., *The Viterbi Algorithm as an aid in text recognition*, IEEE Trans. Information Theory, vol. 21, pp. 222-226, 1975.
- [58] Shinghal, R., and Toussaint, G. T., *Experiments in Text Recognition with the modified Viterbi Algorithm*, IEEE Trans Pattern Analysis Machine Intell., vol. 1, no. 2, pp. 184-193, 1975.
- [59] Raviv, J., *Decision making in Markov chains applied to the problem of pattern recognition*, IEEE Trans. Information Theory, vol. 3, no. 4, pp. 536-551, 1967.
- [60] Shinghal, R., Rosenberg D., and Toussaint, G. T., *A simplified heuristic version of a recursive Bayes algorithm for using context in text recognition*, IEEE Trans. Systems Man Cybernetics, vol. 8, no. 5, pp. 412-414, 1975.

- [61] Goshtasby, A., and Ehrich, R. W., *Contextual word recognition using probabilistic relaxation labelling*, Pattern Recognition, vol 21, no. 5, pp. 455-462, 1988.
- [62] Damerau, F. J., *A technique for computer detection and correction of spelling errors*, Commun. ACM, vol. 7, no. 5, pp. 171-176, 1964.
- [63] Levenshtein, V. I., *Binary codes capable of correcting deletions, insertions, and reversals*, Sov. Phys. Dokl., vol. 10, no. 8, pp. 707-710, 1966.
- [64] Okuda, T., Tanaka, E., and Kasai, T., *A method for the correction of garbled words based on the Levenshtein Metric*, IEEE Trans. Comput., vol. 25, no. 2, pp. 172-178, 1976.
- [65] Tanaka, E., and Kasai, T., *Synchronisation and substitution error correcting codes for the Levenshtein Metric*, IEEE Trans. Information Theory, vol. 22, no. 2, pp. 172-178, 1976.
- [66] Wagner, R. A., and Fischer, M. J., *The string to string correction problem*, J. ACM, vol. 21, no. 1, pp. 168-173, 1974.
- [67] Lowrance, R., and Wagner, R. A., *An extension of the string to string correction problem*, J. ACM, vol. 22, no. 2, pp. 173-183, 1975.
- [68] Masek, W. J., and Paterson, M. S., *A faster algorithm computing string edit distance*, J. Comput. System. Sci., vol. 20, no. 1, pp. 18-31, 1980.
- [69] Hall, P. A. V., and Dowling, G. R., *Approximate string matching*, ACM Comput. Surv., vol. 12, no. 4, pp. 381-401, 1980.
- [70] Kashyap, R. L., and Oommen, B. J., *Spelling correction using probabilistic methods*, pattern recognition Lett., vol. 2, no. 3, pp. 147-154, 1984.

- [71] Knuth, D. E., Digital searching, *The Art of Computer Programming*, vol. 3, pp. 481-4999. Addison-Wesley, Reading, MA, 1973.
- [72] Blumer, A., Blumer, J., Haussler, D., Ehrenfeucht, A., Chen, M. T., and Seiferas, J., *The smallest automation recognising the subwords of a text*, *Theor. Comput. Sci.*, vol. 40, pp. 31-55, 1985.
- [73] Appel, A. W., and Jacobson, G. J., *The world's fastest scrabble program*, *Commun. ACM*, vol. 31, no. 5, pp. 572-579, 1988.
- [74] Sheil, B. A., *Median split trees: a fast lookup technique for frequently occurring keys*, *Commun. ACM*, vol. 21, no. 11, pp. 947-958, 1978.
- [75] Takahashi, H., Itoh, N., Amano, T., and Yamashita, A., *A spelling correction method and its application to an OCR system*, *Pattern Recognition*, vol. 23, no.3/4, pp. 368-377, 1990.
- [76] Kohonen, T., and Reuhkala, E., *A very fast associative method for the recognition and correction of misspelt words, based on redundant hash addressing*, *Proc. 4th Int. Joint Conf. Pattern Recognition*, pp. 807-809, 1978.
- [77] McIlroy, M. D., *Development of a spelling list*, *IEEE Trans. on Commun.*, vol. COM-30, no. 1, January, 1982.
- [78] Shinghal, R., Toussaint, G. T., *A bottom-up and top-down approach to using context in text recognition*, *Int. J. Man-Machine Stud.* vol. 11, pp. 201-212, 1979.
- [79] Shinghal, R., *A hybrid algorithm for contextual text recognition*, *Pattern Recognition*, vol. 16, no. 2, pp. 261-267, 1983.

- [80] Sirhari, S. N., Hull, J. J., and Choudhuri, R., *Integrating diverse knowledge sources in text recognition*, ACM Trans. Office Information sys. vol. 1, no. 1, pp. 68-87, 1983.
- [81] Hull, J. J. Srihari, S. N., and Choudhuri, *An integrated algorithm for text recognition: comparison with a cascaded algorithm*, IEEE Trans. Pattern Analysis Machine Intell., vol. 5, no. 4, pp. 384-395, 1983.
- [82] Shinha, R. M. K., and Prasada, B., *Visual text recognition through contextual processing*, Pattern Recognition, vol. 21, no. 5, pp. 463-479, 1988.
- [83] Ronse, C., and Devijver, P. A., *Connected components in binary images: the detection problem*, New York, Wiley, 1984.
- [84] Wahl, F. M., Wong, K. Y., and Casey R. G., *Block segmentation and text extraction in mixed text/image documents*, Comput. Graphics Image Processing, vol. 20, pp. 375-390, 1982.
- [85] Simon, H. A., *How big is a chunk?*, Science, vol. 183, pp. 482-488, 1974.

Appendix A (The fonts used in the experiment)



ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Aero in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font arial in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Avant Guard in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Baskerton in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Buckingham in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Capelli in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Carnegie in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Century in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Classic Typewriter in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Corporate Condensed in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Dateline in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Eterna in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Futuri in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Garamond in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Gazette in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Gettysburg in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Gibraltar in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Jewel in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Joulliard in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Katrina in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Letter Gothic in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Obelisk in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Oxford in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

0123456789

This is an example of the font Padua in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

This is an example of the font Pica in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

This is an example of the font Prestige in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

This is an example of the font Rockland in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

This is an example of the font Souvienne in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

This is an example of the font Times in 12 points.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789

This is an example of the font Top Hat in 12 points.

Glossary

| | |
|---|--------|
| ACT* | 31 |
| Anchor-point | 15, 22 |
| Back-propagation | 6 |
| Dictionary-lookup techniques | 40 |
| DISCRIMINATE | 15 |
| Elementary Perceiver And Memorizer | 1, 5 |
| EPAM | 5 |
| FAMILIARIZE | 15 |
| Familirazation parameter | 16 |
| Feature | 1 |
| Features | 9 |
| Forgetting | 24 |
| IAM | 6 |
| Image | 8, 9 |
| Ineffectiveness of Letter Constraints | 27 |
| Interactive Activation Model | 6 |
| Intuition | 1 |
| Long-term memory | 7 |
| Markov methods | 39 |
| N-ary | 9 |
| N-grams | 39 |
| Object | 1, 9 |
| OCR | 3, 36 |
| Optical Character Recognition | 3, 36 |
| Oscillation | 24 |
| Parrale distributed precessing | 6 |
| PDP | 6 |
| Probablistic relaxtion | 40 |
| Properties | 9 |
| Recursive Bayes Algorithm | 40 |
| Remembering | 24 |
| Serial-position | 22 |
| Short-term memory | 8 |
| SOAR | 31 |
| Stimuli | 7, 9 |
| Subobjects | 9 |
| Viterbi Algorithm | 40 |
| Word-Letter effect | 27 |
| Word-Superiority effect | 27 |