

Confusion Modelling for Lip-Reading

Dominic Liam Howell

A thesis submitted for the
Degree of Doctor of Philosophy



University of East Anglia
School of Computing Sciences

May 2015

©This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with the author and that use of any information derived there from must be in accordance with current UK Copyright Law. In addition, any quotation or extract must include full attribution.

Abstract

Lip-reading is mostly used as a means of communication by people with hearing difficulties. Recent work has explored the automation of this process, with the aim of building a speech recognition system entirely driven by lip movements. However, this work has so far produced poor results because of factors such as high variability of speaker features, difficulties in mapping from visual features to speech sounds, and high co-articulation of visual features.

The motivation for the work in this thesis is inspired by previous work in dysarthric speech recognition [Morales, 2009]. Dysarthric speakers have poor control over their articulators, often leading to a reduced phonemic repertoire. The premise of this thesis is that recognition of the visual speech signal is a similar problem to recognition of dysarthric speech, in that some information about the speech signal has been lost in both cases, and this brings about a systematic pattern of errors in the decoded output.

This work attempts to exploit the systematic nature of these errors by modelling them in the framework of a weighted finite-state transducer cascade. Results indicate that the technique can achieve slightly lower error rates than the conventional approach. In addition, it explores some interesting more general questions for automated lip-reading.

Acknowledgements

I would like to thank everyone who has supported me through my PhD studies. Foremost, I would like to express my sincere gratitude for my supervisor, Prof. Stephen Cox, for his expert guidance over the past four years. His unsurpassed knowledge, patience, and passion for research has been invaluable to my work and career. I would also like to thank my second supervisor, Dr. Barry-John Theobald, for his insightful comments, support, and guidance through my studies. Thank you also to my examiners, Dr. Ben Milner and Dr. Philip Jackson.

I would like to thank all of the staff and students in the School of Computing Sciences at UEA for making my time at university a memorable one. Most importantly, thank you to Jason, Felix, John, and Andrea for the perpetual entertainment.

A special thanks goes to my fiancée, Amy, for putting up with the constant work and stress. Without your love, support, and continued encouragement, this thesis would not have been possible. Finally, my heartfelt gratitude goes to my parents, Anthony and Lorraine, my brother Benjamin, and my grandparents, Victor and Margaret, for their continued, unrivalled support.

Contents

List of Abbreviations	vi
List of Figures	vii
List of Tables	xviii
1 Introduction	1
1.1 Motivation and Aims	1
1.2 Thesis Structure	4
2 Technical Background	7
2.1 Introduction	7
2.2 Visual Feature Extraction	9
2.2.1 Active Appearance Models	9
2.2.2 Post-Processing	12
2.3 Language Modelling	13
2.3.1 Smoothing Techniques	15
2.4 Hidden Markov Models	16
2.4.1 HMMs with Discrete Probability Densities	17
2.4.2 HMMs with Continuous Probability Densities	18
2.4.3 Hidden Markov Models in Speech Recognition	21
2.4.3.1 Monophone HMMs	23
2.4.3.2 Triphone HMMs	24
2.4.4 Networks for Decoding	26
2.5 Confusion Matrices	29
2.5.1 Confusion Matrix Smoothing	30
2.5.1.1 Base Smoothing	31

2.5.1.2	Exponential Smoothing	32
2.6	Weighted Finite-state Transducers	34
2.6.1	Combinatorial Operations	37
2.6.1.1	Composition	37
2.6.1.2	Union, Epsilon Removal, and Closure	38
2.6.2	Pruning Operations	41
2.6.2.1	Determinization	41
2.6.2.2	Minimization	42
3	Literature Review	44
3.1	Introduction	44
3.2	Human Speech Production	45
3.2.1	Phonemes	47
3.2.2	Visemes	48
3.3	Audio-Visual Speech Recognition	49
3.4	Visual-Only Speech Recognition	51
3.5	Dysarthric Speech Recognition	55
3.6	Weighted Finite-state Transducers in ASR	57
3.7	Conclusions	61
4	Description of Datasets	63
4.1	Introduction	63
4.2	ISO-211 - An Isolated Word Dataset	64
4.3	LILiR-200 - A Small Continuous Speech Dataset	71
4.4	RM-3000 - A Large Continuous Speech Dataset	73
4.5	Summary	77
5	Confusion Modelling for Isolated Words	78
5.1	Motivation and Aims	78
5.2	The Standard Approach	81
5.3	The Proposed Approach	84
5.3.1	Use of the top decoding only	85
5.3.2	The Confusion Model	86
5.3.2.1	Timing Offset Classification	94

5.3.3	Use of top n decodings	102
5.3.4	Discussion	109
5.4	Extending the Proposed Approach	112
5.4.1	Adaptive Confusion Model	113
5.4.2	Bigram Confusion Model	118
5.5	Summary	123
6	Lip-reading for Continuous Speech	125
6.1	Motivation and Aims	125
6.2	LILiR Continuous Speech Task	126
6.3	Issues in Modelling for Lip-Reading	129
6.3.1	Viseme mapping and Homophenous Words	129
6.3.2	Experiments	130
6.3.2.1	Analysis of the effect of the language model on viseme decoding	134
6.3.2.2	Analysis of the role of different viseme classes in recognition	135
6.3.2.3	A Data-Driven Phoneme-to-Viseme Mapping	139
6.3.2.4	A Principled Phoneme-to-viseme Mapping	143
6.4	Conclusions	146
7	Confusion Modelling for Continuous Speech	149
7.1	Motivation and Aims	149
7.2	A Monophone System	150
7.2.1	The Proposed Approach	151
7.2.2	Minimising the number of deleted phonemes	153
7.3	A Triphone System	158
7.3.1	The Standard Approach using Triphone HMMs	158
7.3.2	Word-Level Confusion Model	160
7.3.3	The Proposed Approach using Word Hypotheses	166
7.3.4	The Proposed Approach using Lattices	169
7.4	Conclusions	174
8	Conclusions	177
8.1	Discussion	177

8.2	Future Work	180
8.3	Publications	183
A	Isolated Word Vocabulary	184
	References	187

List of Abbreviations

Abbreviation	Meaning
AAM	Active Appearance Model
ASR	Automatic Speech Recognition
AV	Audio-Visual
AVSR	Audio-Visual Speech Recognition
CRF	Conditional Random Field
DCT	Discrete Cosine Transform
DNN	Deep Neural Network
DP	Dynamic Programming
FSA	Finite-state Automaton
FST	Finite-state transducer
GMM	Gaussian Mixture Model
GPDF	Gaussian Probability Density Function
GSF	Grammar Scale Factor
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
LVASR	Large Vocabulary Automatic Speech Recognition
MFCC	Mel-Frequency Cepstral Coefficients
PCA	Principal Component Analysis
PDF	Probability Density Function
PDM	Point Distribution Model
RM	Resource Management
SLF	Standard Lattice Format
WFST	Weighted Finite-state Transducer

List of Figures

1.1	The standard approach to automatic lip-reading with the proposed additional system to clean up the noisy output transcription. The confusion model part of this system is the focus of this project.	4
2.1	The lip-reading recognition task. Features are extracted from the video before the training data is used to build the models for pattern recognition. The testing (unseen) data is passed through the system independently to produce the recognised output.	8
2.2	Example frames taken from a corpus. Frames have been carefully selected to describe extremities in lip movements in an attempt to capture maximum variation. Each selected frame is manually labelled with k points (where k is equal for all landmark frames)	10
2.3	A graphical representation of how the shape and appearance features are computed. A shape (s) can be obtained from a sum of the mean shape (\bar{s}) and the weighted modes of variation in shape (e.g. lip rounding, mouth opening etc). Similarly, an appearance image (a) can be computed as the sum of the mean appearance (\bar{a}) in the shape-free patch with the weighted modes of appearance variation in the model.	11
2.4	A three-state HMM with discrete probability densities. In this example, there are three possible observations: a, b and c, each with likelihoods associated in all three states.	18
2.5	An example of a Gaussian PDF centered around the mean denoted as μ with a standard deviation, σ . Likelihoods can be produced for a given input feature vector to describe the fit to the GPDF.	19
2.6	An illustration of a Gaussian Mixture Model (GMM). In this example, there are 3 mixture components that form a resulting distribution that is non-Gaussian. Each GPDF has its own mean (μ) and variance (σ).	20

- 2.7 Words can be modelled as network of HMMs. Here, the word *hello* is decomposed into phoneme units and represented as a network of sub-HMMs. Segmented acoustic features can be passed through this network and output the likelihood that the input features came from the set of models. The probabilities between the HMMs relate to the language model likelihoods. 21
- 2.8 An example of both monophone and triphone transcriptions for the phrase “*Hello Sir*”. For monophones, a simple look-up dictionary is required to convert words into phoneme sequences. The *sil* label denotes sentence termination whilst the *sp* label is used to divide phoneme sequences into words. In the triphone case, a cross-word example is shown where boundaries are isolated as monophones but triphones span over the inter-word boundaries. 24
- 2.9 An illustrated example of a phonetic decision tree. Each node stores a question pertaining to the context surrounding the subject phoneme. ‘L’ and ‘R’ refer to the left and right context respectively. Questions can be wide in their scope or channelled down to identifying specific phonemes. 26
- 2.10 The number of deleted and inserted phonemes in the hypothesised sequence as a function of the insertion penalty. A smaller insertion penalty attracts less insertions but more deletions whereas a larger insertion penalty causes more inserted tokens but less deletions. . . . 27
- 2.11 A representation of a decoding lattice for a simple set of sentences. In this example, nodes represent whole words (these can either be modelled as a set of sub-word unit HMMs or whole word HMMs). Edges connect the nodes where each edge has an acoustic likelihood (a) and a language model likelihood (l) associated with the transition. 28
- 2.12 An example of a typical confusion matrix for five classes (letters ‘a’ to ‘e’). The hypothesised and ground-truth sequences are matched together to form a set of confusions. Substitutions are modelled by the 5x5 square section in the middle with rows representing the ground-truth (input) and columns representing the hypothesised symbol (response). Insertions and deletions are represented by an additional row and column. Each entry in this confusion matrix represents a frequency count for the given confusion. 30
- 2.13 Examples of the effects of base smoothing using three different smoothing parameter values (d): 0.25, 0.5, and 0.75. Off-diagonal confusions become stronger and the diagonal is weakened as the distribution parameter is increased. 32

2.14	Exponential smoothing examples using three different values for α : 0.01, 0.1, and 1. Smaller α values re-distribute more of the larger probability mass across the row whereas a much larger α value concentrates the probability on the stronger (i.e. the diagonal) elements.	33
2.15	A example of a weighted finite-state transducer over the tropical semiring. This transducer has one initial state (0) and one final state (3) and only allows one path translating the string 'abc' to 'xyz' with a total path weight under the tropical semiring of $1.2 + 3.2 + 3.3 = 7.7$.	36
2.16	(a) T_1 is a weighted transducer with an output symbol domain that matches the input domain of T_2 , (b) T_2 is a weighted transducer with an input symbol domain that matches the output domain of T_1 , and (c) $T_1 \circ T_2$ is the resulting weighted transducer after T_1 is composed (o) with T_2 .	38
2.17	(a) T_1 is a single-path weighted transducers, (b) T_2 is an alternative single-path weighted transducer, (c) T_3 is an alternative single-path weighted transducer, and (d) $T_1 \cup T_2 \cup T_3$ is the resulting weighted transducer after performing the union operation on T_1 , T_2 , and T_3 .	39
2.18	ϵ -removal of the transducer $T_1 \cup T_2 \cup T_3$ pictured in Figure 2.17(d). Here, epsilon transitions (represented by the '-' symbol in Figure 2.17(d)) exiting the start state (0) from the original transducer are removed, ensuring maximum traversal efficiency when it is required. For much larger transducers, ϵ -removal can significantly improve composition and shortest-path operational efficiency.	40
2.19	A closed weighted transducer derived from the original topology shown in Figure 2.18. The closure operation has introduced epsilon arcs that can be used as a <i>free</i> transition back to the start of the sequence.	40
2.20	(a) a WFST (T) over the tropical semiring, and (b) T_{det} - a determinized version of weighted transducer T that provides equivalent input and output paths and transductions. From the determinization algorithm, residual weights are carried forward through to ensure original path costs from T are maintained.	42
2.21	(a) a WFST, T , defined over the tropical semiring, and (b) T_{min} - a minimized equivalent of transducer T .	43
3.1	A diagram describing the positions of organs in the human speech production system.	45
3.2	A diagram describing the components in an audio-visual speech recognition system. Feature extraction of the audio and visual signal is performed independently before using a fusion method to combine the feature sets to aid recognition.	50

3.3	Speaker-dependent phone accuracy results for audio and visual speech recognition using different combinations of articulatory features. Each point along the x axis represents a cumulative removal of the information that is obtained from the specified articulator [Newman et al., 2010].	52
3.4	P^* - the phoneme-level hypothesis produced by the state-of-the-art ASR system. This symbol sequence is most likely to contain incorrect (noisy) entries that can be corrected in the cascade.	58
3.5	C performs the substitutions, insertions and deletions according to the confusion patterns in the estimated confusion matrix. In this example, most translations are taken from the diagonal elements of the confusion matrix ('null' substitutions). Off-diagonal substitutions (recognising $/w/$ instead of $/y/$, an insertion of $/ax/$ and a deletion of $/ih/$) are also modelled here.	59
3.6	L - the lexicon WFST maps sequences of ground-truth phonemes to whole words. This example models a five-word vocabulary — $\{HELLO, HOW, ARE, YOU, WHO\}$. The input alphabet is a set of phoneme strings whilst output alphabet on the final arc of each path are words. In most cases, words are treated as equally weighted entities in a non-weighted transducer.	59
3.7	G - a sentence-level language model can take the form of an n -gram model (described in Section 2.3). This example is a unigram (1-gram) language model, allowing any word to follow any other word with an associated cost (defined over the tropical semiring).	60
4.1	ss from the BEEP dictionary [Cambridge-University, 2012] with the percentage of the dictionary covered.	65
4.2	A comparison of number of bigram occurrences covered. (a) shows the bigram frequencies taken from the BEEP dictionary (257,059 words) and (b) shows the bigram frequencies from the isolated word data set consisting of 211 words.	66
4.3	An illustration describing the setup used for recording the single-speaker isolated word dataset. The controlled environment was achieved with the use of a spot light against a reflective surface. The subject was sat upright in a chair with the video camera focussed into the whole face and turned into portrait mode.	68
4.4	An example frame from the isolated-word dataset consisting 6 repetitions of 211 words by a female speaker. Landmarks are hand-labelled on 20 to 30 images on the face to aid tracking. Points on other parts of the face other than the lips are discarded for feature extraction. . .	70

4.5	An example frame from the LiLR-200 dataset consisting of one speaker reciting 200 sentences from the Resource Management corpus. Landmarks are hand-labelled on a set of training frames before tracking is performed to obtain landmarks for a video sequence. Points that are only required on the face for tracking purposes are removed before feature extraction to leave landmarks on the inner and outer lip contours.	72
4.6	An example frame from a recording of the 3000-sentence RM database. A total of 93 points are labelled on the face to improve accuracy of the tracking procedure. After tracking, only inner (12 points) and outer (16) points are retained for feature extraction.	75
4.7	Frequency counts of each phoneme observed in the RM-3000 audio-visual speech dataset. Consonants /t/ and /n/ and the /ax/ vowel are used significantly more than the rest of the phonemes due to their popularity amongst the smaller, more frequently occurring words. . .	76
5.1	An illustration of the components in the two approaches to automated lip-reading: the standard approach — decoding using a network of trained HMMs as used in previous work, and the proposed approach — an almost identical recogniser system with an additional confusion module to correct the noisy transcriptions produced by the standard approach recogniser.	80
5.2	Phoneme recognition results on the isolated word dataset using a network of phoneme HMMs - a technique considered to be state-of-the-art in automated lip-reading. A phoneme bigram language model is used to improve recognition accuracy.	83
5.3	Word recognition results on the isolated word dataset using the standard approach to automated lip-reading (HMMs). This result provides the optimal baseline recognition accuracy for the isolated word task.	83
5.4	A standard ASR system has decoded a phoneme sequence that has been used to construct a sequential P^* WFST. The arc weights are calculated using the negative log likelihood for a specific symbol that has been produced by the standard recogniser.	86
5.5	An illustration of the two approaches to populating a phoneme confusion matrix. Top shows the method of decoding one out of the 211 words from the vocabulary. The decoder is forced to take one of the valid 211 paths through the network of phoneme HMMs and produces the one with the highest likelihood. Bottom illustrates an alternative approach, using a phoneme bigram language model to guide the decoder to bigrams with a larger probability.	87

- 5.6 An alignment between the ground-truth and recognised sequences using dynamic programming for the phonetic transcription of the word *different*. Counts in a confusion matrix can be populated using the substitutions (Sub.), insertions (Ins.), and deletions (Del.) observed from the alignment. 88
- 5.7 A confusion matrix that has been estimated from the phoneme alignment in Figure 5.6. The input rows represent the ground-truth phonemes and the response columns represent the decoded phonemes. The counts in the matrix represent the frequency of each confusion between a ground-truth and decoded phoneme with an additional row for insertions and column for deletions. 88
- 5.8 A comparison of the confusion patterns that can be observed when performing recognition using the standard approach to automated lip-reading. The two methods both require phoneme HMMs to be trained in the normal procedure. However, when performing recognition, words can be decoded using the HMM likelihoods and a phoneme-to-word dictionary or a simple phoneme bigram language model can be used to output a phoneme string. 90
- 5.9 A cyclic confusion weighted finite-state transducer to correct the hypothesised sequence produced by the recogniser in Figure 5.6. The deletion of the phoneme *ax* in the hypothesised sequence is modelled in the confusion transducer using the epsilon symbol (ϵ) to reverse the error and insert a phoneme into the hypothesised sequence. This epsilon symbol is reserved to allow ‘free’ transitions between states and is used to model both insertions and deletions. 91
- 5.10 Word-level recognition results using weighted finite-state transducers to correct noisy output transcriptions. The confusion model is trained using the output confusion matrix from the HMM recognition. . . . 93
- 5.11 An example alignment between the ground-truth and recognised phoneme sequences for the word “*bath-chair*”. 95
- 5.12 A time alignment example for the word “*bath-chair*” taken from the isolated word dataset. The ground-truth alignments are captured using a forced alignment procedure using the audio and video features individually, and the recognised alignment is the output sequence that has been produced by the standard HMM approach. In this example, although the symbolic alignment would align *ih* with *aa* and *ch* with *ch*, their respective starting points are poorly synchronised and could be considered as spurious confusions. 95

5.13	An illustration of how the timing offset window is enforced for each ground-truth phoneme against its aligned decoded phoneme. The windows (shown as double-arrowed lines above the transcripts) are centered from the ground-truth phoneme and defined as $\pm x$ standard deviations away from the mean offset for the ground-truth phoneme. .	96
5.14	Analysis of the number of confusion patterns that are accepted as a function of the timing window. Error bars are not shown here because they are too small.	98
5.15	Word recognition results using different timing offset windows to restrict the confusion patterns that are used to populate the confusion matrix. The best accuracy that was achieved without the timing window (shown in the results from Figure 5.10) is shown as a dotted line. The input P^* WFST is modelled using the top decoding that is produced by the standard approach phoneme recogniser (described in Section 5.3.1)	101
5.16	Example of an input P^* weighted finite-state transducer used as input to the recognition cascade. This noisy transcription is produced by the standard approach for the ground-truth word ‘Overachieve’. To build this transducer, 15-best transcriptions were recognised using the standard approach and joined using the union of WFSTs to form a network	104
5.17	Recognition results when using the original union method for the input P^* transducer	105
5.18	An alternative approach to building the P^* transducer is shown here, modelling the noisy transcription from the standard approach for the ground-truth word ‘Overachieve’. This method attempts to find the best alignment between the 9-best recognised transcriptions using dynamic programming alignment. Once the best alignment is found, the transducer is built as shown.	107
5.19	Recognition results when using the adapted version for the input P^* transducer. All n transcriptions are warped together to form the input transducer.	108
5.20	A description of the adaptive confusion model. Confusion probabilities are updated in an iterative process by running the testing set through the standard approach recogniser and the WFST cascade. Computed errors are then used to update the confusion probabilities before the process is repeated. The iterative process continues until the total error starts to increase — representing a point at where the recognition accuracy can not improve any more.	114

5.21	Word recognition results using adaptive confusion modelling. The configuration that achieved the best results in Section 5.3 was used in an iterative training process. These results were achieved using a learning rate (α) of 0.1 — putting a strong influence on the prior probabilities with the new information obtained by the error confusion matrix only contributing 10% of the overall weighting.	116
5.22	An illustration of a bigram confusion model with backoff weights. The vocabulary consists of three symbols: a , b , and c . The unigram backoff arcs are derived from the unigram confusion matrix containing nine entries. Bigram arcs are added to transition out of the start state, to an isolated state and then back to the start state. Only one arc in the bigram sequence requires a weight, defined as the negative logarithm of the bigram probability. The backoff weight, β is applied to all unigram probabilities to determine the influence of the backoff weights. A factor of $(1 - \beta)$ is also applied to the bigram probabilities.	120
5.23	Word-level results for the isolated word dataset using a new bigram confusion model. Results are derived from cross-fold validation over six repetitions. These experiments have been conducted using the best confusion matrix smoothing method from the results presented in Section 5.3.	122
6.1	Phoneme-level recognition results on the LILiR dataset using a network of phoneme HMMs - a technique considered to be state-of-the-art in automated lip-reading. A phoneme bigram language model is used to improve recognition accuracy.	128
6.2	Word-level recognition results on the LILiR dataset using the standard approach to automated lip-reading (HMMs).	128
6.3	Unit recognition performance on 3000 speaker-dependent sentences from the Resource Management Corpus (RM). Phoneme and viseme (unit) accuracies are shown for audio and visual speech recognition as a function of the size of the training set.	132
6.4	Word recognition performance on 3000 speaker-dependent sentences from the Resource Management Corpus (RM). Word recognition results are shown for audio and visual speech recognition using both phoneme and viseme models as a function of the size of the training set.	133
6.5	The effect of the language model on word accuracy when the recogniser is given “perfect” features (i.e. ground-truth features generated by the trained HMMs). With a grammar scale factor of zero, the bigram word-pairings are preserved but each has equal probability. Thereafter, the bigram language model has an increasing influence. .	135

6.6	Unit and word recognition accuracies that are achieved for forming an isolated viseme from the set of visemes in the Fisher mapping (shown in Table 6.1). Each viseme group is formed in turn whilst keeping the other classes remain as phonemes to see if unit or word accuracy could be improved using a partial phoneme-to-viseme mapping. . . .	136
6.7	Average unit and word accuracies for the different viseme class sizes. The Fisher phoneme-to-viseme mapping does not have any classes that consist of five or seven phonemes.	137
6.8	The effect on recognition accuracy of progressively mapping phonemes to visemes. Each viseme class is made in turn, reducing the number of classes progressively from 45 to 14. ALL represents using all 45 phoneme classes.	138
6.9	Distribution of Fisher visemes over RM-3000 dataset	139
6.10	Unit accuracy (top) and word accuracy (bottom) recognition results at each of the 38 grouping steps described in Table 6.2. Each step is dependent on the previous mapping whereby the phonemes that have been assigned to viseme groups previously are maintained and a new grouping is introduced according to the steps in Table 6.2.	142
6.11	Unit accuracy (top) and word accuracy (bottom) recognition results for the data-driven phoneme-to-viseme mapping. Here, the final viseme classes (shown in Table 6.3) that are produced by the mapping steps shown in Table 6.2 are explored. Standard unit and word recognition is performed on the viseme classes independently (only a single viseme class is constructed and all other phonemes remain unmapped).	143
6.12	Recognition results using the principled viseme mappings. The 17 mappings are shown in Table 6.4. Here, each viseme class is constructed independently at each step described in Table 6.4. During each mapping step, all other phonemes outside of the subject viseme class are maintained as separate classes. Standard error is too small to be shown for these results.	144
7.1	Five examples of phoneme transcriptions produced by the phoneme recogniser with the best phoneme accuracy. ‘SENTENCE’ gives the word transcription of the sentence, ‘GT’ is the ground-truth phoneme sequence, and ‘REC’ is the recognised phoneme sequence.	152
7.2	The number of consecutive deletions and insertions that are produced by the standard phoneme recogniser. The length of each deleted/inserted sequence is recorded and represented in a histogram plot. The number of deletions is much larger than the number of insertions as described in Table 7.1	153

- 7.3 The top plot shows the number of inserted and deleted phonemes as a function of insertion penalty values in the HTK standard approach recogniser. Below this is a phoneme accuracy plot as a function of the same insertion penalty. This figure is used to find the optimal insertion penalty to minimise the number of deletions whilst also maximising the phoneme accuracy. 154
- 7.4 Five examples of phoneme transcriptions produced by the standard approach recogniser with the reduced number of deletions. ‘SENTENCE’ gives the word transcription of the sentence, ‘GT’ is the ground-truth phoneme sequence, ‘OPT’ is the new recognised phoneme sequence with fewer deletions, and ‘ORIG’ is the recognised phoneme sequence produced by the phoneme recogniser with the best accuracy (Figure 7.1). 156
- 7.5 A comparison between the number of consecutive deletion/insertion sequences in the two hypotheses, using the best phoneme accuracy and using the decoded output that is optimised to minimise deletions. The top histogram plots the consecutive deletion counts and the bottom histogram shows the consecutive insertion counts 157
- 7.6 Word recognition results using the standard approach to lip-reading. As the RM-3000 dataset provides a richer training set, the standard monophone HMM results presented in Figure 6.4 can be improved by using a triphone HMM system where an HMM models a phoneme in a specific left and right context. 159
- 7.7 Five examples of word transcriptions produced by a monophone HMM standard approach system and a triphone HMM standard approach system. ‘GT’ is the ground-truth sentence, ‘MONO’ is the recognised sentence produced by the monophone HMM approach, and ‘TRI’ is the recognised sentence produced by the triphone HMM approach. . . 161
- 7.8 A simple word-level confusion model example derived from the confusions that are observed in Table 7.5. Weights are derived from the ratio of the specific confusion (i.e. the translation between the ground-truth and recognised word) to the total number of confusions for the given ground-truth word. 164
- 7.9 Five example P^{W*} WFSTs which are constructed from the decoded word sequences that are produced by the standard triphone HMM recogniser. These WFSTs are used as the entry-point to a WFST cascade using word-level confusion patterns. 165
- 7.10 Showing how the standard approach triphone system is used with an additional confusion model to correct phoneme strings. The word output is decomposed into a phoneme string (with short pauses at word boundaries) which is then used in the confusion WFST cascade. 166

- 7.11 An example of a conversion from a simple recognition lattice to a WFST. The difference between the two paradigms lies with the location of the subject symbol. In a recognition lattice, nodes represent the symbols in the network which are joined using transition likelihoods, whereas in the WFST, only arcs store symbols together with weights. To apply this conversion, a single state from the recognition lattice is converted to a mini WFST with two states and a single (unweighted) transition. Weighted transitions are then used to connect these small WFSTs together. 171
- 7.12 (a) is a simple decoding lattice that has been produced by the standard approach recogniser using HTK, and (b) is the equivalent phoneme-level WFST for the lattice shown in (a). 173

List of Tables

2.1	A selection of widely used semirings and their binary relations [Mohri, 2004]. \oplus and \otimes denote the binary operations and $\bar{0}$ and $\bar{1}$ represent the identity elements provided by the two monoids respectively. The definition of the binary operations (\oplus and \otimes) allows the flexibility to apply these WFSTs to the four weighting systems described in this table. Combinatorial and pruning operations (described in Sections 2.6.1 and 2.6.2 respectively) can then be defined on the definition of the binary relations.	34
3.1	A list of the different places of articulation in production of consonant sounds. Letters highlighted in bold indicate which sound has a specific place of articulation within a word.	46
3.2	Configurations of different manners of articulation shown with examples.	47
4.1	A description of the specially-recorded audio-visual isolated word dataset. Six repetitions were recorded from one female speaker reciting 211 carefully-selected words. Recordings spanned over three separate sessions, lasting between 20 to 30 minutes each. Each video was down-sampled from 1920 x 1080 pixels to 640 x 360 pixels to improve tracking and feature extraction computational efficiency.	69
4.2	Statistics from the captured data taken from the selected subset of the Resource Management (RM) corpus.	74
5.1	Word recognition accuracies for the two approaches used in this work so far. The ‘Standard Approach HMM System’ uses the standard approach to lip-reading (using a network of HMMs), and ‘WFST Confusion System with confusion from a symbolic alignment’ uses a WFST cascade with a confusion model that has been trained using the confusion patterns that have been produced from the standard approach phoneme recogniser.	94

5.2	The word recognition performance of three techniques used so far in this work. The ‘Standard Approach HMM System’ uses a network of phoneme-level HMMs formed to recognise one word from the 211-word vocabulary, the ‘WFST Confusion System with confusion from a symbolic alignment’ system uses the confusion matrix produced by the standard HMM phoneme recogniser in the WFST confusion cascade, and ‘WFST Confusion System with timing confusion matrix and top decoding only and base smoothing’ presents the best result using the timing offset classification algorithm and base smoothing. .	101
5.3	A comparison of the word recognition results achieved with the systems used so far.	105
5.4	A comparison of the word recognition results achieved with the systems used so far.	109
5.5	The list of parameters and the values tested in the set of confusion WFST experiments.	110
5.6	A contingency table to provide a comparison between the two approaches. ‘HMM’ denotes the standard approach using a network of phoneme-level HMMs, and ‘WFST’ represents our new approach using a confusion model in a WFST cascade with the decoded phoneme strings that are produced by a standard approach phoneme recogniser.	112
5.7	A summary of the word-level recognition results using different methods. Word accuracy and standard deviation are both reported.	112
5.8	Bigram occurrence statistics in the isolated word dataset. Phoneme pairs (bigrams) are used to populate a bigram confusion matrix before being used to build a bigram confusion model. Figures represent an average over the six folds.	121
5.9	A contingency table to provide a comparison between the HMM and WFST approaches. The ‘HMM’ approach consists of a standard phoneme HMM recogniser, and the ‘WFST’ approach is our system using the confusion WFST cascade using a <i>bigram</i> confusion model. .	122
5.10	All word-level recognition results obtained during the set of experiments in the isolated word task	124
6.1	Description of the Fisher phoneme-to-viseme mappings to collapse 45 phoneme classes into 14 viseme classes. A viseme is reserved for the silence model (/sil/)	130
6.2	Algorithm steps in the data-driven phoneme-to-viseme mapping algorithm. The final viseme class set is shown in Table 6.3.	141

6.3	A description of the viseme classes that are produced by the data-driven phoneme-to-viseme mapping algorithm. Here, 44 phonemes are collapsed into six viseme classes. An additional silence viseme is also used as the seventh viseme class.	142
6.4	Description of the principled phoneme-to-viseme mappings. Each mapping referred to using a ‘Mapping ID’ which corresponds to the recognition results shown in Figure 6.12. The total number of classes is calculated as the sum of number of viseme classes and the number of remaining phoneme classes (i.e. phonemes that are not assigned to a viseme class in the mapping).	145
6.5	Comparison of the best unit and word accuracies that are produced by the phoneme-to-viseme mappings used in this work.	147
7.1	Phoneme Recognition statistics produced by the monophone standard approach using a phoneme bigram language model.	153
7.2	Phoneme Recognition statistics produced by the monophone standard approach using a phoneme bigram language model.	155
7.3	Best word-level results produced by the proposed approach using phoneme transcriptions with minimal deletions. A search was performed to find the optimal timing offset window of ± 0.5 standard deviations.	157
7.4	Comparison between the word recognition statistics achieved with a monophone and triphone HMM system.	160
7.5	Example of a set of confusions that have been identified by the training process. Here, two ground-truth words are confused with different, visually similar words. The frequency counts of these confusions are used to derive the weights for the confusion model (pictured in Figure 7.8).	162
7.6	Comparison between the word recognition accuracies achieved with a monophone and triphone HMM system.	163
7.7	Comparison of the word recognition statistics between the standard approach triphone system and the WFST confusion modelling system using the triphone decodings.	168
7.8	Decodings are cross-matched to construct this 2x2 contingency table to compare the standard approach (HMM) with the proposed confusion system (WFST). The entries in this table are counts of correctly/incorrectly decoded words.	169
7.9	Comparison of the word recognition statistics between the WFST proposed confusion system presented in Section 7.3.3 and the WFST confusion system using decoding lattices.	172

7.10 A summary of all word recognition results for standard approach systems and all WFST confusion systems.	175
--	-----

Chapter 1

Introduction

1.1 Motivation and Aims

Lip-reading uses information obtained from the movement of the lips and the positions of the visual articulators to provide a transcription of a spoken utterance. Although this is predominately used as a method of communication by people with hearing difficulties, some degree of visual information is used subconsciously by people with normal hearing as a method of verifying their interpretation of a spoken utterance especially when the signal-to-noise ratio (SNR) is low. It is fair to assume that the information in speech is more dominant in the audio. However, people with hearing difficulties are able to successfully interpret spoken language with visual-only cues.

There are many applications of automated lip-reading. One is to enhance auditory information in conventional automated speech recognition (ASR) systems [Luettin and Thacker, 1997; Petajan, 1984; Potamianos et al., 2003a]. This proves to be vital when the acoustic conditions deteriorate (for example, when the acoustic speech signal is severely degraded by noise). Other applications include camera-based security systems, medical applications with hearing impaired patients and recovering speech from a mute or noisy video. Visual speech also provides informa-

tion on the position of the visible articulators (the tongue, teeth and lips). This can provide disambiguation between phonemes such as /p/ and /k/ (unvoiced consonants) and /b/ and /d/ (voiced consonants) [Potamianos et al., 2003b]. Lip-reading can also be used as a standalone recognition method to use if the audio track is not available.

Phonemes are abstract signalling units that discriminate words. In audio speech, a single phoneme encompasses a set of speech sounds that are interpreted to be the same signal. Their visual counterparts have been called *visemes*, which are considered as the smallest units of visual speech. Although these can provide discrimination between words in the visual domain, the relationship between visemes and phonemes is not considered to be one-to-one [Chen and Rao, 1998]. Visemes are very loosely defined because of the intra-speaker variation of phonemic sounds produced on the lips. [Cox et al., 2008] discusses the difficulties surrounding multi-speaker lip-reading, concluding from experiments in near-ideal conditions that the variability in visual-only recognition is much greater than that of auditory speech recognition. This is due to the wide variability in speaker features from one speaker to another, despite the effort to remove this by normalisation.

There are two approaches to build ASR systems: *speaker-dependent* and *speaker-independent*. In a speaker-dependent system, the classifier is trained on data from a single speaker before being tested on unseen data from the same speaker. This provides the model with previous knowledge of the speaker before recognition. Conversely, a speaker-independent system uses several speakers' data to train the classifier. In the testing phase, a data set (with different speakers) is used. Although speaker-independence is an ideal for most ASR systems, a trade-off exists between these two methods with the simplification of the task (e.g. the vocabulary size covered). A compromise between *speaker-dependent* and *speaker-independent* recognition is *multi-speaker* recognition. This uses the same speakers to train the classifier and test but uses different speech utterances for each phase. This method has been widely used within visual-only speech recognition [Cox et al., 2008; Matthews et al.,

2002]. Unfortunately, the relative success of this approach has made researchers ignore the fact that accuracy on speaker-independent testing is often very poor. This problem has only recently been addressed [Cox et al., 2008].

Recent work in visual-only recognition has yielded few positive results [Theobald et al., 2006; Lan et al., 2010; Cox et al., 2008], with most work concentrated on rather simple tasks with small vocabularies. The most difficult task presented to date [Lan et al., 2010], used a database consisting of 12 speakers reciting 200 sentences from the Resource Management Corpus [Price et al., 1988]. Best viseme accuracy was about 50% for a multi-speaker system and approximately 43% for a speaker-independent system. However, the relative word accuracy of the systems is much lower (about 10%).

The performance of an ASR system can be viewed as a perfect classifier operating on a signal that contains varying degrees of *noise*. This noise is much more prominent in automated lip-reading than in standard audio ASR, introducing more erroneous hypotheses. These recognition errors can be regarded as *confusions* between the hypothesised sequence and the “ground-truth” (i.e. correct) sequence. *Confusions* are of three kinds: *substitutions*, where one symbol is replaced by another, *insertions*, where one or more symbols are inserted into the ground-truth sequence, and *deletions* where a symbol is removed from a sequence at a certain index.

Visual speech has an interesting relationship to dysarthric speech. Dysarthric speakers have poor control over their articulators because of medical conditions that affect their motor functions (such as cystic fibrosis, stroke etc.). This leads to a phonemic repertoire that is both reduced and distorted, and hence to speech that has low intelligibility, and is difficult to recognise. There is an obvious parallel with visual speech, where certain sounds cannot be distinguished visually because they differ only in a feature that is not available in the visual signal (e.g. voicing). In previous work on dysarthric speech recognition, patterns of phonemic confusions made by a particular talker were learnt by the system. When these confusions were compensated at recognition time, a statistically significant gain in accuracy was achieved

of up to 15% [Morales, 2009]. In this work, we take a similar approach to lip-reading: we model visual speech as if it were a speech signal produced by a speaker who has a limited phonemic repertoire, and learn the patterns of confusion by comparing the ground-truth sequences and the recognised sequences. At recognition time, we find the most likely interpretation of a reduced/distorted phoneme output sequence in the light of these patterns, as was successfully explored in [Morales and Cox, 2008].

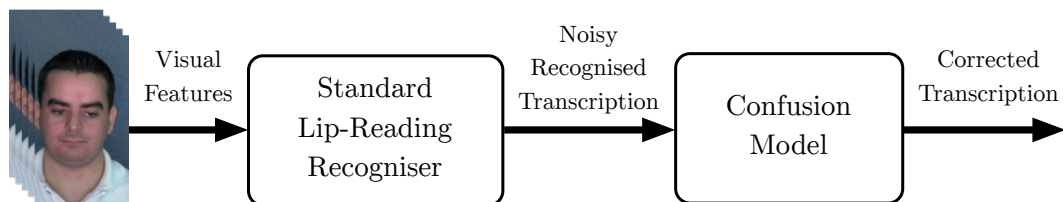


Figure 1.1: The standard approach to automatic lip-reading with the proposed additional system to clean up the noisy output transcription. The confusion model part of this system is the focus of this project.

This work presents an investigation into modelling confusion patterns to improve the recognition accuracy of automated lip-reading. The proposed system, which is broadly illustrated in Figure 1.1, is an additional component added to the existing *standard approach*. This additional module, namely the confusion model, aims to improve recognition accuracy by correcting sounds that have been confused in the standard recognition process.

1.2 Thesis Structure

Chapter 2 provides technical descriptions of the techniques used throughout this work. It covers a range of existing methods in computer vision, speech recognition, and finite-state automata theory. The Chapter starts with a description of the feature extraction process using active shape and active appearance models (AAMs). It then describes the dominant pattern recognition tool that has been used in automated speech recognition (ASR) for over 30 years — Hidden Markov Models (HMMs). This chapter also provides an overview of language modelling and confusion matrices. Finally, this chapter describes weighted finite-state transducers

(WFSTs), a tool for symbol translation that has been used in previous work to implement ASR systems.

Chapter 3 reviews the current literature in the related areas of work. Firstly, it presents a brief description of the human speech production system with definitions for the auditory unit of speech, phonemes, and the visual unit of speech, visemes. This chapter describes current state-of-the-art techniques that are used in audio-visual ASR and visual-only ASR (also known as lip-reading). Motivation for the work in this thesis is described in more detail with the recent work in confusion modelling for dysarthric speech recognition, and finally, this chapter describes the use of weighted finite-state transducers in state-of-the-art speech recognisers.

Two new datasets recorded specially for these experiments are described in Chapter 4. The first new dataset consists of isolated words and is used for the preliminary work in confusion modelling which is described in Chapter 5. The second new dataset is motivated from the work conducted in Chapter 5 and is used in Chapters 6 and 7 for work on continuous speech. We also describe an existing dataset that has been used for lip-reading experiments in previous work.

The first set of experiments are presented in Chapter 5. Here, we use the isolated word database described in Chapter 4 to focus on accurately estimating a phoneme confusion matrix for a speaker. We construct a confusion system using a network of weighted finite-state transducers with a pre-trained (offline) confusion model and extend our technique further by presenting a method for classifying confusion patterns as ‘spurious’ or ‘genuine’. We also explore the use of using multiple hypotheses as the input to our confusion modelling system and find the best way to model these as a weighted finite-state transducer. Finally, we extend our approach, firstly with a system in which the confusion weights are iteratively updated in an adaptive process, and finally by adding context to the confusion information (i.e. modelling the phoneme in the context of the preceding phoneme).

The limitations of the isolated word task are discussed at the end of Chapter 5. We extend our system for the continuous speech task in Chapter 6 by using

two datasets: an existing audio-visual database that has been used in previous automated lip-reading literature [Lan et al., 2010], and a new, larger dataset that has been specially recorded for the purpose of this work. With this larger dataset, we answer two important questions for automated lip-reading. Firstly, how much training data is required to achieve peak recognition accuracy? Secondly, is it better to use phoneme or viseme units for automated lip-reading. We also explore a number of phoneme-to-viseme mappings, identifying any that provide an increase in unit or word recognition accuracy.

Our final body of work explores the use of our confusion modelling techniques for continuous speech in Chapter 7. We apply the techniques that are presented in Chapter 5 to our new, large database and explore additional standard approach techniques that require a large amount of training data. Finally, we use a *recognition lattice* as the input to our confusion model system instead of the multiple hypothesis list that was used previously.

Chapter 8 concludes this thesis by summarising and discussing the results. We also discuss future work to be explored in confusion modelling for lip-reading and we comment on the limitations of this work.

Chapter 2

Technical Background

2.1 Introduction

Lip-reading is the process of recognising speech using only information from the visible articulators. Previous work in automated lip-reading has followed standard approaches to audio speech recognition where the speech signal is represented as a sequence of feature vectors that are passed through a classifier to produce a recognised output. Figure 2.1 illustrates this approach. Here, a set of *training data* are used to optimise the parameters of the classifier (i.e. to train it). At recognition time, the *testing data* are passed through the system and a *decoded sequence* of phonemes or words is produced.

This chapter will discuss the various techniques used this thesis, including techniques from speech recognition, pattern recognition, computer vision and finite-state automata theory. Firstly, in Section 2.2, we discuss the visual feature extraction process where visual speech is represented as a sequence of observation vectors at a rate of a single feature vector per video frame.

Once the frame-wise features are extracted, the models can be trained for pattern recognition. In audio speech recognition, *hidden Markov models* (HMMs) are most commonly used [Rabiner, 1989]. Models can be trained on the smallest units

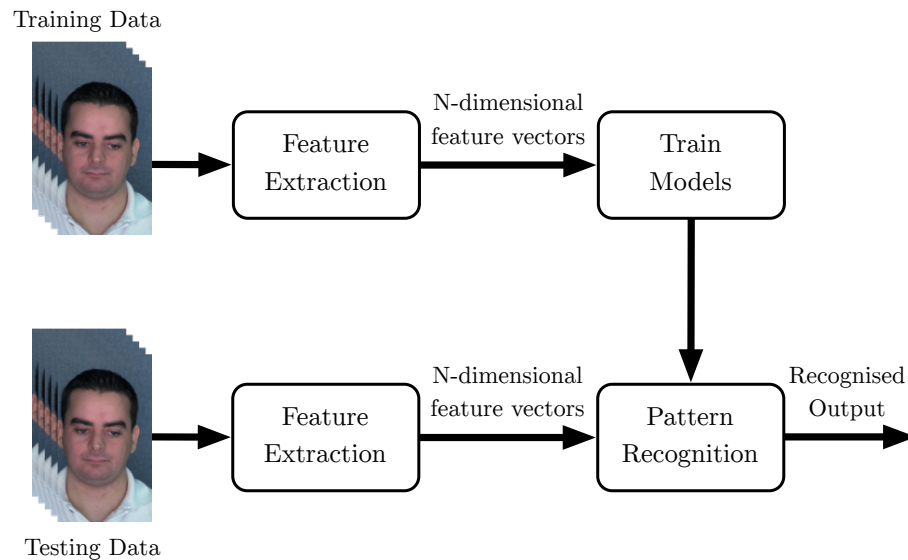


Figure 2.1: The lip-reading recognition task. Features are extracted from the video before the training data is used to build the models for pattern recognition. The testing (unseen) data is passed through the system independently to produce the recognised output.

of speech (phonemes) to produce the most likely phoneme sequence. Section 2.4 describes *hidden Markov models* and their particular application in speech recognition. Section 2.3 describes the language constraint techniques that can be applied to improve recognition. n -gram language models can be used to boost likelihood of sequences (of n tokens) that have been observed frequently in the training data whilst also accounting for rare or unseen events at training.

Confusion matrices can be used to evaluate the performance of an ASR system based on patterns of substitutions, insertions and deletions. Section 2.5 describes the structure of a confusion matrix and the *smoothing techniques* used to re-distribute the probability mass over unseen events.

Finally, *weighted finite-state transducers* (WFSTs) are described in Section 2.6. As a derivative of the widely-used finite-state automaton, weighted finite-state transducers can be applied as a method of statistical pattern recognition to translate symbols with an associated weight — an attribute that is of particular interest when modelling confusion patterns.

2.2 Visual Feature Extraction

The audio and visual speech signal can be represented as a time-series sequence of events at a certain sampling frequency. In the United Kingdom, the frequency standard for recording video is 25 frames per second. Smoother sequences can be achieved by using specialised video cameras that are able to capture at a higher frequency. Most audio equipment is capable of sampling a much higher frequency of 48,000 samples per second (48kHz). However, for most applications, this sampling rate is too high for audio speech recognition, so the audio is usually downsampled.

The process of extracting visual speech features from a video frame has been the topic of much work in visual speech processing [Matthews et al., 2002; Hong et al., 2006; Lan et al., 2009]. Variation in shape, rotation and scale all provide considerable difficulties when developing robust visual features. Recent lip-reading experiments have used simple geometric-based features [Zhi et al., 2004], Discrete Cosine Transform (DCT) features [Hong et al., 2006; Almajai and Milner, 2008] or active appearance models (AAMs) to represent shapes of interest in an image [Newman et al., 2010; Hilder et al., 2009; Cox et al., 2008; Lan et al., 2010; Cootes et al., 1995; Cootes and Taylor, 2001]. With comparisons drawn between these techniques in their application to automated lip-reading, we use active appearance model features as we have found that they perform best in automated lip-reading tasks [Cox et al., 2008; Lan et al., 2009]. This section describes the extraction of active appearance model features for use in automated lip-reading, which are the features used throughout this thesis.

2.2.1 Active Appearance Models

Active Appearance Models (AAMs) are frequently used to represent shapes and textures in a compact form. The term AAM is most frequently used to define a composite model which consists of: the 2D shape — initially extracted from the image as x and y coordinates and to be represented by the Point Distribution

Model (PDM), and the appearance model, representing the pixel intensities of the image inside the shape boundary. Both features are compactly represented by a linear model that is usually computed using Principal Components Analysis (PCA) — a common technique to decorrelate and reduce the dimensionality of correlated features.

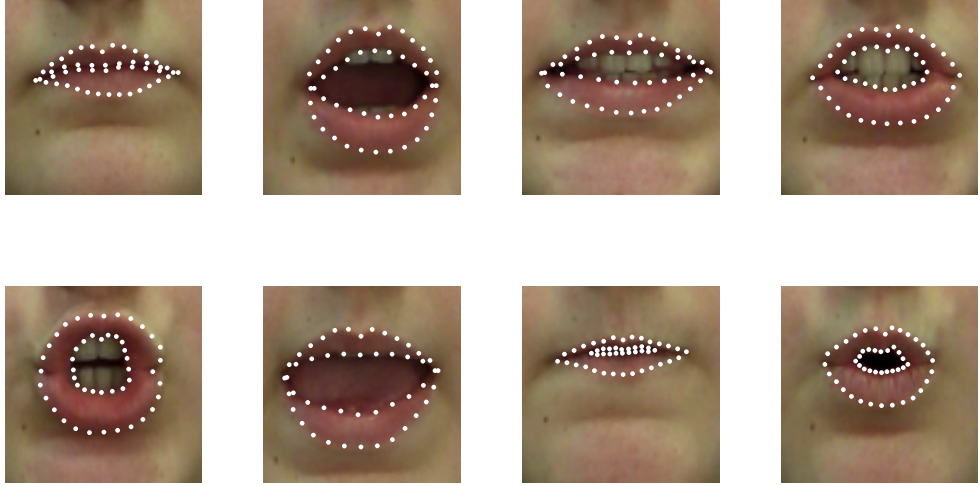


Figure 2.2: Example frames taken from a corpus. Frames have been carefully selected to describe extremities in lip movements in an attempt to capture maximum variation. Each selected frame is manually labelled with k points (where k is equal for all landmark frames)

To build a model of the appearance, a collection of frames is carefully chosen from the data set to represent the extremities of movement — e.g. mouth open, mouth closed etc. [Cootes, 2000] discusses the suitability of landmark locations and defines clear corners or ‘T’ junctions between boundaries as the best landmarks for face tracking and recognition. Inner and outer lip contours are labelled with a set of k landmarks (Figure 2.2) and all feature points are normalised for translation, rotation and scale before being subject to PCA to produce a PDM of the form

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{P}\mathbf{b}_s, \quad (2.1)$$

where $\bar{\mathbf{s}}$ is the mean shape, \mathbf{P} is a matrix containing the n eigenvectors of the covariance matrix, and \mathbf{b}_s defines the contribution of each variation mode to the

resulting shape \mathbf{s} . The permitted variation is typically restricted to a limit of ± 3 standard deviations from the mean shape.

AAMs also model the variation in appearance, with each image being normalised to the mean shape ($\bar{\mathbf{s}}$). PCA is then performed on the shape normalised pixel intensities within the labelled images to give an appearance model of the form

$$\mathbf{a} = \bar{\mathbf{a}} + \mathbf{R}\mathbf{b}_a, \quad (2.2)$$

where $\bar{\mathbf{a}}$ is the mean appearance image, \mathbf{R} is a matrix containing m eigenvectors to define variation modes, and \mathbf{b}_a defines the weighting of each variation in appearance to the resulting image, \mathbf{a} . A graphical example of this model is shown in Figure 2.3. For this example, the resulting shape and appearance is represented by the mean shape and three example modes of weighted variation in shape and appearance.

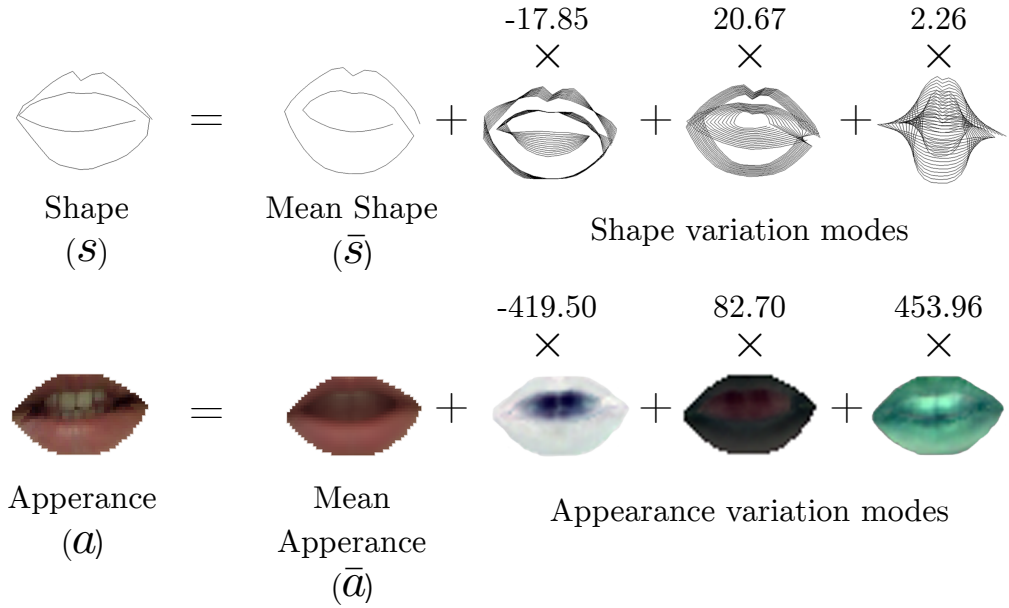


Figure 2.3: A graphical representation of how the shape and appearance features are computed. A shape (s) can be obtained from a sum of the mean shape (\bar{s}) and the weighted modes of variation in shape (e.g. lip rounding, mouth opening etc). Similarly, an appearance image (a) can be computed as the sum of the mean appearance (\bar{s}) in the shape-free patch with the weighted modes of appearance variation in the model.

For an image with a set of labelled landmarks (s), the shape parameters can be computed using the shape and the model:

$$\mathbf{b}_s = \mathbf{P}^T(\mathbf{s} - \bar{\mathbf{s}}). \quad (2.3)$$

Similarly, the appearance parameters are estimated by firstly warping the image to the mean shape ($\bar{\mathbf{s}}$) to produce \mathbf{a} and then using:

$$\mathbf{b}_a = \mathbf{R}^T(\mathbf{a} - \bar{\mathbf{a}}). \quad (2.4)$$

Manually labelling every image from a video is extremely time-consuming. For example, a 5 minute video that has been sampled at 25 frames per second will generate 7,500 image frames, each with k landmark points to synchronise between frames. To make this process more efficient, the inverse compositional project-out algorithm was proposed [Matthews et al., 2004] to track landmarks through a sequence of images.

Shape and appearance features are computed on a per frame basis before being concatenated together into a time-series of feature vectors where $t = 1 \dots T$:

$$F = \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \mathbf{s}_t & \dots & \mathbf{s}_T \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_t & \dots & \mathbf{a}_T \end{bmatrix}. \quad (2.5)$$

2.2.2 Post-Processing

The features computed by Equation 2.5 form a sequence of AAM parameters as a function of time. Each dimension of an AAM vector has a different range. If these values were used in classification algorithms, some dimensions would dominate over others. To overcome this, each dimension is z-score normalised in the following form:

$$norm(F_{i,j}) = \frac{F_{i,j} - \mu_i}{\sigma_i}; \quad (2.6)$$

where $f_{i,j}$ is the unnormalised feature at row i and column j in the feature matrix from Equation 2.5, μ_i is the row-wise mean and σ_i is the row-wise standard deviation.

As speech is a signal of temporal nature, in audio speech recognition it is common practice to append velocity (Δ) and acceleration ($\Delta\Delta$) dimensions to capture the dynamics of speech. Such techniques have also been successfully applied to automated lip reading using the temporal information from the AAM features [Lan et al., 2009]. In this thesis, both of these additional features which are computed over a window of two frames are used.

2.3 Language Modelling

A language model is a statistical model of symbol sequences. Hypothesised symbol sequences can be ranked with respect to the likelihood of the sequence which is advantageous to systems (such as ASR systems) that benefit from the constraints of language. In speech recognition, the language model is used to determine the most likely word sequence and can be used to discriminate between similar sounds in different contexts. In the majority of cases, a language model is used to model the probabilities of whole-word sequences. However, they can also be applied to sub-word units such as phonemes (*phonotactics*). The sparsity of available training data can be a problem when building a language model. Ideally, we wish to compute the probability of a sequence of words ($Pr(w_1, w_2, \dots, w_N)$) by using the chain rule of probability, as shown in Equation 2.7.

$$Pr(w_1, w_2, \dots, w_N) = Pr(w_1) \times P(w_2|w_1) \times Pr(w_3|w_2, w_1) \times \dots \times Pr(w_N|w_1, w_2, \dots, w_{N-1}) \quad (2.7)$$

As N increases, the computation in Equation 2.7 becomes increasingly impossible because of the difficulty in estimating probabilities conditioned on a large number of previous events, each of which can have many outcomes. The most popular solution

to this problem is the n -gram assumption. An n -gram language model provides a structure from which to compute the probability of the next word given the previous $(n - 1)$ words, thus reducing the dependence on having a large training set. There are three popular n -gram models that are regularly used:

1. The **unigram** model uses only frequency counts from the current word to determine its conditional probability.

$$Pr(w_1, w_2, \dots, w_i) \approx Pr(w_1) \times P(w_2) \times Pr(w_3) \times \dots \times Pr(w_i) \quad (2.8)$$

2. The **bigram** model computes the probability of the given word based only on the frequency count of the previous word followed by the current word.

$$Pr(w_1, w_2, \dots, w_i) \approx Pr(w_1) \times P(w_2|w_1) \times Pr(w_3|w_2) \times \dots \times Pr(w_i|w_{i-1}) \quad (2.9)$$

3. The **trigram** assumption uses two previous words to determine the probability of a current word. This technique is commonly used in speech recognition which use large vocabulary data sets.

$$Pr(w_1, w_2, \dots, w_i) \approx Pr(w_1) \times P(w_2|w_1) \times Pr(w_3|w_2, w_1) \times \dots \times Pr(w_i|w_{i-1}, w_{i-2}) \quad (2.10)$$

Although these techniques provide good results, issues can still arise with data sparsity. If an n -gram is not observed in the training procedure, its probability is zero which means it cannot be decoded. This problem can be addressed by using *smoothing techniques*.

2.3.1 Smoothing Techniques

As the value of n in an n -gram increases, the structure of the language is better defined because of the larger context, but the training procedure also requires a larger training set. Smoothing techniques are applied to a language model to remove some probability from higher frequency sequences and re-distribute probability mass across to other lower probability sequences [Goodman, 2001]. The term *smoothing* refers to the technique which attempts to adjust the probability distribution to be more uniform, increasing the much smaller probabilities and decreasing the high probabilities. The main objective in speech recognition is to find the word sequence, W , that maximises $Pr(W|A)$ and A represents the acoustic signal. This is performed by using Bayes' theorem (as shown in Equation 2.11). If $Pr(W)$ is estimated as zero, because of the inadequacy of the language model, $Pr(W|A)$ is zero. Smoothing is applied to the language model to prevent any of these probabilities reaching zero.

$$P(W|A) = \frac{Pr(A|W)P(W)}{P(A)} \quad (2.11)$$

There are many techniques that can be used to smooth an n -gram vocabulary distribution. Popular methods include Laplace smoothing, Good-Turing smoothing, Katz smoothing [Katz, 1987] and Kneser-Ney smoothing [Ney et al., 1994]. [Chen and Goodman, 1999] presents a study on these different smoothing techniques, providing analysis on performance and potential issues.

A simple approach to smoothing is to add a constant to the frequency count of each n -gram, a technique called Laplace smoothing (also known as additive smoothing). Here, a smoothing parameter, δ is added to the frequency counts for a given bigram, $c(w_{i-1}, w_i)$ according to Equation 2.12. Previous studies have shown that this technique performs rather poorly [Chen and Goodman, 1999].

$$Pr(w_i|w_{i-1}) = \frac{\delta + c(w_{i-1}, w_i)}{\sum_{w_i} \delta + c(w_{i-1}, w_i)} \quad (0 < \delta \leq 1) \quad (2.12)$$

An improved smoothing technique is *Katz smoothing* (introduced in [Katz, 1987]). Here, models are defined recursively in terms of lower-order n -gram models (known as a back-off model). For example, if an n -gram sequence is not observed enough times in the training set, a back-off procedure shortens the context window to $(n - 1)$ symbols. For a more comprehensive description of Katz smoothing, see [Katz, 1987; Chen and Goodman, 1999]

In this thesis, we use Katz smoothing for all n -gram language models where $n > 1$ (e.g. bigram models) as it provides the best results using the language model tools that are part of the Hidden Markov Model Toolkit (HTK) [Young et al., 2006].

2.4 Hidden Markov Models

A *Hidden Markov Model* (HMM) is a stochastic process commonly used to model temporal data (e.g. speech). An HMM consists of a number of *states* which are connected by arcs (with associated probabilities). Each state has an associated probability distribution over the feature-space used to characterise the data. An HMM assumes that output observations and transitions between states are dependent only on the current state (and not any previous states), forming a *Markov chain* of states through the model. In a simple *Markov chain*, the state sequence is visible. In a *hidden Markov model*, the underlying state sequence that was used to produce any sequence of observations is not available, hence, the state sequence is *hidden*.

Sections 2.4.1 and 2.4.2 describe HMMs using discrete and continuous probability densities respectively. HMMs have been successfully applied to temporal pattern recognition problems such as handwriting recognition, biological DNA sequences, part-of-speech tagging, speech synthesis and speech recognition.

2.4.1 HMMs with Discrete Probability Densities

In a discrete observation probability HMM, the observed outputs are drawn from a finite alphabet of symbols. When these HMMs are used for speech processing, it is common to derive the symbols from a vector quantisation process applied to the continuous speech signal. Figure 2.4 shows a three-state HMM with discrete observation probabilities for three symbols: a , b , and c , in each state. The parameters for this model can be defined as follows:

- $X = \{x_1, x_2, \dots, x_N\}$ — a set of N observation symbols
- $S = \{s_1, s_2, \dots, s_M\}$ — a set of M states
- $\pi = \{\pi_1, \pi_2, \dots, \pi_M\}$ — a vector of initial state probabilities

- $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1M} \\ a_{21} & a_{22} & \dots & a_{2M} \\ \dots & \dots & \dots & \dots \\ a_{M1} & a_{M2} & \dots & a_{MM} \end{bmatrix}$

— a state transition matrix providing a probability for each transition between every state where a_{ij} is the probability of moving from state i to state j .

- $O = \begin{bmatrix} o_{11} & o_{12} & \dots & o_{1N} \\ o_{21} & o_{22} & \dots & o_{2N} \\ \dots & \dots & \dots & \dots \\ o_{M1} & o_{M2} & \dots & o_{MN} \end{bmatrix}$

— a matrix defining the *discrete* observation probabilities where rows represent different states and observations are represented in the columns. Therefore, an observation o_{ij} provides the probability of being in state i and observing symbol j .

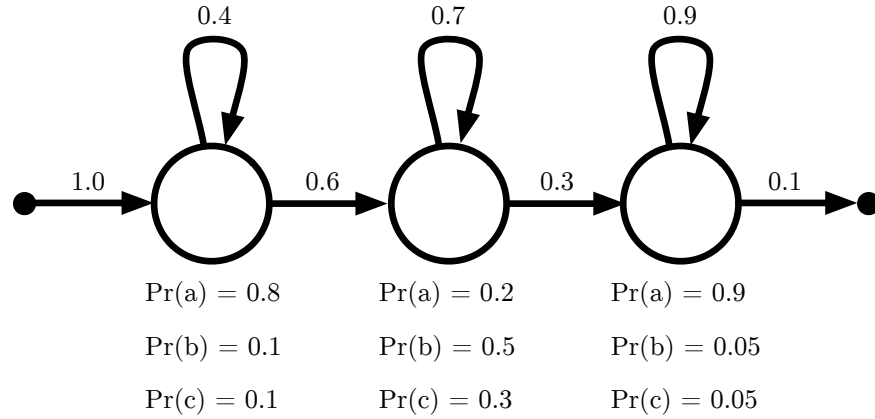


Figure 2.4: A three-state HMM with discrete probability densities. In this example, there are three possible observations: a, b and c, each with likelihoods associated in all three states.

2.4.2 HMMs with Continuous Probability Densities

Using discrete observation probabilities (as detailed in Section 2.4.1) can lead to a loss of accuracy due to the speech signal being quantised [Cox, 1988]. To overcome this problem, it would be possible to increase the number of quantisation points. However, this increases computational cost. If the observations can be represented by a parametric continuous probability density function, the observation matrix will require only the parameters of this distribution. HMMs using continuous probability densities have the same underlying structure as HMMs using discrete probability densities. However, the observations are continuously valued vectors and the observation probability is a likelihood from a parametric distribution. A Gaussian Probability Distribution Function (GPDF) can be used to model observation vectors under the assumption that they are normally distributed. For a multivariate distribution, only the mean vector ($\boldsymbol{\mu}$) and covariance matrix ($\boldsymbol{\Sigma}$) are required to characterise the distribution. Hence, we can use a Gaussian distribution (shown in Figure 2.5) to model the probability of observing feature vector \mathbf{o}_t in state j ($b_j(\mathbf{o}_t)$) as follows:

$$b_j(\mathbf{o}_t) = \frac{1}{(2\pi)^{\frac{D}{2}} |\mathbf{\Sigma}_j|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_j)^T \mathbf{\Sigma}_j^{-1} (\mathbf{o}_t - \boldsymbol{\mu}_j)\right); \quad (2.13)$$

where D is the Gaussian dimensionality, $\boldsymbol{\mu}_j$ is the mean at state j and $\mathbf{\Sigma}_j$ is the covariance matrix at state j .

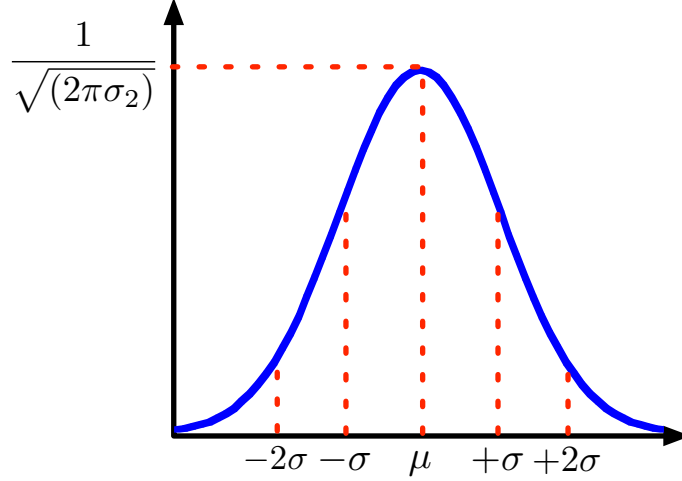


Figure 2.5: An example of a Gaussian PDF centered around the mean denoted as μ with a standard deviation, σ . Likelihoods can be produced for a given input feature vector to describe the fit to the GPDF.

In practice, the observed features from a speech signal, even those modelling a single speech sound, are far from normally distributed. Therefore, ASR systems usually model the speech feature vectors using a *Gaussian Mixture Model* (GMM). GMMs combine a set of independent Gaussian PDFs into a single model, consisting of a mean (μ) and variance (σ) for each GPDF along with an associated weight. The individual *mixture components* are combined to form a more complex distribution (as shown in Figure 2.6).

Using the weighted sum of GPDFs to form GMMs, an HMM can be redefined as follows:

- $S = \{s_1, s_2, \dots, s_M\}$ — a set of M states,
- $\pi = \{\pi_1, \pi_2, \dots, \pi_M\}$ — a vector of initial state probabilities,

- $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1M} \\ a_{21} & a_{22} & \dots & a_{2M} \\ \dots & \dots & \dots & \dots \\ a_{M1} & a_{M2} & \dots & a_{MM} \end{bmatrix}$

— a state transition matrix providing a probability for each transition between every state where a_{ij} is the probability of moving from state i to state j ,

- $P_i = \sum_{k=1}^K W_{ik} N(\mu_{ik}, \Sigma_{ik})$ — where P_i represents the probability distribution over the continuous features for state i , W_{ik} is the weight of the k^{th} mixture component, μ_{ik} is the mean of mixture component k for state i , and Σ_{ik} is the covariance matrix for the mixture component k in state i ($\sum_{k=1}^K W_{ik} = 1$).

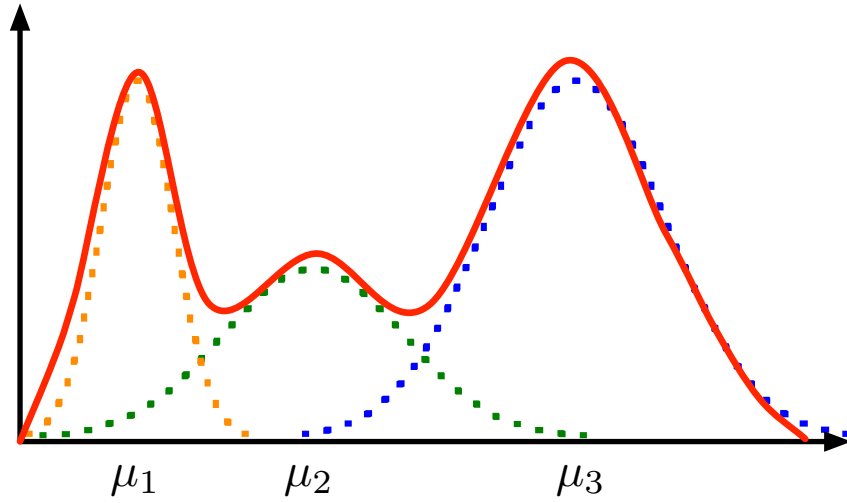


Figure 2.6: An illustration of a Gaussian Mixture Model (GMM). In this example, there are 3 mixture components that form a resulting distribution that is non-Gaussian. Each GPDF has its own mean (μ) and variance (σ).

2.4.3 Hidden Markov Models in Speech Recognition

The speech signal can be thought of as a time series, producing data points as a function of time. Because of the nature of speech, backward transitions (i.e. moving back from the current point in the time series) is an impossible concept. Therefore, the standard HMM topology used in ASR is a left-to-right configuration where the transitions from state i to j for $j < i$ are impossible.

In speech recognition, separate HMMs can be used to model units that make up words. For audio ASR, these units usually represent *phonemes* — the smallest useful unit of acoustic speech. In audio ASR systems, each word is usually modelled as a sequence of phoneme HMMs (as depicted in Figure 2.7) with each phoneme model being a simple HMM. The sequence of HMMs shown in Figure 2.7 models the word *hello* by using the four phoneme models — $/hh/$, $/ax/$, $/l/$ and $/ow/$.

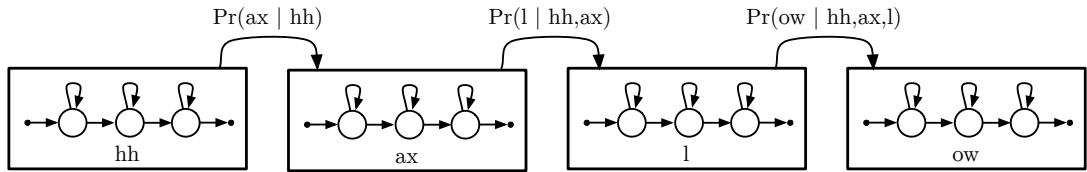


Figure 2.7: Words can be modelled as network of HMMs. Here, the word *hello* is decomposed into phoneme units and represented as a network of sub-HMMs. Segmented acoustic features can be passed through this network and output the likelihood that the input features came from the set of models. The probabilities between the HMMs relate to the language model likelihoods.

Given the topology of an HMM, there are three issues to address when modelling in this way for speech recognition [Rabiner, 1989]:

1. Given a model (λ) and a sequence of observations ($O = \{O_1, O_2, \dots, O_n\}$), compute the probability of the observation sequence given the model ($Pr(O|\lambda)$) [Rabiner, 1989]. This is known as the *evaluation* problem and can be solved using the *forward probability matrix*.
2. Given an initial model for each phoneme and some training observations, O , find the model parameters that maximise $Pr(O|\lambda)$ — known as the *estimation* problem. An obvious approach to maximising $Pr(O|\lambda)$ is to consider all

combinations of state sequences that could be used to produce the observation O , and attempt to find the sequence that maximises the probability. This quickly becomes an impossible task as the number of states and observations increases, as it would require too many possible sequences to be evaluated. The *Baum-Welch algorithm* uses the forward and backward probabilities to maximise $Pr(O|\lambda)$. For a description of this algorithm and how it estimates the new HMM parameters, see [Cox, 1988].

3. The final problem is concerned with finding the most likely state sequence for a given observation sequence, O , and a model, λ (i.e. trying to uncover the ‘hidden’ part of the HMM). This is solved using the *Viterbi algorithm* [Viterbi, 1967]. Unlike the Baum-Welch algorithm, which estimates the transition and observation probabilities from a set of training observations, the Viterbi algorithm finds the most likely state sequence from an HMM (with known transition and observation probabilities) by searching over paths in a Viterbi matrix. For a more detailed description of the Viterbi algorithm, see [Cox, 1988; Viterbi, 1967; Rabiner, 1989].

To compute the accuracy of an HMM system, the recognised sequence is aligned to the ground-truth sequence using dynamic programming (DP). The optimal alignment is considered as the sequence matching that has the lowest score where deletions and insertions carry a weight of 7, substitutions carry a score of 10, and identical symbols carry a zero weight [Young et al., 2006]. Using the optimal alignment, the accuracy of a system is defined as:

$$accuracy = \frac{N - D - S - I}{N} \times 100\%, \quad (2.14)$$

where D is the number of deleted symbols, I is the number of inserted symbols, S is the number of substituted symbols, and N is the total number of symbols in the ground-truth sequence [Young et al., 2006].

2.4.3.1 Monophone HMMs

In a monophone HMM system, each model represents an isolated unit. These units can take the form of sub-word units such as phonemes or whole word units. In some ASR applications, monophone HMMs are built for all phonemes in the English language. Additional silence models and optional short-pause models are also included for word termination and inter-word separation respectively.

We use a tool that has been standardised in the ASR community, namely, the Hidden Markov Model Toolkit (HTK) [Young, 2001] for all HMM-based work in this thesis. HTK can be used in two ways to build monophone HMMs. The first method requires a time-aligned transcription. HMMs are initialised by dividing the observations equally amongst the states of a model (specified by the manually segmented, time-aligned training transcription). The Viterbi algorithm is used to find the state alignment that maximises the likelihood for the given observation. This process is performed iteratively until there is no change in the resulting state alignments. After initialisation, HMM parameters are trained using the Baum-Welch re-estimation algorithm. One disadvantage of this technique is the requirement for time-aligned transcriptions. Usually, this procedure is performed manually by a human. However, for sub-word units such as phonemes, alignments may be time-consuming and inaccurate because of the rather subjective nature of manual sub-word segmentation. To avoid these errors in segmentations, a second method known as the *flat start* can be used. Model initialisation is performed using a uniform segmentation strategy, assigning a global mean and variance to each Gaussian distribution in each HMM. After initialisation, the re-estimation procedure uses *embedded training* where all of the HMM parameters of the HMMs are updated with one iteration of the Baum-Welch algorithm using all of the training data [Young, 2001]. After multiple iterations of *embedded training*, the parameters of the HMMs are trained and ready for recognition of a test sequence using the *Viterbi decoding algorithm*.

2.4.3.2 Triphone HMMs

There have been many studies investigating the influence of surrounding sounds in human speech production, namely, the *coarticulation* effect. This has led to extending monophone HMM systems to model a certain phoneme in a given context. This is known in ASR as *triphone modelling*. A triphone HMM is constructed using the same approach as a monophone HMM except, this time, each phoneme has a left and right context. An example of a triphone transcription is shown in Figure 2.8. Here, the phrase “*Hello Sir*” is converted to a sequence of monophones using a dictionary. To construct the triphone sequence, each phoneme in the monophone transcription is considered as a subject with a left (-) and right (+) context. The sequence shown in Figure 2.8 also shows two special monophones: the termination label (*sil*) which models the silence before and after an utterance; and the short-pause inter-word boundary label (*sp*). The triphones that cross over the word boundaries include context information from previous or next words. This is known as a *cross-word* triphone system [Young, 2001].

	Hello					Sir				
Monophone:	sil	hh	ax	l	ow	sp	s	er	sp	sil
Triphone:	sil	hh+ax	hh-ax+l	ax-l+ow	l-ow+s	sp	ow-s+er	s-er	sp	sil

Figure 2.8: An example of both monophone and triphone transcriptions for the phrase “*Hello Sir*”. For monophones, a simple look-up dictionary is required to convert words into phoneme sequences. The *sil* label denotes sentence termination whilst the *sp* label is used to divide phoneme sequences into words. In the triphone case, a cross-word example is shown where boundaries are isolated as monophones but triphones span over the inter-word boundaries.

Ideally, a triphone system would construct an HMM for every triphone sequence encountered in a language. For the 44 phonemes in the English language, this could potentially involve building 85,184 (44^3) separate HMMs, each needing enough training data to model the triphone successfully. However, because the training data is limited, this is an unachievable target. Initial work in sparsity tying for HMMs was centered around the use of simply tying triphone models to similar sounds [Lee et al.,

1990]. However, as the left and right contexts of a triphone are not independent, model training can be sub-optimal [Young et al., 1994]. The work carried out in [Young, 1992; Young and Woodland, 1993] provides a method for a more effective coupling of triphone models — at the state level.

In state-tying, unseen triphone models from training are *tied* by the states within the HMM. There are two approaches to deciding the coupling of triphone states: data-driven and tree-based clustering. The work carried out in [Young et al., 1994] introduces the tree-based clustering and provides a performance comparison with previous data-driven clustering methods, concluding that recognition performance is improved using tree-based state clustering. This work uses the tree-based clustering technique to produce a set of tied-state triphone models.

Up until the work published in [Young et al., 1994], data-driven clustering was used to tie HMM states to reduce the number of models required. This method works well for word-internal triphone systems as it is always possible to find some training data for a particular triphone. However, for cross-word triphone systems, the number of different training examples required is much larger, leading to data sparsity issues with some triphones that are unseen in the training data. This is where the tree-based clustering methods are advantageous.

In tree-based clustering, a phonetic decision tree is built where each node holds a question that relates to the phonetic context. An example of such a decision tree is shown in Figure 2.9. Here, the root node is associated with the question “*Is the right phoneme a fricative?*”. A tree is constructed for each state of each subject phoneme. The corresponding states are partitioned into the leaf nodes of the tree (in the example shown in Figure 2.9, there will be 8 partitions). Each tree is optimised to maximise the likelihood using all permutable locations of question nodes and different tree topologies. Once the trees have been optimised, unseen triphones can be synthesised by traversing the binary trees for a given phoneme in a specific context. From this optimisation process, states from unseen triphone models can be tied to states in models that have been observed during training.

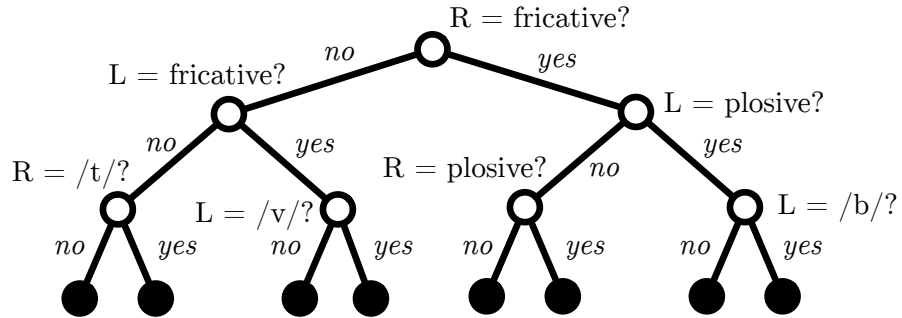


Figure 2.9: An illustrated example of a phonetic decision tree. Each node stores a question pertaining to the context surrounding the subject phoneme. ‘L’ and ‘R’ refer to the left and right context respectively. Questions can be wide in their scope or channelled down to identifying specific phonemes.

2.4.4 Networks for Decoding

Sequences of HMMs can be combined together to form a *network*. Nodes represent units which can either be in the form of sub-word units (e.g. phonemes) or whole words. The decoding network is formed by connecting nodes with edges to form a sequence of units (e.g. words or phonemes). Figure 2.11 demonstrates a simple example of a decoding network with whole word nodes connected by edges which hold two likelihoods: the acoustical likelihood that has been produced by the HMMs given some input data, and the grammar likelihoods which are defined by the language model.

At decoding time, the networks are traversed to produce a most likely hypothesis using the *Viterbi decoding algorithm* with a technique named *token passing*. Here, tokens are passed through the nodes of the network with the total likelihood being recorded at every stage. If a token arrives at transition to multiple nodes, the token is duplicated to continue on all paths through the network, with the history of a token (i.e. the previously visited nodes) being recorded all of the way through the network. The best path through the decoding network is evaluated as the token which reaches the end of the network with the highest likelihood. To aid the decoder in finding a path, an *insertion penalty* may be used. This fixed value is added to each edge (i.e. the transition between words) to penalise the transition to the next node.

Figure 2.10 shows how the number of inserted and deleted units is affected by values of the insertion penalty between -20 and 20. As the insertion penalty is increased, the decoder recognises fewer deleted units but more inserted units, whereas lower insertion penalty values will introduce more deletions but fewer insertions. This value can vary depending on the task and is therefore optimised at recognition time to find the best performance.

In all ASR tasks, a language model has proven to significantly improve decoding performance. With most commercial ASR systems being trained with thousands of utterances and large vocabularies, the use of grammar becomes imperative. Each edge in a decoding lattice stores two likelihoods: acoustic and grammar. Because the grammar and model likelihoods are separated until decoding time, weighting can be applied independently to introduce more influence for either likelihood. The *grammar scale factor* is defined as a fixed positive real number that is applied to the language model likelihood for recognition.

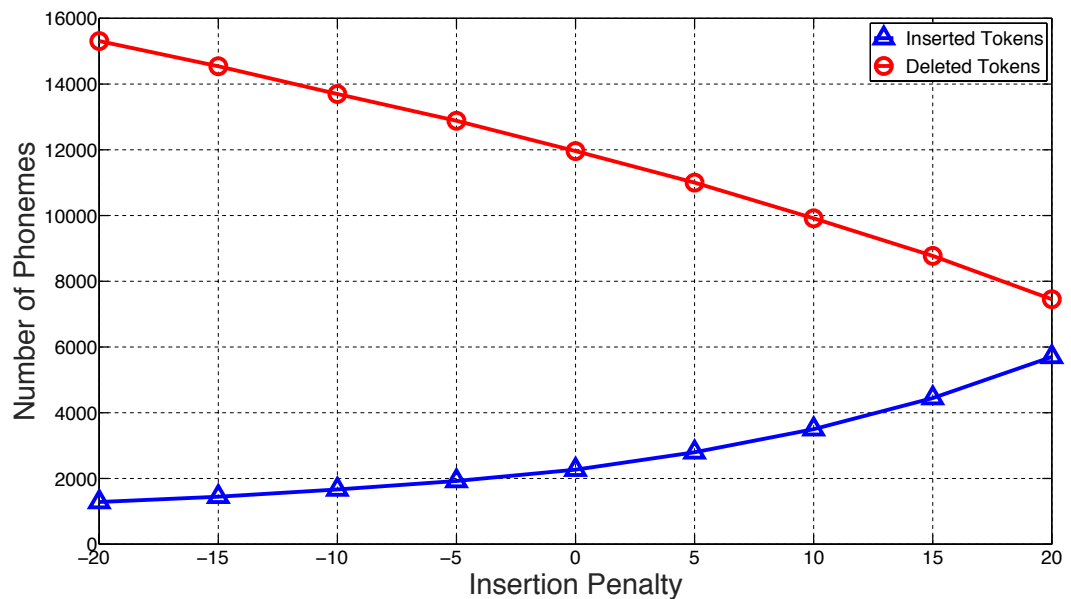


Figure 2.10: The number of deleted and inserted phonemes in the hypothesised sequence as a function of the insertion penalty. A smaller insertion penalty attracts less insertions but more deletions whereas a larger insertion penalty causes more inserted tokens but less deletions.

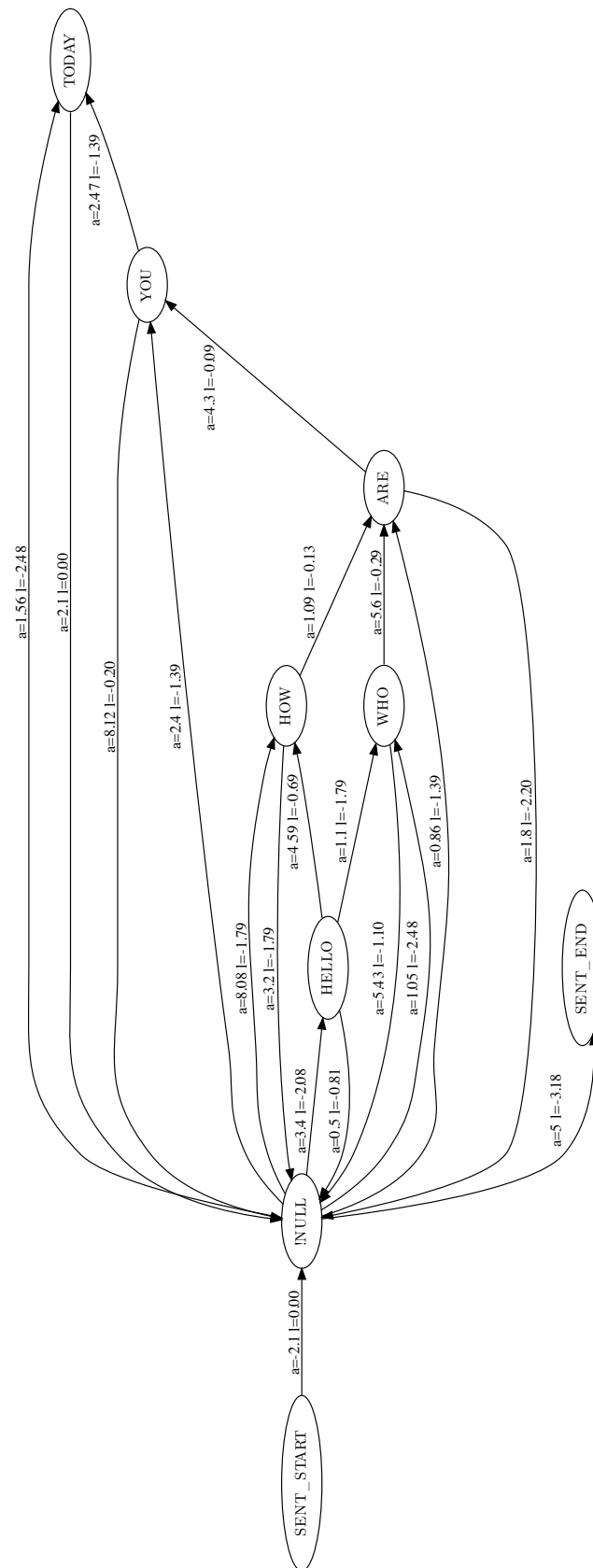


Figure 2.11: A representation of a decoding lattice for a simple set of sentences. In this example, nodes represent whole words (these can either be modelled as a set of sub-word unit HMMs or whole word HMMs). Edges connect the nodes where each edge has an acoustic likelihood (a) and a language model likelihood (l) associated with the transition.

By using a combination of acoustic and language model likelihoods together with the insertion penalty and grammar scale factor constants, a total likelihood can be computed at each stage in the decoding lattice:

$$P = \log(\Pr(A|\lambda)) + (\log(\Pr(w_2|w_1)) \times s) + p; \quad (2.15)$$

where $\Pr(A|\lambda)$ is the likelihood of observing the acoustical features from the model (λ), $\Pr(w_2|w_1)$ represents the language model likelihood (in this case, using a bigram language model to find the probability of observing symbol w_2 after w_1). Due to the underflow of dealing with small likelihoods, log likelihoods are used throughout the HMM training and decoding process. In addition, a grammar scale factor is used to boost the reliance of the language model over the acoustic information (s) and an insertion penalty (p) that is added to each transition to favour/penalise the movement out of a node are used.

When decoding speech, it is occasionally necessary to produce multiple hypotheses (n -best decodings). When decoding an utterance, a list of hypotheses is produced, ranked in order of decreasing likelihood. An n -best list can be generated to retrieve the top ranked n most likely hypotheses. For this task, lattices are advantageous as they provide the ability to traverse multiple paths to produce n -best decoded lists efficiently.

2.5 Confusion Matrices

Confusion matrices document patterns of confusions between recognised units. For a phoneme confusion matrix, columns represent the recognised phoneme from a speaker (i.e. input) whilst rows represent the input phoneme. A significant benefit of using confusion matrices is that patterns of confusions can often be observed. In speech recognition, confusion matrices can be used to model the patterns of *insertions*, *substitutions* and *deletions* that occur at phoneme ([Morales, 2009]) or word-based ([Green et al., 2003]) levels.

Figure 2.12 is a simple example of a typical recognition confusion matrix using five classes (letters ‘a’ to ‘e’). To estimate this confusion matrix, the hypothesised sequence of symbols are aligned to the ground-truth symbols using *dynamic programming* described further in Section 5.3.2). Taking each pair of mapped symbols in turn, the confusion matrix is populated with counts, where the columns represent the recognised sequence (i.e. the response) and the rows represent the ground-truth data (i.e. the input). *Deletions* and *insertions* are also described using an additional column and row respectively.

		Response					
		a	b	c	d	e	DEL
Input	a	6	1	0	2	2	1
	b	2	3	7	2	1	1
	c	0	0	5	2	1	0
	d	0	0	0	4	1	2
	e	1	0	3	0	2	3
INS		1	2	1	5	2	

Figure 2.12: An example of a typical confusion matrix for five classes (letters ‘a’ to ‘e’). The hypothesised and ground-truth sequences are matched together to form a set of confusions. Substitutions are modelled by the 5x5 square section in the middle with rows representing the ground-truth (input) and columns representing the hypothesised symbol (response). Insertions and deletions are represented by an additional row and column. Each entry in this confusion matrix represents a frequency count for the given confusion.

[Morales, 2009] discusses the use of confusion matrices in recognition of dysarthric speech. Using the information obtained from confusions for a particular speaker at the phoneme level, the approach used in [Morales, 2009] introduces a foundation for initialising a *weighted finite state transducer* to model confusions using states and transductions (more details about weighted finite state transducers can be found in Section 2.6).

2.5.1 Confusion Matrix Smoothing

Smoothing techniques have been previously discussed in Section 2.3.1 as a method to re-distribute a portion of the observed probability mass to unseen events in the

language structure. In the same manner, smoothing techniques can also be applied to confusion matrices to redistribute probability to unobserved confusions. For any given input symbol, there are multiple responses — some of which are not observed in the training set and therefore have zero probability. Smoothing techniques share a segment of the probability mass to these events, ensuring that unobserved confusions have a small probability.

In this thesis, we use two different confusion matrix smoothing methods: base smoothing — a technique described in [Morales, 2009] to re-distribute a fixed percentage to off-diagonal events (described in Section 2.5.1.1), and exponential smoothing which uses the exponential function and a smoothing parameter to determine the spread of the probability mass over the rows of a confusion matrix (described in Section 2.5.1.2).

2.5.1.1 Base Smoothing

In a typical confusion matrix, it is common to have a high percentage of confusion count on the diagonal elements (i.e. correctly recognised elements). However, in this work, which relies on the correction of recognised events, the main focus is on off-diagonal elements (i.e. substitutions or insertions and deletions). Base smoothing [Morales, 2009] attempts to solve this problem by re-distributing a proportion of the leading diagonal of a confusion matrix to other classes. For a given matrix containing confusion counts (C), a smoothed count matrix (S) is estimated. An off-diagonal element in this matrix at row i , column j is defined as:

$$S(i, j) = C(i, j) + \frac{C(i, i) \times d}{n} \quad \forall j \neq i, \quad (2.16)$$

where d defines the smoothing parameter where ($0 < d < 1$), and n is the total number of classes (i.e. the number of columns in the confusion matrix). After giving count mass to the off-diagonal elements, the smoothing parameter percentage is then deducted from the on-diagonal elements according to Equation 2.17.

$$S(i, i) = C(i, i) - (C(i, i) \times d) \quad (2.17)$$

Using these rules, the smoothing parameter, d , can be adjusted to change the amount of mass that is re-distributed from the diagonal elements. Figure 2.13 shows the effect of this smoothing technique on an example confusion matrix which has been estimated by performing recognition on the ISO-211 dataset (see Section 4.2). Here, three different distribution parameters (d) are used to produce new estimates of the confusion matrices: 0.25, 0.5, 0.75 (i.e. 25%, 50%, and 75% respectively). These examples demonstrate how the probability of off-diagonal substitutions is increased as a larger percentage of the diagonal elements are re-distributed across the rows.

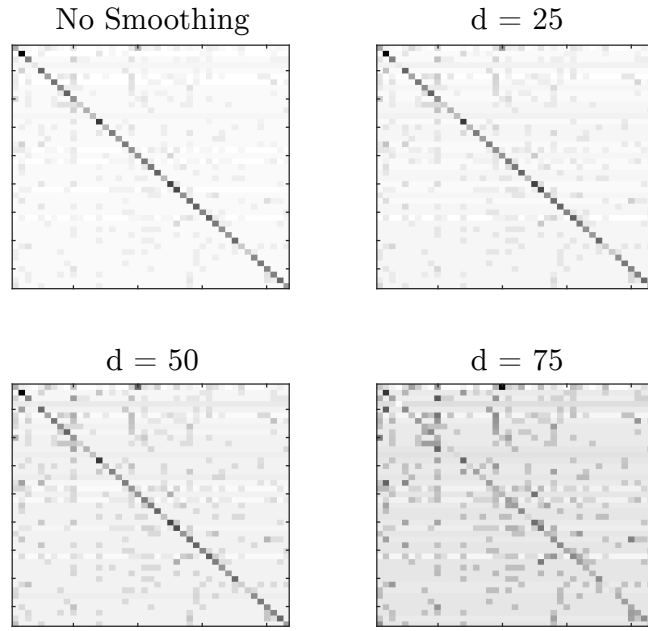


Figure 2.13: Examples of the effects of base smoothing using three different smoothing parameter values (d): 0.25, 0.5, and 0.75. Off-diagonal confusions become stronger and the diagonal is weakened as the distribution parameter is increased.

2.5.1.2 Exponential Smoothing

Exponential smoothing provides an alternative method to weaken the influence of the diagonal count and provide a smoothed matrix using the exponential function

as shown in Equation 2.18.

$$S(i, j) = \frac{e^{\alpha C(i, j)}}{\sum_k e^{\alpha C(k, j)}}, \quad (2.18)$$

where $C(i, j)$ is the number of times phoneme i has been confused with phoneme j , α is a data-dependent constant, and S is the resulting smoothed confusion matrix. The introduction of the parameter α adds stronger control to the degree of smoothing applied. As $\alpha \rightarrow 0$, the probability mass is equally distributed over the row i . Conversely, as $\alpha \rightarrow \infty$, the mass is more concentrated in the highest element. Figure 2.14 illustrates the effect of the smoothing parameter (α) on a confusion matrix using confusions produced by recognition on the ISO-211 dataset (see Section 4.2). Compared with the original confusion matrix (‘No Smoothing’), the smaller values of α (e.g. 0.01) distribute more of the probability mass across the rows. When $\alpha = 1$, the probability mass becomes significantly concentrated in the diagonal elements which makes off-diagonal substitutions less likely. In this thesis, we explore values of α in the range $0.01 < \alpha < 1$ and find that the smaller value of 0.01 works best.

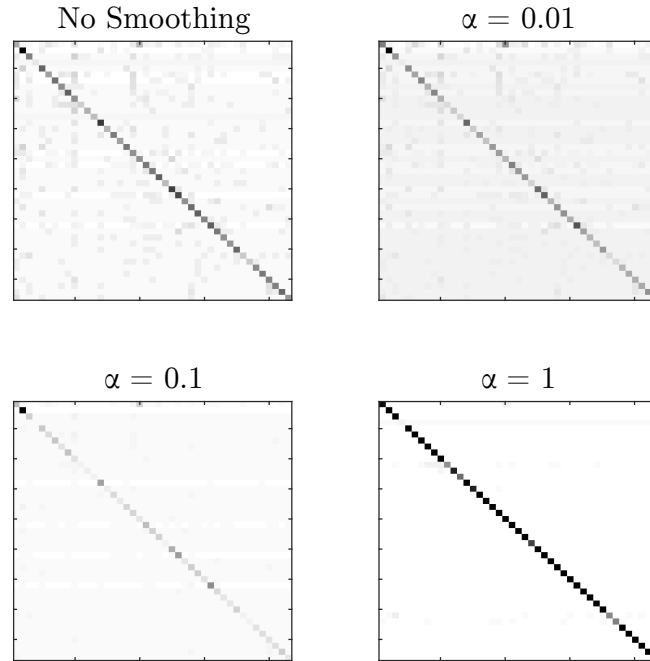


Figure 2.14: Exponential smoothing examples using three different values for α : 0.01, 0.1, and 1. Smaller α values re-distribute more of the larger probability mass across the row whereas a much larger α value concentrates the probability on the stronger (i.e. the diagonal) elements.

2.6 Weighted Finite-state Transducers

A finite state automaton (FSA) is a mathematical model of a sequence of events. An FSA is defined by a finite set of *states* which are connected by *transitions* (i.e. actions). Unlike an HMM, where observations are produced by the state visits, FSAs model discrete actions in the transitions between states. Weighted Finite-state Transducers (WFSTs) are built using the same theory as FSAs, except that a transition between two states becomes a translation between an input and output symbol. These transitions (also known as transductions) can also store weights that can be used to implement a probability distribution over a set of events. With the extra ability to perform this binary translation, a WFST can convert a sequence of input strings to a sequence of output strings with an associated total weight. In finite-state theory, the weighting system is defined over a *semiring* — defined as an algebraic structure over the five-tuple — $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ where $(\mathbb{K}, \oplus, \bar{0})$ is a *commutative* and *associative* monoid with an identity element defined by $\bar{0}$, $(\mathbb{K}, \otimes, \bar{1})$ is an *associative* monoid with an identity element defined by $\bar{1}$. The abstraction to semiring weighting systems allows the WFST algorithms (described in Sections 2.6.1 and 2.6.2) to be generalised over a range of weighting sets. Table 2.1 lists the most widely used semiring weighting systems for WFSTs [Mohri, 2004].

Semiring Type	Set	$\mathbf{a} \oplus \mathbf{b}$	$\mathbf{a} \otimes \mathbf{b}$	$\bar{0}$	$\bar{1}$
Boolean	$\{0, 1\}$	$a \vee b$	$a \wedge b$	0	1
Probability	\mathbb{R}_+	$a + b$	$a \times b$	0	1
Log	$\mathbb{R} \cup -\infty, +\infty$	$-\log(e^{-a} + e^{-b})$	$a + b$	$+\infty$	0
Tropical	$\mathbb{R} \cup -\infty, +\infty$	$\min(a, b)$	$a + b$	$+\infty$	0

Table 2.1: A selection of widely used semirings and their binary relations [Mohri, 2004]. \oplus and \otimes denote the binary operations and $\bar{0}$ and $\bar{1}$ represent the identity elements provided by the two monoids respectively. The definition of the binary operations (\oplus and \otimes) allows the flexibility to apply these WFSTs to the four weighting systems described in this table. Combinatorial and pruning operations (described in Sections 2.6.1 and 2.6.2 respectively) can then be defined on the definition of the binary relations.

A WFST can be defined over a semiring, \mathbb{K} as an eight-tuple: $T = (\Sigma, \Omega, Q, E, I, F, \lambda, f)$ [Mohri et al., 2002; Mohri, 2004] where:

1. Σ : a finite, non-empty set of input symbols
2. Ω : a finite, non-empty set of output symbols
3. Q : a finite, non-empty set of states
4. E : a finite set of transitions that define the relationship between states (Q)
5. I : a set of initial states ($I \subseteq Q$)
6. F : a set of final states ($F \subseteq Q$)
7. λ : an initial weighting function where $\lambda : I \rightarrow \mathbb{K}$
8. f : a final weighting function where $f : F \rightarrow \mathbb{K}$

Using this definition, a weighted finite-state transducer provides a mechanism to encode sequences of mapped symbols (i.e. an input symbol mapped to an output symbol), with an associated weight. As defined by the semiring, weights can represent probabilities, penalties (i.e. cost), duration or any other weighting function that can be used to define a total weighting system through a path sequence. In this thesis, we use the tropical semiring where arc weights are encoded as costs which can be used to favour/penalise a given path. Figure 2.15 is an example illustration of a 3-state WFST over a tropical semiring. The starting states are defined by the bold outline surrounding the state (state 0 in Figure 2.15) and final states are defined by double-line borders around the state (state 3 in Figure 2.15). The advantage of WFSTs over other types of automaton is the ability to take an input string (in this case ‘*abc*’) and transduce (i.e. convert) it to an output string (‘*xyz*’) with an associated total weighting derived from the individual transduction weights. The visual example of a WFST model shown in Figure 2.15 could be compared to an HMM where, for a given input, an output is produced. However, the two models operate

for different purposes and under different conditions. In a WFST, the conversion of a discrete input to a discrete output symbol is performed on the weighted transition. Therefore, the states in a WFST are purely to provide a means of maintaining a current position in an input sequence, unlike the states in an HMM, which model the probability distribution of the features.

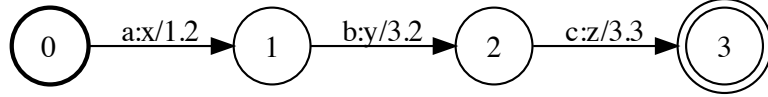


Figure 2.15: A example of a weighted finite-state transducer over the tropical semiring. This transducer has one initial state (0) and one final state (3) and only allows one path translating the string ‘*abc*’ to ‘*xyz*’ with a total path weight under the tropical semiring of $1.2 + 3.2 + 3.3 = 7.7$.

A transition $e \in E$ can be used to define a relationship between two states with a start state $s[e]$ and end state $k[e]$ with a weight defined by $w[e]$. A path (ϕ) can be defined as a set of successive transitions $e_1, e_2, \dots, e_n \subseteq E$ with a set of weights $w[\phi] = w[e_1], w[e_2], \dots, w[e_n]$. Given a path $\pi(p, x, y, q)$ from states p to q with an input label x and an output label y , a total path weight $[T](x, y)$ can be defined [Mohri, 2004; Cortes et al., 2004] as:

$$[T](x, y) = \bigoplus_{\phi \in \pi(I, x, y, F)} \lambda[s[\phi]] \otimes w[\phi] \otimes f[k[\phi]], \quad (2.19)$$

where $w[\phi]$ is the total weight for the path ϕ using the \otimes -multiplication operation over all transitions:

$$w[\phi] = \bigotimes_{i=1}^n w[e_i]. \quad (2.20)$$

In the tropical semiring weighting system, the \otimes -multiplication operation denotes the sum (using the $+$ operation) of all weights along a given path and the best path

(using the \oplus -operation) is computed using a shortest-path algorithm. The work shown in [Mohri, 2002] introduces a new algorithm to find shortest paths through weighted transducers defined over any type of semiring listed in Table 2.1. This *Generic-Single-Source-Shortest-Distance algorithm* covers a general framework of shortest distance algorithms with *Bellman-Ford* and *Dijkstra* being special cases of the generic algorithm based upon using the \otimes and \oplus operations defined by the semiring framework.

2.6.1 Combinatorial Operations

Weighted transducers can be combined together to form larger, more complex models. Some of these binary and unary techniques have been previously defined for other applications (such as automata theory) and adopted to work with the much richer weighted transducer models [Mohri, 2004]. This section explores some of these operations, namely; *composition*, *union*, *epsilon removal* and *closure*.

2.6.1.1 Composition

The *composition* operation (denoted by \circ) provides the ability to combine multiple transducers using the binary relationship between the input and output symbol domains. Simply, if the transduction $a \rightarrow p$ is performed by transducer T_1 and the transduction $p \rightarrow x$ is performed by transducer T_2 , then $T_1 \circ T_2$ (i.e. the transducer built from the composition of T_1 and T_2) models the transitive transduction $a \rightarrow x$. Many composition algorithms for weighted automata have been proposed to improve performance and provide more functionality [Allauzen et al., 2009; Oonishi et al., 2008; Cheng et al., 2007; Hori et al., 2004].

WFST composition has been used to perform complex translations in speech recognition, natural language processing, computational linguistics and many other applications [Roche and Schabes, 1997; Karttunen, 2001; Morales and Cox, 2008; Mohri, 1997]. Successful work has composed transducers together to form a multi-

level *cascade* that models different aspects to aid the recognition process. The applications of WFST composition are described in more detail in Section 3.6.

The weight between two composed weighted transducer paths, with a transitive relationship ($T_1(a, p)$ and $T_2(p, x)$) can be defined as:

$$[T_1 \circ T_2](a, x) = \bigoplus_p T_1(a, p) \otimes T_2(p, x); \quad (2.21)$$

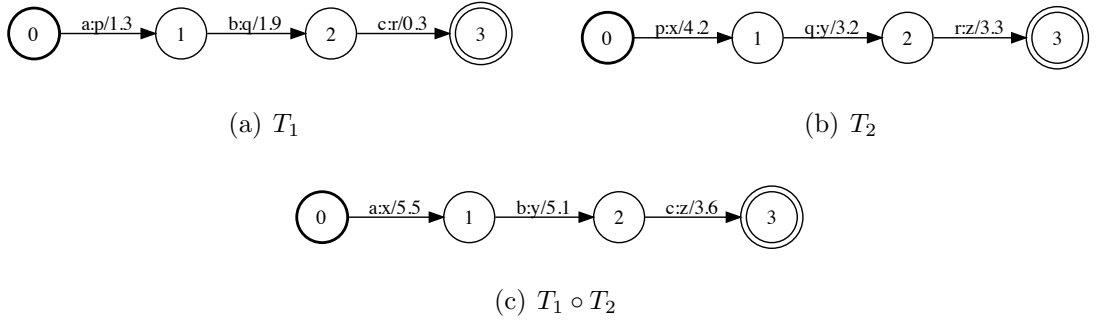


Figure 2.16: (a) T_1 is a weighted transducer with an output symbol domain that matches the input domain of T_2 , (b) T_2 is a weighted transducer with an input symbol domain that matches the output domain of T_1 , and (c) $T_1 \circ T_2$ is the resulting weighted transducer after T_1 is composed (\circ) with T_2 .

2.6.1.2 Union, Epsilon Removal, and Closure

The *union* (or sum) operation uses a combination of multiple transducers. The input string is accepted by the union of two finite state transducers ($T_1 \cup T_2$) if the string is accepted by either of the original transducers (for example T_1 or T_2) [Mohri et al., 2002]. Figures 2.17(a), 2.17(b) and 2.17(c) show three simple weighted transducers, each accepting one string. The union (shown in Figure 2.17(d)) accepts all three strings from the original weighted transducers with their respective arc weights included. Unfortunately, the weighted transducer union algorithm introduces epsilon (‘ ϵ ’) state transitions into the resulting $T_1 \cup T_2 \cup T_3$ transducer. Therefore, it is necessary to remove these and re-configure the transducer topology to improve search

and combinatorial efficiency. The lazy epsilon (ϵ) removal algorithm for weighted automata was introduced in [Mohri et al., 2000] and produces an equivalent weighted transducer as shown in Figure 2.18.

The union between two weighted transducers uses the \oplus semiring operation to combine weights for a given path (x, y) that is common between the two WFSTs: T_1 and T_2 :

$$[T_1 \oplus T_2](x, y) = [T_1](x, y) \oplus [T_2](x, y) \quad (2.22)$$

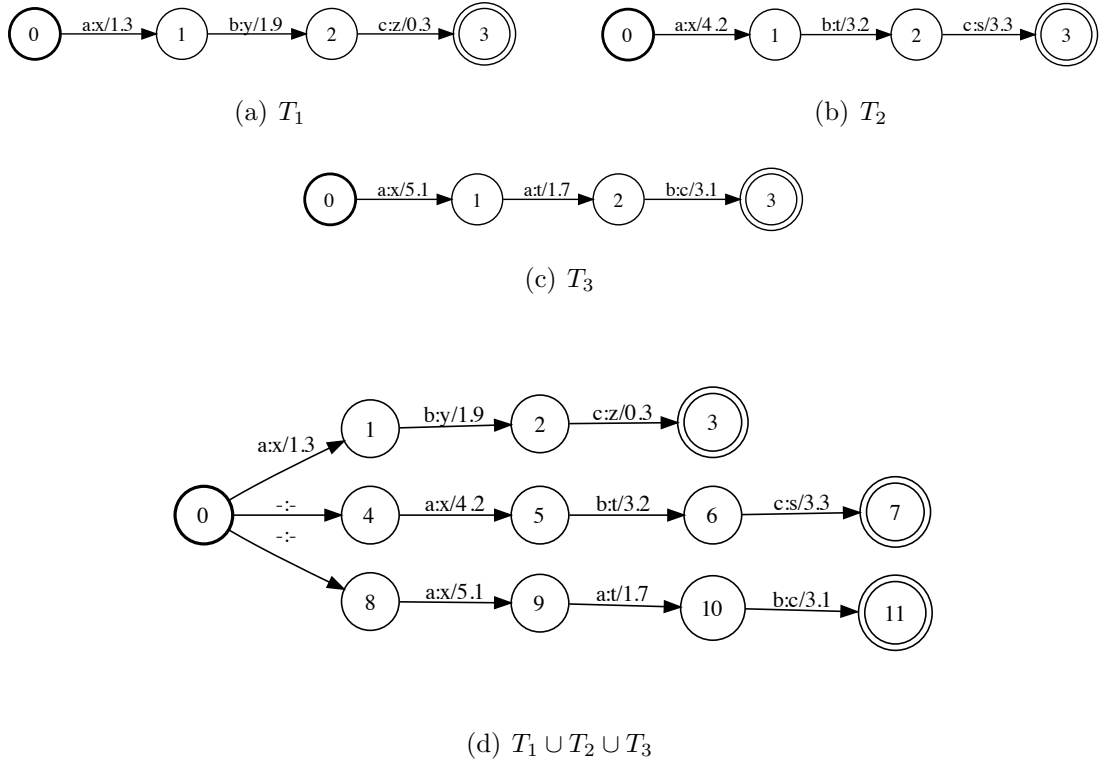


Figure 2.17: (a) T_1 is a single-path weighted transducers, (b) T_2 is an alternative single-path weighted transducer, (c) T_3 is an alternative single-path weighted transducer, and (d) $T_1 \cup T_2 \cup T_3$ is the resulting weighted transducer after performing the union operation on T_1 , T_2 , and T_3 .

When modelling speech confusions in terms of *insertions*, *substitutions* and *deletions*, it is important to introduce the possibility of multi-term output. The *closure* operation introduces additional arcs to enable sequences of output symbols to be duplicated using the work introduced in [Kleene, 1956]. The transducer shown in

Figure 2.18 has three unique paths, each with three transitions. The equivalent *closed* transducer is shown in Figure 2.19 where additional ‘free’ transitions are added to allow for any path to be repeated.

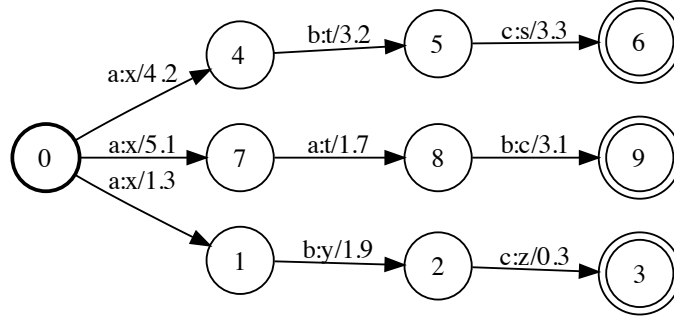


Figure 2.18: ϵ -removal of the transducer $T_1 \cup T_2 \cup T_3$ pictured in Figure 2.17(d). Here, epsilon transitions (represented by the ‘-’ symbol in Figure 2.17(d)) exiting the start state (0) from the original transducer are removed, ensuring maximum traversal efficiency when it is required. For much larger transducers, ϵ -removal can significantly improve composition and shortest-path operational efficiency.

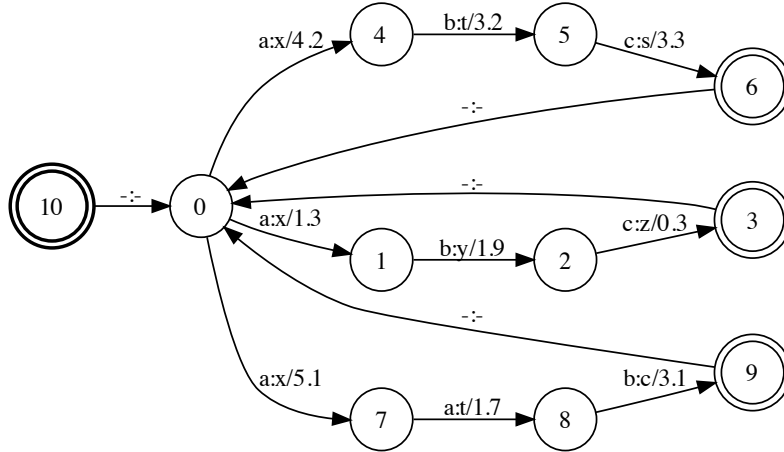


Figure 2.19: A closed weighted transducer derived from the original topology shown in Figure 2.18. The closure operation has introduced epsilon arcs that can be used as a *free* transition back to the start of the sequence.

2.6.2 Pruning Operations

When using combinatorial operations on weighted automata, networks can expand very quickly, causing potential searching issues at decoding time. For such eventualities, *pruning* operations provide a means to prepare an automaton for efficient computation. The dominant forces in weighted transducer pruning, namely, *determinization* and *minimization* are described in this section.

2.6.2.1 Determinization

A finite state automaton is defined as *deterministic* if there are no input epsilon symbols and every state has one or no transition with any given input symbol. The determinization algorithm was first applied to improve efficiency of finite-state automata [Mohri, 1996]. Further work extended this algorithm for weighted transducer applications [Mohri et al., 2002; Mohri, 1997; Mohri and Riley, 1997]. Figure 2.20 demonstrates the process of determinizing a weighted finite state transducer. In this example, Figure 2.20(a) is not deterministic as two arcs with the input symbol ‘a’ are exiting from the starting state (state 0). A new, determinized version is illustrated in Figure 2.20(b) where no two transitions are exiting a state with the same input symbol. [Mohri, 1997] discusses the time complexity of determinization, stating that in the general case it is exponential. However, in the case of speech recognition and *word lattices*, there is significant improvement due to large amounts of redundancy (i.e. many states can be reached by the same set of strings). In this work, a lattice containing 83 million paths is derived from a 2000-word vocabulary speech recogniser. After determinization, efficiency is dramatically improved with only 18 possible paths through an equivalent weighted transducer containing 38 states and 51 transitions [Mohri, 1997].

For successful determinization of a weighted transducer, the algorithm is required to re-distribute weights forward to ensure that an equivalent model is produced. To do this, weights are carried forward through respective paths as shown in the example

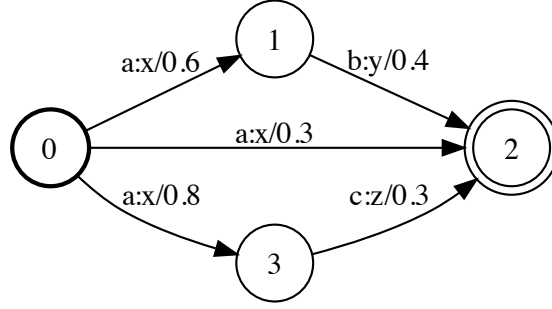
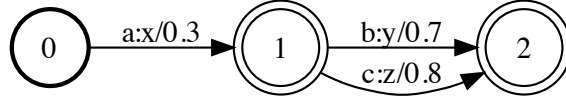
(a) T (b) T_{det}

Figure 2.20: (a) a WFST (T) over the tropical semiring, and (b) T_{det} - a determinized version of weighted transducer T that provides equivalent input and output paths and transductions. From the determinization algorithm, residual weights are carried forward through to ensure original path costs from T are maintained.

in Figure 2.20. Here, the minimum weight for all paths with an input symbol, a is assigned as the new path weight whilst the residual weight from the remaining paths (from state 0 to 1 with a residual weight of 0.3 and from state 0 to 3 with a residual weight of 0.5) are carried forward to the weight on the succeeding arcs.

2.6.2.2 Minimization

The process of *minimizing* a weighted finite state transducer is to enable a new transducer such that it has the least number of states and transitions among all deterministic finite state transducers equivalent to it [Morales, 2009; Mohri, 1997]. The minimization algorithm relies on unique input on the arcs out of any given state

in the original weighted transducer — therefore, it is important to note that this operation can only be performed after determinization. [Mohri, 1997] introduces an efficient algorithm to minimise a weighted finite state automata with a time complexity of $O(m \log n)$ where m represents the number of states and n is the number of transitions in the network. Figure 2.21(a) shows this operation being performed on weighted transducer T . As a result of performing minimization on this transducer T , the number of arcs have been reduced from nine to seven whilst also reducing the number of states from four to three. More specific to the changes in the topology, the minimization algorithm has merged states 1 and 2 together (removing one state), and collapsed the identical transitions between states 1 to 3 and states 2 to 3 into one transition (the transition from state 1 to 2 in the minimised WFST shown in Figure 2.21(b)).

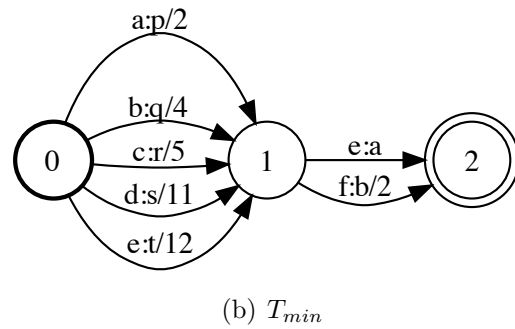
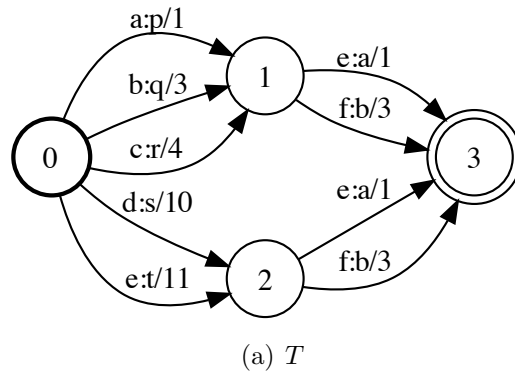


Figure 2.21: (a) a WFST, T , defined over the tropical semiring, and (b) T_{min} - a minimized equivalent of transducer T .

Chapter 3

Literature Review

3.1 Introduction

This chapter presents a review of the current literature relating to auditory and visual speech and confusion modelling techniques used in audio dysarthric speech recognition. Section 3.2 focuses on the production of human speech with a description of the vocal apparatus and the production of words to form sentences with the use of sub-word units (i.e *phonemes*). Section 3.3 discusses previous work in audio-visual speech — a research topic that has attracted audio speech recognition scientists in an attempt to improve speech recognition systems further using visual cues. Section 3.4 explores previous work in lip-reading and visual-only speech recognition (speech-reading), presenting results from recent developments and exploring the state-of-the-art performance. Section 3.5 discusses the work conducted specifically in [Morales, 2009] to improve recognition accuracy of dysarthric speech. Finally, Section 3.6 explores the use of weighted finite-state transducers (as described in Section 2.6) in speech recognition systems before concluding on a review of current literature.

3.2 Human Speech Production

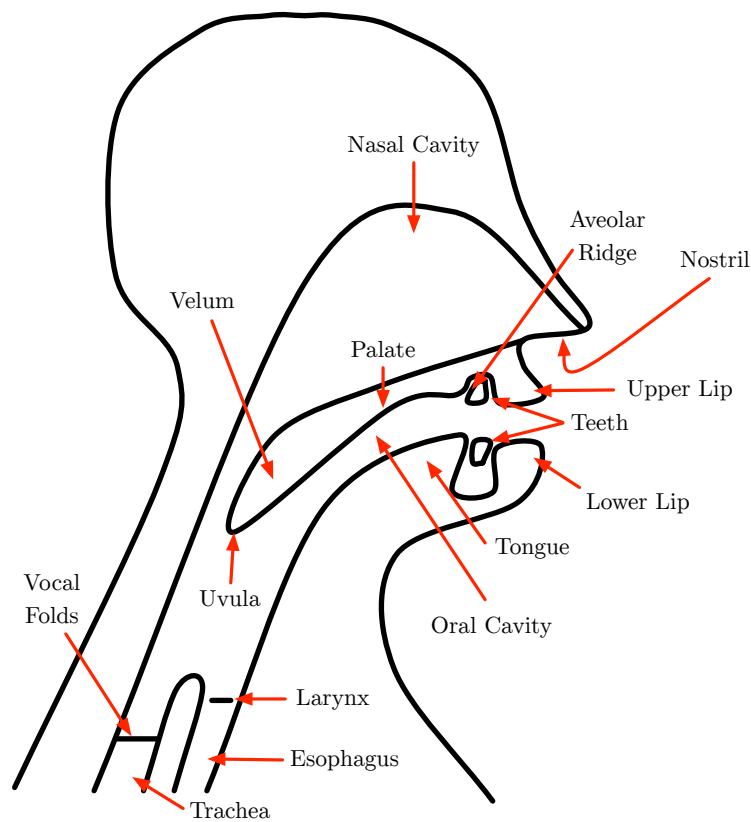


Figure 3.1: A diagram describing the positions of organs in the human speech production system.

Speech is a multi-modal method of communication. The speech signal emanates from the mouth of the speaker as air is exhaled from the lungs before travelling to the ear of the destination [Tatham and Morton, 2011]. As the vocal tract muscles contract and relax, a time-varying signal is produced. For each speech sound produced by the vocal tract, there is a different configuration of the *articulators* that make up the human speech production system. These articulators are: *the vocal folds, tongue, teeth, jaw, velum, lips* and are shown in Figure 3.1. Different articulatory positions can be configured to produce two broad sound categories: *vowels* — produced by a continuous flow of unrestricted air passing from the lungs up to the larynx, and *consonants* — the air flow emanating from the lungs is restricted by a movement by the articulatory organs, causing different sounds to be produced.

Due to the large variation of vocal tract configuration in consonant sounds, there are many different types of sounds that can be produced by adjusting the *manner of articulation* — how the articulators interact with each other in the system, and *place of articulation* — the point at where an obstruction of air-flow occurs to produce the distinctive consonant sound [Ladefoged and Johnson, 2014]. Table 3.1 and 3.2 describe the different types of place and manner of articulation configurations respectively, each with an example phoneme sound highlighted in bold.

Predominantly, speech communication is most informative in the audio domain, with the key ability to distinguish between sounds. However, the visual domain of speech also provides some important cues that can aid understanding. The visual component of speech is composed of the *visible articulators*: the tongue, lips and teeth. Although these are visible articulators, the lips and teeth are only visible when certain sounds are made. The process of decoding speech purely from the information on the visible articulators is known as *lip-reading*. With the complete occlusion of the non-visible articulators and the occasional occlusion of some of the visible articulators, it is common for lip-reading recognition systems to miss cues in the visual domain that are present in the audio domain. This leads to *deleted sounds* in the hypothesised sequence.

Place of Articulation	Place of Air Restriction	Example
Bilabial	Upper and lower lip	<i>bat</i>
Labiodental	Both lips and teeth	<i>fog</i>
Interdental	Tongue between teeth	<i>think</i>
Alveolar	Small gap between the alveolar ridge and tongue	<i>door</i>
Alveopalatal	Small gap between the area behind the alveolar ridge and tongue	<i>check</i>
Velar	Small gap between the velum and tongue	<i>gate</i>

Table 3.1: A list of the different places of articulation in production of consonant sounds. Letters highlighted in bold indicate which sound has a specific place of articulation within a word.

Manner of Articulation	Speech Apparatus Configuration	Example
Plosive	Stop air flow for an amount of time before releasing quickly	<i>bat</i>
Affricate	Stop air flow for an amount of time before releasing to create a friction sound	<i>check</i>
Fricative	Impeding air in the vocal tract to release a friction sound	<i>fought</i>
Nasal	No air escaping from the oral passage but instead from the nasal passage	<i>mum</i>
Lateral	Air being allowed to emanate from sides of the mouth	<i>let</i>
Retroflex	Tongue to the roof of the mouth before retracting	<i>ridge</i>
Semivowel	Small enough gap to differentiate from a vowel	<i>work</i>

Table 3.2: Configurations of different manners of articulation shown with examples.

3.2.1 Phonemes

In linguistics, a *phoneme* is defined as “the smallest contrastive linguistic unit which may bring about a change of meaning” [Cruttenden, 2008]. For example, the words *bat* and *cat* differ in only the first sound (*/b/* or */k/*). Phonemes can be combined together into a time-series to form *words* in a given language. Each language has its own set of phonemes that form the foundation for a vocabulary. Throughout this thesis, we use the BEEP phoneme set [Cambridge-University, 2012], which contains 45 phonemes, each providing a distinctive sound and vocal tract configuration including plosive bilabials such as */p/* to fricative labiodentals such as */f/* and standard open-mouth vowels such as */oh/*. Although these phonemes are considered as isolated units, their target properties (both acoustically and visually) can be severely affected by neighbouring sounds, a phenomenon known as the *coarticulation effect*. The production of speech sounds also introduces different variants, also known as *allophones*. These describe a group of phonemes that have the same phonetic meaning but are produced in a different way. An example of this is the produced */k/* sound in word *calculator* and the same produced sound in the word *skeleton*.

Sounds that are produced by the articulatory process that involve the vibration of the vocal cords are known as *voiced* sounds (e.g. the first sound from the word *zip*). However, there are also sounds which do not require vocal cord vibration, named *unvoiced* sounds (e.g. the first sound from the word *pat*). Phonemes in the English language are typically classified into these two groups.

3.2.2 Visemes

Phonemes are the smallest unit of speech that can change the meaning of a word. Although phonemes are abstract entities, representing the speech signal as a sequence of phonemes and modelling the acoustical properties of these phonemes has formed the basis for successful speech recognition algorithms for the past thirty years. A speaker must be capable of producing sounds that are recognisable as distinct phonemes in order for their speech to be understood. However, because speech is perceived (by the vast majority of people) in audio form, there is no requirement for a speaker's visual signals (e.g. mouth shapes) to form contrastive patterns, and hence there is no natural visual equivalent of the phoneme.

The term *viseme* was introduced by [Fisher, 1968] (made from a combination of the words *visual* and *phoneme*) and is still used in visual speech recognition and synthesis. Usually, a viseme is defined by grouping together a number of phonemes that have a similar visual appearance. Hence phonemes that differ only in a feature that cannot be perceived visually (e.g. in voicing, or in a place of articulation near the back of the mouth) are grouped together as a single viseme. Several many-to-one mappings from phonemes to visemes have been proposed, with much disagreement on the best mapping in the research community [Fisher, 1968; Binnie et al., 1976; Woodward and Barber, 1960; Binnie et al., 1974; Visser et al., 1999; Cappelletta and Harte, 2012; Hilder et al., 2010]. These ambiguities are derived by discrepancies in speaker variation, coarticulation, type of stimuli presented and the phoneme-to-viseme grouping that is being used [Theobald, 2003].

However, although the term viseme, as a visual equivalent of phoneme, is in

common use, it is controversial, because there is no evidence that people are capable of perceiving such visemes as separate visual entities that discriminate between words.

3.3 Audio-Visual Speech Recognition

Speech is predominantly a multimodal method of communication. For face-to-face interaction, humans use a combination of cues from both the audio and visual modalities. With most of the articulatory information in the audio signal, speech recognition research has focussed on the use of the audio signal only for over 30 years. However, recent work in ASR has used the visual signal as an additional source of information to improve the performance of state-of-the-art ASR systems.

The addition of visual information is effective in environments where noise in the audio is having an adverse effect on recognition [Almajai and Milner, 2009]. However, certain acoustic sounds are easier to recognise in the visual modality. For example, place of articulation information is sometimes easy to see, and this can reduce confusion. For instance, /b/ (a bilabial) and /d/ (an alveolar), or nasal sounds like /m/ (a bilabial) and /n/ (an alveolar) are visually distinct but often confused in the audio domain [Potamianos et al., 2004]. The study conducted in [Liang et al., 2002] explores the use of the visual modality to assist audio ASR in noisy environments. The results show that, in noisy environments, error rates in audio ASR can be as high as 80% (at a 0dB signal-to-noise ratio) whereas, audio-visual recognition error rates are much lower at 25%. From these experiments, it is clear that the ASR recognition performance is being improved significantly because the signal from the visual domain is not affected by acoustical noise.

Audio-visual speech recognition (AVSR) has been an interest of speech research for many years [Almajai and Milner, 2008, 2009; Dupont and Luetttin, 2000; Neti et al., 2000; Potamianos et al., 2003a, 2001; Tomlinson et al., 1996; Petajan et al., 1988; Potamianos et al., 2003b]. The first speech recognition system to use informa-

tion from the visual domain was introduced in [Petajan, 1984], using a isolated-word dataset containing 100 words including simple digits and letters. The features were extracted using simple binary images of the mouth and lips, determining shape parameters such as the height, width, area and perimeter of the mouth. The results produced in [Petajan, 1984] sparked an interest in the speech community for audio-visual speech recognition research to improve the accuracy of existing audio ASR systems. Figure 3.2 shows the main components of a typical audio-visual speech recogniser. In this system, the front-end feature extraction for the audio modality can use established techniques that have been employed in state-of-the-art audio ASR systems. However, the feature extraction method for the visual modality is a topic of many strands of research [Almajai and Milner, 2009; Zhi et al., 2004; Dupont and Luettn, 2000].

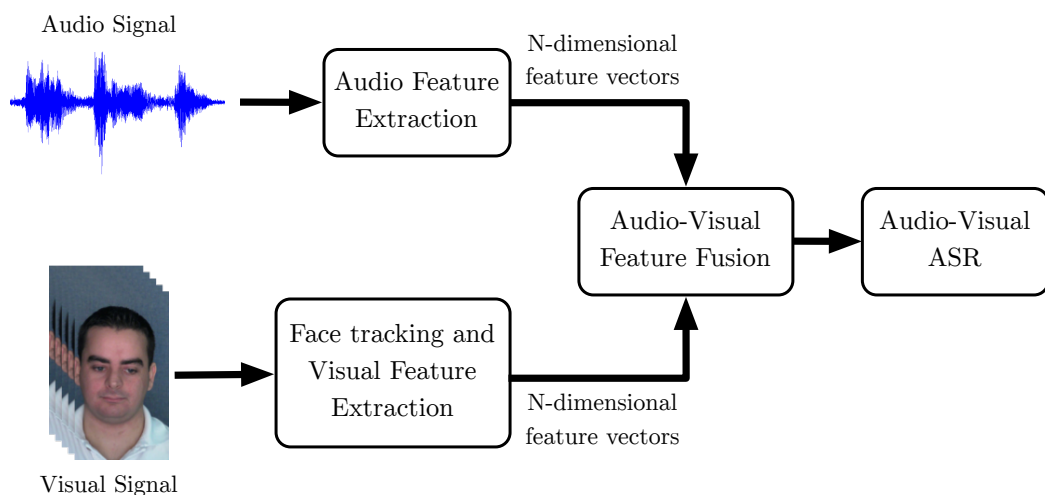


Figure 3.2: A diagram describing the components in an audio-visual speech recognition system. Feature extraction of the audio and visual signal is performed independently before using a fusion method to combine the feature sets to aid recognition.

Various combinations of features have been used in AVSR. The original geometric features used in AVSR such as mouth and lip height, width, area and perimeter have been successfully applied in previous work [Zhi et al., 2004; Sujatha and Santhanam, 2010; Cetingul et al., 2006]. However, such features only represent changes in shape, providing no information about important visual cues such as the appearance of teeth and rounding of lips. Therefore, an alternative appearance-based method

can be used — specifically capturing information contained within the region of interest. The appearance information can be used in isolation or in combination with the shape information to build an active appearance model (AAM) [Cootes et al., 1998]. This process is detailed in Section 2.2.1.

The fusion between the audio and visual modalities can be performed using two different methods: *feature fusion*, and *decision fusion*. In feature fusion, the feature vectors are combined (e.g. concatenated) to produce features that describe both modalities whereas decision fusion uses two separate classifiers (one for audio and another for video) and combines them to model the reliability of both modalities [Potamianos and Graf, 1998; Teissier et al., 1999; Hennecke et al., 1996; Bregler et al., 1993; Adjoudani and Benoit, 1996]. For an overview of these algorithms, see [Potamianos et al., 2004].

3.4 Visual-Only Speech Recognition

When the audio signal is not available or is severely corrupted, visual cues become the only source of information, and in this case, the listener or computer system is performing *lip-reading*. The lack of information obtainable from the visual modality alone has led to automated lip-reading systems achieving low performance [Lan et al., 2010]. As described in Section 3.1, the production of human speech employs the use of all of the articulators to produce the sound. With some of these organs either completely hidden or occasionally obfuscated from the camera’s view, many phonemes are deleted in visual speech recognition. Furthermore, sounds such as /p/, /b/, and /m/ (all bilabials) are impossible to separate in their visual appearance, introducing patterns of substituted phonemes.

A recent study presented in [Newman et al., 2010] uses electromagnetic articulography (EMA) data, providing 2-dimensional feature points to model the position of a speaker’s speech apparatus. Each articulator is withheld in-turn starting from the back of the apparatus (the velum) and eventually reaching the front (the upper

lip). The results from this study are presented in Figure 3.3. As each feature is removed, the lip-reading performance decreases monotonically, suggesting that the recognition accuracy of visual speech will always be inferior to audio ASR due to the frequent occurrence of events that can not be observed on the lips. However, many deaf people are capable of performing lip-reading with enough accuracy to function in a hearing world. The outcome from the experiment conducted in [Newman et al., 2010] confirms that human speech recognition is heavily dependent on knowledge of the context of the discourse. The high level constraints are influenced by the context and expectations of particular words, whereas the lower-level constraints consider the legal words and phrases from the supplied information using the language model. Both of these constraints are used to help the lip-reader decide on the most likely hypothesis.

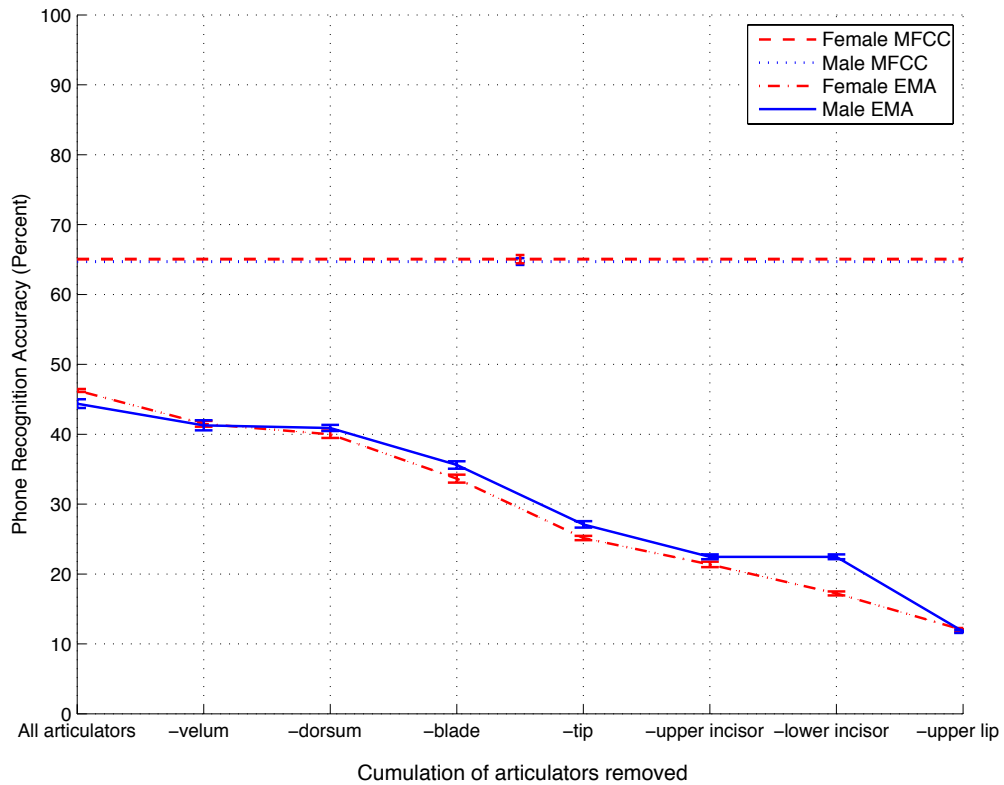


Figure 3.3: Speaker-dependent phone accuracy results for audio and visual speech recognition using different combinations of articulatory features. Each point along the x axis represents a cumulative removal of the information that is obtained from the specified articulator [Newman et al., 2010].

The study conducted in [Matthews et al., 2002] evaluates different visual features for automated lip-reading. In this work, they evaluate each set of features using their own, specially recorded audio-visual database consisting of 780 utterances which was divided into training (520 utterances) and testing (260 utterances). Each utterance consisted of a letter being recited from the alphabet and all videos were of low-resolution grayscale quality. Word (letter)-level HMMs were constructed and the different feature sets were evaluated by the conventional word accuracy score. Best results produced a multi-speaker word accuracy of 41.9% using AAM features (shape and appearance information) and 26.9% using shape-only features.

[Lan et al., 2009] presented a comparison between AAM-based features, Sieves and DCT-based features. In this work, the constrained GRID database [Cooke et al., 2006] was used, consisting of 1,000 utterances from 15 speakers. Viseme HMMs were built using the same procedure as adopted by previous state-of-the-art lip-reading systems. For this speaker-independent task, AAM-based features performed best, achieving 65% word accuracy. Although this is a promising result, audio ASR systems perform significantly better, reporting error rates because the recognition accuracies are so high [Cooke et al., 2006].

Further work conducted in [Lan et al., 2010] provides a comparison of systems using different audio and visual features. The audiovisual corpus consisted of 200 sentences being recited by 12 speakers. The extracted features are passed through a network of viseme HMMs (described in Section 2.4) to perform recognition. The reported viseme accuracies (i.e. an accuracy measure based on counting viseme tokens, not on whole word tokens) demonstrate the significant divide between audio and visual ASR. As expected, the audio task using MFCC features obtained a high accuracy (80%) — a figure that can be expected from state-of-the-art audio ASR systems. However, using AAM features (consisting of shape and appearance information), the viseme accuracy drops significantly to 35%. Although viseme accuracies provide an indication of how well a recogniser is performing, there is no clear translation into word accuracy results.

[Cox et al., 2008] presents word-level accuracies based on a simple isolated letter task. Here, each of the 26 letters of the alphabet are recited for seven repetitions. AAM, Sieve and MFCC (audio) features are extracted and used independently to train word-level HMMs (i.e. an HMM for each isolated letter) for recognition. The results presented from this work show some promise. For the single-speaker case, results using visual AAM features are consistently over 80%. Similarly, for multi-speaker, reported accuracies are also consistently high at over 80%. However, in the speaker-independent task, AAM visual accuracy drops significantly to below 10%. When considering this is a highly-constrained task, it is clear that, in the speaker-independent task, inter-speaker variability is having a detrimental affect on accuracy. Logically, this is due to the large variation in a speaker’s mouth shapes both when at rest and during speech production. This work also presents findings from analysis of the MFCC (audio) and AAM (visual) features, illustrating that AAMs are almost entirely speaker-dependent, unlike its audio equivalent.

Finally, [Hilder et al., 2009] presents a comparison between human-based and machine-based lip-reading. This work compares the use of shape-only features (point-lights projected on a black surface providing positional data of the main landmarks of the face) and shape and appearance features (describing texture variation). The first part of the experiment involved using a small, restricted vocabulary (containing the letters A through F). Using only shape-based information, the machine-based approach is significantly superior to humans with a machine multi-speaker accuracy of 74.29% and a human letter accuracy of 42.9%. [Hilder et al., 2009] concludes that this is likely due to the amount of confusion between the final vowel sounds in three of the letters: C, D, and E. When using shape and appearance (i.e. including the texture information), human performance improved to 71.6% but was still slightly inferior to machine-based performance, obtaining 75.24%.

A further experiment explored the advantages of having data to train the machine-based lip-reading system. Here, a new dataset is recorded consisting of one speaker reciting 225 monosyllabic words and 120 nonsense words consisting of either a vowel-

consonant-vowel (VCV) or consonant-vowel-consonant (CVC) structure. For the human recognition task, 19 people were tested on their lip-reading ability before and after training. The results show that, for the simple monosyllabic word task, humans were able to improve their scores, with a word accuracy increase of 3.95%. In comparison with the machine-based approach, humans were able to perform significantly better at recognising whole words (word accuracy scores of 18.42% for humans and 3.75% for machines). However, for the recognition of phonemes, human scores were poor (31.60%) whilst machine-based scores were very high (80.27%). As discussed in [Hilder et al., 2009], this probably is due to the prior knowledge of the language and the constraints that the language model brings to allowed words, knowledge which is not available to the machine-based system.

3.5 Dysarthric Speech Recognition

Dysarthria is a disorder affecting the control and coordination of articulatory muscles, causing less intelligible speech. The muscles affected may include the lungs, pharynx, soft palate or front-end articulators such as the lips, teeth or jaw. Poor control over articulatory muscles can cause speech to become slurred, introducing errors in the produced speech in the form of substitutions, insertions, and deletions. The work introduced in [Morales, 2009] attempts to improve the recognition accuracy of dysarthric speech recognition by studying confusion patterns in phoneme substitutions, deletions and insertions that are a direct result of poor muscle control. Initial work introduced in [Morales and Cox, 2007] explores the use of *metamodels* to model the speaker’s confusion matrix. [Morales and Cox, 2007] describes the meta-model as a structure similar to an HMM whereby a sequence of states are connected with weighted transitions which model the possible phoneme sequence (including insertions) to be produced by the recogniser. These metamodel structures are used to model the observed confusion patterns of a speaker and used to correct decoded phoneme sequences. For low-intelligibility speakers, accuracy increased significantly using this method due to the strong substitution patterns observed by the system

during training. However, this accuracy gain is not observed in more intelligible dysarthric speakers. Furthermore, the modelling of *deleted* phonemes is difficult within the metamodel framework so they are usually modelled using an additional phoneme class labelled DELETION [Morales and Cox, 2007] — another limitation which could affect recognition accuracy for certain speakers. [Morales, 2009] continued by exploring the use of a weighted finite-state transducer (WFST) cascade to correct decoded sequences. Unlike metamodels, WFSTs have the advantage of modelling a deletion (or an insertion) with the use of an epsilon arc (described in Section 2.6.1.2). WFST cascades require a large amount of phoneme-level confusion information in order to estimate their parameters. Therefore, the work extends the confusion model to use non-negative matrix factorisation (NMF) to build a less sparse confusion matrix estimate for a dysarthric speaker. Using these techniques, Morales was able to improve recognition of dysarthric ASR for both low- and high-intelligibility speakers from 59% to 74% word accuracy.

The work presented in [Morales and Cox, 2008] was predicated on the patterns of substitutions, deletions and insertions that are produced by a dysarthric speaker. In this way, the requirements of a confusion model for dysarthric speech can be compared to that of a lip-reading system. With information that is not observed on the visible articulators in speech, many acoustical sounds are incorrectly recognised because they have identical lip shapes (e.g. /b/ and /m/). Deletions are also a common occurrence in automated lip-reading, with missing information from the velum and vocal cord in the visual speech signal. With the epsilon (i.e. the ‘free move’) arc, WFSTs can model deleted sequences easily. The comparison between confusion modelling for dysarthric ASR and automated lip-reading provides the main motivation for this work.

3.6 Weighted Finite-state Transducers in ASR

Weighted finite-state transducers (WFST) are described in Section 2.6. Here, the idea of *composition* was introduced as a method to perform complex transitive translation between symbols. This has significant relevance for applications to ASR, with much work focussing on using finite-state theory to improve recognition performance [Mohri et al., 2005; Mohri, 1997; Pereira and Riley, 1997]. The *ASR composition cascade* consists of a set of WFSTs that, when composed together, form an efficient speech recognition system. [Mohri, 1997] defines each of these models as the following:

- O : the acoustic observations,
- A : an acoustic model to map the acoustical observations to context-dependent phones,
- C : a model to map context-dependent to context-independent phones,
- L : a lexicon (pronunciation dictionary) to map the sequences of context-independent phones to words,
- G : the sentence-level grammar model to map words to sentences.

For decoding using weighted transducers over the tropical semiring (as described in Section 2.6), a hypothesis is produced as the path with the shortest cost (denoted by π) through the set of composed weighted transducers:

$$\pi\left(O \circ A \circ C \circ L \circ G\right) \quad (3.1)$$

In general, these composed networks can expand exponentially in size through the composition cascade, introducing space and time efficiency issues. Therefore, it is necessary to prune networks at each level of the composition cascade using *determinization* and *minimization* for weighted automata (described in Sections 2.6.2.1 and 2.6.2.2 respectively).

The WFST approach to ASR can be adapted to incorporate confusion patterns. Substitutions, insertion and deletion patterns can be modelled explicitly using a dedicated *confusion weighted transducer*, taking advantage of the transducer’s ability to translate symbols given a specified weight. This technique has been applied to open-vocabulary spoken utterance retrieval [Hori et al., 2007], keyword-spotting [Karanasou et al., 2012], and dysarthric ASR [Morales and Cox, 2008]. In the work presented by [Morales and Cox, 2008], observed confusion patterns are used to build a confusion matrix modelling the substitutions, insertions, and deletions of a dysarthric speaker. In this work, the four weighted transducers that are included in the composition cascade are defined as: P^* , C , L , and G . Each weighted transducer is described in detail with an example in Figures 3.4, 3.5, 3.6, and 3.7.

- P^* : the decoded sequence of phonemes produced by the recogniser,

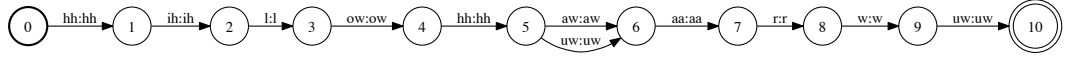


Figure 3.4: P^* - the phoneme-level hypothesis produced by the state-of-the-art ASR system. This symbol sequence is most likely to contain incorrect (noisy) entries that can be corrected in the cascade.

- C : a transducer modelling the possible confusions and associated probabilities of insertions, substitutions and deletions. The confusion matrices are used in this process to model substitutions whilst information regarding patterns of deletions and insertions are observed from the training sequences and modelled using additional columns in the confusion matrix (see Section 2.5),

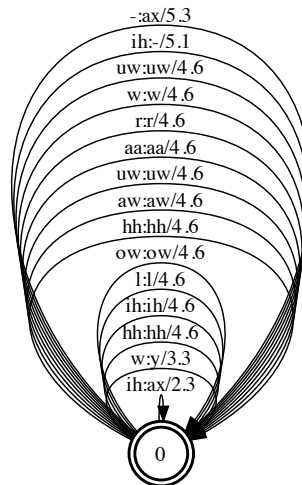


Figure 3.5: C performs the substitutions, insertions and deletions according to the confusion patterns in the estimated confusion matrix. In this example, most translations are taken from the diagonal elements of the confusion matrix (‘null’ substitutions). Off-diagonal substitutions (recognising $/w/$ instead of $/y/$, an insertion of $/ax/$ and a deletion of $/ih/$) are also modelled here.

- L : the lexicon (dictionary) transducer - mapping sequences of legal phoneme sequences into complete words,

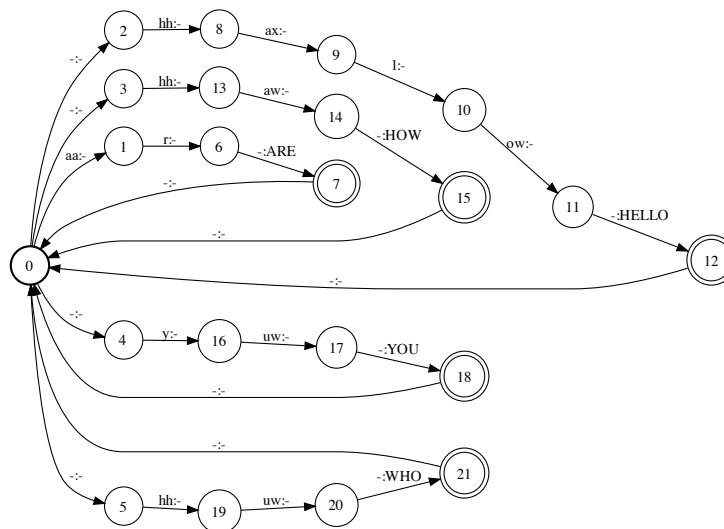


Figure 3.6: L - the lexicon WFST maps sequences of ground-truth phonemes to whole words. This example models a five-word vocabulary — $\{\textit{HELLO}, \textit{HOW}, \textit{ARE}, \textit{YOU}, \textit{WHO}\}$. The input alphabet is a set of phoneme strings whilst output alphabet on the final arc of each path are words. In most cases, words are treated as equally weighted entities in a non-weighted transducer.

- G : defines a sentence-level grammar which has been defined by the language model.

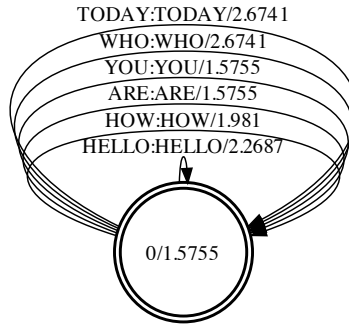


Figure 3.7: G - a sentence-level language model can take the form of an n -gram model (described in Section 2.3). This example is a unigram (1-gram) language model, allowing any word to follow any other word with an associated cost (defined over the tropical semiring).

This framework provides a powerful decoding system whereby speaker confusions can be corrected before the enforcement of the language model. However, as with the standard WFST cascade used for ASR tasks, the composition of these weighted transducers can expand the resulting network by exponential growth. Depending on the topology and complexity of the confusion transducer (C), these problems can be overcome with the use of determinization and minimization. The work presented in [Mohri and Riley, 1997] evaluates the efficiency gain that can be obtained by performing determinization and minimization in a LVASR task. They use word networks taken from the ARPA ATIS task with a 1500-word vocabulary. For an example sentence (one of the smallest in the corpus), the network contains approximately 151 million paths. Using only determinization, the number of states can be reduced by a factor of three and the number of transitions also reduced by a factor of nine. After determinization and minimization, the network is optimised further, producing a pruned network that contains five times less states and 17 times less transitions than the original network [Mohri and Riley, 1997].

In this thesis, we use the OpenFST library [Allauzen et al., 2007] to construct, manipulate and compose WFSTs. We also use an additional tool, OpenGRM [Roark

et al., 2012], to construct word-level n -gram language models which are used in the composition cascade.

3.7 Conclusions

This chapter has presented a review of the current literature in fields related to this work. Firstly, the human speech production system is described — focussing on the adjustable parameters that produce distinctive sounds (i.e. the articulators). From understanding the physiology of the speech system, phonemes provide a quantifiable unit of acoustic sound, with a sequence of phonemes being concatenated together to form words in any language. Although the phoneme provides a strongly-defined unit of acoustical speech sounds, the lack of information in the visual modality introduces phoneme confusions. For phonemic sounds that are similar in their visual appearance, visemes have been defined as a visual equivalent to the well-defined audio phoneme. However, finding a strongly defined phoneme-to-viseme mapping has been the subject of much research for many years.

Recent work in multi-modal ASR has sparked an interest in the visual speech signal, with visual cues providing an additional source of information. Under noisy acoustical conditions, information extracted from the visual modality can provide important cues to improve ASR decoding accuracy. The work presented in [Liang et al., 2002] shows that ASR word error rates decrease dramatically when noise is introduced into the acoustic signal where the visual signal is unaffected. Automated lip-reading can be considered as an extreme case of audio-visual speech recognition, where the auditory contribution of speech is unintelligible as a result of a noisy channel [Stork and Hennecke, 1996].

Recent work in visual-only ASR has mimicked that of a conventional audio ASR system. Visual feature extraction methods have been compared in previous literature, with AAMs being the best compact representation of the lip shape and appearance. Despite some initial promise, automated lip-reading systems have yielded

low accuracies, with most lip-reading systems either reporting viseme accuracy or performing recognition on simple, language-constrained tasks.

The structure of sound-based confusions in lip-reading can be compared to that of a dysarthric speaker. With very little control over articulatory muscles, dysarthric speakers introduce confusability into their speech sounds in the form of substitutions, insertions and deletions. [Morales, 2009] presents work that focuses on modelling these confusion patterns to improve decoding of dysarthric speech. The initial work uses a series of *metamodels* to correct the confusions occurring in speech production. However, due to the structure of these models, limitations exist for modelling deleted sounds. A more flexible solution is proposed, using a powerful network of weighted finite-state transducers to perform corrections. WFSTs provide a framework to model deletions using the ϵ -symbol transition.

Chapter 4

Description of Datasets

4.1 Introduction

Collecting audio-visual speech corpora is a time-consuming process. Unlike audio-only speech corpora, these datasets need to be recorded in an environment that is appropriate for both modalities. Consequently, there are few publically-available audio-visual speech corpora that can be used for lip-reading experiments, and most include some undesired variability that could affect the visual signal. To fulfill the requirements of this work, two new datasets have been specially recorded.

Section 4.2 describes a new isolated word dataset that is designed to concentrate on modelling the confusion patterns in visual speech on a small scale. A small vocabulary consisting of 211 words is carefully selected. Words are selected to cover a broad range of phoneme bigram frequencies.

An isolated word task provides a simple framework to concentrate on techniques to model visual confusions in speech. However, it does lack the richness that continuous speech provides. Section 4.3 describes an existing dataset used for the language independent lip-reading (LILiR) project and introduced in [Lan et al., 2010]. With 200 sentences of continuous speech, the LILiR dataset provides a rich context, and more natural speech from which to build confusion models.

Finally, Section 4.4 describes a new, much larger dataset that has been specially recorded for the purpose of this work, consisting of 3000 sentences taken from the Resource Management corpus [Price et al., 1988], a standard corpus used previously in ASR work. This is the largest-known single-speaker, continuous dataset that has been designed for automated lip-reading tasks.

4.2 ISO-211 - An Isolated Word Dataset

An isolated-word dataset has been specially-recorded for the purpose of this work. In this work, we are attempting to construct confusion models of phonemes. For our initial efforts, we need as little influence as possible from a language model and from complex coarticulation effects that can occur across several words. Hence, we start with an isolated word task.

Co-articulation has a significant influence of the lip movements of speech [Benguerel and Pichora-Fuller, 1982; Cohen and Massaro, 1993; Fowler, 1984]. To avoid coarticulation having an effect on the initial experiments, words were chosen from the BEEP dictionary [Cambridge-University, 2012] with careful consideration to provide the maximum coverage of bigram phonemes over the total bigram phoneme count observed in the corpus. Figure 4.1 shows the distribution of bigram occurrences in the dictionary (consisting of 257,059 words). As expected, the majority of the dictionary can be covered by selecting a relatively small number of bigrams. For example, by taking only 800 bigrams (just over half of the total amount), the dictionary coverage can be as high as 98%. To minimise the amount of recording required by the guest subject, it was important to find a good compromise between maximising the number of bigram phonemes observed and the total percentage of the dictionary covered by these bigrams. Hence, it was considered acceptable to cover 80.2% of the dictionary with a total of 211 words. These words can be found in Appendix A.

Figure 4.2 compares bigram coverage in the BEEP dictionary with coverage in

the new isolated word corpus. Although the frequencies are more well-defined in the BEEP dictionary, it is clear to see similarities in bigram relationships between these two sets of data. Both sets contain higher frequencies of *vowel* followed by *consonant* and *consonant* followed by *vowel*. Patterns away from these highly populated areas are also similar between the two sets, suggesting strong similarities in the nature of bigram occurrences. Ultimately, Figure 4.2 shows that the extracted words used in the new isolated word dataset provide a good representation of the BEEP dictionary data but with very few words.

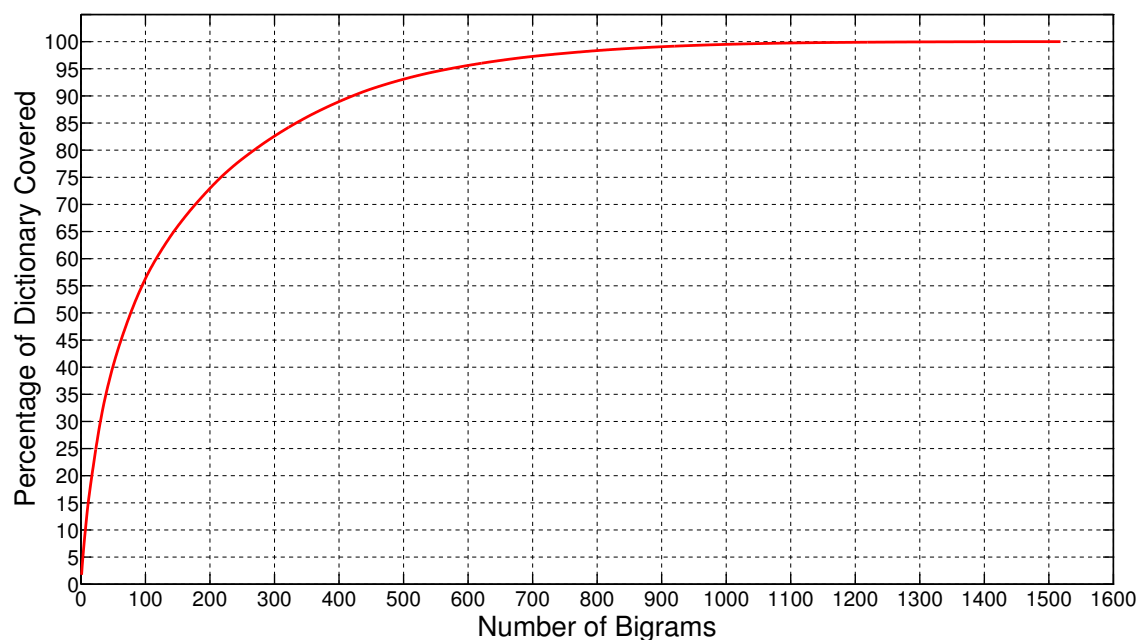


Figure 4.1: ss from the BEEP dictionary [Cambridge-University, 2012] with the percentage of the dictionary covered.

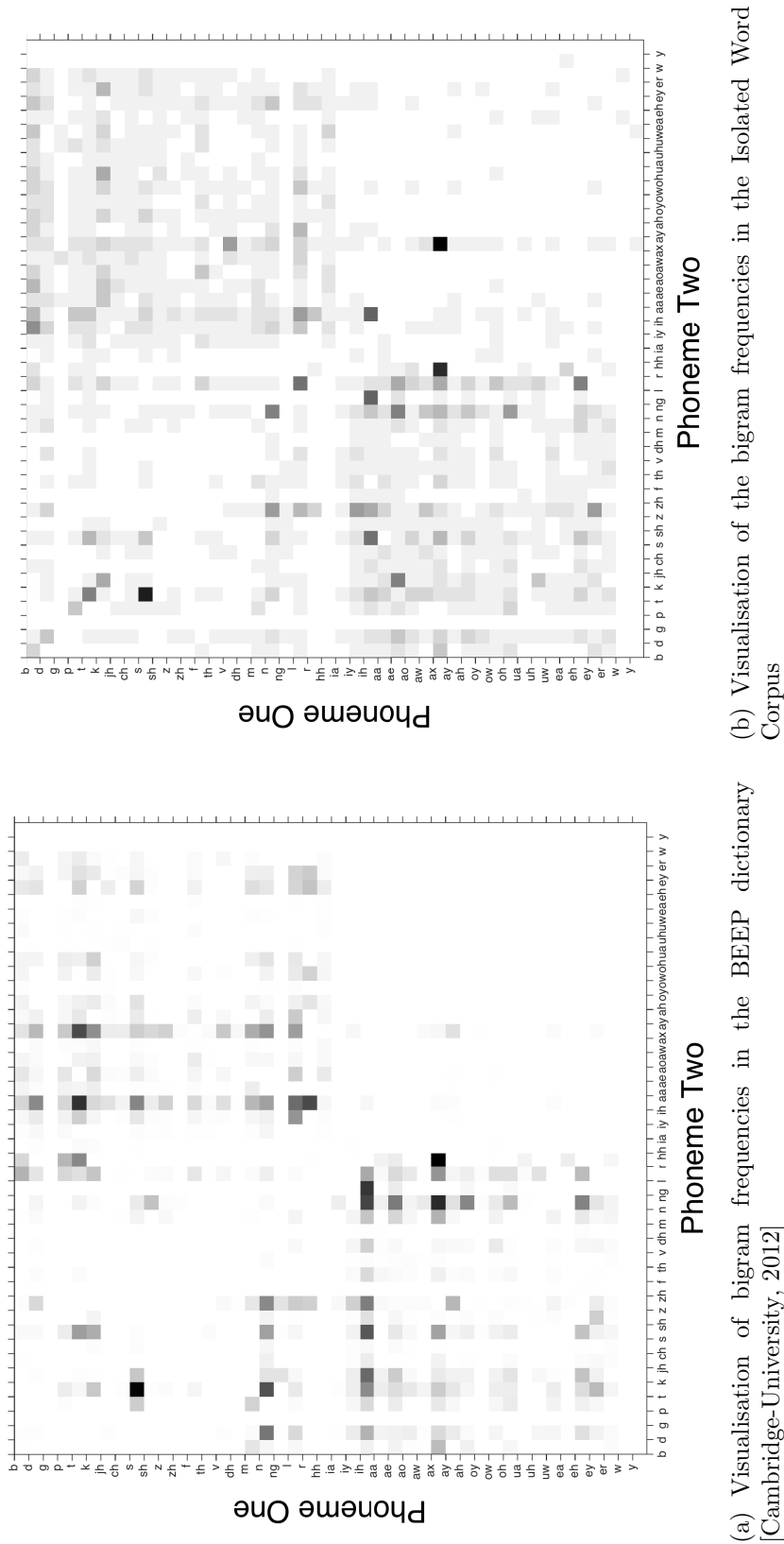


Figure 4.2: A comparison of number of bigram occurrences covered. (a) shows the bigram frequencies taken from the BEEP dictionary (257,059 words) and (b) shows the bigram frequencies from the isolated word data set consisting of 211 words.

The data were captured from a single female speaker in a specialised recording environment with the equipment setup as pictured in Figure 4.3. Video was recorded using a Sanyo Xacti camera in portrait orientation at 1080 x 1920 pixel resolution using progressive scan at a sampling frequency of 59.94 frames per second (fps). Audio was captured using a clip microphone recording at a sampling frequency of 48kHz. The microphone was positioned on the shirt collar of the subject (as pictured in Figure 4.4). To provide an easier video to track and to avoid any potential issues with the camera focus, video was recorded from the whole face with a full-frontal pose. Six repetitions were captured and words were mixed between folds to avoid any session bias. Although this mix was performed, no fold obtained more than one repetition of a word. This was done to maintain an even word and phoneme distribution across all folds.

The complete dataset consists of six repetitions of each word. To eliminate any bias towards the order, words were randomised for each repetition. Each word was displayed in turn on a laptop screen positioned in front of the subject and there was a five second interval to pronounce the word before prompting the next word with a beep tone. The subject was advised to sit as still as possible and to pronounce the words to the best of her ability. Recording repetitions lasted approximately 30 minutes each and were collected over three separate sessions. As expected, during this time, the subject occluded the areas of interest for a small number of frames, but, these occurred mostly during the silence or sound of the beep and so did not affect the resulting data. Words where the subject did occlude the lip region were removed from the dataset. A description of the recording plan can be found in Table 4.1.

[Newman, 2011] explores the use of three different video resolutions (640 x 360, 1080 x 720 and 1920 x 1080) in visual-phone lip-reading recognition and find no gain in accuracy when using higher resolutions. Therefore, to improve the time efficiency of the feature extraction process, videos were down-sampled to a third of their original resolution to 360 x 640 pixels. Firstly, a selection of between 20 to

30 frames were selected from each of the six session videos for hand-labelling. A total of 111 2D points were labelled over the whole face to ensure stability when tracking using the inverse compositional project-out algorithm. An example frame is shown in Figure 4.4 with landmark points labelled on the face: eight points on each eyebrow, 12 points on each eye, 2 points per nostril, 19 points around the chin and up the edge of the head to eye-level, 28 points on the outer lip contour, and 20 on the inner lip contour. After tracking, inner and outer lip contour points are the only landmarks to be retained for the AAM feature extraction process. PCA is performed on the shape and appearance features and 85% of variation is retained in both modes (described in more detail in Section 2.2.1). Velocity (Δ) and acceleration ($\Delta\Delta$) features are also concatenated before a dimension-wise z -score normalisation is performed (as described in Section 2.2.2). Finally, words are randomised between repetitions to remove any session bias.

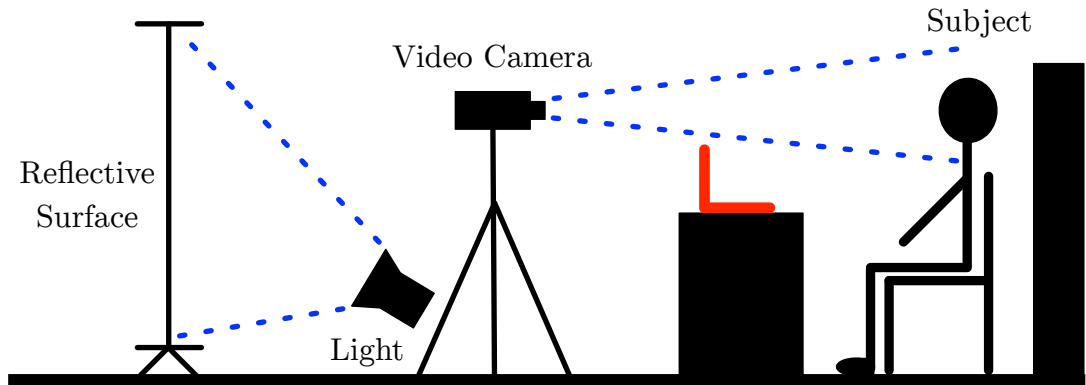


Figure 4.3: An illustration describing the setup used for recording the single-speaker isolated word dataset. The controlled environment was achieved with the use of a spot light against a reflective surface. The subject was sat upright in a chair with the video camera focussed into the whole face and turned into portrait mode.







Repetition	Session	No. of Words	No. of Frames	No. of Hand-Labelled Frames	Example Frame
1	1	211	70,024	20	
2	2	205	72,648	16	
3	2	210	72,471	18	
4	3	209	73,609	15	
5	3	211	73,669	15	
6	3	210	73,788	15	

Table 4.1: A description of the specially-recorded audio-visual isolated word dataset. Six repetitions were recorded from one female speaker reciting 211 carefully-selected words. Recordings spanned over three separate sessions, lasting between 20 to 30 minutes each. Each video was down-sampled from 1920 x 1080 pixels to 640 x 360 pixels to improve tracking and feature extraction computational efficiency.

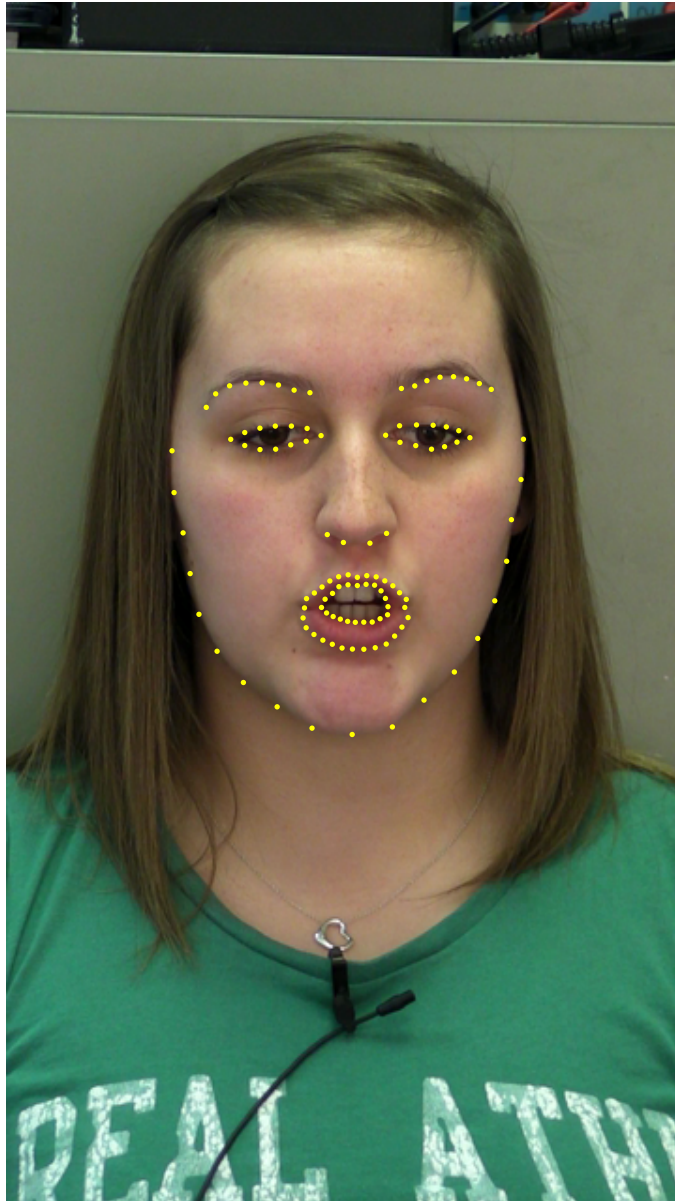


Figure 4.4: An example frame from the isolated-word dataset consisting 6 repetitions of 211 words by a female speaker. Landmarks are hand-labelled on 20 to 30 images on the face to aid tracking. Points on other parts of the face other than the lips are discarded for feature extraction.

4.3 LILiR-200 - A Small Continuous Speech Dataset

The Language Independent Lip Reading (LILiR) project is a collaboration between the Speech and Language Processing Laboratory at the University of East Anglia with the Centre for Vision, Speech and Signal Processing Laboratory at the University of Surrey. When the project began, very few audio-visual speech corpora were publicly available. Therefore, the first task on the LILiR project was to construct a large, multi-speaker, audio-visual dataset.

Data were collected in a specialised recording environment in controlled lighting conditions. The original dataset consists of 20 speakers each reciting 200 sentences from the Resource Management Corpus [Price et al., 1988]. Each speaker records their contribution in a single sitting to avoid any subtle differences in illumination. The video was recorded using a tri-chip Thomson Viper FilmStream high-definition camera at a sampling frequency of 25 frames per second and recorded with the subjects in full-frontal pose. To provide a smoother frame transition and a frame rate that was comparable with the audio, the original video is up-sampled to 100 frames per second. All speakers were instructed to remain as still as possible and to avoid the occlusion of any facial area.

Speaker variability in lip-reading is an area that has been clearly identified in previous work [Cox et al., 2008; Lan et al., 2010]. However, as this work focuses on developing an improved technique for lip-reading, for present purposes, we use the data of a single (male) speaker here. An example of a single frame from the chosen speaker’s session is shown in Figure 4.5. Here, the lips are tracked using the inverse compositional project-out algorithm with additional points added to the eyebrows and eyes to ensure the tracker remains in the correct position. These points are then removed before the feature extraction process to leave 16 points on the inner lip contour and 22 points on the outer lip contour.



Figure 4.5: An example frame from the LiLR-200 dataset consisting of one speaker reciting 200 sentences from the Resource Management corpus. Landmarks are hand-labelled on a set of training frames before tracking is performed to obtain landmarks for a video sequence. Points that are only required on the face for tracking purposes are removed before feature extraction to leave landmarks on the inner and outer lip contours.

4.4 RM-3000 - A Large Continuous Speech Dataset

It was found in Chapter 5 that the isolated word task was too easy for development of our confusion modelling techniques, and a new dataset was required that used continuous speech. The already-available LILiR dataset was ideal for preliminary experiments. However, although this dataset benefits from being a realistic task with a rich set of speakers, the amount of data for an individual speaker is 200 sentences, which was found to be inadequate to train the confusion matrix. To avoid speaker variability, work continued on the speaker-dependent case to refine the confusion modelling techniques. The RM-3000 dataset was recorded for the work described in Chapter 6 and Chapter 7.

The data consists of 3000 sentences spoken by a native English male speaker. The sentences were randomly chosen from the 8000 sentences in the Resource Management (RM) Corpus [Price et al., 1988]. The choice of corpus was motivated by the popularity of the RM corpus in previous audio and lip-reading tasks. Furthermore, recent work in automated lip-reading that has been conducted by our research group has used the RM corpus for their experiments [Lan et al., 2010], so it will be possible to make comparisons to current state-of-the-art lip-reading systems. The RM database has a vocabulary of almost 1000 words — an appropriate size for the current state-of-the-art in visual speech recognition. Phoneme transcriptions were derived from the first (the most common) pronunciation found in the BEEP Dictionary [Cambridge-University, 2012]. Statistics from the captured data are shown in Table 4.2.

Sentences were recorded in 19 sessions spanning a three day period. Videos were captured using a Sanyo Xacti high-definition camera using progressive scan at 59.94 frames per second. High quality audio was captured simultaneously using a clip-microphone attached to the subject’s shirt collar and recording at a bit depth of 16-bit and a sampling rate of 48 kHz on a mono channel. The videos were recorded with the camera in portrait mode with the subject in full-frontal pose and down-sampled from 1920 pixels x 1080 pixels to 360 pixels x 640 pixels (one third of the

Statistic	
Total number of sentences	3,000
Total number of unique phonemes	45
Total Number of phoneme tokens	105,561
Total Number of unique words	979
Total number of word tokens	26,114
Average number of phonemes per sentence	35.19
Average number of words per sentence	8.70
Average number of phonemes per word	4.04

Table 4.2: Statistics from the captured data taken from the selected subset of the Resource Management (RM) corpus.

original size). The recording apparatus was set up in the room using an identical configuration to that used in the isolated word dataset capture (illustrated in Figure 4.3).

After recording, the videos were post-processed to extract image frames and a small set of between 20 to 30 images were hand-labelled from each session to define the landmarks. To improve the accuracy of the tracking process, as with previous data capture, additional landmarks were labelled including the eyes, eye-brows, nose, and jaw-line up to the forehead. Using the hand-labelled images and an AAM, the landmarks are tracked using the inverse compositional project-out algorithm (described in Section 2.2.1). Only inner (12 points) and outer (16 points) lip contour points were retained from the complete set of 93 landmarks for the feature extraction process. For the entire corpus, a manual segmentation was performed to obtain the start and end frame marker for each sentence. Combining the tracked image sequences and the sentence frame boundaries, landmark points were defined for each frame before AAM parameters are computed, retaining 95% of variation in the shape mode and 90% variation in the appearance mode.

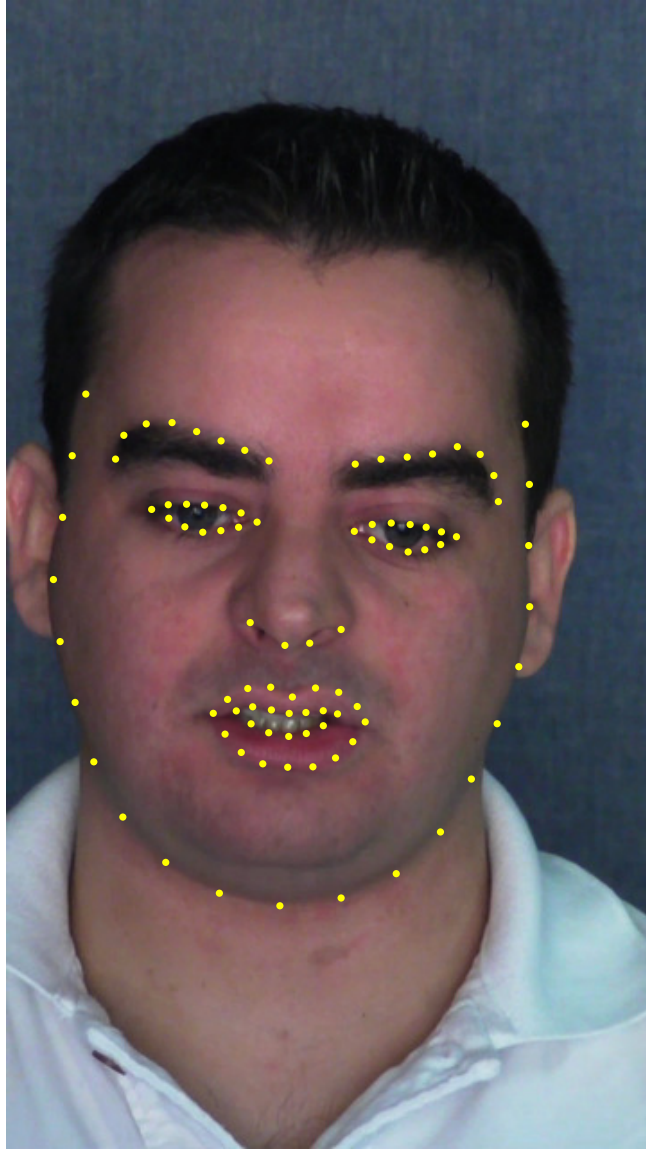


Figure 4.6: An example frame from a recording of the 3000-sentence RM database. A total of 93 points are labelled on the face to improve accuracy of the tracking procedure. After tracking, only inner (12 points) and outer (16) points are retained for feature extraction.

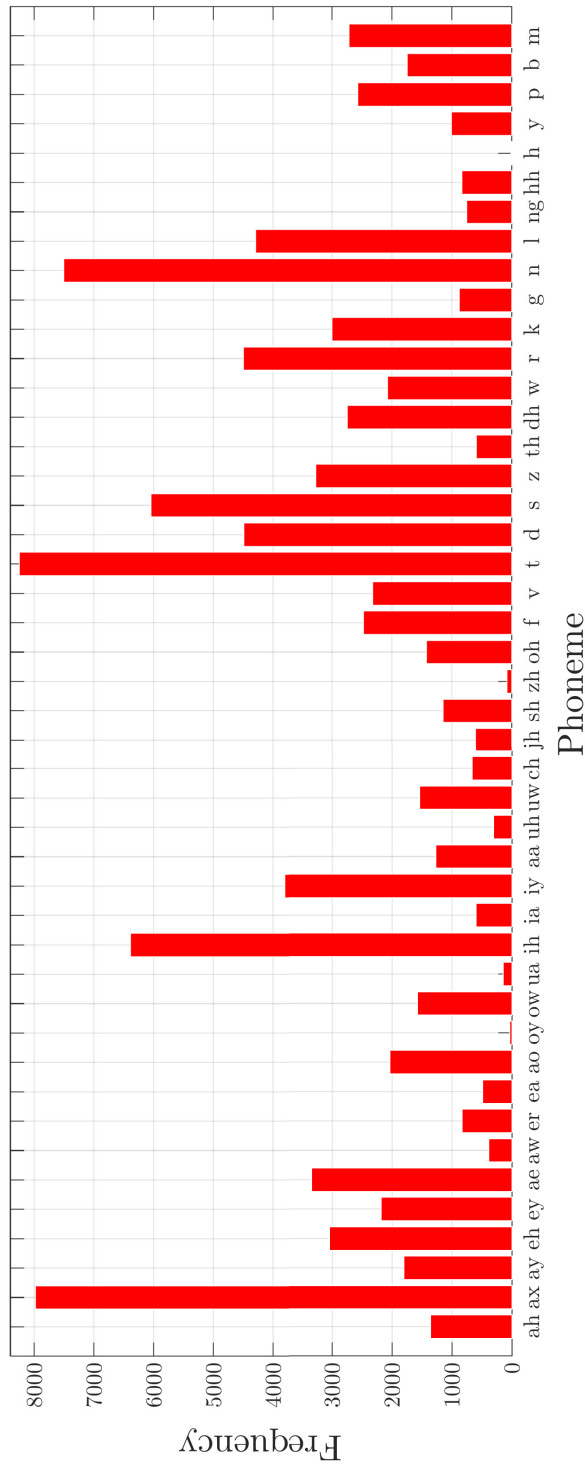


Figure 4.7: Frequency counts of each phoneme observed in the RM-3000 audio-visual speech dataset. Consonants $/t/$ and $/n/$ and the $/ax/$ vowel are used significantly more than the rest of the phonemes due to their popularity amongst the smaller, more frequently occurring words.

4.5 Summary

This chapter presents the three audio-visual datasets which have been used throughout this thesis. For initial experiments, we avoided the effects caused by continuous speech and maintained a focus on modelling the patterns of confusions by recording an isolated word dataset (ISO-211) consisting of approximately 1250 words from a single speaker. This gives us a controlled environment in which to perform an initial investigation but also comes with its limitations.

The progression of our work leads to the continuous speech task where we use two datasets. The first is LILiR-200 — a subset of an existing dataset which has been used in previous lip-reading work. At the start of Chapter 6, we evaluate the recognition accuracy for the LILiR-200 dataset using the standard approach and find that word accuracy is severely affected. Finally, we record a much larger dataset, named RM-3000, which contains 3000 sentences and uses a medium-sized vocabulary of approximately 1000 words.

Throughout this thesis, we maintain the use of single-speaker datasets to avoid the variation between the lip movements of different speakers. All of the subjects that were recorded in these datasets were native English speakers and videos were recorded in controlled lighting conditions in all datasets.

Chapter 5

Confusion Modelling for Isolated Words

5.1 Motivation and Aims

Chapter 3 presented a review of the current state-of-the-art automated lip-reading systems, with recent studies having shown that automated lip-reading performs significantly worse than audio speech recognition [Cox, 2008; Lan et al., 2010]. These poor results are most likely due to the lack of speech information available in a visual signal. For example, the position of some articulators cannot be seen, so there is no way to tell, for example, whether a sound is voiced or unvoiced. In addition, the purpose of speech is to produce distinctive sounds to convey a message, and the particular mouth-shapes used to produce these sounds are (usually) of no concern to the speaker: it is quite possible to produce a perfectly intelligible audio signal from mouth-shapes that are not distinct, something that is verified by human lip-readers who report that some people are much more “readable” than others. Furthermore, mouth shapes are severely affected by co-articulation [Jackson, 1988]. These limitations lead human lip-readers to make heavy use of pragmatics and contextual information to understand what is being spoken [Hansen and Coleman, 2005].

The original motivation for this work (detailed in Section 1.1) comes from previous work in confusion modelling for dysarthric speech recognition (see Section 3.5). Dysarthric speakers have poor control over their articulators which leads to the substitution, insertion, and deletion of sounds in the speech signal. The visual speech signal has an interesting relationship with the audio speech signal from a dysarthric speaker, with both paradigms producing a limited phonetic repertoire. We exploit patterns of confusions between the ground-truth and recognised sequences to find the most likely decoded sequence.

This chapter describes the first set of experiments in confusion modelling with the aim of improving the recognition accuracy of automated lip-reading. We start by using a simple, isolated word dataset which has been specially recorded for the purpose of this work (described in Section 4.2). The isolated word dataset has a vocabulary of only 211 words where each utterance consists of just one of these words. The simplicity of the task ensures that the work can concentrate on modelling the observable lip confusion patterns without additional variability (e.g. speaker-variability or word boundaries in a sentence) being introduced. The baseline system is known as the “standard approach” (described in Section 5.2) whilst our new confusion modelling technique is known as the “proposed approach”. A comparison of these two approaches is illustrated in Figure 5.1. In the proposed system, a correction module is inserted at the end of the recognition pipeline to correct the noisy transcriptions produced by the standard approach. With a model that is able to correct phoneme strings based on confusion likelihoods, a cascade of weighted finite-state transducers (as discussed in Section 3.6) can be used to find the best translation of the input phoneme string to a word string.

This chapter is structured as follows: Section 5.2 describes the standard approach to lip-reading and the application to the isolated word dataset, presenting baseline recognition results using a network of HMMs. Following this, a WFST composition cascade is introduced with an additional confusion model that is aimed at improving recognition accuracy.

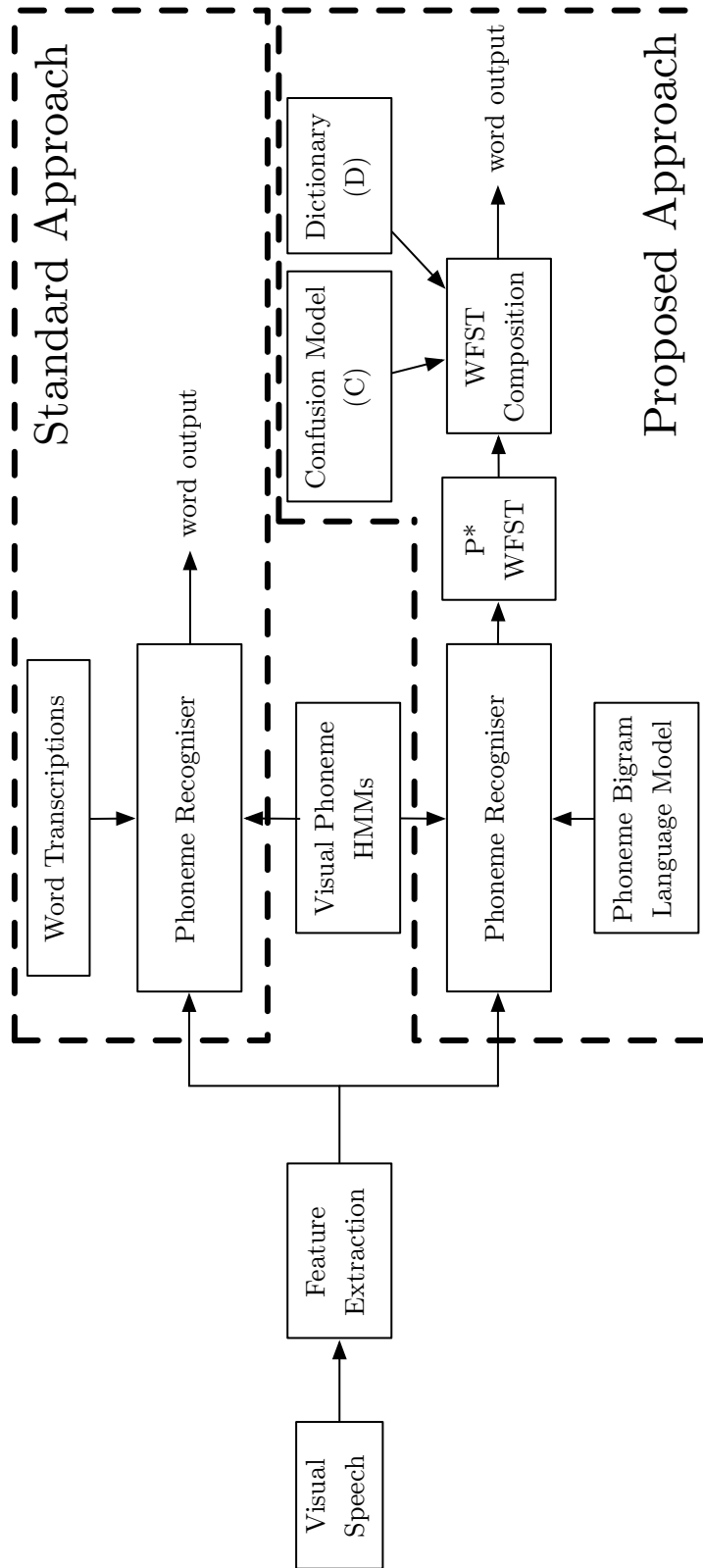


Figure 5.1: An illustration of the components in the two approaches to automated lip-reading: the standard approach — decoding using a network of trained HMMs as used in previous work, and the proposed approach — an almost identical recogniser system with an additional confusion module to correct the noisy transcriptions produced by the standard approach recogniser.

Although the results from preliminary experiments show some promise, we address the issues with using confusions derived purely from a symbolic alignment and propose a new classification method using timing information of the aligned phonemes. A new algorithm is proposed in Section 5.3.2.1 to identify confusions produced by a symbolic alignment as ‘*genuine*’ or ‘*spurious*’ by enforcing constraints on allowed confusions. Section 5.4 describes two additional techniques to improve the performance of the confusion modelling system further: adaptive confusion training (Section 5.4.1), and bigram confusion modelling (Section 5.4.2). Finally, this chapter is concluded in Section 5.5, where limitations are discussed and motivations for the next chapter are presented.

5.2 The Standard Approach

A baseline accuracy is first set using the standard approach. We build a state-of-the-art automated lip-reading system following a similar approach to that presented in previous literature [Lan et al., 2010; Newman et al., 2010]. In these works, a collection of viseme HMMs are built, each representing a (supposedly) unique gesture in the visual domain of speech. The phoneme-to-viseme groupings typically collapse the 45-phoneme symbol space into approximately 14 symbols (depending which phoneme-to-viseme mapping is deployed). A typical example of a phoneme-to-viseme mapping is combining the bilabial phonemes $/p/$, $/b/$, and $/m/$ into a single visemic class. However, we are attempting to model the confusion patterns between phonemes to ‘correct’ noisy phoneme output strings. To group the confusable phonemes together from the outset would defeat the purpose of the work. Therefore, we retain phoneme models of visual speech. The issue of viseme/phoneme modelling is discussed at length in Chapter 6.

The standard approach to automated lip-reading uses a network of mono-viseme HMMs (described in Section 2.4) to capture the variation in the time-series visual features (computed on a frame-by-frame basis). As with a typical state-of-the-art

audio ASR systems, an HMM is built for each of the 44 phonemes in the English language with an additional phoneme to model silence (for the isolated word task, silences will only appear at the start and end of each word). These *monophone* HMMs have a left-to-right topology, conforming to the progressive nature of speech production and are initialised using the flat start procedure and ten iterations of embedded training with the Baum-Welch algorithm using HTK.

The 1256 words are split evenly into six *folds*. To avoid any bias towards a particular session, words are randomised between folds such that no word appears in the same fold more than once. The six folds are then split into two sets for standard automated lip-reading: a training set consisting of five folds, and a testing set consisting of the remaining fold. Cross-fold validation is performed to provide a fair test with each fold being used for testing. Results over cross-fold tests can be computed as the average accuracy over the six recognition tests.

All conceivable left-to-right HMM topologies were explored during these tests, with the number of states and number of mixture components per model individually ranging from 1 to 15. Figure 5.2 presents the results from performing phoneme-level recognition using phoneme HMMs with a phoneme bigram language model. Over all topologies, the HMM with five emitting states and eleven mixture components per state provided the best parameters to maximise the word recognition accuracy at 60%. The results presented in Figure 5.3 provide word-level recognition results using the standard approach. In this case, the word-level accuracy peaks at 59% — an accuracy that is very close to the phoneme accuracy. When recognising whole words using phoneme-level HMMs, the HTK toolkit [Young, 2001] builds a word recognition network containing sub-word (phoneme) HMMs. Along with a language model, the decoding process chooses the most likely word from the vocabulary (in this case, only 211 possible words). However, when performing phoneme-level recognition, each word consists of six to eight phonemes on average, with each phoneme having a maximum of 44 possible symbols (the remaining *sil* model is reserved only for termination purposes). An insertion penalty is also optimised to reduce or increase

the number of decoded phonemes.

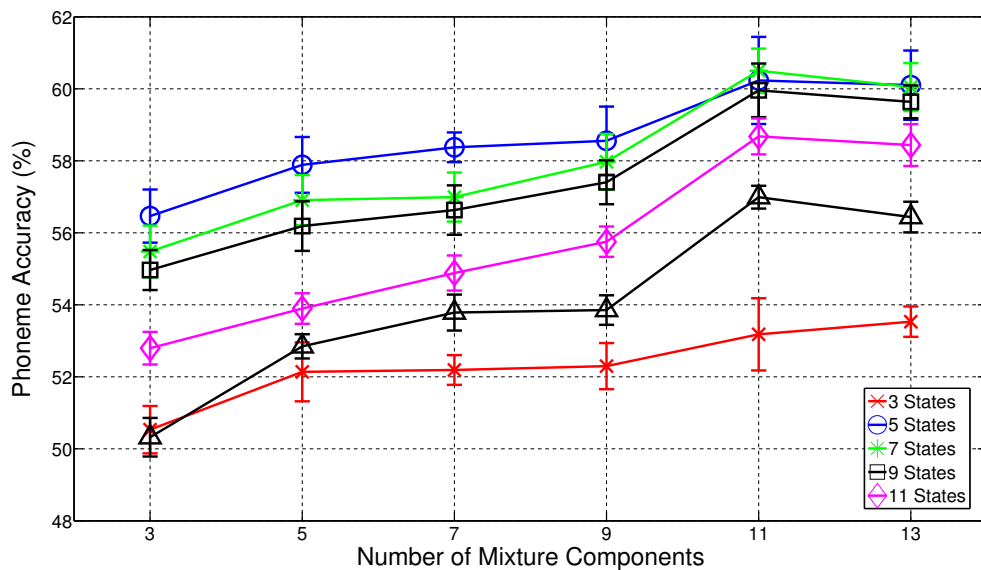


Figure 5.2: Phoneme recognition results on the isolated word dataset using a network of phoneme HMMs - a technique considered to be state-of-the-art in automated lip-reading. A phoneme bigram language model is used to improve recognition accuracy.

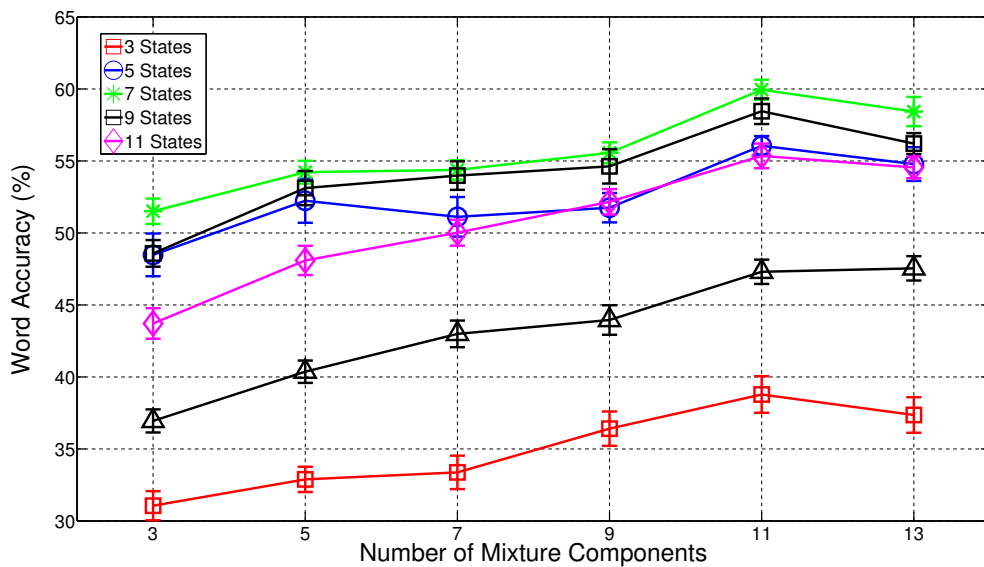


Figure 5.3: Word recognition results on the isolated word dataset using the standard approach to automated lip-reading (HMMs). This result provides the optimal baseline recognition accuracy for the isolated word task.

5.3 The Proposed Approach

The lip-reading accuracy reported in Section 5.2 (60%) is significantly inferior to that of a state-of-the-art audio ASR system. The proposed technique builds an additional confusion model to correct noisy phoneme strings produced by the standard ASR recogniser. Dysarthric audio ASR uses a WFST cascade structure and here, we adopt a similar approach to [Morales, 2009] by adding a confusion transducer to make corrections to the noisy transcriptions based upon visually-confused lip movements of sounds. The confusion cascade used for this work consists of three WFSTs composed as shown in Equation 5.1 where: P^* models the noisy decoded phoneme sequence, C models the confusion patterns (observed in the form of substitutions, insertions, or deletions), and D provides a mapping for phoneme sequences to be translated into words. For sentence-level experiments, an additional language model transducer is composed after the final stage to restrict sequences of words using an n -gram model. However, for the application to the isolated word task, this WFST is not required. Thus, the output from the recogniser is given as the best path (π) through the network:

$$\pi\left(P^* \circ C \circ D\right), \quad (5.1)$$

where π denotes an operation to find the shortest path through the composed network of WFSTs.

In the standard approach, the data were split into two sets: a training set consisting of five folds, and a testing set consisting of one fold. However, in the proposed approach, the additional confusion model also needs to be trained using a further held-out segment of the data. Therefore, for the experiments conducted on the proposed approach, the dataset is divided into three segments: training — four folds used to train the HMMs in the same procedure as the standard approach, testing — one fold used to train the confusion model, and the final validation fold to test the performance of the new confusion model.

5.3.1 Use of the top decoding only

The first WFST to be constructed in the composition cascade is P^* , a weighted transducer that represents the phoneme string to be corrected (i.e. the noisy phoneme sequence produced by the standard decoding approach). Phoneme recognition (using the standard approach) is performed independently on the testing set and the validation set over all folds using a phoneme bigram language model. This produces a set of decoded phoneme strings. To ensure maximum accuracy is achieved, the HMM parameters are chosen based on the most accurate phoneme decoding from the results presented in Figure 5.2. These decoded strings are then used to build a set of P^* WFSTs.

As the P^* WFST is the first transducer in the WFST cascade, there is no requirement for symbol translation. For this reason, P^* WFSTs are constructed such that each input symbol is always transduced to the same output symbol (equivalent to a finite-state automaton). A P^* WFST for a 1-best (i.e. most-likely) decoded sequence is modelled as a sequential path from a start state to an end state. Each arc represents a symbol decoded at a specific time. Section 3.6 introduces the use of the composition cascade in audio ASR, providing an example of a P^* WFST for a small example sentence (see Figure 3.4). In this chapter, the P^* WFST represents the decoded phonetic sequence of a single, isolated word (including the ‘*sil*’ termination symbols). The example illustrated in Figure 5.4 represents the noisy decoded phoneme sequence that has been produced by the standard phoneme HMM recogniser. The weights associated with each path are the negative log likelihood of observing the specific phoneme at a specified time. These are computed by the HMM network using a combination of acoustic and language model likelihoods following Equation 2.15.

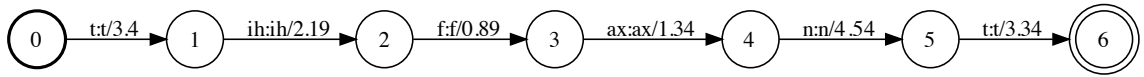


Figure 5.4: A standard ASR system has decoded a phoneme sequence that has been used to construct a sequential P^* WFST. The arc weights are calculated using the negative log likelihood for a specific symbol that has been produced by the standard recogniser.

5.3.2 The Confusion Model

A confusion matrix can be derived from the decoded sequences produced by the standard approach to lip-reading. Each decoded phoneme sequence is aligned to its corresponding ground-truth phoneme sequence using a dynamic programming algorithm. Figure 5.6 illustrates an example of an alignment between a ground-truth and a decoded sequence. A confusion matrix can be estimated from these alignments in the form of substitutions, insertions and deletions, where rows represent the ground-truth phoneme and columns represent the recognised/decoded phoneme. Figure 5.7 illustrates a confusion matrix that has been estimated from the phoneme alignment in Figure 5.6 with counts representing the frequency of the confusion between two phonemes.

There are two possible methods of inferring the entries in a phoneme confusion matrix. These are illustrated in Figure 5.5:

1. Decode the visual speech to be one of the 211 words in the vocabulary. Then, translate the decoded word into a phoneme string and align it to the phoneme transcription of the ground-truth word.
2. Decode the visual speech using a phoneme recogniser as a sequence of phonemes and align this sequence to the phoneme transcription of the ground-truth word. A phoneme bigram language model is used in the decoding process to guide the decoding towards more likely bigrams. Unlike method 1, the decoded output

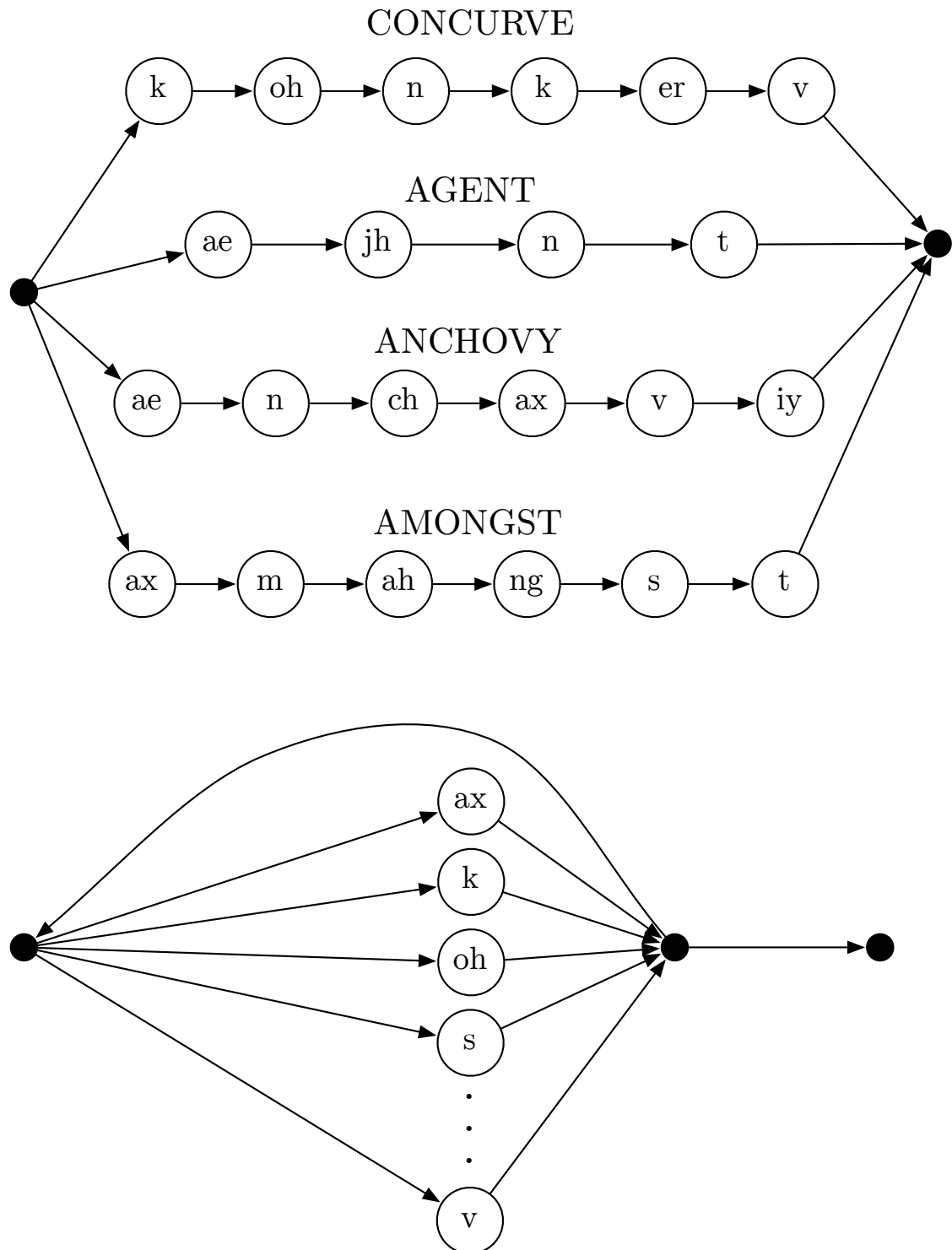


Figure 5.5: An illustration of the two approaches to populating a phoneme confusion matrix. Top shows the method of decoding one out of the 211 words from the vocabulary. The decoder is forced to take one of the valid 211 paths through the network of phoneme HMMs and produces the one with the highest likelihood. Bottom illustrates an alternative approach, using a phoneme bigram language model to guide the decoder to bigrams with a larger probability.

is not restricted to a specific vocabulary and it is often made up of a sequence that is not a word in the vocabulary.

	Sub.			Ins.		Del.		
Ground-Truth:	d	ih	f		r	ax	n	t
Recognised:	t	ih	f	v	r		n	t

Figure 5.6: An alignment between the ground-truth and recognised sequences using dynamic programming for the phonetic transcription of the word *different*. Counts in a confusion matrix can be populated using the substitutions (Sub.), insertions (Ins.), and deletions (Del.) observed from the alignment.

		Response								
Input		d	f	n	t	r	v	ax	ih	DEL
	d	0	0	0	1	0	0	0	0	0
	f	0	1	0	0	0	0	0	0	0
	n	0	0	1	0	0	0	0	0	0
	t	0	0	0	1	0	0	0	0	0
	r	0	0	0	0	1	0	0	0	0
	v	0	0	0	0	0	0	0	0	0
	ax	0	0	0	0	0	0	0	0	1
	ih	0	0	0	0	0	0	0	1	0
	INS	0	0	0	0	0	1	0	0	

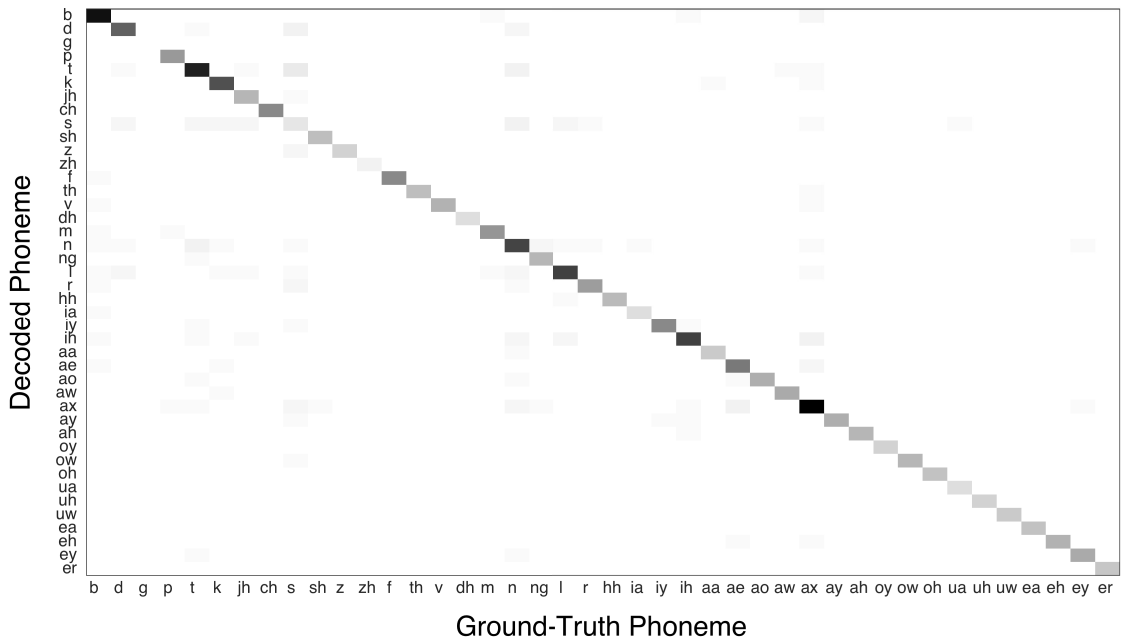
Figure 5.7: A confusion matrix that has been estimated from the phoneme alignment in Figure 5.6. The input rows represent the ground-truth phonemes and the response columns represent the decoded phonemes. The counts in the matrix represent the frequency of each confusion between a ground-truth and decoded phoneme with an additional row for insertions and column for deletions.

Figure 5.8 illustrates the phoneme confusion patterns that are produced by these approaches: using a strict network enabling paths through legal sequences of phonemes to build words, and using a phoneme bigram language model to enable non-legal sequences of phonemes to be decoded. Strong visual confusion patterns can easily be identified in Figure 5.8(b) using the phoneme bigram language model.

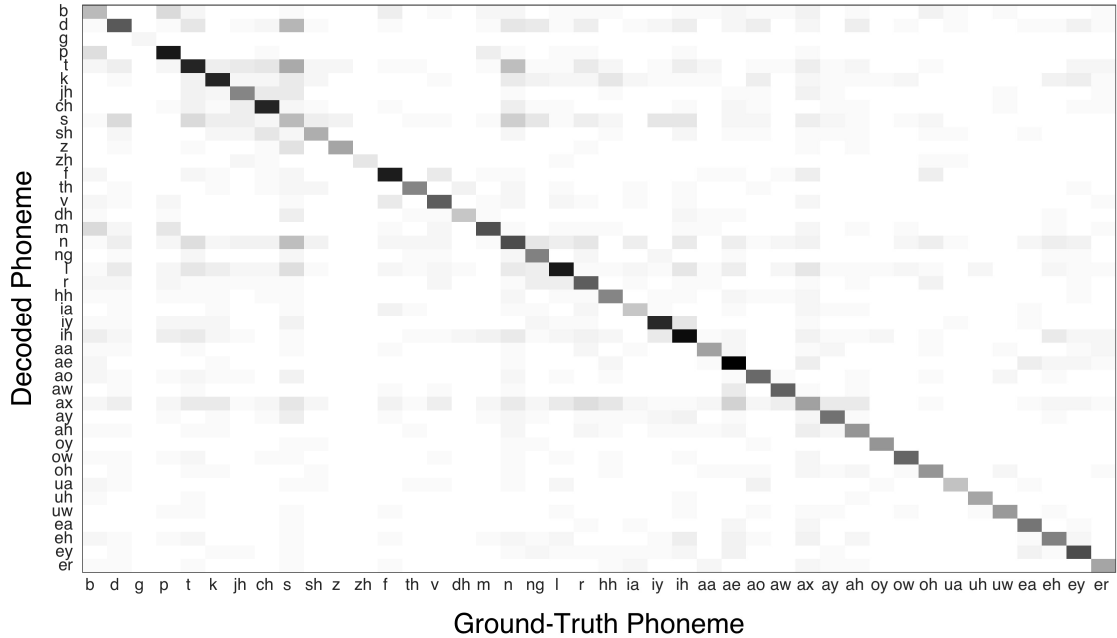
Strong visually-confused sounds are substituted using this technique such as /d/ being mis-recognised as /s/ and the phoneme /s/ being confused with /t/, both entirely plausible confusions in visual speech. However, the confusions in Figure 5.8(a) have little structure due to the influence of the strong language model, forcing the decoded string to be derived from one of the 211 possible word paths.

The method of correction using these confusion patterns can be considered as being a translation between two symbols with an associated probability. To correct isolated phonemes in a given string, the translation can be modelled to replace the hypothesised (incorrect) symbol with the ground-truth (correct) symbol. For example, the first confusion observed in the alignment illustrated in Figure 5.6 can be considered as a substitution, mapping the hypothesised symbol /t/ to the ground-truth symbol /d/. Similarly, the *deletion* of the phoneme /ax/ can be modelled as a translation from the hypothesised phoneme, the empty string, /-/ to the ground-truth phoneme /ax/. This translation flips the original insertions to become deletions and the deletions to become insertions. Figure 5.9 illustrates the use of these confusions to correct the hypothesised phoneme sequence produced by the recogniser. The confusion WFST adopts a cyclic structure so that it can be applied in the composition cascade with a P^* WFST of any size as input. By passing this incorrect sequence through this cyclic confusion transducer, the incorrect phonemes are corrected whilst phonemes that are already considered to be correct are unchanged after the transduction.

The output of the network that is produced by the composition of P^* and C is a rich set of hypotheses. To ensure that the confusion model can perform plausible translations that will improve the noisy phoneme sequence using the large set of hypotheses available, a probabilistic framework needs to be used. Rows from the count confusion matrix (as shown in Figure 5.8(b)) can be normalised using Equation 5.2 to produce a set of confusion probabilities where $c(i, j)$ is the number of times that phoneme p_i was recognised as phoneme p_j , and $prob(i, j)$ is the new entry in the probability confusion for the corresponding confusion:



(a) A phoneme confusion matrix produced by performing word-level decoding using the standard approach with a phoneme-to-word dictionary. Decoded words are decomposed into phonemes after recognition and aligned to the ground-truth to produce a set of substitutions, insertions, and deletions.



(b) A phoneme confusion matrix derived from recognition using a phoneme bigram language model. In contrast to using a word lattice, the bigram language model only favours phoneme pairs that frequently occur at training but does not restrict the output to sequences of phonemes that make words.

Figure 5.8: A comparison of the confusion patterns that can be observed when performing recognition using the standard approach to automated lip-reading. The two methods both require phoneme HMMs to be trained in the normal procedure. However, when performing recognition, words can be decoded using the HMM likelihoods and a phoneme-to-word dictionary or a simple phoneme bigram language model can be used to output a phoneme string.

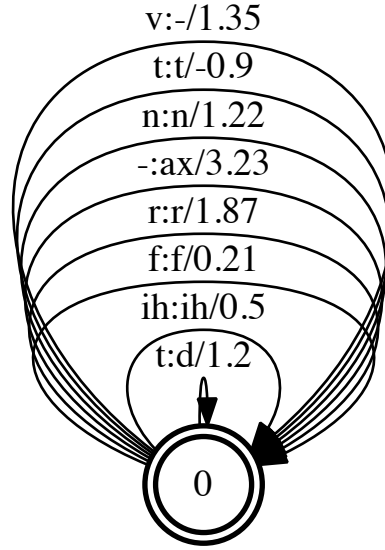


Figure 5.9: A cyclic confusion weighted finite-state transducer to correct the hypothesised sequence produced by the recogniser in Figure 5.6. The deletion of the phoneme *ax* in the hypothesised sequence is modelled in the confusion transducer using the epsilon symbol (-) to reverse the error and insert a phoneme into the hypothesised sequence. This epsilon symbol is reserved to allow ‘free’ transitions between states and is used to model both insertions and deletions.

$$prob(i, j) = \frac{c(i, j)}{\sum_{m=1}^n c(i, m)}. \quad (5.2)$$

Once an initial probabilistic model of confusions has been estimated, it is important to smooth it to re-distribute some of the diagonal probability mass to other elements on the row. This ensures that the diagonal (‘null substitution’) path is not always taken. We can do this using confusion matrix smoothing techniques (as described in Section 2.5.1). In this work, three smoothing methods are used:

1. **Base smoothing** — re-assigning a fixed percentage of the diagonal probability to all other classes on the same row. More detail on base smoothing can be found in Section 2.5.1.1.
2. **Exponential smoothing** — giving more control of the probability mass that is re-distributed using a smoothing parameter α in Equation 2.18 to compute the new probability.

3. **Base smoothing using visemic classes** — using the same theory as base smoothing (described in Section 2.5.1.1) but re-distributing probability mass based upon prior knowledge of expected confusions in visual speech. Firstly, we re-distribute an amount from the diagonal count across the row to all phonemes *except* those in the same visemic class. For phonemes in the same visemic class, the remaining diagonal count is split evenly between these classes. Smoothing the counts for a given phoneme in this class — a specified percentage of the diagonal count is re-distributed across the row (as with the standard base smoothing procedure) before distributing the residual diagonal count evenly between these three classes, making them equally likely to be confused. For this experiment, we use the Fisher phoneme-to-viseme mapping [Fisher, 1968].

Finally, once the smoothing technique has been applied, the new probabilities can be translated into negative log likelihoods using Equation 5.3. This conversion allows the WFSTs to adopt the tropical semiring (described in Section 2.6) framework using the following weight computation:

$$weight(p_1, p_2) = -\log(Pr(p_2|p_1)), \quad (5.3)$$

where $Pr(p_1, p_2)$ is the probability of observing p_2 (the decoded phoneme) when it should have been p_1 (the ground-truth phoneme) after smoothing. After composing the P^* WFST with the confusion WFST (C), a rich set of hypotheses is produced. For example, the composition of the P^* WFST shown in Figure 5.4 and the confusion WFST shown in Figure 5.9, the first symbol could be translated from $/t/$ to either $/t/$ or $/d/$. The next WFST in the cascade further narrows the set of possible paths by only allowing the input to accept valid phoneme strings that produce words (the dictionary transducer (D)).

Initial work was carried out using the confusion information obtained using a standard recogniser. The `HVite` tool (which is part of HTK) produces a phoneme-level confusion matrix based on an alignment between the recognised output and

the ground-truth transcription. This is used to build a simple one-state transducer modelling the unigram phoneme confusions (as illustrated in Figure 5.9). Because we have a limited dataset consisting of 211 isolated words per fold, we produce the top 100-best phoneme hypotheses from the standard approach recogniser to train the confusion model, where the word accuracies are shown in Figure 5.10 using the confusion matrix produced from the output of the standard recogniser. These results are produced by using the three-level WFST composition with two confusion matrix smoothing methods; base smoothing and base smoothing with visemic classes (both described in previously in this section). The preliminary results are clearly inferior to the results achieved by the standard approach with our system achieving a best word accuracy of 21.42% using base smoothing with 5% distribution.

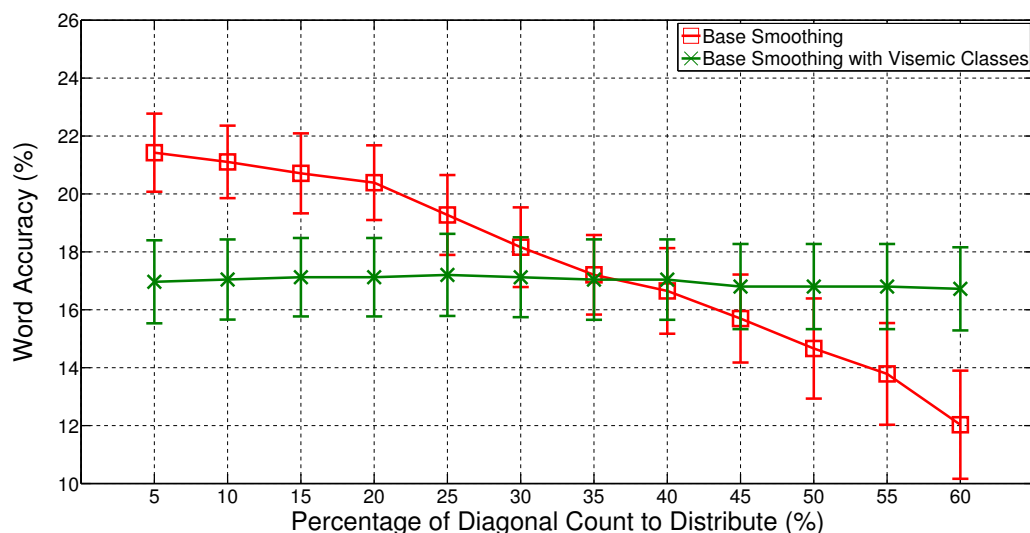


Figure 5.10: Word-level recognition results using weighted finite-state transducers to correct noisy output transcriptions. The confusion model is trained using the output confusion matrix from the HMM recognition.

When forming the confusion matrix, because of the purely symbolic nature of the DP alignment when forming the confusion matrix, the timing information of aligned phonemes is ignored. It is considered that this could introduce noise into the confusion model. To overcome this, an alternative method has been used to populate the confusion matrix, which filters confusions based on the timing of decoded phonemes.

5.3.2.1 Timing Offset Classification

System	Word Accuracy (std. deviation)
Standard Approach HMM System	60.0% (4.2)
WFST Confusion System with confusion from a symbolic alignment	21.4% (3.3)

Table 5.1: Word recognition accuracies for the two approaches used in this work so far. The ‘Standard Approach HMM System’ uses the standard approach to lip-reading (using a network of HMMs), and ‘WFST Confusion System with confusion from a symbolic alignment’ uses a WFST cascade with a confusion model that has been trained using the confusion patterns that have been produced from the standard approach phoneme recogniser.

The results shown in Table 5.1 compare the baseline system (i.e. the standard HMM approach) with our confusion modelling approach using the confusion matrix produced by the standard phoneme recogniser. The alignment procedure used in the confusion modelling approach uses dynamic programming (an illustration of this alignment is shown in Figure 5.6). This provides a purely symbolic alignment of the two sequences. Although this is a reasonable method to produce recognition statistics (such as decoding accuracy), if the alignment is used to produce confusion patterns, some may be spurious. Figure 5.11 illustrates an example alignment between a ground-truth and recognised phoneme sequence for a single word. The DP algorithm finds the alignment with the shortest cost and therefore gives precedence to aligning correctly recognised phonemes in the correct position. Despite these considerations, the DP algorithm is based purely on the alignment of symbols in a sequence. Figure 5.12 illustrates the discrepancies in the time registration of each recognised phoneme. Here, the audio ground-truth timing information is produced from a forced-alignment procedure using the MFCC features — a segmentation that corresponds to the patterns that can be seen in the audio spectrogram. However, the video ground-truth alignment (also estimated by performing a forced-alignment but using the AAM features) is different from the audio alignment, a result that we would expect as the time registration of events on the visible articulators might

come before or after the events that are produced by the unseen articulators.

Ground-Truth: **sil** **b** **aa** **th** **ch** **ea** **sil**
 Recognised: **sil** **ih** **ch** **ea** **sil**

Figure 5.11: An example alignment between the ground-truth and recognised phoneme sequences for the word “*bath-chair*”.

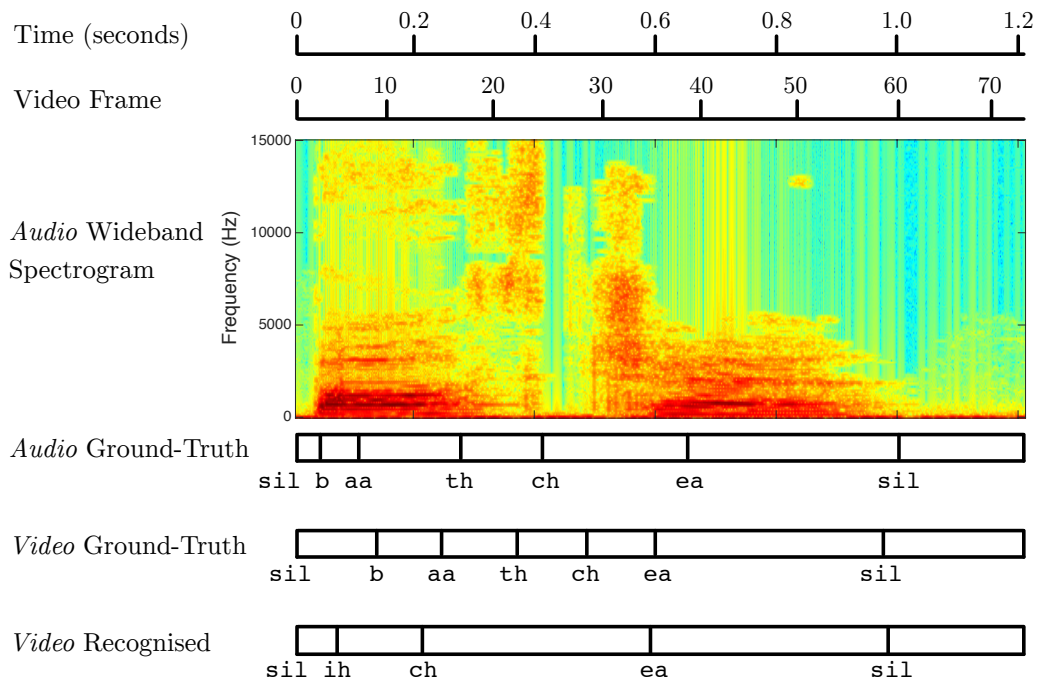


Figure 5.12: A time alignment example for the word “*bath-chair*” taken from the isolated word dataset. The ground-truth alignments are captured using a forced alignment procedure using the audio and video features individually, and the recognised alignment is the output sequence that has been produced by the standard HMM approach. In this example, although the symbolic alignment would align *ih* with *aa* and *ch* with *ch*, their respective starting points are poorly synchronised and could be considered as spurious confusions.

Figure 5.12 also shows the time registration of the recognised phoneme sequence produced by the standard HMM approach. The time registration of the recognised phoneme /*ea*/ is perfectly aligned with the video ground-truth alignment. However, the timing offset (defined as the absolute difference between the time registration of the ground-truth phoneme and the time registration of the recognised phoneme) is large for the recognised phonemes /*ih*/, and /*ch*/.

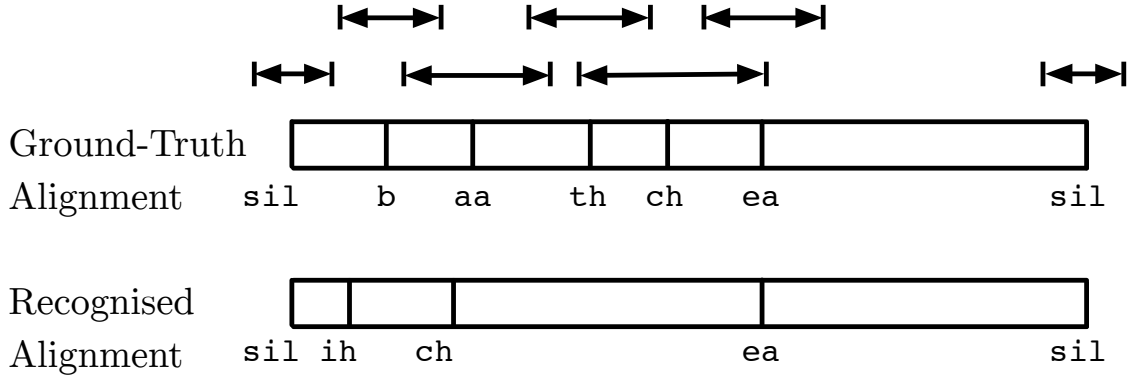


Figure 5.13: An illustration of how the timing offset window is enforced for each ground-truth phoneme against its aligned decoded phoneme. The windows (shown as double-headed lines above the transcripts) are centered from the ground-truth phoneme and defined as $\pm x$ standard deviations away from the mean offset for the ground-truth phoneme.

phonemes have been symbolically aligned, the difference in the time registration of these phonemes (i.e. the timing offset) is very large. This poor alignment may lead to spurious confusion patterns introducing noise into the confusion model. We introduce a new *timing offset classification algorithm*, providing a system to filter nonsensical confusions.

Firstly, we align recognised and ground-truth sequences, but for each alignment, we also note the time registration of the decoded and ground-truth symbol. The absolute difference between these two points (i.e. the timing offset) is recorded along with the identity of the ground-truth phoneme. All aligned phonemes are required to be in the same articulatory class (vowel or consonant) before their timing offset is considered. When this process has been completed for each training utterance, the timing offsets for a particular phoneme are summarised as a mean and a standard deviation. The algorithm used to train the offset windows for all phonemes is described in Algorithm 1.

After the training phase, the testing process populates the confusion model using the trained timing offsets. Each recognised phoneme sequence is passed through the

alignment algorithm again. However, this time, a check is performed to ensure that the timing offset is within a boundary (defined as $\pm x$ standard deviations away from the mean timing offset for the ground-truth phoneme). Given the timing offset for a given alignment, a decision is made to classify the confusion as genuine if it fits within the window (i.e. within $(\pm x)\sigma_p$ from the mean offset (μ_p)), or as a spurious confusion (i.e. outside of the window defined as $\pm x$ standard deviations from the mean). All timing considerations are made with respect to the ground-truth phoneme in the confusion (p). This algorithm ensures that implausible confusions with respect to their timing, are ignored whereas genuine confusions (defined using the offset boundary) are used to build a cleaner confusion model. Figure 5.13 illustrates how the phoneme timing windows relate to an example alignment and the algorithm used to populate confusion matrix in this way is shown in Algorithm 2.

Figure 5.14 illustrates the effect that the timing offset classification has on the number of observed confusion patterns that are used to construct the phoneme confusion matrix. The timing offset window is defined as $\pm x$ standard deviations from the mean timing offset for the reference (ground-truth) phoneme. At first, the number of ‘allowed’ confusions linearly increases as a function of the timing window (for the timing windows 0.5, 1, and 1.5). After this, the number of confusions are stable up to 3 standard deviations, meaning that most ‘spurious’ confusions are expected to be outside of the window defined by three standard deviations. Most importantly, the enforcement of a timing window (with any value between 0.5 and 3 standard deviations) reduces the number of confusions dramatically from the number of counts in the original confusion matrix (i.e. with no timing offset window and using every observed confusion pattern in the training data). One would expect that the number of confusions when using an offset window of ± 3 standard deviations would capture over 99% of the confusions assuming that the timing offsets are normally distributed. However, we found that, although most offset windows for the consonant sounds were normally distributed, there were many confusion counts for vowels that did not conform to a parametric distribution in the same way. This is expected due to the inherent difficulties with observing articulatory cues for vowel

sounds in visual speech. The contribution of these confusions can be measured in Figure 5.14 from ± 3 standard deviations to ‘No Window’ — approximately 11% of the total number of confusions.

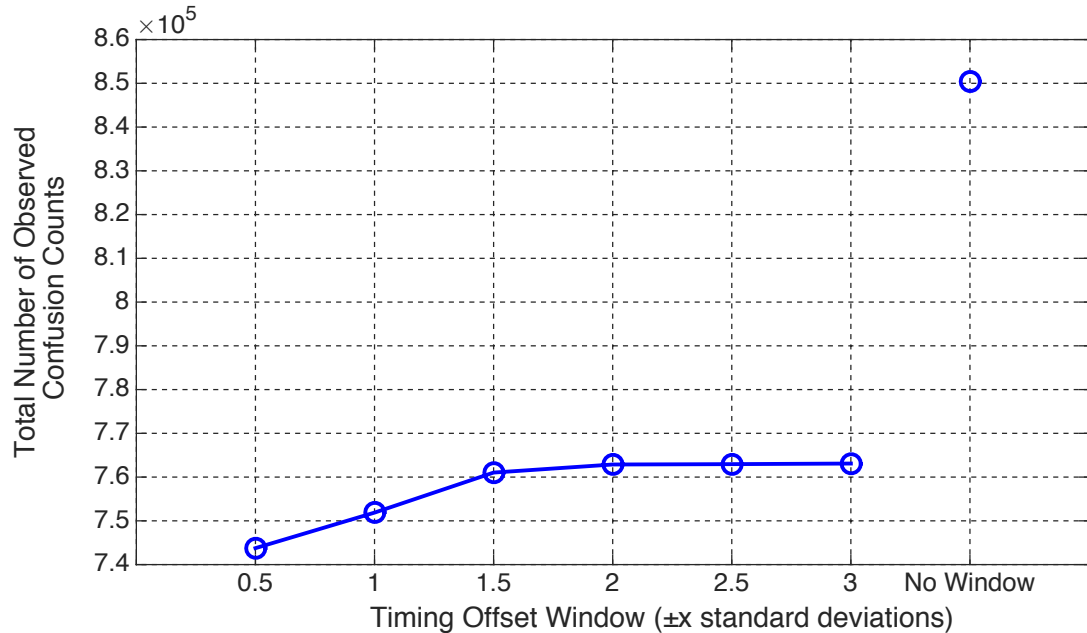


Figure 5.14: Analysis of the number of confusion patterns that are accepted as a function of the timing window. Error bars are not shown here because they are too small.

The word recognition results presented in Figure 5.15 are produced by our WFST confusion modelling system using the timing offset classification algorithm. Here, we compare the three confusion matrix smoothing methods described in Section 5.3.2 and report word accuracy using different timing offset windows ranging from 0.5 standard deviations to 3 standard deviations. The results for all smoothing methods show no significant gain in word accuracy can be achieved for any specific timing window. However, the two base smoothing methods (base smoothing and base smoothing with visemic classes) provide a significant improvement over the previous system using no timing information giving a word accuracy of 46.1%, an increase of approximately 24%.

Data: Testing set for fold x

Result: Statistics to characterise the timing offset window for each distinct ground-truth phoneme

```

phonemeList  $\leftarrow$  {p,b,m...iy}; /* containing all distinct phonemes */
offsets  $\leftarrow$   $\emptyset$ ;
/* populate with an empty array for each phoneme */
for  $i \leftarrow \text{length}(\text{phonemeList})$  do
    | offsets[i]  $\leftarrow$   $\emptyset$ ;
end
for rec  $\leftarrow$  each transcription in the testing set for fold  $x$  do
    | gt  $\leftarrow$  ground-truth transcription for the test utterance;
    | {alignedGT,alignedRec}  $\leftarrow$  align(gt,rec);
    | for  $p \leftarrow$  all phonemes in this alignment do
    |     | /* Check if the aligned phonemes are in the same phoneme
    |     |   class (i.e. consonant, vowel, or silence) */
    |     | if class(alignedRec[p]) == class(alignedGT[p]) then
    |     |     | recTime  $\leftarrow$  retrieve time stamp for phoneme alignedRec[p];
    |     |     | gtTime  $\leftarrow$  retrieve time stamp for phoneme alignedGT[p];
    |     |     | timingOffset  $\leftarrow$  abs(recTime - gtTime);
    |     |     | idx  $\leftarrow$  find index of phoneme alignedGT[p] in phonemeList;
    |     |     | offsets[idx]  $\leftarrow$  {offsets[idx] timingOffset};
    |     | end
    | end
end
/* Finally, characterise the timing offset of each phoneme
alignment by computing the mean and standard deviation. */
meanOffsets  $\leftarrow$   $\emptyset$ ;
stdDevOffsets  $\leftarrow$   $\emptyset$ ;
for  $i \leftarrow$  each element in offsets do
    | meanOffsets  $\leftarrow$  mean(offsets[i]);
    | stdDevOffsets  $\leftarrow$  standardDeviation(offsets[i]);
end

```

Algorithm 1: Algorithm to train the timing offset window for all unique phonemes. After alignment and classification, the timing offset is characterised by the mean and standard deviation over the set of offset measurements for the respective ground-truth phoneme.

```

Data: Testing set for fold  $x$ 
Result: Populate the confusion matrix by accepting or rejecting counts
           based on the timing offset window

/* Variables 'phonemeList', 'meanOffsets' and 'stdDevOffsets'
   have been pre-computed using Algorithm 1 */
/* 'win' is a variable defining the timing window to use (can be
   0.5, 1, 1.5, 2, 2.5, 3) */
/* Populate a 2D confusion matrix */
confMat  $\leftarrow \emptyset$ ;
for  $i \leftarrow 1 \dots \text{length}(\text{phonemeList})$  do
    confMat[i] =  $\emptyset$ ;
    for  $j \leftarrow 1 \dots \text{length}(\text{phonemeList})$  do
        confMat[i][j] = 0;
    end
end
for  $\text{rec} \leftarrow \text{each transcription in the testing set for fold } x$  do
    gt  $\leftarrow$  ground-truth transcription for the test utterance;
    {alignedGT, alignedRec}  $\leftarrow$  align(gt, rec);
    for  $p \leftarrow \text{all phonemes in this alignment}$  do
        recTime  $\leftarrow$  retrieve time stamp for phoneme alignedRec[p];
        gtTime  $\leftarrow$  retrieve time stamp for phoneme alignedGT[p];
        timingOffset  $\leftarrow$  abs(recTime - gtTime);
        idx  $\leftarrow$  find index of phoneme alignedGT[p] in phonemeList;
        if ((timingOffset > (meanOffsets[idx] - (stdDevOffsets[idx] * win))) &&
            timingOffset < (meanOffsets[idx] + (stdDevOffsets[idx] * win))) &&
            (class(alignedRec[p]) == class(alignedGT[p])) then
            /* The timing window is within the parameters and in
               the same class so this is classified as a genuine
               confusion */
            gtIndex  $\leftarrow$  find index of alignedGT[p] in phonemeList;
            recIndex  $\leftarrow$  find index of alignedRec[p] in phonemeList;
            confMat[gtIndex][recIndex]  $\leftarrow$  confMat[gtIndex][recIndex] + 1;
        end
    end
end
save(confMat); /* Save the confusion matrix for this window */

```

Algorithm 2: Algorithm to populate the phoneme count confusion matrix. Due to the possibility of poor alignment introducing noise to the confusion model, the timing offset between the forced-aligned ground-truth and the decoded phoneme sequences are used to classify confusions as genuine or spurious. The length of the offset window can be altered by adjusting the number of accepted standard deviations away from the mean offset.

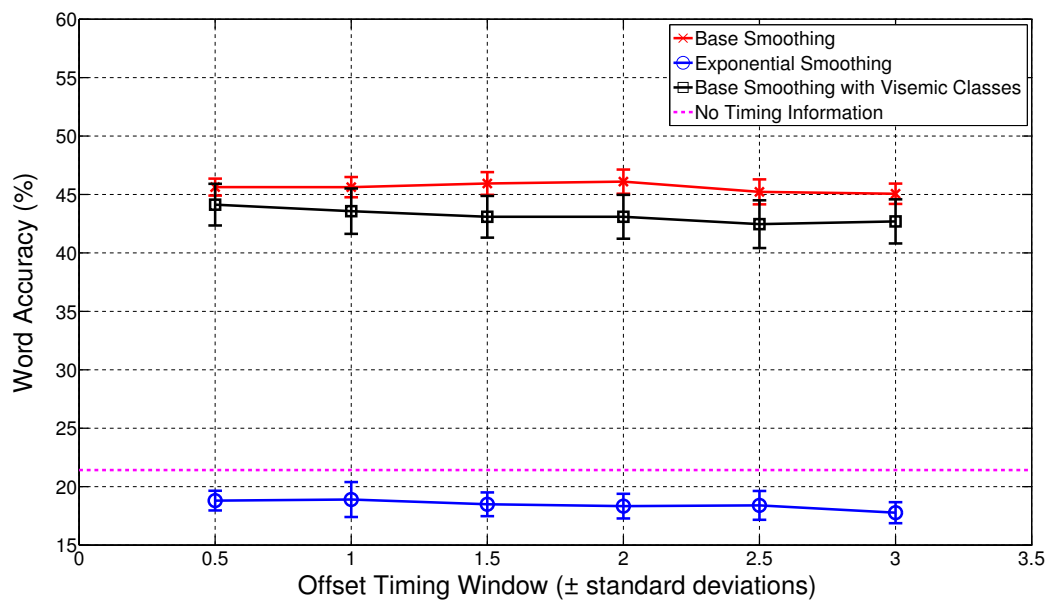


Figure 5.15: Word recognition results using different timing offset windows to restrict the confusion patterns that are used to populate the confusion matrix. The best accuracy that was achieved without the timing window (shown in the results from Figure 5.10) is shown as a dotted line. The input P^* WFST is modelled using the top decoding that is produced by the standard approach phoneme recogniser (described in Section 5.3.1)

System	Word Accuracy (std. deviation)
Standard Approach HMM System	60.0% (4.2)
WFST Confusion System with confusion from a symbolic alignment	21.4% (3.3)
WFST Confusion System with timing confusion matrix and top decoding only and base smoothing	46.1% (1.0)

Table 5.2: The word recognition performance of three techniques used so far in this work. The ‘Standard Approach HMM System’ uses a network of phoneme-level HMMs formed to recognise one word from the 211-word vocabulary, the ‘WFST Confusion System with confusion from a symbolic alignment’ system uses the confusion matrix produced by the standard HMM phoneme recogniser in the WFST confusion cascade, and ‘WFST Confusion System with timing confusion matrix and top decoding only and base smoothing’ presents the best result using the timing offset classification algorithm and base smoothing.

5.3.3 Use of top n decodings

In a typical ASR system, the best decoded sequence is considered to be the path through the network of HMMs with the largest likelihood. However, it is not necessarily the most accurate. An alternative approach is known as *n-best decoding*. Here, n decoded sequences are produced, ranked in order of decreasing likelihood. In a similar approach to building WFSTs from the single most likely decoding (described in Section 5.3.1), the set of decodings can also be modelled as a P^* transducer which accepts the n decoded sequences. In this work, we explore two approaches to modelling the n -best decodings.

The first method builds isolated P^* WFSTs for each of the n decoded sequences (adopting the approach described in Section 5.3.1). The set of n WFSTs are then combined using the WFST union operation (see Section 2.6.1.2) to produce a single WFST that accepts all n decodings (an example of this is shown in Figure 5.16). The algorithm used to derive these WFSTs is described in Algorithm 3. To improve performance, determinization and minimization are applied to the resulting WFST to provide a pruned, more efficient, network.

We conduct a set of experiments using this technique to create the P^* WFSTs for n -best decodings and use the confusion model described earlier in this section in the WFST cascade. The results presented in Figure 5.17 are shown for the three confusion matrix smoothing methods adopted in this work and the different timing window sizes. Base smoothing performs significantly better than the other two smoothing methods but can only achieve 35.97% which provides a decline in word accuracy of about 23% from the standard approach.

As discussed previously, the most likely (1-best) transcription is not necessarily the most accurate. We construct a WFST using the union of single-path P^* WFSTs from the top n decodings produced by the standard HMM phoneme recogniser. However, it could also be considered that the most likely phoneme sequence is actually a *combination* of the top n transcriptions with all n transcriptions participating in a vote to determine which phoneme is decoded at each point in time. Although

the union of WFSTs does provide the ability to model the n -best transcriptions, it only aims to model the n paths from the WFSTs that have been combined. This produces a WFST which accepts strings from either of the original WFSTs. Using this technique, the P^* model does not accept combinations of symbols from multiple n transcriptions. To overcome this, an alternative approach is proposed using an n -dimensional alignment procedure.

```

Data: The  $n$ -best decoded phoneme strings produced by the standard
         approach ASR
Result: A weighted finite-state acceptor that accepts the  $n$ -best decoded
         phoneme strings.
/* The  $n$ -best decoded phoneme sequences are stored in a 2-D
   array named 'sentences'. The first dimension represents the
   sentence ID and second provides the indices into the  $n$ 
   transcription */
for  $i \leftarrow$  each sentence in the validation set for fold  $x$  do
    nBestSentences  $\leftarrow \emptyset$ ;
    for  $j \leftarrow$  each  $n$ -best decoded transcription for this sentence do
        startState  $\leftarrow 0$ ;
        endState  $\leftarrow 1$ ;
        for  $m \leftarrow$  number of phonemes in transcription  $j$  for sentence  $i$  do
            phoneme  $\leftarrow$  sentences[ $i$ ][ $j$ ][ $m$ ];
            weight  $\leftarrow$  weights[ $i$ ][ $j$ ][ $m$ ];
            arcs[ $m$ ] = BuildWFSTArc(startState,endState,phoneme,weight);
            startState++;
            endState++;
        end
        wfst = BuildWFST(arcs);
        nBestSentences[ $j$ ] = wfst;
        if  $j==1$  then
            | nBestWfst = wfst;
        else
            | nBestWfst = WFSTUnion(nBestWfst,wfst);
        end
    end
end
end

```

Algorithm 3: A simple algorithm to combine all n -best strings into a single transducer. The output WFST only accepts the n -best decoded strings produced by the standard approach ASR.

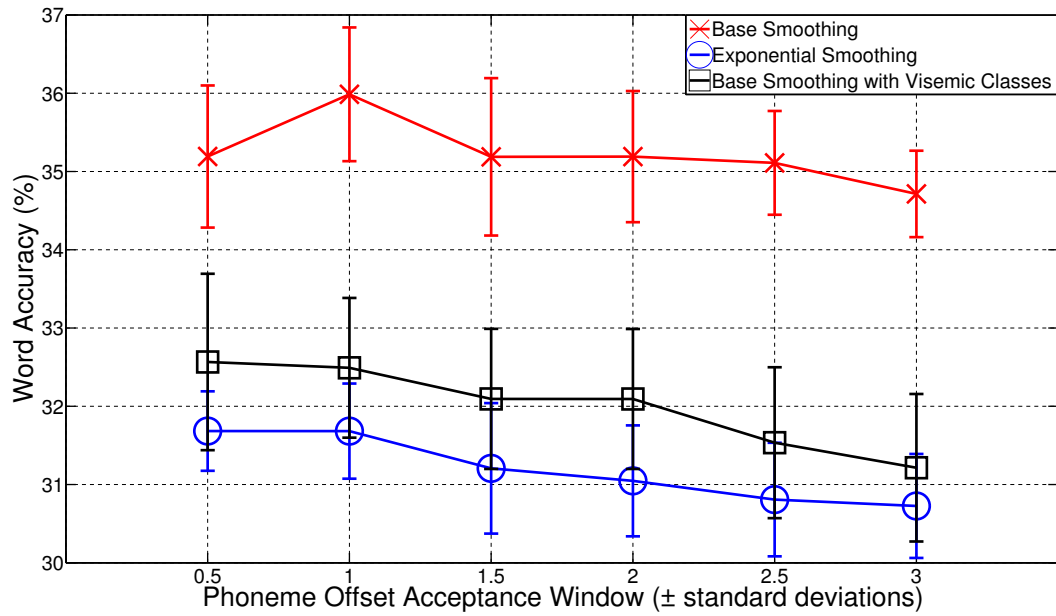


Figure 5.17: Recognition results when using the original union method for the input P^* transducer

System	Word Accuracy (std. deviation)
Standard Approach HMM System	60.0% (4.2)
WFST Confusion System with confusion from a symbolic alignment	21.4% (3.3)
WFST Confusion System with timing confusion matrix and top decoding only and base smoothing	46.1% (1.0)
WFST Confusion System with timing confusion matrix and n -best decodings combined into a WFST using Algorithm 3	36.0 (0.1)

Table 5.3: A comparison of the word recognition results achieved with the systems used so far.

```

Data: The  $n$ -best decoded phoneme strings produced by the standard
        approach ASR
Result: A finite-state acceptor that accepts all  $n$ -best strings and all
        combinations of phoneme strings from the top  $n$  phonemes
        decodings by using an  $n$ -dimensional alignment.

/* sentences is a 2-D array containing the  $n$ -best strings. The
   first dimension represents the sentence ID and second
   provides the indices into the  $n$  transcription. */
alignedNBestSentences  $\leftarrow \emptyset$ ;
for  $i \leftarrow$  each sentence in the validation set for fold  $x$  do
    nAlignedTrans  $\leftarrow \emptyset$ ;
    /*  $n$  is the number of decoded strings (i.e.  $n$ -best) */
    longestTranscIdx  $\leftarrow$  find index of longest transcription;
    longestTransc = sentences[ $i$ ][longestTranscIdx];
     $j \leftarrow 1$ ;
    while  $j \leq n$  do
        if  $j \neq$  longestTranscIdx then
            gt  $\leftarrow$  ground-truth transcription for the test utterance;
            {alignedGT, alignedRec}  $\leftarrow$  align(gt,transcriptions[ $j$ ]);
            if length(alignedRec) > length(longestTransc) then
                longestTranscIdx  $\leftarrow j$  /* Reset the longest trans */
                longestTransc = sentences[ $i$ ][longestTranscIdx];
                 $j \leftarrow 1$ ;
            else
                nAlignedTrans[ $j$ ] = sentences[ $i$ ][longestTranscIdx];
            end
        end
    end
    alignedNBestSentences[ $i$ ]  $\leftarrow$  nAlignedTrans ;
end
startState  $\leftarrow 0$ ; endState  $\leftarrow 1$ ;
for  $i \leftarrow$  length(alignedNBestSentences) do
    for  $t \leftarrow$  length(longestTransc) do
        {phns, liklihd}  $\leftarrow$  get all phonemes and log likelihoods at timeslot  $t$ ;
        {uniqPhns, uniqWgts}  $\leftarrow$  unique(phns, liklihd); /* Find all
            unique phonemes at a given timestamp in the warp. Also
            sum log likelihoods where the symbols are equal. */
        for  $m \leftarrow 1 \dots \text{length}(\text{uniqPhns})$  do
            arcs[ $m$ ] =
                BuildWFSTArc(startState,endState,uniqPhns[ $m$ ],liklihd[ $m$ ]);
        end
        startState++; endState++;
    end
end

```

Algorithm 4: An algorithm to align n -best decoded strings to build a WFST which can model combinations of phoneme sequences from the top n decodings. The output WFST accepts all n -best strings and all paths through all n -best strings

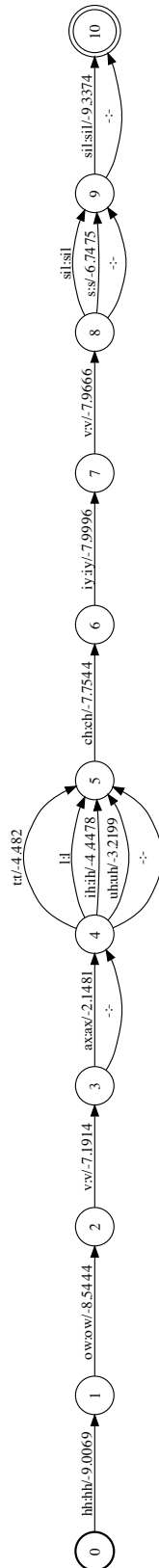


Figure 5.18: An alternative approach to building the P^* transducer is shown here, modelling the noisy transcription from the standard approach for the ground-truth word ‘Overachiever’. This method attempts to find the best alignment between the 9-best recognised transcriptions using dynamic programming alignment. Once the best alignment is found, the transducer is built as shown.

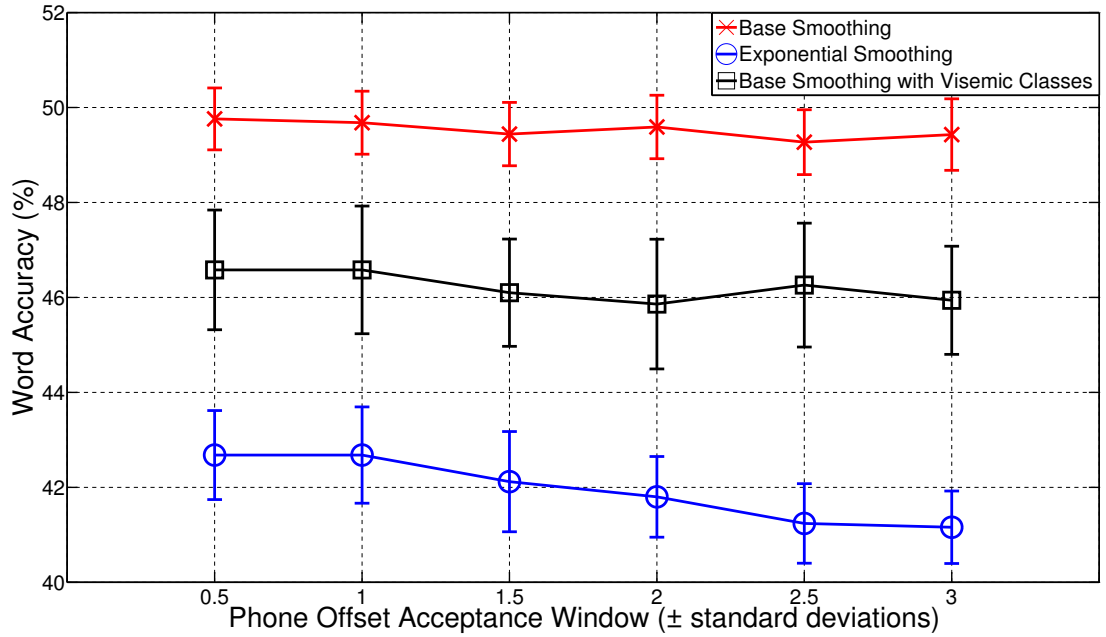


Figure 5.19: Recognition results when using the adapted version for the input P^* transducer. All n transcriptions are warped together to form the input transducer.

To align the top n sequences together, we use a reference sequence which is the longest decoded sequence from the n -best decodings. All other transcriptions are aligned to the reference sequence to produce a set of aligned n -best transcriptions that are of the same length. The n -best sequences can then be modelled using a WFST (an example is pictured in Figure 5.18) where multiple transitions are included where the top n decodings disagree with each other. This technique produces more compact WFSTs to allow faster computation in the cascade with a minimum number of arcs and states whilst also modelling all combinations of phoneme sequences from all n decodings. However, it also increases the out-degree, i.e. the number of arcs exiting from each state, which could hinder performance in bigger systems. Algorithm 4 describes the process of performing the n -dimensional alignment to build the P^* WFSTs.

The word-level recognition results produced using this new n -best P^* transducer are shown in Figure 5.19. This new technique provides an increase of approximately

System	Word Accuracy (std. deviation)
Standard Approach HMM System	60.0% (4.2)
WFST Confusion System with confusion from a symbolic alignment	21.4% (3.3)
WFST Confusion System with timing confusion matrix and top decoding only and base smoothing	46.1% (1.0)
WFST Confusion System with timing confusion matrix and n -best decodings combined into a WFST using Algorithm 3	36.0 (0.9)
WFST Confusion System with timing confusion matrix and n -best decodings combined into a WFST using Algorithm 4	49.7 (1.6)

Table 5.4: A comparison of the word recognition results achieved with the systems used so far.

14% in word accuracy over the previous technique. However, these results indicate that this method is inferior to using HMMs with the standard approach (60% word accuracy). Simple base smoothing (without visemic classes) provides the best results, whereas, exponential smoothing proves to be the most ineffective for this task.

5.3.4 Discussion

The work presented so far introduces a new confusion modelling technique for automated lip-reading. Many parameters have been adjusted during this series of experiments, with over 1500 experiments being performed over six folds. The full list of parameters that have been explored in this work are shown in Table 5.5. Because of the number of parameters and their possible values, the presented results are chosen as the parameter values that achieve the highest word accuracy.

Most interestingly, Table 5.6 shows that the proposed approach does correctly recognise some words that are recognised incorrectly by the standard approach.

Table 5.5: The list of parameters and the values tested in the set of confusion WFST experiments.

Parameter	Parameter Description	Range of Possible Values	Best Parameter Value
n	the number of top likelihood transcriptions to include in the P^* transducer	1, 3, 5, 7, 9, 13, 11, 15	9
<i>smoothMeth</i>	the type of smoothing applied to the confusion transducer counts	Base Smoothing, Base smoothing with visemic classes, Exponential Smoothing	Base Smoothing
<i>win</i>	the number of standard deviations from the mean phoneme offset acceptance window	0.5, 1, 1.5, 2, 2.5, 3	0.5
<i>percentDist</i>	the percentage of the diagonal confusion count to distribute to unseen events (only used for base smoothing and its derivatives)	5%, 10%, 15%, 20%, 25%, 30%, 35%, 40%, 45%, 50%, 55%, 60%	10%
α	the variable used in exponential smoothing (see Equation 2.18)	0.01 0.05 0.1 0.15 0.25 0.5 1	0.01
<i>training n</i>	the number of n-best transcriptions used to train the confusion model	100	100

However, the standard approach recognises a total of 173 words correctly where the proposed approach recognises incorrectly, meaning that the standard approach is still superior in the recognition task. The results in Table 5.6 show that there is clearly scope for fusing the outputs of the two algorithms to arrive at a more accurate decision, but this has not been pursued in this work.

Table 5.7 gives a comparison of the results of the experiments on isolated word recognition. The Baseline 1 result (row A) using the standard approach achieved 60%, which is quite good for a vocabulary of 211 words, although it should be noted that this is a speaker-dependent system. The Baseline 2 (row B) result shown in Table 5.7 (20.2%) is produced from a system which finds the ground-truth phoneme

sequence from all possible 211 words that has the lowest alignment cost (using the dynamic programming algorithm) to the recognised phoneme sequence. As described in Section 2.4.3, we use an alignment procedure where penalties are set for each alignment: 7 for an insertion or deletion, 10 for a substitution, and zero for a correct symbol alignment. Because of the noise in the decoded phoneme sequences, the baseline 2 result is significantly inferior to the baseline 1 result. Turning to the experiments on our proposed method, using an identity confusion-matrix (row C) with very small uniform probabilities on the off-diagonal elements to enable the cascade to explore substitutions, insertions and deletions (but with no prior knowledge of the confusion patterns), gives an accuracy of 35.4%. The small off-diagonal probability are required because the confusion matrix must have non-zero probabilities in the off-diagonal elements in order for any legal word to be decoded by the dictionary (D) transducer, which enables a rich set of candidate words. By contrast, the dynamic programming algorithm of Baseline 2 (row B) uses a cost function that finds only the closest match.

If the confusion matrix is estimated from the output of the baseline recogniser with no timing information being used, accuracy falls to 21.4% (row D). This result suggests that many of the alignments are not genuine confusions, but are in fact an artefact of the alignment process (discussed in Section 5.3.2.1). Using timing information improves the accuracy hugely (rows E, F, and G) with base smoothing giving a better result (49.7%) than exponential smoothing (42.7%). However, the best result from the WFSTs is still about 10% lower than the standard HMM approach.

		WFST	
		Correct	Incorrect
HMM	Correct	580	173
	Incorrect	55	448

Table 5.6: A contingency table to provide a comparison between the two approaches. ‘HMM’ denotes the standard approach using a network of phoneme-level HMMs, and ‘WFST’ represents our new approach using a confusion model in a WFST cascade with the decoded phoneme strings that are produced by a standard approach phoneme recogniser.

	System	Word Accuracy (std. deviation)
A	Standard Approach HMM System as shown in Figure 5.1 (Baseline 1)	60.0% (4.2)
B	Phone decoding followed by string-matching (Baseline 2)	20.2% (1.4)
C	WFSTs with identity confusion matrix (to avoid $-\infty$ log probabilities on off-diagonal elements, a small probability mass is added to every element)	35.4% (2.3)
D	WFST Confusion System with confusion from a symbolic alignment	21.4% (3.3)
E	WFST Confusion System with timing confusion matrix and top decoding only and base smoothing	46.1% (1.0)
F	WFST Confusion System with timing confusion matrix and n -best decodings combined into a WFST using Algorithm 3	36.0 (0.9)
G	WFST Confusion System with timing confusion matrix and n -best decodings combined into a WFST using Algorithm 4	49.70 (1.6)

Table 5.7: A summary of the word-level recognition results using different methods. Word accuracy and standard deviation are both reported.

5.4 Extending the Proposed Approach

The results presented so far (shown in Table 5.7) show that our new approach is still inferior to the standard HMM system. In this section, we explore two extensions to improve the recognition accuracy of the new system: *adaptive confusion training*, and *bigram confusion modelling*.

Firstly, in Section 5.4.1, we explore a new method of iteratively updating the confusion matrix based upon the evaluation of word accuracy from the WFST cascade — a technique that we term *adaptive confusion modelling*. Finally, we extend the confusion model to introduce phonemic context. This increases the complexity of the confusion model, adding confusions of phoneme pairs to the existing unigram phoneme confusion matrix.

5.4.1 Adaptive Confusion Model

The unigram confusion model presented in previous work has shown promise with a peak accuracy of 49.7%. However, this recognition accuracy is still inferior to that of the standard approach (60%). The results presented in Table 5.7 show that a gain of 28.3% in recognition accuracy can be achieved by discarding spurious confusions using the timing offset classifier described in Section 5.3.2.1. A possible limitation with this work could be that the confusion model is estimated offline only once at training time. Here, we explore a new technique for confusion modelling whereby the confusion probabilities are updated iteratively.

To improve the reliability of the confusion model, an adaptive training approach is presented. Unlike the previous offline-trained confusion model, the adaptive confusion framework iteratively updates the probability confusion matrix based upon decoding errors observed on a known set (in this case, the testing set). Figure 5.20 illustrates the components in our adaptive system. The approach is performed as follows:

1. A standard phoneme recogniser is used to produce a set of decoded strings using the *testing set*.
2. The confusion WFST is constructed using the alignment between the decoded strings produced by step 1 and the ground-truth strings (using the techniques presented in Section 5.3.2).

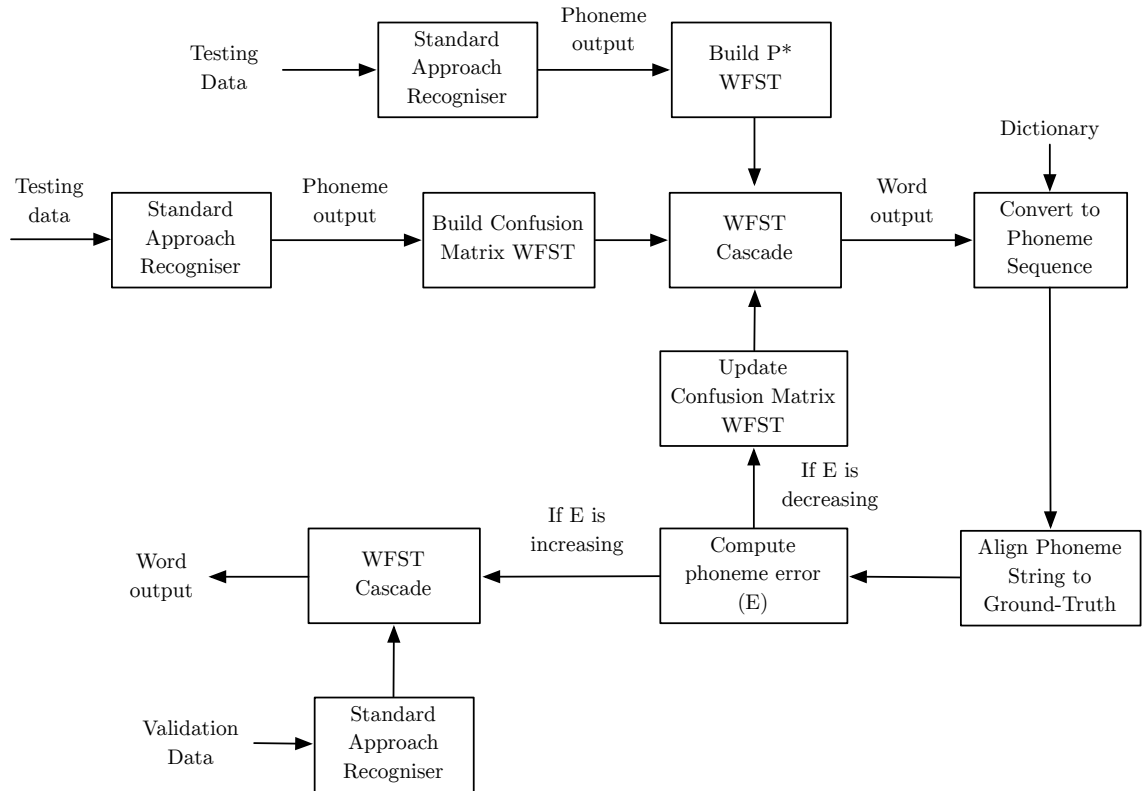


Figure 5.20: A description of the adaptive confusion model. Confusion probabilities are updated in an iterative process by running the testing set through the standard approach recogniser and the WFST cascade. Computed errors are then used to update the confusion probabilities before the process is repeated. The iterative process continues until the total error starts to increase — representing a point at where the recognition accuracy can not improve any more.

3. The phoneme strings produced by step 1 are modelled as WFSTs and passed through the WFST cascade using the confusion WFST that has been trained in step 2.
4. The *word* output from the WFST cascade is converted to a phoneme string using a word-to-phoneme dictionary and aligned to the ground-truth phoneme string.
5. An error confusion matrix is produced which documents the incorrectly decoded phonemes.
6. If the phoneme error is *decreasing*, the confusion matrix weights are updated using the new phoneme strings with Equation 5.4.

7. If the phoneme error is *increasing*, it is likely that the confusion matrix weights have been overtrained on the testing set, so we revert to the confusion weights that were used for the previous iteration.
8. A standard phoneme recogniser is used to produce a set of decoded strings using the *validation set*.
9. Run the WFST cascade with the phoneme strings produced by the *validation set* (in step 8) and the recent confusion matrix (from step 7) and report the final word accuracy.

Confusion weights are updated by adapting the original probability confusion matrix. At each iteration, updated probabilities (posteriors) are derived by using a combination of the current information and the new information (obtained from the error confusion matrix) as follows:

$$P'(i, j) = \frac{(1 - \alpha)P(i, j) + \alpha N(i, j)}{\sum_j (1 - \alpha)P(i, j) + \alpha N(i, j)}, \quad (5.4)$$

where P is the current probability confusion matrix before the update, P' is the updated probability confusion matrix, N is the error confusion matrix containing the new information observed on this iteration, (i, j) represents the entry in the confusion matrix for ground-truth phoneme i and recognised phoneme j , and α determines the *learning rate* (i.e. the degree of influence that the new information has on the updated probability) where $0 < \alpha < 1$. Lower learning rate values (α) force prior probabilities (P) to have more influence on the posterior probabilities (P') whereas higher learning rates will give more weighting to the newly observed information (N). Using the WFST tropical semiring (described in Section 2.6), all posterior confusion probabilities are used to compute the negative logarithm costs used in the WFST arc weights.

If the total error on a specified iteration is decreasing, the recognition performance is improving — therefore, the adaptive process should continue. However, as soon as the error increases (and recognition accuracy decreases), the adaptive process is

stopped and the confusion matrix used on the previous iteration is used in a WFST cascade with a special held-out validation set.

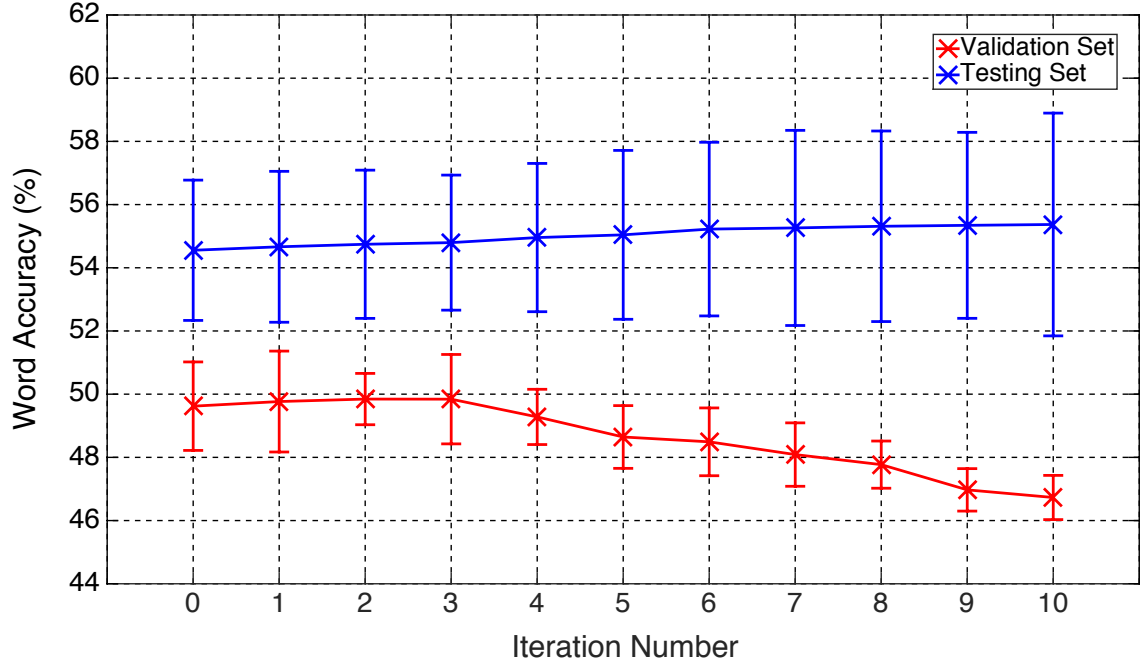


Figure 5.21: Word recognition results using adaptive confusion modelling. The configuration that achieved the best results in Section 5.3 was used in an iterative training process. These results were achieved using a learning rate (α) of 0.1 — putting a strong influence on the prior probabilities with the new information obtained by the error confusion matrix only contributing 10% of the overall weighting.

The set of adaptive experiments conducted here have used an identical configuration to the best word accuracy results presented in Section 5.3 with 9-best decoded phoneme strings being used as input to the WFST cascade, and base smoothing used to re-distribute 10% of the diagonal elements to off-diagonal elements. The initial confusion matrix also uses a timing offset window of ± 0.5 standard deviations. However, after the first iteration, a timing offset window is not defined as the WFST cascade output is incapable of including timing information. Cross-fold validation is performed as in previous experiments with the dataset divided into six repetitions. Different learning rates (α) have been tested to update the probabilities in Equation 5.4 where $\alpha = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The results presented in Figure 5.21 compare the word accuracy on each iteration for both the

testing set (used to compute the error and adapt the weights) and validation set (an unseen segment of the data). Iteration 0 represents the initial state of the confusion weights which are derived from the best WFST system shown in Table 5.7. The best results in Figure 5.21 are presented and use a learning rate (α) value of 0.1. This brings a 10% contribution from the new information (produced by the error confusion matrix on a given iteration) and a 90% contribution from the existing information (priors). Although the accuracy improves at a very slow rate for the testing set, the accuracy on the unseen validation set first increases by 0.08% to and then declines by 3.03% over 10 iterations.

One major advantage of the WFST composition cascade is its ability to translate illegal sequences of phonemes into legal words (with the use of the lexicon — mapping legal phoneme sequences to whole words). This feature, however, does also provide limitations for the adaptive system with a small vocabulary. To update the phoneme confusion probability matrix from the WFST output, a decoded isolated word is converted to a phoneme sequence before alignment to the ground-truth is performed. For the isolated word task, the WFST cascade will output a single word from 211-word vocabulary, based on the input phoneme string and the decoded confusion patterns. With such a limited vocabulary, it is possible that an incorrectly decoded word could be very different from the ground-truth word. When aligning these incorrect words to the ground-truth, spurious confusions could be introduced. The propagation of this error through each iteration of the adaptive system could have a detrimental affect on the update procedure and, hence, hinder the recognition accuracy of unseen data. The results in Figure 5.21 show that the initial confusion matrix produces the best word accuracy on the validation set. As the confusion matrix is updated in a slow process (using $\alpha = 0.1$), the word accuracy of the (unseen) validation set climbs for the first three iterations before slowly deteriorating. Meanwhile, the performance on the testing set improves very slightly (by nearly 1%). The decline in validation set performance is likely because the initial confusion matrix is becoming corrupted by spurious confusions that have been introduced by unrealistic word alignments.

5.4.2 Bigram Confusion Model

Contextual information is often used to improve the performance of state-of-the-art speech recognisers. This information can be provided in the form of an n -gram model (described in Section 2.3). The simplest form of contextual n -gram modelling is the *bigram language model*. Here, pairs of symbols are coupled together to form a *bigram sequence*. A simple counting procedure on these bigrams means that $Pr(p_2|p_1)$ (the probability of observing p_2 given that the preceding phoneme was p_1) can be estimated. The binding of two context-dependent phonemes forms an interesting framework for which to model confusions in lip-reading. The co-articulation effect has a larger impact on the visual modality than the audio. Therefore, the addition of contextual information to confusion patterns is a promising technique for modelling visual speech confusions.

We extend the work conducted so far to include *bigram confusion patterns*. In the unigram confusion model (shown in Figure 5.9), we use a single-state, cyclic WFST. In the bigram confusion model, this is extended to account for valid sets of two phonemes — i.e. with the transitions coming in and out of state 0 to states 1, 2, 3, and 4 in Figure 5.22. It is unlikely that we will observe all possible phoneme pairs (bigrams) in the corpus. Therefore, most conventional n -gram models employ a back-off procedure to revert to the unigram model when required (see Section 2.3). In this work, we maintain the unigram confusion matrix used in Section 5.3 as a back-off model. The unigram confusions are inserted in the same configuration as previous work, a cyclic structure entering and exiting from state 0 in Figure 5.22. The bigram confusion matrix is populated using the same alignment procedure as described in Section 5.3.2 but with a window covering two phonemes instead of one. However, owing to the sparsity of the populated bigram confusion matrix, only observed bigrams are used in the construction of the WFST confusion model. For unseen bigrams, the flexible structure of the confusion model allows for a back-off event to the unigram probability. A simple illustrated example of a bigram confusion model is shown in Figure 5.22. This model has been constructed using

a small vocabulary of only three symbols: a , b , and c . Arc weights are defined using the negative logarithm of the entries in the bigram and unigram probability matrices. Owing to the strong influence of the unigram confusion matrix, a back-off weight, β is applied to each unigram probability (where $0 < \beta < 1$). To counteract this weighting, the bigram probabilities are weighted by $(1 - \beta)$. A higher β value allows more unigram probability to filter through whereas a lower β value allows the bigram probabilities to influence the decoded path. Experiments were conducted using a β value with a 0.1 increment from 0.1 to 0.9 with the best accuracy being achieved when $\beta = 0.7$ (i.e. a 70% contribution from the unigram confusion patterns and a 30% contribution from the bigram confusion patterns).

The extension of the confusion model to allow for context does improve the ability to capture the contextual effects present in visual speech. However, it does come at a computational cost. In the unigram confusion model described in Section 5.3.2, each entry in the confusion matrix is included as an arc, allowing all possible substitutions, deletions, and insertions. Therefore, for a set of 44 phonemes, the confusion model consists of 44^2 substitution arcs, 44 deletion arcs, and 44 insertion arcs — a total of 2024 arcs. In the bigram confusion model, the 2024 unigram arcs are also included as back-off paths. However, the addition of bigram paths can significantly expand the model (if all bigram substitutions are observed at training time, this would total 3,751,968 arcs). The sparsity of the bigram confusion matrix enables the model to be simplified. The statistics shown in Table 5.8 demonstrate the degree of sparsity in the computed bigram confusion matrices. Although there are 3,751,968 possible entries in the bigram confusion matrix, there are only 6785 populated entries on average. If one assumes that all diagonal elements are populated, this leaves only 4849 non-zero entries. With a unigram back-off employed, this sparsity can be managed by only including non-zero bigram entries in the confusion WFST. Furthermore, by reducing the number of bigrams in the confusion model to only allow for non-zero entries (i.e. only observed sequences), the exponential growth in the number of arcs is avoided.

Experiments were conducted using cross-fold validation over the six randomised repetitions of each isolated word. To reduce the computation time, this work uses the best-performing smoothing method from the experiments conducted in Section 5.3, with simple base smoothing achieving the highest word accuracy by distributing 10% of the diagonal mass to unseen confusions. The tests were also conducted over

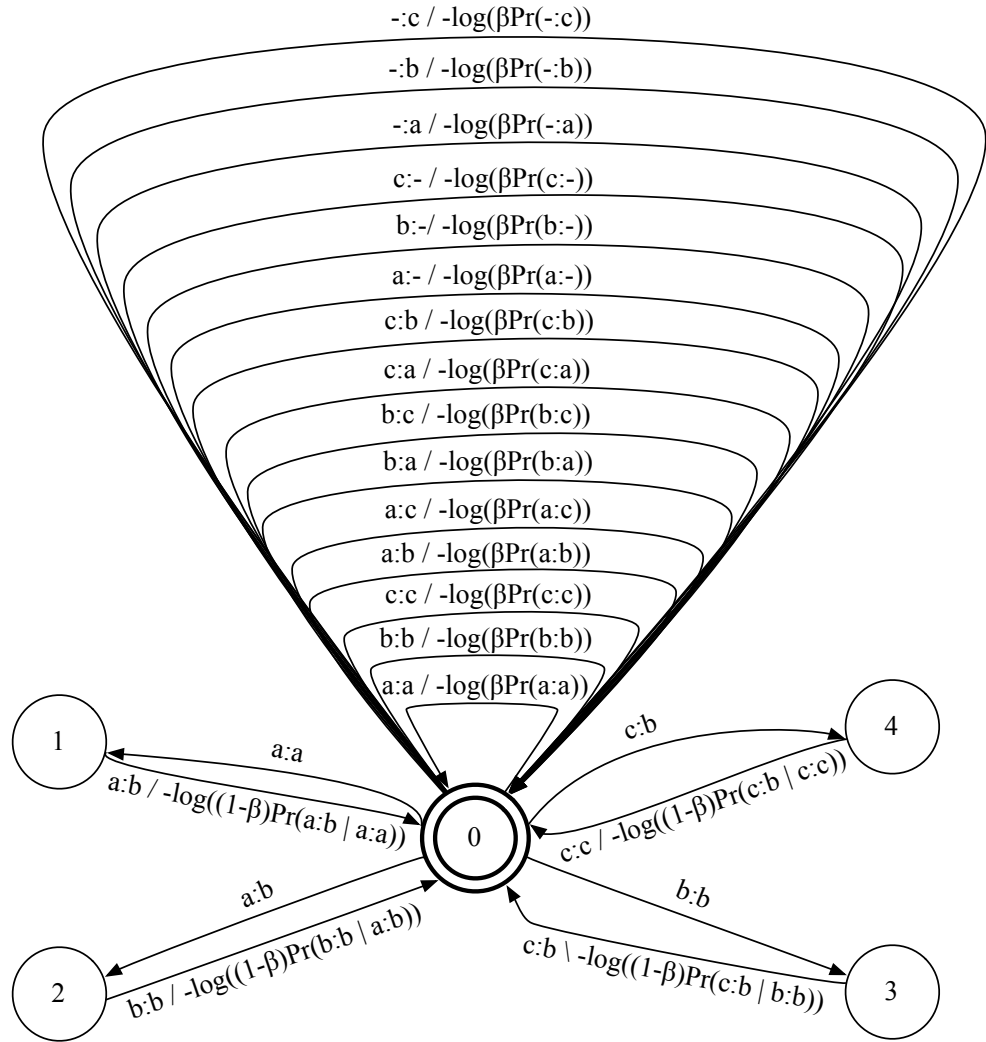


Figure 5.22: An illustration of a bigram confusion model with backoff weights. The vocabulary consists of three symbols a , b , and c . The unigram backoff arcs are derived from the unigram confusion matrix containing nine entries. Bigram arcs are added to transition out of the start state, to an isolated state and then back to the start state. Only one arc in the bigram sequence requires a weight, defined as the negative logarithm of the bigram probability. The backoff weight, β is applied to all unigram probabilities to determine the influence of the backoff weights. A factor of $(1 - \beta)$ is also applied to the bigram probabilities.

Number of Unique Bigrams Observed	Total number of Bigrams Observed	Total Number of Arcs
6785	73452	8810

Table 5.8: Bigram occurrence statistics in the isolated word dataset. Phoneme pairs (bigrams) are used to populate a bigram confusion matrix before being used to build a bigram confusion model. Figures represent an average over the six folds.

all n -best decoded phoneme strings where $n = \{1, 3, 5, 7, 9, 11, 13, 15\}$ with 11-best strings achieving the best word accuracy. The timing offset classification algorithm is also used on the unigram and bigram confusion matrices. The bigram confusion results presented in Figure 5.23 give a word accuracy of 53%, an improvement over the previous unigram confusion system of approximately 3.2% (using a timing offset window of ± 0.5 standard deviations). However, it is also interesting to note that, unlike the unigram confusion work discussed previously in this chapter, the accuracies obtained with the best timing offset window of ± 0.5 standard deviations are not statistically significant over using any other offset window up to ± 3 standard deviations.

The results presented in Table 5.9 show a contingency table of correct and incorrect words for the standard approach (HMM) and the bigram proposed approach (WFST). When comparing this table to that for the unigram confusion model analysis in Table 5.6, most of the observations are expected. Firstly, the number of words that both systems correctly decoded has increased (from 578 to 589) whereas the number of incorrectly decoded words has decreased (from 445 to 426). A more interesting finding is observed in the off-diagonal elements of the matrix (HMM correct WFST incorrect, and HMM incorrect WFST correct). As the accuracy of the WFST system has improved, there are 10 fewer words that have been correctly decoded by the HMM and incorrectly decoded by the WFST. Furthermore, the number of words decoded incorrectly by the HMM but decoded correctly by the WFST has increased by 24 — indicating that the WFST system is able to correctly decode 24 more words using bigram confusions which the standard approach would

decode incorrectly.

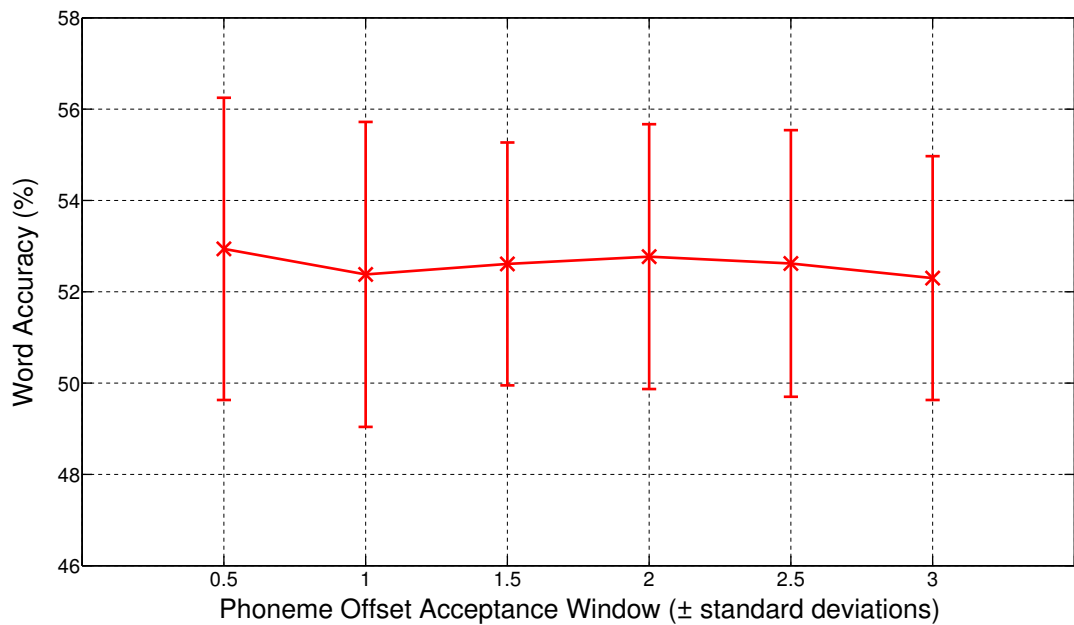


Figure 5.23: Word-level results for the isolated word dataset using a new bigram confusion model. Results are derived from cross-fold validation over six repetitions. These experiments have been conducted using the best confusion matrix smoothing method from the results presented in Section 5.3.

		WFST	
		Correct	Incorrect
HMM	Correct	589	163
	Incorrect	78	426

Table 5.9: A contingency table to provide a comparison between the HMM and WFST approaches. The ‘HMM’ approach consists of a standard phoneme HMM recogniser, and the ‘WFST’ approach is our system using the confusion WFST cascade using a *bigram* confusion model.

5.5 Summary

We have presented a new approach to confusion modelling for automated lip-reading using a WFST cascade. Initial WFST experiments using the confusion matrix output from the standard approach were significantly inferior to the baseline (60% word accuracy), with a word accuracy of 21.4%. We attribute this to the poor quality of the estimated confusion matrix. The alignments that are produced by the dynamic programming algorithm take no account of the time registration of the symbols aligned, leading to possible spurious confusions being identified. To overcome this problem, a timing offset classification algorithm is presented in Section 5.3.2.1. Firstly, at training time, a timing registration offset mean and standard deviation are estimated from the training data. When a new confusion is being considered, a timing offset window for a given ground-truth phoneme defines the boundaries for which the confusion can be considered ‘genuine’. If the offset falls outside of this boundary, it is discarded and considered as a ‘spurious’ confusion. Word accuracy dramatically improves with the introduction of the timing offset algorithm to 49.7%.

Further work described in Section 5.4 focuses on improving the confusion model to improve the word accuracy. Firstly, an adaptive confusion technique is proposed. Here, the confusion probability matrix is updated in an iterative process with the errors produced by the WFST cascade being used to update confusion probabilities. However, there are limitations that have been identified during this work with recognition accuracy on the unseen data declining on each iteration of the confusion model update. This is likely because the decoded words (one out of the 211 words in the vocabulary) can be very different from the ground-truth words, which will introduce spurious confusion patterns. Finally, we extended the current confusion model to use contextual information with a bigram confusion model, which improved on the unigram confusion model results (from 49.7% to 53%).

However, limitations with the dataset have been identified. With 1256 words in the corpus divided over six repetitions, the number of observed bigrams is very small, resulting in a sparse bigram confusion matrix. We use an additional back-

System	Word Accuracy (std. deviation)
Standard Approach HMM System as shown in Figure 5.1 (Baseline 1)	60.0% (4.2)
Phone decoding followed by string-matching (Baseline 2)	20.2% (1.4)
WFSTs with identity confusion matrix (to avoid $-\infty$ log probabilities on off-diagonal elements, a small probability mass is added to every element)	35.4% (2.3)
WFST Confusion System with confusion from a symbolic alignment	21.4% (3.3)
WFST Confusion System with timing confusion matrix and top decoding only and base smoothing	46.1% (1.0)
WFST Confusion System with timing confusion matrix and n -best decodings combined into a WFST using Algorithm 3	36.0 (0.9)
WFST Confusion System with timing confusion matrix and n -best decodings combined into a WFST using Algorithm 4	49.7 (1.6)
WFST <i>Adaptive</i> Confusion System with timing confusion matrix and learning rate (α) set to 0.1	49.8% (0.8)
WFST <i>Bigram</i> Confusion System with a timing confusion matrix with the backoff weight (β) set to 0.7	53.0% (3.3)

Table 5.10: All word-level recognition results obtained during the set of experiments in the isolated word task

off parameter to scale the unigram and bigram probabilities and find that the best recognition accuracy can be obtained by using a back-off value where the unigram probabilities dominate over the bigram probabilities (by 70% to 30%). This indicates that the sparsity of the bigram confusion matrix is having an effect on its ability to perform corrections.

The simplicity of the isolated word task is important because it enables us to focus on the core technique of modelling confusion patterns without interference from longer-range coarticulation effects. However, we have shown that the lack of data available to train a confusion model also has a detrimental affect on the ability to model strong patterns of confusions.

Chapter 6

Lip-reading for Continuous Speech

6.1 Motivation and Aims

Chapter 5 presents a new approach to lip-reading recognition, using a trained phonemic confusion model to correct the decoded string produced by the standard ASR system. Performance on the isolated word task using this technique was lower than the conventional approach. There are several possible reasons for this, but the most likely one is the sparsity of data available, with only six repetitions of 211 different words provided. A bigram confusion model was proposed in Section 5.4.2, where the unigram confusions are extended to include contextual information. However, with limited data, the number of observed bigrams in the corpus is too small to reliably model contextual confusions (the degree of this sparsity is illustrated in Table 5.8).

This chapter presents work on a much larger corpus of continuous speech data. Section 6.2 presents work using a single speaker from the LILiR corpus, a dataset used in recent lip-reading work [Lan et al., 2010]. We demonstrate that, although this data provides more ‘realistic and fluent speech’ in a richer context, it provides poor word accuracies. We examine the most likely cause for these poor lip-reading accuracies.

Until now, the automated lip-reading community has primarily focussed on refin-

ing classification techniques and features to improve recognition accuracy. However, most work has concentrated on restricted tasks (i.e. isolated letters or digits), or small amounts of continuous speech data. Here, we record a new dataset consisting of 3000 sentences of natural speech (described in Section 4.4). The availability of such a large corpus invites an interesting question — how much training data do we need to reach the peak recognition rate for automated lip-reading? We also explore the use of phoneme-to-viseme mappings in automated lip-reading and provide comparisons to using the standardised acoustic unit of speech, phonemes.

6.2 LILiR Continuous Speech Task

The complete LILiR dataset (described in Section 4.3) consists of 20 speakers reciting 200 sentences each from the Resource Management corpus. We continue our work using a single speaker to avoid the influence of inter-speaker variability on recognition (e.g. different speaking rates, different accents etc), and, therefore, use the first speaker from this multi-speaker corpus.

Firstly, the data were split evenly into five folds providing 40 sentences per fold. Two groups of folds were used in the standard approach experiments: training — consisting of four folds, and testing — consisting of one fold. Cross-fold validation was performed whereby each fold was used as the testing data with all other folds being used for training. The language models (a phoneme bigram language model for the phoneme recogniser and a word bigram language model for the word recogniser) were trained using the 160 training sentences for each cross-fold experiment. For the standard ASR approach, we build 44 monophone left-to-right HMMs (one for each phoneme) with an additional silence HMM (*sil*) used at the start and end of each utterance. The progression to continuous speech introduces an additional HMM to model any short pauses between words (*sp*). This model consists of a single emitting state to enable the decoder to recognise a minimum of one short pause frame, and its parameters are copied from the middle state of the silence model. The HMMs

are initialised in a *flat-start* procedure and trained using 20 iterations of the Baum-Welch re-estimation algorithm. We tested many different HMM topologies, with the number of states ranging from 3 to 13 and the number of mixture components from 1 to 15, and with many grammar scale factor and insertion penalty values. Only relevant results are presented here. Figure 6.1 and Figure 6.2 show phoneme accuracy and word accuracy respectively on the LILiR dataset. Both results are significantly worse than those achieved in the isolated word task. Word accuracy peaks at $\sim 9\%$ and phoneme accuracy peaks at $\sim 32\%$. Audio word and phoneme accuracies are 75.40% and 63.45% respectively. The decrease in word performance is likely due to the expansion of the grammar task. Unlike with the isolated words, the continuous speech decoding process uses a less-restrictive grammar model (i.e. a bigram grammar model), providing probabilistic influence on the decoded path, rather than forcing the decoder to output one word from the vocabulary, as was the case in the isolated word task.

Following the techniques used for the isolated word task, the phoneme decoder uses a bigram phoneme language model to aid recognition. One interesting observation is the low phoneme accuracy that is achieved here (shown in Figure 6.1). In the isolated word task, the phoneme accuracy was approximately 60%, a figure that could be considered as typical for audio speech recognition. However, the peak phoneme accuracy reported in Figure 6.1 is much lower (approximately 32%). This is likely to be caused by the lack of data to train the phoneme bigram language model (only 160 sentences), and to reliably train the parameters of the HMM.

The proposed approach presented in Chapter 5 relies on the presence of strong phonemic confusion patterns to correct decoded strings and, hence improve word recognition accuracy. However, with such low phoneme accuracies as those observed in Figure 6.1, it may be impossible to find strong confusion structure. There are three factors that have contributed to the inferior performance compared with the isolated word task: the increase in the complexity of the problem (i.e. from a simple, predictable language structure to an utterance of unknown length), the extra co-

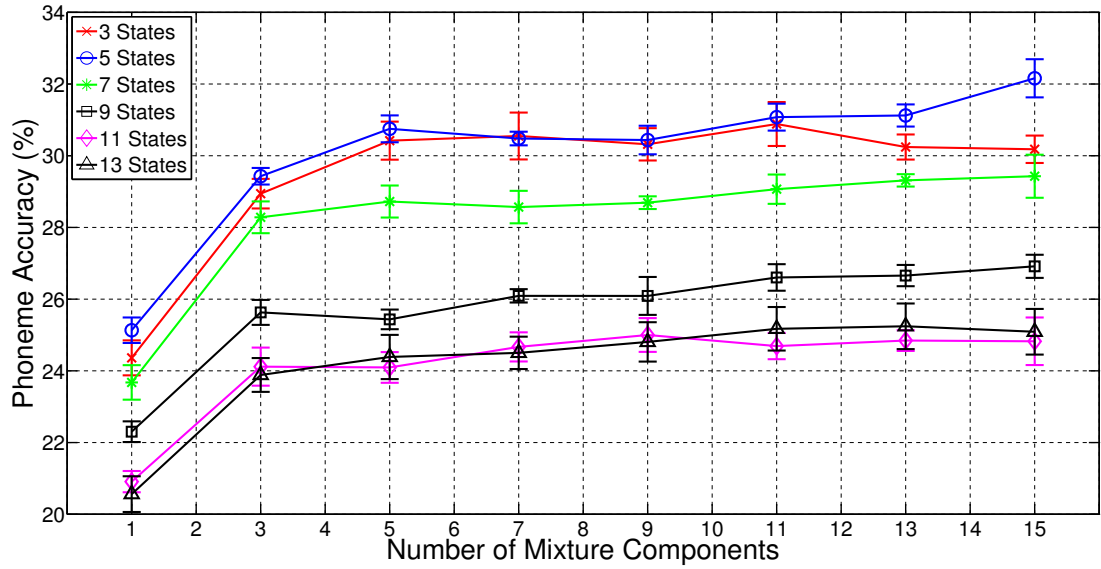


Figure 6.1: Phoneme-level recognition results on the LILiR dataset using a network of phoneme HMMs - a technique considered to be state-of-the-art in automated lip-reading. A phoneme bigram language model is used to improve recognition accuracy.

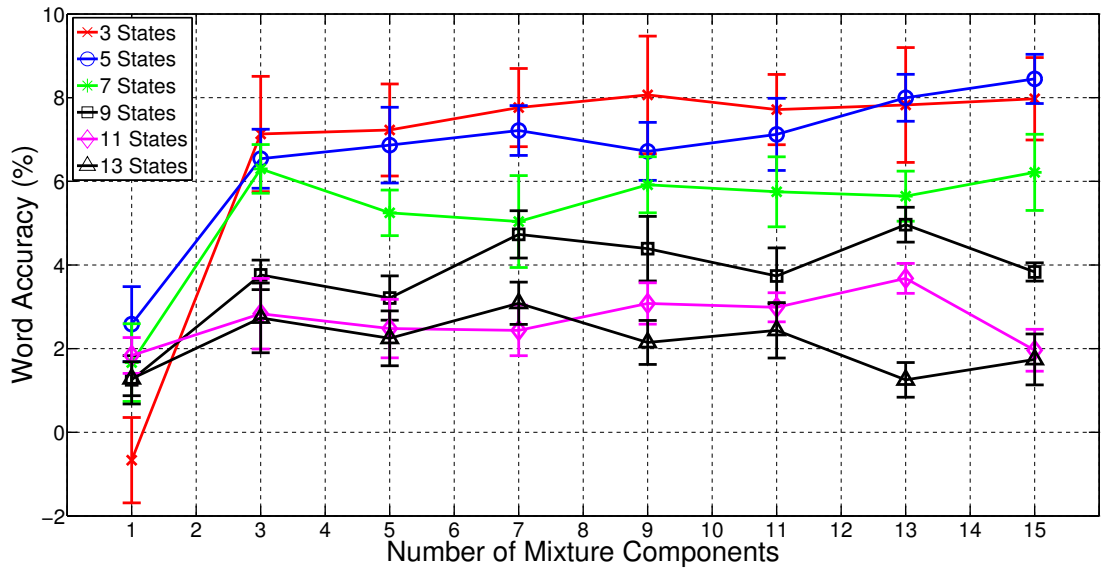


Figure 6.2: Word-level recognition results on the LILiR dataset using the standard approach to automated lip-reading (HMMs).

articulation from continuous speech, and lack of training data.

Although an audio ASR system can achieve good recognition accuracy with limited data, most state-of-the-art audio ASR systems are trained on large speech

databases to enable optimal performance. With this in mind, it was decided that this work should focus on recording a larger audio-visual dataset from a single speaker to obtain the peak recognition rates.

6.3 Issues in Modelling for Lip-Reading

Previous work in automated lip-reading has been performed on tasks of varying difficulty. For example, the work presented in [Cox et al., 2008] achieves high speaker-dependent accuracy on the 26 letters of the alphabet. However, the task with more natural speech conducted in [Lan et al., 2010] achieved only 35% *viseme* accuracy. In this work, we utilise a new dataset (RM-3000) consisting of 3000 sentences from a single speaker (described in Section 4.4). This provides us with a large database of natural speech to explore the peak recognition accuracy that can be achieved with speaker-dependent automated lip-reading.

Much attention in visual speech recognition has focused on building viseme models. Visemes (discussed in more detail in Section 3.2.2) are visually similar phonemes that are mapped to the same visual class, producing a smaller class set than phonemes (e.g. 14 visemes instead of 45 phonemes). We utilise our new speaker-dependent video dataset to investigate another fundamental question: is it optimal to use phonemes or visemes as modelling units for lip-reading?

6.3.1 Viseme mapping and Homophenous Words

Homophones are words that sound the same but have a different meaning (e.g. “for” and “four”, or “bored” and “board”). Similarly, *homophenous* words look visually identical, but sound different (e.g. “bat”, “pat” and “mat”). Some studies have calculated that as many as 40%–60% of English spoken words could be homophenous, something that poses a significant problem for visual speech recognition [Berger, 1972]. In this work, we define a set of words to be homophenous if they all have the same viseme transcription. Of the 979 different words spoken in our database, 106

Viseme Class	Mapped Phonemes
V1	/b/ /p/ /m/
V2	/f/ /v/
V3	/t/ /d/ /s/ /z/ /th/ /dh/
V4	/w/ /r/
V5	/k/ /g/ /n/ /l/ /ng/ /hh/ /y/
V6	/ch/ /jh/ /sh/ /zh/
V7	/eh /ey/ /ae/ /aw/ /er/ /ea/
V8	/uh/ /uw/
V9	/iy/ /ih/ /ia/
V10	/ah/ /ax/ /ay/
V11	/ao/ /oy/ /ow/ /ua/
V12	/aa/
V13	/oh/
V14	/sil/

Table 6.1: Description of the Fisher phoneme-to-viseme mappings to collapse 45 phoneme classes into 14 viseme classes. A viseme is reserved for the silence model (/sil/)

(10.83%) are homophenous. However, because of the uneven distribution of words over the 3000 sentences, these homophenous words account for 8988 (34.42%) of all word tokens out of a total of 26114 tokens. Therefore, even with perfect viseme recognition, the recogniser’s performance could be as low as 65.58% if it were always to make the wrong choice between a group of homophenous words. We have used the Fisher mapping from phonemes to visemes [Fisher, 1968], shown in Table 6.1. Notice that it maps 45 phonemes to only 14 visemes.

6.3.2 Experiments

As with previous experiments, we use left-to-right HMMs for recognition, an approach that has been successful for both audio ASR and automated lip-reading [Cox et al., 2008; Luetttin and Thacker, 1997; Hilder et al., 2009; Rabiner, 1989]. As with previous work, we perform segmentation using the ‘flat-start’ initialisation

process. We build monophone models of recognition units in all cases and train using 20 iterations of the embedded Baum-Welch re-estimation algorithm. We perform an exhaustive search to find the optimum number of states (three) and mixture components (19 per state). The RM-3000 dataset is split randomly into 10 folds (providing 300 sentences per fold). Following conventions used for recognition of continuous speech, a short-pause model (*sp*) is tied to the centre state of the HMM that models silence to allow short-duration silence between words.

In our recognition system, there are two sets of probabilities that determine the decoded output sequence of words or units: the probabilities of the input features being generated by the HMMs of the units, and the language model probabilities (we use bigram probabilities of either words or units). Because visual speech does not convey as much information as audio speech, best performance in lip-reading is achieved by placing more weight on the language model probabilities when compared with audio ASR. This is done by using a *grammar scale factor* (GSF) as described in Section 2.4.4 to boost the importance of the language model probabilities over the model or visual probabilities.

Our 3000-sentence speaker-dependent dataset is taken as a subset from the 8000-sentence RM corpus. Therefore, for these experiments, we build word, viseme and phoneme bigram language models that have been trained on the remaining (unseen) 5000 RM sentences.

Figure 6.3 shows the results obtained for audio and visual recognition as a function of the number of sentences used as training data. Audio Phoneme shows the phoneme accuracy obtained on audio data using 45 phoneme units, and Audio Viseme the accuracy on audio data when the 45 phonemes are mapped to 14 visemes. Visual Phoneme and Visual Viseme show the accuracy under the same conditions but using visual rather than audio data. As expected, we can achieve very good phoneme recognition accuracy on single-speaker audio data. It is interesting to note that viseme recognition accuracy is actually a little lower (about 2%) than phoneme accuracy when using audio data, despite the number of viseme classes

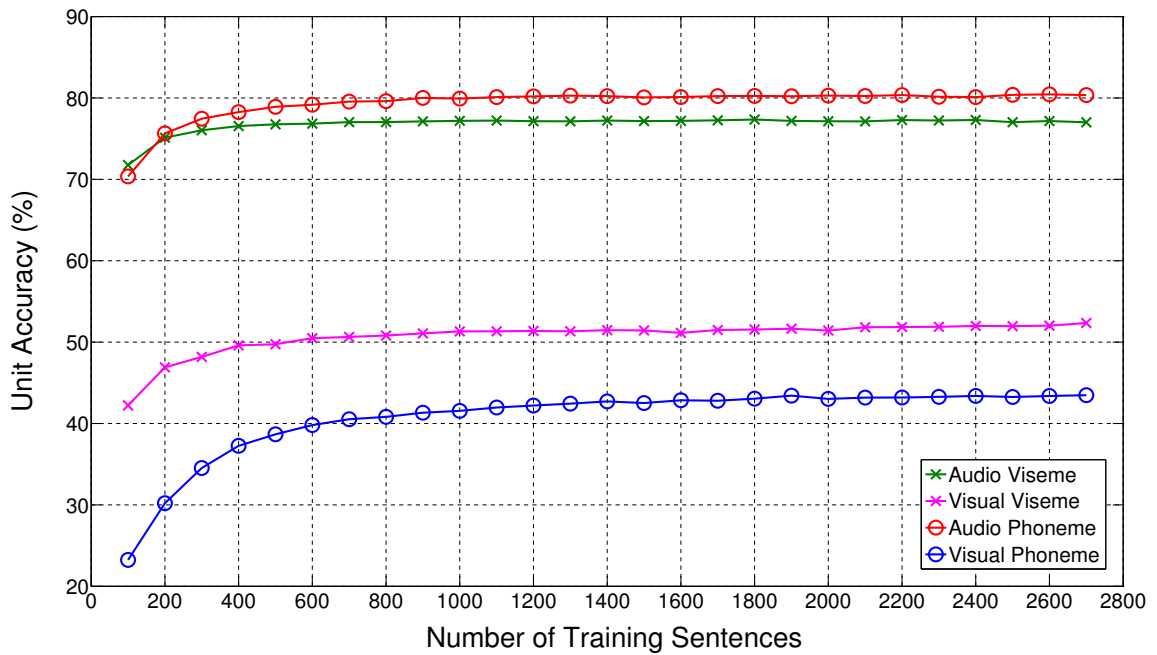


Figure 6.3: Unit recognition performance on 3000 speaker-dependent sentences from the Resource Management Corpus (RM). Phoneme and viseme (unit) accuracies are shown for audio and visual speech recognition as a function of the size of the training set.

being less than one third of the number of phoneme classes. We can attribute this to the fact that the phoneme-to-viseme mapping groups phonemes that have very different acoustic features, and so the variation in the features within the classes would be very high and therefore difficult to model. Using visual data, the situation is reversed: we obtain better accuracy (near 10% better) using visemes rather than phonemes, which is what we would expect from using the phoneme-to-viseme mapping, which is designed to combine visually similar phonemes into a lower number of relatively homogeneous classes. However, the accuracy is significantly lower than that obtained with audio data.

Figure 6.4 shows what happens when we use either phoneme or viseme units to recognise words. For audio data, it is not a surprise to find that the best performance (about 96% accuracy) is obtained when the units are phonemes. When viseme units are used with audio data, performance suffers considerably (about 15% lower). There are, presumably two reasons for this: the accuracy on viseme units is lower anyway,

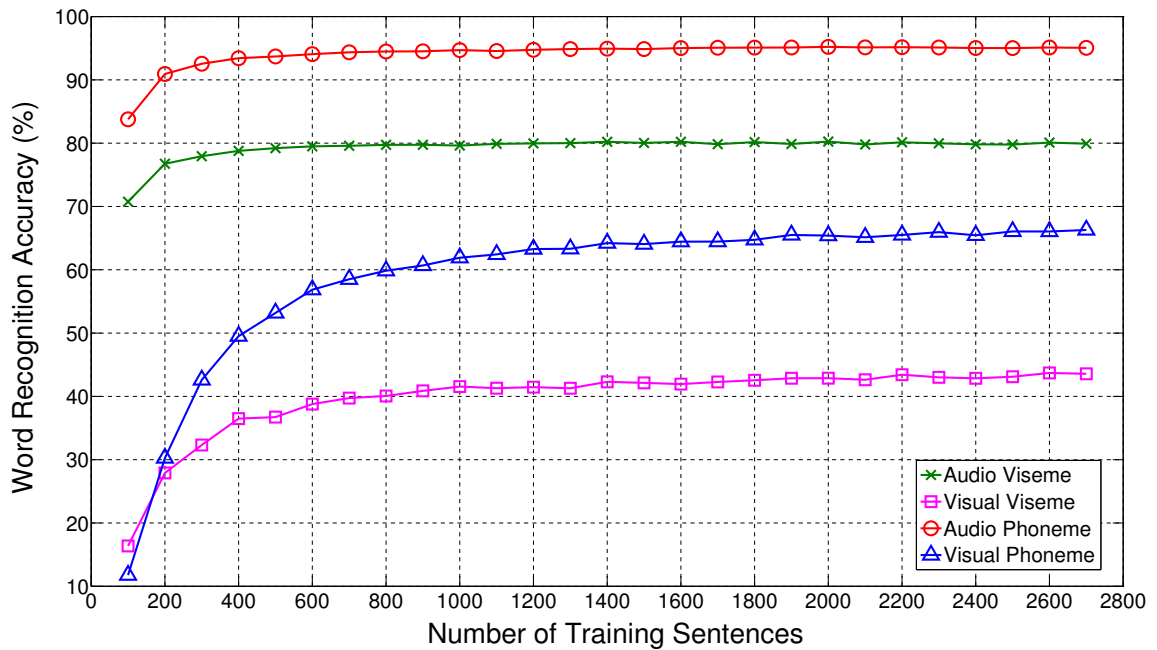


Figure 6.4: Word recognition performance on 3000 speaker-dependent sentences from the Resource Management Corpus (RM). Word recognition results are shown for audio and visual speech recognition using both phoneme and viseme models as a function of the size of the training set.

and visemes introduce homophenous words resulting in ambiguity and thus lower performance.

It is interesting that, for visual data, much better word recognition results are obtained using phoneme units than viseme units, about 15% better. Of course, the presence of homophenous words accounts for some of this loss in performance, but it was expected that the combination of higher accuracy on viseme units (as shown in Figure 6.3) and a high value of the grammar scale factor (we found a GSF of 30 gave the best results for word recognition on visual data using visemes) would mean that visemes would be superior for word recognition.

For phoneme, viseme or word recognition, Figure 6.3 and Figure 6.4 show that with audio data, optimum recognition performance is obtained with about 600 training sentences, whereas for visual data, performance is still increasing when the full set of 2700 sentences has been used for training. This confirms our suspicion that

lip-reading requires much more data for the same task than audio ASR, because of the inherent ambiguity in the classes and the noise in the features. It is also interesting to note that word recognition performance is about the same using both viseme and phoneme units when only 200 sentences are used for training, but performance using phonemes outstrips performance using visemes as more training sentences are added. This may be explained by the fact that phonemes require more training to achieve maximum performance because there are three times as many phoneme classes as viseme classes.

6.3.2.1 Analysis of the effect of the language model on viseme decoding

The operation of the grammar scale factor (GSF) is of interest in understanding the interaction between unit and word recognition. It seemed to us that the language model would be essential in aiding the recogniser choose the most likely word from a set of homophenous words. However, if the GSF is too large, it would override the information from the unit HMMs at the expense of the language model, and accuracy might be adversely affected.

To test this theory, we synthesised a set of “perfect” features for a set of sentences from our corpus. The mean vector of each state of the sequence of viseme HMMs that corresponded to the transcribed word sequence of a sentence was output as a “feature”. This resulted in a sequence of synthetic “features” that actually matched perfectly to the sequence of viseme HMMs corresponding to the sequence of words in the sentence. However, there was inherent ambiguity present in the different possible segmentations of the viseme string, and also in the presence of homophenous words. Figure 6.5 shows the effect on the word accuracy of increasing the GSF when these features were decoded by the recogniser. When the GSF is 0, any word is allowed to follow any other word with equal probability, i.e the language model is having no effect, and the word accuracy is rather low (92%) because of the ambiguity of different possible segmentations and the presence of homophenous words. If the GSF is increased to 1, the language model now chooses more correctly from

the possible segmentations and from the sets of homophenous words, and accuracy increases. As an example, the words “TEXAS” and “SENSORS” have the same visemic transcription (V3 V7 V5 V3 V10 V3). When the GSF is zero, the system recognises “SENSORS” as “TEXAS”, which is an error. When the GSF is one, this error is corrected under the influence of the language model. However, as the GSF is further increased, performance deteriorates, because the language model “overrules” the evidence from the features and chooses word sequences that have high bigram probabilities.

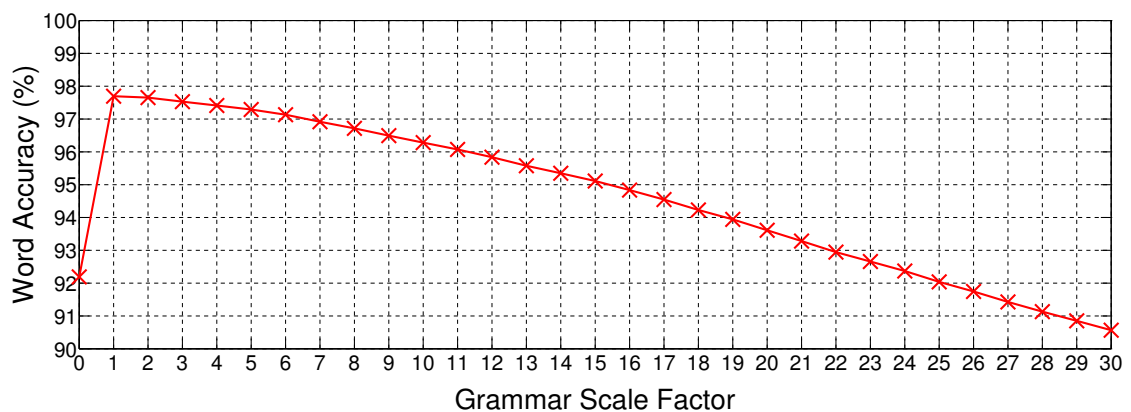


Figure 6.5: The effect of the language model on word accuracy when the recogniser is given “perfect” features (i.e. ground-truth features generated by the trained HMMs). With a grammar scale factor of zero, the bigram word-pairings are preserved but each has equal probability. Thereafter, the bigram language model has an increasing influence.

6.3.2.2 Analysis of the role of different viseme classes in recognition

Viseme classes vary in both the accuracy with which they are recognised and in the degree of ambiguity that they introduce into word transcriptions. To investigate the effect on viseme and word recognition accuracy, we ran an experiment in which each of the viseme classes shown in Table 6.1 was created in turn whilst keeping the remaining set of phonemes intact (i.e. for any experiment, there was a single viseme class constructed whilst the other classes remained as phonemes). The resulting set of units was trained, and then used to perform word recognition. Using these results, we are able to rank the viseme mappings in order of their effect on both unit

and word accuracy.

Figure 6.6 shows how the word and unit recognition accuracy is affected by the grouping together of phonemes to form each of the viseme classes. The first accuracy along the set of viseme classes is a basic recogniser using all 45 phonemes separately. After this, each viseme class is established according to the Fisher phoneme-to-viseme mapping from Table 6.1 and recognition is performed. For unit accuracy, there is consistency in most of the results with most around the 43% to 44% region. Although some mappings provide an increase in unit accuracy, none of the mappings improve the word accuracy over using phonemes. An example of this is shown with V1, containing the bilabial phonemes ($/b/$, $/p/$, and $/m/$) that are considered indistinguishable in the visual speech signal. This mapping gives over a 1% increase in unit accuracy, however, it fails to provide the improvement in accuracy that we would have envisaged.

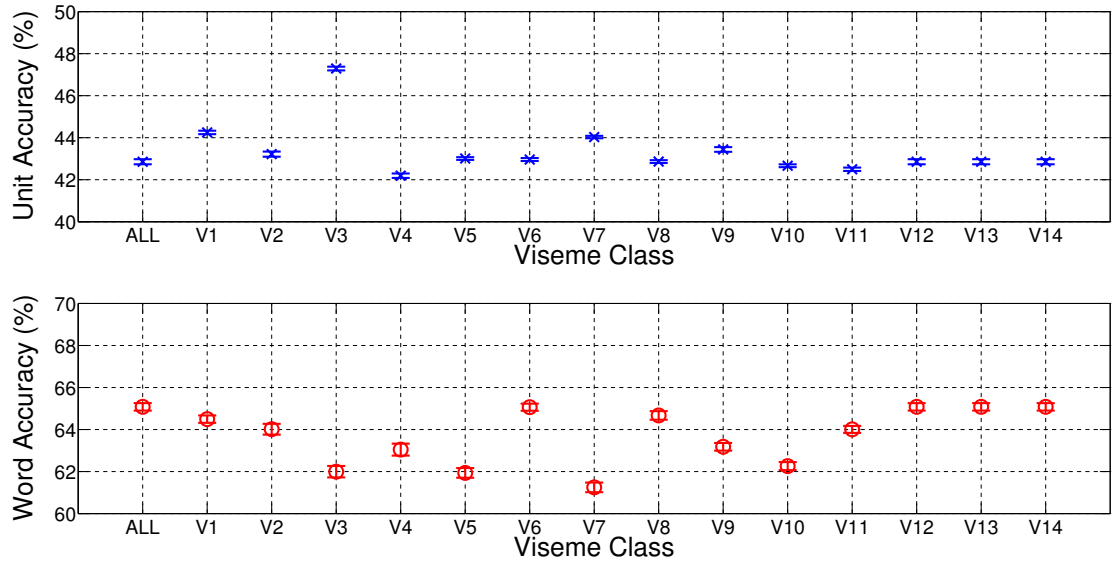


Figure 6.6: Unit and word recognition accuracies that are achieved for forming an isolated viseme from the set of visemes in the Fisher mapping (shown in Table 6.1). Each viseme group is formed in turn whilst keeping the other classes remain as phonemes to see if unit or word accuracy could be improved using a partial phoneme-to-viseme mapping.

Word accuracies reported in Figure 6.6 have more variation across the viseme groupings. The viseme classes that include the most phonemes in a single group

come from V3, V5, and V7. Figure 6.7 shows the average unit and word accuracies for each viseme class size in the Fisher mapping (shown in Table 6.1). The unit accuracy is highest when we use the Fisher classes that contain six phonemes. This is likely due to the reduced symbol set that large classes require. For word recognition, there is an interesting relationship between the class size and word accuracy. Higher class sizes (six and eight) give lower word accuracies than smaller classes. This decrease in accuracy may be due to the increase in the number of homophenous words that are introduced by the large class sizes.

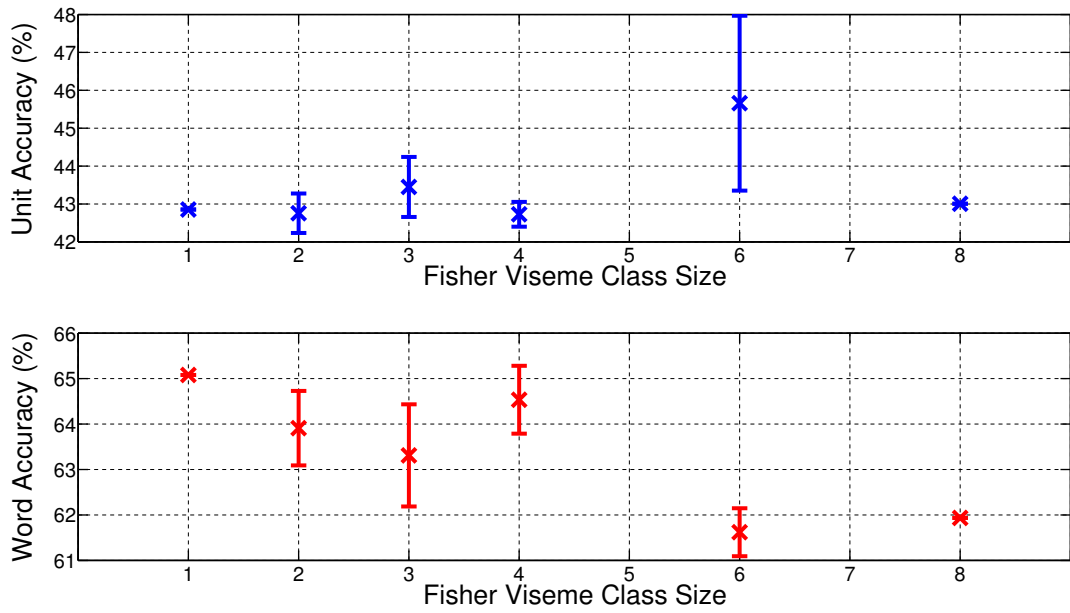


Figure 6.7: Average unit and word accuracies for the different viseme class sizes. The Fisher phoneme-to-viseme mapping does not have any classes that consist of five or seven phonemes.

Figure 6.8 demonstrates how the systems are affected by the *progressive* categorisation of phonemes into visemes. Here, we sorted the visemes by their effect on accuracy and introduced the classes in order of effect on accuracy (from Figure 6.6), lowest effect first. We start with 45 phonemes (ALL) and progressively reduce the number of phonemes by introducing a new viseme and performing recognition. The initial mappings have no effect on word or viseme accuracy (V12, V13 and V14 are one-to-one mappings). Thereafter, viseme recognition accuracy improves but word

recognition accuracy declines. Mapping lip-rounded shapes together (i.e. V6) does not affect either word or viseme accuracy because of the nature of this distinctive lip shape and its weaker coupling with neighbouring sounds (co-articulation). However, mapping sounds that are highly dependent on neighbouring sounds improves viseme accuracy (i.e. V3) whilst lowering word recognition (i.e. V3 and V5). Furthermore, there is a strong correlation between the steep improvement/decline in viseme/word accuracy and the distribution of visemes over the 3000 recorded sentences.

To take this point further, Figure 6.9 shows the distribution of visemes over the 3000 sentence dataset. The uneven distribution is because viseme classes have a different number of phonemes mapped to a single class. Comparing Figure 6.9 to the results for word recognition in Figure 6.8, we see that the phoneme-to-viseme mappings that have the greatest effect on performance are those that occur the most frequently (e.g. for V3, V5, V7, V9 and V10). Although viseme recognition performance increases as these large classes are introduced, because more phonemes are mapped to a small number of classes, these mappings also introduce more ambiguity into word transcriptions and hence, word accuracy decreases.

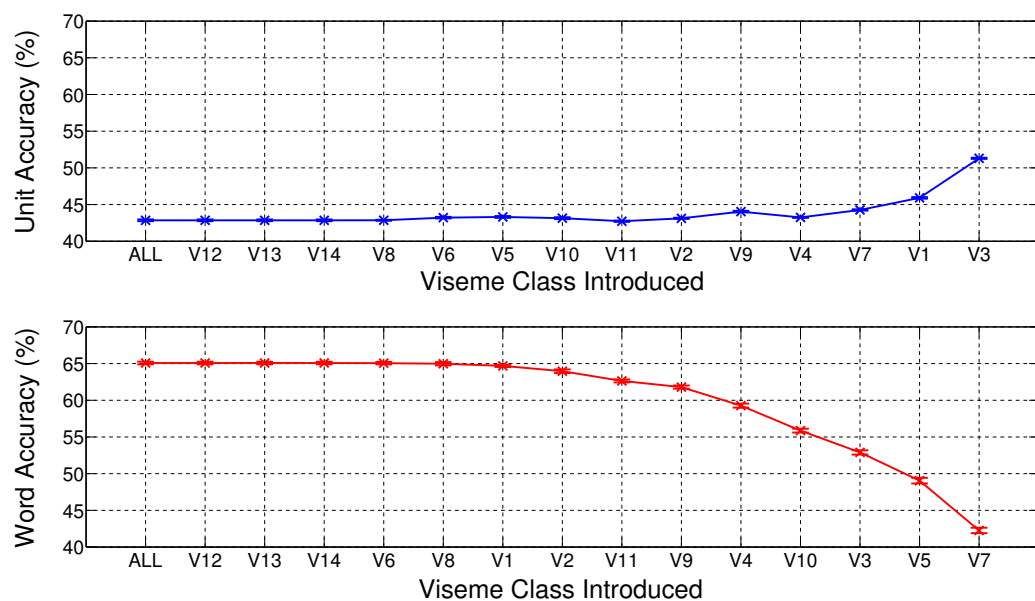


Figure 6.8: The effect on recognition accuracy of progressively mapping phonemes to visemes. Each viseme class is made in turn, reducing the number of classes progressively from 45 to 14. ALL represents using all 45 phoneme classes.

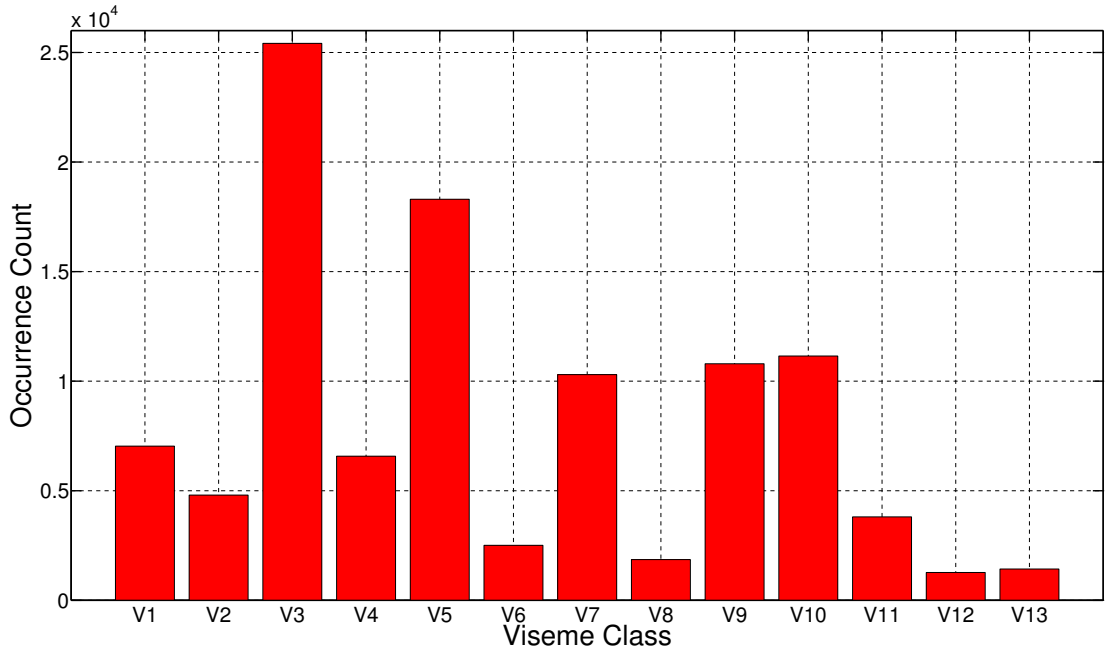


Figure 6.9: Distribution of Fisher visemes over RM-3000 dataset

6.3.2.3 A Data-Driven Phoneme-to-Viseme Mapping

Phoneme-to-viseme mappings have been the subject of much debate in the perception of visual speech, with many different mappings being suggested. Although the Fisher mapping describes a good viseme grouping, there may be a more optimal mapping to maximise the performance of an automated lip-reading system. So an interesting idea is to use confusion information that has been obtained during recognition to guide the formation of viseme classes, rather than defining viseme classes from phonetic principles. We first perform phoneme recognition using the standard approach to automated lip-reading. As in previous experiments, we build a network of phoneme HMMs and use a phoneme bigram language model to direct the recogniser to a decoded phoneme sequence. The HTK toolkit [Young, 2001] provides a post-recognition tool to align the decoded sequences to their corresponding ground-truth sequences to produce a phoneme confusion matrix, here a matrix containing 45^2 frequency entries (we ignore insertions and deletions here). We use the computed confusion matrix to identify strong confusion patterns. Phonemes that are

confused with each other are then grouped together into viseme classes.

Confusion matrix entries are ranked in descending count order. Starting from the confusion pair (i.e. aligned phonemes that come from the ground-truth and recognised sequence) with the largest count, phonemes are grouped together iteratively through the ranked frequency counts until all phonemes have been assigned to a class. Because this method is purely based on the confusions that occur during recognition, the final number of viseme classes that are produced by the algorithm cannot be controlled. The grouping algorithm defines three rules by which to classify phonemes into the set of viseme classes:

1. If both phonemes in a confusion pair are not already members of a viseme class, they are assigned to a newly created viseme class.
2. If a confusion pair occurs in which only one of the phonemes is already assigned to a viseme class, the other phoneme is added to the classified phoneme's viseme class.
3. If a confusion pair is found in which both phonemes have already been assigned to different classes, the confusion is ignored.

Using this set of rules on the RM-3000 dataset, the algorithm used 39 grouping steps to construct **seven** different viseme classes (including a class reserved for the */sil/* phoneme). These grouping steps are shown in Table 6.2 and the final groupings are shown in Table 6.3. The results presented in Figure 6.10 correspond to the mapping steps shown in Table 6.2 with the trends in both unit and word accuracy following the Fisher mapping strategy shown in Figure 6.8. However, the gain in unit accuracy and drop in word accuracy are much more emphasised than the previous work using the Fisher mapping because the final mapping is to seven symbols rather than the 14 symbols. Therefore, the unit accuracy increases monotonically to 60% because there are fewer possible symbols, and the word accuracy declines from approximately 60% to 10% as the symbol set is much reduced and many more

ambiguous word transcriptions (homophenous words) are introduced. In comparison with the Fisher mapping described in Section 6.3.2.2, the data-driven results show a large increase in unit accuracy (from 51% to 60%) which is expected because the number of classes is reduced from 14 to 7. Word accuracy using the data-driven mapping only reaches 60% on the first step which is still lower using phoneme classes (approximately 65%). For completeness, Figure 6.11 shows results from constructing each of the *final* phoneme-to-viseme mappings produced by this approach *independently* (i.e. constructing each viseme class whilst keeping all other phonemes unmapped). As expected, unit accuracy is dramatically improved with the larger consonant mapping (V1), but this introduces ambiguity into word transcriptions, leading to a significant fall in word accuracy.

Step	Phonemes to Viseme Class	Step	Phonemes to Viseme Class
1	{/t/ /n/} to V1	20	{/g/} to V1
2	{/s/} to V1	21	{/ao/} to V2
3	{/ih/ /iy/} to V2	22	{/r/} to V1
4	{/z/} to V1	23	{/ng/} to V2
5	{/ae/ /eh/} to V3	24	{/ch/} to V1
6	{/d/} to V1	25	{/hh/} to V1
7	{/ax/} to V2	26	{/ah/} to V2
8	{/b/ /p/} to V4	27	{/ea/} to V3
9	{/m/} to V4	28	{/ay/} to V2
10	{/dh/} to V1	29	{/ia/} to V2
11	{/k/} to V1	30	{/sh/ /jh/} to V6
12	{/ey/} to V3	31	{/th/} to V5
13	{/w/} to V1	32	{/er/} to V2
14	{/y/} to V1	33	{/aa/} to V2
15	{/l/} to V1	34	{/aw/} to V3
16	{/f/ /v/} to V5	35	{/uh/} to V1
17	{/oh/} to V2	36	{/ua/} to V2
18	{/uw/} to V1	37	{/zh/} to V1
19	{/ow/} to V2	38	{/oy/} to V2

Table 6.2: Algorithm steps in the data-driven phoneme-to-viseme mapping algorithm. The final viseme class set is shown in Table 6.3.

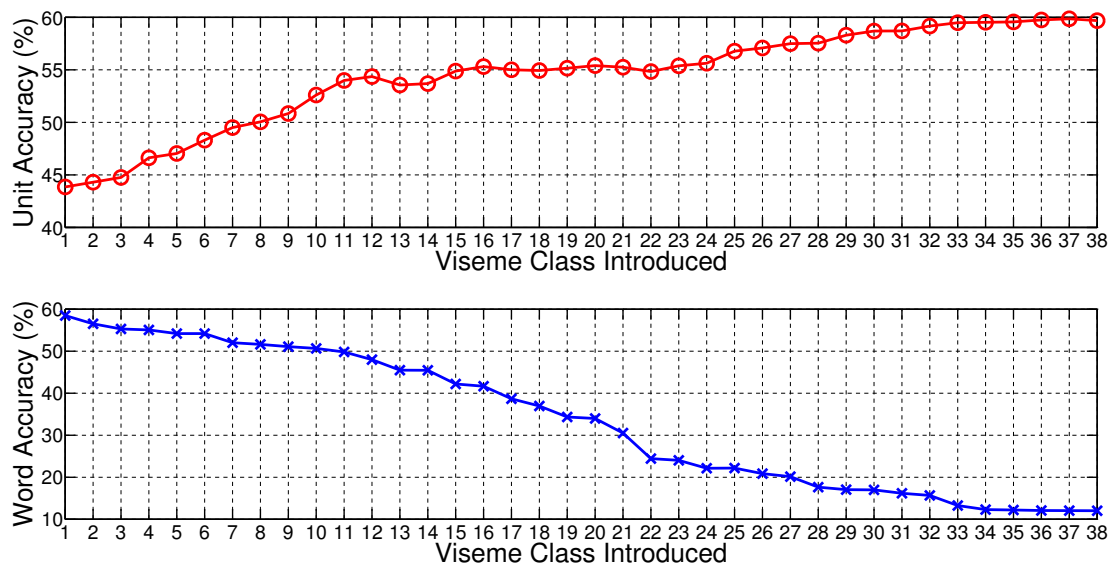


Figure 6.10: Unit accuracy (top) and word accuracy (bottom) recognition results at each of the 38 grouping steps described in Table 6.2. Each step is dependent on the previous mapping whereby the phonemes that have been assigned to viseme groups previously are maintained and a new grouping is introduced according to the steps in Table 6.2.

Viseme Class	Mapped Phonemes
V1	/t/ /n/ /s/ /z/ /d/ /dh/ /k/ /w/ /y/ /l/ /uw/ /g/ /r/ /ch/ /hh/ /uh/ /zh/
V2	/ih/ /iy/ /ax/ /oh/ /ow/ /ao/ /ng/ /ah/ /ay/ /ia/ /er/ /aa/ /ua/ /oy/
V3	/ae/ /eh/ /ey/ /ea/ /aw/
V4	/b/ /p/ /m/
V5	/f/ /v/ /th/
V6	/jh/ /sh/

Table 6.3: A description of the viseme classes that are produced by the data-driven phoneme-to-viseme mapping algorithm. Here, 44 phonemes are collapsed into six viseme classes. An additional silence viseme is also used as the seventh viseme class.

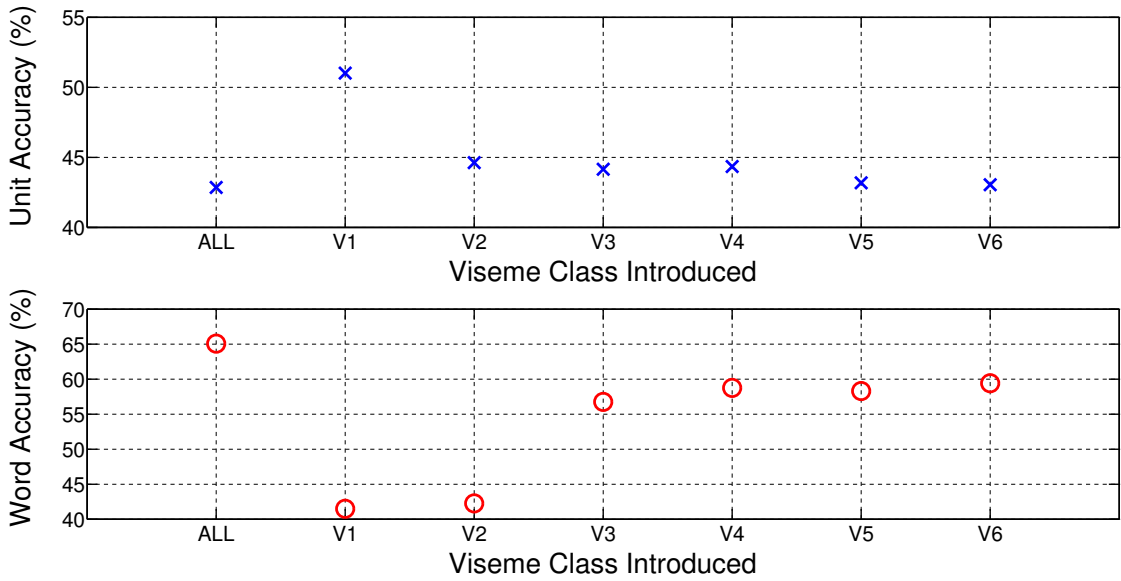


Figure 6.11: Unit accuracy (top) and word accuracy (bottom) recognition results for the data-driven phoneme-to-viseme mapping. Here, the final viseme classes (shown in Table 6.3) that are produced by the mapping steps shown in Table 6.2 are explored. Standard unit and word recognition is performed on the viseme classes independently (only a single viseme class is constructed and all other phonemes remain unmapped).

6.3.2.4 A Principled Phoneme-to-viseme Mapping

So far, we have had little success in finding a phoneme-to-viseme mapping strategy that improves the recognition accuracy for our RM-3000 task. We describe a final experiment in which we define viseme classes using sounds that can be easily grouped due to their indistinguishable lip shapes. Unlike the experiments conducted using a progressive strategy in Figure 6.8, and Figure 6.10 (whereby the phoneme set is progressively reduced to the number of visemes), we perform independent recognition experiments, each including only the viseme classes that are required for the specific step. Table 6.4 describes the viseme mappings used for this experiment with the step number to correspond to the recognition results in Figure 6.12. Notice that, unlike the data-driven approach where we could not control the final number of viseme classes, we group the phonemes into viseme classes as described in Table 6.4 but maintain all other phonemes as individual classes and produce mappings that have fewer (35) or more (44) total number of classes. One might assume that

obvious phoneme mappings, would yield much higher word and unit accuracies, however, this is not true. Step 5 in Table 6.4 describes the mapping of two bilabial phonemes $/b/$ and $/p/$. Here, all examples of these two phonemes are mapped to the same viseme class with all other phonemes remaining in their own classes. The results for step 5 in Figure 6.12 show a small improvement in unit accuracy from 42.85% to 43.55% which is purely due to the reduction in the number of possible symbols to recognise. However, with the introduction of ambiguous transcriptions (homophenous words), the word accuracy for step 5 suffered a small decline from 59.33% to 59.30%. Other mappings (such as step 2 containing fricatives and stops) improved unit accuracy significantly but always adversely affected the word accuracy with none of the mappings able to achieve better word accuracy than using phoneme classes. The partial viseme mappings shown here are inferior in both unit and word accuracy to the Fisher viseme mappings that were described in Section 6.3.2.2, with nine of the partial Fisher mappings achieving over 60% word accuracy (a figure that is not exceeded by here using the principled mapping technique).

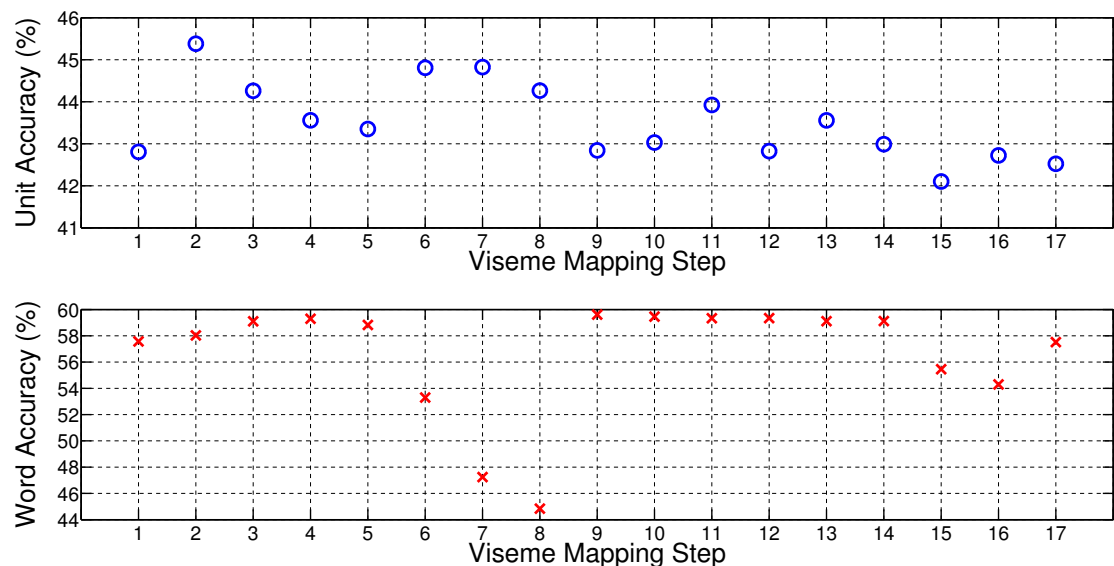


Figure 6.12: Recognition results using the principled viseme mappings. The 17 mappings are shown in Table 6.4. Here, each viseme class is constructed independently at each step described in Table 6.4. During each mapping step, all other phonemes outside of the subject viseme class are maintained as separate classes. Standard error is too small to be shown for these results.

Step	Class Description	Viseme Classes Introduced	Total Number of Classes
1	Fricatives	{/s/ /sh/}, {/z/ /zh/}, {/th/ /dh/} , {/f/ /v/}	41
2	Fricatives and stops	{/t/ /d/}, {/b/ /p/}, {/g/ /k/}, {/jh/ /ch/}, {/s/ /z/}, {/sh/ /zh/}, {/f/ /v/}, {/th/ /dh/}	37
3	Stops	{/t/ /d/}, {/b/ /p/}, {/g/ /k/}, {/jh/ /ch/}	41
4	b and p mapping	{/b/ /p/}	44
5	f and v mapping	{/f/ /v/}	44
6	Fricatives, stops, and vowel mapping 1	{/t/ /d/}, {/b/ /p/}, {/g/ /k/}, {/jh/ /ch/}, {/s/ /z/}, {/sh/ /zh/}, {/f/ /v/}, {/th/ /dh/}, {/ax/ /aa/ /ae/}	35
7	Fricatives, stops, and vowel mapping 2	{/t/ /d/}, {/b/ /p/}, {/g/ /k/}, {/jh/ /ch/}, {/s/ /z/}, {/sh/ /zh/}, {/f/ /v/}, {/th/ /dh/}, {/eh/ /iy/ /ih/}	35
8	Fricatives, stops, and vowel mapping 3	{/t/ /d/}, {/b/ /p/}, {/g/ /k/}, {/jh/ /ch/}, {/s/ /z/}, {/sh/ /zh/}, {/f/ /v/}, {/th/ /dh/}, {/aw/ /ay/ /ow/}	35
9	g and k mapping	{/g/ /k/}	44
10	jh and ch mapping	{/jh/ /ch/}	44
11	s and z mapping	{/s/ /z/}	44
12	sh and zh mapping	{/sh/ /zh/}	44
13	t and d mapping	{/t/ /d/}	44
14	th and dh mapping	{/th/ /dh/}	44
15	Vowel mapping 1	{/ax/ /aa/ /ae/}	43
16	Vowel mapping 2	{/eh/ /iy/ /ih/}	43
17	Vowel Mapping 3	{/aw/ /ay/ /ow/}	43

Table 6.4: Description of the principled phoneme-to-viseme mappings. Each mapping referred to using a ‘Mapping ID’ which corresponds to the recognition results shown in Figure 6.12. The total number of classes is calculated as the sum of number of viseme classes and the number of remaining phoneme classes (i.e. phonemes that are not assigned to a viseme class in the mapping).

6.4 Conclusions

This chapter has presented work on two continuous speech datasets. Firstly, we used an existing dataset that has been used in previous lip-reading work, the LILiR corpus. However, we found that lip-reading word and phoneme recognition accuracies with this dataset are too poor to work with.

We recorded a new dataset to eliminate the possibility that lack of data is affecting our results. The new dataset, RM-3000, consists of 3000 sentences recorded from a single speaker over multiple sessions (described in Section 4.4). We use this dataset to determine the amount of training data required for peak automated lip-reading accuracy. The highest word accuracy produced by our speaker-dependent lip-reading system is 66%. Although this may seem quite low in ASR terms, it should be compared with a measured accuracy for *human* lip-reading of 18.4% on monosyllabic words [Hilder et al., 2009]. We have also found that lip-reading recognition performance was still increasing slightly after 2700 sentences had been used for training, whereas audio speech recognition achieved peak accuracy after about 600 sentences were used. This shows that more training data is necessary for lip-reading because of the inherent ambiguity in the visual features.

Recent work in automated lip-reading has primarily focussed on the assumption that visemes are the most appropriate unit of visual speech. However, with many different works proposing different phoneme-to-viseme mappings, there seems to be disagreement amongst the visual speech community. We examined the use of visemes in automated lip-reading with a large, speaker-dependent task and found that, although visemes outperform phonemes when measuring *unit* accuracy, they introduce more ambiguity amongst *word* transcriptions, hence, reducing word accuracy.

Our work then analysed the effect of different viseme mappings on unit and word accuracy. Firstly, we performed independent and progressive tests using the Fisher phoneme-to-viseme mapping, and concluded that the most frequently occur-

ring visemes provide the largest reduction in word accuracy. We then developed our own phoneme-to-viseme mapping which was derived from a phoneme confusion matrix after recognition. We found that, given our grouping algorithm, the number of symbols was reduced significantly to just seven symbols, causing unit accuracy to improve (by reducing the number of possible decoded symbols) but also causing word accuracy to decline (by introducing more ambiguous word transcriptions). Finally, we derive a phoneme-to-viseme mapping using a principled approach in Section 6.3.2.4, grouping phonemes into large and small groups (of as little as two phonemes) using our understanding of visual speech. We find that, although some of the phonemes mapped to a single viseme class are indistinguishable in thought, they are in fact separable with most mappings providing higher unit accuracy (because the number of possible symbols is reduced), but lower word accuracies (after introducing homophenous words).

Phoneme-to-Viseme Mapping	Word Accuracy	Unit Accuracy
Best Partial Fisher Mapping (Section 6.3.2.2)	64.49%	47.29%
Final Data-Driven Mapping (Section 6.3.2.3)	51.02%	59.43%
Best Partial Principled Mapping (Section 6.3.2.4)	45.38%	59.62%
Phonemes	65.08%	42.85%

Table 6.5: Comparison of the best unit and word accuracies that are produced by the phoneme-to-viseme mappings used in this work.

The work presented in this chapter has explored the use of phoneme-to-viseme mappings to improve the recognition accuracy of an automated lip-reading system. Table 6.5 compares the best unit and word recognition accuracies that are obtained for each mapping strategy (i.e. the best result from using any of the partial mappings). We have discovered the counter-intuitive result that the best word recognition accuracy in automated lip-reading is achieved by using phoneme classes,

even though some phonemes can not be visually distinguished. There are two likely reasons for these findings. Firstly, the reduction of the number of classes introduces homophenous words that add ambiguity amongst words and cause some words to be incorrectly recognised. However, the most significant contribution comes from the wide variety of co-articulatory information of the units. In the viseme case, phonemes which have indistinguishable lip shapes might not necessarily have similar co-articulatory features. With the large amount of data used in this work, there are many examples of each phoneme available to train individual phoneme HMMs. With the phoneme HMM system producing the best word recognition accuracies, the system is able to more accurately model individual phoneme units with respect to their distinct co-articulatory properties. These results provide an interesting foundation for future work.

Chapter 7

Confusion Modelling for Continuous Speech

7.1 Motivation and Aims

Chapter 5 presented a new approach to lip-reading, using a confusion model to correct the noisy decoded phoneme sequence that has been produced by a phoneme recogniser. The best recognition accuracy that could be achieved by the new (proposed) approach was still slightly inferior to the standard approach accuracy. However, there was evidence to suggest that this technique had potential, with a large word accuracy gain (29.5%) over a shortest-cost alignment using dynamic programming. The work presented in Chapter 5 used a specially-recorded, isolated word dataset consisting of a 211-word vocabulary, which was a relatively simple task designed to focus on modelling confusion patterns in lip-reading. However, with limited data available, the accuracy of the confusion model is limited. This lack of data also has a detrimental effect on the accuracy of the bigram confusion model introduced in Section 5.4.2, with data sparsity becoming more apparent as the number of symbols is increased.

In this chapter, we overcome the data sparsity problem by recording a new, large

audio-visual corpus consisting of 3000 sentences (described in Section 4.4), which is capable of providing a much richer set of confusions, and, hence, stronger confusion patterns for the confusion WFST to model. We start by performing experiments with the techniques introduced in Chapter 5 on the RM-3000 dataset and examine the confusion patterns that are observed in this process. The availability of a larger dataset is also a catalyst for work to extend the standard approach system (currently using monophone HMMs) to use context-dependent models (triphone HMMs). This produces a set of confusions with more structure, leading to a more accurate correction module. The triphone recognition system also opens up further room for exploration and in Section 7.3.4, we extend the current confusion system by using *recognition lattices*, which provide a richer set of possible hypotheses compared to the previous n -best approach.

7.2 A Monophone System

To measure the improvement in word recognition accuracy, we first need a baseline system. Chapter 5 presents the standard approach using a network of HMMs to recognise isolated words (Section 5.2) and a similar approach is used with the RM-3000 dataset. Section 6.3.2 describes the standard approach setup for the work presented in this chapter with word recognition results shown in Figure 6.4. Following the approach described in Section 5.3 and the evidence provided in Section 6.3.2, we continue to use phoneme units as they have proven best to translate into the best word accuracy. As in the experiments conducted in Sections 5.2 and 6.3.2, we explore a large number of possible left-to-right HMM topologies for phoneme models and experiment with the number of mixture components per state to maximise recognition accuracy. The best word recognition accuracy that was achieved using the standard approach was **66.29%** (shown in Figure 6.4). This used a word bigram language model that had been pre-trained on the held-out 5000 sentences from the RM corpus.

7.2.1 The Proposed Approach

At the heart of our proposed approach, is the confusion module. Our system (shown in Figure 5.1) uses the output from a standard approach phoneme recogniser as the input to a WFST cascade. The most important part of the cascade for our work is the confusion transducer which models the substitutions, insertions, and deletions. We presented a set of techniques in Section 5.3.2 for estimating the lip-reading confusion matrix using the confusions that are present in an offline training phase.

The recognition accuracy of a speech recognition system is based on a count of correctly decoded symbols versus incorrectly decoded symbols after aligning the symbol strings using dynamic programming. However, it does not describe the individual counts of substitutions, insertions, and deletions. In confusion modelling, it is most desirable to observe substitutions because they form strong patterns in the confusion matrix. In lip-reading, many sounds are not observed on the lips because speech involves the use of articulators that cannot be seen, and this leads to deletions in the decoded sequence. Table 7.1 shows statistics that are produced from the best phoneme accuracy shown in Figure 6.3. Although these figures have been produced by the best phoneme recogniser, they expose the issues that are faced when reporting unit accuracy (using phonemes or visemes) for automated lip-reading. The number of deleted tokens (phonemes) is more than three times the number of insertions and over 12000 than the number of substituted tokens. Furthermore, the number of deleted tokens represents approximately one quarter of the total number of tokens, meaning that a deletion occurs once for nearly every four other tokens. Figure 7.1 illustrates the extent to which these deletions affect the decoded phoneme sequences, with many neighbouring phonemes being deleted in long strings of as many as seven phonemes — usually making up whole words. Figure 7.2 illustrates this problem further. Here, we count the number of consecutive deleted and inserted phoneme sequences (i.e. the number of phonemes being inserted/deleted in a row). As one would expect, there are many more deletion counts than insertion counts. Most deleted/inserted sequences are between one and three phonemes long but there are

still 10318 deletions that are more than three phonemes long. For a confusion model to have a chance of correcting these long sequences of deletions, a high-order n -gram model would be required to fit over the widest window of deleted phonemes. However, even with the large database used here, n -gram sparsity is still a significant issue.

SENTENCE: REDO FIGURES FOR HORNE

GT: r iy d uw f ih g ax z f ao hh ao r n

REC: d uw f eh t f ao

SENTENCE: HOW FAST IS THE SAMPLE

GT: hh aw f aa s t ay z dh ax s aa m p l

REC: hh aw f ay s iy s ah b l

SENTENCE: WOULDNT IT HAVE TAKEN LONGER WITH JASON

GT: w uh d n t ih t ae v t ey k ax n l oh ng ax w ih dh jh ey s ax n

REC: w uh t iy v l ey l oh ng ax b ax l ey z

SENTENCE: WHERE'S THE SWORDFISH'S HOME PORT

GT: w ea r z dh ax s ao d f ih sh ih z hh ow m p ao t

REC: w ah n t ao f ih s ah m p ao t

SENTENCE: HOW NEAR TO THE MARS IS IT

GT: hh aw n ia ay t uw dh ax m aa z t ax ih t

REC: w iy hh aw r iy t b ay th w ih n t

Figure 7.1: Five examples of phoneme transcriptions produced by the phoneme recogniser with the best phoneme accuracy. ‘SENTENCE’ gives the word transcription of the sentence, ‘GT’ is the ground-truth phoneme sequence, and ‘REC’ is the recognised phoneme sequence.

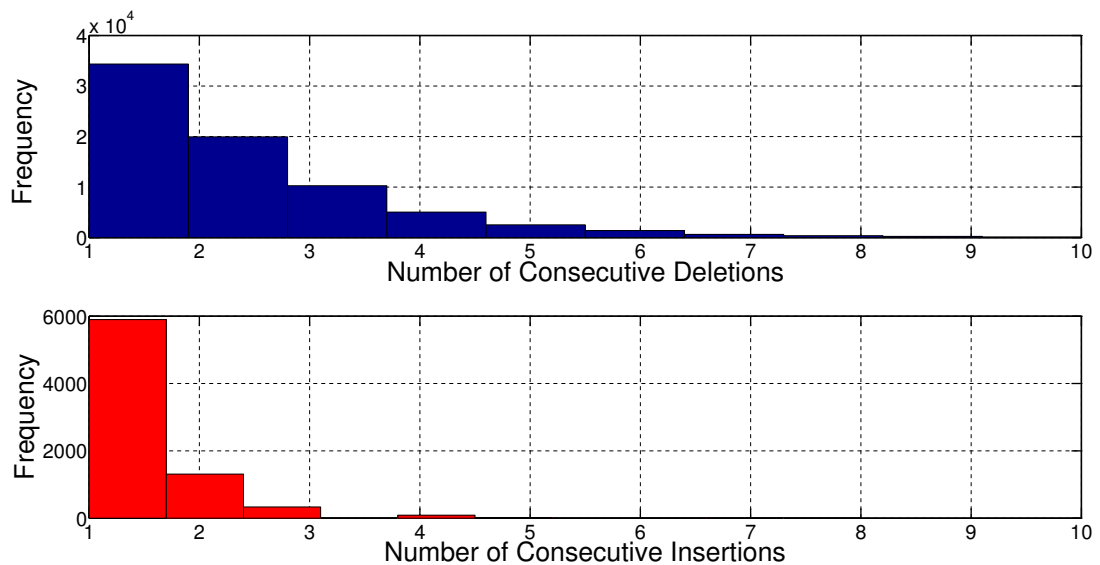


Figure 7.2: The number of consecutive deletions and insertions that are produced by the standard phoneme recogniser. The length of each deleted/inserted sequence is recorded and represented in a histogram plot. The number of deletions is much larger than the number of insertions as described in Table 7.1

	Best Accuracy Configuration
Insertion Penalty	15
No. of Correct Tokens	135,924 (51.68%)
No. of Deleted Tokens	69,712 (26.51%)
No. of Substituted Tokens	57,364 (21.81%)
No. of Inserted Tokens	21,939 (8.34%)
Total No. of Tokens	263,000
Phoneme Accuracy (%)	43.34

Table 7.1: Phoneme Recognition statistics produced by the monophone standard approach using a phoneme bigram language model.

7.2.2 Minimising the number of deleted phonemes

The highest phoneme recognition accuracy that can be achieved with the standard approach is 43.34%. However, this recognition accuracy figure fails to describe the nature of the phoneme confusions that appear in the decoded sequences. Table 7.1 shows a set of statistics that are important for our confusion modelling work. Although this configuration achieves the best phoneme accuracy, it also introduces a large number of deletions and insertions. Furthermore, Figure 7.2 shows that the deleted phonemes can be in long sequences of up to six phonemes. As it is

easier to delete information rather than insert *unknown* information, we wish to find another configuration where the number of deletions (and ideally, insertions too), is minimised but accuracy is not significantly lower than the best configuration.

An insertion penalty (described in Section 2.4.4) is used in the standard approach ASR system to control the ratio of inserted phonemes to deleted phonemes in our recognition system using a fixed value to penalise the transition from one HMM to another. Figure 7.3 shows the relationship between insertion penalty values and the frequencies of insertions and deletions, and also with the phoneme accuracy. When the phoneme accuracy reaches its peak at an insertion penalty value of 15, there are fewer deletions, but these are still greater than the number of insertions. However, we also want to avoid a high number of insertions (and low phoneme accuracies) that come with insertion penalties greater than 30. Therefore, we choose an insertion penalty of 30 to have fewer deletions than insertions with only a small decline in phoneme accuracy (7.25%) from the best configuration.

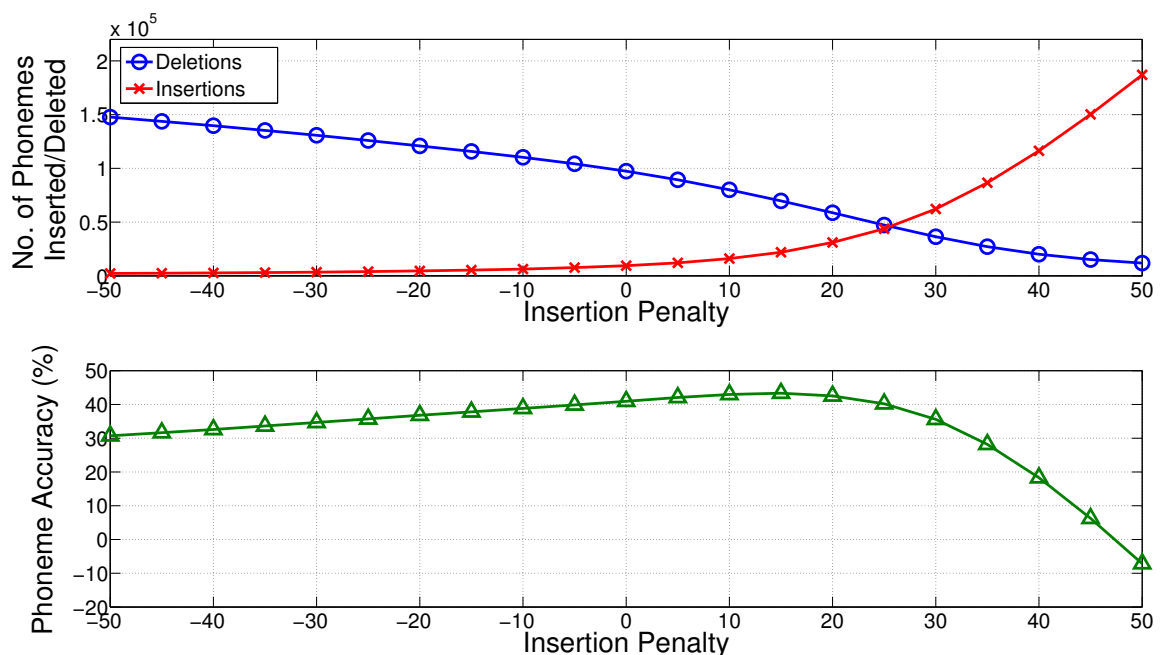


Figure 7.3: The top plot shows the number of inserted and deleted phonemes as a function of insertion penalty values in the HTK standard approach recogniser. Below this is a phoneme accuracy plot as a function of the same insertion penalty. This figure is used to find the optimal insertion penalty to minimise the number of deletions whilst also maximising the phoneme accuracy.

Table 7.2 compares the recognition statistics from the best phoneme accuracy (with an insertion penalty of 15) and the new phoneme accuracy (with an insertion penalty of 30). With some sounds not visible on the lips, a typical visual speech confusion model would include a large number of deleted sounds. For a unigram confusion model, the deletion pattern is not likely to show much structure, causing issues when performing correction with an insertion operation. Insertions, however, are events that have been observed in the decoded sequence but are not present in the ground-truth sequence. In general, it is easier to delete spurious information than to have to add information that has been lost. Therefore, we balance the number of deletions and insertions with minimisation of the former being the priority. By experimenting with different values for the insertion penalty, the number of deleted tokens has been dramatically reduced to approximately a half of the original size (from 69,712 to 36,368) whilst the number of inserted tokens has increased by nearly three times the original amount (from 21,939 to 62,212).

	Best Accuracy	Minimised Deletions
Insertion Penalty	15	30
No. of Correct Tokens	135,924 (51.68%)	155,839 (59.25%)
No. of Deleted Tokens	69,712 (26.51%)	36,368 (13.83%)
No. of Substituted Tokens	57,364 (21.81%)	70,793 (26.92%)
No. of Inserted Tokens	21,939 (8.34%)	62,212 (23.65%)
Total No. of Tokens	263,000	263,000
Phoneme Accuracy (%)	43.34	35.60

Table 7.2: Phoneme Recognition statistics produced by the monophone standard approach using a phoneme bigram language model.

Figure 7.4 shows the improved phoneme transcriptions with the same words from Figure 7.1. Although the transcriptions include more insertions, the maximum number of consecutive insertions are much smaller, providing the confusion model with a more reliable set of confusions in a more localised space. Figure 7.5 draws a comparison between the nature of insertions and deletions in the new decoding (optimised for deletions), and the previous decoding (using the best phoneme accuracy). As expected, the new phoneme decoding has fewer deletions but these are (mostly) small in length with most sequences of less than three deleted symbols. Whilst limiting

the number of deletions, the number of insertions is increased in the new decoding but these are also small in length with most sequences under three phonemes but mostly just single symbol insertions.

SENTENCE: REDO FIGURES FOR HORNE		
GT:	r iy d uw f ih g ax z f ao hh ao r n	
ORIG:	d uw f eh t f ao	
OPT:	d r iy t y uw f ih g n f ao m	
SENTENCE: HOW FAST IS THE SAMPLE		
GT:	hh aw f aa s t ay z dh ax s aa m p l	
ORIG:	hh aw f ay s iy s ah b l	
OPT:	hh ae n f ay z dh ax dh ax s ah b l	
SENTENCE: WOULDNT IT HAVE TAKEN LONGER WITH JASON		
GT:	w uh d n t ih t ae v t ey k ax n l oh ng ax w ih dh jh ey s ax n	
ORIG:	w uh t iy v l ey l oh ng ax b ax l ey z	
OPT:	w ih n dh ax dh ax v ey eh n l oh n dh ax dh ae n s	
SENTENCE: WHERE'S THE SWORDFISH'S HOME PORT		
GT:	w ea r z dh ax s ao d f ih sh ih z hh ow m p ao t	
ORIG:	w ah n t ao f ih s ah m p ao t	
OPT:	w ah n dh ax s ao f ay z hh p ao dh ax	
SENTENCE: HOW NEAR TO THE MARS IS IT		
GT:	hh aw n ia ay z dh ax m aa z t ax ih t	
ORIG:	w iy hh aw r iy t b ay th w ih n t	
OPT:	hh aw n b iy t m aa th dh ax w ih n t	

Figure 7.4: Five examples of phoneme transcriptions produced by the standard approach recogniser with the reduced number of deletions. ‘SENTENCE’ gives the word transcription of the sentence, ‘GT’ is the ground-truth phoneme sequence, ‘OPT’ is the new recognised phoneme sequence with fewer deletions, and ‘ORIG’ is the recognised phoneme sequence produced by the phoneme recogniser with the best accuracy (Figure 7.1).

With this improved phoneme decoding, we perform the proposed approach as shown for the isolated word dataset in Section 5.3. To run preliminary experiments, we first test our approach using the most likely phoneme sequence produced by the standard approach (i.e. 1-best). The confusion model is trained using different timing offset windows (ranging from ± 0.5 to ± 3 standard deviations). Only base smoothing is used after the superior results achieved with base smoothing over other smoothing methods for the isolated word experiments. The best word accuracy using this approach is **12.79%** (shown in Table 7.3) using a ± 0.5 standard deviation timing

window. Upon inspection, it was discovered that, although there were fewer longer sequences of deletions, the overall number of deletions was still too high for the unigram confusion model to correctly insert phonemes frequently. We also explored the use of an insertion penalty in the WFST cascade to reduce the large number of insertions in the confusion model, but these experiments did not improve word accuracy.

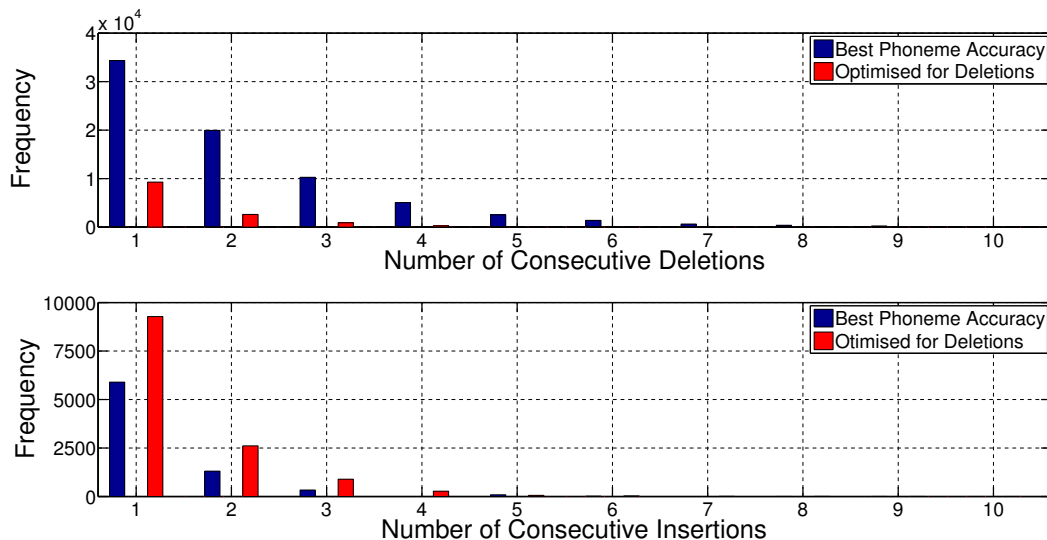


Figure 7.5: A comparison between the number of consecutive deletion/insertion sequences in the two hypotheses, using the best phoneme accuracy and using the decoded output that is optimised to minimise deletions. The top histogram plots the consecutive deletion counts and the bottom histogram shows the consecutive insertion counts

System	Word Accuracy (%)
Standard Approach with a word bigram language model	66.29
Proposed approach using WFST cascade and a bigram language model	12.79

Table 7.3: Best word-level results produced by the proposed approach using phoneme transcriptions with minimal deletions. A search was performed to find the optimal timing offset window of ± 0.5 standard deviations.

7.3 A Triphone System

Most state-of-the-art audio ASR systems take advantage of the data available in thousands of hours of speech data to train triphone HMMs. Triphones are an extension of monophone HMMs in which each model represents a single phoneme but with a left (preceding) and right (following) context (triphones are described in more detail in Section 2.4.3.2). The limited amount of data in visual speech means that lip-reading recognisers usually use monophone HMM systems. With the new, large RM-3000 corpus, we have been able to extend the standard monophone approach system to use triphone HMMs for lip-reading. Furthermore, the use of a more complex standard approach recogniser invites new possibilities in the use of weighted finite-state transducers in confusion modelling.

7.3.1 The Standard Approach using Triphone HMMs

To maintain consistency in our approach, we use HTK [Young et al., 2006] but build a triphone HMM system. Firstly, we construct a set of monophone HMMs as described in previous standard approach experiments. Context-dependent triphone HMMs are constructed using a copy of the monophone HMMs with an additional re-estimation using a triphone training transcription. We use cross-word triphones whereby each word boundary is modelled using the triphone sequences that span over two words. This improves the modelling of cross-word co-articulatory effects that are present in the visual speech signal (for more detail on cross-word triphones, see Section 2.4.3.2). State tying is performed according to a set of clustering rules so that triphones that have not been observed in the training procedure can be modelled. In this work, we use the tree-based clustering rules that are used in conventional audio ASR systems [Young et al., 2006]. As in the previous monophone work, we train the parameters of the HMMs using the embedded Baum-Welch training algorithm. We perform an exhaustive search over the number of states and number of mixture components per state to find the configuration that achieves the best word accuracy.

We also use a word bigram language model trained on the held-out 5000 sentences from the RM corpus and find the optimal grammar scale factor (set to 30). We also perform an exhaustive search to find the best insertion penalty value to produce the most accurate word decoding using the best combination of insertion and deletion counts.

Figure 7.6 presents the word recognition results for the triphone HMM system when varying the number of mixture components per state. We also performed recognition using more than 19 mixture components per state but found that word accuracy starts to decline after 19 components. Table 7.4 compares the best word recognition accuracy that can be achieved using the standard approach with monophone and triphone HMMs. It was expected that the triphone HMM system would outperform the monophone HMM system in a speech recognition task of this size. However, the large gain in word accuracy that is observed here (nearly 10%) indicates that co-articulation, which is exactly what triphones are good at modelling, is very prevalent in the visual signal.

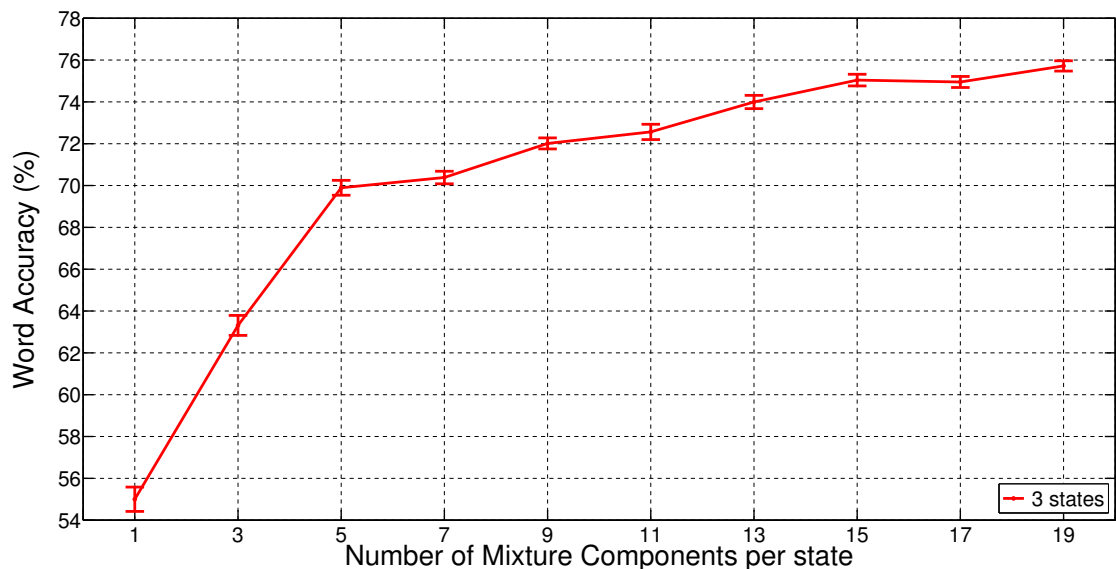


Figure 7.6: Word recognition results using the standard approach to lip-reading. As the RM-3000 dataset provides a richer training set, the standard monophone HMM results presented in Figure 6.4 can be improved by using a triphone HMM system where an HMM models a phoneme in a specific left and right context.

	Monophone System	Triphone System
No. of Correct Words	18,191 (69.66%)	20,500 (78.50%)
No. of Deleted Words	2,860 (10.95%)	2,308 (8.84%)
No. of Substituted Words	5,063 (19.39%)	3,306 (12.66%)
No. of Inserted Words	879 (3.37%)	763 (2.92%)
Total No. of Words in Ground-Truth	26,114	26,114
Word Accuracy (%)	66.29	75.58

Table 7.4: Comparison between the word recognition statistics achieved with a monophone and triphone HMM system.

7.3.2 Word-Level Confusion Model

Upon examination of the word confusion patterns that are produced by the recogniser, we also identify some interesting findings. Figure 7.7 compares five word transcriptions that are produced by the previous monophone HMM system with the current triphone HMM system. The increased accuracy from triphones gives a stronger structure to the word confusions. For example, the confusion between ‘HORNE’ and ‘ON’ in the first transcription, and between ‘WHENS’ and ‘WHERE’S’ in the fourth transcription. These plausible confusions are of keen interest in our work. In almost all cases, the triphone decoding provides more information than the monophone decoding. What is particularly impressive is the reduced number of deletions that are produced by the triphone system.

The word confusions are phonetically similar, and this means that when the words are written out as phonemes and aligned, the phoneme confusions are more accurate. Here, we experimented with building a word-level confusion model to see if we could improve word accuracy at a higher level. However, if we built a word-level confusion matrix using the same approach as that implemented in the phoneme confusion matrix for the isolated word task, we would require a matrix with $979^2 = 958,441$ entries. With most of these elements set to zero, confusion sparsity would be a significant issue.

GT:	REDO	FIGURES	FOR	HORNE
MONO:	REDO	FIGURES	FOR	
TRI:	REDO	FIGURES	FOR	ON
GT:	HOW	FAST	IS	THE SAMPLE
MONO:	HOW	FAR	IS	THE SUBS
TRI:	HOW	FAR	IS	THE SAMPLE
GT:	WOULDNT	IT	HAVE	TAKEN LONGER WITH JASON
MONO:	WOULDNT	IT	HAVE	TAKEN LONGER WITH TEXAS
TRI:	WOULDNT	IT	HAVE	TAKEN LONGER WITH TEST
GT:	WHERE'S	THE	SWORDFISH'S	HOME PORT
MONO:		THE	SWORDFISH	HOME PORT
TRI:	WHENS		SWORDFISH'S	HOME PORT
GT:	HOW	NEAR	IS	THE MARS TO IT
MONO:	HOW		MANY	TO BERING STRAIT
TRI:	HOW	NEAR	IS	THE MIDWAYS

Figure 7.7: Five examples of word transcriptions produced by a monophone HMM standard approach system and a triphone HMM standard approach system. ‘GT’ is the ground-truth sentence, ‘MONO’ is the recognised sentence produced by the monophone HMM approach, and ‘TRI’ is the recognised sentence produced by the triphone HMM approach.

To overcome this, we adopt a similar approach to that taken with the sparsity problem encountered in the phoneme bigram confusion model (presented in Section 5.4.2). The confusion WFST is built using all of the non-zero elements of the confusion matrix only. Therefore, if a confusion has never been observed at training time, we assume that there is a zero probability of the confusion system correcting. Although this assumption could be considered as naive, the regular occurrence of plausible confusions (e.g. ‘WHENS’ for ‘WHERE’S’, and ‘WHAT’ for ‘WHAT’S’) should enable the recogniser to correct popular plausible confusions, and hence, improve the word recognition accuracy. We build a confusion WFST with the same topology as performed in the experiments presented in Chapter 5, using a single state cyclic model. The weights are calculated using the proportion of the specific confusion frequency which is normalised by the total number of confusions for the ground-truth word. Table 7.5 shows the confusions that are observed for two typical ground-truth words. The ground-truth word ‘WHAT’S’ has been confused with

‘WHAT’, ‘WHERE’, and itself, and the ground-truth word ‘WHERE’S’ is confused with ‘WHERE’, ‘WHYS’, ‘WHAT’S’ and itself. The probability of each confusion ($Pr(W, W^*)$) is calculated using Equation 7.1 where $C(W, W^*)$ is the number of times that the ground-truth word W is confused with the recognised word W^* , and $C(W)$ is the total number of times that the ground-truth word W has been encountered over all recognised words. This probability is converted to a negative log cost and used in the confusion WFST (shown in Figure 7.8).

$$Pr(W, W^*) = \frac{C(W, W^*)}{C(W)} \quad (7.1)$$

Ground-Truth Word	Recognised Words
WHATS	WHAT, WHAT, WHATS, WHAT, WHATS, WHERE, WHERE, WHATS, WHATS, WHATS
WHERE’S	WHERE, WHERE’S, WHERE’S, WHERE’S, WHYS, WHERE’S, WHERE, WHERE, WHATS, WHATS

Table 7.5: Example of a set of confusions that have been identified by the training process. Here, two ground-truth words are confused with different, visually similar words. The frequency counts of these confusions are used to derive the weights for the confusion model (pictured in Figure 7.8).

As we are now working with word-level confusion patterns, the P^* WFST is changed to model the word sequence (not the phoneme sequence) that has been produced by the standard approach triphone system. As in our previous work, we retain the output token likelihoods (in this case, word log likelihoods) and convert them to negative costs for use with the WFST. For this set of initial experiments on word-level confusion modelling, we use the top decoding (i.e. 1-best).

In our new word-level confusion model system, we define a new WFST cascade consisting of three transducers:

$$P^{W^*} \circ C^W \circ M, \quad (7.2)$$

where P^{W*} is the input WFST modelling the word decoding produced by the triphone standard approach recogniser (five examples of P^{W*} WFSTs are shown in Figure 7.9), C^W is the word-level confusion model, and M is a word bigram language model. The sentences in the RM-3000 corpus are divided into ten folds and cross-fold validation is performed with three sets: training (used to train the standard approach recogniser), testing (used to train the word-level confusion model), and validation (used as a test set for the WFST confusion cascade). The M transducer is a bigram word-level language model that has been trained using the held-out 5000 sentences from the RM corpus [Price et al., 1988]. Table 7.6 compares the recognition results produced by the word-level confusion system with the results obtained with the previous standard approach system. Using a combination of triphones and a word confusion matrix increased the accuracy of our system from 12.79% accuracy to 66.34%, which is slightly higher than the standard approach using monophones but not as high as the standard approach using triphones.

System	Word Accuracy (%)
Standard Approach with monophone HMMs and a word bigram language model	66.29
Standard approach with triphone HMMs and a word bigram language model	75.58
WFSTs using triphone HMM standard approach and a word confusion model	66.34

Table 7.6: Comparison between the word recognition accuracies achieved with a monophone and triphone HMM system.

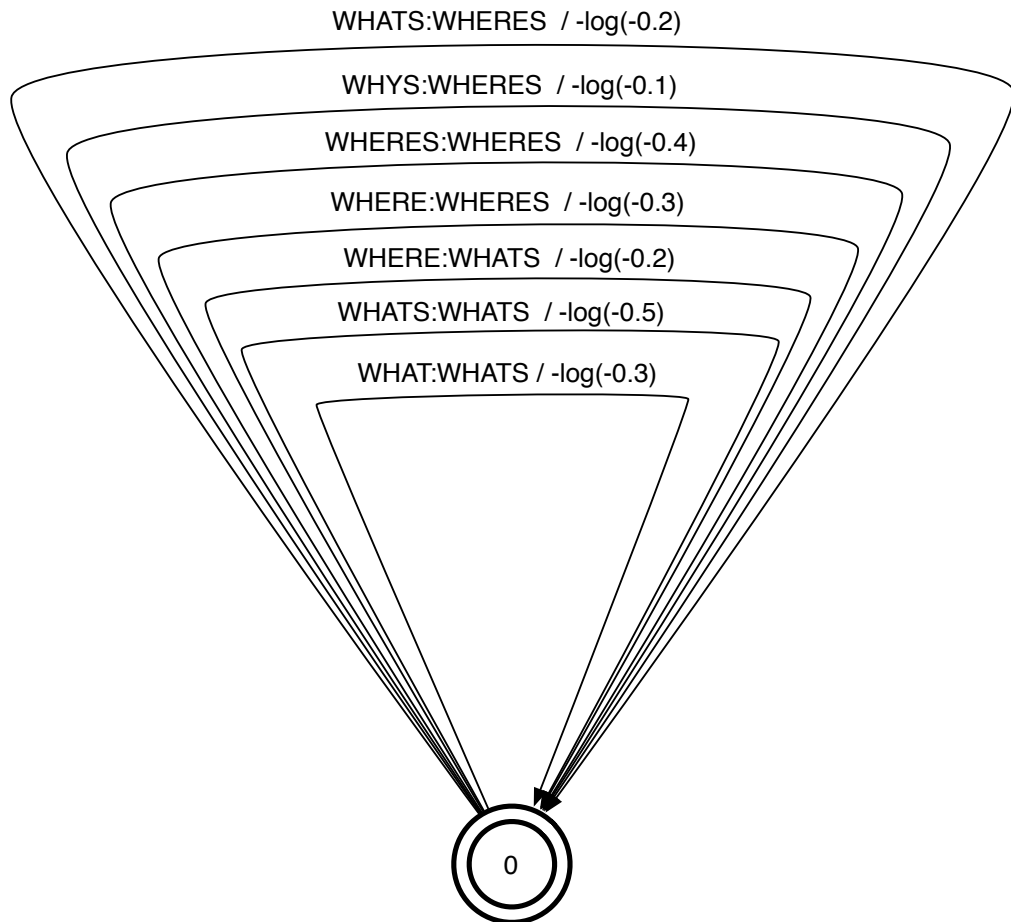


Figure 7.8: A simple word-level confusion model example derived from the confusions that are observed in Table 7.5. Weights are derived from the ratio of the specific confusion (i.e. the translation between the ground-truth and recognised word) to the total number of confusions for the given ground-truth word.

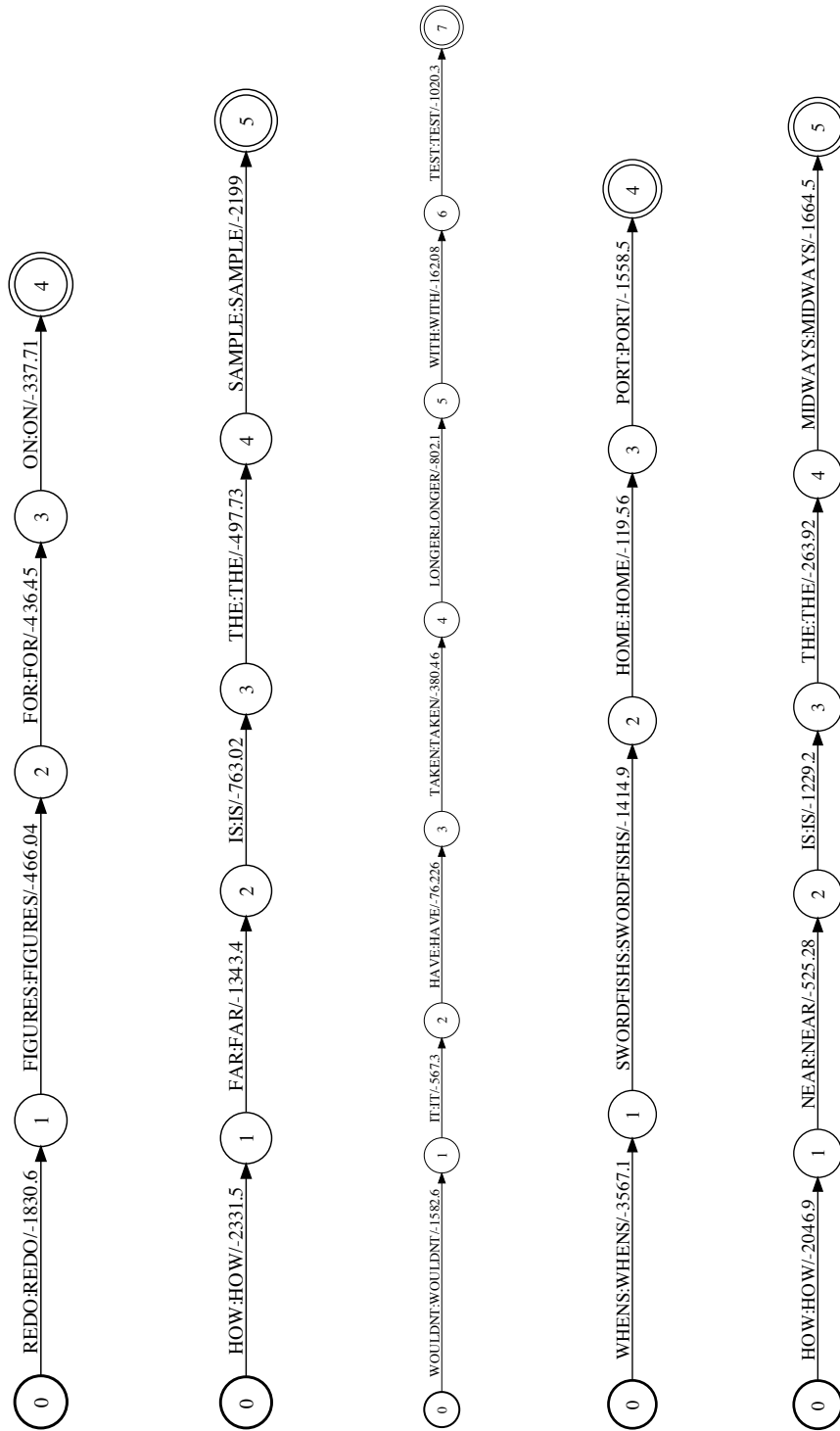


Figure 7.9: Five example P^{W*} WFSTs which are constructed from the decoded word sequences that are produced by the standard triphone HMM recogniser. These WFSTs are used as the entry-point to a WFST cascade using word-level confusion patterns.

7.3.3 The Proposed Approach using Word Hypotheses

The HTK triphone system is an established recognition tool for ASR. However, the built-in recogniser is tailored towards producing word decodings only, which is a possible limitation for our confusion modelling techniques which use the decoded phoneme strings under the influence of a phoneme bigram language model. However, with the attractive gain in word accuracy that can be achieved using the triphone system, we adapt our WFST confusion system to accommodate the word-level recognition output. Figure 7.10 illustrates a new confusion modelling approach to take advantage of the triphone HMM standard approach. Here, an additional word-to-phoneme dictionary is used to provide the translation from a decoded word output to a phoneme string. This phoneme string can then be used with our confusion matrix estimation techniques presented in Chapter 5.

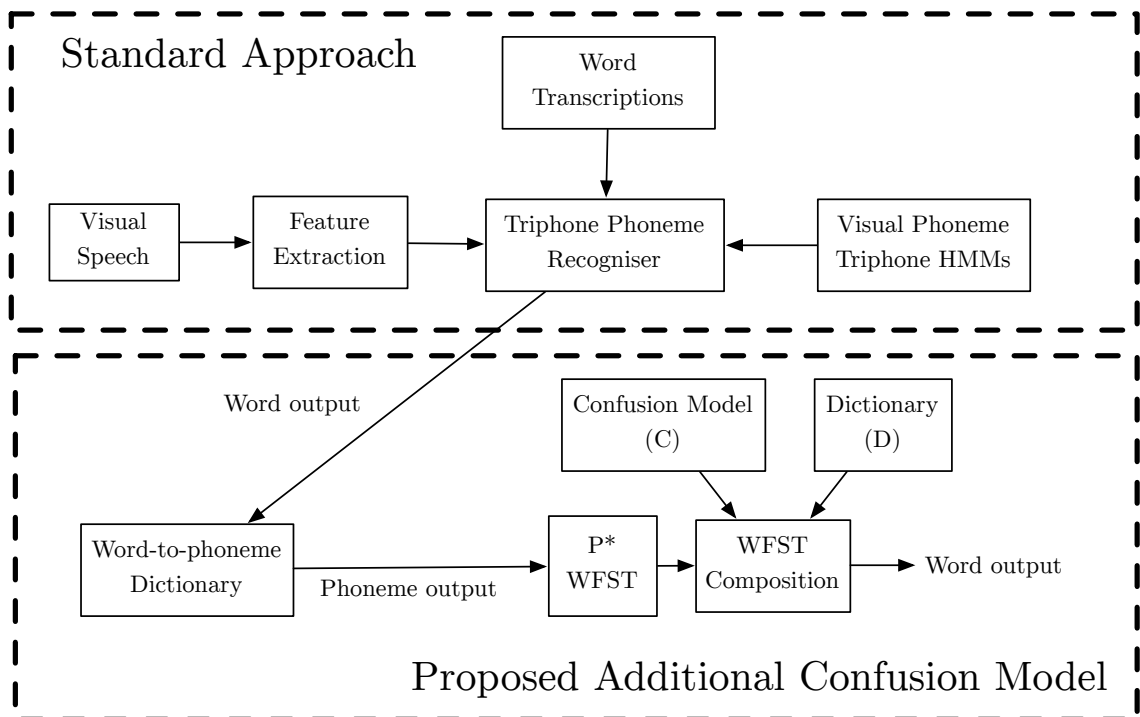


Figure 7.10: Showing how the standard approach triphone system is used with an additional confusion model to correct phoneme strings. The word output is decomposed into a phoneme string (with short pauses at word boundaries) which is then used in the confusion WFST cascade.

The RM-3000 dataset is split into ten folds, each containing 300 sentences. From these folds, three sets are formed: a training set (consisting of eight folds) which is used to train the triphone standard approach recogniser, a testing set (one of the remaining folds) which is used to train the confusion model with decoded strings produced by the standard approach recogniser, and a validation set (the final fold) which is used as test data to the WFST confusion cascade. Cross-fold validation is performed with each validation set used as unseen test data.

Reverting to a phonemic confusion model ensures that our previously developed techniques can be explored using our new triphone decodings. We also restore the use of base smoothing, a confusion matrix smoothing technique that achieved the best results for the work presented in Chapter 5. As we are now dealing with phoneme sequences that have been decomposed from word sequences, the probability mass on the diagonal elements will be much stronger than previous experiments. Therefore, we experiment with varying the parameter which defines the percentage of diagonal probability to re-distribute to the off-diagonal elements and find that the best accuracy is achieved using a distribution parameter of 40%, a much higher percentage than the 10% value that had previously been used in Chapter 5. The built-in word recogniser in HTK also provides an option to produce sub-word (phoneme) timing alignments when using the word-level output. Therefore, although there are limitations for our work when using triphone decodings (using the phoneme string that has been produced by a word decoding), we are able to use these phoneme timing offset information to classify confusions as ‘spurious’ or ‘genuine’ — a technique that significantly improves the recognition accuracy of the confusion model for the isolated word task (described in more detail in Section 5.3.2).

In this set of experiments, we perform an exhaustive search over all possible parameter combinations, adjusting the timing offset window, the number of n -best transcriptions to use (we found that 1-best worked best), and the base smoothing distribution percentage to run over 10000 experiments. Table 7.7 compares the recognition statistics between our system (WFST proposed confusion system), and

	Standard Approach HMM Triphone System	WFST Proposed Confusion System
No. of Correct Words	20,500 (78.50%)	20,477 (78.41%)
No. of Deleted Words	2,308 (8.84%)	2,363 (9.05%)
No. of Substituted Words	3,306 (12.66%)	3,274 (12.54%)
No. of Inserted Words	763 (2.92%)	596 (2.29%)
Total No. of Words in Ground-Truth	26,114	26,114
Word Accuracy (%)	75.58	76.14

Table 7.7: Comparison of the word recognition statistics between the standard approach triphone system and the WFST confusion modelling system using the triphone decodings.

the standard approach (standard approach HMM triphone system). Most importantly, our new approach achieves a 0.56% gain in word accuracy over the standard approach despite correctly recognising fewer words. The number of deletions is also higher in our proposed system. However, our approach produces a word output with 32 fewer substitutions and 167 fewer insertions. The statistics in Table 7.8 show a comparison between the correct and incorrect recognition of all words for the WFST confusion system and the standard HMM system. To construct this table, the decoded word sequences from both approaches were aligned to the ground-truth. For each word in all of the 3000 sentences, the decodings were aligned to one another using DP and classified as one of four possible outcomes: the standard approach and the proposed approach both decode correctly, the standard approach and the proposed approach both decode incorrectly, the standard approach decodes correctly but the proposed approach decodes incorrectly, and the standard approach decodes incorrectly but the proposed approach decodes correctly. In the off-diagonal elements of this contingency table, it is interesting to find that our WFST confusion system recognises 351 words where the standard approach recognises incorrectly, whereas the standard approach correctly only recognises 214 words correctly where the WFST confusion system fails. Furthermore, a McNemar's significance test has concluded that the difference between the two systems is statistically significant with $p < 0.001$ [Gillick and Cox, 1989].

		WFST		Total
		Correct	Incorrect	
HMM	Correct	20411	214	20625
	Incorrect	351	5901	6252
Total		20762	6115	26877

Table 7.8: Decodings are cross-matched to construct this 2x2 contingency table to compare the standard approach (HMM) with the proposed confusion system (WFST). The entries in this table are counts of correctly/incorrectly decoded words.

7.3.4 The Proposed Approach using Lattices

N -best lists provide multiple top hypotheses which can be used to build a richer set of word sequences. Section 5.3.3 introduced an n -dimensional alignment algorithm to build a P^* WFST for any n -best list. This representation can then be used in the WFST confusion cascade to model any combination of the aligned sequences from all of the n transcriptions. The standard approach decoder (built in to HTK) outputs a *recognition lattice* using likelihoods produced by the models (known as the acoustic likelihoods), and the language model. Using this lattice, the system finds the top n transcriptions (i.e. the transcriptions with the highest likelihoods) and produces this as an n -best list. However, our idea was to use the raw recognition network that was produced by the HTK system directly. This section presents a further extension to the WFST confusion cascade by replacing the n -best P^* WFST at the entry-point to the cascade with a recognition lattice which provides richer information for the confusion system. We use **HDecode**, an additional program in the HTK toolkit, to produce recognition lattices in the Standard Lattice Format (SLF) using a standard HMM cross-word triphone system. Given the recognition lattice for a particular test utterance, we present an algorithm to convert a *word-level* recognition lattice (in the SLF format) to a *phoneme-level* WFST.

Although the conversion from a recognition lattice to a WFST might at first be

considered as a trivial task, the two networks use different topology rules that need to be considered and translated. The recognition lattices store symbols in the states and costs on the transitions between the states, whereas the WFST topology only allows for symbols to be stored with the transitions on the arcs. Therefore, in the recognition lattice, the arc weights store the log likelihoods of transitioning from a one symbol to another, whereas, in the WFST, the transition belongs to the arc which stores the symbol translation itself.

For a straight conversion, each state from the recognition lattice is converted to an isolated two-state transducer with a single transition (and zero weight) as shown in Figure 7.11. The start and end state numbers are also stored for each word. A transition in the recognition lattice from one word to another is converted to a WFST arc by adding a new weighted transition between the end state of the WFST which models the first word to the start state of the WFST which models the second word. The weight for this transition is defined over the tropical semiring as the negation of the sum of the total log likelihood from the original recognition lattice using the model likelihood, a , and the language model likelihood, l from the recognition lattice with an insertion penalty, p , and a grammar scale factor, s to follow Equation 2.15. We perform an adaptation of this method to convert the word-level recognition lattice (produced by `HDecode` using the cross-word triphone HMMs) to a phoneme lattice by expanding each state conversion (shown in Figure 7.11) to a sequential phoneme WFST. Figure 7.12 illustrates a very simple example of the conversion procedure from a word recognition lattice to a phoneme WFST. As discussed previously, each state in the lattice has been represented by a sequence of states in the WFST with the corresponding phoneme sequence for a given word (found using a word-to-phoneme dictionary). Determinization and minimization are performed on the final lattice P^* WFST to improve efficiency.

The size of the recognition lattice can be controlled using a pruning parameter in the `HDecode` program. This parameter is used to remove any paths through the recognition lattice where the log likelihood falls under a threshold. If m is the

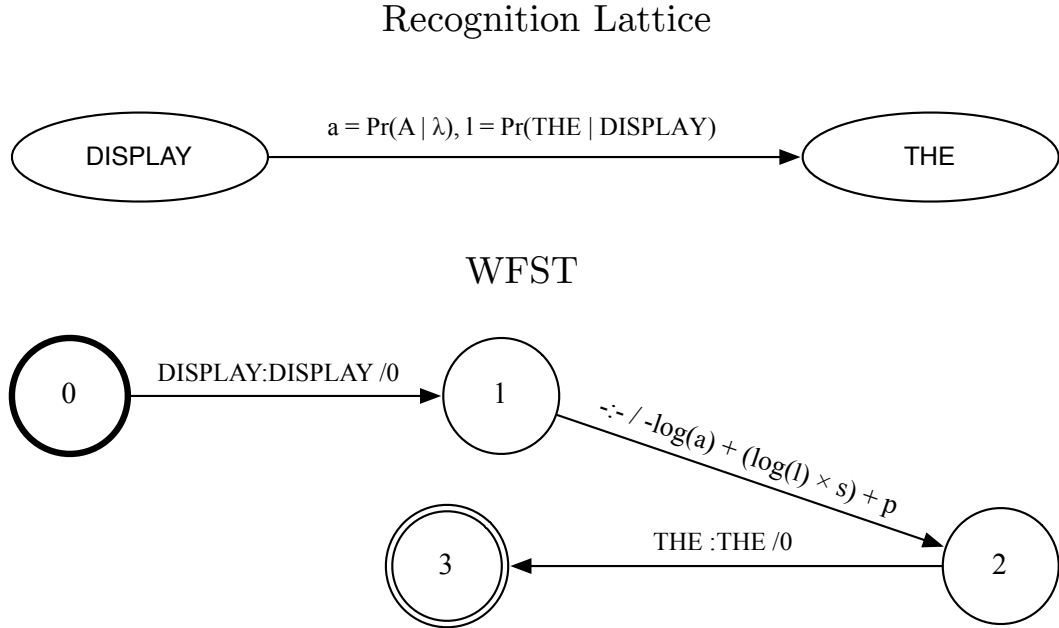


Figure 7.11: An example of a conversion from a simple recognition lattice to a WFST. The difference between the two paradigms lies with the location of the subject symbol. In a recognition lattice, nodes represent the symbols in the network which are joined using transition likelihoods, whereas in the WFST, only arcs store symbols together with weights. To apply this conversion, a single state from the recognition lattice is converted to a mini WFST with two states and a single (unweighted) transition. Weighted transitions are then used to connect these small WFSTs together.

maximum likelihood of all models in the network, and p is the pruning threshold defined at recognition time, all models with a likelihood that is less than $(m - p)$ are removed from the recognition lattice at the pruning stage. We use this parameter to control the size of the recognition lattice and, hence, we also control the size of the equivalent P^* WFST. With an adjustment in the number of states and arcs in the input P^* WFST, the computational time of the cascade is also controlled. We set the pruning threshold to $p = 400$ after experimenting with different possible values to maximise the rich set of candidate word paths whilst ensuring that the network is not large enough to cause memory overflow.

We build cross-word triphone HMMs as described in Section 7.3.1 with the RM-3000 dataset split into training, testing, and validation sets. The phoneme confusion

model is trained using the most likely word decoding from the triphone standard approach transcribed with a word-to-phoneme dictionary. We use base smoothing and perform a search to find the optimal percentage value to re-distribute from the on-diagonal to off-diagonal elements (calculated at 40%) and maintain the use of the timing offset classification method, with the best configuration using a window of ± 2 standard deviations.

Table 7.9 presents a comparison between the recognition statistics for the WFST system with lattices against the n -best system presented in Section 7.3.3. In most cases, the WFST confusion system with lattices is outperformed by the n -best WFST system, with 851 fewer correct words and over 600 more deletions. However, there are over 150 fewer inserted words, a dramatic reduction from the number of insertions in the standard approach triphone system by over 300.

	WFST Proposed Confusion System using n-best	WFST Proposed Confusion System using lattices
No. of Correct Words	20,477 (78.41%)	19,626 (75.16%)
No. of Deleted Words	2,363 (9.05%)	3,003 (11.50%)
No. of Substituted Words	3,274 (12.54%)	3,485 (13.34%)
No. of Inserted Words	596 (2.28%)	442 (1.69%)
Total No. of Words in Ground-Truth	26,114	26,114
Word Accuracy (%)	76.14	73.46

Table 7.9: Comparison of the word recognition statistics between the WFST proposed confusion system presented in Section 7.3.3 and the WFST confusion system using decoding lattices.

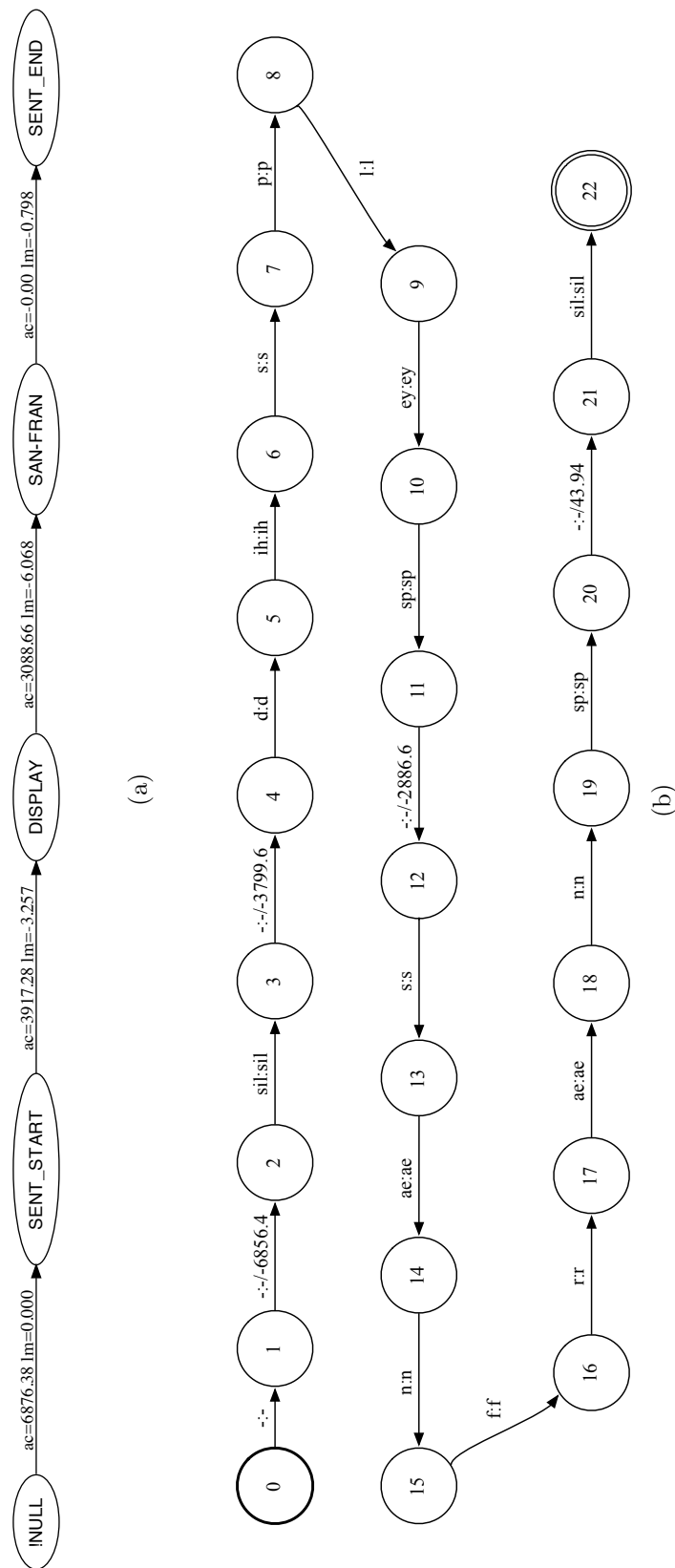


Figure 7.12: (a) is a simple decoding lattice that has been produced by the standard approach recogniser using HTK, and (b) is the equivalent phoneme-level WFST for the lattice shown in (a).

7.4 Conclusions

The work in this chapter has focussed on modelling confusions in continuous speech with the use of a large specially-recorded dataset (RM-3000). Using the standard approach with monophone models, we obtain a 66.29% word accuracy with this data which is a good result for lip-reading. However, this is still inferior to audio ASR performance with a standard ASR system, which achieves over 95% word accuracy (shown in Figure 6.4). The large number of deleted phonemes produced by a lip-reading phoneme recogniser is a concern when considering a confusion modelling system. The work presented in Section 7.2.2 uses the insertion penalty value in the HTK toolkit to define the relative number of insertions and deletions and produce a phoneme decoding that can be modelled using our confusion system. However, the results from this approach were poor with a best word accuracy of 12.79%. This poor result was attributed to the large number of insertions and deletions that were still in the decoded phoneme sequences even after performing a search to find the optimal decoding with fewer deletions (see Figure 7.5).

We used the large amount of training data to extend the standard approach from monophone HMMs to triphone HMMs, increasing the number of models from 44 to 3047. These triphone models are highly relevant for lip-reading because they model co-articulatory effects which are very strong in lip-reading. The new triphone standard approach achieves much improved performance over the monophone approach, reaching 75.58% word accuracy compared to the monophone's best result of 66.29%. Furthermore, there appears to be a stronger structure to the confusion patterns produced by the triphone recogniser, with words becoming phonetically similar. Section 7.3.2 presents work on building a simpler WFST cascade guided by word-level confusion patterns. However, we find that these do not improve the word accuracy from the standard triphone approach.

We then returned to modelling confusions at the phoneme level by taking the decoded word sequence from the standard triphone HMM approach and using a word-to-phoneme dictionary to build a phoneme P^* WFST. By doing this, we also

System	Word Accuracy (%)
Standard Approach with monophone HMMs and a word bigram language model	66.29
Standard approach with triphone HMMs and a word bigram language model	75.58
WFSTs using triphone HMM standard approach and a word confusion model with a weighted P^*	66.34
WFSTs using triphone HMM standard approach and the decoded word output decomposed to a phoneme string	76.14
WFSTs using triphone HMM standard approach and the decoding lattice	73.46

Table 7.10: A summary of all word recognition results for standard approach systems and all WFST confusion systems.

used the techniques explored in the work on the isolated word dataset in Chapter 5 to improve the recognition performance (i.e. the timing offset classification method). Here, we are able to improve word accuracy by 0.56% from 75.58% (using the standard approach with triphones) to 76.14% (using the WFST confusion system). The statistics presented in Table 7.8 show that, although this gain in accuracy is small, there are some words that can be correctly recognised by the WFST confusion system, but, at the same time, are incorrectly recognised by the HMM standard approach. A McNemar’s significance test has shown that our accuracy improvement is statistically significant. Our final work explored the use of recognition lattices in confusion modelling. This represents a richer version of the n -best lists used in previous experiments as it encodes all possible sentence decodings (except for those that are removed by the pruning threshold). We presented an algorithm to convert a word recognition lattice to a phoneme-level WFST for use in our con-

fusion cascade and perform experiments with these new networks. However, these lattices provide no improvement over the standard HMM triphone approach with the number of correct words declining by over 800 but the number of deletion errors rising by over 600 words.

Table 7.10 summarises the results for the HMM standard approach and WFST confusion modelling approaches used in this chapter with the best recognition accuracy (76.14%) being achieved using the WFST confusion modelling technique with the most likely (1-best) output.

Chapter 8

Conclusions

8.1 Discussion

Automated lip-reading is the process of decoding speech using information obtained from the visible articulators only. Recent work in lip-reading has produced recognition accuracies that are significantly inferior to audio ASR. This thesis has presented a new approach to improve the recognition accuracy of an automated lip-reading system by modelling the error patterns that have occurred in decoding. We treat these decoding errors as confusion patterns in the form of substitutions (replacing one sound for another), insertions (adding a sound into the decoded sequence), and deletions (omitting a sound that was in the ground-truth sequence).

This thesis has presented work using a series of weighted finite-state transducers in a composition cascade. We compose four WFSTs consisting of: the phoneme sequence that has been produced by a standard phoneme (HMM) recogniser; the confusion model which provides a translation between phonemes in the form of substitutions, insertions, and deletions; a dictionary which maps the sequences of phonemes to words; and a grammar which models the word-level language structure. In our initial experiments on an isolated word dataset, we estimated the confusion matrix from the symbolic alignment between the recognised and ground-truth

phoneme sequences and achieved a word accuracy of 21.42% using our system. During these experiments, we identified potential spurious confusions that were added to the confusion matrix due to poor alignment. We improved the confusion model by enforcing an offset timing window on all aligned phonemes and achieved a gain in word accuracy of nearly 25%. Recognition accuracy was then improved to 49.70% using the n -best decodings. However, these results were still inferior to the standard HMM recogniser (60%). We extended our proposed approach using two techniques to improve the accuracy of the confusion model: adaptive confusion training, and bigram confusion modelling. The bigram system achieved a better accuracy than our unigram system (by over 3%) but was severely affected by the sparsity of available training data (211 isolated words with six repetitions of each).

As there is only limited audio-visual speech corpora appropriate for lip-reading available in the research community, we recorded a new, much larger dataset consisting of 3000 sentences. We continued to use a single speaker to focus on the modelling of visual speech confusion patterns without introducing variability between speakers. Extending the task from isolated words to continuous speech introduced a large number of insertions and deletions in the HMM phoneme decoding. We explored the use of different insertion penalties to reduce the number of insertions and deletions so that we could use our WFST confusion cascade on the RM-3000 dataset. However, even though we managed to reduce the number of large sequences of insertions and deletions, there were still too many of these confusions for our WFST cascade to correct. We then extended our standard approach recogniser to use triphone HMMs (where each model represents a phoneme with a left and right context) instead of monophone HMMs (where each model represents a single, isolated phoneme). Upon inspection of the word-level confusions that occurred in the triphone decodings, we found that there were many confusions between phonetically similar words. We modified our existing proposed approach (using phoneme strings as input) to use words with a word-level confusion model, and were able to improve recognition over the standard monophone approach (by 0.05%), but not over the triphone standard approach.

When using triphone HMMs, the output from the HMM recogniser (and input to our confusion system) is words, not phonemes. We overcame this by re-writing the decoded words as phoneme strings and modelling the input to the confusion system on these phoneme strings. With this new representation and using n -best decodings, we performed experiments using our confusion system and achieved an accuracy of 76.14%, an improvement over the standard HMM-based triphone system. We also investigated the use of decoding lattices to improve the accuracy of our system further. We developed an algorithm to convert a word decoding lattice (produced by the triphone system) into a phoneme-level WFST to use in our confusion system. This system achieved a word accuracy of 73.46% — nearly 3% lower than our best result using n -best decodings.

This thesis has also addressed two unanswered questions in lip-reading with our new, much larger database (RM-3000): how much training data is required to reach peak recognition accuracy, and is it better to use phoneme or viseme units for lip-reading? It seems that, using this 1000-word vocabulary, automated lip-reading requires around 1500 training sentences to achieve a word accuracy close to its peak, which is much higher than the training data required to reach a peak word accuracy with audio ASR (around 400 sentences). Although it was expected that lip-reading requires more training data than audio ASR, this result shows that most recent lip-reading work, which has used small amounts of data (e.g. only 200 sentences per speaker [Lan et al., 2010]), could be improved by training on a larger dataset. Note however, these figures might vary for different speakers as only a single speaker was considered here.

We continued by evaluating word, phoneme, and viseme results using phoneme and viseme HMMs and found, somewhat counter-intuitively, that it was better to model visual phonemes than visual visemes for word recognition. We investigated this further by exploring partial phoneme-to-viseme mappings and found that word recognition with phoneme models was still better than any of the partial viseme sets that were made. There are two likely reasons for these findings: the phoneme-to-

viseme mapping produces ambiguous unit transcriptions (i.e. homophenous words) that cannot be distinguished from one another, causing some words to be incorrectly recognised. Secondly, the larger dataset (RM-3000) provides many examples of each phoneme in many different contexts. Therefore, phonemes that are mapped to the same visemic class could have different neighbouring sounds (co-articulation) which will lead to inaccurate viseme HMMs being built.

The final word recognition accuracy on our RM-3000 dataset using our confusion system is 76.14% — an accuracy that is the highest seen in lip-reading recognition so far and a statistically significant improvement over the word accuracy obtained by the standard HMM word recogniser ($p < 0.001$). The work presented on improving dysarthric ASR improves recognition accuracy by up to 27% [Morales, 2009]. Although we have not seen the same improvements using similar techniques for lip-reading, we have shown that our confusion system correctly recognises 351 words which were incorrectly recognised by the HMM recogniser (see Table 7.8). The smaller improvements compared to the dysarthric ASR work is likely caused by less predictable confusion patterns, mostly from the effect of coarticulation in visual speech. To obtain more improvements for lip-reading, there are some interesting approaches to pursue for future work such as combining the outputs from the two techniques to provide a more accurate hypothesis.

8.2 Future Work

We have presented new approaches to confusion modelling and explored the application of such systems in automated lip-reading. This section identifies the potential limitations with this work, whilst also exploring areas of further study.

Time and space efficiency are potential issues to address with the WFST composition cascade used in this thesis. The WFST composition algorithm used in the OpenFST toolkit [Allauzen et al., 2007] is dependent on three factors: the number of states, the maximum out-degree of the first WFST (i.e. the number of transitions

exiting a state), and the maximum multiplicity of the second WFST (i.e. the maximum number of times that a symbol is repeated in all transitions exiting the states). In our confusion WFST system, we model all entries in the confusion matrix. Therefore, for any phoneme in the P^* WFST, the out-degree of the confusion WFST is to map the input phoneme to every other possible phoneme plus an insertion and deletion operation. With this, the composed networks become very large, especially for continuous speech and large vocabulary tasks. Future work could improve the time and space efficiency of the composition cascade by reducing the out-degree, multiplicity, and number of states in each WFST from the cascade.

The work described in Section 7.3 uses the word decodings that are produced by the triphone HMM recogniser. The results using our WFST confusion system show a statistically significant improvement over the standard approach. A promising area for future research could focus on combining the decodings from the HMM and WFST techniques using a confidence measure. Such systems have been developed for audio-visual speech recognition with decision fusion (described in Section 3.3) to combine the decodings from the two modalities. If the confidence measure were accurate enough to correctly decide which system was correct for all words that one of the systems decoded incorrectly where the other decoded correctly (i.e. the off-diagonal mass from Table 7.8 is moved to the ‘correct’ entry), the word accuracy could be improved further by around 2% to 3%, making our system accuracy close to 80%.

The work presented in this thesis has focussed on modelling the confusion patterns of a single speaker. An area of future work could investigate speaker-independent or multi-speaker confusion patterns with a view to making a confusion system that models common substitutions, insertions, and deletions amongst speakers. In Chapter 6, we discovered that lip-reading recognition accuracy starts to plateau when the recogniser is trained on about 1600 sentences. To continue our study into confusion modelling, a much larger database would need to be recorded with multiple speakers. This would require a substantial investment in time to record and post-process.

The work presented in Section 5.4.2 uses contextual confusion patterns in the form of phoneme pairs (bigrams). However, the results indicated that the limited amount of data used to train the bigram confusion model is too limited. We recorded a much larger database (RM-3000) and used this to improve recognition using context-dependent models (triphone HMMs), which produced more phonetically-plausible confusions than context-independent models (monophone HMMs). Further work could explore the use of higher order n -gram confusion models with this large dataset. Possible restrictions to the size of these contextual confusion models could be enforced by the size of the composed WFST networks and this may require more computational resources.

Another approach to contextual confusion modelling could reduce the size of the n -gram confusion model whilst maintaining the higher order information. *Dynamically expanding context* was first introduced in [Kohonen, 1986] and applied to correction for speech recognition in [Cox, 2004]. In this work, phoneme translations are modelled with many different context lengths rather than a fixed length model (e.g. a bigram model). In a similar way to Cox’s approach, future work could construct a dynamically expanding contextual confusion model and represent this as a WFST in the confusion cascade system. The large amount of data that is available in the RM-3000 dataset could enable longer, frequently occurring confusion patterns to be more accurately modelled.

Conditional random fields (CRFs) were first introduced in [Lafferty et al., 2001] as a statistical model to segment and label sequential data. Since then, they have been successfully used in many applications including natural language processing, computer vision, and computational biology. In contrast to an HMM, which models the joint probability distribution ($Pr(x, y)$) over the states and observations, CRFs model the conditional probability directly ($Pr(y|x)$). This ensures that dependencies between the input features (x) do not need to be explicitly modelled [Sutton and McCallum, 2006]. One main advantage of CRFs is the ability to model contextual information, something that could be essential for our confusion modelling system.

We have conducted preliminary research into using CRFs for our work and these have produced some promising results.

During the period taken to complete the work described in this thesis, the state-of-the-art in speech recognition has evolved with advances in Deep Neural Networks and additional features increasingly being used in the speech community. Most research in the speech community has switched to using new tools which can integrate the previously well-established techniques (such as HMMs) with the more recent technologies in ASR (DNNs, FSTs). The Kaldi toolkit [Povey et al., 2011] is a new open-source library that has been built on the use of FSTs in speech recognition. Because this software is built on the OpenFST library (which is also used throughout this thesis), there is a great opportunity to integrate our confusion system into the Kaldi composition cascade.

8.3 Publications

The following publication has been produced by the work in this thesis:

Howell, D., Cox, S. and Theobald, B., Confusion Modelling for Automated Lip-Reading using Weighted Finite-State Transducers. In *Proceedings of the International Conference on Auditory-Visual Speech Processing 2013*, pages 197-203, 2013

Appendix A

Isolated Word Vocabulary

A.R	BITTAKER	GESTURER
ABBOTTS	BLACKCAP	HALLIERS
ABEYANCE	BLIPPERS	HASHID
ABJURE	BOAT-TRAIN	HAYHURST
ABNORMAL	BOBBITS	HEATHEN
ABSORBED	BONNEVILLE	HEDGUEST
ACCORDION	BOOK-KEEPER	HODGEPODGE
ACCOUNTS	BOOMTOWN	HOI-POLLOI
ACHE	BOORSTIN	HOOFBAT
ACTUALS	BOOZIEST	HORSEHAIR
ADHESION	BOUGIE	HORSESHOEING
ADJOINING	BOYHOODS	HUFFIER
ADJUDGER	BRIMMAGE	HUTTERER
ADMEASURER	BRUSH-OFF	HUZZAING
ADMITTEE	BUCKLELESS	JACK-KNIFE
ADVERBS	BUDDIES	JAIPUR
ADVOWSON	BURRILLS	JEU
AERIFIED	BUSCARL	JOAN

AEROFOIL	C.I.A	JORNALLING
AESTHESIA	CADDOCKS	JUPE
AFFAIRS	CAPUCHED	LAIRING
AFFIRMEDS	CAREPLUS	LITHIA
AFFLUENCE	CASTELLAW	LOADSTAR
AFORE-THOUGHT	CATCHPOLE	MARE
AGENT	CENSURER	MAUNDERER
AIR-SHAFTS	CERTAINNESS	MIAOUED
AIR-TO-AIR	CHAMBERER	MOOR
ALADDIN	CHANDECK	MOUTHFUL
ALICKS	CHANGCHUN	MOUTHY
ALLPATHS	CHAWER	MOVIEST
ALONENESS	CHEER	NIGHTTIMES
ALONGSHORE	CHEWER	NIRVANA
ALTHOUGH	CHILDCARE	NOISY
ALTHOUSE	CHISELLIKE	NOUVEAU
ALVECHURCH	CHOICER	NUTSHELL
AMONGST	CHOPPIEST	OATHOUT
AMPULLAR	CHOWCHOW	OF
ANAEMIAS	CHUCKHOLE	OSAKAR
ANCHOVY	COHERER	OUTVYING
ANGELOFF	COLLIVER	OVERACHIEVE
APISHLY	CONCURVE	OVERASSURED
ARISTAR	COSMOS	PERSHARE
ARTCARVED	COSSACKED	PINNATION
ARTICHOKE	COURGETTES	PREPPIES
ASH-PANS	COVERERING	PULVAR
ASSOCIATE	CUBBISON	RIPPEON

ATHEISM	CUSHIEST	SAO-PAOLO
ATHLETES	D.J	SEERESS
AUTOPSY	DANNELLEY	SHY
BACKCLOTH	DARENT	SMITHIES
BACKDOORS	DECOYER	SONGED
BACKSHEESH	DEFLOWERER	SOY
BALKIEST	DEUTSCHMARK	SPACE-SUIT
BALSAR	DIANTHA	SUBMERGE
BANKCARD	DOUBTFUL	SUDDENNESS
BARRETTTER	DOURER	TERMI
BATH-CHAIR	EIGHTHED	THEREBY
BATTONS	EMPTYING	THEY
BEACHCOMB	ETHIOPIA	THIGH
BEARDLESS	EYESHADE	THOUING
BEEFIEST	FACE-SAVER	TOOTHACHES
BEEHIVES	FALMOUTH	TOOTHY
BELL-BUOYS	FAR-FETCHED	TORCHLIGHT
BELLINO	FEATHERERS	TOURAGE
BENNINGS	FOLIAGED	TOYING
BIASSING	FOODTOWN	UNBATHED
BIBBINS	FORSYTHS	UNKNOTTED
BIDDANCE	FOULKS	VAVASOUR
BIRDFEED	FULL-LENGTH	VOUCHSAFE
BIRTHDAYS	GEORGESON	WELSH

Bibliography

- Adjoudani, A. and Benoit, C. (1996). On the integration of auditory and visual parameters in an HMM-based ASR. In *Speechreading by humans and machines*, pages 461–471. Springer.
- Allauzen, C., Mohri, M., Ibarra, O., and Ravikumar, B. (2009). N-way composition of weighted finite-state transducers. *International Journal of Foundations of Computer Science*, 20(4):613.
- Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., and Mohri, M. (2007). Openfst: A general and efficient weighted finite-state transducer library. *Implementation and Application of Automata*, pages 11–23.
- Almajai, I. and Milner, B. (2008). Using audio-visual features for robust voice activity detection in clean and noisy speech. In *Proceedings of the EUSIPCO*.
- Almajai, I. and Milner, B. (2009). Enhancing audio speech using visual speech features. In *Tenth Annual Conference of the International Speech Communication Association*.
- Benguerel, A. and Pichora-Fuller, M. (1982). Coarticulation effects in lipreading. *Journal of Speech, Language, and Hearing Research*, 25:600–607.
- Berger, K. (1972). Visemes and homophenous words. *Teacher of the Deaf*, 70(415):396–399.
- Binnie, C. A., Jackson, P. L., and Montgomery, A. A. (1976). Visual intelligibility of consonants: A lipreading screening test with implications for aural rehabilitation. *Journal of Speech and Hearing Disorders*, 41(4):530–539.
- Binnie, C. A., Montgomery, A. A., and Jackson, P. L. (1974). Auditory and visual contributions to the perception of consonants. *Journal of speech, language, and hearing research*, 17(4):619–630.

- Bregler, C., Hild, H., Manke, S., and Waibel, A. (1993). Improving connected letter recognition by lipreading. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 1, pages 557–560. IEEE.
- Cambridge-University (2012). BEEP Dictionary. <http://mi.eng.cam.ac.uk/comp.speech/Section1/Lexical/beep.html>.
- Cappelletta, L. and Harte, N. (2012). Phoneme-to-viseme mapping for visual speech recognition. In *International Conference on Pattern Recognition Applications and Methods*, pages 322–329.
- Cetingul, H., Yemez, Y., Erzin, E., and Tekalp, A. (2006). Discriminative analysis of lip motion features for speaker identification and speech-reading. *Image Processing, IEEE Transactions on*, 15(10):2879–2891.
- Chen, S. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.
- Chen, T. and Rao, R. (1998). Audio-visual integration in multimodal communication. *Proceedings of the IEEE*, 86(5):837–852.
- Cheng, O., Dines, J., and Doss, M. (2007). A generalized dynamic composition algorithm of weighted finite state transducers for large vocabulary speech recognition. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–345. IEEE.
- Cohen, M. and Massaro, D. (1993). Modeling coarticulation in synthetic visual speech. *Models and techniques in computer animation*, 92.
- Cooke, M., Barker, J., Cunningham, S., and Shao, X. (2006). An audio-visual corpus for speech perception and automatic speech recognition. *The Journal of the Acoustical Society of America*, 120(5):2421–2424.
- Cootes, T. (2000). An introduction to active shape models. *Image Processing and Analysis*, pages 223–248.
- Cootes, T., Edwards, G., and Taylor, C. (1998). Active appearance models. *European Conference on Computer Vision 1998*, pages 484–498.
- Cootes, T. and Taylor, C. (2001). Statistical models of appearance for medical image analysis and computer vision. In *Proceedings of the International Society for Optics and Photonics (SPIE) – Medical Imaging*, volume 4322, pages 236–248.

- Cootes, T., Taylor, C., Cooper, D., Graham, J., et al. (1995). Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59.
- Cortes, C., Haffner, P., and Mohri, M. (2004). Rational kernels: Theory and algorithms. *The Journal of Machine Learning Research*, 5:1035–1062.
- Cox, S. (1988). *Hidden Markov models for automatic speech recognition: theory and application*. Royal Signals & Radar Establishment.
- Cox, S. (2004). Using context to correct phone recognition errors. In *Proceedings of the International Conference on Spoken Language Processing*.
- Cox, S. (2008). On estimation of a speaker’s confusion matrix from sparse data. In *Ninth Annual Conference of the International Speech Communication Association*.
- Cox, S., Harvey, R., Lan, Y., Newman, J., and Theobald, B. (2008). The challenge of multispeaker lip-reading. In *International Conference on Auditory-Visual Speech Processing*, pages 179–184.
- Cruttenden, A. (2008). *Gimson’s Pronunciation of English*. Routledge, 7th edition.
- Dupont, S. and Luettin, J. (2000). Audio-visual speech modeling for continuous speech recognition. *Multimedia, IEEE Transactions on*, 2(3):141–151.
- Fisher, C. (1968). Confusions among visually perceived consonants. *Journal of Speech and Hearing Research*, 11(4):796 – 804.
- Fowler, C. (1984). Segmentation of coarticulated speech in perception. *Attention, Perception, & Psychophysics*, 36(4):359–368.
- Gillick, L. and Cox, S. J. (1989). Some statistical issues in the comparison of speech recognition algorithms. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 532–535. IEEE.
- Goodman, J. (2001). A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434.
- Green, P., Carmichael, J., Hatzis, A., Enderby, P., Hawley, M., and Parker, M. (2003). Automatic speech recognition with sparse training data for dysarthric speakers. In *Eighth European Conference on Speech Communication and Technology*.

- Hansen, P. and Coleman, M. (2005). Speechreading skill and visual movement sensitivity are related in deaf speechreaders. *Perception*, 34:205–216.
- Hennecke, M. E., Stork, D. G., and Prasad, K. V. (1996). Visionary speech: Looking ahead to practical speechreading systems. In *Speechreading by Humans and Machines*, pages 331–349. Springer.
- Hilder, S., Harvey, R., and Theobald, B. (2009). Comparison of human and machine-based lip-reading. In *In the Proceedings of the International Conference on Auditory-Visual Speech Processing (AVSP)*, pages 86–89.
- Hilder, S., Theobald, B., and Harvey, R. (2010). In pursuit of visemes. In *Auditory-Visual Speech Processing 2010*.
- Hong, X., Yao, H., Wan, Y., and Chen, R. (2006). A pca based visual dct feature extraction method for lip-reading. In *Intelligent Information Hiding and Multimedia Signal Processing, 2006. IHH-MSP'06. International Conference on*, pages 321–326. IEEE.
- Hori, T., Hetherington, I. L., Hazen, T. J., and Glass, J. R. (2007). Open-vocabulary spoken utterance retrieval using confusion networks. In *ICASSP (4)*, pages 73–76.
- Hori, T., Hori, C., and Minami, Y. (2004). Fast on-the-fly composition for weighted finite-state transducers in 1.8 million-word vocabulary continuous speech recognition. In *Eighth International Conference on Spoken Language Processing*.
- Jackson, P. (1988). The theoretical minimal unit for visual speech perception: Visemes and coarticulation. *The Volta Review*.
- Karanasou, P., Burget, L., Vergyri, D., Akbacak, M., and Mandal, A. (2012). Discriminatively trained phoneme confusion model for keyword spotting. In *INTER-SPEECH*.
- Karttunen, L. (2001). Applications of finite-state transducers in natural language processing. *Implementation and application of automata*, pages 34–46.
- Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 35(3):400–401.
- Kleene, S. C. (1956). Representation of events in nerve nets and finite automata. *Automata Studies*.

- Kohonen, T. (1986). Dynamically expanding context, with application to the correction of symbol strings in the recognition of continuous speech. pages 1148–1151.
- Ladefoged, P. and Johnson, K. (2014). *A course in phonetics*. Cengage learning.
- Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labelling sequence data. pages 282–289.
- Lan, Y., Harvey, R., Theobald, B., Ong, E., and Bowden, R. (2009). Comparing visual features for lipreading. In *Procs. of Int. Conf. Auditory-visual Speech Processing. Norwich, UK*.
- Lan, Y., Theobald, B., Harvey, R., Ong, E., and Bowden, R. (2010). Improving visual features for lip-reading. In *Auditory-Visual Speech Processing 2010*.
- Lee, K.-F., Hon, H.-W., and Reddy, R. (1990). An overview of the sphinx speech recognition system. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(1):35–45.
- Liang, L., Liu, X., Zhao, Y., Pi, X., and Nefian, A. V. (2002). Speaker independent audio-visual continuous speech recognition. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, volume 2, pages 25–28. IEEE.
- Luetttin, J. and Thacker, N. (1997). Speechreading using probabilistic models. *Computer Vision and Image Understanding*, 65(2):163–178.
- Matthews, I., Cootes, T., Bangham, J., Cox, S., and Harvey, R. (2002). Extraction of visual features for lipreading. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):198–213.
- Matthews, I., Ishikawa, T., and Baker, S. (2004). The template update problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):810–815.
- Mohri, M. (1996). On some applications of finite-state automata theory to natural language processing. *Natural Language Engineering*, 2(1):61–80.
- Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational linguistics*, 23(2):269–311.
- Mohri, M. (2002). Semiring frameworks and algorithms for shortest-distance problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350.

- Mohri, M. (2004). Weighted finite-state transducer algorithms. an overview. *STUDIES IN FUZZINESS AND SOFT COMPUTING*, 148:551–564.
- Mohri, M., Pereira, F., and Riley, M. (2000). The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231(1):17–32.
- Mohri, M., Pereira, F., and Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88.
- Mohri, M., Pereira, F., and Riley, M. (2005). Weighted automata in text and speech processing. *Arxiv preprint cs/0503077*.
- Mohri, M. and Riley, M. (1997). Weighted determinization and minimization for large vocabulary speech recognition. In *Eurospeech*.
- Morales, S. and Cox, S. (2007). Modelling confusion matrices to improve speech recognition accuracy, with an application to dysarthric speech. In *Eighth Annual Conference of the International Speech Communication Association*.
- Morales, S. O. C. (2009). *Error Modelling Techniques to Improve Automatic Recognition of Dysarthric Speech*. PhD thesis, School of Computing Sciences, University of East Anglia.
- Morales, S. O. C. and Cox, S. (2008). Application of weighted finite-state transducers to improve recognition accuracy for dysarthric speech. In *Ninth Annual Conference of the International Speech Communication Association*.
- Neti, C., Potamianos, G., Luetttin, J., Matthews, I., Glotin, H., Vergyri, D., Sison, J., Mashari, A., and Zhou, J. (2000). Audio-visual speech recognition. In *Final Workshop 2000 Report*, volume 764.
- Newman, J. (2011). *Language Identification using Visual Features*. PhD thesis, School of Computing Sciences, University of East Anglia.
- Newman, J., Theobald, B., and Cox, S. (2010). Limitations of visual speech recognition. In *Auditory-Visual Speech Processing 2010*.
- Ney, H., Essen, U., and Kneser, R. (1994). On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, 8(1):1–38.
- Oonishi, T., Dixon, P., Iwano, K., and Furui, S. (2008). Implementation and evaluation of fast on-the-fly wfst composition algorithms. In *Ninth Annual Conference of the International Speech Communication Association*.

- Pereira, F. and Riley, M. (1997). 15 speech recognition by composition of weighted finite automata. *Finite-state language processing*, page 431.
- Petajan, E. (1984). *Automatic lipreading to enhance speech recognition*. PhD thesis, University of Illinois.
- Petajan, E., Bischoff, B., Bodoff, D., and Brooke, N. (1988). An improved automatic lipreading system to enhance speech recognition. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 19–25. ACM.
- Potamianos, G. and Graf, H. P. (1998). Discriminative training of hmm stream exponents for audio-visual speech recognition. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 6, pages 3733–3736. IEEE.
- Potamianos, G., Neti, C., and Deligne, S. (2003a). Joint audio-visual speech processing for recognition and enhancement. In *AVSP 2003-International Conference on Audio-Visual Speech Processing*.
- Potamianos, G., Neti, C., Gravier, G., Garg, A., and Senior, A. (2003b). Recent advances in the automatic recognition of audiovisual speech. *Proceedings of the IEEE*, 91(9):1306–1326.
- Potamianos, G., Neti, C., Iyengar, G., and Helmuth, E. (2001). Large-vocabulary audio-visual speech recognition by machines and humans. In *Seventh European Conference on Speech Communication and Technology*.
- Potamianos, G., Neti, C., Luettin, J., and Matthews, I. (2004). Audio-visual automatic speech recognition: An overview. *Issues in Visual and Audio-Visual Speech Processing*, pages 356–396.
- Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. (2011). The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.
- Price, P., Fisher, W. M., Bernstein, J., and Pallett, D. S. (1988). The darpa 1000-word resource management database for continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 651–654. IEEE.

- Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.
- Roark, B., Sproat, R., Allauzen, C., Riley, M., Sorensen, J., and Tai, T. (2012). The opengrm open-source finite-state grammar software libraries. *Association for Computational Linguistics*.
- Roche, E. and Schabes, Y. (1997). *Finite-state language processing*. MIT press.
- Stork, D. G. and Hennecke, M. E. (1996). *Speechreading by humans and machines: models, systems, and applications*, volume 150. Springer.
- Sujatha, B. and Santhanam, T. (2010). A novel approach integrating geometric and gabor wavelet approaches to improvise visual lip-reading. *Int. J. Soft Comput*, 5:13–18.
- Sutton, C. and McCallum, A. (2006). An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 93–128.
- Tatham, M. and Morton, K. (2011). *A Guide to Speech Production and Perception*. Edinburgh University Press.
- Teissier, P., Robert-Ribes, J., Schwartz, J.-L., and Guérin-Dugué, A. (1999). Comparing models for audiovisual fusion in a noisy-vowel recognition task. *Speech and Audio Processing, IEEE Transactions on*, 7(6):629–642.
- Theobald, B., Harvey, R., Cox, S., Owen, G., and Lewis, C. (2006). Lip-reading enhancement for law enforcement. In *SPIE conference on Optics and Photonics for Counterterrorism and Crime Fighting*, volume 6402, page 640.
- Theobald, B.-J. (2003). *Visual speech synthesis using shape and appearance models*. PhD thesis, University of East Anglia.
- Tomlinson, M., Russell, M., and Brooke, N. (1996). Integrating audio and visual information to provide highly robust speech recognition. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 2, pages 821–824. IEEE.
- Visser, M., Poel, M., and Nijholt, A. (1999). Classifying visemes for automatic lipreading. In *Text, Speech and Dialogue*, pages 843–843. Springer.

- Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269.
- Woodward, M. F. and Barber, C. G. (1960). Phoneme perception in lipreading. *Journal of Speech, Language, and Hearing Research*, 3(3):212–222.
- Young, S. (2001). Statistical modelling in continuous speech recognition (csr). In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 562–571. Morgan Kaufmann Publishers Inc.
- Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moor, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., and Woodland, P. (2006). *The HTK Book Version 3.4*. Cambridge University Press.
- Young, S. J. (1992). The general use of tying in phoneme-based hmm speech recognisers. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 569–572. IEEE.
- Young, S. J., Odell, J., and Woodland, P. C. (1994). Tree-based state tying for high accuracy acoustic modelling. In *Proceedings of the workshop on Human Language Technology*, pages 307–312. Association for Computational Linguistics.
- Young, S. J. and Woodland, P. C. (1993). The use of state tying in continuous speech recognition. In *Eurospeech*.
- Zhi, Q., Cheok, A., Sengupta, K., Jian, Z., Chung, K., et al. (2004). Analysis of lip geometric features for audio-visual speech recognition. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 34(4):564–570.