Efficient High-Order Hidden Markov Modelling

by

Johan A. du Preez

Dissertation approved for the degree of

Doctor of Philosophy

in

Electrical and Electronic Engineering

at the

University of Stellenbosch

November 1997

Promoters: Dr. E. Barnard

Dr. D.M. Weber

Declaration

I, the undersigned, hereby declare that the work contained in this dissertation is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: __________________ Date: ___________

Abstract

Currently, first-order hidden Markov models (HMMs) form the backbone around which most automatic speech processing applications are built. Their higher-order extensions are known to be more powerful, but, due to their complexity and computational demands, they are seldomly used. It is the purpose of this work to advance their application

In this work we unify HMMs of all orders by deriving and proving the ORder rEDucing (ORED) algorithm. This algorithm will reduce any higher-order HMM (also mixed-order) to an equivalent first-order representation. This *makes it possible to process any higher-order HMM using known first-order algorithms*, thereby making unnecessary the current approach of extending specific HMM algorithms to specific higher orders. It also provides an alternative theoretical basis to reason about high-order HMMs. From this perspective high-order transition probabilities are simply powerful mathematical specifications of first-order topology. We use this insight to explain old topologies and to design new ones.

We address computational concerns by developing the Fast Incremental Training (FIT) algorithm. This algorithm avoids training redundant high-order probabilities by noting which lower-order transition probabilities are zero. This considerably reduces the memory and processor requirements during training. In addition, the resultant models have far fewer parameters and generalise better on previously unseen data.

To show the practical applicability of our methodology we apply it to automatic language recognition. We find that it compares well with systems that require expensive transcribed databases (our system does not require this).

Opsomming

Tans vorm eerste orde verskuilde Markov modelle (HMMs) die ruggraat waarom meeste outomatiese spraakverwerkingstelsels gebou word. Dit is bekend dat hoër orde modelle kragtiger is, maar vanweë hulle kompleksiteit en verwerkings-vereistes word hulle selde gebruik. Hierdie werk het ten doel om hulle toepassing te bevorder.

Met die daarstelling en bewys van die orde-reduserings-algoritme (ORED), word HMMs van alle ordes verenig. Hierdie algoritme reduseer enige hoër orde HMM (ook gemengde orde) na ‘n ekwivalente eerste orde voorstelling. Dit maak dit moontlik om enige hoër orde HMM met behulp van eerste orde algoritmes te verwerk. Daardeur word die huidige benadering waarmee spesifieke HMM algoritmes tot spesifieke ordes uitgebrei word, oorbodig. Dit verskaf ook ‘n alternatiewe teoretiese basis waaruit oor hoër orde HMMs geredeneer kan word. Vanuit hierdie perspektief is hoër orde oorgangswaarskynlikhede maar net kragtige wiskundige spesifikasies van eerste orde topologië. Hierdie insig verklaar bekende topologië en word gebruik om nuwes te ontwerp.

Hoë verwerkings-vereistes word beperk met ‘n vinnige inkrementele afrig-algoritme (FIT). Hierdie algoritme bespeur en vermy oortollige hoër orde oorgangswaarskynlikhede deur te let op laer orde oorgangswaarskynlikhede wat nul is. Dit verminder die geheue- en verwerkingsvereistes aansienlik. Verder is die resulterende modelle meer kompak. Hulle veralgemeen ook beter wanneer hulle op nuwe data toegepas word.

Die praktiese bruikbaarheid van hierdie tegnieke word gestaaf deur dit toe te pas op outomatiese taalherkenning. Sonder om tydens afrigting transkripsies te gebruik, vergelyk die stelsel goed met huidige stelsels wat wel transkripsies nodig het.

Acknowledgements

While a study like this is more of a personal endeavour than any other I have done before, many people have played a shaping and supportive role in it.

The first part of this study was done without any study-leader. When Etienne Barnard and David Weber arrived on the scene, they contributed much needed focus and experience. Etienne, thank you for your willingness, your timeous and well-considered responses to my email queries. The directions you provided were seeds the fruit of which you will recognise in this work. David, I have led many students during my academic career, but you have redefined the term for me. The many hours that your meticulous attention, thorough evaluation and good ideas must have required have certainly not gone by unnoticed.

The South-African Defence Research Council and DataFusion Systems provided financial support over many years. Wynand Coetzer introduced me to the field of speech processing and has ever since played a very important mentoring role in my development. The Oregon Graduate Institute of Science and Technology kindly made their Multi-Language Telephone Speech database (OGI-TS) available to me.

During the last year colleagues Johan Lourens and David Weber took over some of my teaching load. This has given me the time to complete this work. Hans Grobler and Rudolph van der Merwe kept our Linux system running. The speech-processing group at this university has developed a powerful and flexible pattern recognition system, also enabling this research. I want to acknowledge the role of all the students who have contributed over the years to its development.

I am blessed with parents that are also friends. Dad, the many walks up in the mountain with you as a sounding board helped a lot to clear my mind. Then there is

my dear wife Sanet and my children Jarien and Tiana. I am not overstating it when I say that this has been a very trying time for all of us. Thank you for all the sacrifices you have made. Sanet, without your constant support and encouragement this work would not have been possible.

Finally I cannot otherwise than see the presence of a Higher Hand who blessed me with all these people and guided me along unknown but fruitful paths.

To all of you I am indebted and truly grateful.

Contents

CHAPTER 1 PURPOSE, OBJECTIVES AND CONTRIBUTION 1

1.1 MOTIVATION FOR AND TOPICALITY OF THE RESEARCH..1

1.2 STATEMENT OF THE PROBLEM 2

1.3 RESEARCH OBJECTIVES 3

1.4 HMM CONCEPTS 3

1.5 PRIOR WORK ON HIGH-ORDER HIDDEN MARKOV MODELS 6

1.6 OVERVIEW OF THIS RESEARCH...9

1.6.1 The ORder rEDucing (ORED) algorithm..10

1.6.2 The Fast Incremental Training (FIT) algorithm..12

1.6.3 Practical application to language recognition 14

1.7 CONTRIBUTION OF THIS WORK..15

CHAPTER 2 HIGH-ORDER HIDDEN MARKOV MODELS...18

2.1 INTRODUCTION 18

2.2 HMM BACKGROUND 19

2.2.1 Definition and notation 19

2.2.2 HMM assumptions...21

2.2.3 Standard HMM algorithms..23

2.2.4 Different types of HMMs 27

2.2.5 Related concepts: Markov chains and N-grams..28

2.3 REDUCING A HIGH-ORDER TO AN EQUIVALENT FIRST-ORDER HMM 32

2.3.1 The ORder rEDucing (ORED) algorithm..33

2.3.2 Notes on the ORED algorithm..36

2.3.3 Proof that the ORED algorithm results in equivalent first-order models......................37

2.4 EXAMPLES USING THE ORED ALGORITHM..41

2.4.1 A simple mixed-order model...41

2.4.2 Inconsistent transition probabilities...43

2.4.3 Non-supporting transition probabilities 44

2.5 Interesting higher-order HMM topologies 45

2.5.1 Left to right HMM3 with one state skip 46

2.5.2 Duration modelling with high-order HMMs....48

2.5.3 Context modelling with high-order HMMs....50

2.6 Relationship of prior work to the ORED approach. 53

2.7 Practical issues with higher-order HMMs....57

2.8 Summary and conclusions....58

CHAPTER 3 TRAINING HIGH-ORDER HMMS 60

3.1 Introduction 60

3.2 Incremental training of fixed-order HMMs 61

3.2.1 Basis for Fast Incremental Training 61

3.2.2 The FIT algorithm for the training of fixed-order HMMs 64

3.3 Mixed-order variants of the FIT algorithm 65

3.3.1 Duration modelling with the FIT algorithm 67

3.3.2 Context emphasised mixed-order HMMs 68

3.3.3 Combinations of context and duration models....69

3.4 Example of training via the extended/ORED and FIT algorithms....70

3.5 Simulations to examine the convergence of extended/ORED and FIT based training 73

3.5.1 Generating simulated data 75

3.5.2 Comparing FIT vs. extended/ORED trained models....76

3.6 Summary and conclusions....83

CHAPTER 4 HIGH-ORDER HMMS APPLIED TO AUTOMATIC LANGUAGE RECOGNITION....84

4.1 Introduction....84

4.2 Background on language recognition 86

4.3 Database and signal processing....91

4.4 Basic system structure and initialisation 92

4.5 Training procedure 93

4.6 COMPARISON OF FIT- VS EXTENDED/ORED-TRAINED HMMS 95

4.6.1 Topology and training method 95

4.6.2 Computational requirements 95

4.6.3 Classification accuracy 98

4.7 INDEPENDENT VARIATION OF CONTEXT AND DURATION ORDERS 99

4.7.1 Topologies used 100

4.7.2 Model sizes 101

4.7.3 Classification accuracy on training set 102

4.7.4 Classification accuracy on testing set 102

4.8 COMPARISON WITH PRIOR WORK 103

4.9 OUTSTANDING ISSUES 105

4.10 SUMMARY AND CONCLUSIONS 107

CHAPTER 5 CONCLUSION 108

5.1 CONCLUDING PERSPECTIVE 108

5.2 TOPICS FOR FURTHER STUDY 110

APPENDIX A DETAILED EXAMPLE OF THE ORED PROCEDURE 121

APPENDIX B ORED REDUCTION FOR CONTEXT MODELLING 127

APPENDIX C STATISTICAL SIGNIFICANCE TESTS 131

C.1 OVERVIEW OF MCNEMAR SIGNIFICANCE TEST 131

C.2 SIMULATION EXPERIMENTS OF SECTION 3.5.2 133

C.2.1 Configurations evaluated 133

C.2.2 Results on training data 134

C.2.3 Results on testing data set 138

C.3 AUTOMATIC LANGUAGE RECOGNITION EXPERIMENTS (SECTIONS 4.6 AND 4.7) 142

C.3.1 Configurations evaluated 142

C.3.2 Results on training data 143

C.3.3 Testing set results 147

List of Figures

Figure 1-1. Part of a first-order HMM. The pdf. f_i describes features from state *i*. 4

Figure 1-2. The second-order version of the HMM in Figure 1-1. 4

Figure 1-3. Mapping the original states of Figure 1-2 into their twofold Cartesian product implicitly adds one extra step of state history to the model. 10

Figure 2-1. Base HMM1. The pdf. f_i describes features associated with state *i*. 22

Figure 2-2. Second-order version of the HMM in Figure 2-1. 23

Figure 2-3. Discrete HMM. 29

Figure 2-4. Discrete HMM without pdf.s. 30

Figure 2-5. A mixed-order HMM (maximum order 3)..41

Figure 2-6. Equivalent of Figure 2-5 with highest order two. The intermediate composite indexing scheme is retained for easier reference to the algorithm. 42

Figure 2-7. First-order equivalent of Figure 2-5. 42

Figure 2-8. An inconsistent mixed-order model..43

Figure 2-9. First-order equivalent (still using the intermediate indexing scheme). States 1 and (1,1) should have been tied, but they have a different number of transitions. 44

Figure 2-10. A mixed-order model with dangling states. 45

Figure 2-11. First-order equivalent (still using the intermediate indexing scheme). Note the dangling state (2,2). 45

Figure 2-12. Third-order left-to-right HMM. 46

Figure 2-13. Equivalent HMM2 for the HMM3 in Figure 2-12, using the intermediate composite state indexing scheme. 46

Figure 2-14. The HMM1 equivalent to the HMM in Figure 2-12. The states of the original model in Figure 2-12 correspond here to the set of states with the same pdf. (f_i)......47

Figure 2-15. Durational left to right HMM3 with one state skip. Note the shared transitions which constrains Figure 2-12 to focus on duration modelling. 49

Figure 2-16. First-order equivalent of the HMM in Figure 2-15. Note the similarity to Ferguson model shown in Figure 2-17. 49

Figure 2-17. Ferguson model of Figure 2-1 (with 2 time lags). 50

Figure 2-18. Third contextual order left to right HMM with one state skip. The notation k^{+} is used to indicate one or more occurrences of index *k*. The (infinite) family of transition probabilities indicated by each symbol *a* in the figure are all identical. This is a mixed-order model with a maximum order of infinity. 51

Figure 2-19. First-order equivalent of the HMM in Figure 2-18. 52

Figure 2-20. a) Second-order and b) equivalent first-order HMMs (from Mari *et al.*, 1997). 55

Figure 3-1. Durational left to right HMM2 with one state skip. 68

Figure 3-2. First-order version of Figure 3-1. 68

Figure 3-3. Model used for generating simulated data. 71

Figure 3-4. Model structures after a) first, b) second and c) third-order stages of the FIT algorithm. 72

Figure 3-5. An example of a set of Gaussian pdf. centroids from HMMs used to simulate data. The circles are the one standard deviation contour on the pdf.s. 76

Figure 3-6. Excess transitions in extended/ORED and FIT trained HMMs relative to those of the true underlying model. The HMMs are identified by tuples such as (3,8), indicating a third-order HMM with eight states. 79

Figure 3-7. Relative error increase when classifying *training data* with extended/ORED, FIT and first-order trained HMMs, compared to that of the true underlying model. 80

Figure 3-8. Relative error increase when classifying *independent testing data* with extended/ORED, FIT and first-order trained HMMs, compared to that of the true underlying model. 82

Figure 4-1. A four-state ergodic HMM. (States 0 and 5 are the initial and final null states.) We based our experiments on sixteen-state ergodic HMMs. 92

Figure 4-2. Maximum memory requirements during training of the HMMs. Based on the size needed for the Viterbi path matrix for a 5s chunk of speech, as well as the memory required to store the transition probabilities (using a sparse representation). 96

Figure 4-3. Number of transition probability operations required during the training of the HMMs. (Averaged over the English and Hindi models.) 97

Figure 4-4. Number of transition probabilities in the trained HMMs. (Averaged over the English and Hindi models.) 97

Figure 4-5. Identities and FIT training dependencies (indicated by arrows) of ALR models. *C* is the context order and *D* is the duration order. 100

Figure 4-6. Number of transition probabilities in the trained HMMs. (Averaged over the English and Hindi models.) 101

Figure A-1. A mixed-order HMM (maximum order 3). 121

Figure A-2. ORED iteration 1, after completion of step 4. Dead states S=2,3 and 4, are supposed to be removed in step 3, but are kept to illustrate all states. 122

Figure A-3. ORED iteration 1, after completion of step 5a. This is the initial model for the next iteration of ORED. Only links departing from S=3 are still of order higher than one. 123

Figure A-4. ORED iteration 2, after completion of step 4. Since there are only first-order transition probabilities present in model, step 5 will be followed by termination in step 6. Transition probability values to the left of the assignment still refer to Figure A-3. Dead state S=3 is supposed to be removed in step 3, but is kept to show all states. States S=5 and S= (3,5) are tied. 125

Figure A-5. ORED iteration 2, after completion of step 5. States S=6 and S=8 are tied and lead to the same destinations. They can therefore be merged. 126

Figure A-6. Finally, the first-order equivalent of Figure A-1. 126

Figure B-1. Second contextual order left to right HMM with one state skip. The notation k^+ is used to indicate one or more occurrences of index *k*. The family of transition probabilities indicated by each symbol *A* in the figure are all identical. 127

Figure B-2. Result after first cycle of ORED algorithm on Figure B-1. 127

Figure B-3. Result after second cycle of ORED algorithm. 128

Figure B-4. Repeating (infinite) substructures of Figure B-2 and Figure B-3. 129

Figure B-5. Finite equivalent of Figure B-4. 130

Figure B-6. First-order equivalent of the HMM in Figure B-1. 130

List of Tables

TABLE 2-1. HMMS WITH DIFFERENT PDF. TYPES. 28

TABLE 3-1. MEAN DEVIATION BETWEEN TRAINED AND ACTUAL PARAMETERS. 73

TABLE 3-2. TOTAL LIKELIHOOD $\log[f(\mathbf{X}_1^L | M)]$ AFTER TRAINING. 73

TABLE 3-3. THE FOUR DIFFERENT GENERATING HMM EXPERIMENTS EVALUATED. "σ SPACING" IS THE DISTANCE BETWEEN THE GAUSSIAN CENTROIDS. 78

TABLE 4-1. COMPARISON OF COMPUTATIONAL REQUIREMENTS AND FINAL MODEL SIZES FOR 16-STATE ERGODIC HMMS TRAINED VIA EXTENDED/ORED AND FIT ALGORITHMS. 98

TABLE 4-2. ACCURACY MEASURED ON TRAINING SET FOR 16-STATE ERGODIC HMMS TRAINED VIA EXTENDED/ORED AND FIT ALGORITHMS. 98

TABLE 4-3. ACCURACY MEASURED ON TESTING SET FOR 16-STATE ERGODIC HMMS TRAINED VIA EXTENDED/ORED AND FIT ALGORITHMS. 98

TABLE 4-4. TRAINING SET CLASSIFICATION ACCURACY FOR DIFFERENT MODELS. 102

TABLE 4-5. TESTING SET CLASSIFICATION ACCURACY FOR DIFFERENT MODELS. 103

TABLE C-1. THE NUMBER OF OCCURRENCES OF JOINT CLASSIFICATION OUTCOMES FOR THE TWO ALGORITHMS. *N* IS THE RANDOM VARIABLE WHILE *n* IS ITS OUTCOME. 131

TABLE C-2. THE FOUR DIFFERENT GENERATING HMM EXPERIMENTS EVALUATED. "σ SPACING" IS THE DISTANCE BETWEEN THE GAUSSIAN CENTROIDS. 133

TABLE C-3. MCNEMAR COUNTS FOR DATA CONFIGURATION 1 (SEE TABLE C-2). THE MODELS EVALUATED ARE GIVEN IN THE TOP ROW AND RIGHT-MOST COLUMN. THE FOUR VALUES AT THE INTERSECTION OF A GIVEN PAIR OF MODELS REPRESENT THE JOINT CLASSIFICATION COUNTS, AS IS ALSO ILLUSTRATED AND EXPLAINED IN TABLE C-1. 134

TABLE C-4. THE PROBABILITY THAT OBSERVED DIFFERENCES BETWEEN ANY TWO MODELS AROSE BY CHANCE. CALCULATED FROM THE MCNEMAR COUNTS OF TABLE C-3. MODELS LIKELY TO DIFFER IN ACCURACY ARE SHADED. 134

TABLE C-5. MCNEMAR COUNTS FOR DATA CONFIGURATION 2 (SEE TABLE C-2). THE MODELS EVALUATED ARE GIVEN IN THE TOP ROW AND RIGHT-MOST COLUMN. THE FOUR VALUES AT THE INTERSECTION OF A GIVEN PAIR OF MODELS REPRESENT THE JOINT CLASSIFICATION COUNTS, AS IS ALSO ILLUSTRATED AND EXPLAINED IN TABLE C-1. 135

Table C-6. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-5. Models likely to differ in accuracy are shaded. 135

Table C-7. McNemar counts for data configuration 3 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1. 136

Table C-8. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-7. Models likely to differ in accuracy are shaded. 136

Table C-9. McNemar counts for data configuration 4 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1. 137

Table C-10. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-9. Models likely to differ in accuracy are shaded. 137

Table C-11. McNemar counts for data configuration 1 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1. 138

Table C-12. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-11. Models likely to differ in accuracy are shaded. 138

Table C-13. McNemar counts for data configuration 2 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1. 139

Table C-14. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-13. Models likely to differ in accuracy are shaded. 139

Table C-15. McNemar counts for data configuration 3 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1. 140

Table C-16. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-15. Models likely to differ in accuracy are shaded. 140

Table C-17. McNemar counts for data configuration 4 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1. 141

Table C-18. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-17. Models likely to differ in accuracy are shaded. 141

Table C-19. Identities and descriptions of models used in automatic language recognition experiments. All models indicated as extending another model are trained via the FIT algorithm, models 1, X2 and X3 are trained directly using the Extended/ORED approach.......... 142

Table C-20. The McNemar classification counts on 5s language recognition trials using 16 state HMMs (see Table C-19). The four values at the intersection of a given pair of models represent the joint classification counts, as in Table C-1. 143

Table C-21. The probability that observed differences between any two models arose by chance. Calculated according to the McNemar test as applied to the counts of Table C-20. Models likely to differ in accuracy are shaded. 144

Table C-22. The McNemar classification counts on 45s language recognition trials using 16 state HMMs (see Table C-19). The four values at the intersection of a given pair of models represent the joint classification counts, as in Table C-1. 145

Table C-23. The probability that observed differences between any two models arose by chance. Calculated according to the McNemar test as applied to the counts of Table C-22. Models likely to differ in accuracy are shaded. 146

Table C-24. The McNemar classification counts on 5s language recognition trials using 16 state HMMs (see Table C-19). The four values at the intersection of a given pair of models represent the joint classification counts, as in Table C-1. 147

Table C-25. The probability that observed differences between any two models arose by chance. Calculated according to the McNemar test as applied to the counts of Table C-24. Models likely to differ in accuracy are shaded. 148

Table C-26. The McNemar classification counts on the NIST'95 45s language recognition trials using 16 state HMMs (see Table C-19). The four values at the intersection of a given pair of models represent the joint classification counts, as in Table C-1........ 149

Table C-27. The probability that observed differences between any two models arose by chance. Calculated according to the McNemar test as applied to the counts of Table C-26. Model F3 differ significantly (97% confidence) from model 1................ 150

Chapter 1 Purpose, objectives and contribution

1.1 Motivation for and topicality of the research

Before we begin a detailed discussion on the subject, we note some perspectives of a few well-known authors on higher-order hidden Markov Models (HMMs)[1]:

> In his book on Speech Communication, O'Shaughnessy (1990, p. 461) comments on this by noting that "*Higher-order Markov processes could exploit restrictions on which sounds may occur in sequence within words, but the computational complexity of such models has thus far precluded their application to acoustical analysis in ASR*".
>
> In their handbook, Deller, Proakis and Hansen (1993, p. 681) supply only a single remark in a footnote: "... *in theory, nothing precludes dependence upon N past states, but the complexity of training this model and recognition using it increases dramatically with each increment ....*".
>
> Juang and Rabiner (1992, p. 544) devotes a full paragraph to the topic: "*Until now almost all HMM formulations for speech recognition are based on a simple first-order Markov chain. ... When an HMM is used in higher levels of a recognition system, such as syntactic or semantic processing, the first-order formulation turns out to be inadequate. ...Although the structural simplicity of a first-order model makes the computation simple and straightforward, there may be a need to complete the analytical framework of higher-order models.*

[1] These quotations represent the only discussion of high-order HMMs in these references.

> *Also, for such higher-order models to be practically useful, many of the implementational advantages of the first-order case may have to be formulated in an appropriate manner.*"

These quotations highlight advantages that may be obtained from the longer past state sequence that a high-order HMM takes into account when making a transition. There are many situations where a pattern recognition system can benefit from incorporating longer-term dependencies in the data that it is modelling. Viewing our world through first-order HMM spectacles, is equivalent to saying that what is going to happen next, is dependent only on the here-and now. When humans communicate, this is obviously not true. The physical constraints on speech production, as well as the necessity to make our language robust, gave rise to longer-term structure. Our expectation of future sounds is rooted in a far larger context than just current events.

These quotations next focus on the vastly increased computational requirements of higher-order HMMs as the reason for their almost total absence from literature. The need expressed in the last part of the quotation from Juang and Rabiner (1992) is the primary inspiration for this research. In this work we expand the analytical framework of high-order HMMs and develop techniques to address some of the problems indicated by Juang and Rabiner. We are able to show practical application of higher-order HMMs to real-world speech problems.

1.2 Statement of the problem

Very little literature is available on high-order HMMs, and what is available, is specifically concerned with extensions to second-order (He, 1988; Kriouile, Mari and Haton, 1990). The current piecemeal approach to extending HMM algorithms to higher order has some shortcomings. Current approaches implement specialised algorithms for each HMM order, which must certainly have impeded its widespread use. It also fetters a wider perspective on HMMs irrespective of order; fundamental questions, such as whether HMMs of different order have equivalent representational capacity, or should be considered as different subdivisions on the larger tapestry of the Chomsky hierarchy of languages, are not immediately evident. During the late seventies, effective implementation catapulted first-order HMMs to the front line of speech processing. On higher-order HMMs, the art still needs to progress beyond just acknowledging that they are very expensive. Little is known about their training behaviour and possibilities for efficient implementation.

1.3 Research objectives

This work aims to:

- find a *unifying* perspective linking HMMs of all orders,
- find a technique facilitating *efficient* training of high-order HMMs while at the same time maintaining or enhancing its quality,
- demonstrate the *practical* applicability of these techniques using a non-trivial real-life problem.

1.4 HMM concepts

In subsequent sections of this chapter we discuss the literature on high-order HMMs and also provide an overview of our own research. The discussion here will concentrate on the necessary key concepts, leaving the details and mathematical treatment to Section 2.2.

We use HMMs (for a good introduction see Rabiner & Juang, 1985) to model a sequence of observations and their relationship to each other. For our purpose these are descriptions of short consecutive speech segments that together describe a whole utterance. An HMM models utterances like these with a finite number of states. Each state has a probability density function (pdf.) describing the nature of the speech that it associates with. These pdf.s are indicated in Figure 1-1 by f_i where i is the state number. Having described the individual sounds, the way in which they may combine is described by arrows that join states. The probability of such a combination is indicated by the a's in Figure 1-1 and Figure 1-2.

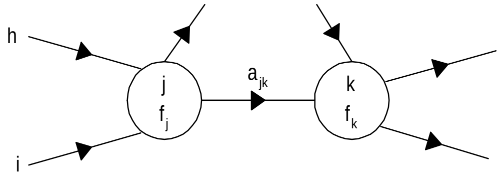

Figure 1-1. Part of a first-order HMM. The pdf. f_i describes features from state *i*.

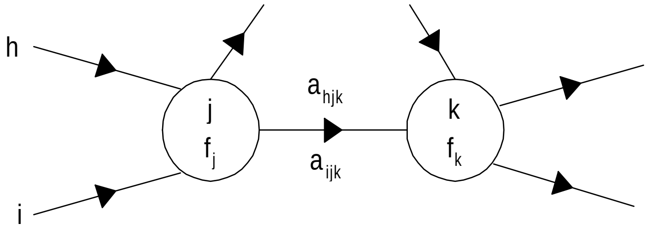

Figure 1-2. The second-order version of the HMM in Figure 1-1.

In Figure 1-1, the transition to state k depends only on the current state j, thereby making a_{jk} a first-order transition probability. In Figure 1-2, however, the terms a_{hjk} and a_{ijk} describe the probability of making a transition to state k given that the current state is j and the previous one was respectively i or h. Because two prior states are taken into account when making the transition to the destination state k, these are second-order transition probabilities. The order of the model therefore specifies the context (history) that is being taken into account when deciding on the appropriate combination of states. Using the first-order transition probability (a_{jk}) in Figure 1-1, we will not be aware of whether the prior state was *h* or *i* when making a transition from *j* to *k*. The second-order transition probabilities explicitly account for this extra context. As will be seen later, computational cost increases exponentially with the order of the HMM, making high-order modelling very expensive indeed. There are three modes of applying (first-order) HMMs (Rabiner & Juang, 1985):

- When we use HMMs to classify observation sequences, we assume that the observation sequence was generated by one of them. To decide which one, we need to evaluate the likelihood of each of them producing this observation sequence. The HMM formulation, however, "hides"[2] the exact state sequence that generated the data by means of the pdf.s f_i. Taking the contribution of all possible state sequences into account is intractable when done in a naive way. The efficient way in which the Baum-Welch algorithm, and its (close) approximation, the Viterbi algorithm does this, catapulted first-order HMMs to the principle tool used in automatic speech recognition.
- States and groups of states are often associated with concepts like phonemes and words. Since the recovery of those concepts from a larger observation sequence

[2] Hence the name *hidden* Markov model. It is this characteristic that distinguishes HMMs from Markov chains and N-gram models. Section 2.2.4 expands on the similarities of, and differences between, these topics.

(speech) is of prime importance in some applications, we need to be able to determine the state sequence most likely to have generated an observation sequence. Since the Baum-Welch algorithm simultaneously considers all state sequences, it is not useful for this. The Viterbi algorithm, however, specifically only considers the most likely state sequence, making it an obvious choice for doing segmentation (or decoding) of a larger observation sequence.

- Lastly, we should be able to optimise or train the parameters of an HMM to optimally reflect the observation sequences it represents. Embedding either the Baum-Welch or the Viterbi algorithms in an expectation-maximisation (EM) algorithm (Dempster, Laird & Rubin, 1977) results in the so-called re-estimation equations commonly used to train HMMs with. These algorithms are guaranteed to yield a locally optimal solution, but are not necessarily globally optimal (Levinson, 1985).

1.5 Prior work on high-order hidden Markov models

Due to the difficulties associated with higher-order HMMs, literature[3] is (surprisingly) sparse. In this section we discuss these without much reference to our research, thereby sketching the backdrop against which this study has taken place. We will, however, in Section 2.6 return to this topic to provide more perspective on the relationship of our research to the existing literature.

[3] In this section we will not concern ourselves with Markov chains or N-gram models, but will only focus on the available literature on higher order HMMs (see Section 2.2.4 for a discussion of these concepts).

The earliest reference found is by He (1988). He uses a twofold product of the original state space to derive a second-order Viterbi algorithm with. He also indicates that extending the Viterbi algorithm to even higher orders, follows along the same lines as the extension to second-order, but does not present such results. In a related publication Kundu, He and Bahl (1988) report improved recognition of hand-written words, using this algorithm to restrict the available letter combinations.

Kriouile, Mari and Haton (1990) extend the available techniques by deriving an extended Baum-Welch re-estimation algorithm specific to second-order discrete HMMs. They compare the use of first-order versus second-order HMMs in an isolated digit recognition experiment. The popular “left to right, one state skip” topology was used for each digit model. Doing the tests in multi-speaker mode, they were able to decrease the recognition error by 75% by using second-order HMMs (From 96% to 99% accurate). In speaker-independent mode they also obtained slightly better results (91% to 93% accurate). They also demonstrate a third model termed as a “transition equivalent model”. This model, which is not as accurate as their second-order models, has a transition structure similar to that obtained using our ORder rEDucing (ORED) algorithm (to be detailed in Section 2.3).

In a series of papers, second-order HMMs were next applied to continuous speech recognition (Mari & Haton, 1994; Mari, Fohr & Junqua; 1996; Mari, Haton & Kriouile, 1997). They focus on the enhanced duration modelling capabilities of these models as primary factor in their enhanced performance, pointing out similarities[4] with Ferguson models (Ferguson, 1980). The self-loops on first-order HMM states are poor duration models, allowing the occurrence of singular state alignments when observations are matched to the wrong model. The extra time step available in the state history of second-order HMMs avoids this difficulty for one additional time step, thereby preventing such singular state alignments. Using second-order HMMs to recognise letters in a continuously spelled name task, Mari *et al.* (1996) slightly improve the accuracy compared with the use of first-order HMMs. Also in a continuous phoneme recognition task, they show a slight improvement. From our own experience (Du Preez, 1991a) we can confirm the advantages of employing better duration modelling in continuous recognition tasks. We do, however, believe that explicit duration modelling (Levinson, 1986) is a viable alternative to bringing the full power of higher-order HMMs to bear on this problem[5].

This viewpoint is confirmed by Mari *et al.* (1994, 1997) in an experiment on connected digit recognition. Although in a direct comparison, the second-order HMMs improve on the results from first-order HMMs, this is no longer the case when the first-order model employs durational postprocessing. These papers are the first that we are aware of that mention the concept of a first-order HMM that is equivalent to a second-order HMM. This order reduction is based on mapping the original states into a twofold Cartesian product of states (compare Figure 1-2 with Figure 1-3 for an

[4] High-order HMMs are much more powerful than Ferguson models. They can however be constrained to produce Ferguson models. Section 2.5 deals with this and other useful topologies.

[5] We do however find the relationship between high-order HMMs and duration modelling, intriguing and devote Section 2.5.2 to its discussion.

example). This is a very intuitive notion, also noticed by Howard (1972, p. 4) in the context of Markov chains. No formal algorithm[6] is, however, given in any of these references. After illustrating such a first-order equivalent for a three-state left-to-right second-order HMM, Mari *et al.* favour extending the Viterbi and Baum-Welch algorithms to second-order over using first-order equivalents. As reason for this they motivate that the latter dramatically increases the number of states.

To summarise, previous work (He, 1988; Kriouile *et al.*, 1990; Mari *et al.*, 1994, 1996, 1997) focused on *extending algorithms* for processing *second-order* HMMs. Although the authors indicate that extension to algorithms for HMMs of order higher than two will follow a similar pattern as the extension to second-order, no such algorithm is presented. Although they are aware of the possibility of reducing second-order HMMs to equivalent first-order models, on the grounds of the resulting increase in the number of states, they prefer rather to use algorithms specifically extended to second-order.

1.6 Overview of this research

This section provides a high-level summary of the work done in this research. The discussion will focus on the main concepts and issues, details and motivations are reserved for discussion in later chapters. Chapters 2, 3 and 4 closely follow the research objectives set out in Section 1.3 namely a unifying perspective, efficient implementation and practical applicability. The following three subsections outline these chapters.

[6] As we shall see in Section 2.3.1, such an algorithm involves quite a few intricacies.

1.6.1 The ORder rEDucing (ORED) algorithm

In Figure 1-2 the second-order transition probability a_{ijk} describes the probability of making a transition to state *k* given that the current state is *j* and the previous one was *i*. First-order transition probabilities like a_{jk} involve only the two states that are joined by them. In contrast, the second-order dependence of a_{ijk} on state *i*, cannot be inferred from its adjoining states but is only encoded in the subscripts of the transition probability itself. Let us now create a new model with states corresponding to pairs of linked states from the original model, as is illustrated in Figure 1-3. Each state shares the same pdf. as the second one of the original pair of states does. Transition probabilities are inserted between the states that respectively match the first and the last two subscripts of the transition probability e.g. a_{ijk} is inserted between states (*i*,*j*) and (*j*,*k*).

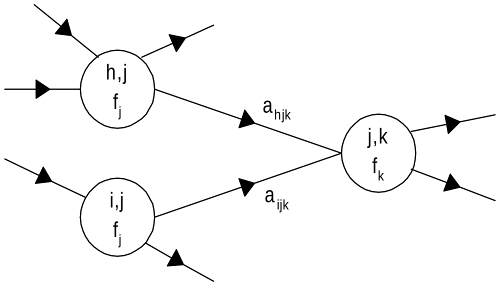

Figure 1-3. Mapping the original states of Figure 1-2 into their twofold Cartesian product implicitly adds one extra step of state history to the model.

Now the indexes of the states adjoining the second-order transition probability fully describe the subscripts of this transition probability. This effectively means that we can now interpret a_{ijk} as a first-order transition probability joining states (*i*,*j*) and (*j*,*k*). By enlarging the number of states in the way we did, we were able to reduce effectively the order of the model by one, without losing any representational capability. This simple observation forms the basis of the ORder rEDucing (ORED) algorithm, which is detailed in Section 2.3.1. If we are able to reduce a second-order model to a first-order model, it should also be possible to do so for models of higher order. Section 2.3 confirms that this is indeed so. An illustration of this process is given by the third-order HMM of Figure 2-12 (p. 46) which is reduced to an equivalent first-order version given by Figure 2-14 (p. 47). However, the proper handling of higher orders, initial conditions (the first transition in the model by necessity is of first order) and the desire to process models of mixed-order, introduce quite a few intricacies to the algorithm that are not evident from the intuitive notion on which it is based. These complexities are detailed in Chapter 2.

Although related to the above literature, this approach differs in fundamental respects. Instead of *increasing the order of specific algorithms* to specific orders (only second-order in the literature), this algorithm *reduces all higher-order HMMs* to equivalent first-order versions. This has a number of important implications. These are now outlined:

On a practical level this allows the *application of any standard HMM algorithm to any higher-order* HMM, greatly enhancing the usefulness of this technology. Although concerns were expressed about the effect the increase in the number of states might have (Mari *et. al*, 1994, 1997), we show in Section 2.6 that it does not contribute in any way to additional computational requirements.

On a theoretical level, it provides a unifying paradigm for reasoning about HMMs of any order because it makes the relationship between HMM topology and HMM order explicit. Using this insight, *HMMs can be designed using higher-order specifications* and then reduced to make its topology explicit using a first-order equivalent model.

For example, in Section 2.5.2 we present a mixed order model that constrains higher-order transition probabilities to specifically model *state repetitions (duration)*, irrespective of past history, while restricting the transitions between distinct states to first-order (see Figure 2-15 p. 49). This results in a Ferguson model (Ferguson, 1980) as its first-order equivalent (Figure 2-16 p. 49). This topology plays an important role in later work. In Section 2.5.3 we develop the natural counterpart of the above duration model. We design a model that is guaranteed always to take a fixed context of distinct prior states into account irrespective of their possible repetitions (see Figure 2-18 p. 51 for a simple example, with its first-order equivalent shown in Figure 2-19 p. 52). This mixed order model has enhanced capabilities for modelling sequences of distinct states and also plays a key role in later work.

1.6.2 The Fast Incremental Training (FIT) algorithm

As already mentioned, high-order HMMs can be vastly more expensive than their first-order counterparts. The processing and memory requirements are serious issues that can easily place such a model outside the available computing capacity. Typically though, the transition structure of a high-order HMM is quite sparse. Because it may be difficult to tell, prior to training, which transitions will be redundant, training normally commences with all the transitions that are potentially useful. For many problems considerable training effort is therefore expended on estimating parameters that will eventually become zero. Referring back to Figure 1-1 and Figure 1-2, it will be realised that a single transition probability in the lower-order model, is simply being replaced by a set of refined probabilities in the next higher-order model. It will result in significant savings if the training of redundant sets of higher-order probabilities can be avoided by noting which corresponding lower-order probabilities are zero. This observation forms the basis of the Fast Incremental Training (FIT) algorithm, detailed in Section 3.2.2 and summarised below:

1) Set up a first-order HMM for the application at hand.

2) Run the training algorithm on the first-order model. Non-viable transitions will disappear.

3) Convert the optimised first-order model to a second-order model by expanding the subscripts of the remaining non-zero transition probabilities with one extra prior state. These expanded transition probabilities are initialised with the value of the lower-order transition probability they were extended from.

4) Use the ORED algorithm to create a first-order equivalent of this model.

5) Now, by repeating the algorithm from step 2, train this model. This will refine the transition probabilities to their required higher-order values. Repeating this process trains successively higher-order models.

The above formulation is geared towards training fixed-order HMMs. In Section 3.3 we also extend it to include certain useful mixed-order topologies. Specifically, we can also efficiently train the duration models of Section 2.5.2, the context models of Section 2.5.3, or any combination of them, by using the FIT algorithm.

Because the FIT algorithm does not utilise the same initial conditions[7] as models trained via extended algorithms or the ORED reduction[8], local optima may cause the trained models to vary. In Section 3.5 we use well-controlled simulation experiments

[7] They cannot since they do not even have the same number of initial parameters.
[8] We will refer to these (equivalent) approaches as the extended/ORED approach.

to investigate the quality of FIT models relative to that of the extended/ORED approach. Subsequent work on real speech (see Section 4.6) confirms the results in a real-life situation. We find that, not only is the FIT approach much *more efficient* (up to an order of magnitude faster), but it also results in much *more compact* models that *generalise better* to unseen data.

1.6.3 Practical application to language recognition

To show that the concepts developed in Chapters 2 and 3 are indeed applicable to non-trivial real-life situations, we apply them to automatic language recognition in Chapter 4. Our ALR system distinguishes itself from most others (see Section 4.2 for ALR literature) in that it requires no transcription of the training database, making it much easier to expand to new languages. Two sets of experiments were conducted. Common aspects like the database, signal processing and training procedures, are discussed in Sections 4.3 to 4.5.

The first set of experiments (Section 4.6) was concerned with verifying the results of the simulation experiments of Chapter 3. To do this, sixteen-state ergodic HMMs of second- and third-orders were trained on English and Hindi speech[9] using both the extended/ORED and the FIT approaches. A first-order version served as baseline reference. The results confirm those of the simulations of Chapter 3. For example, training a third-order FIT model requires 13% of the memory, and 7% of the CPU that would be expended on extended/ORED training. The resultant FIT model is twenty

[9] From the OGI-TS database, kindly supplied to us by the Oregon Graduate Institute.

times smaller and yields an accuracy of 97.4% compared to the 89.7% of its extended/ORED counterpart.

In a second set of experiments (Section 4.7) we applied some of the explicit duration and context modelling mixed-order HMMs, as well as combinations of them, to the ALR task. All training is performed via the FIT algorithm. All the topologies have been proved to be practically viable. Duration modelling proved useful throughout. Modelling a context of distinct states is promising, but appears to require larger training databases. This can be explained by noticing that this model will reflect longer state histories than a normal fixed-order model will, thereby increasing the size of the required database. Due to the limited test set, statistically significant differences were only detected between the baseline first-order model and several of the higher-order models.

When we compare our ALR approach to others (see Section 4.8) we find that it is better than previous systems not requiring transcriptions (Lund, Ma & Gish, 1996), and very competitive when compared to currently popular systems based on phoneme recognition followed by N-gram modelling (Zissman, 1995a; Kadambe & Hieronymus, 1995). This is achieved in spite of the fact that many obvious enhancements can be applied to our ALR system (see Section 4.9).

1.7 Contribution of this work

1) Although the intuitive notion of reducing high-order HMMs to first-order equivalents is not new, it has never been formalised before. **The ORED algorithm:**

   a) provides a precise and verified description of the order reduction process which is rather more intricate than what the underlying intuitive base would suggest;
   b) includes mixed order models for which the notion of a first-order equivalence is new;
   c) allows any higher-order HMM to be processed by using first-order algorithms;
   d) provides a unifying viewpoint creating a common framework in terms of which HMMs of all orders can be considered;
   e) makes the interplay between Markov order and HMM topology explicit;
   f) makes it possible to design HMM functionality using higher-order descriptions, afterwards reducing it to the appropriate first-order topology. We illustrate this with duration and enhanced contextual models.

2) Based on the ORED algorithm we develop **the FIT algorithm** which:
   a) greatly reduces the computational requirements for training high-order HMMs by training incrementally while avoiding redundant parameters, thereby making the training of large systems practical.
   b) results in much smaller[10] models, thereby reducing the computational resources required for applying it;

[10] Fewer non-zero transition probabilities, making it more compact.

c) makes the resultant models less susceptible to specialising on the training data; prior to this work specialisation in higher-order HMMs received no attention in the literature;

d) accommodates certain well-defined mixed order topologies;

e) provides a flexible framework for applying combinations of various enhancements such as independently controlling the context of distinct states and the degree of duration modelling used.

3) In the field of **automatic language recognition** we show that:

a) high-order ergodic HMMs do produce competitive ALR systems that require no training database transcriptions;

b) high-order HMMs can be used to design topologies that are well suited to the phonotactic constraints utilised in ALR systems;

c) these topologies can be efficiently training via the FIT algorithm.

Chapter 2 High-order hidden Markov models

2.1 Introduction

Hidden Markov theory forms the backbone of modern speech processing systems (Young, 1996) and has been thoroughly investigated for a very wide range of applications. In spite of this, very little literature exists on the use of the more powerful second- and higher-order HMMs. A few extensions of the standard first-order analysis and training algorithms to their second-order versions are known (He, 1988; Kriouile, Mari & Haton, 1990). Literature on third or higher-order models is non-existent.

The main purpose of this chapter is to introduce and motivate the ORder rEDucing (ORED) algorithm. This algorithm can transform any higher (fixed or mixed) order HMM to an equivalent first-order version. As such it plays a unifying role for HMMs of all orders and will play an important role in making this technology accessible to general speech processing and other applications.

The necessary HMM background is created in Section 2.2. Notation, assumptions, standard algorithms and their complexity are discussed. Section 2.3 introduces and proves the ORED algorithm. Its use is illustrated in Section 2.4 with several examples designed to enhance understanding the process. Section 2.5 shows how the application of the ORED algorithm can put interesting topologies in a new perspective. It also introduces two new concepts namely the *context order* and the *duration order*, which serves to illuminate the capabilities of high-order HMMs. In Section 2.6 the existing literature on higher-order HMMs is examined from the viewpoint of the ORED algorithm. Some very real problems in high order hidden Markov modelling are highlighted in Section 2.7.

2.2 HMM background

We now introduce the notations, conventions and mathematical preliminaries required for presenting, proving and demonstrating subsequent algorithms.

2.2.1 Definition and notation

$\mathbf{X}_1^L = \{\mathbf{x}_1, \mathbf{x}_2, \cdots \mathbf{x}_\ell, \cdots \mathbf{x}_L\}$ is an observation sequence or string of length L, which must be matched to the HMM. An HMM M is defined as a set of N conventional emitting states, as well as an initial and terminal state that are so-called non-emitting or null states, yielding a total of $N+2$ states. The symbols S and Q are variables taking on the index value of the state under consideration. In normal HMMs this index value will be a scalar, such as $Q=i$ (single indexed). In the transformed models developed in this chapter, composite values such as $Q=(i,j)$ (composite indexed) will be used for intermediate indexing (see Section 2.3.1 p. 33). This composite indexing is used to track state histories. Subscripts are used to specify the time at which a certain state occurs. An expression such as $S_\ell = j$ will indicate the occurrence of the j^{th} state at time ℓ. A sequence of states occurring from time m up to time n will be denoted by S_m^n. Time indexes lower than 1 or higher than L are not associated with physical time as is measured by the indexes of the observation string, but rather indicate null states preceding or following the input string.

States are coupled by transitions, each with probabilities describing its likelihood of occurring. The initial state will have no transitions entering it and the terminal state will have no transitions leaving it. Transition probabilities are indicated by the symbol a, with subscripts to index the states involved. To distinguish between the transition probabilities of the original higher-order models and its transformed equivalents, a prime (′) symbol is used for the former. For example, a'_{ijk} is the symbol used in the original second-order HMM to indicate the probability of moving from state j to state k, given that the preceding state was i.

Different states may share a set of transition probabilities (Bahl, Jelinek & Mercer, 1983). This will be referred to as *tied transitions,* and simply means the transition probabilities leaving from each of these states will be identical to that of every other state that it is tied to.

Each emitting state[11] S has an associated probability density function (pdf.) $\mathrm{f}(\mathbf{x}|S,M)$ which quantifies the similarity between a feature vector $\mathbf{x}$ and the state S. For brevity, the i^{th} pdf. will be identified as f_i in the diagrams. The null states, indicated in state diagrams by dashed circles, do not have pdf.s associated with them. Transitions to them do not consume a time-step; this makes them useful tying points for groups of states, enabling the use of single initial and termination states whilst maintaining the functionality of the multiple case. This simplifies the representation by making the customary inclusion of extra parameters to indicate multiple initial and termination states unnecessary. It should also be pointed out that, since no time step is needed to enter the first state (state 0 in the diagrams), the process initially (already)

[11] Another variant of the HMM exists which associates the pdf.s with the transitions instead of the states. This is termed the Mealy form whereas the version we describe and use throughout this work is the Moore form (Deller *et. al*, 1993, p. 680). It is possible to transform from one form to the other.

occupies that state. This is contrary to the more conventional case where it is entered only at the first time step.

In the following discussion, extensive use will be made of states that share pdf.s (Bahl *et al.*, 1983; Lee, 1989). This will be referred to as *tied pdf.s*. When states share both their transitions and their pdf.s, we will refer to them as *tied states*.

Some reflection will indicate that when states are tied, and the tied transitions leaving from each of them also share a common destination state, those tied states can be replaced by a single state without affecting the operation of the model. We will term this operation a *merging* of states. (The example in Appendix A demonstrates this.)

2.2.2 HMM assumptions

The match between $\mathbf{X}_1^L$ and M is quantified by the likelihood $\mathrm{f}(\mathbf{X}_1^L|M)$. To make the calculation of such a quantity tractable, certain simplifying assumptions are necessary:

AS.1) The observation independence assumption states that

$$\mathrm{f}(\mathbf{x}_\ell|\mathbf{X}_1^{\ell-1}, Q_0^\ell, M) = \mathrm{f}(\mathbf{x}_\ell|Q_\ell, M)\,. \tag{2-1}$$

This means that the likelihood of the ℓth feature vector is dependent only on the current state and is therefore otherwise unaffected by previous states and feature vectors. This assumption is not affected by the order of the HMM.

AS.2) By definition, an HMM includes the Markov order assumption

$$P(Q_\ell|Q_0^{\ell-1}, \mathbf{X}_1^{\ell-1}, M) = P(Q_\ell|Q_{\ell-R}^{\ell-1}, M)\,, \tag{2-2}$$

where R is known as the order of the HMM.

This assumption states that any states or features, other than the identity of the immediately preceding R states, do not affect the probability of occurrence of the next state.

The vast majority of applications use $R=1$, resulting in a first-order HMM (HMM1) of which an example is shown in Figure 2-1. The probability of jumping to a specific state at the next time step is dependent only on the state that is being occupied at the current time. Therefore, only a single transition probability occurs on the link between any two states. These probabilities are often presented as a matrix, and algorithms used to calculate the required likelihood need to keep track of only the behaviour of states one time step earlier (Poritz, 1988).

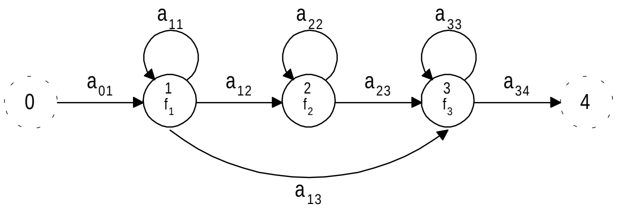

Figure 2-1. Base HMM1. The pdf. f_i describes features associated with state *i*.

Since a transition to a next state is dependent only on the current state, more subtle restrictions on the allowable combinations of states are not effectively modelled (He, 1988). In addition, the self-loops on the emitting states are poor duration models in most practical applications (Levinson, 1986). The richer modelling capability resulting from increasing the order of the model, can mitigate these difficulties (He, 1988; Mari e*t al.* 1997). In Figure 2-2 the second-order version (HMM2) of the previous example is given. More precise modelling is now possible, but it comes at the cost of an increased number of transition probabilities. Note that, unlike the HMM1, both the previous and the current state determine the correct choice amongst the number of alternative probabilities on a given transition. Typically these probabilities are represented in a cubic structure, and the calculation of $\mathrm{f}(\mathbf{X}_1^L \mid M)$ requires specialised algorithms to track the longer history of states (Kriouile *et al.*, 1990).

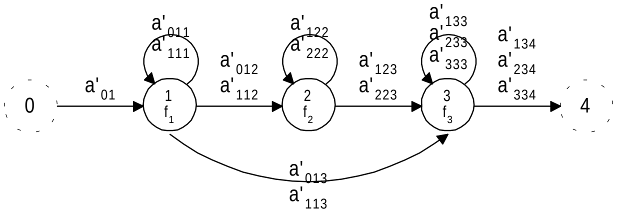

Figure 2-2. Second-order version of the HMM in Figure 2-1.

By extending the value of R to even higher values, a longer history of states can be taken into account, resulting in a R^{th}-order model (abbreviated HMMR). If the order of the probabilities on any transition is allowed to vary, a mixed-order HMM results. In following sections, a (possibly) mixed-order HMM with highest order R is referred to as M_R, where R is the highest order transition probability present in the model, be it of mixed- or fixed-order.

2.2.3 Standard HMM algorithms

There are several excellent sources describing the mathematics and algorithms applicable to HMM1s (Rabiner & Juang, 1986; Levinson, 1985; Bahl *et al.*, 1983; Poritz, 1988; Picone, 1990; Deller *et al.*, 1993). Relevant aspects of this work are highlighted here.

Three principle issues are distinguished in HMM processing, namely the evaluation problem, the decoding problem and the training problem. The evaluation problem concerns itself with how to determine the match between the observations $\mathbf{X}_1^L$ and the model M using the likelihood $\mathrm{f}(\mathbf{X}_1^L | M)$. Determining the single most likely state sequence accounting for $\mathbf{X}_1^L$ is the domain of the decoding problem. Lastly the training problem is concerned with optimising the parameters of the model based on the observations.

HMM processing is made non-trivial due to the "hidden" part indicated in its name. Many different state sequences can account for a set of observations, but with varying degrees. One approach for solving the first two problems would be to enumerate all possible sequences of states Q_0^{L+1}. Determining for each sequence the value of $\mathrm{f}(\mathbf{X}_1^L, Q_0^{L+1} \mid M)$ and then summing over all of them will solve the evaluation problem. The single sequence that provides the biggest contribution will solve the decoding problem. Unfortunately the number of possible sequences is proportional to N^L, making this approach intractable. Using the assumptions discussed in the previous section, this problem can be simplified as we now discuss.

2.2.3.1 First-order solutions

Using the observation independence assumption (AS.1) and setting the Markov order $R = 1$ allow the introduction of an (intermediate) forward variable $\alpha_\ell(j) = \mathrm{f}(\mathbf{X}_{1,}^\ell Q_\ell = j \mid M)$, as well as a related backward variable $\beta_\ell(j)$. The advantage of doing this is that these variables can be efficiently calculated from their previous values (Rabiner & Juang, 1986)[12].

[12] Only the general (iterative) expressions for the emitting states are considered here. Boundary conditions, described by additional supplementary expressions are omitted, as they do not contribute to the current discussion.

$$\alpha_{\ell+1}(j) = [\sum_{i=0}^{N+1} \alpha_{\ell}(i) a_{ij}] \mathrm{f}(\mathbf{x}_{\ell+1} \mid Q_{\ell+1} = j, M) , \quad \ell \geq 0 \tag{2-3}$$

$$\beta_{\ell}(i) = \sum_{j=0}^{N+1} \beta_{\ell+1}(j) a_{ij} \, \mathrm{f}(\mathbf{x}_{\ell+1} \mid Q_{\ell+1} = j, M) , \quad \ell \geq 0 .$$

This drastically reduces the computation required as compared to the approach of enumerating all state sequences. Consider, for example, the operations involved in calculating all the forward variables for a fully connected (ergodic) HMM with a separate pdf. on each of the emitting states. The total number of operations involving transition probabilities is proportional to N^2L while the number of pdf. calculations, as well as the memory required to store the forward variables, is proportional to NL. Either the memory requirements, or the pdf. calculations, can be avoided for the backward variables; the number of operations involving transition probabilities is similar to that of the forward variables.

From these forward and backward variables the evaluation problem is solved using the forward-backward algorithm, details of which may be obtained from Rabiner and Juang, (1986). Replacing the summation of (2-3) with maximisation results in considering only the most likely state sequence. This is formalised in the Viterbi algorithm, which solves the decoding problem. Training can be accomplished by iteratively using either the forward-backward or the Viterbi algorithms to find the current match between the observations and the model. The parameters of the model are then updated accordingly. These training procedures, termed Baum-Welch re-estimation and Viterbi re-estimation[13], are applications of the more general expectation-maximisation (EM) algorithms (Dempster, Laird & Rubin, 1977) used for optimising statistical models. These techniques guarantee an improvement on each iteration. Unfortunately, the solution it converges to may only be a local optimum. A good tutorial on the EM algorithm is provided by Moon (1996).

2.2.3.2 Higher-order solutions

Algorithms extending the Baum-Welch and Viterbi approaches to HMM2s, have been reported in literature (He, 1988; Kriouile, Mari & Haton, 1990) As in the first-order case, the forward and backward equations play a central role. The general form of the second-order forward equation is given by (the backward equations follow a similar pattern):

$$\alpha_{\ell+1}(j,k) = [\sum_{i=0}^{N+1} \alpha_{\ell}(i,j) a_{ijk}] \mathrm{f}(\mathbf{x}_{\ell+1} \mid Q_{\ell+1} = k, M), \quad \ell \geq 1 \tag{2-4}$$

Although no reference could be found to HMMs with orders higher than two, such extensions are straightforward:

$$\alpha_{\ell+1}(i_2,i_3,\ldots i_{R+1}) = [\sum_{i_1=0}^{N+1} \alpha_{\ell}(i_1,i_2,\ldots i_R) a_{i_1,i_2,\ldots i_{R+1}}] \mathrm{f}(\mathbf{x}_{\ell+1} \mid Q_{\ell+1} = R+1, M), \quad \ell \geq R-1 \tag{2-5}$$

The algorithms utilising these variables can be extended to accommodate the higher dimensional structures. As these higher dimensional structures may be quite sparse, it is important to implement measures to avoid redundant calculations (Mari, Haton &

[13] Since the training process often needs extensive quantities of data, the faster Viterbi re-estimation algorithm was used in this work.

Kriouile, 1997). It is clear that processing higher-order HMMs can be very expensive. For example, in higher-order ergodic HMMs the number of transition calculations are proportional to $N^{R+1}L$ and the memory required to store the forward variables is proportional to $N^{R}L$. The pdf. calculations are unaffected by the HMM order.

2.2.4 Different types of HMMs

We now discuss dimensions on which we distinguish between different types of HMMs. We do not intend to provide a full taxonomy; new models are evolving continually. The discussion will be brief, with references for further investigation.

There are three main sources which give rise to different HMM variants, namely the state pdf.s, the topology of the model (the structure dictating which states are coupled to which) and the Markov order of the state transition probabilities.

1) Most variants find their origin in the type of pdf. used. Table 2-1 lists some examples. In this research we are not concerned with pdf. variants and thus restrict ourselves to continuous (diagonal Gaussian) pdf.s.

2) The ultimate application of an HMM often determines its topology. Phoneme and word models often use a left to right form (see Figure 2-1 on p. 22, also Young, 1996). Early language recognition systems employed a ergodic HMM (see Figure 4-1 on p. 92, also Savic, Acosta, & Gupta, 1991). When duration modelling is important, Ferguson models can be used (see Figure 2-17 on p. 50, also Ferguson, 1980). One of the contributions of this research, is to show that some ad hoc topologies are, in reality, the end-result of designs using higher-order HMM specifications.

3) The final factor we consider is the Markov order of the model e.g. first-order, second-order etc., (see Figure 2-12 on p. 46, also Mari *et al.*, 1997). We have already stated that the Markov order and the topology are related. The ORED algorithm, to be discussed in 2.3.1, provides a mechanism to reduce a high-order HMM to an equivalent first-order HMM, thereby making its topology explicit.

Combining the above gives rise to diverse models such as (for example) a *second-order ergodic semi-continuous HMM*.

Table 2-1. HMMs with different pdf. types.

Density Type	HMM Type	Reference
Discrete	Discrete	Rabiner, Levinson & Sondhi, 1983.
Continuous	Continuous	Juang, 1985.
Continuous mixture	Mixture density	Juang & Rabiner, 1985.
Tied mixture	Semi-continuous	Huang & Jack, 1989.
Neural approximation	Hybrid	Bourlard & Wellekens, 1990.

2.2.5 Related concepts: Markov chains and N-grams

Markov chains and N-grams are concepts with a close relationship to HMMs. The former is frequently used in statistical modelling (Leon-Garcia, 1989), while the latter is currently very popular as the language model in large automatic speech recognition (ASR) systems (Young, 1996). Although higher-order versions are possible, Markov chains are mostly used in their first-order version. N-grams, on the other hand, are specifically popular in their higher-order forms. As we will soon see, they actually represent the same concept and are a degenerate case of the HMM. To put the work on higher-order HMMs here in proper perspective, it is important to understand these relationships.

2.2.5.1 Markov chains

If, in an HMM, we completely ignore the notion of feature vectors and pdf.s describing them, but instead focus only on the random state changes in the model, the model would function as a Markov chain (Howard, 1971; Leon-Garcia, 1989, p. 437; Deller, Proakis & Hansen, 1993, p. 682). In an HMM, the activity of this underlying Markov chain is obscured (*hidden*) by another stochastical process, namely the pdf.s that produce the observations. At first glance it would seem that the presence of the pdf.s in the HMM is the determining factor in making it “hidden”. The presence of the pdf.s transforms a “singly” stochastic process (Markov chain) into a “doubly” stochastic process (HMM). There is a caveat though. As can be seen from Figure 2-3 and Figure 2-4, any discrete HMM (see Section 2.2.4) can be written without any explicit pdf.s.

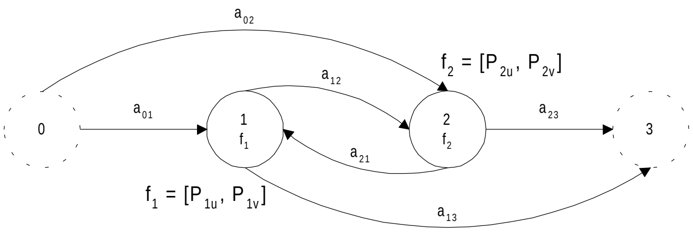

Figure 2-3. Discrete HMM.

The difficulty with interpreting Figure 2-4 as a Markov chain is that the observation symbol *u* is associated with both states (*1*,*u*) and (*2*,*u*). Similarly *v* is associated with both (*1*,*v*) and (*2*,*v*). With Markov chains the states of the model can be directly observed from its output. In the given example this is clearly not possible.

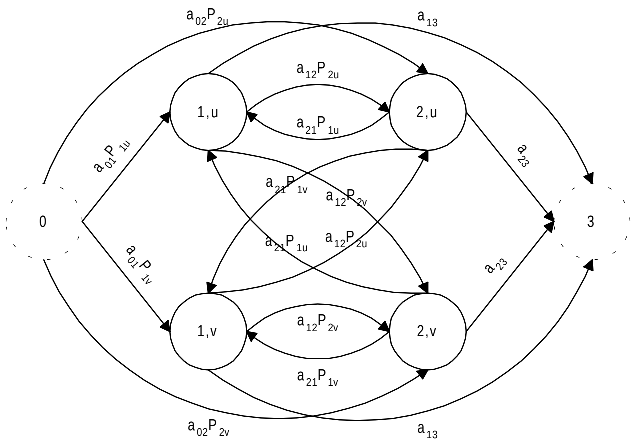

Figure 2-4. Discrete HMM without pdf.s.

Overlapping pdf.s lead to ambiguities in the chain, which is now the form in which the *hidden* part of the HMM emerges. Any Markov chain can be represented as an HMM, but then the pdf.s will not overlap. Stated differently, any HMM with non-overlapping pdf.s is not really a *hidden* Markov model.

2.2.5.2 N-grams

In automatic speech recognition (ASR) research, so-called "N-grams" are quite popular as language models (Bahl, Jelinek & Mercer, 1983; Levinson, 1985; Riccardi, Pieraccini & Bocchieri, 1996). The function of these language models is to dictate (or restrict) the allowable sequence of words, thereby reducing the perplexity[14] of the total recogniser. A tri-gram specification, for instance, will provide probabilities $p_{ijk} = \mathrm{P}(w_k \mid w_i, w_j)$ where w_i, w_j, w_k is a sequence of successive words. These N-grams have some interesting associations (Levinson, 1986):

[14] A measure related to entropy, describing the average "branching factor" (number of choices) at each decision point. It quantifies the difficulty of the task. See Bahl, Jelinek and Mercer (1983) for details.

1) Clearly, if there was no uncertainty about the identity of the fundamental units under consideration (words in the above case), these N-gram word orders could be specified via a higher-order Markov chain. However, the N-gram language model is often integrated with lower level HMMs to form a single, large, monolithic HMM.

2) Both HMMs and N-grams are instances of stochastic regular grammars (Chomsky type 3 grammar).

The first point deals with the relationship between Markov chains and discrete *hidden* Markov models, something we have already discussed in the previous section. If the category identities (words in this case) could be determined with absolute certainty, the ambiguity and hence “hidden” aspect disappears, resulting in a Markov chain. In an ASR system, however, the word identities are uncertain. The N-gram probabilities in this context are merely replacements for the actual probabilities coupling these more nebulous categories. Their estimation is based on external deterministic knowledge (via counting and interpolation techniques) and not on the actual probabilistic categories within the system.

The second point above is just a restatement in alternative terms of the first point. Without digressing too far afield into the theory of stochastic grammars (Fu, 1982; Schalkoff, 1992), it can be said that an N-gram results in an unambiguous stochastic regular grammar, whereas an HMM generally results in an ambiguous one. This makes an N-gram a special degenerate case of the HMM. Although it can be processed, using normal HMM techniques, the special requirements of the “hidden” part of the HMM in reality is not necessary. If HMM forward-backward re-estimation is used to infer N-gram probabilities, only one state sequence is viable for any observation sequence. From this, the re-estimation reduces to a simple counting procedure (Bahl *et al.*, 1983). In contrast, for a general HMM this information is only known in probabilistic terms, calling for more involved processing. Deller *et al.* (1993, pp. 784-785) reflects on the relationship between stochastic regular grammars and N-grams, but fails to take into account the role that ambiguity plays.

2.3 Reducing a high-order to an equivalent first-order HMM

Instead of finding algorithms for HMMs of different orders, the following algorithm will transform any (mixed- or fixed-order) HMM to an HMM1 that is mathematically equivalent to the original. Proof of this equivalence is given in Section 2.3.3. The underlying (and quite intuitive) idea is to map the states of the original HMM M_R, using a twofold product of the original state space, to a reduced model M_{R-1}. Transitions in this new state space implicitly have an extra time step of state history specified, thereby reducing the order of the transition by one.

The basic notion is not new, Howard (1971, p. 4) writes in the context of standard (not hidden) Markov processes that "*… suppose that the last two states occupied both influenced the transition to the next state. Then we could define a new process with N^2 states – each state in the new process would correspond to a pair of successive states in the old process. …Any dependence of future behaviour on a finite amount of past history can, at least theoretically, be treated in the same way.* " The next section though, makes this notion concrete with an explicit algorithm, while at the same time extending it to hidden Markov models. As will become evident, there are several subtleties that arise during the actualisation of this basic idea.

In the following discussion, states in M_R will be denoted by Q while the variable S will be used for those in M_{R-1}. It will be assumed that M_R has N emitting states, augmented by initial and terminal null states, respectively denoted by $Q=0$ and $Q=N+1$. All the state indexes of M_R are assumed to be single (not composite) values.

In Appendix A a detailed example is provided that illustrates the ORED algorithm. Section 2.3.2 comments on the role of the different steps in the algorithm. Together they are useful guides for studying this algorithm.

2.3.1 The ORder rEDucing (ORED) algorithm

1) Create the set of states of M_{R-1}:

   a) Create the states $S=0, S=1, \ldots S=N+1$ in M_{R-1}. (Only $S=0$ for fixed-order)

   b) For each of the available higher-order ($R' \geq 2$ with R' the order) transition probabilities $a_{i_1 i_2 \ldots i_{R'+1}}$ present in the model, add the states $(i_1, i_2), (i_2, i_3), \ldots (i_{R'}, i_{R'+1})$ to M_{R-1} without duplication.

   c) If the terminal state $Q = N+1$ has transitions originating at more than one state entering it and at least one of them is of higher order ($R' \geq 2$), create a new terminal null state $S = N+2$.

2) Allocate transitions and their probabilities:

   a) First-order probabilities $a'_{i_1 i_2}$:

      If state $S=(i_1, i_2)$ exists, identify all available states $S=i_1$ and $S=(i,i_1)$, $0 \leq i \leq N+1$ as source states for the transition.

The transition probability $a'_{i_1 i_2}$ links each of them to destination state $S=(i_1,i_2)$, resulting in transition probabilities of the form $a_{i_1(i_1,i_2)}=a'_{i_1 i_2}$ and $a_{(i,i_1)(i_1,i_2)}=a'_{i_1 i_2}$. If there is more than one source state, their links are tied.

Otherwise it is placed from source states $S=i_1$ and $S=(i,i_1)$ (as defined above) to destination state $S=i_2$, resulting in transition probabilities of the form $a_{i_1 i_2}=a'_{i_1 i_2}$ and $a_{(i,i_1)i_2}=a'_{i_1 i_2}$. If there is more than one source state, their links are tied.

b) Higher-order probabilities $a'_{i_1 i_2 \ldots i_{R'+1}}, R' \geq 2$: These probabilities (one or more) are placed between states $S=(i_{R'-1},i_{R'})$ and $S=(i_{R'},i_{R'+1})$, that is $a_{(i_1,i_2)(i_2,i_3)\ldots(i_{R'},i_{R'+1})}=a'_{i_1 i_2 \ldots i_{R'+1}}$.

c) If a new terminal state $S=N+2$ has been created, create a unit probability transition between source states $S=N+1$, $S=(i,N+1)$ and destination state $S=N+2$. That is $a_{(N+1)(N+2)}=a_{(i,N+1)(N+2)}=1$.

3) Dead state removal:

Remove all states that cannot be reached from the initial state or from which the termination state cannot be reached (typically an iterative algorithm is used for this). Redundant null states can also be removed at this stage (optional).

4) Allocate pdf.s:

   a) Every existing state $S=(i,k)$ and/or $S=k$ shares[15], via parameter tying, the same pdf. as state $Q=k$. That is, they are both either null states, or

   $\mathrm{f}(\mathbf{x} \mid S=k, M_{R-1}) = \mathrm{f}(\mathbf{x} \mid S=(i,k), M_{R-1}) = \mathrm{f}(\mathbf{x} \mid Q=k, M_R)$.

   b) If a new termination state $S=N+2$ was created, it is a null state.

5) Iterate:

   a) Rename all the composite state indexes to unique single indexes, and modify the indexes in the transition probabilities accordingly. The resulting model is the $R-1$th-order equivalent of the original Rth-order model.

   b) All of the above steps can be repeated until all transition probabilities are subscripted by only two state indexes. By definition this is an HMM1. We show in Section 2.3.3 that it is equivalent to the original higher-order model.

6) Merge states (Unnecessary for fixed-order HMMs):

   Merge tied states that have the same destination for each of the shared transitions. Renumber the states and transition probability indexes to reflect the smaller number of states.

[15] If the Mealy form HMM was used, a separate pdf. will be associated with each transition probability. This greatly escalates the number of parameters in the model.

2.3.2 Notes on the ORED algorithm

The states created in step 1a) are needed as the beginning and/or end points for certain of the first-order transition probabilities which occur in a mixed-order model. Step 1b) creates the necessary states to ensure that one extra step of state history is made explicit in the new model. Step 1c) ensures that the model will have a unique termination state. This step is not strictly necessary, but results in a cleaner representation.

Step 2a) inserts the first-order transition probabilities. When a first-order transition probability $a_{i_1 i_2}$ shares a transition with higher-order probabilities, it is necessary to duplicate the first-order probability over the different contexts created to reduce the order of the higher-order probabilities. This spontaneously leads to states sharing identical transition probabilities (so-called tied links). With fixed-order models no such link tying should occur. Step 2b) now reduces the order of the higher-order probabilities by one by inserting them between states specially created to "remember" implicitly one extra time step. There can be multiple probabilities associated with each transition. The sequence $i_1 \cdots i_{R'-2}$ is used to enumerate those probabilities. Step 2c) reflects the desire to work with a single termination state and is included for a cleaner representation. The null states leading to the termination state $S = N + 2$ are actually superfluous and may be optionally eliminated.

It may happen that some of the new states are not in a path leading from the initial to the terminal state (see Section 2.4.3). Step 3) eliminates them. Since the pdf. associated with each state is not affected by the order of the model, step 4) must ensure that the status quo is maintained in the new model. To make sure that the model responds properly during training, replicated pdf.s must be implemented as shared. With the previous steps in the algorithm having reduced the order of the model by one, step 5) merely uses recursion to continue the process. Step 5a) does the necessary renaming to get the new model in the notation[16] assumed by step 1). Step 5b) is the actual recursion. Step 6) merges tied states that share the same transition targets. This step needs to be done on the final first-order model, since what might appear to be the same target state at one of the intermediate stages of the algorithm, might at a later stage split into different states.

2.3.3 Proof that the ORED algorithm results in equivalent first-order models

As can be seen from the above, the normal R^{th}-order model, and ORED first-order version, contain the same free parameters in a different structure. The following proofs show that any higher-order HMM M_R (fixed- or mixed-order) and its M_{R-1} version which results from one iteration of the ORED algorithm, are mathematically equivalent. Repeated application of the ORED algorithm therefore ultimately results in an M_1 equivalent to the original M_R. In the following M_R consists of N emitting states, augmented by an initial and a terminal null state. It is also assumed that all of the states of M_R lie somewhere on a connected path joining the initial state and termination states. That is, we assume that there are no “dead” states in M_R.

[16] The use of single state indexes in the base model is only to keep the formulation simpler. Another attractive alternative would be to use composite indexes indicating the applicable history of states according to the order of the model.

Definition: Any two models M_R and M_{R-1} are defined as equivalent if $\mathrm{f}(\mathbf{X}_1^L \mid M_R) = \mathrm{f}(\mathbf{X}_1^L \mid M_{R-1})$ for any arbitrary observation sequence $\mathbf{X}_1^L$. In other words, two models are only considered as equivalent if they yield the same likelihood, regardless of the specific observation sequence.

Theorem 1: Consider any sequence of states

$$Q_0^{L+1} = \{Q_0 = 0, Q_1, Q_2, \cdots, Q_L, Q_{L+1} = N+1\} \tag{2-6}$$

that may, with non-zero probability, follow each other in M_R. From this construct the state sequence S_0^{L+1} for M_{R-1}:

$$S_\ell = \begin{cases} Q_0, & \ell = 0 \\ \begin{cases} (Q_{\ell-1}, Q_\ell) & \text{if it exists} \\ Q_\ell & \text{otherwise} \end{cases}, & 1 \le \ell \le L+1 \end{cases} \tag{2-7}$$

Then $Q_0^{L+1} \leftrightarrow S_0^{L+1}$ forms a one-to-one mapping of all valid state sequences in M_R to all valid state sequences in M_{R-1}.

Proof: Consider two successive states in this sequence namely

$$S_{\ell-1} = \begin{cases} (Q_{\ell-2}, Q_{\ell-1}) & \text{if it exists} \\ Q_{\ell-1} & \text{otherwise} \end{cases} \text{ to } S_\ell = \begin{cases} (Q_{\ell-1}, Q_\ell) & \text{if it exists} \\ Q_\ell & \text{otherwise} \end{cases}.$$

Steps 1b) and 2) of the ORED algorithm allow for any of the four possible combinations of $S_{\ell-1}$ and S_ℓ. Since $Q_{\ell-2}^{\ell}$ lies somewhere on a connected path joining the initial state and termination states of M_R, step 2) of the ORED algorithm guarantees that $S_{\ell-1}^{\ell}$ lies somewhere on a connected path joining the initial state and termination states of M_{R-1}. Therefore, S_0^{L+1} is a valid sequence of states for model M_{R-1}. Furthermore, from the given construction, Q_0^{L+1} maps to a unique S_0^{L+1} and vice versa. Therefore $Q_0^{L+1} \leftrightarrow S_0^{L+1}$ forms a one-to-one mapping of all valid state sequences in M_R to all valid state sequences in M_{R-1}. ■

Theorem 2: The likelihood of an observation sequence $\mathbf{X}_1^L$ and a specific state sequence Q_0^{L+1} in M_R is equal to the likelihood of $\mathbf{X}_1^L$ and the associated unique state sequence S_0^{L+1} (defined in Theorem 1) in M_{R-1} i.e.

$$\mathrm{f}(\mathbf{X}_1^L, Q_0^{L+1} \,|\, M_R) = \mathrm{f}(\mathbf{X}_1^L, S_0^{L+1} \,|\, M_{R-1}) \,.$$

Proof: Firstly, determine the likelihood using M_R. Let R'_ℓ be the order of the transition from state $Q_{\ell-1}$ to state Q_ℓ. Using the definition of conditional probability, as well as the HMM assumptions (AS.1 and AS.2, p. 21), yields

$$\begin{aligned} \mathrm{f}(\mathbf{X}_1^L, Q_0^{L+1} \,|\, M_R) &= \mathrm{P}(Q_{L+1} \,|\, \mathbf{X}_1^L, Q_0^L, M_R)\, \mathrm{f}(\mathbf{X}_1^L, Q_0^L \,|\, M_R) \\ &= \mathrm{P}(Q_{L+1} \,|\, Q_{L+1-R'_{L+1}}^{L}, M_R)\, \mathrm{f}(\mathbf{x}_L \,|\, \mathbf{X}_1^{L-1}, Q_0^L, M_R)\, \mathrm{f}(\mathbf{X}_1^{L-1}, Q_0^L \,|\, M_R) \\ &= a'_{Q_{L+1-R'_{L+1}}^{L+1}}\, \mathrm{f}(\mathbf{x}_L \,|\, Q_L, M_R)\, \mathrm{f}(\mathbf{X}_1^{L-1}, Q_0^L \,|\, M_R). \end{aligned} \tag{2-8}$$

The last factor on the right hand side of (2-8) is of the same form as the left-hand side. Recursive application of this expression then yields:

$$\mathrm{f}(\mathbf{X}_1^L, Q_0^{L+1} \,|\, M_R) = a'_{Q_{L+1-R'_{L+1}}^{L+1}}\, \mathrm{f}(\mathbf{x}_L \,|\, Q_L, M_R) a'_{Q_{L-R'_L}^{L}} \cdots \mathrm{f}(\mathbf{x}_2 \,|\, Q_2, M_R) a'_{Q_{2-R'_2}^{2}}\, \mathrm{f}(\mathbf{x}_1 \,|\, Q_1, M_R) a'_{Q_0^1}. \tag{2-9}$$

Let R_ℓ be the order of the transition from state $S_{\ell-1}$ to state S_ℓ. Following the development of (2-9), the likelihood given M_{R-1} can be shown to be:

$$\mathrm{f}(\mathbf{X}_1^L, S_0^{L+1} \,|\, M_{R-1}) = a_{S_0^1}\, \mathrm{f}(\mathbf{x}_1 \,|\, S_1, M_{R-1}) a_{S_{2-R_2}^2}\, \mathrm{f}(\mathbf{x}_2 \,|\, S_2, M_{R-1}) \cdots a_{S_{L-R_L}^L}\, \mathrm{f}(\mathbf{x}_L \,|\, S_L, M_{R-1}) a_{S_{L+1-R_{L+1}}^{L+1}} . \tag{2-10}$$

It is now necessary to establish the relationship between (2-10) and (2-9). From (2-7), and (2-10), the transition from state $Q_{\ell-1}$ to state Q_ℓ (with probability $a'_{Q_{\ell-R'_\ell}^{\ell}}$)

corresponds to the transition from state $S_{\ell-1}$ to state S_ℓ with probability $a_{S^{\ell}_{\ell-R_\ell}}$. From step 2a) $R'_\ell = 1$ implies that $R_\ell = 1$ and $a_{S^{\ell}_{\ell-R_\ell}} = a_{S^{\ell}_{\ell-1}} = a'_{Q^{\ell}_{\ell-1}}$. From step 2b) $R'_\ell > 1$ implies that $R_\ell = R'_\ell - 1$ and $a_{S^{\ell}_{\ell-R_\ell}} = a'_{Q^{\ell}_{\ell-R'_\ell}}$ (this is the order reduction step). In general therefore

$$a_{S^{\ell}_{\ell-R_\ell}} = a'_{Q^{\ell}_{\ell-R'_\ell}} \tag{2-11}$$

From ORED step 4) it also follows that

$$\mathrm{f}(\mathbf{x}_\ell \mid S_\ell, M_{R-1}) = \mathrm{f}(\mathbf{x}_\ell \mid Q_\ell, M_R) \tag{2-12}$$

Substitute (2-11) and (2-12) into (2-10) to obtain the required result

$$\mathrm{f}(\mathbf{X}_1^L, Q_0^{L+1} \mid M_R) = \mathrm{f}(\mathbf{X}_1^L, S_0^{L+1} \mid M_{R-1}) .$$ ■

Theorem 3: The likelihood of attributing an observation sequence to M_R is equal to it being attributed to M_{R-1}.

Proof: Theorem 2 states that $\mathrm{f}(\mathbf{X}_1^L, Q_0^{L+1} \mid M_R) = \mathrm{f}(\mathbf{X}_1^L, S_0^{L+1} \mid M_{R-1})$, where the two state sequences correspond as is defined in Theorem 1. It is also known from Theorem 1 that for these two models there is a one-to-one mapping of such sequences. Therefore

$$\sum_{\forall S_0^{L+1}} \mathrm{f}(\mathbf{X}_1^L, S_0^{L+1} \mid M_{R-1}) = \sum_{\forall Q_0^{L+1}} \mathrm{f}(\mathbf{X}_1^L, Q_0^{L+1} \mid M_R) \tag{2-13}$$

The respective state sequences that are summed, are mutually exclusive and cover the whole sample space of such sequences (they thus form a partition). Therefore both sides of (2-13) reduce to marginal pdf.s giving the required result:

$$\mathrm{f}(\mathbf{X}_1^L \mid M_R) = \mathrm{f}(\mathbf{X}_1^L \mid M_{R-1})$$ ■

This result shows that the two models result in identical likelihoods. *One iteration of the ORED algorithm results in an equivalent model with order reduced by one.*

2.4 Examples using the ORED algorithm

We now illustrate the ORED algorithm by means of examples. The example in Section 2.4.1 below is chosen to illustrate several aspects of the ORED algorithm, amongst others the use of both single and composite states, as well as the use of tied states leading to state merging. Higher-order Markov models in general are prone to subtle difficulties, not so easily noticed when the model is represented in the traditional way. The examples in Sections 2.4.2 and 2.4.3 illustrate this point with simple models containing some inconsistencies. If undetected such inconsistencies may hamper or even fatally flaw the HMM.

2.4.1 A simple mixed-order model

Figure 2-5 is a mixed-order HMM with a maximum order of three. The result of applying one iteration of the ORED algorithm (up to step 4)) is shown in Figure 2-6, while the final equivalent HMM1 is in Figure 2-7. A detailed step-by-step exposition is given in Appendix A.

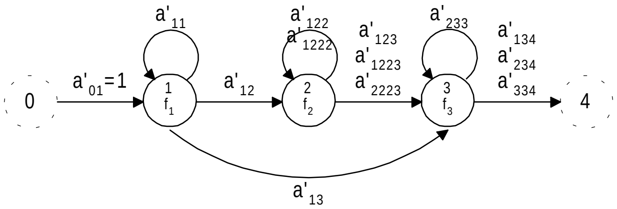

Figure 2-5. A mixed-order HMM (maximum order 3).

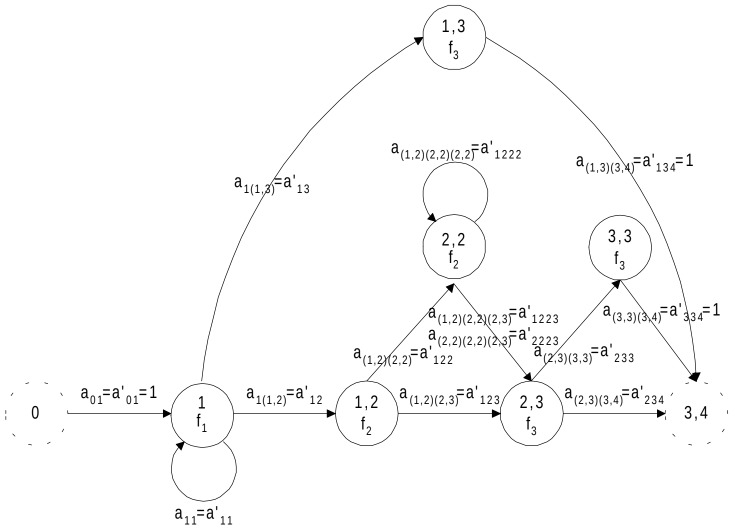
1,3
f_3
$a_{(1,2)(2,2)(2,2)}=a'_{1222}$
$a_{(1,3)(3,4)}=a'_{134}=1$
$a_{1(1,3)}=a'_{13}$
2,2
f_2
3,3
f_3
$a_{(1,2)(2,2)(2,3)}=a'_{1223}$
$a_{(2,2)(2,2)(2,3)}=a'_{2223}$
$a_{(3,3)(3,4)}=a'_{334}=1$
$a_{(1,2)(2,2)}=a'_{122}$
$a_{(2,3)(3,3)}=a'_{233}$
0
$a_{01}=a'_{01}=1$
1
f_1
$a_{1(1,2)}=a'_{12}$
1,2
f_2
$a_{(1,2)(2,3)}=a'_{123}$
2,3
f_3
$a_{(2,3)(3,4)}=a'_{234}$
3,4
$a_{11}=a'_{11}$

Figure 2-6. Equivalent of Figure 2-5 with highest order two. The intermediate composite indexing scheme is retained for easier reference to the algorithm.

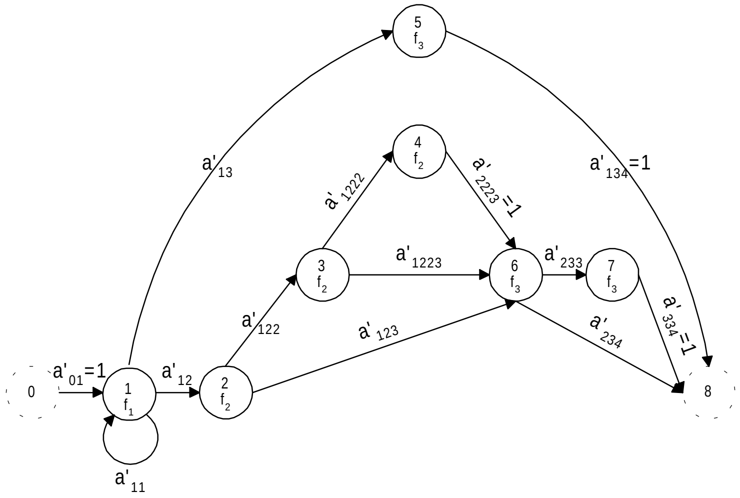

Figure 2-7. First-order equivalent of Figure 2-5.

There are a couple of points that one can easily miss when interpreting this model from Figure 2-5; apparently states 1 to 3 allow self-loops. In reality it is only true of state 1. As is evident from the associated pdf. indexes in Figure 2-7, state 2 allows a maximum of two repetitions. State 3 allows only one repetition and then only under the precondition that the previous state was state 2. Also, ignoring a'_{01}, which is obviously a unitary transition, it would appear from Figure 2-5 that there are twelve other (free) transition probabilities in the model. However, Figure 2-7 makes it clear that three further probabilities are also unitary. Although all these facts could also be gleaned by careful inspection of Figure 2-5, they are quite apparent in Figure 2-7.

2.4.2 Inconsistent transition probabilities

It is well known from HMM1 theory that the probabilities of transitions leaving a specific state should sum to unity. The same principle, of course, also holds for higher-order transition probabilities. In this case transition probabilities leaving a specific history of states should sum to unity. Recognising a violation of this principle is, however, slightly more subtle with mixed-order versions, Figure 2-8 being a (simple) case in point.

Implicit to the model is that $a'_{012} = 0$ (as it is omitted). To satisfy the requirement that probabilities must sum to unity $a'_{011} + a'_{012} = 1$ must hold. This implies that $a'_{011} = 1$. The first-order a'_{11} implies[17] a (hidden) second-order $a'_{011} = a'_{111}$. Therefore $a'_{111} = 1$. Similarly, $a'_{111} + a'_{112} = 1$. From this follows then that $a'_{112} = 0$, thereby breaking the only possible path to the termination state.

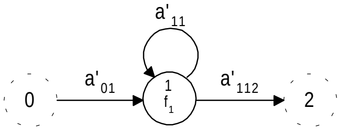

Figure 2-8. An inconsistent mixed-order model.

[17] When a first order transition probability is specified, its non-dependence on states earlier than the current, simply means that all its higher order extensions are equal, thereby making their explicit specification redundant.

When trying to find a first-order equivalent for this model, the inconsistencies become explicit. Execution of the ORED algorithm shows that states 1 and (1,1) in Figure 2-9 should be tied. That implies that the two sets of transition probabilities originating at them should be constrained to be identical. The two states, however, do not have the same number of transition probabilities making it impossible to do this. In addition, it is easily seen from this figure that $a'_{112} = 0$, thereby breaking the link to the termination state.

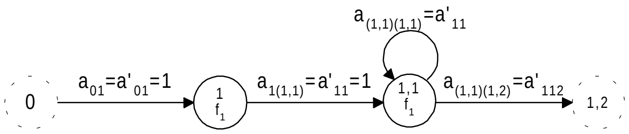

Figure 2-9. First-order equivalent (still using the intermediate indexing scheme). States 1 and (1,1) should have been tied, but they have a different number of transitions.

2.4.3 Non-supporting transition probabilities

Another subtle flaw, that can be specified into higher-order HMMs (fixed or mixed-order), occurs when the sequence of states presumed by a higher-order transition probability is not supported by previously occurring transitions or vice versa. This then results in sections of the model that are not on a path which joins the initial and termination states. Such dangling sections will be removed by the pruning and, except for the loss of some unusable parameters, are not necessarily a problem. If, however, this section was a link on the only path joining the initial and termination states, a “broken” model would result.

As an example, the model of Figure 2-10 apparently has eight free transition probabilities. Figure 2-11, however, shows that, due to “dangling” state (2,2), transition probabilities $a'_{022} = a'_{122} = 0$. This again, in its turn, implies that

$a'_{023} = a'_{123} = 1$, ultimately leaving four transition probabilities of which only two can be varied independently. This will certainly affect the modelling capability of this HMM.

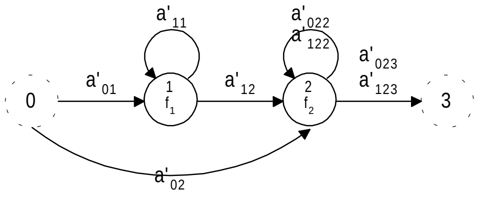

Figure 2-10. A mixed-order model with dangling states.

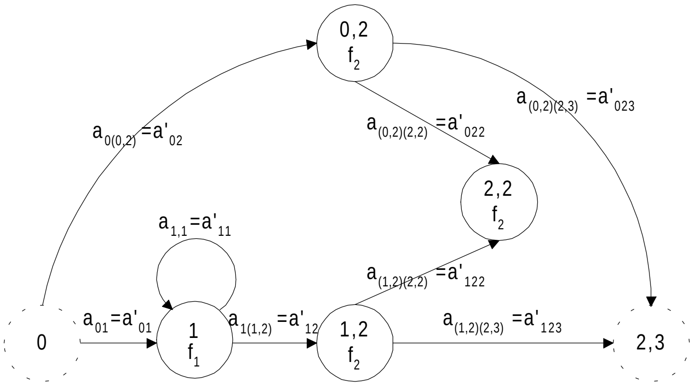

Figure 2-11. First-order equivalent (still using the intermediate indexing scheme). Note the dangling state (2,2).

2.5 Interesting higher-order HMM topologies

In Section 2.5.1 we illustrate the topology resulting from applying the ORED algorithm to a fixed order HMM. Such a model incorporates duration and context modelling. In Sections 2.5.2 and 2.5.3 we demonstrate topologies arising from special high order models which distinguish between these concepts.

2.5.1 Left to right HMM3 with one state skip

In this section the ORED algorithm is applied to the HMM3 of Figure 2-12. Figure 2-13 shows the resulting HMM2 equivalent to this model. The composite state indexes have been retained for illustrative purposes.

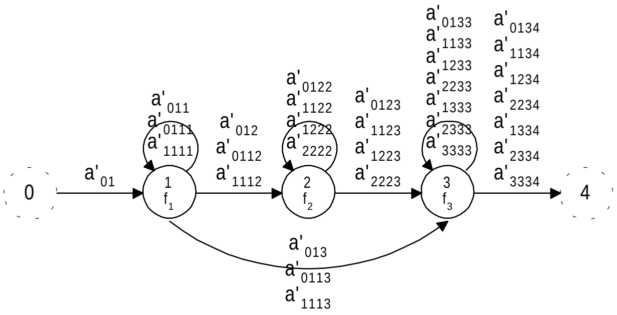
a'_{0133} a'_{1133} a'_{1233} a'_{2233} a'_{1333} a'_{2333} a'_{3333}
a'_{0134} a'_{1134} a'_{1234} a'_{2234} a'_{1334} a'_{2334} a'_{3334}
a'_{0122} a'_{1122} a'_{1222} a'_{2222}
a'_{0123} a'_{1123} a'_{1223} a'_{2223}
a'_{011} a'_{0111} a'_{1111}
a'_{012} a'_{0112} a'_{1112}
a'_{01}
0
1 f_1
2 f_2
3 f_3
4
a'_{013} a'_{0113} a'_{1113}

Figure 2-12. Third-order left-to-right HMM.

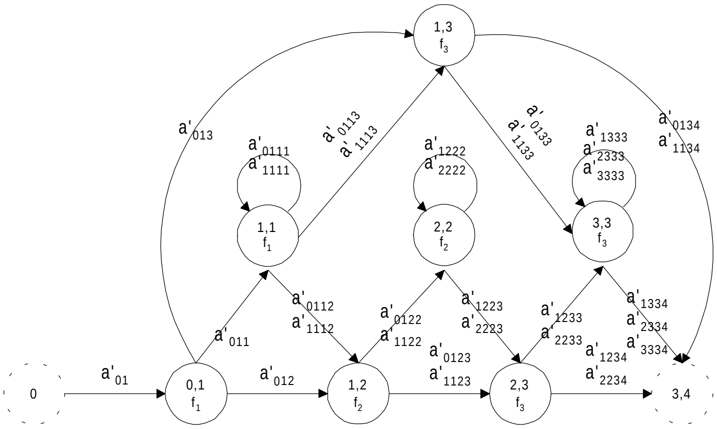

Figure 2-13. Equivalent HMM2[18] for the HMM3 in Figure 2-12, using the intermediate composite state indexing scheme.

In Figure 2-14 the equivalent HMM1 for this model is shown. As will be noted from these diagrams, the resulting models have a rich structure. The role that the context of

[18] If the first index is omitted in cases where a has 3 and a' has 4 indexes, thereby leaving only single probabilities on each transition, this would also be the equivalent HMM1 for Figure 2-2.

previous states plays in a given transition is explicit. Also note that the longest path without self-loops in Figure 2-1 occupies three physical time steps[19], six time steps for the second-order model[20] and nine time steps for the third-order model[21]. The shortest paths, of course, do not differ from the original first-order model. This makes for more precise duration modelling and is an alternative to often used phoneme models such as Ferguson models (Ferguson, 1980) and the models used in the Sphinx system (Lee, 1989, p. 55).

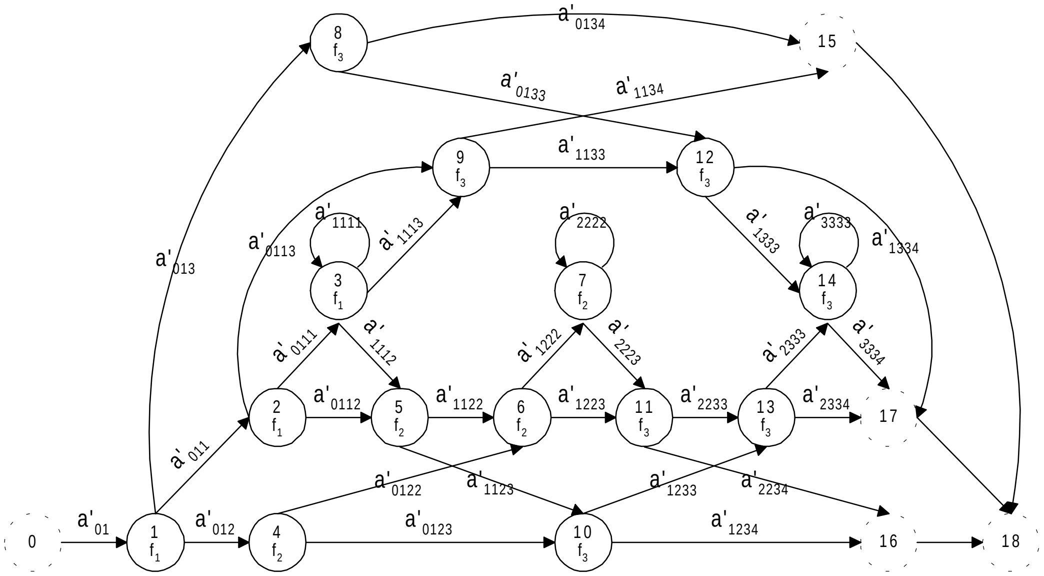

Figure 2-14. The HMM1 equivalent to the HMM in Figure 2-12. The states of the original model in Figure 2-12 correspond here to the set of states with the same pdf. (f_i).

[19] states 1, 2 and 3 in Figure 2-1.

[20] states (0,1), (1,1), (1,2), (2,2), (2,3) and (3,3) in Figure 2-13.

[21] states 1, 2, 3, 5, 6, 7, 11, 13 and 14 in Figure 2-14.

2.5.2 Duration modelling with high-order HMMs

In the previous section we saw that fixed-order HMMs model both context and duration information. With the following example we want to illustrate a topology arising from emphasising the duration modelling aspects of the model while neglecting the contextual modelling that is not directly involved with the modelling of the duration of a state. This type of modelling only involves states with self-loops. In such a state we need to identify the sets of departing transition probabilities that share the same destination and involve the same number of repetitions of this state. If all the transition probabilities in such a set are constrained to be identical regardless of prior states, the resulting model will be focussed towards modelling the duration of this state. This is formalised in the following algorithm.

Algorithm for a duration emphasised HMM:

1) Start with a fixed-order model of the appropriate order and topology.

2) For each state j with self-loops (i.e. has a $a'_{i_1^{D'-1}jj} \equiv a'_{i_1 i_2 \cdots i_{D'-1}jj}$ with D' the order of the transition): all transition probabilities $a'_{i_1^{D'-1}ji_{D'+1}}$ sharing the same destination state $i_{D'+1}$, with the same number of repetitions of j preceding it, are constrained to have an identical value.

3) States without self-loops: all transition probabilities leaving from such states are reduced to first-order.

This algorithm results in a mixed-order model. Since all higher-order transition probabilities are only used to model the number of self-transitions, we will term $D = \max(D')$ the *duration order*. Transition probabilities not involved with duration are restricted to first-order. As an example of this technique, Figure 2-15 applies these restrictions to the model of Figure 2-12**.** Reducing it to equivalent first-order via the ORED algorithm results in Figure 2-16. In later work we will refer to the stacks of states occurring in high-order duration models such as Figure 2-16, as *duration stacks*.

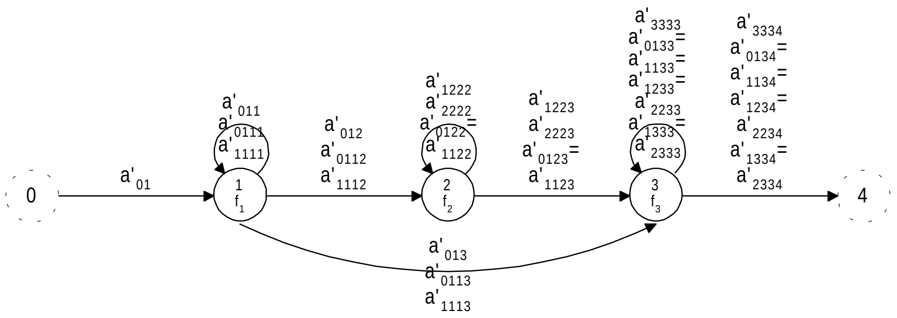

Figure 2-15. Durational left to right HMM3 with one state skip. Note the shared transitions which constrains Figure 2-12 to focus on duration modelling.

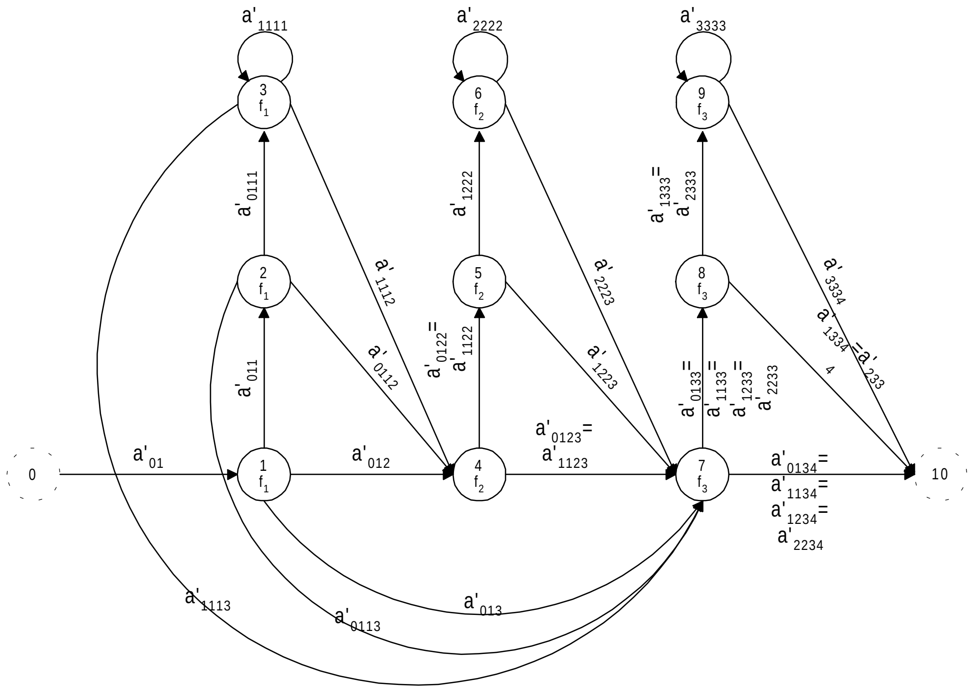

Figure 2-16. First-order equivalent of the HMM in Figure 2-15. Note the similarity to Ferguson model shown in Figure 2-17.

Each duration stack functions as a type of super-state that enables duration modelling. The individual members in such a stack only differ from each other in the duration information that they record via their associated transition probabilities. Transitions *entering* a stack are dependent only on immediately preceding stacks and not on earlier stacks, hence the context modelling is of first-order. Transitions *internal to, or departing from*, a stack do not incorporate any information on the duration of prior stacks. This is contrary to fixed-order models where both previous states and their duration affect the duration of later states.

As can be seen from Figure 2-17, this diagram has a striking resemblance to the well-known models used by Ferguson (1980) to model state duration. Reversing the direction of this model while maintaining the original order of the pdf.s results in a Ferguson model. Both model durations of up to three state repetitions explicitly (without any self-loops). Longer state durations are (in both cases) modelled with an exponentially decaying probability. This gives them the same capacity for modelling duration.

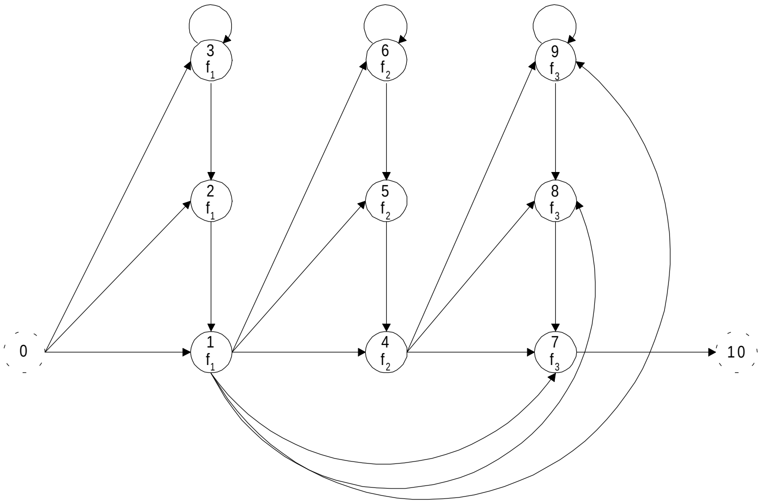

Figure 2-17. Ferguson model of Figure 2-1 (with 2 time lags).

2.5.3 Context modelling with high-order HMMs

In certain applications, of which language recognition (Chapter 4) is an example, we are more interested in a good specification of the sequence of *distinct* states that may follow each other, and not so much in the duration of each state. This can be accomplished by constraining transition probabilities, only varying in the number of repetitions of previous states, to be identical to each other. In other words, suppose we want to model only the effect of the *C* different previous states on a given transition. Let i^+ indicate one or more occurrences of *i*. Constrain, via state tying, all transition probabilities of the form $a'_{i_1 i_2^+ \cdots i_C^+ i_{C+1}}$ to have the same value. This is an interesting mixed-order topology. While the maximum order is infinity, a *context-order C* may be identified. Due to the heavy sharing of parameters, this topology can still be implemented in a finite structure. Figure 2-18 provides an example of this by applying this technique to the structure of Figure 2-12. Due to the infinite HMM order, this example is impossible to process directly with the ORED algorithm, as the model will grow without bounds. However, doing the appropriate merging on the pattern that emerges (see example in Appendix B), results in the structure of Figure 2-19.

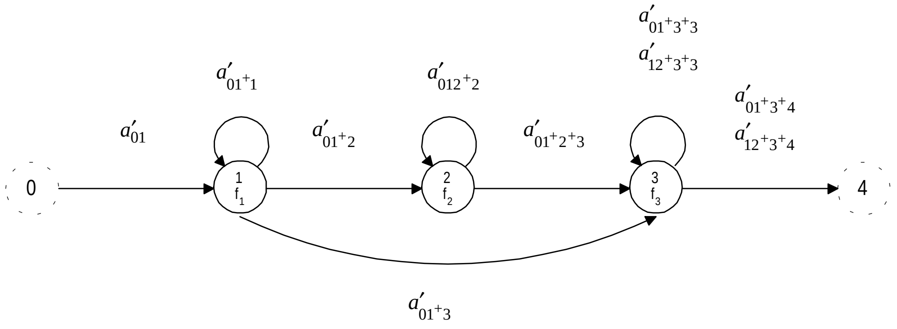

Figure 2-18. Third contextual order left to right HMM with one state skip. The notation k^+ is used to indicate one or more occurrences of index *k*. The (infinite) family of transition probabilities indicated by each symbol *a* in the figure are all identical. This is a mixed-order model with a maximum order of infinity.

Notice how, regardless of the number of repetitions of any specific state, the sequence of the last three distinct states is remembered. This is a very simple model, but still suffices to illustrate the "longer" state memory. For example, assume that state 3 of the HMM in Figure 2-12 always produces at least three observations. When making a transition to state 4, these three observations will fill the whole state history taken into account by the third-order transition probabilities. Due to this, information about whether the *distinct* state preceding state 3 was state 2 or state 1, is not being taken into account any more. The repetitions of state 3 effectively push this information beyond the horizon of visible prior states taken into account by the training algorithm when estimating the transition probabilities. Therefore, when fixed-order models are trained on data with such a repetitive nature[22], the implicit context order of the models (or sections of it), will be decreased to $C < R$. The HMM of Figure 2-19, however, does not suffer this problem, but guarantees to maintain its C. In Section 3.3.2 we will provide an efficient algorithm for developing and training this topology.

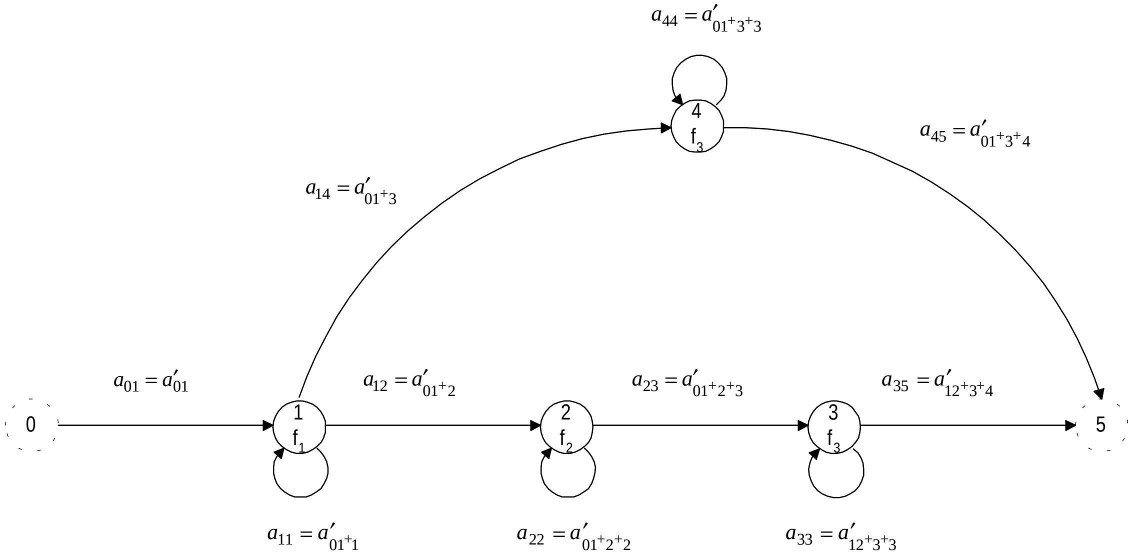

Figure 2-19. First-order equivalent of the HMM in Figure 2-18.

Applying duration and context modelling in tandem results in powerful alternatives to standard high-order HMM modelling. These topologies allow independent variation of context and duration orders. This topic will be developed further in Section 3.3.3,

[22] In Section 4.3 p. 91 we will see that the typical analysis applied to speech puts it in this category.

following on the development of the FIT algorithm. In Section 4.7 we will illustrate its application in the context of automatic language recognition.

2.6 Relationship of prior work to the ORED approach.

In this section we will not concern ourselves with Markov chains or N-gram models, but will focus only on how prior higher-order HMMs formulations relate to the ORED reduction. The idea of using a twofold product of the original state space to simplify the formulation frequently surfaces in the literature. This is a very intuitive notion that spontaneously arises from the mathematical formulation of higher-order HMMs (see Section 2.2.3.2). It also forms the basis of the ORED algorithm.

Using this twofold product of the original state space, He (1988) derives a second-order Viterbi algorithm and indicates that extending the Viterbi algorithm to even higher orders follows along the same lines. This is very reminiscent of the ORED approach, the main difference being that, instead of increasing the order of the algorithm, the ORED algorithm uses this principle to reduce the order of the model.

Kriouile, Mari and Haton (1990) extend the available techniques by deriving an extended Baum-Welch re-estimation algorithm specific to second-order discrete HMMs. As in the first-order case, the forward and backward equations play a central role. In this case the forward variable is a three-dimensional structure $\alpha_\ell(j,k)$, instead of the two-dimensional $\alpha_\ell(k)$ found in first-order algorithms. The Baum-Welch algorithm is adapted accordingly. An alternative view is taken in the ORED algorithm. Instead of increasing the dimension of the forward variable (and likewise the backward one), we consider *(i,j)* and *(j,k)* in equation (2-4) to be composite indexes of single states. Although very similar to their view, this allows us to view the second-order equations as first-order ones, resulting in the ORED algorithm and the resulting model structures as illustrated in previous sections. This different viewpoint has important implications: instead of increasing the order of a specific algorithm, the ORED algorithm iteratively reduces the order of the model to being one. Then it can be processed with *any* of the standard first-order algorithms. In an isolated digit recognition experiment, Kriouile *et al.* compare the use of first-order HMMs with second-order versions and a third model termed as a “transition equivalent model”. The transition structure of this model is identical to what we would obtain using the ORED algorithm. Using this last model in multi-speaker experiments, they achieve an accuracy of 97% whereas their second-order models had an accuracy of 99%. In speaker-independent mode this transition equivalent model could not improve on the 91% of the first-order models, whereas the second-order model yielded 93%. We proved in Section 2.3.3 that this algorithm results in a model that is fully equivalent to the original higher-order model. It is unclear why they achieve results inferior to those of their second-order HMM[23].

A recent paper by Mari, Haton and Kriouile, (1997) also reveals some interesting correspondence with the ORED algorithm. The idea of using a twofold Cartesian product of states once again surfaces. An example, reproduced in Figure 2-20, is given of a simple HMM2 and its first-order equivalent.

[23] This may be attributed to different local optima due to different initialisation. Alternatively it could be that one of the other requirements for full equivalence, was not met.

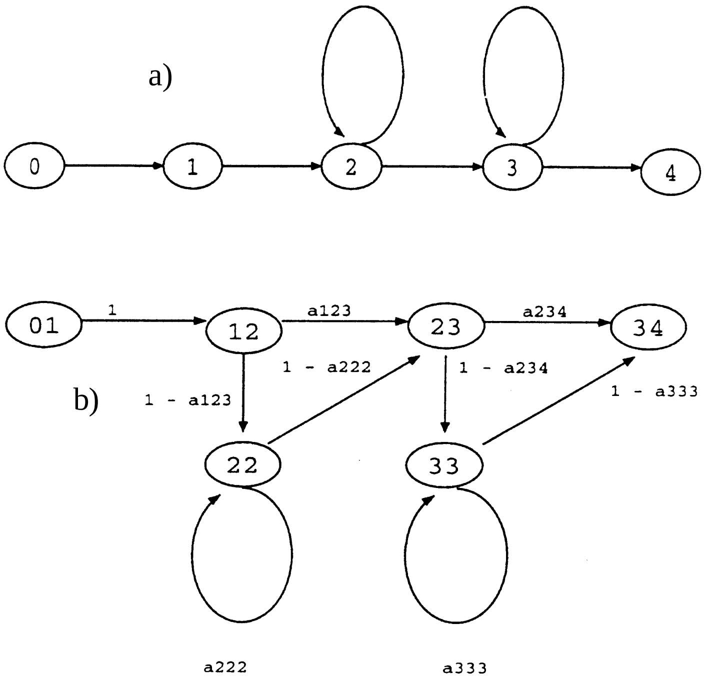

Figure 2-20. a) Second-order and b) equivalent first-order HMMs (from Mari *et al.*, 1997).

The first-order equivalent closely[24] resembles the topology that the ORED algorithm will generate. As far as we could ascertain, this is the first reference to a fully equivalent first order model in the published literature. It is likened to Ferguson models (Ferguson, 1980), which are specifically designed to improve HMM duration modelling. As shown in Section 2.5.2, a special mixed-order HMM results in Ferguson models as its first order equivalents, whereas the fixed order model also devotes some of its capacity to modelling state context. If the designer of the HMM topology is specifically interested in duration modelling, it will be more efficient to apply this or other approaches (Levinson, 1986) specifically designed for this purpose.

[24] The initial state (presumably a redundant null state) is missing from their equivalent.

After illustrating the equivalent first-order model, Mari *et al.* favour extending the Viterbi and Baum-Welch algorithms to second-order rather than using first-order equivalents. As reason for this they motivate that the latter dramatically increases the number of states. While this remark is obviously true, further investigation reveals that this increase in states need not be a disadvantage. From the preceding work on the ORED algorithm, it should be clear that the number of free parameters is not increased in the process. Tied pdf.s only need to be evaluated once, and the same number of transition probabilities will have to be considered in both cases. (As they are working with a (fixed-order) HMM2, no tying of transitions will occur.) Therefore, the use of the equivalent first-order model will not have any negative impact on the processing requirements.

As far as memory requirements are concerned, instead of working with the three-dimensional $\alpha_t(j,k)$ structure of equation (2-4), the equivalent HMM1 uses a two-dimensional structure of the same size. One of the important findings of Mari *et al.* actually deals precisely with this aspect. For computational efficiency, they implement special measures to avoid combinations of *(i,j)* or *(j,k)* that are not coupled via a transition in the calculation of (2-4) or its backward version. This calls for the use of sparse structures. This situation is not confined to high-order models. In our implementation using first-order models, such sparse structures are also utilised. Therefore, their algorithm will have the same memory requirements as those resulting from the ORED algorithm followed by standard first-order processing.

To summarise, previous work (He, 1988; Kriouile *et al.*, 1990; Mari *et al.*, 1997) focused on *extending algorithms* for processing *second-order* HMMs. Although the authors indicate that extension to algorithms for HMMs of order higher than two will follow a similar pattern as the extension to second-order, no such algorithm is presented. Their algorithms are based on the same principles as the ORED algorithm and there is an awareness of the possibility of reducing the order of HMMs. However, no algorithms formalising this concept are given (studying the ORED algorithm will reveal aspects that are not so intuitive as the notion on which the algorithm is based). In contrast to extending a specific algorithm to a higher order, the objective of our ORED algorithm is to *transform any higher-order (also mixed order) HMM* to an equivalent first-order model. This permits the *application of any (unaltered) algorithm applicable to first-order HMMs, to HMMs of any other order*.

2.7 Practical issues with higher-order HMMs

In Section 2.3.3 we prove that using the ORED algorithm to first reduce a model to an HMM1 before matching observations to it, is equivalent to matching directly the data to the higher-order HMMs. In Section 2.6 investigation of an algorithm extended to HMM2s naturally leads us to the structures dictated by the ORED algorithm. Taking all this into account, it is important to note the full equivalence between extending the order of the algorithms, or alternatively, reducing the order of the model and then processing it with a standard first-order algorithm. In the following, comments made about directly processing higher-order HMMs equally apply to processing its HMM1 equivalents obtained via the ORED algorithm. With the exception of ease of use, the two approaches are identical for all practical purposes. With this in mind, we will in the rest of this work refer to the combination of the ORED algorithm followed by standard first-order processing as the extended/ORED approach.

As seen in Section 2.2.3.2, using high order HMMs can be vastly more expensive than HMM1s. The processing and memory requirements are serious issues that can easily place such a model outside the available computing capacity. Typically though, the transition structure of a high-order HMM is quite sparse. In Chapter 3 we will investigate a technique to reduce computational requirements by exploiting this sparseness.

Another subtler problem also lurks in the large parameter spaces of high-order HMMs. Due to this large number of free parameters, the surface over which it is optimised during training, might (will?) be quite complex. In Chapter 3 and 4 we show that this can cause the training algorithm to converge often to inferior solutions.

The longer history of states specified in a high-order HMMs also has severe implications for the quantity of training data that should be available to optimise properly the transition probabilities. The number of potential combinations grows exponentially with the order of the model. At the same time, the number of those that can be physically realised in a given training set actually decreases with order. In combination this easily results in serious data deficiencies. Although good training algorithms can maximise the usefulness of a given database, in the end there is no substitute for more data.

2.8 Summary and conclusions

This chapter details and proves the ORder rEDucing (ORED) algorithm, useful for the processing of high order HMMs. In contrast to existing approaches, which extends algorithms to higher orders, this algorithm compresses any higher-order HMM to first-order, thereby allowing the use of all standard HMM algorithms on it. In addition, the first-order representation makes models easier to interpret. Subtle inconsistencies become explicit and easily recognisable. The availability of well-optimised first-order algorithms is another benefit.

Mixed-order HMMs might appear to be somewhat esoteric concepts. Using the ORED algorithm, it is shown here that frequently used ad hoc topologies like Ferguson models (Ferguson, 1980), are in reality equivalent first-order versions of mixed-order HMMs specifically designed to enhance duration modelling. Another extension, designed to enhance contextual modelling, is also proposed. The requirements for training high order HMMs reveal serious, sometimes prohibitive, computational demands. Reducing this, while at the same time maintaining or enhancing the quality of the resulting model, is the subject of the next chapter.

Chapter 3 Training high-order HMMs

3.1 Introduction

As noted in the previous chapter, HMM training algorithms can be extended to higher orders. Alternatively, the use of our new ORED algorithm enables high order models to be transformed to equivalent first-order versions. These models can then be trained by using standard first-order HMM optimisation algorithms. This in effect extends the algorithms used for training first-order HMMs to HMMs of arbitrary order and is fully equivalent to using specially extended algorithms. In the following, both the ORED approach and extending existing algorithms to higher orders (He, 1988; Kriouile *et al.*, 1990; Mari *et al.*, 1997) will collectively be referred to as the *extended/ORED* approach.

Unfortunately, due to the large number of parameters involved in such models, training HMMs via the extended/ORED approach can be a very (even prohibitively) computationally expensive task. We address this by introducing the Fast Incremental Training (FIT) algorithm for fixed-order HMMs in Section 3.2. This algorithm incrementally “grows” a high-order HMM by using lower order HMMs to prune redundant transition probabilities before the next higher order is attempted. As will be seen in following chapters, this can lead to large savings in computational requirements. Section 3.3 investigates issues and techniques for applying FIT to mixed-order models.

The FIT algorithm is not equivalent to the extended/ORED approach. The reason for this is that the optimisation process (EM algorithm, see Moon, 1996) can, due to initial conditions, converge to different local optima, thereby resulting in different solutions. Using carefully controlled simulation experiments, the quality of extended/ORED and FIT trained models are investigated in Section 3.5. The results show that the FIT algorithm can reduce computational requirements substantially, yielding at the same time models that are more accurate, more compact, and generalise better than models trained by using the extended/ORED approach.

3.2 Incremental training of fixed-order HMMs

3.2.1 Basis for Fast Incremental Training

Experience shows that in many practical situations involving non-trivial models, a large percentage of transitions disappears during training. For many problems, considerable training effort is therefore expended on estimating parameters that will eventually become zero. Referring back to the examples of Section 2.4, it will be realised that a single transition probability in the $R-1$ th-order model, is simply being replaced by a set of probabilities in the R th-order model. It will result in significant savings if the training of redundant sets of R th-order probabilities can be avoided by noting which corresponding $R-1$ th-order probabilities are zero.

We now investigate the target transition probabilities of a first-order model (such as M_1 in Figure 2-1) when it is trained with data that have been generated by the corresponding second-order model (such as M_2 in Figure 2-2). Training algorithms, such as Baum-Welch re-estimation, estimate transition probabilities by counting the expected number of transitions on each of the links leaving from a state (Poritz, 1988). Consider a transition in M_2 from state $Q_{\ell-1} = j$ to $Q_\ell = k$ occurring with non-zero probability. If M_1 was used in the place of M_2, this same transition would be observed with a potentially different, but still non-zero probability. This suggests that a first-order model can be trained to determine which sets of second-order transition probabilities are viable. Redundant second-order probabilities can then be avoided during subsequent training.

Consider the first-order probability in model M_2 of state $Q = j, \quad j > 0$ being followed by state $Q = k$ (without any consideration of earlier states). Transitions originating at the initial state are excluded since they are inherently of first-order. Let A_j represent the set of states that have a direct transition leading to state $Q = j$. Using the definition of total probability, conditional probability and the Bayes rule yield:

$$\begin{aligned} \mathrm{P}(Q_\ell = k | Q_{\ell-1} = j, M_2) &= \sum_{i \in A_j} \mathrm{P}(Q_\ell = k, Q_{\ell-2} = i | Q_{\ell-1} = j, M_2), \quad j > 0, \ell > 1 \\ &= \sum_{i \in A_j} \mathrm{P}(Q_\ell = k | Q_{\ell-1} = j, Q_{\ell-2} = i, M_2)\, \mathrm{P}(Q_{\ell-2} = i | Q_{\ell-1} = j, M_2) \\ &= \sum_{i \in A_j} a'_{ijk}\, \mathrm{P}(Q_{\ell-1} = j | Q_{\ell-2} = i, M_2) \frac{\mathrm{P}(Q_{\ell-2} = i | M_2)}{\mathrm{P}(Q_{\ell-1} = j | M_2)}. \end{aligned} \tag{3-1}$$

Due to the $\mathrm{P}(Q_{\ell-2} = i \,|\, M_2) / \mathrm{P}(Q_{\ell-1} = j \,|\, M_2)$ factor, the required probability is a function of time. This suggests that the closest approximation to a second-order HMM using a first-order HMM of similar structure would make use of restricted time-varying transition probabilities. To make this time-dependence explicit, (3-1) can be rewritten as:

$$\begin{aligned} a_{jk}(\ell) &= \sum_{i \in A_j} a'_{ijk} a_{ij}(\ell - 1) \frac{P(Q_{\ell-2} = i \,|\, M_2)}{P(Q_{\ell-1} = j \,|\, M_2)}, \quad j > 0,\ \ell > 1. \\ &= \sum_{i \in A_j} a'_{ijk} w_{ij}(\ell - 1) \quad \text{with} \quad w_{ij}(\ell - 1) = a_{ij}(\ell - 1) \frac{P(Q_{\ell-2} = i \,|\, M_2)}{P(Q_{\ell-1} = j \,|\, M_2)}. \end{aligned} \tag{3-2}$$

If we assume that $Q = i$ is reachable from the initial state (no dead states), it implies that $\exists_{\ell} w_{ij}(\ell - 1) > 0$. From (3-2) it follows that $\exists_{i} a'_{ijk} \neq 0 \;\Rightarrow\; \exists_{\ell} a_{jk}(\ell) \neq 0$ and also $\forall_{i} a'_{ijk} = 0 \;\Rightarrow\; \forall_{\ell} a_{jk}(\ell) = 0$. In other words, if at least one of a set of second-order transition probabilities is non-zero, then the time-varying first-order approximation will be non-zero somewhere in time. Vice-versa if all of the second-order transition probabilities are zero, its first-order approximation will also be.

First-order HMMs do not account for the time varying nature of (3-2), but will approximate it with a single constant, let's say $a_{jk} = \mathrm{g}(a_{jk}(\ell))$ where g() represents the unknown approximation. If $\exists_{\ell} a_{jk}(\ell) \neq 0$, any reasonable approximation $a_{jk} = \mathrm{g}(a_{jk}(\ell))$ will result in $a_{jk} \neq 0$. Therefore, for any reasonable g() it follows that $\exists_{i} a'_{ijk} \neq 0 \;\Rightarrow\; a_{jk} \neq 0$. This leads us to conclude that if any in a set of second-order transition probabilities are greater than zero, any reasonable (constant) first-order approximation of it will also be greater than zero. Non-zero high-order transition probabilities are therefore not lost during the estimation of their first-order approximations.

Also if $\forall_{i} a'_{ijk} = 0$, the only reasonable approximation is $a_{jk} = g(a_{jk}(\ell) \equiv 0) = 0$. This is what makes it possible to detect sets of redundant high-order transition probabilities by noting which lower-order ones are zero. In practical situations where the allowable combinations of categories (such as phonemes) are sparse, avoiding the training of exponentially growing numbers of redundant transition probabilities, can (and does) lead to considerable savings.

Note that training algorithms such as the Baum-Welch algorithm, while guaranteeing an improvement on each training iteration, can converge to any of a number of local optima (Poritz, 1988). This makes it impossible to determine exactly to what constant the first-order training algorithm will converge to. The approximation g() might be very complex and dependent on many factors.

3.2.2 The FIT algorithm for the training of fixed-order HMMs

1) Set up a first-order HMM for the application at hand.

2) Run the training algorithm on the first-order model. Non-viable transitions should disappear.

3) Convert the optimised first-order model to a second-order model. Let A_j be the set of states that directly precedes state $S = j$. Replace the probability a_{jk} that joins the states $S = j$ and $S = k$ by the multiple probabilities a_{ijk} where $i \in A_j$[25]. Initialise these probabilities with the first-order values, $a_{ijk} = a_{jk}$, $i \in A_j$. This conversion will increase the number of transition parameters. If any transitions disappeared while training the first-order model, they will not propagate to the second-order model, thus avoiding the training of this larger number of second-order transition probabilities. To benefit from this a sparse transition matrix representation must be used.

4) Use the ORED algorithm to create a first-order equivalent of this model. Initially this model will match an unknown observation string with the same likelihood as the original (trained) first-order model.

[25] If the Mealy form HMM was used, a separate pdf. will be associated with each transition probability.

5) Now, by repeating the algorithm from step 2, train this model. This will refine the transition probabilities to their required higher-order values. As was previously mentioned, an R^{th}-order model simply extends the memory of an $R-1^{\text{th}}$-order model by one time step. Therefore this process can be repeated to train even higher-order models, bearing in mind that deficiencies might arise from inadequate quantities of training data.

3.3 Mixed-order variants of the FIT algorithm

It will be extremely helpful if we can develop a version of the FIT algorithm that can incrementally train general mixed-order HMMs. Such and algorithm will, at each stage of the FIT algorithm, ascertain which of the transition probabilities should be extended to the next higher order, and which must be retained at its current order. The model can then expand only those sections where dependence on earlier states is of importance. Sections already operating on their proper Markov order can be retained in their present form. Due to the more compact use of transition probabilities, the models will train faster and the probabilities will be more reliable. While this seems to be a very useful idea for reducing computational and other difficulties, it contains some formidable obstacles. The problem lies in how to decide which probabilities to extend.

When a mixed-order system is implemented as a fixed-order system, with the same highest order as the mixed-order system, each of the lower order transition probabilities will be implemented as a set of identical higher-order transition probabilities. If all the higher-order probabilities are hierarchically divided into sets according to the lower order probabilities from which they originate, these sets can be searched to find which have closely related values. Such sets then indicate the possible presence of implicitly lower order transition probabilities in a mixed-order system. As also discussed in Section 2.3.1, if these transitions share the same destinations, the states from which they originate can be merged into a single state. Otherwise, these states should at least be tied to each other to provide greater robustness through parameter tying.

At first glance one might attempt to extend the FIT algorithm to mixed-orders by including such a merging and tying procedure after each training cycle. Unfortunately this approach will fail. To see this, consider a first-order transition probability a_{jk} that is extended during the first cycle to the set $a_{ijk}\ ,\quad i \in A_j$. If not merged, each of these will be expanded in the next cycle to $a_{hijk}\ ,\quad h \in A_i$. The problem now is that, although all of the probabilities in the set $a_{ijk}\ ,\quad i \in A_j$ may be identical, this may not necessarily be so for the sets $a_{hijk}\ ,\quad h \in A_i$. The only way to determine this would be to inspect the sets $a_{hijk}\ ,\quad h \in A_i$. This unfortunately means that we cannot merge lower order transition probabilities on a per cycle basis, but will have to wait until all cycles of the standard FIT algorithm have been completed. At that late stage all the computation of the standard FIT algorithm has already been expended. Merging and tying at this stage, coupled with some retraining can, however, make the final model more compact. Due to the above-mentioned difficulties, this approach has not been explored further.

There is a viable approach to training mixed-order HMMs via the FIT algorithm. If the mixed-order model is not of general structure, but is rather constrained to a very specific mixed-order topology, the prior information on this topology can be used when deciding how transitions should be extended. This then results in special cases for steps 3) and 4) of the FIT algorithm (page 48). As should be clear from the ORED algorithm itself, mixed-order models lead to tied transitions and the requirement to merge certain states. The merging of states add extra complexity. However, steps 3) and 4) of the FIT algorithm can be combined into one simplified step. Basically it involves observing the effect of applying these two steps in a given situation. Then these steps can be replaced by directly modifying the model accordingly. In the following this notion will be applied to the topologies of Sections 2.5.2 and 2.5.3.

3.3.1 Duration modelling with the FIT algorithm

Recall from Section 2.5.2 that a higher-order HMM can specifically focus on duration modelling by modifying the transitions on states with self-loops. In particular, all transition probabilities $a'_{i_1^{D'-1} j i_{D'+1}}$ sharing the same destination state $i_{D'+1}$, with the same number of repetitions of *j* preceding it, are constrained to have an identical value. This can be accomplished by modifying step 3) of the FIT algorithm to read:

3) If state *S*=*j* has a transition a'_{jj} (self-loop) leaving from it, let A_j be the set of states directly preceding it. If it does not have a self-loop, A_j is empty.

   Now replace the probability a_{jk} that joins the states $S = j$ and $S = k$ by the multiple probabilities a_{ijk} where $i \in A_j$. Probabilities a_{ijk} , $i \neq j$ are tied (identical). All are initialised with the corresponding first-order value, that is $a_{ijk} = a_{jk}$, $i \in A_j$. These tied probabilities will lead to tied states, which may be merged by the ORED algorithm subsequently used in step 4).

As suggested above, steps 3) and 4) can be merged into one simplified step that directly modifies the model in question. As illustrated in Figure 3-1 and Figure 3-2, the ORED algorithm in this situation adds an extra state for each existing state with a self-loop. The original self-loop is redirected to this new state. The values and destinations of all the other transition probabilities of this original state are duplicated at the new state.

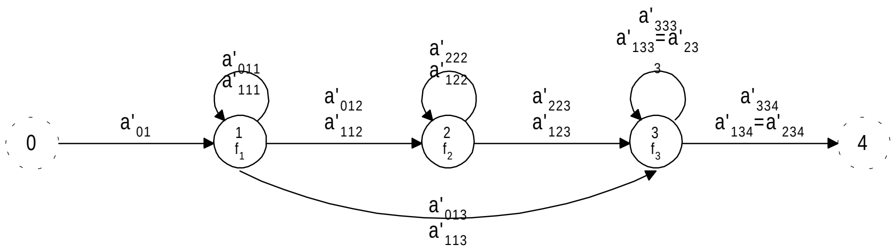

Figure 3-1. Durational left to right HMM2 with one state skip.

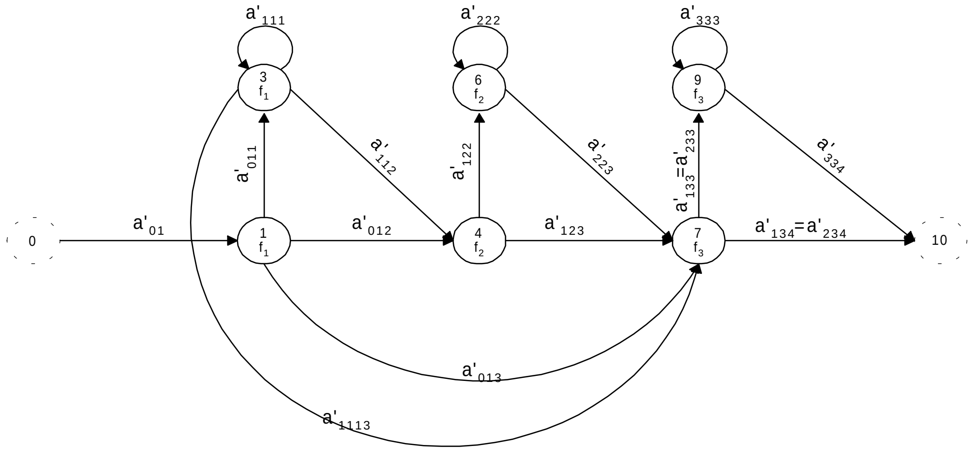

Figure 3-2. First-order version of Figure 3-1.

Repeated application of the duration modelling version of the FIT algorithm will result in stacks of associated states, termed *duration stacks* in Section 2.5.2, p. 48.

3.3.2 Context emphasised mixed-order HMMs

From Section 2.5.3, a higher-order HMM can specifically focus on the combination of differing states by ignoring the number of self-transitions. In particular, all transition probabilities of the form $a'_{i_1 i_2^+ \cdots i_C^+ i_{C+1}}$ are constrained to have the same value via state tying. As indicated in Section 2.5.3 and Appendix B, the infinite order that arises from the arbitrary number of repetitions, causes ORED to grow without bounds. Only after inspecting and extrapolating the resulting structure can the necessary merging be implemented. This is, of course, not amenable to automatic processing. Steps 3) and 4) of the FIT algorithm can, however, be simplified into a combined step that takes all the intermediate processing into account:

From the first-order model, identify and record the self-transitions on states, that is, find each state $Q = j$ that has a transition a_{jj}. Then remove all these self-transitions from the model. Increase the order of this model (that has no self-loops) according to steps 3) and 4) of the standard FIT algorithm, but do not renumber immediately the composite state indexes of the resultant model. For all states $Q = j$ for which self-loop a_{jj} exists in the original model, identify the set of states $S = j$ and/or $S = (i, j)$. All these states get an additional self-loop with value a_{jj}. After this the renumbering of state indexes to single values can proceed, thereby completing the replacement step for steps 3) and 4).

3.3.3 Combinations of context and duration models

A context model (Sections 2.5.3 and 3.3.2) with context order *C* ensures that the effect of *C* distinct prior states is taken into account when making a transition. The duration modelling is, however, restricted to a simple self-loop, suffering from the same problems as first-order HMMs. Expanding this model further by using the duration modelling of Sections 2.5.2 and 3.3.1 (resulting in a *CD* model), re-endows it with duration modelling without detriment to its context-modelling abilities. As discussed earlier, this duration modelling does not take the duration of prior duration stacks into account. This is the price paid for guaranteeing a fixed context and duration order. In Section 4.7 we illustrate the use of such models by applying it to a language recognition task.

Interchanging the order of the techniques to produce a *DC* model, the context modelling will enlarge the dependence on prior stacks. This will also include dependence on states from these prior duration stacks, thereby also re-introducing dependence on their duration. Unfortunately this dependence also once again re-introduces a horizon[26] over which, during training, the full *C* distinct prior states may not be visible any more. By not recognising the duration modelling purpose of the states in the duration model, the *DC* combination could, therefore, once again lose some of its context modelling capabilities during training.

Many other combinations are possible. We will however content ourselves with those discussed above, leaving further investigation to other studies.

3.4 Example of training via the extended/ORED and FIT algorithms

This section illustrates[27] the above by recovering a high-order HMM from simulated data. This is done by using both the extended/ORED algorithm as well the FIT algorithm. The model of Figure 3-3 was used to generate simulated data. This model was determined by removing some of the transition probabilities from the model of Figure 2-12. These removals were chosen to show specific effects at different levels (iterations) of the FIT algorithm.

[26] Similar to the situation sketched in Section 2.5.3.

[27] Section 3.5 presents a formal comparison of the convergence of the FIT and extended/ORED approaches. The current section only serves to enhance understanding of the algorithms.

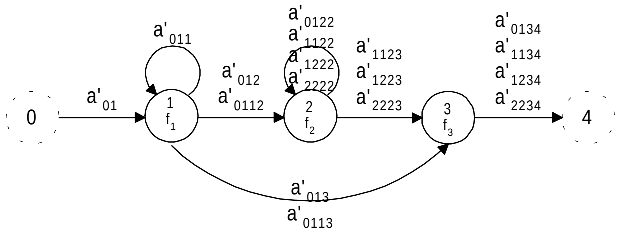

Figure 3-3. Model used for generating simulated data.

Firstly, all transitions involving a transition of state $S = 3$ to itself were removed (a_{33} in Figure 2-1). This is a *first-order effect* that eliminates four probabilities in the second-order model and ultimately eliminates nine probabilities in the third-order model (all a' in Figure 2-2 and Figure 2-12 with two or more 3's in the subscript). Next all transitions from state $S = 1$ to itself, given that the previous state was also $S = 1$, have also been removed (a'_{111} in Figure 2-2). This is a *second-order effect* that ultimately eliminates four probabilities in the third-order model (all a' in Figure 2-12 with three or more 1's in the subscript). Finally, a *third-order* transition probability, namely a'_{0123}, has also been removed. The remaining probabilities were fixed at arbitrarily chosen values. Two-dimensional data are assumed. The three diagonal Gaussian pdf.s of the emitting states were centred on the points (0,0), (0,1) and (1,1). In two sub-experiments the standard deviations used in the pdf.s, were varied. In the first, the standard deviation $\sigma = 0.2$, resulting in little overlap between the pdf.s. The other experiment used a wider standard deviation, namely $\sigma = 0.33$, giving considerable overlap. For each of these sub-experiments 1000 different strings[28] of simulated feature vectors were drawn. The extended/ORED approach was based on the structure of Figure 2-14 as initial configuration, whereas the incremental approach

[28] As specified by the model (see Figure 3-3), the number of vectors in each string will vary randomly.

was initially based on Figure 2-1. The initial transition probabilities for the self-loops were fixed at 0.8 and the remainder was equally divided between the other transitions. Initial values for the pdf.s were determined from vector quantisation (Gray, 1984).

Figure 3-4 shows the model topologies as they evolved from stage to stage in the incremental training process. Both sets of standard deviations yielded identical model structures. Compare this figure to those of Figure 2-12, Figure 2-13 and Figure 2-14. *The transitions that disappear during each stage of the incremental training correspond exactly to the model used to generate the data with.* Due to the presence of local optima, this will not necessarily be so in all cases.

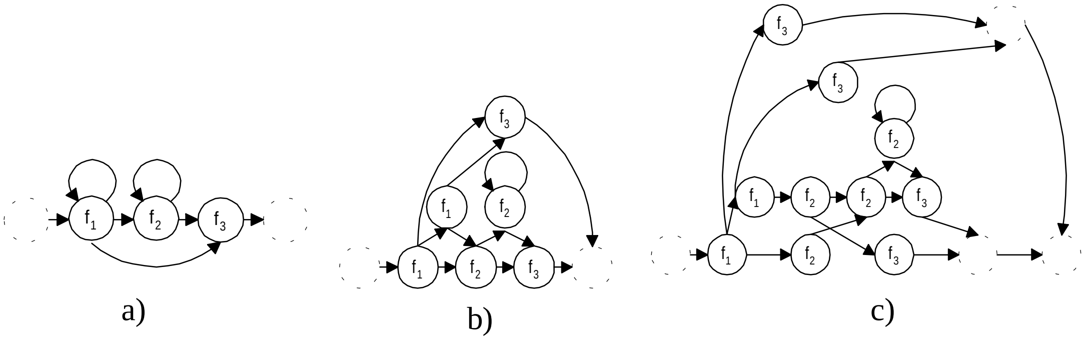

Figure 3-4. Model structures after a) first, b) second and c) third-order stages of the FIT algorithm.

Two different quantities were calculated with which to judge the finer differences between the models. The transitions and mean values in the models are roughly of comparable magnitude. The first quantity used measures the average difference between these trained parameters and their actual underlying values. Using the narrow pdf.s ($\sigma = 0.2$) both the extended/ORED and FIT approaches converged to the correct structure with identical parameters. As can be seen from Table 3-1, the trained parameters closely matched the true underlying values. With the wider pdf.s ($\sigma = 0.33$), the extended/ORED approach converged to a sub-optimal structure, clearly reflected in the rather large deviation from the true values. The incremental approach again yielded the correct structure.

From the final likelihoods of Table 3-2 the same pattern emerges. The example shows that the local optima to which both approaches converge, may or may not coincide. It is interesting to note how, in the incremental approach, the likelihood improves rapidly with increasing order until the proper order is reached. Thereafter only a marginal improvement, which can be attributed to specialisation on the training data, takes place. It was also previously observed that the extended/ORED training approach in the sub-experiment with the wider pdf.s yielded a large deviation between the resulting and actual parameters. In spite of this, the total likelihood achieved is reasonable and compares with the second-order approximation of the incremental approach. This is the typical behaviour of convergence towards a local optimum.

Table 3-1. Mean deviation between trained and actual parameters.

Pdf st dev σ	Extended/ORED training	Incremental (FIT) training
0.20	0.015	0.015
0.33	0.307	0.017

Table 3-2. Total likelihood $\log[\mathrm{f}(\mathbf{X}_1^L \mid M)]$ after training.

St. dev.	Extended/	Incremental (FIT) training order				
σ.	ORED	1	2	3	4	5
0.20	-503.74	-853.28	-649.17	-503.74	-501.46	-501.35
0.33	-5703.66	-5944.01	-5700.87	-5611.79	-5609.52	-5609.25

3.5 Simulations to examine the convergence of extended/ORED and FIT based training

As seen in Section 3.4, this algorithm will not necessarily converge to the same values as the extended/ORED approach. Underlying training algorithms, such as the Baum-Welch algorithm, only guarantees converging to a local optimum (Poritz, 1988). The

FIT algorithm starts out with a different number of differently valued initial parameters from those of the extended/ORED approach. It may, therefore, converge to a different local optimum from that of the extended/ORED approach. A valid question is whether one technique will yield more accurate results than the other. The incremental approach with its much smaller initial set of parameters initially denies the model the full flexibility that the extended/ORED approach can benefit from. This may lead to the ultimate selection of a less beneficial solution. On the other hand, with the smaller sets of well-chosen initial parameters that the incremental approach provides, the space over which it is optimising will probably be less convoluted. It is possible that the focus in training, created in this way, will contribute towards decreasing the effects of local optima in the parameter space. To investigate the quality of the FIT solution, carefully controlled experiments are needed. In particular, it is desirable to use data obtained from a hidden Markov source with known order and transition probabilities. Clearly, this is not possible with speech data, so synthetic data was used. This allows precise control over the Markov order, access to the underlying transition probabilities and control of the difficulty of the problem. For each synthetic experiment, two HMMs of a given order and complexity were generated. The extended/ORED and FIT algorithms were tasked with inferring the parameters of these models, using data that was generated by them. Test data generated by the two models is then classified according to which model generated it. Access to the underlying HMMs allows testing against the actual generating model (not the FIT or extended/ORED estimated model), and therefore allows an estimate of the optimal classification performance achievable. This is useful as it can identify specialisation problems and determine expected performance bounds.

3.5.1 Generating simulated data

A hidden Markov model consists of a component that measures the similarity between feature vectors and states. A transition structure then models how these states may combine to form the whole pattern under consideration.

In the current situation, we will model the local similarities by a set of two-dimensional Gaussian densities with an isotropic variance. The number of these densities will be specified in each experiment. Their centroids are randomly placed on a square two-dimensional plane. Each centroid is constrained to be within a certain specified range of standard deviations from its closest neighbour. Figure 3-5 illustrates a typical set of density centroids obtained in this way. In the following experiments the number of densities, as well as the separation between them, will be varied. The transition structure was manipulated by starting with a randomly initialised ergodic model structure of a specified Markov order. The number of states in this ergodic structure was determined from the specified number of densities in the model. Two extra states were added to supply the model with an initial and terminating state. Transition probabilities were chosen randomly, with biases favouring self-loops and reducing the probability of termination[29]. This was done to ensure that very short feature vector sequences would not be dominant (the actual length will be random). These transitions are then pruned to obtain a controllably

[29] Where L is the number of links leaving from a state, random variables X_i, $1 \leq i \leq L$ were drawn from an uniform distribution on the range [0,1]. The link leading to the termination state was reduced to $X_i/10$, while the one corresponding to a self-transition was enhanced to $4X_i(L-1)$. After pruning, the remaining probabilities were normalised to a unity sum.

sparse transition structure. If no path exists between the initial and termination state, the model is discarded and the whole process is repeated.

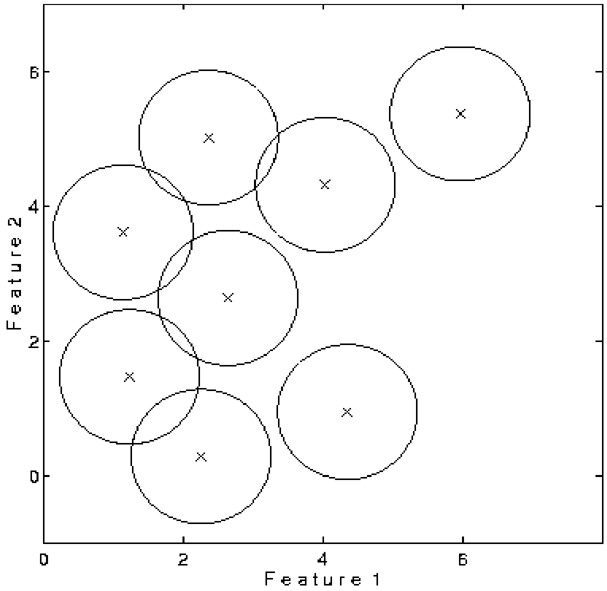

Figure 3-5. An example of a set of Gaussian pdf. centroids from HMMs used to simulate data. The circles are the one standard deviation contour on the pdf.s.

The key parameters regarding the transition structure of the model are the order of the model, the number of states, the final number of transition probabilities and the sparseness of the transition structure. The last two are related but are also random and somewhat difficult to control since a single deactivated transition could potentially disable large sections of the model. The ability to transform any higher-order HMM to an equivalent first-order model by means of the ORED algorithm meant that the data could be generated using a first-order HMM.

3.5.2 Comparing FIT vs. extended/ORED trained models

3.5.2.1 Methodology

In this section we compare the classification accuracy and the number of transition probabilities present in models trained via the extended/ORED and FIT algorithms. In the classification experiment, the object is to determine from which of two (synthetic) HMMs the sequence is most likely to have come. This is a two-class classification problem. We compare extended/ORED and FIT trained results with a HMM1 as well as with the actual underlying HMMs used to generate the synthetic data. From this, conclusions can be drawn about the relative merits of the various techniques.

A single comparison experiment consists of generating two comparable HMMs as described in Section 3.5.1. The number of transition probabilities in each is recorded. These two models are then used to each generate 1000 (training) feature vector sequences. Each set of 1000 sequences is vector quantised into the number of Gaussian clusters pertinent to the model. The initial ergodic models required by the extended/ORED, FIT and HMM1 training algorithms are configured from this. The sets of training data are then used to train the models using the respective algorithms. The numbers of transition probabilities present in the trained models are noted. The 1000 training sequences are then classified (as to which model generated the sequence) using the 4 different sets of models, i.e. firstly with the true underlying HMMs, then with those trained via the extended/ORED algorithm, then with those resulting from the FIT algorithm and lastly with the HMM1s. From all of this the various recognition accuracies are noted. The known underlying models are then used each to generate another 1000 independent (testing) feature vector sequences. These testing sets are used to verify the classification results on data not used during the training phases.

The above procedure is repeated twenty times, resulting in 20000 trials per experiment on a given HMM configuration (different orders and pdf. centroid placements). A total of 1.28 million trials were undertaken for these experiments. The first three experiments generate data using eight state (ten counting the initial and termination states) HMMs of order two, three and four. The Gaussian centroids were spaced to be between 1.5 and 2 standard deviations from its closest neighbour. To put this into perspective, note that at roughly 2 standard deviations spacing between the centroids, the envelope of the densities merges into a single peak. The last experiment (experiment 4) uses a second-order HMM with thirty-two states and more severe pruning. This results in a second-order model with the transition sparseness comparable to that of the previously used fourth-order model. We also placed the Gaussian pdf.s more densely by restricting the centroids of closest neighbours to be between 1 and 2 standard deviations from each other. Table 3-3 summarises the four conditions evaluated in this experiment.

Table 3-3. The four different generating HMM experiments evaluated. "σ Spacing" is the distance between the Gaussian centroids.

Experiment	1	2	3	4
Order	2	3	4	2
States	8	8	8	32
σ Spacing	1.5 -3	1.5 -3	1.5 -3	1-2
Max links	656	5264	42128	34880
Ave links	134	398	1232	974
Sparseness	20.4%	7.6%	2.9%	2.8%

3.5.2.2 Model sizes

We first consider the compactness, i.e. the sparseness of models derived via the extended/ORED and FIT algorithms as this indicates the computational complexity of the training procedure. In Figure 3-6 the excess number of non-zero transition probabilities (expressed as a percentage relative to the actual number of non-zero transition probabilities), resulting from these algorithms when compared to the true underlying model, is shown. The sizes of the 8 state models, which have fairly well separated Gaussian centroids, are well approximated by the FIT algorithm, irrespective of the sparseness factor. This, however, *does not hold for the extended/ORED approach*, which rapidly escalates in size.

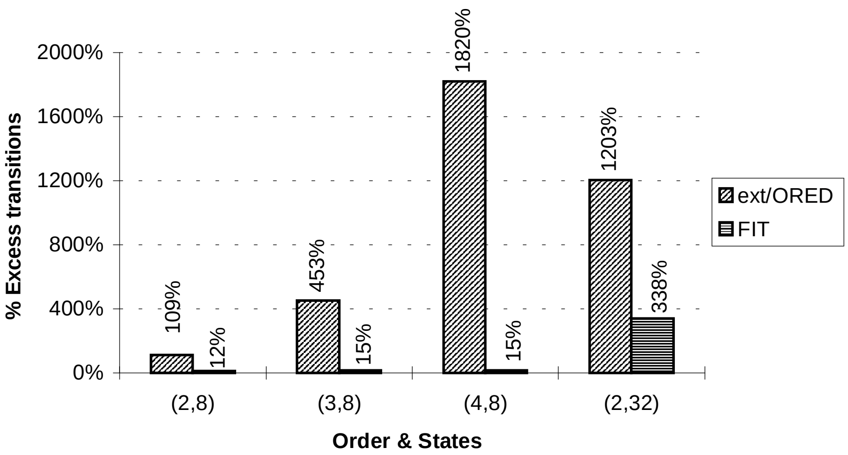

Figure 3-6. Excess transitions in extended/ORED and FIT trained HMMs relative to those of the true underlying model. The HMMs are identified by tuples such as (3,8), indicating a third-order HMM with eight states.

Remember that as far as using the extended/ORED approach is concerned, we are merely training a very large and complex first-order HMM, which happens to be mathematically equivalent to the R^{th}-order HMM. The behaviour of the extended/ORED approach is therefore not surprising as it is common for pattern recognition systems to converge to bad local optima when confronted with a number of training parameters that far exceeds those of the actual underlying model. This is termed "specialisation" (Raudys & Jain, 1991). The FIT approach, which does not immediately consider all the available parameters while training, but "grows" them as needed, is able to avoid this problem for experiments 1, 2, and 3. In the 4th experiment, the spacing between Gaussian centroids is less than 2 standard deviations, so none of the individual density functions are recognisable in the envelope density (i.e. they do not form distinct peaks) which makes it hard to estimate the original model from this data. While the model determined via the FIT algorithm is four times the size of the actual model, it is substantially (72%) smaller than the result obtained by using the extended/ORED algorithm.

In the following sections we consider the classification accuracy of the various models. Results on training data are considered separately from those achieved on independent testing data, since the contrasts between them demonstrates the generalisation ability of the respective techniques.

3.5.2.3 Accuracy on training data sets

Figure 3-7 summarises the classification accuracies achieved on the training data. This graph shows the relative increase in number of errors compared to that obtained from the known (ideal) underlying models, i.e. if the underlying model has 100 errors and extended/ORED model 150, this is indicated as 50% excess. When interpreting these numbers one should remember that it is a relative measure, the absolute error rates for all the simulation experiments reported here are all well below 10%.

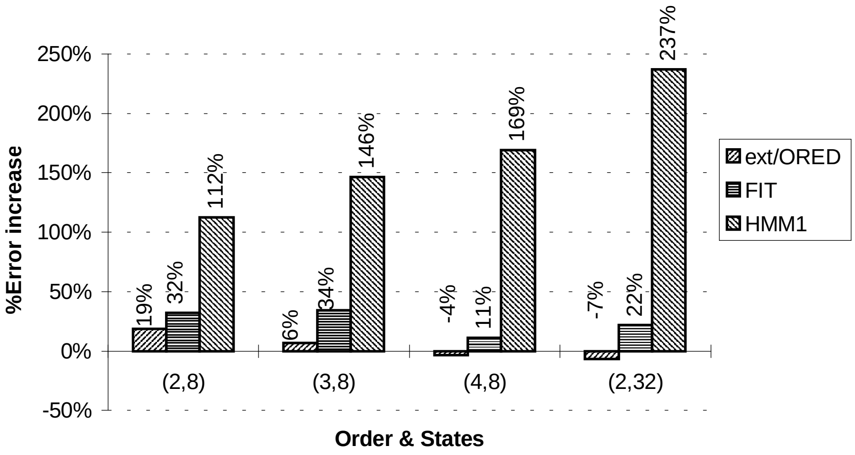

Figure 3-7. Relative error increase when classifying *training data* with extended/ORED, FIT and first-order trained HMMs, compared to that of the true underlying model.

Note that the first-order HMM produces (unsurprisingly) poor performance on data originating from higher-order models, thus illustrating the importance of using higher-order HMMs where appropriate. An important weakness of the extended/ORED algorithm, namely *training data specialisation,* is clearly highlighted in these tests. As the underlying HMM transition matrices become sparser, extended/ORED's performance on training data improves until, for the (2,32) case, its error rate is 7% lower than that of the true underlying model. This specialisation (over training) results from excessive free parameters.

To determine whether the differences in accuracy of the various training algorithms are statistically significant, we use the McNemar test (Gillick & Cox, 1989). This sensitive test is appropriate for comparing classifiers evaluated on common test data. Of the 24 pair wise combinations[30] present, only 3 pairs were found that were not statistically significantly different with 95% confidence (see Appendix C.2.2). In spite of the large number of repetitions, the accuracy of the extended/ORED models for the (3,8), (4,8) and (2,32) HMMs could not be differentiated from the underlying models. It is interesting to note that the simplest and least sparse HMM, namely (2,8), is absent from this list. We believe that the extended/ORED algorithm could not specialise to the same extent on the training data in this case, as the number of redundant parameters (transition probabilities) is too small.

3.5.2.4 Accuracy on testing data sets

Figure 3-8 shows the results obtained by classifying independent testing data generated by the same underlying models. On the least sparse (2,8) model, the

[30] The generating, extended/ORED trained, FIT trained and HMM1 models can be arranged in six pair-wise combinations. This is done for four generating conditions.

extended/ORED performance compares well with the FIT algorithm. This model has relatively few free parameters and will not specialise easily. FIT performs substantially better than extended/ORED for the remaining models as the underlying transition probability matrices are more sparse. This occurs because of the extended/ORED algorithm's inability to prune unneeded transitions as efficiently as FIT is able to, resulting in models with very high degrees of freedom, which causes specialisation and hence poor test data performance. These results clearly indicate that the extended/ORED algorithm has a serious propensity to specialise on the training data, a deficiency that does not affect the FIT algorithm to the same extent. Of significant importance is that for the more difficult (4,8) and (2,32) HMMs, the results from the higher-order extended/ORED models are very similar to those obtained by using the first-order HMM! Clearly the extended/ORED approach does not exhibit a clear performance advantage over first-order approaches, considering the computational simplicity of the latter.

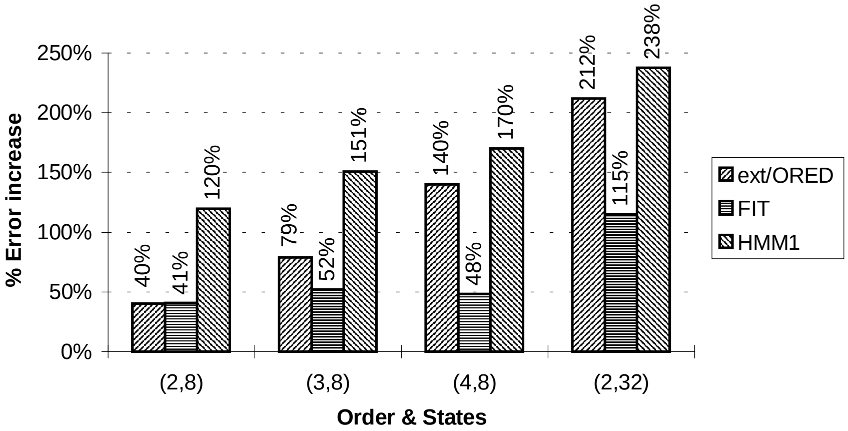

Figure 3-8. Relative error increase when classifying *independent testing data* with extended/ORED, FIT and first-order trained HMMs, compared to that of the true underlying model.

Again, we employ the McNemar test with a 95% confidence level to establish which training algorithms yield statistically significant different test data performances. We find that all but the extended/ORED vs. FIT results on the data from the (2,8) model, and the extended/ORED vs. HMM1 results on data from the (2,32) model, differ significantly (see Appendix C.2.3). Both exceptions correspond to observations we have previously made in this section.

If, due to the effect of local optima and specialisation, expensive higher-order HMMs yield results comparable to first-order HMMs, then it is hardly surprising that this technology has found so little penetration in current pattern recognition systems. By not trying to optimise the whole parameter space simultaneously, the FIT algorithm provides an alternative that produces far better results with substantially reduced computational requirements during both the training and testing phases.

3.6 Summary and conclusions

The ORED algorithm followed by standard first-order training, and by implication any direct higher-order extensions of the re-estimation equations normally used for training first-order HMMs, has been compared to the new Fast Incremental Training algorithm that we have developed. By means of simulations we found that the former is inclined to specialise on the training data. This results in bulky models that do not generalise well on data unseen during training. We find that in some cases, the extended/ORED approach leads to results comparable to those of a simple and inexpensive first-order HMM. The FIT algorithm, like most optimising pattern recognition algorithms, can also suffer from specialisation. In the experiments conducted here, it does, however, seem less prone to this weakness. Lastly, we want to point out that, while the FIT algorithm is computationally efficient, the problem of inadequate amounts of training data still remains and can be addressed using interpolation or related techniques (Bahl *et al.* 1983).

Chapter 4 High-order HMMs applied to Automatic Language Recognition

4.1 Introduction

In previous chapters we developed theory for high-order hidden Markov modelling. In this chapter we demonstrate its practical applicability. Phonotactic modelling in automatic language recognition (ALR) systems is a large and complex model of the interdependence of the phonemes of each language (Yan & Barnard, 1995). Because of the size and complexity of these models, we have chosen this as a suitable field to demonstrate our techniques with. It must be stressed at the outset that the intention is not to develop a fully-fledged ALR system; this is a formidable exercise in its own right. Instead *the intention is merely to demonstrate that the concepts* discussed in previous chapters are indeed *applicable to practical systems*. At the same time, behaviours found in simulation experiments can be verified on a non-trivial real-life problem. While the intention of this chapter regarding ALR appears modest, the application of our methodology to this problem and to the development of more advanced systems, is very promising.

Language identification presents a formidable problem for implementation in automatic systems, mainly because of the difficulty in quantifying parameters that provide discrimination. The problem is exacerbated further by the various sources of variability in typical scenarios. Such factors include

- speaker dependence,
- regional accents,

- text dependence,
- channel dependence, and
- noise.

We base our ALR system on modelling different languages with higher-order variants of ergodic HMMs. In this way, a model of the sounds and sound clusters occurring in the particular languages can be acquired. We *will show that this can be done while avoiding the formidable obstacle of phonetically transcribing the large speech database* that this problem demands.

In Section 4.2 we supply some background information on ALR. We position our ALR approach relative to some systems reported in literature. We then conduct two sets of ALR experiments. Sections 4.3, 4.4 and 4.5 discuss database, signal processing and modelling concepts common to both sets. In Section 4.6 we extend the simulation results of Section 3.5 by using both FIT and extended/ORED training to model the English and Hindi languages with second- and third-order ergodic HMMs. The results correlate with those we found in the simulations, namely that the FIT models are more compact and provide comparable or better accuracy. In the second set of experiments (Section 4.7), we investigate the use of combining explicit context and duration modelling (see Section 3.3.3) in ALR systems. Although showing promising tendencies, the amount of testing data was only sufficient to reveal significant differences between the baseline HMM1 and the various higher-order HMMs. In spite of this, useful indications of the capability of these techniques, and how they differ from standard fixed-order modelling, emerge. Section 4.8 indicates that, compared to other systems not requiring transcriptions, our system is indeed very successful. Even when compared to a very popular group of ALR systems that do require transcriptions, we are quite competitive. In Section 4.9 we include a list of obvious enhancements that a future ALR system, based on our HMM methodology, should include.

4.2 Background on language recognition

We do not intend to provide full coverage of the available literature on ALR in this section; the interested reader can find excellent material on this in Zissman (1996) and Muthusamy, Barnard and Cole (1994). Instead we will provide a broad outline of and comment on the main approaches and show how this relates to our own work.

Useful features for language recognition appear at different levels (Bond & Fokes, 1991; House & Neuberg, 1977). Acoustic-phonetics describes the nature and inventory of phonemes. Prosodics cover duration and intonation. Phonotactics describe the rules according to which phonemes may combine in a given language. The lexicon of a given language, and the syntax, according to which its words may combine, are also language specific features. Other distinctions are also possible (Kadambe & Hieronymus, 1995). While acoustic-phonetics and prosody are useful for ALR (Hazen & Zue, 1997; Marcharet & Savic, 1997; Lund, Ma & Gish, 1996; De Bruin & Du Preez, 1993; Sugiyama, 1991; Foil 1986; Goodman, Martin & Wolford, 1989), they do not directly contribute to the discussion at hand and will therefore not receive any special attention here. The focus of this chapter will rather be on describing the phonotactic structure present in a language through the use of higher-order HMMs. Some attention will also be given to how our work relates to large vocabulary continuous speech recognition (LVCSR) based language recognition systems.

Partly due to a lack of good transcribed databases, early ALR attempts focussed on techniques that do not require such resources. The use of ergodic HMMs to model *untranscribed* sequences of speech symbols was first suggested by House and Neuberg (1977). The basic idea is to use the transition probabilities of the HMM to model phonotactic constraints on pairs of phonemes or broad phonetic categories. Using 5-state ergodic HMMs Savic, Acosta, and Gupta (1991) managed faultless identification of four languages. The database used for the test was, however, rather small. Zissman (1993) turned the tide against the use of ergodic HMMs when he concluded that it showed no advantage over simple Gaussian mixtures, which, of course, do not model phonotactics, but rather acoustic-phonetics. In our own work, we found ergodic HMMs quite useful on high quality speech (Du Preez, 1991b). The accuracy of this direct approach was, however, reduced when used on telephone speech (Du Preez, 1992). Inspection revealed that the acoustic component as reflected in the HMM densities (pdf.s) were contributing to this loss of accuracy, probably due to the interaction with the telephone channel’s transfer function. A solution to this was to use one common set of densities for all the language HMMs involved (Du Preez, 1992, 1993). This prevents systematic biases in the acoustic component of the model, and focuses the system on the phonotactic information contained in the transition probabilities. With such a system, we managed accuracy in excess of 90% for a three language ALR system. Subsequent work by Mendoza *et al.* (1996) confirms the usefulness of desensitising the system to the acoustic component. In their system, after matching the speech to a model, they completely remove the acoustical component from the resultant score. This contrasts with the previous conclusion of Zissman (1993) who found no benefit in the structural component of the HMM.

The next generation of ALR systems employed as first stage a phoneme recogniser that transforms the speech into a sequence of (non-ideal) phoneme labels. Several of these phoneme recognisers may be used in parallel and they need not correspond to the languages being modelled (Zissman & Singer, 1994; Yan & Barnard, 1995). An N-gram or similar model is used to capture the phonotactic structure in these sequences. This information is language specific and therefore allows classification. Although arbitrary lengths of phoneme sequences can be modelled, insufficient quantities of training data make bi-grams a popular choice (Zissman 1995a). Another additional factor that we would like to suggest is that, due to the hard errors introduced by the phoneme recognition system, the variability in longer phoneme sequences may just be too large for reliable modelling. Zissman (1995a) averages an accuracy of 97.9% classifying between pairs of languages (NIST’94 45s. set). To achieve this they also include gender and duration modelling. Yan and Barnard (1995, 1996) approximate tri-grams in an efficient manner by combining forward and backward bi-grams. Also incorporating (amongst others) duration information, they achieve a respectable 91.96% accuracy on a six-language test (NIST’94 45s. set). In a similar vein Navrátil and Zühlke (1997) enlarge the phonotactic context by combining two bi-gram models, in this case the standard backward bi-gram and the bi-gram obtained by considering the current and once-removed previous phoneme (termed a skip-gram). Kadambe and Hieronymus (1995) uses a Continuous Variable Duration HMM, trained from transcribed data, to model phonemes. They argue that Consonant Vowel Consonant (CVC) clusters are effective for describing languages. They therefore imbed these phoneme models into a second-order ergodic HMM, its transition probabilities being estimated from text (tri-gram analysis). Classifying language pairs (NIST 94 45s. set) they obtain results varying from 86% up to 100% (average 94.8%), depending on the specific language pair.

From our perspective, the advantage of this phoneme recognition followed by N-gram approach lies in its ability to make use of a wide phoneme context. Due to data scarcity though, this is often not exploited fully. Another advantage is that linguistically well-defined concepts are brought to bear upon ALR. On the down side, later statistical modelling can never fully compensate for hard decisions about phoneme classes introduced in the earlier stages of the system. Another very important disadvantage is the requirement for transcribed databases. Establishing transcribed speech databases and phoneme recognisers for new languages is very costly. Using existing phoneme recognisers from other languages can at most be a useful surrogate (Lund, Ma & Gish, 1996), which is likely to decrease performance with dissimilar languages (such as the indigenous languages of Africa; South Africa has 11 official languages). Such cross-usage also detrimentally prevents inclusion of language dependent factors during the phoneme recognition phase.

The next generation of ALR systems enhanced contextual modelling while avoiding the dangers of early hard decisions by incorporating lexical and linguistic modelling (Kadambe & Hieronymus, 1995; Mendoza, Gillick, Ito, Lowe & Newman, 1996; Schultz, Rogina & Waibel, 1996; Hieronymus & Kadambe, 1997). Modelling languages through Large Vocabulary Continuous Speech Recognition (LVCSR) systems promises very accurate systems. On the NIST'95 45s set, Mendoza *et al.* report accuracies exceeding 99% on a language pairs task. It is interesting to reflect on some aspects of such a system from a higher-order HMM perspective. Although the state-level structure, internal to phoneme and word models, typically makes use of first-order transitions, the larger word-level structure is based on external knowledge. Typically a left-to-right no state-skip structure is used, which can be shown to be a special case of our context-emphasised models introduced in Section 2.5.3. Word structure implicitly creates a larger context within which each of the first-order transition probabilities operates. In a certain sense then, a word model can be thought of as a first-order realisation of a higher-order specification. A language model, N-gram or otherwise based, extends this concept even further. Therefore, from our perspective, a LVCSR based ALR system resembles to a certain extent a large mixed order HMM. External knowledge plays a large role in demarcating its structure while training refines its parameters within the given boundaries. The main disadvantage of this approach lies in the extreme expense of creating a LVCSR system for a new language, dwarfing even that of a phoneme-recognition system (which often is one sub-component of it).

The expense of creating transcribed databases for new languages has led some researchers to (re)consider once again techniques that do not rely on this. Lund *et al.* introduces a system, focussing on acoustic-phonetic aspects, that do not require any transcriptions during training. By using time-varying trajectory models of speech, and experimenting with the use of various databases for training, they achieve an average accuracy of between 85.2% and 93% on language pair tasks (NIST’95 45s. set), depending on the training databases incorporated.

In summary, the strength of current successful ALR systems lies in their ability to model long contexts of speech. Possible weaknesses lie in the early use of hard decisions, and the expense incurred by the need for transcriptions. In the following section we will address the weaknesses, while attempting to maintain the strengths as far as possible.

4.3 Database and signal processing

Our prior work with first-order HMMs (Du Preez, 1993) indicated that about 1 hour of speech was adequate for modelling a language. Preliminary experiments indicated that the higher-order Markov models would need substantially more data. Since they need not be transcribed, this does not necessarily pose a problem. From the OGI-TS database[31], we had roughly 100 minutes of free-format English and Hindi speech available as training data. These were the two largest collections[32] of data available to us and were therefore used in the experiments. Silence sections in the recordings were removed automatically by using an energy criterion. The power in the remainder was normalised to compensate for recording volume. After pre-emphasis, tenth-order LPC-cepstra (Markel & Gray, 1976; Davis & Mermelstein, 1980) and delta-cepstra (Lee, 1989 p. 65) were calculated from 32ms time frames spaced at 16ms. intervals. Cepstral mean subtraction (Atal, 1974; Furui, 1981) was used to compensate for channel variation. For testing data we processed a set of independent 5s segments, as well as the 45s NIST’95 LID set, in a similar manner. Each language model had its own transition probability description, while one central set of pdf.s was referenced collectively by all the language models. This shared pdf. arrangement was first reported by us (Du Preez, 1993).

[31] The Oregon Graduate Institute kindly made the OGI-TS database available to us.

[32] The other recordings in this corpus have about 60 minutes per language available.

4.4 Basic system structure and initialisation

In applications like word recognition, a left-to-right HMM is normally used (Rabiner, Juang, Levinson & Sondhi, 1985). In an application like language recognition, the model might start off in any one of a number of states and move to and fro between states as dictated by the characteristics of the particular language. The strict left-to-right form normally used is too restrictive for this situation and a more general structure is called for. In previous work (see Section 4.2) we successfully utilised the first order ergodic HMM (illustrated in Figure 4-1), for modelling a language. The ALR system consists of a bank of such models, each optimised to a specific language. Unknown speech is matched to each of the models and classified according to which one fits best. As also discussed in Section 4.2, sharing only one set of pdf.s between all the language HMMs enhances robust recognition.

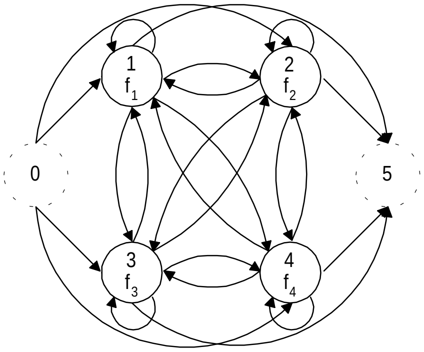

Figure 4-1. A four-state ergodic HMM. (States 0 and 5 are the initial and final null states.) We based our experiments on sixteen-state ergodic HMMs.

In the following sections, various extensions of this basic ergodic structure (we use a sixteen-state version) will be investigated. Because of the presence of local optima, it is important to initialise an HMM properly before training takes place. Models derived via the FIT algorithm are automatically initialised with the parameters of the models that they extend and therefore ultimately only require the initialisation of the basic sixteen-state first-order HMM. Models directly trained via the ORED reduction were configured using an order expanding procedure similar to that occurring within the FIT algorithm itself (See step 3) on page 64). In this process the initial parameters of the sixteen-state first-order ergodic HMM are transferred to the higher-order HMM. It is, therefore, only necessary for us to consider the initialisation of the basic first-order HMM.

Before configuring any models, the training data for both languages were pooled and clustered into 16 diagonal Gaussian clusters. This was done by first using a non-uniform binary split K-means algorithm (Gray, 1984) to find the approximate cluster centres. These centres were then used to initialise Gaussian clusters, which were re-clustered using a dynamic clustering procedure (Devijver & Kittler, 1982, pp 407 – 413). This set of pdf.s was used to initialise the first-order ergodic HMM. All self-looped transitions in this model were initialised to a value of 0.8. At each state the remaining part (not devoted to the self-loops) was spread equally over the remaining transitions. This base model is directly (via order expansion) or indirectly (via the FIT algorithm) used to initialise all other models.

4.5 Training procedure

Because it is much faster and in practice gives comparable performance to the Baum Welch algorithm, all training was done using the Viterbi re-estimation algorithm (Levinson, 1985; Picone, 1990). Higher-order models were always reduced to equivalent first order form by using the ORED algorithm, enabling the use of the first-order re-estimation algorithm in all cases. The Viterbi algorithm includes a matrix that records the optimal path between states as a function of time. In a first-order HMM system, the product of the number of states with the number of time frames in the speech segment, dictates the size of this matrix. To reduce the demands on memory, the training sequences were subdivided into 5s sequences that formed the basic patterns presented to the system.

The pdf. component, common to all the language models, was implemented as a shared structure. Only after all the training exemplars had been presented to their respective models, did updating of the common set of pdf.s as well as the individual sets of transition probabilities, take place[33]. To avoid unnecessary computation, redundant transition probabilities, pdf.s and states were pruned after each training cycle. These cycles were repeated until the increase in total likelihood decreased lower than a specified threshold. Extended/ORED-based training directly used this procedure, while the other models used it to train incrementally from earlier models (according to the FIT procedure).

Transitions originating at the initial state or leading to the final state occur only once in a 5s segment and thus were very sparse. We wanted to allow the system to start off in any of the original sixteen states, and also to be able to terminate at any of them. This was done after training by resetting the transition probabilities leading from the initial state to all be equal (value 1/16). Transition probabilities leading to the termination state were all set to an unity value[34].

[33] This means that models cannot be optimised separately, but are trained simultaneously.

[34] Strictly speaking this violates the probabilistic base since probabilities at those states will exceed a total value of one. Since this arrangement merely passes the likelihood values at those states directly through to the final state, we accepted this inconsistency.

4.6 Comparison of FIT- vs extended/ORED-trained HMMs

4.6.1 Topology and training method

To verify the results obtained with the simulations of Section 3.5.2, we extended the basic system of Section 4.4 to second- and third-orders. Direct training via the ORED reduction results in models X2 and X3 (X for eXtended/ORED; generically we will refer to the group of models as Xm). Training via the FIT algorithm resulted in models F2 and F3 (F for Fixed order FIT; generically we will refer to the group as Fm). For reference we also include the results from the base-line first-order system (Model 1). Training proceeded as discussed in Section 4.5.

4.6.2 Computational requirements

To explore the computational cost of training via the FIT or extended/ORED approaches, parameters determining this were recorded during the training process. The memory requirements, shown in Figure 4-2, are based on the space required by both the Viterbi-paths matrix, as well as a sparse representation of the transition probabilities themselves. During training the X2 model requires close to 1.5 times the space of the F2 model. With the third-order model this escalates close to 8 times larger[35].

For the CPU calculation we ignored the contribution of evaluating the pdf.s and only considered the processing requirements of the transition probability component. For

[35] Other informal experiments indicate that this gets even worse for models with more states or higher orders.

models with many transitions and relatively few pdf.s (as is often the case), this dominates the CPU usage. The F2 and X2 models required approximately the same amount of transition operations[36]. The F3 model trained close to fifteen times faster than the X3 version. Figure 4-4 reports the number of transition probabilities in the trained models, thereby indicating its compactness. From this it is clear that the FIT algorithm results in much more compact models, getting more so as the order increases.

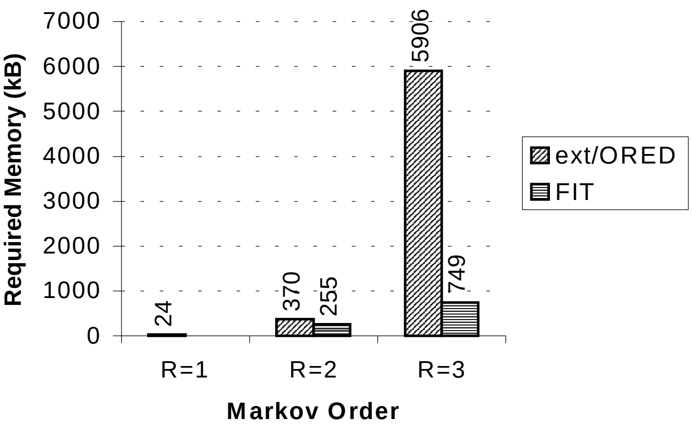

Figure 4-2. Maximum memory requirements during training of the HMMs. Based on the size needed for the Viterbi path matrix for a 5s chunk of speech, as well as the memory required to store the transition probabilities (using a sparse representation).

Table 4-1 summarises the computational requirement as well as the final number of transition probabilities in the resultant FIT trained models relative to the extended/ORED approach. Clearly the FIT approach results in more compact models.

[36] We consider this result to be somewhat of a outlier since the FIT algorithm iterated unusually long in this case. If the same number of training cycles were used, the FIT algorithm would have used less than 60% of the number of calculations required for the direct approach. In our experience the FIT algorithm usually converges in fewer cycles than the direct approach. Never the less, we report the figure as we found it.

The large computational differences found in especially the F3 vs. X3 models are sufficient to justify the use of the FIT algorithm.

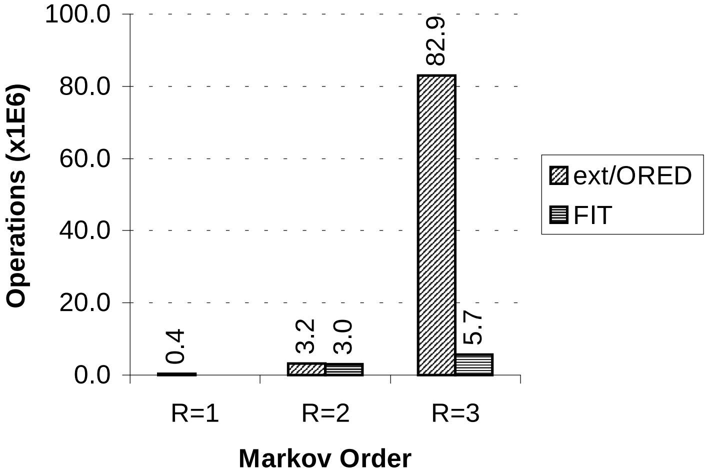

Figure 4-3. Number of transition probability operations required during the training of the HMMs. (Averaged over the English and Hindi models.)

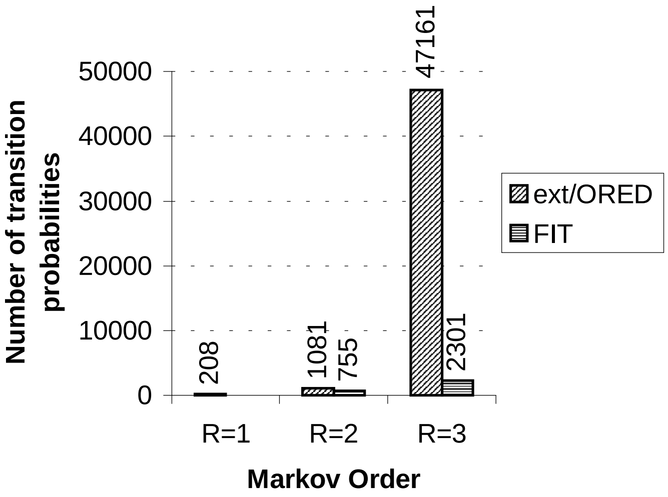

Figure 4-4. Number of transition probabilities in the trained HMMs. (Averaged over the English and Hindi models.)

Table 4-1. Comparison of computational requirements and final model sizes for 16-state ergodic HMMs trained via extended/ORED and FIT algorithms.

	Ratio FIT/ORED		
Order	MEM	CPU	Size
2	69%	94%	70%
3	13%	7%	5%

4.6.3 Classification accuracy

Having discussed some key parameters for the training of these high-order HMMs, we now turn our attention to the accuracy of the resultant models.

Table 4-2. Accuracy measured on training set for 16-state ergodic HMMs trained via extended/ORED and FIT algorithms.

	5s (2169 trials)		45s (339 trials)	
Order	ext/ORED	FIT	ext/ORED	FIT
1	83.4%	-	92.6%	-
2	86.8%	85.2%	95.3%	95.3%
3	92.7%	89.6%	98.8%	99.1%

Table 4-3. Accuracy measured on testing set for 16-state ergodic HMMs trained via extended/ORED and FIT algorithms.

	5s (247 trials)		45s (39 trials; NIST’95)	
Order	ext/ORED	FIT	ext/ORED	FIT
1	69.2%	-	82.1%	-
2	75.7%	76.1%	87.2%	89.7%
3	79.8%	79.8%	89.7%	97.4%

Table 4-2 and Table 4-3 summarise the accuracies of models trained via the FIT algorithm and those trained via the extended/ORED approach. On the training data the accuracy of the FIT and extended/ORED trained models are comparable. On independent testing data, however, in all cases the FIT trained models perform similarly or better than the extended/ORED trained models. Furthermore, all the FIT trained models result in smaller differences between training and testing set accuracies than those achieved by the extended/ORED trained models. This, combined with the larger models that resulted from extended/ORED training, indicates greater specialisation in such models. With the 5s classification trials, a McNemar test with a

90% significance level (see Appendix C.3.3) shows all the higher-order models to be more accurate than the baseline HMM1. Due to the rather small test set, the experiment based on the NIST'95 45s set (39 trials) could only show the 3rd-order FIT model (F3) to be more accurate than the 1st-order HMM on a 90% significance level. No significant differences could be detected between the extended/ORED and FIT trained models. This is accomplished at a much-reduced computational cost. In general the results confirm those found in the simulations of the previous chapter.

4.7 Independent variation of context and duration orders

In Sections 2.5.2 and 2.5.3 we introduced topologies that enhance the capability of high-order HMMs to focus on either duration modelling or modelling the context of *different* previous states (termed context modelling for brevity.). From the first we defined the *duration order D* as the maximum number of repetitions of the current state that is taken into account when making a transition to a next state. The *context order C* was defined as the maximum number of distinct previous states (ignoring repetitions) that are taken into account when making a transition to a next state. In Sections 3.3.1 and 3.3.2 we have shown that the FIT algorithm can be adapted to this type of modelling. Section 3.3.3 discusses possibilities of using them in combination to achieve independent control over *C* and *D*. To gain better insight into these techniques and to investigate specifically the role that *C* and *D* plays, they are now applied to ALR in the following sections.

4.7.1 Topologies used

In this section we experiment with some extensions to higher-order HMMs which shows promise for ALR. In Figure 4-5 we categorise them by noting their context and duration orders (*C* and *D*), as well as their FIT training-dependencies.

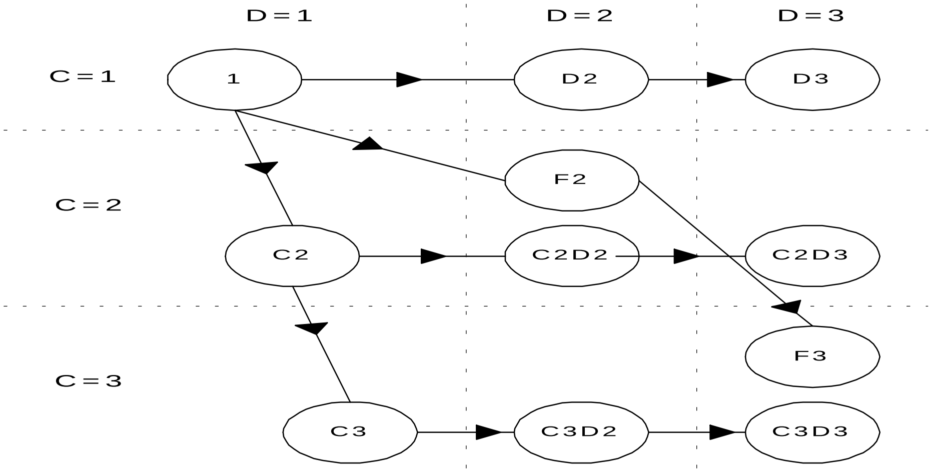

Figure 4-5. Identities and FIT training dependencies (indicated by arrows) of ALR models. *C* is the context order and *D* is the duration order.

Model 1 is the same first-order 16 state ergodic HMM that we used in Section 4.6. It serves as a base-line model and all other models are extensions of it. The fixed-order models F2 and F3 (generically termed Fm) are also directly imported from this previous section. As noted in Section 2.5.3 (p. 50), training on data patterns that repeats before moving on to a new state, can effectively diminish the *C* value of models and/or portions thereof. The C2 and C3 models (generically termed Cm), while neglecting duration modelling are *ensuring* that the system “knows” about respectively 2 and 3 *distinct* prior *states* when making a transition to a next state. The longer history that is being modelled by maintaining the *C* values in this way, necessarily makes these models more sensitive to training data deficiencies. Models D1 and D2 (generically Dn) adds duration to model 1, while C2D2, C2D3, C3D2 and

C3D3 (generically CmDn) add duration information to the respective models which they extend. DnCm models were not considered since they, similar to the Fm models, also suffer from a reduced context order when trained on repetitive data (see Section 3.3.3 p. 69). We do not suggest that the above topologies are the only viable ones. Investigating others, however, is reserved for future work.

4.7.2 Model sizes

The number of transition probabilities in the trained models is shown in Figure 4-6. From this it is quite clear that the various Cm models are considerably larger than the corresponding Fm models (though still smaller than the Xm models considered in a previous section; C3D3 is five times smaller than X3 – marked ext/ORED *R*=3 in Figure 4-4).

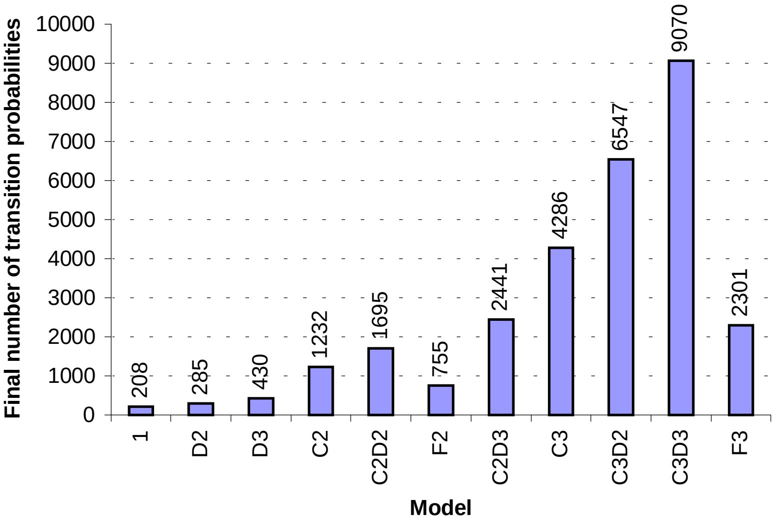

Figure 4-6. Number of transition probabilities in the trained HMMs. (Averaged over the English and Hindi models.)

It must be kept in mind though that these topologies were designed for different purposes. We suspect that the growth in the number of transition probabilities of the Cm models can be attributed to the larger history of states that it is modelling. The longer such a history is, the greater the potential number of state combinations will be. When duration modelling is added to form the CmDn models, it is done on an already enlarged model, thereby contributing to a marked growth in the number of parameters. As previously explained, multiple feature frames describing the same phoneme hamper context modelling in the Fm models.

4.7.3 Classification accuracy on training set

The classification accuracy on the training set is given in Table 4-4. The added benefit of each successive FIT extension is clear. The rapid increase in accuracy of the various context-emphasised models indicates an increasing ability to fit the training data.

Table 4-4. Training set classification accuracy for different models.

	5s (2169 trials)			45s (339 trials)		
Ctx\Dur	D=1	D=2	D=3	D=1	D=2	D=3
C=1	1: 83.4%	D2: 84.1%	D3: 84.6%	1: 92.6%	D2: 94.1%	D3: 94.7%
C=2	C2: 89.0%	F2: 85.2% C2D2: 90.3%	C2D3: 91.4%	C2: 98.5%	F2: 95.3% C2D2: 99.1%	C2D3: 99.1%
C=3	C3: 94.6%	C3D2: 95.4%	F3: 89.6% C3D3: 97.3%	C3: 99.7%	C3D2: 100%	F3: 98.8% C3D3: 100%

4.7.4 Classification accuracy on testing set

Table 4-5 reports the classification accuracy on an independent set of testing data. In general the pattern of improvement with each new FIT extension is followed. At the *C*=2 level, the context-emphasised models appear to function very competitively. We suspect that our training database was too small to sustain the long history span utilised at the *C*=3 levels. The large difference in accuracy between the training and testing sets (i.e. specialisation is taking place) for the C3 family also confirms this. In general models benefit from duration modelling.

McNemar significance tests reveal almost identical results to those obtained in Section 4.6.3. In general the higher-order models improve on the baseline HMM1, while no interesting differences could be detected between the higher-order models (see Appendix C.3.3).

Table 4-5. Testing set classification accuracy for different models.

	5s (247 trials)			45s (39 trials; NIST’95)		
Ctx\Dur	D=1	D=2	D=3	D=1	D=2	D=3
C=1	1: 69.2%	D2: 79.4%	D3: 80.6%	1: 82.1%	D2: 92.3%	D3: 92.3%
C=2	C2: 75.7%	F2: 76.1% C2D2: 78.1%	C2D3: 77.7%	C2: 92.3%	F2: 89.7% C2D2: 92.3%	C2D3: 94.9%
C=3	C3: 76.9%	C3D2: 76.1%	F3: 79.8% C3D3: 76.5%	C3: 89.7%	C3D2: 89.7%	F3: 97.4% C3D3: 94.9%

It is safe to conclude that increasing the duration and/or context orders is indeed beneficial, as long the training database is large enough to sustain it. The various CmDn models hold much promise for ALR, but needs more extensive testing on larger databases. Although it might seem unfair to compare results from the simple first-order model to that of large higher-order models, the purpose of these experiments was to investigate the role of context and duration modelling HMMs, and not to compare models containing an equal number of parameters.

4.8 Comparison with prior work

The best ALR system (F3) from the previous section can be described as 1) not needing a transcribed training database while 2) primarily relying on phonotactic constraints as the principal source for language identification. To compare with other systems, we will only consider our results for a two-language task, namely 3) using the NIST’95 45s set for English and Hindi data. Unfortunately, we could not find any direct comparisons for this specific language pair in the available literature.

Furthermore, this is a rather limited test set. Therefore results should correspondingly be interpreted as providing only a broad indication of behaviour. It should also be kept in mind that most of the systems that we will compare with are optimised products including many enhancements. Our system was designed to illustrate high-order HMM concepts. Many enhancements are available to improve it further (see Section 4.9).

This work is conceptually similar to the original work by House and Neuberg (1977), Savic *et al.* (1991) and to Du Preez (1991b-1993). Unfortunately, as standardised ALR databases were not yet available, those early efforts used small in-house databases and direct comparison is not possible. By duplicating our original work using the OGI-TS database, we have shown in the previous section that higher-order HMMs can significantly improve on those early systems.

Lund *et al.* (1996) introduces a system, focussing on acoustic-phonetic aspects, that *does not require any transcriptions* during training and can thus be directly compared to our work. On language pairs from the NIST'95 set, they achieve accuracy ranging from 85.2% to 93.6%. Although based on different principles, the 97% accuracy that we achieved compares well with this.

To make further comparisons, we need to look at *techniques that do incorporate transcriptions* in the training process. The phoneme-recognition followed by N-gram approach is related to ours in the sense that both focus on phonotactic information. Zissman (1995a), also modelling gender information and using interpolation to smooth transition probabilities, averages an accuracy of 97.9% when classifying between pairs of languages (NIST'94 45s. set). With a fairly sophisticated system

Kadambe and Hieronymus (1995) achieve results varying from 86% up to 100% (average 94.8%), depending on the specific language pair.

From all of these results, taking into account that our system does not implement enhancements and also does not require transcribed databases, we conclude that higher-order HMM modelling can result in competitive ALR systems. Indications are that, since our system cannot benefit from prior information in the form of pre-trained phoneme models, it will also require larger training databases for adequate training. Finally, we also expect that approaches based on explicit phoneme recognition preceding N-gram modelling will degrade more rapidly when presented with difficult conditions such as those found on very noisy channels. Since our system does not use early hard decisions, degradation should be more graceful.

The investment required for, and complexity of, LVCSR based ALR systems put them outside the scope of systems that are comparable to our approach.

4.9 Outstanding issues

We view the ALR experiments of Section 4.7 as a prototypical demonstration of concept. Many necessary refinements are absent from it and there are also several aspects that require further investigation. In order to expand it into a full-blown ALR system incorporating many languages, at least the following should receive attention:

1) From the preceding work it is clear that several topologies are useful. The different alternatives (existing and new) must be investigated further.

2) Other parameters like the order (standard, context and duration) and the number of states need to be optimised.

3) A thorough investigation of the relationship between the size of the models and the size of a database sufficiently large to train it is necessary.

4) As we can accept that we will never have quite enough training data, we also need to smooth the values of badly estimated higher-order probabilities using better estimated lower order probabilities. Interpolation techniques are commonly used in N-grams applications (Zissman, 1995a).

5) From previously cited work it is quite clear that explicitly incorporating gender in the models, will increase accuracy. A simple and frequently used approach to do this is to simply use separate models for the sexes, possibly combining their scores in a soft manner (Zissman, 1995a).

6) Previous work (Mendoza *et al.*, 1996) indicates the usefulness of explicitly removing the acoustic component from the resultant score when matching unknown speech to a language model. Although our system uses a common acoustical model for the languages concerned, removing it altogether from the final matching score should be investigated.

7) The current system models pseudo-phonemes (or broad sound categories) with single HMM states. It may prove fruitful instead to imbed detailed phoneme models in a larger HMM. Similar to our context model, these sub-models can then be arranged in the bigger structure to allow higher-order modelling of specific sequences of them. This structure can be grown incrementally by using a special form of the FIT algorithm. This approach is likely to re-introduce a requirement for transcriptions, although this can be limited to initialising the sub-models. Alternatively, related models can be “grown” by subdividing states in our current system (Ostendorf & Singer, 1997).

4.10 Summary and conclusions

This chapter shows that the FIT algorithm is indeed practical for training large real-life high-order HMMs. It confirms the benefits over extended/ORED training that we found in previous simulation experiments. The FIT algorithm provides greater computational efficiency, results in more compact models and if anything, increases accuracy. We demonstrated some very promising prospects for implementing ALR systems that do not need transcriptions. We also demonstrated techniques that achieve independent control over context modelling (modelling sequences of consecutive states while ignoring their individual repetitions) and duration modelling (modelling the duration/repetitions for which a specific state is active) in higher-order HMMs.

Chapter 5 Conclusion

5.1 Concluding perspective

As noted in the opening paragraphs of this dissertation, high-order HMMs have, until now, generally been considered to be powerful but unpractically expensive, with capabilities exceeding those of the popular first-order HMM. This work helps to capture this elusive power. Our perspective is that HMMs of all orders are just simply first-order HMMs, and can all be expressed in a common form by using the ORder rEDucing (ORED) algorithm (Section 2.3). They all share the same inherent representational capacity. *High-order transition probabilities are simply an elegant mathematical way of specifying the topology of the model.* This insight cuts through the confusing aspects to expose the important high-order HMM issues:

- How can they be trained efficiently? The topology arising from high-order specification can result in such a rich set of parameters that training them can be prohibitive. To this end we developed the Fast Incremental Training (FIT) algorithm (Section 3.2). It utilises the relationship between transition probabilities of different orders to incrementally avoid the calculation of redundant parameters. This can substantially reduce computational requirements.

- How can training methods improve the quality of the resultant model? The parameter space that is being optimised has local optima. Large pattern recognition systems are prone to train to optima that provides a good fit to (the peculiarities of) the training data, but does not generalise well on previously unseen data. Special strategies are necessary to avoid this. The FIT algorithm does so by first optimising on less complicated parameter spaces and then incrementally expanding it to better-behaved larger parameter spaces (Sections 3.5 and 4.6).

- How can we design topologies to provide specified functionality? This is a very rich field that can now be addressed through the use of high-order hidden Markov modelling. We have found the following useful topologies:

  - ♦ It is immediately clear that first-order topologies can be imbued with a greater sense of context by expanding them to a (fixed) higher order (Section 2.5.1). The potential, however, is richer than this.

  - ♦ Specifying high-order transition probabilities to emphasise the modelling of state duration while maintaining inter-state dependence at first order results in the well known Ferguson duration modelling topology (Section 2.5.2).

  - ♦ In phonotactic modelling (used in automatic language recognition systems), the combinations of distinct phonemes are of importance. In practical systems, the short analysis frames (typically 10 to 20ms) cause apparent phoneme repetitions, but in reality they reflect duration and not phonotactic information. Using high-order transition probabilities, we were able to design a topology that can guarantee a certain width of phonotactic context (Section 2.5.3).

  - ♦ This enhanced context modelling can be combined with duration modelling resulting in topologies with independent control over the duration and phonotactic (distinct state) contexts (Section 3.3.3). These topologies were shown to be applicable to automatic language recognition (Section 4.7).

Topology design also includes an interesting aspect not addressed here. In the field of information theory, the automatic determination of the number of states and topology of HMMs are receiving attention (Liu & Narayan, 1994). Similarly, techniques to grow larger HMMs by subdividing the states of smaller models are also being researched (Ostendorf & Singer, 1997). These techniques aim to infer the appropriate topology directly from the training data alone. Our high-order HMM approach also benefits from external knowledge about the desirable characteristics of the model. It will be interesting to follow the interplay of these approaches in future work.

5.2 Topics for further study

This study is a first exploration of a rich, multi-faceted and largely unknown field. By necessity many aspects are neglected. Some of them are listed in the following:

- In Section 3.3 we discussed the possibility of analysing the states and transitions of a trained fixed-order HMM to detect the presence of mixed-order transitions. By a process of state tying and/or merging followed by retraining (possibly iterative), the number of free parameters in the model can be reduced, thereby enhancing its robustness while decreasing its computational demands.
- A study on the exact mechanisms involved when incrementally training (or growing?) the parameters of a model (as the FIT algorithm does) can be very useful. The way in which this guides the model to converge on better local optima will provide perspectives applicable to pattern recognition systems in general.
- The concept of using high-order transition probabilities to design topology is in its infancy. The examples that we do have were brought about by specific application requirements. Research is needed to determine the *scope* of this technique and establish *procedures* for practising it. This will lead to topologies optimised to the requirements of specific applications.

- The relationship between the size and structure of a model, and the size of a database sufficiently large to train it, must be investigated.

- Due to the large number of parameters in high-order HMMs, techniques providing robustness against training data scarcity is a major issue. With the exception of the convergence properties of the FIT algorithm it received no attention in this research. We expect that techniques from language modelling (Jelinek, Mercer & Roukos, 1992) will be applicable here.

- Our methodology invites large-scale practical application. The work in Chapter 4 is a single demonstration of its applicability to ALR. An evaluation using larger training and testing databases is necessary to make firm conclusions possible. It should include more languages and incorporate the enhancements listed in Section 4.9.

Bibliography

Arslan, L.M. and Hansen, J.H.L. (1997). Frequency characteristics of foreign accented speech. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. 1123 – 1126, Munich, Germany.

Atal, B.S. (1974). Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *Journal of the Acoustical Society of America*, vol. 55, pp 1304 – 1312.

Bahl, L.R., Jelinek, F. and Mercer, R.L. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PAMI-5 no. 2, pp. 179 - 190.

Bond, Z.S. and Fokes, J. (1991). Identifying foreign languages. *XIIth International congress of phonetic sciences*, pp. 198 - 201.

Bourlard, H. and Wellekens, C.J. (1990). Links between Markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 12, no. 12, pp. 1167 – 1178.

Cole, R.A., Inouye, J.W.T., Muthusamy, Y.K. and Gopalakrishnan (1989). Language identification with neural networks: a feasibility study. 2nd *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 525 - 529.

Davis, S.B. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 31, pp. 357 – 366.

De Bruin, J.C. and Du Preez, J.A. (1993). Automatic language recognition based on discriminating features in pitch contours. *Proceedings of the IEEE COMSIG Conference*, pp. 133 - 138, Johannesburg, South Africa.

Deller, J.R., Proakis, J.G. and Hansen J.H.L. (1993). *Discrete time processing of speech signals*. Macmillan.

Dempster, A.P., Laird, N.M. and Rubin, D.B. (1977). Maximum likelihood for incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B,* vol. 39 no. 1, pp. 1 - 38.

Devijver, P.A. and Kittler, J. (1982). Pattern recognition, a statistical approach. Prentice-Hall International.

Du Preez, J.A. (1991a). Modelling duration in hidden Markov models with application to word spotting. *Proceedings of the IEEE COMSIG Conference*, pp. 1 - 5, Sandton, South Africa.

Du Preez, J.A. (1991b). Initial report on Language Recognition by means of Ergodic Hidden Markov Models. *Internal research report*, University of Stellenbosch, South Africa.

Du Preez, J.A. (1992). Language Recognition by means of Ergodic Hidden Markov Models. *Proceedings of the IEEE COMSIG Conference*, pp. 33 - 38, Cape Town, South Africa.

Du Preez, J.A. (1993). Discriminating between Languages by means of Hidden Markov Models. *Internal research report*, University of Stellenbosch, South Africa.

Du Preez, J.A. (1997). Efficient training of high-order hidden Markov models, using first-order representations. Accepted for publication in *Computer Speech and Language.*

Ferguson, J.D. (1980). Variable duration models for speech. *Proceedings of the Symposium on the Application of Hidden Markov Models to Text and Speech* (J.D. Ferguson, editor), pp. 143 - 179. Princeton, New Jersey.

Foil J.T. (1986). Language identification using noisy speech. *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, vol. 2, pp. 861 – 864, Tokyo, Japan.

Fu, K.S. (1982). *Syntactic pattern recognition and applications*. Prentice-Hall.

Furui, S. (1981). Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29 no. 2, pp. 254 – 272.

Gillick, L. and Cox, S.J. (1989). Some statistical issues in the comparison of speech recognition algorithms. *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, pp. 532 - 535, Glasgow, UK.

Goodman, F.J., Martin, A.F. and Wohlford, R.E. (1989). Improved automatic language identification in noisy speech. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 528 - 531, Glasgow, UK.

Gray, R.M. (1984). Vector quantization. *IEEE Acoustics Speech and Signal Processing Magazine*, vol. 1 no. 2, pp. 4 – 29.

Hazen, T.J. and Zue, V.W. (1997). Segment-based automatic language identification. *Journal of the Acoustical Society of America* vol. 101, no. 4, pp. 2323 – 2331.

He, Y. (1988). Extended Viterbi algorithm for second-order hidden Markov process. *Proceedings of the IEEE 9th International Conference on Pattern Recognition*, pp. 718 - 720. Rome, Italy.

Hieronymus, J.L. and Kadambe, S. (1997). Robust spoken language identification using large vocabulary speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. 1111 – 1114, Munich, Germany.

House, A.S. and Neuberg, E.P. (1977). Toward automatic identification of the language of an utterance. I. Preliminary methodological considerations. *Journal of the Acoustical Society of America* vol. 62 no. 3, pp. 708 – 713.

Howard, R.A. (1971). *Dynamic probabilistic systems. Volume 1: Markov models.* John Wiley and Sons.

Huang, X.D. and Jack, M.A., Semi-continuous hidden Markov models for speech signals. *Computer Speech and Language*, vol. 7, no. 4, pp. 239 – 252.

Jelinek, F. Mercer, R.L. and Roukos S. (1992). Principles of lexical language modelling for speech recognition. In *Advances in speech signal processing. (S. Furui and M.M. Sondhi eds.), Marcel Dekker. New York.*

Juang, B.H. (1985). Maximum likelihood estimation for mixture multivariate stochastic observations of Markov chains. *AT&T Technical Journal,* vol. 64, no. 6, pp. 1235 – 1249.

Juang, B.-H. and Rabiner, L.R. (1985). Mixture autoregressive hidden Markov models for speech signals. *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 33, no. 6, pp. 1404 - 1413.

Juang, B.-H. and Rabiner, L.R. (1992). Issues in using hidden Markov models for speech recognition. In *Advances in speech signal processing. (S. Furui and M.M. Sondhi eds.), Marcel Dekker. New York.*

Kadambe, S. and Hieronymus, J.L. (1995). Language identification with phonological and lexical models. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3507 - 3510. Detroit, USA.

Kriouile, A., Mari J.-F. and Haton J.-P. (1990). Some improvements in speech recognition based on HMM. *Proceedings of the IEEE International Conference on Acoustics*, pp. 545 - 548. Albuquerque, USA.

Kundu, A., He, Y. and Bahl, P. (1988). Recognition of handwritten word: First and second-order hidden Markov model based approach. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 457 - 462. Ann Arbor, Michigan.

Lee, K.F. (1989). *Automatic speech recognition: The development of the SPHINX system*. Kluwer Academic publishers.

Leon-Garcia (1989) *Probability and random processes for Electrical engineering.* Addison Wesley.

Levinson, S.E. (1985). Structural methods in automatic speech recognition. *Proceedings of the IEEE*, vol. 73 no. 1, pp. 1626 - 1650.

Levinson, S.E. (1986). Continuously variable duration hidden Markov models for automatic speech recognition. *Computer Speech and Language*, vol. 1 no. 1, pp. 29 - 45.

Li, K.-P. (1995). Experimental improvements of a language id system. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3515 - 3518. Detroit, USA.

Liu, C.-C. and Narayan, P. (1994). Order estimation and sequential universal data compression of a hidden Markov source by the method of mixtures. *IEEE Transactions on Information Theory*, vol. 40, no. 4, pp. 1167 - 1180.

Lund, M.A., Ma, K. and Gish, H. (1996). Statistical language identification based on untranscribed training. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 793 - 796. Atlanta, USA.

Marcheret, E. and Savic, M.I., (1997). Random walk theory applied to language identification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. 1119 – 1122, Munich, Germany.

Mari, J.-F., Fohr D. and Junqua J.C. (1996). A second-order HMM for high performance word and phoneme-based continuous speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 435 - 438. Atlanta, USA.

Mari, J.-F. and Haton, J.-P. (1994). Automatic word recognition based on second-order hidden Markov models. In *ICSLP-94*, pp. 247 - 250.

Mari, J.-F., Haton, J.-P. and Kriouile A. (1997). Automatic word recognition based on second-order hidden Markov models. *IEEE Transactions on Speech and Audio processing*, vol. 5 no. 1, pp. 22 - 25.

Markel, J.D. and Gray, A.H. (1976). *Linear prediction of speech.* Springer-Verlag.

Mendoza, S., Gillick, L., Ito, Y., Lowe, S. and Newman, M. (1996). Automatic language identification using large vocabulary continuous speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 785 - 788. Atlanta, USA.

Moon, T.K. (1996). The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, vol. 13 no. 6, pp. 47 - 60.

Muthusamy, Y.K., Cole, R.A. and Oshika, B.T. (1992). The OGI Multi-language Telephone Speech Corpus. *Proceedings of the International Conference on Spoken Language Processing*, pp. 895 - 898. Banf, Alberta, Canada.

Muthusamy, Y.K., Barnard, E. and Cole, R.A. (1994). Reviewing automatic language identification. *IEEE Signal Processing Magazine*, vol. 11 no. 4, pp. 33 – 41.

Navrátil, J. and Zühlke, W. (1997). Double bigram-decoding in phonotactic language identification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. 1115 – 1118, Munich, Germany.

O'Shaugnessy, D. (1990). *Speech Communication: Human and Machine*. Addison Wesley.

Ostendorf, M. and Singer, H. (1997). HMM topology design using maximum likelihood successive state splitting *Computer Speech and Language*, vol. 11, no. 1, pp. 17 – 41.

Parris, E.S., and Carey, M.J. (1995). Language identification using multiple knowledge sources. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3519 - 3522. Detroit, USA.

Poritz, A.B. (1988). Hidden Markov models: a guided tour. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7 - 12. New York, USA.

Prasad, K.V.K.K. and Lamba, T.S. (1988). Automatic language recognition based on phonetic structure of languages. *Journal: Institution of Electronics and Telecommunication Engineers (India)*, vol. 34 no. 1, pp. 63 - 67.

Rabiner, L.R., Levinson, S.E. and Sondhi, N.M. (1983). On the application of vector quantization and hidden Markov models to speaker-independent, isolated word recognition. *Bell System Technical Journal*, vol. 62, pp. 1075 - 1105.

Rabiner, L.R., Juang, B.-H., Levinson, S.E. and Sondhi, N.M. (1985). Recognition of isolated digits using HMMs with continuous mixture densities. *AT&T Technical Journal*, vol. 64, no. 6, pp. 1211 – 1233.

Rabiner, L.R. and Juang, B.-H. (1986). An introduction to hidden Markov models. *IEEE Acoustics Speech and Signal Processing Magazine*, vol. 3 no. 1, pp. 4 – 16.

Raudys, S. and Jain, A. (1991). Small sample size effects in statistical pattern recognition: Recommendations for practitioners, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp 252 – 264.

Riccardi, G., Pieraccini, R. and Bocchieri, E. (1996). Stochastic automata for language modelling. *Computer Speech and Language*, vol. 10, no. 4, pp. 265 - 293.

Savic, M., Acosta, E. and Gupta, S.K. (1991). An automatic language recognition system. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 817 - 820. Toronto, Canada.

Schalkoff, R.J. (1992). *Pattern recognition: statistical, structural and neural approaches.* Wiley.

Schultz, T., Rogina, I. and Waibel, A. (1996). LVCSR-based languate identification. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 781 - 784. Atlanta, USA.

Sugiyama, M. (1991). Automatic language recognition using acoustic features. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 813 - 816. Toronto, Canada.

Yan, Y. and Barnard, E. (1995). An approach to automatic language identification based on language-dependent phone recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3511 - 3514. Detroit, USA.

Yan, Y. and Barnard, E. (1996). Experiments for an approach to language identification with conversational speech. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 789 - 792. Atlanta, USA.

Young, S. (1996). A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing Magazine,* vol. 13 no. 5, pp. 45 - 57.

Zissman, M.A. (1993). Automatic language identification using Gaussian mixture and hidden Markov models.. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol 2, pp. 399 – 402. Minnesota, USA.

Zissman, M.A. and Singer, E. (1994). Automatic language identification of telephone speech messages using phoneme recognition and N-gram modelling. *Proceedings of the IEEE International Conference on Acoustics*, *Speech and Signal Processing*, pp. I-305 - I-308. Adelaide, Australia.

Zissman, M.A. (1995a). Language identification using phoneme recognition and phonotactic language modelling. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3503 - 3506. Detroit, USA.

Zissman, M.A., Gleason, T.P., Rekart, D.M., and Losiewicz, B.L. (1995b). Automatic dialect identification of extemporaneous conversational, Latin American Spanish speech. *Proceedings of the IEEE International Conference on Acoustics*, *Speech and Signal Processing*, pp. 777 - 780. Detroit, USA.

Zissman, M.A. (1996). Comparison of four approaches to automatic language identification of telephone speech. *IEEE Transactions on Speech and Audio processing*, vol. 4 no. 1, pp. 31 - 44.

Appendix A Detailed example of the ORED procedure

In Section 2.4.1 we briefly demonstrated how the ORED algorithm reduces a simple mixed-order HMM (shown in Figure A-1) to its equivalent first-order version. This section provides the step-by-step details for this process.

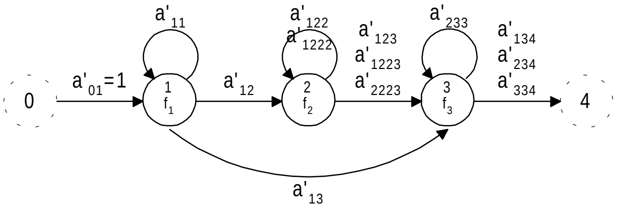

Figure A-1. A mixed-order HMM (maximum order 3).

Iteration 1:

Step 1a: Create the states S=0, 1, 2, 3 and 4

Step 1b: Add to them the states S=(1,2), (2,2), (2,3), (3,3), (1,3) and (3,4)

Step 1c: Not applicable (Q=4 is only entered by links originating at Q=3)

Step 2a: a'_{01} between S=0 and S=1

a'_{11} between S=1 and S=1

a'_{12} between S=1 and S=(1,2)

a'_{13} between S=1 and S=(1,3)

Step 2b: a'_{122} between S=(1,2) and S=(2,2)

a'_{1222} between S=(2,2) and S=(2,2)

a'_{123} between S=(1,2) and S=(2,3)

a'_{1223} between S=(2,2) and S=(2,3)

a'_{2223} between S=(2,2) and S=(2,3)

a'_{233} between S=(2,3) and S=(3,3)

a'_{134} between S=(1,3) and S=(3,4)

a'_{234} between S=(2,3) and S=(3,4)

a'_{334} between S=(3,3) and S=(3,4)

Step 2c: Not applicable

Step 3: Remove states S =2, 3 and 4.

Step 4: S = 0 and (3,4) are null states.

f_1 to S=1

f_2 to S=(1,2) and (2,2)

f_3 to S=(1,3), (2,3) and (3,3)

(See Figure A-2 at this point. Step 3 is omitted to show the full set of states.)

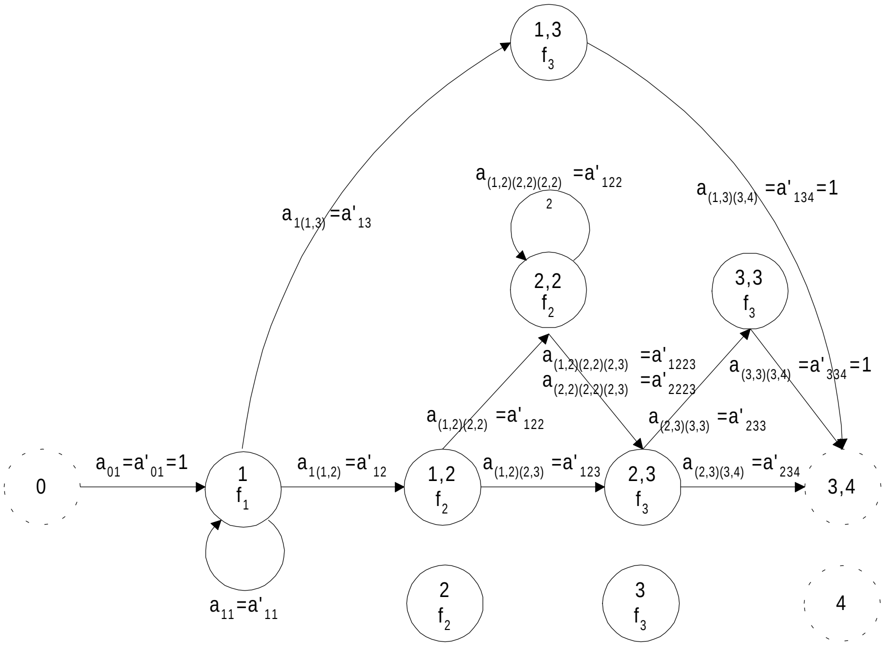
1,3
f_3
$a_{(1,2)(2,2)(2,2)}$ $=a'_{122}$
2
$a_{(1,3)(3,4)}$ $=a'_{134}=1$
$a_{1(1,3)}=a'_{13}$
2,2
f_2
3,3
f_3
$a_{(1,2)(2,2)(2,3)}$ $=a'_{1223}$
$a_{(2,2)(2,2)(2,3)}$ $=a'_{2223}$
$a_{(3,3)(3,4)}$ $=a'_{334}=1$
$a_{(1,2)(2,2)}$ $=a'_{122}$
$a_{(2,3)(3,3)}$ $=a'_{233}$
0
$a_{01}=a'_{01}=1$
1
f_1
$a_{1(1,2)}=a'_{12}$
1,2
f_2
$a_{(1,2)(2,3)}$ $=a'_{123}$
2,3
f_3
$a_{(2,3)(3,4)}$ $=a'_{234}$
3,4
$a_{11}=a'_{11}$
2
f_2
3
f_3
4

Figure A-2. ORED iteration 1, after completion of step 4. Dead states S=2,3 and 4, are supposed to be removed in step 3, but are kept to illustrate all states.

Step 5a: States S are renumbered to form the states Q for the next cycle as shown in Figure A-3

Step 5b: As transition probabilities a_{233}, a_{235} and a_{335} ($a_{(1,2)(2,2)(2,2)}$, $a_{(1,2)(2,2)(2,3)}$ and $a_{(2,2)(2,2)(2,3)}$ in Figure A-2) contains more than two subscripts (i.e. they have an order > 1), we now iterate by returning to step 1 with Figure A-3 as initial model.

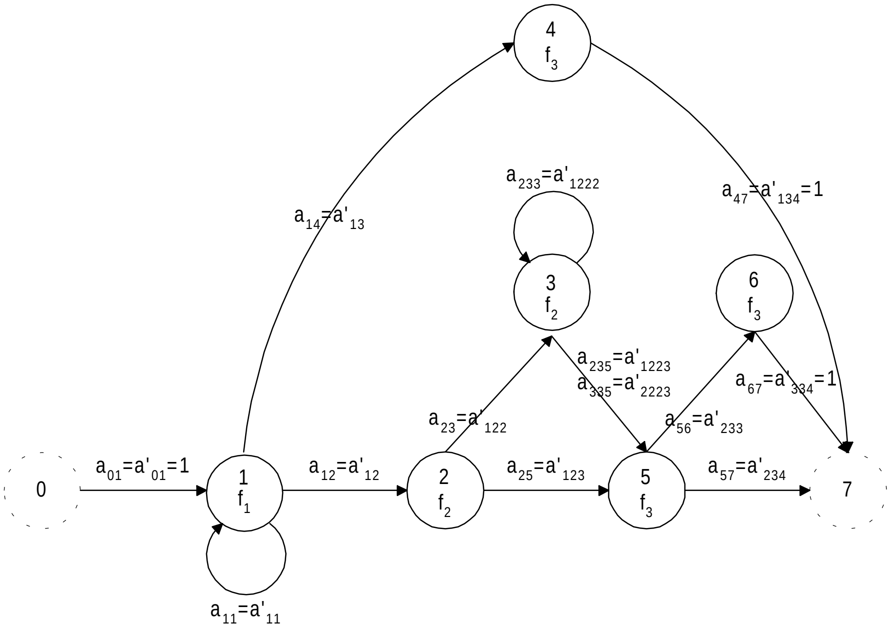

Figure A-3. ORED iteration 1, after completion of step 5a. This is the initial model for the next iteration of ORED. Only links departing from S=3 are still of order higher than one.

Iteration 2:

Step 1a: Create the states S=0, 1, 2, 3, 4, 5, 6 and 7

Step 1b: Add to them the states S=(2,3), (3,3) and (3,5)

Step 1c: Not applicable (Q=7 is only entered by first-order links)

Step 2a: $a_{01} = a'_{01}$ between S=0 and S=1;

$a_{11} = a'_{11}$ between S=1 and S=1

$a_{12} = a'_{12}$ between S=1 and S=2

$a_{14} = a'_{13}$ between S=1 and S=4

$a_{23} = a'_{122}$ between S=2 and S=(2,3)

$a_{25} = a'_{123}$ between S=2 and S=5

$a_{56} = a'_{233}$ between S=5 and S=6, between S=(3,5) and S=6 (tied)

$a_{47} = a'_{134}$ between S=(1,3) and S=(3,4)

$a_{57} = a'_{234}$ between S=4 and S=7, between S=(3,5) and S=7 (tied)

$a_{67} = a'_{334}$ between S=6 and S=7

Step 2b:

$a_{233} = a'_{1222}$ between S=(2,3) and S=(3,3)

$a_{235} = a'_{1223}$ between S=(2,3) and S=(3,5)

$a_{335} = a'_{2223}$ between S=(3,3) and S=(3,5)

Step 2c: Not applicable

Step 3: Remove state S =3

Step 4: S = 0 and 7 are null states.

f_1 to S=1

f_2 to S=2, (2,3) and (3,3)

f_3 to S=4, 5, (3,5) and 6

(See Figure A-4 for result at this point. Step 3 is omitted to show all states.)

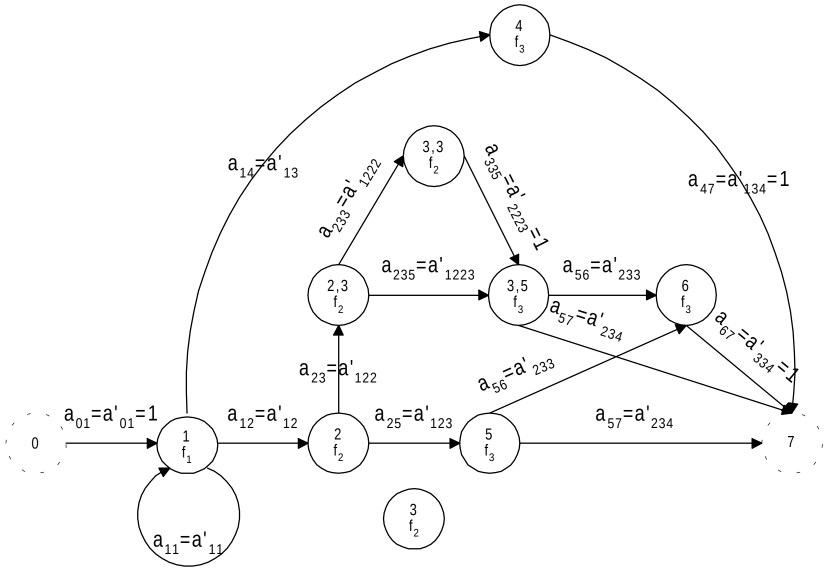

Figure A-4. ORED iteration 2, after completion of step 4. Since there are only first-order transition probabilities present in model, step 5 will be followed by termination in step 6. Transition probability values to the left of the assignment still refer to Figure A-3. Dead state S=3 is supposed to be removed in step 3, but is kept to show all states. States S=5 and S= (3,5) are tied.

Step 5a: States are renumbered.

Step 5b: All transition probabilities have only two subscripts i.e. they are all of first order.

(See Figure A-5 for result at this point, once again step 3 is omitted.)

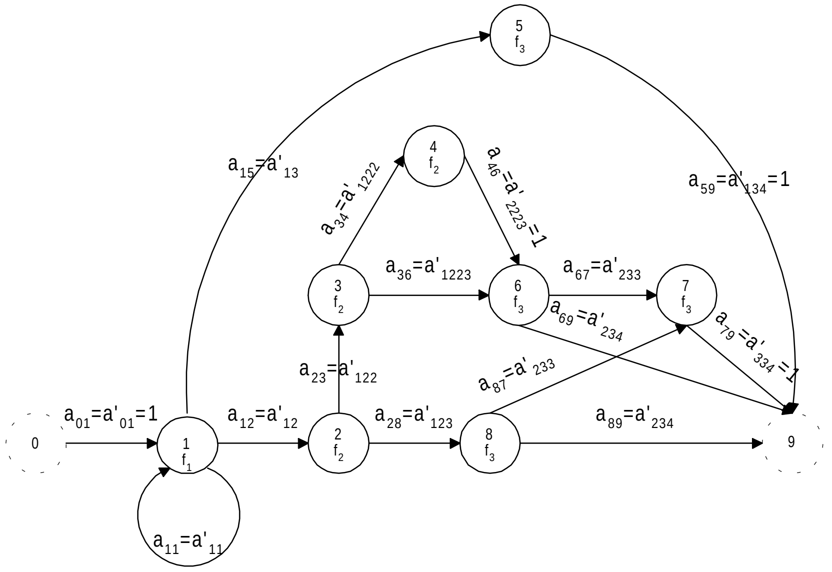

Figure A-5. ORED iteration 2, after completion of step 5. States S=6 and S=8 are tied and lead to the same destinations. They can therefore be merged.

Step 6: Since states S=6 and S=8 in Figure A-5 are tied and also share the same successors, they are merged into state S=6 of Figure A-6. States are renumbered and the algorithm terminates.

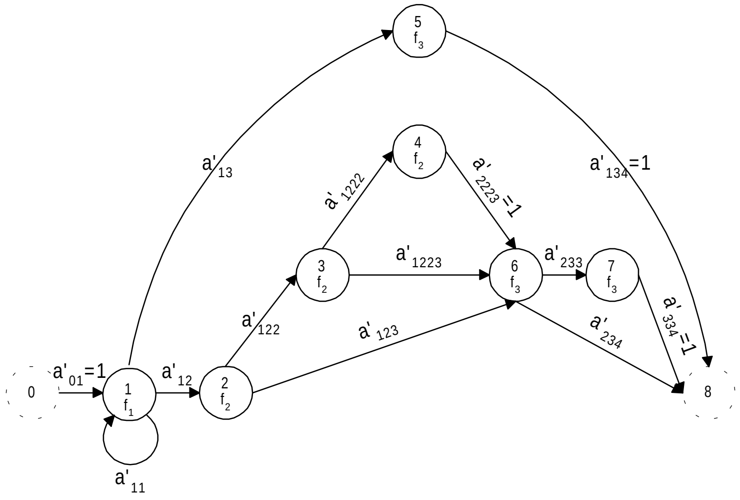

Figure A-6. Finally, the first-order equivalent of Figure A-1.

Appendix B ORED reduction for context modelling

In Section 2.5.3 we introduced an HMM topology that guarantees that a context of *C* (the context order) distinct states will be taken into account when making a transition. We indicated in that section that this model is of mixed-order and has a maximum order of infinity. This makes direct application of the ORED algorithm impossible. In the following we provide an ORED procedure reducing the model of Figure B-1. Figure B-2 and Figure B-3 show the result of the ORED algorithm after the first and second cycles.

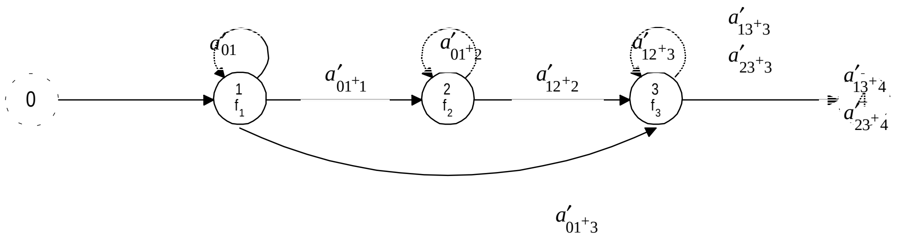

Figure B-1. Second contextual order left to right HMM with one state skip. The notation k^+ is used to indicate one or more occurrences of index *k*. The family of transition probabilities indicated by each symbol *a* in the figure are all identical.

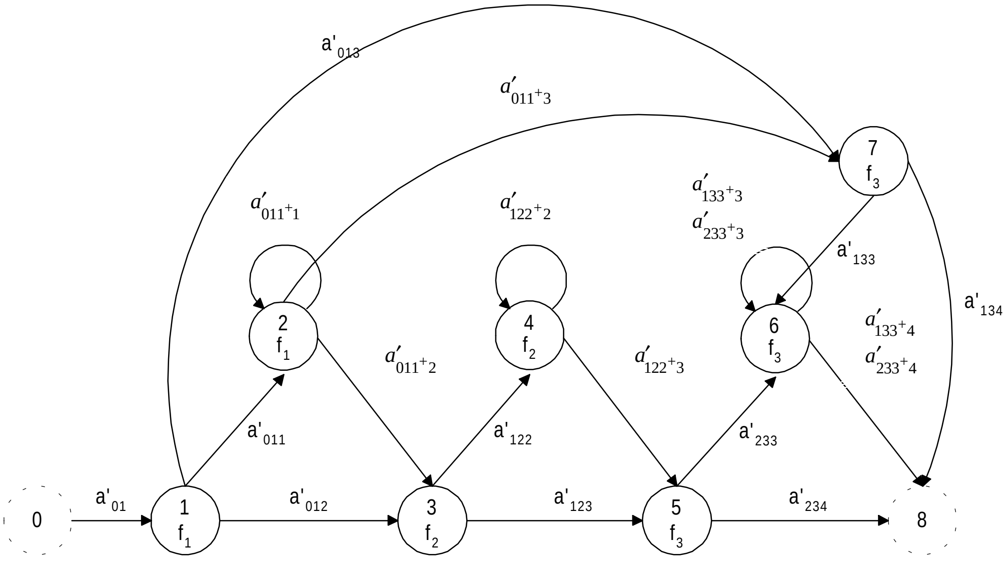

Figure B-2. Result after first cycle of ORED algorithm on Figure B-1.

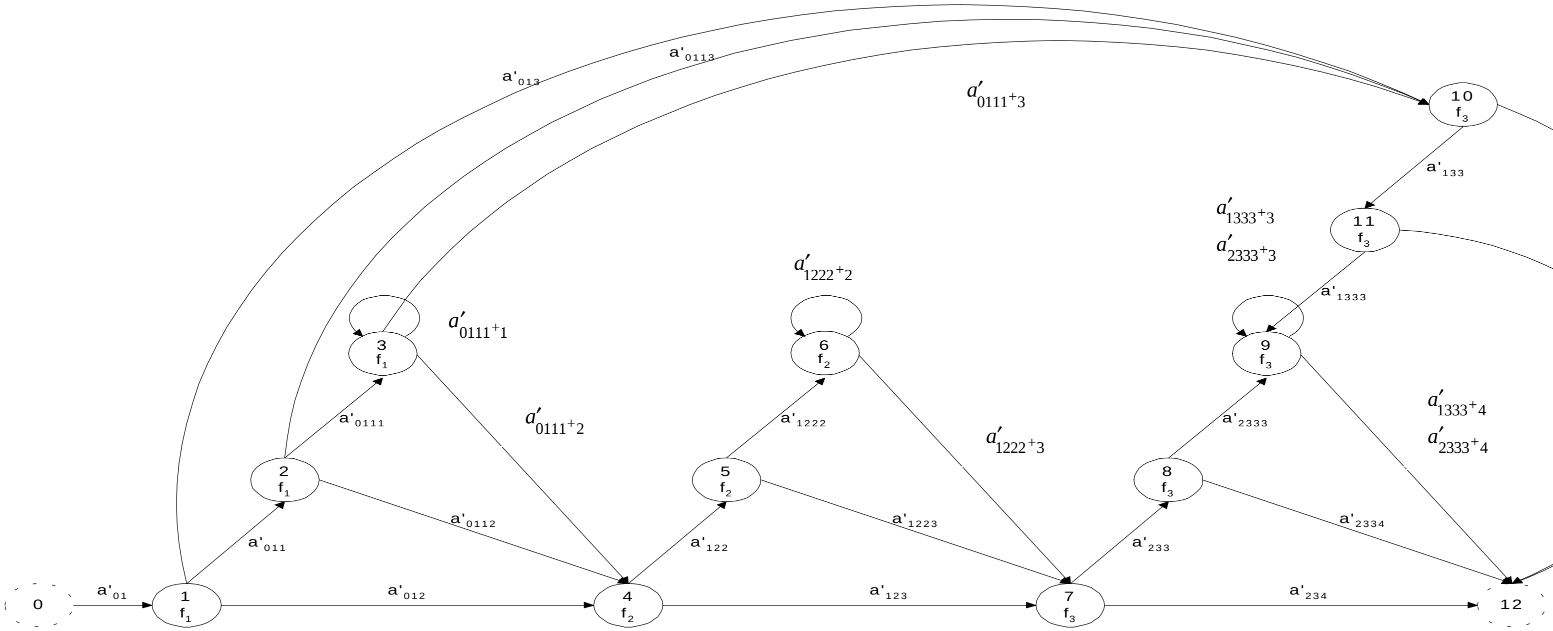

Figure B-3. Result after second cycle of ORED algorithm[37].

[37] Merging takes place between tied states that share a common set of destination states for their transitions. The ORED algorithm dictates that merging should take place during step 6 (after all cycles have completed). Merging can be moved to just after step 2 if none of the (common) destination states will split in following cycles. This is the case if only first order transition probabilities enter or leave the destination states. This early merging is used in Figure B-3 to simplify the procedure.

From each of these figures, note the four substructures[38] of which a generalised version is shown in Figure B-4. These piles will grow one state higher with each cycle of the ORED algorithm. Due to the infinite order of the model they will ultimately be infinitely high, making direct application of ORED impractical. The transition probabilities a'_{hi^+i} ascending this pile are all equal to each other. All the states they ascend to share the same pdf. f_i. Similarly all the transitions leaving the pile share a common transition probability $a'_{hi\,j}{\bullet}$. This makes it possible to reduce this infinite pile to the simple model illustrated in Figure B-5. Applying this reasoning to either Figure B-2 or Figure B-3 results in Figure B-6 as the first-order equivalent of Figure B-1.

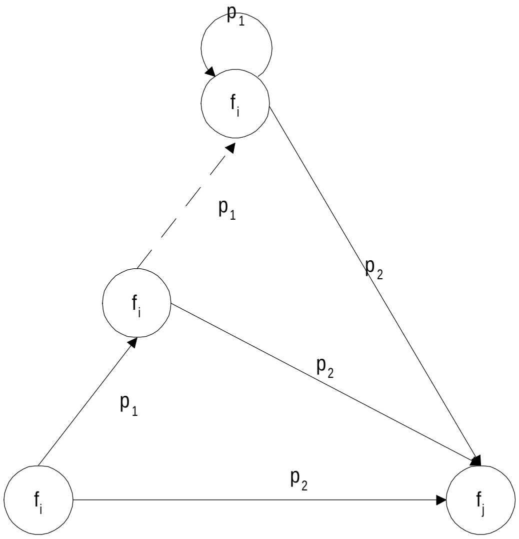

Figure B-4. Repeating (infinite) substructures of Figure B-2 and Figure B-3.

[38] In Figure B-2 the four groups are (ignoring the single state to the right): {1, 2}, {3, 4}, {5, 6} and {7, 6}. In Figure B-3 they are {1, 2, 3}, {4, 5, 6}, {7, 8, 9} and {10, 11, 9}.

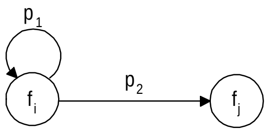

Figure B-5. Finite equivalent of Figure B-4.

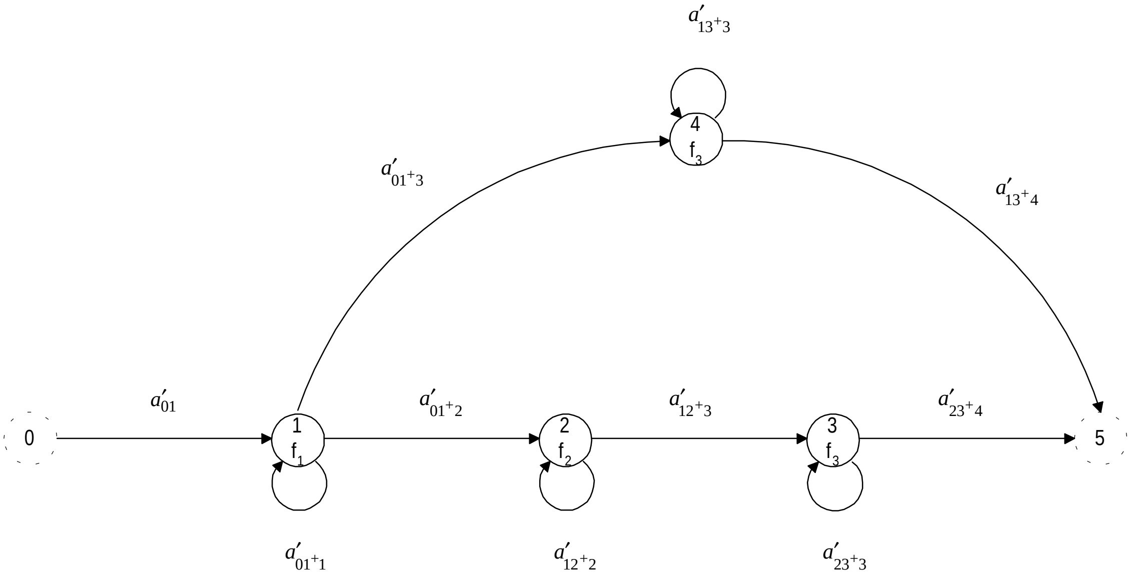

Figure B-6. First-order equivalent of the HMM in Figure B-1.

Appendix C Statistical significance tests

C.1 Overview of McNemar significance test

When comparing two algorithms for classification accuracy, it is important to bear in mind that the measured accuracies are also random variables. We therefore need to ascertain if the measured differences between them are indeed statistically significant. It is necessary to test the following two hypotheses in a mathematically principled manner:

$\mathbf{H_0}$: The algorithms are equally accurate.

$\mathbf{H_1}$: The algorithms are not equally accurate.

Specifically, if the probability that the differences between the algorithms can be attributed to chance can be calculated, conclusions can be drawn with specified certainty. This quantity is a function of the amount of data that is used to evaluate the hypothesis. If only small differences exist between two algorithms, a large set of data will be necessary to establish this with confidence. In other words, the rejection of $\mathbf{H_1}$ may be traced to two sources. Either the two algorithms are in reality equally accurate, or it may be possible that the set of data used in the test was just not large enough to establish clearly the difference. The McNemar significance test is applicable when a number of common data segments are to be classified with two different algorithms. The joint performance of the two algorithms can be represented in a 2x2 matrix as follows:

Table C-1. The number of occurrences of joint classification outcomes for the two algorithms. *N* is the random variable while *n* is its outcome.

		Algorithm 2	
		Correct	Incorrect
Algorithm 1	Correct	$N_{00}=n_{00}$	$N_{01}=n_{01}$
	Incorrect	$N_{10}=n_{10}$	$N_{11}=n_{11}$

The MacNemar test takes the viewpoint that N_{00} and N_{11} describe the behaviour common to both algorithms and should therefore be ignored. The off-diagonal elements N_{10} and N_{01}, however, represent the differences between them. The number of occurrences for which only one of the algorithms made an error is given by $K = N_{10}+N_{01}$, with outcome $K=k$. It can be shown (Gillick & Cox, 1989) that under hypothesis $\mathbf{H_0}$, N_{10} has a binomial B(k, 0.5) distribution. The probability P of observing the given value can then be calculated as:

$$P = \begin{cases} 2\,\mathrm{P}(n_{10} \le N_{10} \le k) \text{ when } n_{10} > k/2 \\ 2\,\mathrm{P}(0 \le N_{10} \le n_{10}) \text{ when } n_{10} < k/2 \\ 1.0 \qquad\qquad\qquad \text{ when } n_{10} = k/2 \end{cases} . \tag{C-1}$$

For k > 50 and n_{10} not too close to 0 or k, $W = \dfrac{|\,N_{10} - k/2\,| - 0.5}{\sqrt{k/4}}$ approximates a N(0,1) Gaussian random variable. The probability P can then be approximated as:

$$P = 2\,\mathrm{P}(W \ge w)\,. \tag{C-2}$$

C.2 Simulation experiments of Section 3.5.2

C.2.1 Configurations evaluated

Table C-2. The four different generating HMM experiments evaluated. “σ Spacing” is the distance between the Gaussian centroids.

Experiment	1	2	3	4
Order	2	3	4	2
States	8	8	8	32
σ Spacing	1.5 -3	1.5 -3	1.5 -3	1-2
Max links	656	5264	42128	34880
Ave links	134	398	1232	974
Sparseness	20.4%	7.6%	2.9%	2.8%

C.2.2 Results on training data

Table C-3. McNemar counts for data configuration 1 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1.

Generating		Extended/ORED		FIT		HMM1		
39395	0	39068	327	39001	394	38501	894	Generating
0	605	215	390	199	406	215	390	
		39283	0	38931	352	38388	895	Extended/ORED
		0	717	269	448	328	389	
				39200	0	38472	728	FIT
				0	800	244	556	
						38716	0	HMM1
						0	1284	

Table C-4. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-3. Models likely to differ in accuracy are shaded.

Extended/ORED	FIT	HMM1	
1.86E-06	1.55E-15	0	Generating
	0.001	0	Extended/ORED
		0	FIT

Table C-5. McNemar counts for data configuration 2 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1.

Generating		Extended/ORED		FIT		HMM1		
39198	0	38768	430	38685	513	37727	1471	Generating
0	802	378	424	238	564	297	505	
		39146	0	38594	552	37657	1489	Extended/ ORED
		0	854	329	525	367	487	
				38923	0	37667	1256	FIT
				0	1077	357	720	
						38024	0	HMM1
						0	1976	

Table C-6. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-5. Models likely to differ in accuracy are shaded.

Extended/ORED	FIT	HMM1	
0.072785	0	0	Generating
	7.53E-14	0	Extended/ORED
		0	FIT

Table C-7. McNemar counts for data configuration 3 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1.

Generating		Extended/ORED		FIT		HMM1		
39247	0	38939	308	38954	293	37710	1537	Generating
0	753	336	417	210	543	264	489	
		39275	0	38894	381	37689	1586	Extended/ ORED
		0	725	270	455	285	440	
				39164	0	37716	1448	FIT
				0	836	258	578	
						37974	0	HMM1
						0	2026	

Table C-8. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-7. Models likely to differ in accuracy are shaded.

Extended/ORED	FIT	HMM1	
0.287352	0.000256	0	Generating
	1.62E-05	0	Extended/ORED
		0	FIT

Table C-9. McNemar counts for data configuration 4 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1.

Generating		Extended/ORED		FIT		HMM1		
39894	0	39841	53	39831	63	39608	286	Generating
0	106	60	46	40	66	35	71	
		39901	0	39815	86	39601	300	Extended/ ORED
		0	99	56	43	42	57	
				39871	0	39611	260	FIT
				0	129	32	97	
						39643	0	HMM1
						0	357	

Table C-10. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-9. Models likely to differ in accuracy are shaded.

Extended/ORED	FIT	HMM1	
0.57246	0.03018	0	Generating
	0.014948	0	Extended/ORED
		0	FIT

C.2.3 Results on testing data set

Table C-11. McNemar counts for data configuration 1 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1.

Generating		Extended/ORED		FIT		HMM1		
39401	0	38987	414	38974	427	38460	941	Generating
0	599	172	427	184	415	223	376	
		39159	0	38847	312	38288	871	Extended/
		0	841	311	530	395	446	ORED
				39158	0	38436	722	FIT
				0	842	247	595	
						38683	0	HMM1
						0	1317	

Table C-12. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-11. Models likely to differ in accuracy are shaded.

Extended/ORED	FIT	HMM1	
0	0	0	Generating
	1	0	Extended/ORED
		0	FIT

Table C-13. McNemar counts for data configuration 2 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1.

Generating		Extended/ORED		FIT		HMM1		
39187	0	38273	914	38558	629	37667	1520	Generating
0	813	275	538	206	607	295	518	
		38548	0	38059	489	37227	1321	Extended/ ORED
		0	1452	705	747	735	717	
				38764	0	37546	1218	FIT
				0	1236	416	820	
						37962	0	HMM1
						0	2038	

Table C-14. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-13. Models likely to differ in accuracy are shaded.

Extended/ORED	FIT	HMM1	
0	0	0	Generating
	4.93E-10	0	Extended/ORED
		0	FIT

Table C-15. McNemar counts for data configuration 3 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1.

Generating		Extended/ORED		FIT		HMM1		
39240	0	37977	1263	38705	535	37689	1551	Generating
0	760	197	563	167	593	258	502	
		38174	0	37805	369	36872	1302	Extended/ORED
		0	1826	1067	759	1075	751	
				38872	0	37519	1353	FIT
				0	1128	428	700	
						37947	0	HMM1
						0	2053	

Table C-16. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-15. Models likely to differ in accuracy are shaded.

Extended/ORED	FIT	HMM1	
0	0	0	Generating
	0	3.57E-06	Extended/ORED
		0	FIT

Table C-17. McNemar counts for data configuration 4 (see Table C-2). The models evaluated are given in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification counts, as is also illustrated and explained in Table C-1.

Generating		Extended/ORED		FIT		HMM1		
39873	0	39567	306	39699	174	39510	363	Generating
0	127	37	90	28	99	61	66	
		39604	0	39485	119	39299	305	Extended/ ORED
		0	396	242	154	272	124	
				39727	0	39433	294	FIT
				0	273	138	135	
						39571	0	HMM1
						0	429	

Table C-18. The probability that observed differences between any two models arose by chance. Calculated from the McNemar counts of Table C-17. Models likely to differ in accuracy are shaded.

Extended/ORED	FIT	HMM1	
0	0	0	Generating
	1.36E-10	0.182802	Extended/ORED
		8.9E-14	FIT

C.3 Automatic language recognition experiments (Sections 4.6 and 4.7)

C.3.1 Configurations evaluated

Table C-19. Identities and descriptions of models used in automatic language recognition experiments. All models indicated as extending another model are trained via the FIT algorithm, models 1, X2 and X3 are trained directly using the Extended/ORED approach.

ID	Context order (C)	Duration order (D)	Extends	Description
1	1	1	-	First-order ergodic
D2	1	2	1	to second-order duration modelling
D3	1	3	D2	to third-order duration modelling
F2	2	2	1	to second-order ergodic
C2	2	1	1	to second-order different state context
C2D2	2	2	C2	to second-order duration modelling
C2D3	2	3	C2D2	to third-order duration modelling
X2	2	2	-	Second-order ergodic, Extended/ORED
F3	3	3	F2	to third-order ergodic
C3	3	1	C2	to third-order different state context
C3D2	3	2	C3	to second-order duration modelling
C3D3	3	3	C3D2	to third-order duration modelling
X3	3	3	-	Third-order ergodic, Extended/ORED

C.3.2 Results on training data

Table C-20. The McNemar classification counts on 5s language recognition trials using 16 state HMMs (see Table C-19). The four values at the intersection of a given pair of models represent the joint classification counts, as in Table C-1.

1	D2	D3	F2	C2	C2D2	C2D3	X2	F3	C3	C3D2	C3D3	X3	
1810 0 0 359	1646 164 179 180	1632 178 202 157	1653 157 194 165	1725 85 206 153	1700 110 258 101	1700 110 282 77	1711 99 172 187	1692 118 251 108	1740 70 312 47	1750 60 320 39	1773 37 337 22	1739 71 272 87	1
	1825 0 0 344	1735 90 99 245	1727 98 120 224	1709 116 222 122	1740 85 218 126	1734 91 248 96	1695 130 188 156	1740 85 203 141	1760 65 292 52	1769 56 301 43	1791 34 319 25	1750 75 261 83	D2
		1834 0 0 335	1703 131 144 191	1707 127 224 111	1739 95 219 116	1744 90 238 97	1692 142 191 144	1749 85 194 141	1764 70 288 47	1771 63 299 36	1801 33 309 26	1758 76 253 82	D3
			1847 0 0 322	1731 116 200 122	1764 83 194 128	1765 82 217 105	1720 127 163 159	1774 73 169 153	1788 59 264 58	1793 54 277 45	1816 31 294 28	1769 78 242 80	F2
				1931 0 0 238	1812 119 146 92	1816 115 166 72	1764 167 119 119	1795 136 148 90	1873 58 179 59	1873 58 197 41	1895 36 215 23	1837 94 174 64	C2
					1958 0 0 211	1884 74 98 113	1765 193 118 93	1828 130 115 96	1900 58 152 59	1911 47 159 52	1927 31 183 28	1854 104 157 54	C2D2
						1982 0 0 187	1775 207 108 79	1853 129 90 97	1905 77 147 40	1922 60 148 39	1952 30 158 29	1874 108 137 50	C2D3
							1883 0 0 286	1766 117 177 109	1806 77 246 40	1822 61 248 38	1842 41 268 18	1816 67 195 91	X2
								1943 0 0 226	1870 73 182 44	1886 57 184 42	1909 34 201 25	1854 89 157 69	F3
									2052 0 0 117	2004 48 66 51	2021 31 89 28	1918 134 93 24	C3
										2070 0 0 99	2048 22 62 37	1933 137 78 21	C3D2
											2110 0 0 59	1966 144 45 14	C3D3
												2011 0 0 158	X3

Table C-21. The probability that observed differences between any two models arose by chance. Calculated according to the McNemar test as applied to the counts of Table C-20. Models likely to differ in accuracy are shaded.

D2	D3	F2	C2	C2D2	C2D3	X2	F3	C3	C3D2	C3D3	X3	
0.45	0.24	0.05	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	1
	0.56	0.15	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	<0.01	D2
		0.47	<0.01	<0.01	<0.01	0.01	<0.01	<0.01	<0.01	<0.01	<0.01	D3
			<0.01	<0.01	<0.01	0.04	<0.01	<0.01	<0.01	<0.01	<0.01	F2
				0.11	<0.01	0.01	0.51	<0.01	<0.01	<0.01	<0.01	C2
					0.08	<0.01	0.37	<0.01	<0.01	<0.01	<0.01	C2D2
						<0.01	0.01	<0.01	<0.01	<0.01	0.07	C2D3
							<0.01	<0.01	<0.01	<0.01	<0.01	X2
								<0.01	<0.01	<0.01	<0.01	F3
									0.11	<0.01	0.01	C3
										<0.01	<0.01	C3D2
											<0.01	C3D3

Table C-22. The McNemar classification counts on 45s language recognition trials using 16 state HMMs (see Table C-19). The four values at the intersection of a given pair of models represent the joint classification counts, as in Table C-1.

1	D2	D3	F2	C2	C2D2	C2D3	X2	F3	C3	C3D2	C3D3	X3	
314 0	307 7	308 6	310 4	314 0	314 0	314 0	310 4	312 2	314 0	314 0	314 0	314 0	1
0 25	12 13	13 12	13 12	20 5	22 3	22 3	13 12	23 2	24 1	25 0	25 0	22 3	
	319 0	315 4	314 5	317 2	318 1	319 0	310 9	318 1	319 0	319 0	319 0	319 0	D2
	0 20	6 14	9 11	17 3	18 2	17 3	13 7	17 3	19 1	20 0	20 0	17 3	
		321 0	316 5	319 2	321 0	321 0	312 9	320 1	321 0	321 0	321 0	320 1	D3
		0 18	7 11	15 3	15 3	15 3	11 7	15 3	17 1	18 0	18 0	16 2	
			323 0	321 2	323 0	323 0	317 6	322 1	323 0	323 0	323 0	323 0	F2
			0 16	13 3	13 3	13 3	6 10	13 3	15 1	16 0	16 0	13 3	
				334 0	333 1	333 1	322 12	332 2	334 0	334 0	334 0	333 1	C2
				0 5	3 2	3 2	1 4	3 2	4 1	5 0	5 0	3 2	
					336 0	335 1	323 13	334 2	336 0	336 0	336 0	335 1	C2D2
					0 3	1 2	0 3	1 2	2 1	3 0	3 0	1 2	
						336 0	322 14	334 2	336 0	336 0	336 0	335 1	C2D3
						0 3	1 2	1 2	2 1	3 0	3 0	1 2	
							323 0	322 1	323 0	323 0	323 0	323 0	X2
							0 16	13 3	15 1	16 0	16 0	13 3	
								335 0	335 0	335 0	335 0	334 1	F3
								0 4	3 1	4 0	4 0	2 2	
									338 0	338 0	338 0	336 2	C3
									0 1	1 0	1 0	0 1	
										339 0	339 0	336 3	C3D2
										0 0	0 0	0 0	
											339 0	336 3	C3D3
											0 0	0 0	
												336 0	X3
												0 3	

Table C-23. The probability that observed differences between any two models arose by chance. Calculated according to the McNemar test as applied to the counts of Table C-22. Models likely to differ in accuracy are shaded.

D2	D3	F2	C2	C2D2	C2D3	X2	F3	C3	C3D2	C3D3	X3	
0.36	0.17	0.05	<0.01	<0.01	<0.01	0.05	<0.01	<0.01	<0.01	<0.01	<0.01	1
	0.75	0.42	<0.01	<0.01	<0.01	0.52	<0.01	<0.01	<0.01	<0.01	<0.01	D2
		0.77	<0.01	<0.01	<0.01	0.82	<0.01	<0.01	<0.01	<0.01	<0.01	D3
			0.01	<0.01	<0.01	1.00	<0.01	<0.01	<0.01	<0.01	<0.01	F2
				0.63	0.63	<0.01	1.00	0.13	0.06	0.06	0.63	C2
					1.00	<0.01	1.00	0.50	0.25	0.25	1.00	C2D2
						<0.01	1.00	0.50	0.25	0.25	1.00	C2D3
							<0.01	<0.01	<0.01	<0.01	<0.01	X2
								0.25	0.13	0.13	1.00	F3
									1.00	1.00	0.50	C3
										1.00	0.25	C3D2
											0.25	C3D3

C.3.3 Testing set results

Table C-24. The McNemar classification counts on 5s language recognition trials using 16 state HMMs (see Table C-19). The four values at the intersection of a given pair of models represent the joint classification counts, as in Table C-1.

1	D2	D3	F2	C2	C2D2	C2D3	X2	F3	C3	C3D2	C3D3	X3	
171 0 0 76	156 15 40 36	156 15 43 33	152 19 36 40	156 15 31 45	149 22 44 32	149 22 43 33	160 11 27 49	154 17 43 33	150 21 40 36	143 28 45 31	142 29 47 29	153 18 44 32	1
	196 0 0 51	185 11 14 37	172 24 16 35	167 29 20 31	170 26 23 28	165 31 27 24	171 25 16 35	175 21 22 29	166 30 24 27	164 32 24 27	164 32 25 26	169 27 28 23	D2
		199 0 0 48	177 22 11 37	166 33 21 27	168 31 25 23	166 33 26 22	176 23 11 37	180 19 17 31	170 29 20 28	167 32 21 27	165 34 24 24	171 28 26 22	D3
			188 0 0 59	159 29 28 31	164 24 29 30	161 27 31 28	167 21 20 39	176 12 21 38	158 30 32 27	158 30 30 29	156 32 33 26	163 25 34 25	F2
				187 0 0 60	165 22 28 32	163 24 29 31	156 31 31 29	160 27 37 23	163 24 27 33	156 31 32 28	153 34 36 24	160 27 37 23	C2
					193 0 0 54	178 15 14 40	159 34 28 26	169 24 28 26	162 31 28 26	161 32 27 27	160 33 29 25	166 27 31 23	C2D2
						192 0 0 55	159 33 28 27	169 23 28 27	161 31 29 26	159 33 29 26	162 30 27 28	164 28 33 22	C2D3
							187 0 0 60	169 18 28 32	160 27 30 30	158 29 30 30	156 31 33 27	167 20 30 30	X2
								197 0 0 50	167 30 23 27	161 36 27 23	159 38 30 20	171 26 26 24	F3
									190 0 0 57	168 22 20 37	164 26 25 32	161 29 36 21	C3
										188 0 0 59	175 13 14 45	155 33 42 17	C3D2
											189 0 0 58	155 34 42 16	C3D3
												197 0 0 50	X3

Table C-25. The probability that observed differences between any two models arose by chance. Calculated according to the McNemar test as applied to the counts of Table C-24. Models likely to differ in accuracy are shaded.

D2	D3	F2	C2	C2D2	C2D3	X2	F3	C3	C3D2	C3D3	X3	
<0.01	<0.01	0.03	0.03	0.01	0.01	0.01	<0.01	0.02	0.06	0.05	<0.01	1
	0.69	0.27	0.25	0.78	0.69	0.21	1.00	0.50	0.35	0.43	1.00	D2
		0.08	0.13	0.50	0.43	0.06	0.87	0.25	0.17	0.24	0.89	D3
			1.00	0.58	0.69	1.00	0.16	0.90	0.90	1.00	0.30	F2
				0.48	0.58	0.90	0.26	0.78	1.00	0.90	0.26	C2
					1.00	0.53	0.68	0.79	0.60	0.70	0.69	C2D2
						0.61	0.58	0.90	0.70	0.79	0.61	C2D3
							0.18	0.79	1.00	0.90	0.20	X2
								0.41	0.31	0.40	0.89	F3
									0.88	1.00	0.46	C3
										1.00	0.36	C3D2
											0.42	C3D3

Table C-26. The McNemar classification counts on the NIST'95 45s language recognition trials using 16 state HMMs (see Table C-19). The four values at the intersection of a given pair of models represent the joint classification counts, as in Table C-1.

1		D2		D3		F2		C2		C2D2		C2D3		X2		F3		C3		C3D2		C3D3		X3		
32	0	31	1	32	0	30	2	31	1	31	1	31	1	31	1	32	0	31	1	30	2	31	1	31	1	1
0	7	5	2	4	3	5	2	5	2	5	2	6	1	3	4	6	1	4	3	5	2	6	1	4	3	
		36	0	35	1	35	1	34	2	34	2	34	2	34	2	35	1	34	2	33	3	34	2	35	1	D2
		0	3	1	2	0	3	2	1	2	1	3	0	0	3	3	0	1	2	2	1	3	0	0	3	
				36	0	34	2	35	1	34	2	34	2	33	3	36	0	35	1	34	2	35	1	35	1	D3
				0	3	1	2	1	2	2	1	3	0	1	2	2	1	0	3	1	2	2	1	0	3	
						35	0	33	2	34	1	34	1	33	2	34	1	33	2	33	2	33	2	34	1	F2
						0	4	3	1	2	2	3	1	1	3	4	0	2	2	2	2	4	0	1	3	
								36	0	34	2	34	2	32	4	36	0	35	1	34	2	36	0	34	2	C2
								0	3	2	1	3	0	2	1	2	1	0	3	1	2	1	2	1	2	
										36	0	36	0	32	4	35	1	33	3	33	3	34	2	33	3	C2D2
										0	3	1	2	2	1	3	0	2	1	2	1	3	0	2	1	
												37	0	32	5	36	1	33	4	34	3	35	2	33	4	C2D3
												0	2	2	0	2	0	2	0	1	1	2	0	2	0	
														34	0	33	1	32	2	31	3	32	2	33	1	X2
														0	5	5	0	3	2	4	1	5	0	2	3	
																38	0	35	3	35	3	37	1	35	3	F3
																0	1	0	1	0	1	0	1	0	1	
																		35	0	34	1	35	0	34	1	C3
																		0	4	1	3	2	2	1	3	
																				35	0	35	0	33	2	C3D2
																				0	4	2	2	2	2	
																						37	0	34	3	C3D3
																						0	2	1	1	
																								35	0	X3
																								0	4	

Table C-27. The probability that observed differences between any two models arose by chance. Calculated according to the McNemar test as applied to the counts of Table C-26. Model F3 differ significantly (97% confidence) from model 1.

D2	D3	F2	C2	C2D2	C2D3	X2	F3	C3	C3D2	C3D3	X3	
0.22	0.13	0.45	0.22	0.22	0.13	0.63	0.03	0.38	0.45	0.13	0.38	1
	1.00	1.00	1.00	1.00	1.00	0.50	0.63	1.00	1.00	1.00	1.00	D2
		1.00	1.00	1.00	1.00	0.63	0.50	1.00	1.00	1.00	1.00	D3
			1.00	1.00	0.63	1.00	0.38	1.00	1.00	0.69	1.00	F2
				1.00	1.00	0.69	0.50	1.00	1.00	1.00	1.00	C2
					1.00	0.69	0.63	1.00	1.00	1.00	1.00	C2D2
						0.45	1.00	0.69	0.63	1.00	0.69	C2D3
							0.22	1	1	0.45	1	X2
								0.25	0.25	1	0.25	F3
									1	0.5	1	C3
										0.5	1	C3D2
											0.63	C3D3