Submitted in partial requirement for the
MPhil in Computer Speech and Language Processing

Speaker Tracking

Sue Johnson
Jesus College

27 August 1997

Department of Engineering

University of Cambridge

Abstract

This project investigates the problem of labelling segments in a speaker-tracking system. A mathematical representation of each segment is sought which encaptures the speaker-dependent information available. It is shown that both the covariance matrix and the Maximum Likelihood Linear Regression (MLLR) matrix provide such a representation with over 90% success rate in a speaker identification task.

Several alternative distance metrics to measure the "closeness" of the segments are investigated and it is found the covariance data performs best on those based on the mathematical means of the eigenvalues of one matrix relative to another. The MLLR matrix is by contrast found to work best with elementwise metrics confirming the hypothesis that the individual elements of the matrix are more significant in this case.

Several hierarchical clustering schemes are then investigated and shown to produce speaker-specific groups on two and three speaker problems. A full-scale implementation is then described and tested on data from the 1996 Broadcast News database. A new criterion for evaluating the clusters is defined and shown to be a good indication of speaker split. Six clustering schemes are evaluated using this new criterion and the discriminative Lance-Williams scheme is found to perform the best. Furthest neighbour clustering is also shown to perform well in some cases.

Tree diagrams for the best cases of three and four clusters are presented and explained in terms of the clustering strategy which produced them. They illustrate the feasibility of such a system, with only a few segments being obviously mis-classified. Possible improvements for the system are then discussed and finally recommendations for further work are given.

Declaration

This thesis is substantially my own work. Where reference has been made to other research this is acknowledged in the text and bibliography. It has not been submitted in whole or part for a degree at any other university.

The length of this thesis including footnotes, appendices and bibliography is 14943 words.

Signed:

Acknowledgements

I would like to take this opportunity to thank the many people in the Cambridge University Speech, Vision and Robotics group for their help, advice and sharing of technical expertise during the course of this project. In particular Dr John Openshaw, Antranig Basman, Dr Mark Gales and Gavin Smith, who provided help and inspiration when it was most needed.

Thanks also go to all the members of the MPhil course and teaching staff who made the year fun as well as educational and I am grateful to EPSRC for funding my place on the course through an Advanced Studentship.

I must also mention my family, Dale, Rachel, Steve, Phil, Clare, Neil and Mavis whos constant support and friendship have brightened up many days throughout the year.

My final note of thanks must go to my supervisor, Phil Woodland, for all he has done for me throughout the year. His dedication and enthusiasm for his work have inspired me, his ideas have motivated me, his efforts with HTK have helped me, and his jokes have amused me on many occasions. I look forward to working with him again over the next few years.

Contents

1 Introduction

1.1 What is Speaker Tracking?

Speaker tracking is the process of following who says what in an audio stream. It has many applications ranging from identifying speakers specifically, for example in forensic evidence, to pooling data from the same speaker to increase the performance of speaker-adaptive recognition systems.

Tracking can broadly be divided into two problems:

- Locating the points of speaker change *(Segmentation)*
- Identifying the speaker in each segment *(Labelling)*

Segmentation can be thought of as labelling on a very fine scale. For example consider the case of having two distinct segments. Suppose you can accurately determine whether they originate from the same speaker or different speakers. This means the labelling problem has been solved. Coarse segmentation can be achieved by regularly generating segments throughout the audio and then joining together the adjacent segments which originate from the same speaker.

This has a large disadvantage in that it only allows a coarse resolution in the time domain. Suppose it takes 500 frames (5 seconds worth of speech) to produce the information for a segment which allows it to be identified. Then the point of speaker change will be uncertain to within roughly $\pm$ 2.5s. Worse than that if the speaker changes more than once during that time this information will be lost, and the interjecting speaker may not be identified at all.

This problem can be overcome to some extent by using a sliding window across the audio stream. This allows a finer resolution in the time domain, as the frequency of possible speaker boundaries is now determined by the window rate and not the minimum segment size. This is illustrated in figure 1.

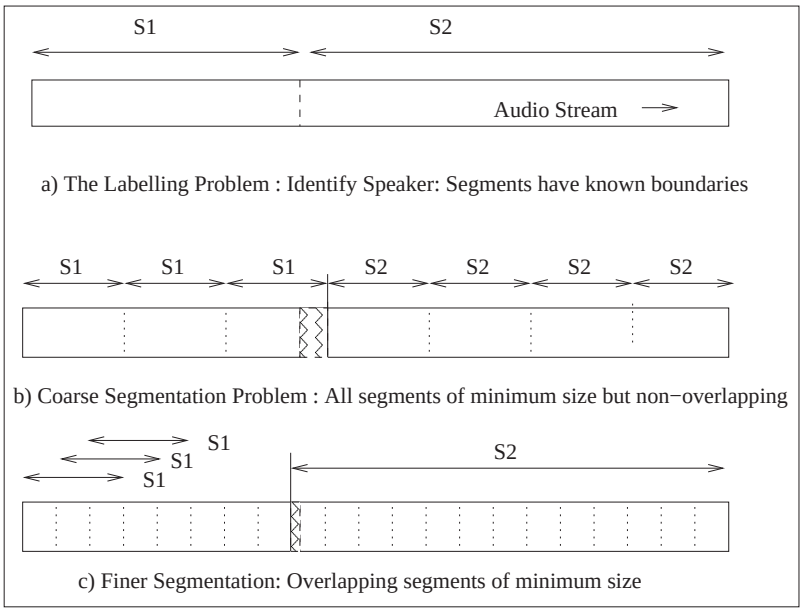

Figure 1: Segmentation and Labelling

The maximum window rate is determined by the required time resolution, whilst the minimum window rate is determined by how easy it is to detect a speaker change. Since much of the data between adjacent windows will be shared, in order to determine a possible speaker boundary the differences between the speakers must be sufficiently high to overcome the smoothing effect

of using the overlapping window. This means that a very good labelling scheme must be available in order to successfully track speakers in an audio stream, but given this labelling scheme, the segmentation problem should be solvable simply by redefining the segments as described above. This project is concerned with developing a labelling scheme to identify speakers on a pre-segmented audio track. Further work on a segmentation system involving the ideas given above can be found in [16].

The labelling problem reduces to finding a representation of each segment which captures the information about the speaker, whilst, if possible, minimising the *intra-speaker* variation. These representations must then be compared to each other to ascertain which ones are most similar and hence determine which speakers uttered which segments.

Finding such a representation is a difficult problem. For example, if the speech is coded in PLP parameters, taking the mean vector over a small segment may retain some speaker-specific information (such as gender), but it will also be highly dependent on which phoneme was being uttered at that time. One method of reducing intra-speaker variation, which has already been used in problems similar to this one [1, 21] is using the covariance of the data over a reasonably sized segment (e.g. ≥ 5 seconds of speech). This method is *text-independent* i.e. it does not require a transcription of what was said, but instead effectively averages out the phoneme variation over the segment.

Potentially, if a putative transcription of the soundtrack exists, it should be possible to exploit this information to improve performance. This project also looks at using the *Maximum Likelihood Linear Regression* (MLLR) transform of each segment as a possible representation which takes the transcription into account, and compares its performance to that of the standard covariance case.

Performance is evaluated both for the case of speaker identification and speaker clustering. The former defines certain segments as references and the others as tests. Each test segment is allocated to the "closest" reference segment according to the *distance metric* being used. Performance is then stated simply as the ratio of the number of speakers which were correctly identified to the number which could have been correctly identified. [1] Sections 2 and 3 look into this problem for the covariance and MLLR representations respectively.

Speaker clustering is concerned more with the improvement of speaker-adaptive recognition systems. Some speakers in the soundtrack may only utter a few short segments of speech. In this case there may be insufficient adaptation data available to improve the recognition performance. However, if the identity of the speaker is not critical, it may be possible to pool data for "similar" speakers to produce enough adaptation data to improve performance for both speakers.

Segments are clustered into groups which are in some sense more similar to members of their own group than those of the other groups. The ideal case would be if every cluster represented a different speaker, but this is obviously dependent on the number of final clusters and the number of speakers in the soundtrack (which is not necessarily known in advance).

Since infrequent speakers may be clustered together without necessarily losing out in recognition performance, a different method of evaluation is needed for this task. Section 4 looks into methods of clustering the data whilst section 5 defines a possible performance measure in terms of a likelihood function of an MLLR-based speaker adaptive system. The final implementation of the system and results are presented in sections 5 with possible improvements being given in section 6. Conclusions are then offered in section 7 along with recommendations for further work on this topic.

[1] i.e. the speaker of the test segment must also be the speaker for at least one reference segment

1.2 Data

The data used throughout this project was from the 1996 Broadcast News database. The speech had been split into acoustically homogeneous segments which had a single speaker and audio category as defined in table 1. [23] The segments of over 5 seconds duration from the shows `a960521` `a960522` `a960528` `a960604` `a960610` `a960621` `a960624` `a960625` and `a960626` were extracted giving 625 segments in total. [2]

Each segment was parameterised using a modified form of PLP [11] based on Mel-Frequency Cepstral Coefficient filter-bank analysis. [23] The zeroth to twelfth static coefficients were used for the representation. Later experiments also included delta and acceleration coefficients.

Category	Description	Dialect	Mode	Fidelity	Background
F0	Baseline Broadcast	Native	Planned	High	Clean
F1	Spontaneous Speech	Native	Spontaneous	High	Clean
F2	Telephone Channel	Native	Any	Medium/Low	Clean
F3	Background Music	Native	Any	High	Music
F4	Degraded Acoustics	Native	Any	High	Speech/Other
F5	Non-Native Speakers	Non-Native	Planned	High	Clean
FX	All Other Combinations				

Table 1: Broadcast News 1996 Categories

[2] A more detailed description of the segments used can be found in appendix H

2 Covariance-Based Methods

2.1 Introduction

Recent work in speaker identification has demonstrated the ability of the covariance matrix to encapsulate speaker information. [1, 3, 21] Given a sufficiently large segment, [3] the effects of the variation due to different phonemes being uttered is averaged out and the information is a truer representation of the speakers characteristics.

Given a feature (row) vector X_t at time t of dimension D, the covariance [4] matrix for a segment of N frames is given by

$$\begin{aligned} \mu &= \frac{1}{N}\sum_{t=1}^{t=N} X_t \\ \Sigma &= \frac{1}{N}\sum_{t=1}^{t=N}(X_t-\mu)(X_t-\mu)^T \end{aligned}$$

2.2 Distance Measures

Once the covariance has been computed for each segment, some measure of distance must be used to calculate the closeness of the segments in D-dimensional space.

Given the covariance of a reference segment, **X**, and a test segment, **Y**, a further matrix, YX^{-1}, can be formed which attempts to capture in some sense the similarity between **Y** and **X**. For example, if **X** and **Y** are identical in that they come from the same segment, then YX^{-1} will be the identity. The proximity measure between two segments however must be a scalar, so a more formal way of capturing the similarity of the two matrices is needed.

A family of measures can be obtained based on using the *eigenvalues* of YX^{-1} (termed the eigenvalues of **Y** relative to **X**.) Let these be denoted λ_i. The converse eigenvectors of **X** relative to **Y** can be seen simply to be $1/\lambda_i$. The closer these eigenvalues are to unity, the better the match of the segments. The arithmetic, geometric and harmonic means of these eigenvalues can then be calculated using the formulae

$$\begin{aligned} A(\lambda_1,...\lambda_D) &= \frac{1}{D}\sum_{i=1}^{D}\lambda_i \\ G(\lambda_1,...\lambda_D) &= \sqrt[D]{\prod_{i=1}^{D}\lambda_i} \\ H(\lambda_1,...\lambda_D) &= D\left(\sum_{i=1}^{D}\frac{1}{\lambda_i}\right)^{-1} \end{aligned}$$

These values are particularly appealing because they do not require the calculation of the eigenvalues explicitly and thus are significantly more computationally efficient than distance metrics using the eigenvalues individually. To see this, note that

$$\begin{aligned} A &= D^{-1}tr(YX^{-1}) \\ G &= (det(YX^{-1}))^{1/D} \\ H &= D(tr(XY^{-1}))^{-1} \end{aligned}$$

where *det* represents the determinant and *tr* the trace of the matrix.

[3] segments ≥ 5 seconds long are used throughout this project

[4] Alternatively the *correlation* can be used: $R = \frac{1}{N}\sum_{t=1}^{t=N} X_t X_t^T$

Five distance measures using these eigenvalues have been investigated by Gish [8], Bimbot and Mathan [1, 2] and Smith [21]. These are: [5]

$$\begin{aligned} d1(X,Y) &= \log\left(\frac{A}{H}\right) \\ d2(X,Y) &= \log\left(\frac{A}{G}\right) \\ d3(X,Y) &= A - \log(G) - 1 \\ d4(X,Y) &= \frac{1}{D}\sum_{i=1}^{D}|\lambda_i - 1| \\ d5(X,Y) &= \frac{1}{D}\sum_{i=1}^{D}|\min(\lambda_i, \frac{1}{\lambda_i}) - 1| \end{aligned}$$

Gish has shown that d3 is equivalent to the classical log likelihood ratio [8] and he notes that robustness can be gained by restricting the eigenvalues in the summation to include only those near unity, hence adapting d4 to give d5.

In the case where X=Y, then YX^{-1} is the identity and therefore all $\lambda_i = 1$. Thus A=G=H=1 and all the distance metrics given above can be seen to be equal to zero. This is an important property of distance measures. Other properties which distance measures should exhibit if possible are

- Positivity : the distance must never be negative i.e. d(X,Y) ≥ 0
- Symmetry : the distance should not be dependent on the order in which the segments are presented, i.e. d(X,Y)=d(Y,X)
- Triangle Equality : d(X,Z) $\leq$ d(X,Y)+d(Y,Z)
  This is the least important of the criteria

Since all the eigenvalues of XY^{-1} are positive, then

$$A \geq G \geq H$$

with the equality in the case when are eigenvalues are equal, thus the distance metrics given above all conform to the positivity rule.

By noting that

$$\begin{aligned} A(\frac{1}{\lambda_1}, ..., \frac{1}{\lambda_D}) &= \frac{1}{H(\lambda_1 ... \lambda_D)} \\ G(\frac{1}{\lambda_1}, ..., \frac{1}{\lambda_D}) &= \frac{1}{G(\lambda_1 ... \lambda_D)} \\ H(\frac{1}{\lambda_1}, ..., \frac{1}{\lambda_D}) &= \frac{1}{A(\lambda_1 ... \lambda_D)} \end{aligned}$$

it can be seen that

$$\begin{aligned} d1(Y,X) &= \log\left(\frac{A(\frac{1}{\lambda_i})}{H(\frac{1}{\lambda_i})}\right) = \log\left(\frac{A(\lambda_i)}{H(\lambda_i)}\right) = d1(X,Y) \\ d2(Y,X) &= \log\left(\frac{A(\frac{1}{\lambda_i})}{G(\frac{1}{\lambda_i})}\right) = \log\left(\frac{G(\lambda_i)}{H(\lambda_i)}\right) \neq d2(X,Y) \\ d3(Y,X) &= A\left(\frac{1}{\lambda_i}\right) - \log\left(G\left(\frac{1}{\lambda_i}\right)\right) - 1 = H(\lambda_i) + \log(G(\lambda_i)) - 1 \neq d3(X,Y) \\ d4(Y,X) &= \frac{1}{D}\sum_{i=1}^{D}|\frac{1}{\lambda_i} - 1| \neq d4(X,Y) \\ d5(Y,X) &= \frac{1}{D}\sum_{i=1}^{D}|\min(\frac{1}{\lambda_i}, \lambda_i) - 1| = d5(X,Y) \end{aligned}$$

[5] all the distance metrics described in this thesis are reproduced in appendix A for the readers convenience

2.3 Initial Experiments

This series of experiments used segments of ≥ 5 seconds duration from the shows : `a960521 a960522 a960528 a960604 a960610 a960621 a960624 a960625 a960626` in the Broadcast News 1996 database. Every third segment in each show was marked as a test segment whilst the others were assigned to be reference segments. The covariance of each segment was calculated using `HCompV` in `HTK` and a MATLAB program was written to assign each test segment to the closest reference segment for each distance metric.

A PERL program then counted the proportion of test segments (for which at least one reference segment existed with the same speaker) which had been identified. In all there were 205 test segments of which 176 had at least one instance of the same speaker in the reference set, 203 had a reference condition and 160 had a match for both condition and speaker.

In addition to the distance measures given above, I added five more of my own which obeyed the zero, symmetry and positivity constraints, namely:

$$
\begin{aligned}
d6 &= \max_{i=1:D} |\max(\lambda_i, \frac{1}{\lambda_i}) - 1| \\
d7 &= \min_{i=1:D} |\min(\lambda_i, \frac{1}{\lambda_i}) - 1| \\
d8 &= \frac{1}{D}\sum_{i=1}^{D}(\lambda_i - \frac{1}{\lambda_i})^2 \\
d9 &= \left(\prod_{i=1}^{D} |\max(\lambda_i, \frac{1}{\lambda_i}) - 1|\right)^{1/D} \\
d10 &= \left(\prod_{i=1}^{D} |\min(\lambda_i, \frac{1}{\lambda_i}) - 1|\right)^{1/D}
\end{aligned}
$$

The results are given in table 2 and illustrated in figure 2

ALL FILES; 420 references; 205 testers (160 seen exactly;176 speakers seen;203 conditions seen)

DISTANCE	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
Correct Speaker	159	159	156	143	144	147	63	142	137	133
Correct Conditions	134	137	129	110	116	124	78	119	118	121
Both Correct	108	108	103	94	99	106	36	100	98	97
Speaker Match(%)	90.34	90.34	88.64	81.25	81.82	83.52	35.80	80.68	77.84	75.57
Condition Match(%)	66.01	67.49	63.55	54.19	57.14	61.08	38.42	58.62	58.13	59.61
Total Match(%)	67.50	67.50	64.38	58.75	61.88	66.25	22.50	62.50	61.25	60.62

Table 2: Results using the Covariance Data on d1 to d10

The behaviour of A G and H themselves was also investigated as d11, d12 and d13 respectively.

DISTANCE	d11=A	d12=G	d13=H
Speaker Match	72.16	25.00	2.27
Condition Match	46.80	22.66	12.32
Total Match	46.88	13.12	0.00

These results confirm the ability of the covariance matrix to model speaker characteristics, with over 90% of test speakers correctly recognised using the arithmetic-harmonic sphericity distance measure. Note that the best performance comes from the first three distance measures which are also the least computationally expensive to compute.

An interesting fact from the data is the amazingly poor performance of the harmonic mean as a distance measure. Only 2% of speakers were correctly classified, a figure which is lower than

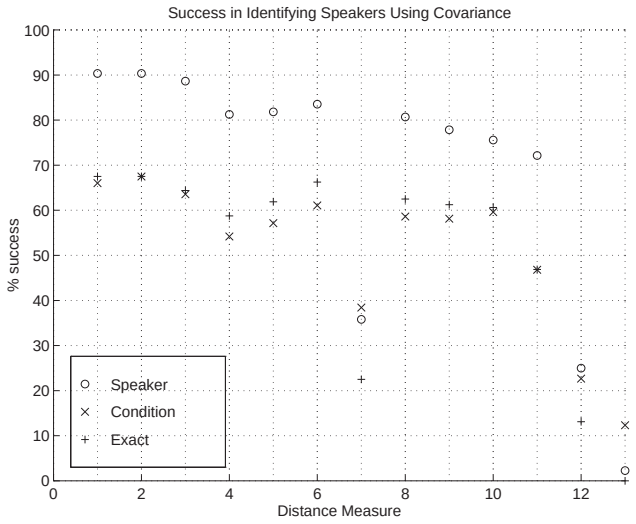

Figure 2: Results of using distance metrics d1-d13 on broadcast news data

that which would be obtained by chance. This suggests that H could be used as an *anti-distance measure* of some sort, i.e. the reciprocal or negative of H could be used as a distance measure. To check this hypothesis the experiment was re-run with some modified distance measures, d14-d16. The Bhattacharyya and divergence metrics for Gaussian distributions [6] were also added as d17-18.

$$
\begin{aligned}
d14 &= \log\left(\frac{A}{GH}\right) \\
d15 &= \log\left(\frac{A^2}{GH}\right) \\
d16 &= A - \log(G) - H \\
d17 &= \frac{1}{8}(\mu_x - \mu_y)^T \left(\frac{X+Y}{2}\right)^{-1} (\mu_x - \mu_y) + \frac{1}{2}\ln\left(\frac{|(X+Y)/2|}{|X|^{\frac{1}{2}}|Y|^{\frac{1}{2}}}\right) \\
d18 &= \frac{1}{2}tr(X^{-1}Y + Y^{-1}X - 2I) + \frac{1}{2}(\mu_x - \mu_y)^T(X^{-1} + Y^{-1})(\mu_x - \mu_y)
\end{aligned}
$$

The results are given in table 3 and illustrated in figure 3.

DISTANCE	d14	d15	d16	d17	d18
Speaker Match(%)	61.36	91.48	89.20	89.77	90.34
Condition Match(%)	52.71	67.00	67.49	70.44	70.44
Total Match(%)	48.75	68.13	68.75	73.13	73.13

Table 3: Results using the Covariance Data on d14 to d18

All the distance metrics so far have been based on the fact that the representative matrix for each segment is a covariance of samples from a Gaussian distribution. Later work makes use of a different matrix whose properties are not as well-defined. In order to be able to make a fair comparison between these representations, three more distance metrics were added which can be used between any arbitrary equal-sized matrices as they work in an *element-wise* fashion. These are the Euclidean, city and angular elementwise distances, d19-21.

[6] The derivations of these formulae can be found on [22, p142].

The maximum singular value of the difference (Y-X) [7] and the Euclidean distance of the mean were also added as d22 and d23 respectively.

$$
\begin{aligned}
d19 &= \sum_{i=1}^{D}\sum_{j=1}^{D}(x_{ij}-y_{ij})^2 \\
d20 &= \sum_{i=1}^{D}\sum_{j=1}^{D}|x_{ij}-y_{ij}| \\
d21 &= 1 - \frac{\sum_{i=1}^{D}\sum_{j=1}^{D}x_{ij}y_{ij}}{\left(\sum_{i=1}^{D}\sum_{j=1}^{D}x_{ij}^2\sum_{i=1}^{D}\sum_{j=1}^{D}y_{ij}^2\right)^{1/2}} \\
d22 &= \sigma^{+}(Y-X) \\
d23 &= \frac{1}{D}\sum_{i=1}^{D}(\mu_{yi}-\mu_{xi})^2
\end{aligned}
$$

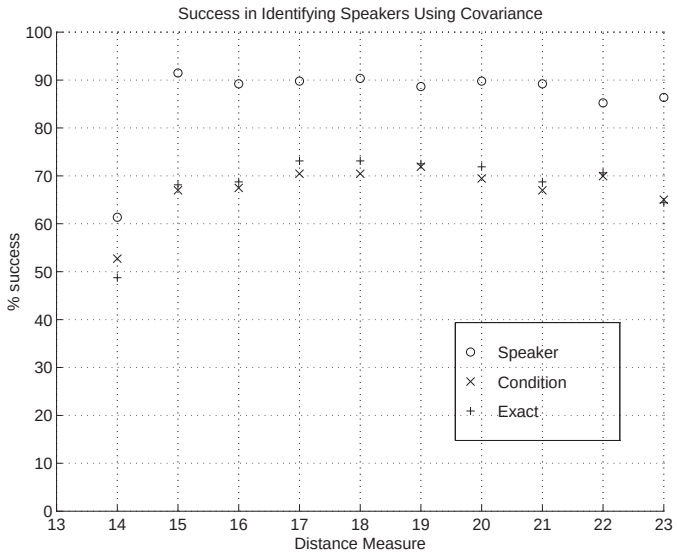

Figure 3: Results of using d14-d23 on broadcast news data

The results are given in table 4 and illustrated in figure 3.

DISTANCE	d19	d20	d21	d22	d23
Speaker Match(%)	88.64	89.77	89.20	85.23	86.36
Condition Match(%)	71.92	69.46	67.00	69.95	65.02
Condition Match(%)	72.50	71.88	68.75	70.63	64.38

Table 4: Results using the Covariance Data on d19 to d23

[7] representing the maximum gain of the matrix

2.4 Conclusions

The covariance information has been shown to offer a very good representation of the speaker-specific information in the signal, producing accuracy of over 90% with several different distance measures.

Table 5 summarises the properties of the best distance metrics which are used in later experiments in this project.

distance	Name	Speaker %	Symmetric	Zero if	positive
d15	$\log(A^2/GH)$	91.48	N	X=kY	Y
d1	Arithmetic-Harmonic Sphericity	90.34	Y	X=kY	Y
d2	Geometric-Harmonic Sphericity	90.34	N	X=kY	Y
d18	Gaussian Divergence	90.34	Y	X=Y, $\mu_x = \mu_y$	Y
d17	Bhattacharyya Distance	89.77	Y	X=Y, $\mu_x = \mu_y$	Y
d20	City Elementwise	89.77	Y	X=Y	Y
d16	A - log(G) - H	89.20	N	X=Y	Y
d21	Angular Elementwise	89.20	Y	X=kY	Y
d3	Log Likelihood Ratio	88.64	N	X=Y	Y
d19	Euclidean Elementwise	88.64	Y	X=Y	Y

Table 5: The best results obtained with covariance data

3 MLLR-based Methods

3.1 The Theory of MLLR

Maximum Likelihood Linear Regression (MLLR) is a speaker-adaptation technique which has attracted much recent research in speech recognition [6, 7, 13, 14, 17, 20]. A speaker independent system is built and then optimised in a maximum likelihood sense to adapt to a new speaker thus producing better performance and in the limit, the speaker-dependent rate.

Two methods of adaptation are possible

- Adapting the models to fit the data
- Normalising the data to fit the models

The original method of model adaptation assumed that the principal difference between speakers lay in the means and therefore just adapted the mean vector of the models [13]. A putative transcription is obtained (for example from forward-backwards alignment of the data) and a linear transform is applied to the mean parameters of the model so as to maximised the likelihood of the data.

Many models will have parameters *tied* together to ensure there is sufficient adaptation data for each set of parameters. This can be based on a regression class tree. For example, broad phoneme classes may be used such as stops, liquids, front, mid and back vowels, silence, fricatives and nasals. Each of these classes can then have the maximum likelihood transform matrix calculated.

An alternative to this approach is to apply the same transform structure to both mean and variance information. This is called *constrained model-space transformation*. [7] The model parameters are now changed by:

$$\begin{aligned} \hat{\mu} &= A_m \mu - b_m \\ \hat{\Sigma} &= A_m \Sigma A_m^T \end{aligned}$$

This can be shown to be equivalent to transforming the observed data with the addition of a log term [7]:

$$\begin{aligned} \hat{o}(\tau) &= A_m^{-1} o(\tau) + A_m^{-1} b_m \\ &= A o(\tau) + b \\ \mathcal{L}(o(\tau); \mu, \Sigma, A, b) &= \mathcal{N}(A o(\tau) + b; \mu, \Sigma) + \log(|A|) \end{aligned}$$

For the case of forming a single global transform matrix for each segment, this log term does not alter the performance and the method can be implemented by applying the transformation to the data and keeping the models fixed. This will subsequently referred to as the *data-based* MLLR transform. When applied to the data it normalises the input so that it matches the model parameters more closely.

By forming a global MLLR transform matrix for each segment, the variation due to the changing phonemes has been eliminated for both methods described above, since the transform at any given time will be from the data to the most active phoneme model. This suggests that the transform matrix itself could be a good representation of speaker identity.

3.2 Testing the MLLR matrices

The data-based MLLR transform was used, which warped the data from the original (x) to the new (y) domain according to the formula:

$$y = Ax + b = [b \;\; A] \begin{bmatrix} 1 \\ x \end{bmatrix}$$

The transform was generated using static, delta and acceleration coefficients in block diagonal form, i.e.:

$$A = \begin{bmatrix} A_s & 0 & 0 \\ 0 & A_d & 0 \\ 0 & 0 & A_a \end{bmatrix}$$

but these experiments were only conducted with the static information.

The MLLR (A) matrix itself is very different to the covariance matrix. It is not symmetric and it does not model a multi-dimensional Gaussian. This means that several of the previous distance measures used for the covariance matrix no longer apply. The Bhattacharyya and divergence distances were derived with the Gaussian assumption, and many of the metrics depend on the eigenvalues of the product XY^{-1}.

Considering the transform matrix warps the data to a set of fixed parameters, the information should be stored in an elementwise fashion within the matrix. Consider applying a rotation to the matrix A. Any distance metrics based on the determinant of A will give the same results as previously, but the rotated A matrix will no longer represent the MLLR transform of the data. This implies that using distance metrics based purely on the eigenvalues of matrices will not truly capture the speaker-dependent information stored in the A-matrix.

It is expected therefore that the best results will be obtained from elementwise operations, such as d19-d21, and as the matrix acts as an operator, the maximum singular value (d22) should also offer an insight into the speaker as it represents the 'gain' of the matrix.

The results from using the data-based MLLR matrices with the previous distance measures are given in table 6. For comparison, the results using the model-based MLLR transform are given in table 7 and both are plotted in figure 4.

DISTANCE	d1	d2	d3	d4	d5	d6	d7	d8	d9
Speaker Match(%)	2.27	1.14	0.00	79.55	78.41	77.84	32.95	2.27	76.70
Condition Match(%)	12.32	12.81	15.27	56.65	56.16	54.19	28.08	18.23	54.68
Total Match(%)	0.62	0.00	0.00	61.25	61.25	57.50	17.50	1.25	58.75
DISTANCE	d10	d11	d12	d13	d14	d5	d16	d17	d18
Speaker Match(%)	75.57	4.55	6.82	6.82	1.70	1.70	1.70	41.48	39.77
Condition Match(%)	54.19	17.24	19.70	21.18	25.12	11.82	24.63	42.86	41.38
Total Match(%)	58.12	1.88	2.50	2.50	1.25	0.62	1.25	30.00	28.75

	DISTANCE	d19	d20	d21	d22	d23
elementwise measures	**Speaker (%)**	90.91	90.34	90.91	85.23	50.00
	Condition (%)	62.56	64.04	64.04	63.05	41.38
	Total (%)	66.25	68.12	66.88	65.62	35.00

Table 6: Data-based MLLR transform Matrices results

These results confirm the inappropriateness of using many of the previous distance metrics on the MLLR transform matrices. As predicted however, the maximum singular value (d22) and the elementwise distances (d19-d21) work well, and show that the MLLR transform does indeed store information about the identity of the speaker.

The data-based transform shows these effects more markedly than the model-based transform and will be used as the representative MLLR transform for the rest of this project, as a higher

performance (90.91%) can be obtained with its use.

Note the poor performance of d23, which uses the offset vector. This fairs far worse than the equivalent mean-based measure in section 2.

DISTANCE	d1	d2	d3	d4	d5	d6	d7	d8	d9
Speaker Match(%)	26.70	33.52	35.80	78.98	78.41	67.61	32.39	16.48	77.27
Condition Match(%)	36.45	35.96	39.90	49.75	53.20	53.20	29.06	27.09	53.69
Total Match(%)	20.00	24.38	25.00	56.25	58.12	51.25	16.25	6.88	56.25
DISTANCE	d10	d11	d12	d13	d14	d15	d16	d17	d18
Speaker Match(%)	74.43	19.89	31.25	26.14	20.45	34.66	20.45	64.20	44.32
Condition Match(%)	49.75	18.72	20.20	15.27	35.96	39.90	37.93	54.68	44.83
Total Match(%)	52.50	7.50	15.62	10.62	15.00	25.62	16.88	46.25	29.38

elementwise measures

DISTANCE	d19	d20	d21	d22	d23
Speaker (%)	83.52	82.39	84.09	73.86	57.39
Condition (%)	61.58	60.59	59.11	55.67	48.28
Total (%)	61.25	60.00	60.63	53.12	41.25

Table 7: Model-based MLLR transform Matrices results

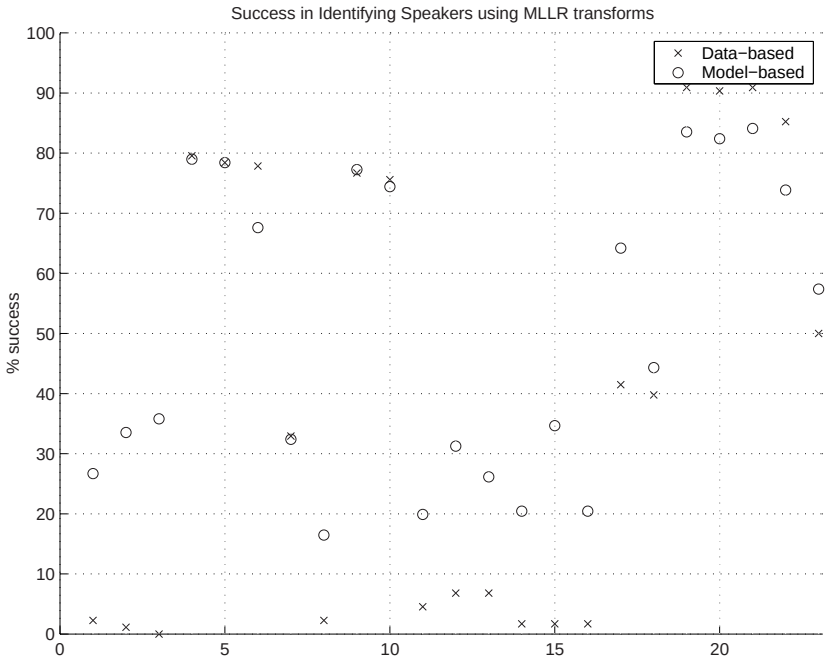

Figure 4: Results with MLLR transform Matrices

3.3 Transforming the Covariance Matrices

Let the original data be represented by mean μ_x and covariance Σ_x. After the data-based MLLR transform has been applied, the transformed data has mean μ_y and covariance Σ_y.

$$
\begin{aligned}
\mu_x &= E[x] \\
\mu_y &= E[y] \\
&= E[Ax + b] \\
&= A\mu_x + b
\end{aligned}
$$

$$
\begin{aligned}
\Sigma_x &= E[(x - \mu_x)(x - \mu_x)^T] \\
&= E[xx^T] - E[x]E[x]^T \\
\Sigma_y &= E[yy^T] - E[y]E[y]^T \\
&= E[(Ax + b)(x^T A^T + b^T)] - E[(Ax + b)]E[(Ax + b)]^T \\
&= E\left[[b \;\; A] \begin{bmatrix} 1 \\ x \end{bmatrix} [1 \; x^T] \begin{bmatrix} b^T \\ A^T \end{bmatrix}\right] - [b \; A] \begin{bmatrix} 1 \\ E[x] \end{bmatrix} [1 \; E[x]^T] \begin{bmatrix} b^T \\ A^T \end{bmatrix} \\
&= [b \;\; A] \begin{bmatrix} (1-1) & (E[x^T] - E[x]^T) \\ (E[x] - E[x]) & (E[xx^T] - E[x]E[x]^T) \end{bmatrix} \begin{bmatrix} b^T \\ A^T \end{bmatrix} \\
&= A \; \Sigma_x \; A^T
\end{aligned}
$$

3.4 Testing the Transformed Covariances

Applying the data-based MLLR transform to the covariance of the data is designed to increase recogniser performance by warping the data into a more normalised form, thus matching the generic models better. This implies that applying the transform to the data will remove some of the speaker-specific information and thus using the transformed covariance after the transform has been applied will be detrimental to the task of speaker recognition.

This hypothesis is confirmed by the results for the transformed covariance given in table 8 and illustrated in figure 5.

DISTANCE	d1	d2	d3	d4	d5	d6	d7	d8	d9
Speaker Match(%)	77.84	76.14	75.00	70.45	70.45	65.91	26.14	68.18	64.20
Condition Match(%)	56.16	44.83	48.77	45.81	25.12	48.28	45.81	46.80	29.56
Total Match(%)	52.50	46.88	52.50	45.62	10.62	48.12	45.62	44.38	22.50
DISTANCE	d10	d11	d12	d13	d14	d15	d16	d17	d18
Speaker Match(%)	62.50	39.77	32.39	22.73	13.07	77.84	17.61	73.86	73.30
Condition Match(%)	54.19	54.68	23.15	17.73	17.73	54.68	18.72	63.05	63.55
Total Match(%)	53.75	52.50	13.75	8.12	5.00	54.38	6.88	57.50	57.50

elementwise measures

DISTANCE	d19	d20	d21	d22	d23
Speaker (%)	69.32	72.73	71.02	64.77	69.32
Condition (%)	54.68	53.20	51.23	52.22	61.58
Total (%)	49.38	51.25	48.12	47.50	54.38

Table 8: Results from the Transformed Covariance

These results confirm that applying the MLLR transform to the data does indeed normalise it to some degree by removing speaker-specific information. It is interesting to note that the general trend is for the transformed covariance results to follow the same pattern as the original results, with the best results again being obtained from d15 ($\log(A^2/GH)$). Noticeable exceptions to this however, are shown as a dramatic drop in success with d11 (A), but *increase* in success for d12 (G) and d13 (H). This combined produces a large decrease in performance in d14 (log(A/GH)), and d16 (A-log(G)-H). In view of this it is rather surprising that d15 ($\log(A^2/GH)$) has remained relatively unscathed.

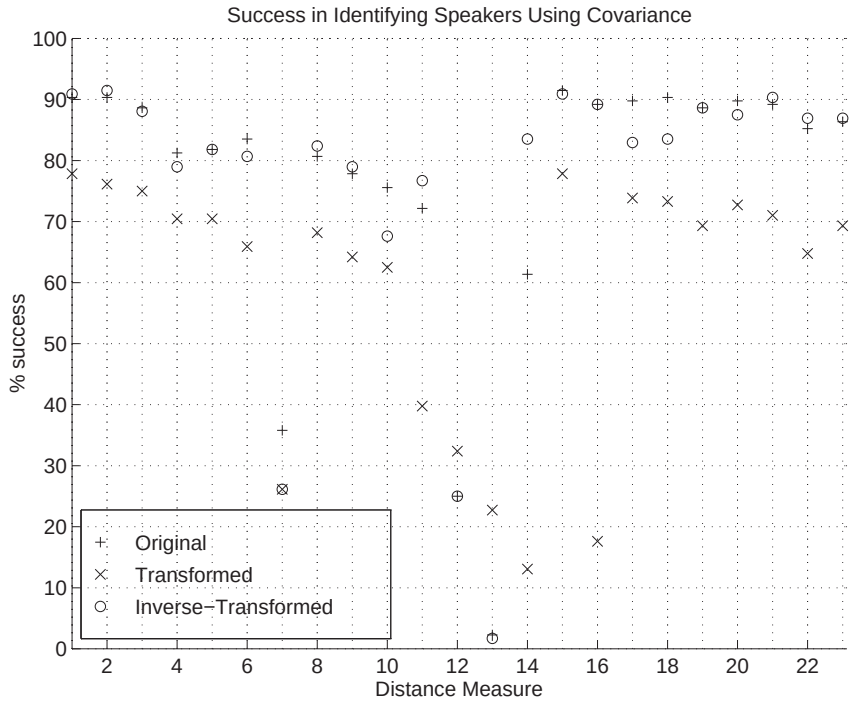

Figure 5: The Effect of Transforming the Covariance Matrices

Since applying a transform which holds speaker-specific information to a covariance matrix which also holds speaker-specific information reduces performance by compensating for the speaker-dependent effect, it might be possible to exaggerate the speaker-specific properties of the matrix by applying the *inverse* transform to the covariance. This concept is illustrated in figure 6.

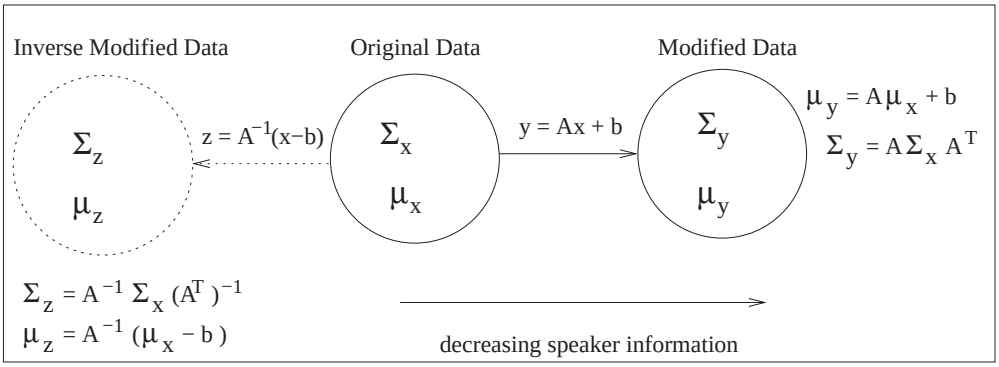

Figure 6: Transforming the Covariance Data

The new covariance equations become:

$$\mu_z = A^{-1}(\mu_x - b) \quad (1)$$

$$\Sigma_z = A^{-1}\Sigma_x(A^T)^{-1} \quad (2)$$

The results from "*inverse-transforming*" the covariance data are given in table 9 and illustrated in figure 5

DISTANCE	d1	d2	d3	d4	d5	d6	d7	d8	d9
Speaker Match(%)	90.91	91.48	88.07	78.98	81.82	80.68	26.14	82.39	78.98
Condition Match(%)	69.95	69.46	63.05	52.22	57.14	59.11	33.00	60.10	56.16
Total Match(%)	74.38	75.00	65.62	56.88	63.12	63.75	16.25	67.50	58.75
DISTANCE	d10	d11	d12	d13	d14	d15	d16	d17	d18
Speaker Match(%)	67.61	76.70	25.00	1.70	83.52	90.91	89.20	82.95	83.52
Condition Match(%)	48.77	52.22	22.66	12.32	68.47	68.47	65.52	66.01	67.49
Total Match(%)	46.88	53.75	13.12	0.62	70.00	73.75	69.38	65.62	64.38

	DISTANCE	d19	d20	d21	d22	d23
elementwise measures	**Speaker (%)**	88.64	87.50	90.34	86.93	86.93
	Condition (%)	67.98	67.49	68.47	65.02	69.46
	Total (%)	72.50	72.50	71.88	70.62	70.62

Table 9: The Effect of Inverse-Transforming the Covariance Matrices

3.5 Conclusions

The experiments done in this section have confirmed the ability of the MLLR transform matrix to store speaker-dependent information. They have also shown the success in applying MLLR transforms to normalise input data from different speakers with a resulting drop in speaker-recognition rate.

The application of the inverse-transform to the covariance matrices produces comparable results to the untransformed case with a success rate of over 90% being obtained with four distance metrics.

It is interesting to note that the offset vector does not appear to contain as much speaker-dependent information as the mean-vector for the Gaussian case. (d23)

The best results obtained using MLLR data are given in table 10 and are seen to be comparable with the best covariance results given in section 2. The particular values for these distance metrics for the original covariance case are also shown for ease of comparison.

Method	**Distance**		**Speaker %**	**Covariance**
Data-based MLLR matrix	d19	Euclidean Elementwise	90.91	88.64
	d21	Angular Elementwise	90.91	89.20
	d20	City Elementwise	90.34	89.77
	d22	Maximum Singular Value	85.23	85.23
Inverse-Transformed Covariance	d2	Geometric-Harmonic Sphericity	91.48	90.34
	d1	Arithmetic-Harmonic Sphericity	90.91	90.34
	d15	$\log(A^2/GH)$	90.91	91.48
	d21	Angular Elementwise	90.34	89.20
	d16	A-log(G)-H	89.20	89.20

Table 10: The Best Speaker Recognition Results using MLLR Matrices

4 Hierarchical Clustering

4.1 What is Clustering?

The results presented in sections 2 and 3 show that the covariance and MLLR transform matrices hold sufficient information to be useful in the speaker recognition task. In the previous experiments many samples of reference speakers existed and each unknown test segment was compared to them and assigned to the closest match. Whilst demonstrating the feasibility of speaker identification with these matrices, it did not provide much help for real-world applications, where no reference speaker information exists.

For example, consider analysing a broadcast news show and trying to determine which segments came from the same speaker. No reference segments are available, so the closest speakers must be matched "on the fly". This involves grouping the segments into clusters whose members are in some way more similar to the other members of their cluster than members of the other clusters.

Several problems exist with clustering techniques. Often, since proximity decisions may be made on a local basis, there is no guarantee the solution found will be a global optimum. Also it is not always apparent what the correct solution should be and so judging how "successful" a clustering scheme is becomes a matter of subjective opinion. For example, consider the 2-dimensional data shown in figure 7. It is not obvious how many clusters the data falls in to despite the fact that it appears that clustering would be beneficial in this case.

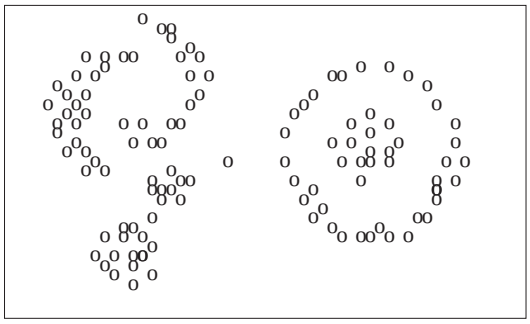

Figure 7: Uncovering Clusters in 2D data

Another difficulty is knowing when to stop clustering. If the success of clustering is based on some average measure of intra and inter-cluster distance, then *splitting* a cluster will always produce "better" performance. In the limiting case the best clusters would just be having one segment per cluster, which obviously is unacceptable if the purpose of clustering is to group segments from the same speaker together. The solution to this problem is either to specify before clustering begins how many clusters are desired, or to define some suitable stopping criterion, for example forcing all clusters to have a minimum occupancy count.

This section investigates several methods of clustering data based on two and three-speaker problems in order to ascertain which would be useful in a speaker-tracking system.

4.2 Different Types of Hierarchical Clustering

Agglomerative Methods

Agglomerative clustering starts by defining a cluster centred on each individual segment. The nearest pair of distinct clusters are then found and merged to form a new cluster. Some statistic is generated to represent this new cluster. This merging process is then repeated until the desired number of clusters is obtained.

If a record is maintained of the clustering decision at each stage the clustering process only needs to be done once, and results for different numbers of clusters can be obtained by retracing the resulting *dendogram* until the desired number of clusters exist.

The difference between agglomerative clustering schemes lies in the way the closest clusters are found and how the representative statistic of the new cluster is generated. For more detailed information on clustering methods and applications see [4].

4.2.1 Merging Statistics of Clusters

Concatenation

This method is not generally used in clustering, but is relevant for this particular problem. The clusters to merge are simply given by the ones which lie closest together according to the distance metric being used. The representative statistic is found by assuming the segments in the cluster originate from the same speaker, and thus the true statistics of that speaker can be found by calculating new mean and variance information which would result had the data in both groups been concatenated.

$$\begin{aligned} n &= n_1 + n_2 \\ \mu &= \frac{n_1\mu_1 + n_2\mu_2}{n} \\ \Sigma &= \frac{n_1(\Sigma_1 + \mu_1\mu_1^T) + n_2(\Sigma_2 + \mu_2\mu_2^T)}{n} - \mu\mu_T \end{aligned}$$

This method will not work successfully for cross-speaker clustering as the data does not originate from the same speaker and therefore there is no guarantee that the representative statistic will be a fair representation of the combined covariances. Consider the 2-dimensional data in figure 8. The covariance of the two speakers is roughly the same leading to them being clustered together. However, the joint covariance (shown dotted) does not capture this information at all.

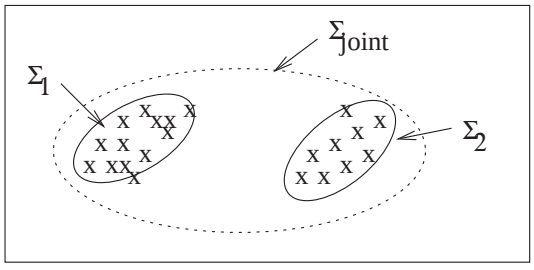

Figure 8: Problems with Concatenation Based Clustering

Centroid Clustering

This is similar to concatenation-based clustering, only instead of combining the statistics by assuming all the data came from the same speaker the weighted average values are used. Thus the combinatory formulae now become

$$\begin{aligned} n &= n_1 + n_2 \\ \mu &= \frac{n_1\mu_1 + n_2\mu_2}{n} \\ \Sigma &= \frac{n_1\Sigma_1 + n_2\Sigma_2}{n} \end{aligned}$$

This reduces the problem of a pooled covariance, but still causes problems when the mean is used as a representative statistic (see figure 9).

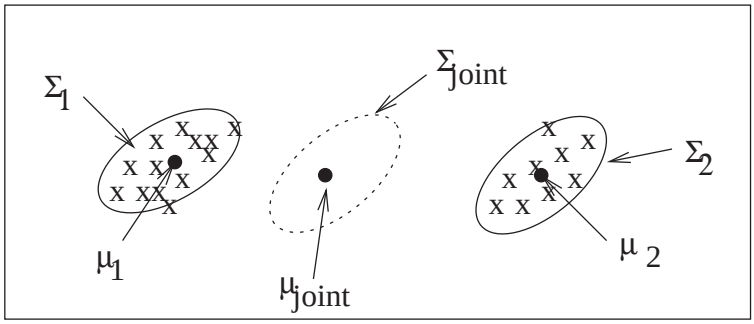

Figure 9: Problems with Centroid and Median Based Clustering

Median Clustering

Median clustering is the same as centroid clustering except that it ignores the number of data points in each cluster. This means that the data is not weighted by the confidence in that covariance being an accurate representation of the segment, and thus the performance using median clustering should be worse than that of centroid clustering.

$$\begin{aligned} n &= n_1 + n_2 \\ \mu &= \frac{\mu_1 + \mu_2}{2} \\ \Sigma &= \frac{\Sigma_1 + \Sigma_2}{2} \end{aligned}$$

Median clustering also suffers from problems associated with a pooled mean, illustrated in figure 9.

4.2.2 Neighbourhood Clustering Schemes

Nearest Neighbour (single linkage)

The distance from every segment to every other segment is calculated and stored in the global distance matrix. A record is maintained of which segments belong to which cluster. The inter-cluster distance between clusters A and B is then given by the minimum distance between any member of A and any member of B. This is illustrated in figure 10.

The two groups with the smallest inter-cluster distance are then merged simply by re-defining the members of the new cluster and de-activating the old two groups.

Unlike in the previous algorithms, no new distances need to be calculated and clustering can be done based purely on the global distance matrix.

Furthest Neighbour (complete linkage)

Furthest neighbour clustering is identical to nearest neighbour except the maximum distance between any member of cluster A and any member of cluster B is used to define the inter-cluster distance.

Furthest neighbour clustering has generally been shown to work better than nearest neighbour as it is more robust to data which contains a few isolated outliers.

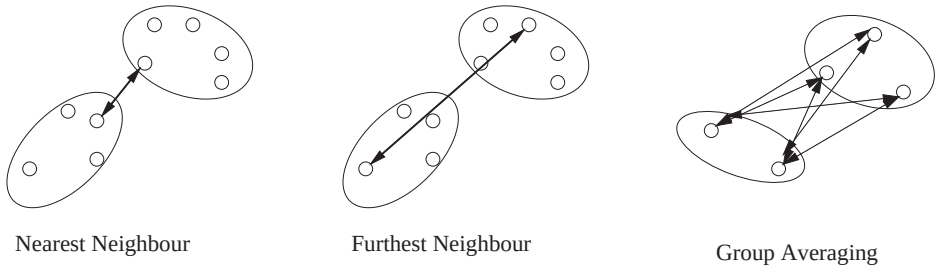

Figure 10: Distances used in Clustering

Mutual Nearest Neighbour

Mutual nearest neighbour clustering tries to take into account the relative proximity of the combining clusters to all the others. Suppose the individual *k* is the *m*th nearest neighbour of individual *j* and individual *j* is the *n*th nearest neighbour of individual *k* then the mutual neighbourhood value is given by (*n*+*m*). This value is then substituted into the global distance metric used by the nearest neighbour clustering algorithm.

Note because a ranking is used rather than the distance values explicitly this method gives rise to an increased likelihood of ties in scoring, so care should be taken by looking at the critical values to see if they are equal before reading too much into the clusters produced.

Group Averaging

Group average clustering is again similar to nearest neighbour clustering, except that the inter-cluster distance is now defined as the average of the distances between all pairs of individuals consisting of one member in A and the other in B. This is illustrated in figure 10.

Although computationally more complex than nearest neighbour it should offer a more robust representation and also does not need recomputation of the distance metric (unlike the statistical-merging based methods).

Ward's Hierarchical Method

Ward's clustering forms the partitions which minimise the loss associated with each grouping. At each step the merging of every possible pair of clusters is considered and the fusion which results in the minimum increase in an *information loss* is implemented. This information loss is defined in terms of an error sum-of-squares criterion (ess).

For example the 1D case each group k, has

$$ESS_k = \sum_{i=1, i \epsilon k}^{n_k} (x_i - \overline{x}_i)^2$$

with total information loss being

$$ESS = \sum_k ESS_k$$

This can be extended to include vectors by averaging over all features and to the matrix case by computing the sum elementwise. Note this can be seen to be loosely equivalent to using a squared-Euclidean distance metric.

Ward's method has been shown to impose a spherical solution onto clustering problems [4, p70] and therefore may not perform well on real data.

4.2.3 Discriminatory Schemes

Lance and Williams' recurrence formula

The Lance-Williams recurrence formula gives the distance from a point k to a combined group $\{ij\}$ as:

$$d_{k(ij)} = \alpha_i d_{ki} + \alpha_j d_{kj} + \beta d_{ij} + \gamma |d_{ki} - d_{kj}|$$

where d_{ik} represents the distance from i to k.

For example in the case where

$$\alpha_i = \alpha_j = 0.5; \quad \beta = 0; \quad \gamma = -0.5;$$

then

$$\begin{aligned} d_{k(ij)} &= 0.5(d_{ki} + d_{kj}) - 0.5|d_{ki} - d_{kj}| \\ &= d_{kj} \quad iff \quad d_{ki} > d_{kj} \\ &= d_{ki} \quad iff \quad d_{kj} > d_{ki} \end{aligned}$$

and thus these values represents nearest neighbour clustering.

Similarly, furthest neighbour clustering is defined by

$$\alpha_i = \alpha_j = 0.5; \quad \beta = 0; \quad \gamma = 0.5;$$

and group average by

$$\alpha_i = \left(\frac{n_i}{n_i + n_j}\right); \quad \alpha_j = \left(\frac{n_j}{n_i + n_j}\right); \quad \beta = 0; \quad \gamma = 0$$

These values, although using the same distances as those discussed in section 4.2.2 can potentially give different clusters. This is because the method of calculating which groups to cluster is altered to make the clustering more discriminatory. In the traditional nearest-neighbour routine, the groups which possess the closest neighbours are combined. In this version, the total distance from all other clusters to the new combination is calculated for each possible merge, using the formula above. The *maximum* of this cumulative distance is then found to give the clustering combination which is furthest away (in some sense) to the rest of the clusters. This merge is then applied and the distance matrix is then updated taking it into account.

Lance-Williams suggest the following values may perform well:

$$\alpha_i + \alpha_j + \beta = 1; \quad \alpha_i = \alpha_j; \quad \beta < 1; \quad \gamma = 0$$

They suggest

$$\begin{aligned} \beta &= -0.25; \quad \alpha_i = \alpha_j = 0.625 \\ or \ \beta &= -0.5; \quad \alpha_i = \alpha_j = 0.75 \end{aligned}$$

Divisive Methods

Divisive methods use a top-down approach to clustering. They start with one large cluster and split it into two smaller clusters. This can then be repeated recursively until the desired number of clusters is reached.

4.2.4 Polythetic clustering - MacNaughton/Smith Method

All the individuals initially start in one large cluster. The individual which is furthest from all the other individuals is then split off to form a splinter group. All the individuals left in the main group then have the average distance from themselves to the member(s) of the splinter group and the average distance of themselves to the other member(s) of the main group calculated.

A mathematical measure is then calculated for each individual of the main group by subtracting the average-to-splinter from the average-within-main values. The maximum value of this measure across all individuals in the main group is found. If this is greater than zero, that individual is added to the splinter group and the process is repeated, otherwise the splitting is stopped.

4.3 Initial Experiments

As a simple test for the clustering methods described above, six segments from the show `a960521` were taken. To make it easier, the first three were speaker "`Ted_Koppel`", with acoustic condition "`F0`" [8] whilst the second three were "`a960521_janedoe001`" with acoustic condition "`F2`".

All the numerical results from these experiments are reproduced in appendix B.1 for the readers reference. When a distance measure was needed, the arithmetic-harmonic sphericity metric was used.

4.3.1 Statistical Merging of Clusters

The statistical merging methods all clustered the data based on the arithmetic-harmonic sphericity distance matrix in the way shown in table 11. The minimum and second minimum values are also given in the table. This provides an indication of the *confidence* associated with each clustering decision. The greater the relative difference between the two values the higher the confidence. As can be seen from the table, there is very little difference between the methods for this particular data.

						Concat		Centroid		Median	
01	02	03	04	05	06	min	2ndmin	min	2ndmin	min	2ndmin
|	|	|	|	|	|						
07	|	07	|	|	|	0.22	0.26	0.22	0.26	0.22	0.26
|	|	|	|	|	|						
08	08	08	|	|	|	0.23	0.47	0.23	0.47	0.23	0.47
|	|	|	|	|	|						
|	|	|	09	09	|	0.47	0.50	0.47	0.50	0.47	0.50
|	|	|	|	|	|						
|	|	|	10	10	10	0.43	0.63	0.42	0.62	0.41	0.64
|	|	|	|	|	|						
11	11	11	11	11	11	0.55	---	0.55	----	0.58	----

Table 11: Statistical Merging Clustering on 2-Speaker Problem

[8] see table 1 for a definition of these categories

4.3.2 Neighbourhood Clustering Schemes

The dendograms for the neighbourhood clustering methods using the arithmetic-harmonic sphericity distance measure are given in figure 11, whilst those obtained from Ward's algorithm on the mean and covariance of the data are given in figure 12.

All of these clustering schemes split the data into the desired clusters, but the Ward's mean-based results can clearly be seen to offer the most conclusive evidence of the presence of two clusters within the data.

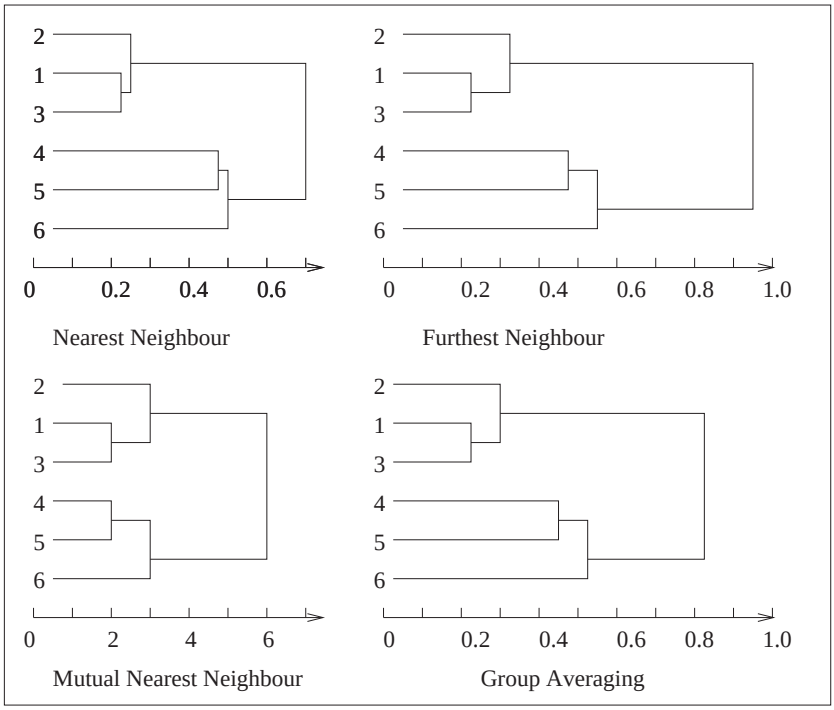

Figure 11: Dendograms for Neighbourhood Clustering Methods

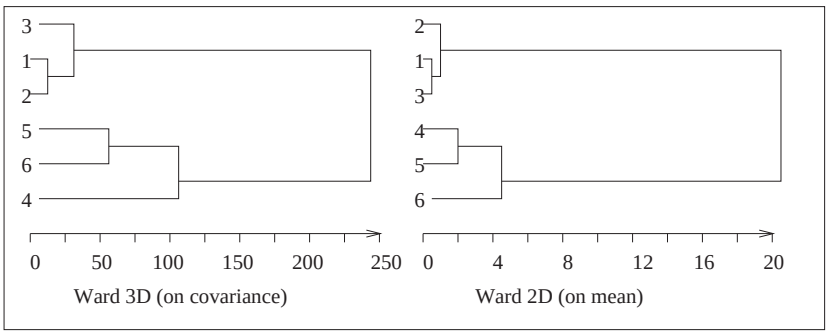

Figure 12: Dendograms for Ward's Clustering

4.3.3 Lance-Williams' Clustering

The dendogram for the Lance-Williams' clustering is given in figure 13. Notice that since a new distance matrix is calculated at each step of the algorithm, there is no guarantee that the maximum values on merging at each stage will decrease monotonically. This can be seen in the case where β=-0.5 where the first merging occurs with value 3.65 but the second with 3.86

Again, all four schemes split the data into the correct clusters.

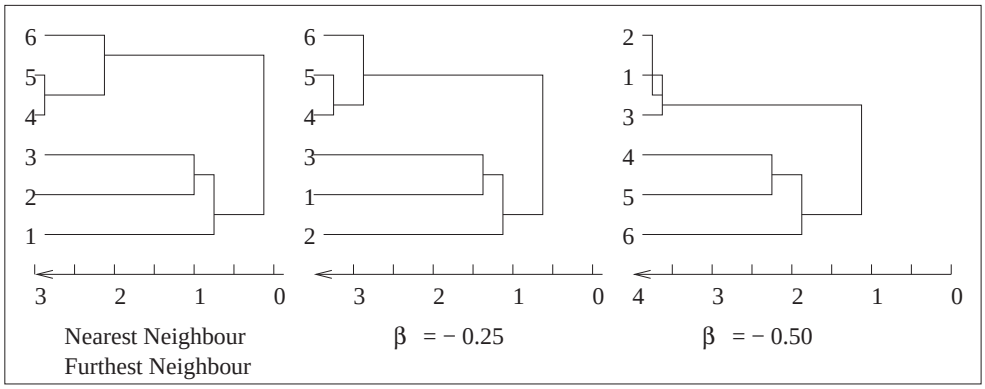

Figure 13: Dendograms for Lance-Williams

4.3.4 Polythetic Clustering

Only the groupings are returned from the polythetic clustering algorithm. These results which again are correct, are:

```
>> [splinter,rest] = split_cluster(dist_ahm)
splinter =
     5     4     6

rest =
     1     2     3
```

4.3.5 Summary

Every method of clustering discussed in section 4.2 correctly classified the segments into two speakers. This is encouraging, even though the task was made artificially easy by using a man and a woman with different acoustic conditions.

To discriminate between the clustering methods, a harder classification task is needed.

4.4 Further Experiments

4.4.1 Experiment 2 - Increasing the Number of Segments

The difficulty of the classification task grows as the number of segments increases. For the two class problem, the growth is exponential as the number of ways of splitting n segments into 2 clusters is given by *N(n,2)*

$$N(n, 2) = 2^{n-1} - 1$$

New acoustic conditions were also included to make the classification more difficult and the effect of changing the distance metric was investigated.
The new initial segments are now:

```
% # S1:a960521_Ted_Koppel_128818_130801_F0
% # S2:a960521_Ted_Koppel_133983_134965_F0
% # S3:a960521_Ted_Koppel_135021_136039_F0
% # S4:a960521_Ted_Koppel_107184_108304_FX
% # S5:a960521_Ted_Koppel_15093_17593_F2
% # S6:a960521_Ted_Koppel_173951_175700_F4
% # S7:a960521_a960521_janedoe001_48331_49159_F2
% # S8:a960521_a960521_janedoe001_51630_52187_F2
% # S9:a960521_a960521_janedoe001_52297_53064_F2
% # S10:a960521_a960521_janedoe001_49159_49942_FX
```

showing the desired clustering combination to be {1 2 3 4 5 6} and { 7 8 9 10}. The results using the divergence (d18), Bhattacharyya distance (d17), arithmetic-harmonic sphericity (d1), arithmetic-geometric sphericity (d2), log-likelihood ratio (d3) and log(A^2/GH) (d15) are given in table 12. More detailed results are given in appendix B.2.

	Experiment 2						**Experiment 3**	
Distance	d18	d17	d1	d17	d3	d15	gender split	3-speaker split
Concatenation	Y	Y	Y	Y	Y	Y	Y	
Centroid	Y	Y	Y	Y	Y	Y	Y	
Median	Y	Y	Y	Y	Y	Y	Y	
Nearest Neighbour (NN)	Y	Y	Y	Y	Y	Y	Y	Y
Furthest Neighbour (FN)	Y	Y	Y	Y	Y	Y	Y	Y
Mutual N Neighbour	Y	Y	Y	Y	Y	Y	Y	Y
Group Averaging	Y	Y	Y	Y	Y	Y	Y	Y
Lance Will. $\beta = -0.5$ (LW1)		Y					Y	
Lance Will. $\beta = -0.25$ (LW2)					Y			
LW. Nearest Neigh (LW4)								
LW. Furthest Neigh (LW8)								
divisive method	Y	Y	Y	Y	Y	Y	Y	Y
Ward 2D (means)	Y							
Ward 3D (variances)	Y							

Table 12: Results for Expt 2: 10-segment 2 cluster, and Expt 3: 20-segment 3 cluster

4.4.2 Experiment 3 - Adding More Speakers

A three-speaker experiment was then run to investigate the success of the system when more than two speakers were present.

The data used was

	F0	F2	F4	FX
Ted_Koppel	3	1	1	4
a960521_janedoe001		3		1
a960521_johndoe007				7

The results are presented in table 12 with more detailed results given in appendix B.3.

4.4.3 Experiment 4 - Short Segments

The success of the clustering methods using short segments was evaluated in order to investigate their robustness. Shorter segments do not have as much information available to make up their representative matrix and hence are more vulnerable to the effects of noise. This experiment used the first 100 frames (1 second) of speech from two `Ted_Koppel_FX` and two `a960521_janedoe001_F2` segments. The results are given in table 13.

Distance	div	bhat	log(A/H)	log (A/G)	lhr	log(AA/GH)
Concatenation						
Centroid						
Median						
Nearest Neighbour						
Furthest Neighbour	Y	Y	Y	Y		Y
Mutual N Neighbour		Y				
Group Averaging	Y	Y	Y	Y	Y	Y
Lance Will. $\beta = -0.5$	Y	Y	Y	Y		Y
Lance Will. $\beta = -0.25$	Y	Y	Y			Y
LW. Nearest Neigh	Y	Y	Y	Y		Y
LW. Furthest Neigh	Y	Y	Y			Y
divisive method	Y	Y	Y			

Ward 2D (means)	Y	Ward 3D (variances)	-

Table 13: Results for 4 short-segment 2 cluster case

4.5 Conclusions

The results show that all the clustering algorithms presented here work reasonably well on small problems. The neighbourhood methods produced the best results in the 3-speaker case whilst the Lance-Williams formulations worked well for the short segment case indicating they might be more robust than several of the other methods.

Furthest neighbour clustering was more successful than nearest neighbour [9] but was itself just out-performed by the computationally more expensive group-averaging scheme. Since the final system to be built will involve clustering many more segments than presented here, computational complexity becomes an increasingly important factor in choosing the desired clustering scheme.

[9] This result is consistent with that found in [10]

5 Scaling Up the System

5.1 Quantitative Evaluation of Clustering Performance

The previous section of this report evaluated the success of a clustering scheme in terms of whether or not it could separate different speakers into distinct clusters. Whilst this is a good criterion for the case of many segments with only a few distinct speakers, it does not necessarily provide the best measure of performance in all cases. [12]

Consider a typical broadcast news show, there may be two or three anchor-men, each of whom speak for a considerable time. Also in the show there may be upto thirty "extra" people, (`john/janedoes`), taking part in interviews or discussions, who speak infrequently and only for brief periods. Whilst it is obviously advantageous to pick out the anchor-men as separate speakers, in some cases it is not necessary to distinguish between the `johndoes` and indeed can be better not to.

The recognition rate of speaker-adaptive systems increases with the amount of adaptation data available per speaker. If `johndoe001` and `johndoe002` have similar enough characteristics to be clustered together then it may also be true that the estimated parameters for the speaker-adaptive system for both of them are similar and hence the union of their segments would make more adaptation data available and hence allow a better model to be built without loss of information, thus increasing the performance of the system.

The introduction of a quantifiable measure is also necessary. The number of possible groupings of n individuals into p clusters is: [10]

$$N(n,p) = \frac{1}{p!}\sum_{i=1}^{p}(-1)^{p-i}\begin{pmatrix} p \\ i \end{pmatrix} i^n$$

Given a show with 60 initial segments from 5 speakers, 7.23x10^{39} possible clustering combinations exist. The chance of any of the clustering algorithms getting the correct classification, forgetting the problem of defining what that classification is, is extremely remote. Since all the clusterings are likely to be slightly wrong, some measure of deciding which are "less wrong" than the others is required. To remove the subjectivity from this decision, a quantitative measure is required.

For the case when the speaker-clusters are to be used in an MLLR-based speaker-adaptive recognition system, the auxiliary function value produced after the adaptation has taken place may provide the basis for such a measure. [11] This value represents the log likelihood of the data per frame after the MLLR transformation, given a particular alignment. The figure of merit for a given set of clusters can therefore be calculated by multiplying this likelihood value for each cluster (aux_i) by the number of frames in that cluster (f_i) and then summing over all the clusters.

$$FOM = \sum_{i=1}^{n} aux_i * f_i \tag{3}$$

where n = the number of clusters

In summary, the figure of merit for a given clustering scheme is calculated by the following process:

1. Cluster all the data into n clusters
2. Output the name of the clusters with their members to a text file
3. For each cluster:

[10] from [15]
[11] This was suggested by Dr. M. Gales.

- perform MLLR estimation on all the data in that cluster
- Multiply the resulting auxiliary function by the total number of frames in the cluster

4. Sum this product for all clusters

This value is negative and the higher (less negative) it is, the "better" the clustering performs under MLLR adaptation. It is postulated that maximising this value also corresponds to a good speaker-split.

5.2 Testing the Figure of Merit

The figure of merit given in equation 3 was evaluated for various artificially-formed clusters from the show `a960610` to ascertain whether or not it was a good representation of speaker-split.

The theoretical lower bound on performance was obtained by allocating all the data to one cluster. Splitting any combination of data off from this group to form a new cluster would automatically match the data better and therefore produce a higher FOM. The upper bound was also found by assigning each segment to its own cluster. Combining any of these clusters would lose information and therefore produce a lower FOM.

All segments in 1 cluster (1122.86s)	Lower Bound	-7226296
1 segment per cluster	Upper Bound	-6982096

The results given in table 14 are for the case of two clusters. The clusters were formed by splitting off all instances of segments whose name contained a certain string into one cluster and assigning the remainder to cluster two. Since the segment names are of the form

`a960610_Chris_Beary_22783_23378_F0.plp`

this allows all the instances of one speaker to be easily isolated. [12] This was done for several different speakers and the male/female split was also implemented as it was expected to yield the "best" possible performance for the two-cluster case. Splitting based on conditions, not speakers, was also tried, to offer a set of values against which the speaker-splits could be compared. The results are given in table 14 along with the length of the smallest cluster.

	FOM	Segments split off	Smallest Cluster	
	-7223075	all F3	20.29s	}
	-7220145	all F4	29.52s	}
	-7218867	all F2	93.78s	} condition splits
	-7215061	all F0 and F1	177.79s	}
increasing	-7214153	all F0	249.25s	}
performance	-7211182	all F1	427.04s	}
	-7200651	all janedoes	207.23s	}
⇓	-7195990	all john and janedoes	532.70s	}
	-7188574	Chris Beary	237.69s	} speaker splits
	-7186080	all johndoes	382.93s	}
	-7184229	Cokie Roberts	237.69s	}
	-7173364	all females	495.25s	} male/female split

Table 14: FOMs for two-cluster splits of a960610 show

These results confirm:

[12] The shell-scripts `handsplit` and `handsplit2` to accomplish this task are described in more detail in appendix I along with all the other implementations discussed in this section.

- all the two-cluster splits have a FOM which lies between the lower and upper bound.
- the best two-cluster split occurs in the male/female case
- the FOM is higher for all instances of a speaker being separated than for all instances of a condition being separated, suggesting that it is indeed a good measure of the separation of speakers
- whilst the size of the samples naturally has an effect on the score, with more evenly sized clusters generally producing better results, this is not as significant a factor as the "goodness" of split. This can be seen by noting that separating off the speech of Chris Beary produces a considerably better FOM than separating `F1` segments despite the latter having much more evenly-sized clusters.

5.3 Implementation of the Clustering Methods

The results from section 4 show that different clustering strategies work best on different data. For example, the Lance-Williams' recurrence formula worked best for shorter segments, whilst the neighbourhood methods were more successful on the 3-speaker test.

A program `cluster.c` [13] was written which, when given a distance matrix between the segments, produces a list of which cluster each segment has been assigned to. No information about the original segments or the way the distance matrix was generated was included. This allowed the implementation of the neighbourhood methods, the Lance-Williams' recurrence formulae and the divisive clustering scheme. The latter was not implemented as the top-down approach is more computationally expensive than agglomerative methods whilst group averaging and mutual nearest neighbour techniques were rejected on the grounds of being similar to, but again more computationally expensive than the other neighbourhood schemes.

More details of the program can be found in appendix I. The user can chose between nearest neighbour, furthest neighbour and all four types of Lance-Williams' clustering using the `-c` and `-w` options. The output file is specified using `-o` whilst the log file is set with `-l` and the trace level with `-T`. The `-f` flag gives the name of the file containing the distance matrix and `-n` is used to specify the number of desired clusters. `-d` can be used to change the form of the output from the standard form to one which shows all the clustering decisions, or one which just lists the members of each cluster.

5.4 Generation of the Distance Matrices

Two programs were written to generate an inter-segment distance matrix. The first, `gendist.c` uses the MLLR transform matrices as the basic representation of the segments. Elementwise Euclidean, city, and angular distances were allowed by setting the `-d` flag, along with the Arithmetic-Harmonic Sphericity measure to act as a comparison. The `-c` option allows the choice of coefficients to be made (statics, deltas and/or acceleration), whilst the `-v` flag specifies whether the block-diagonal (A) matrix, the (b) offset vector, or both should be used in the calculations.

The `-f` option is used to specify a list of files from which the A-matrices can be read. The identity of the speakers of these files used within the program is written to the speaker-ID file given by the `-s` option. The resulting distance matrix is output to the file specified by `-o`, whilst the log file is set with `-l` and trace level with `-T` as before.

The second program `gendistcov.c` kept the same `-T -l -o` and `-s` options, but changed the `-f` option to specify the list of files from which the *covariance* information could be obtained. The allowable distances (`-d`) were also altered to be in keeping with the results from section 2. The user can chose between the divergence, log(A/H), log(A/G), the likelihood ratio, log(AA/GH),

[13] A library of i/o, numerical, vector and matrix functions were written in `matlib.h` and `matlib.c`, which were used by all the programs discussed in this section. More details can be found in appendix I

A-log(G)-H or the Bhattacharyya distance measures. The `-a` option allows a list of A-matrices to be given if adaptation of the covariances is required.

5.5 Running of the system

A flow-chart showing the implementation of the clustering system is given in figure 14. This tracks the process from the initial segmentation right through to the final evaluation of the figure of merit.

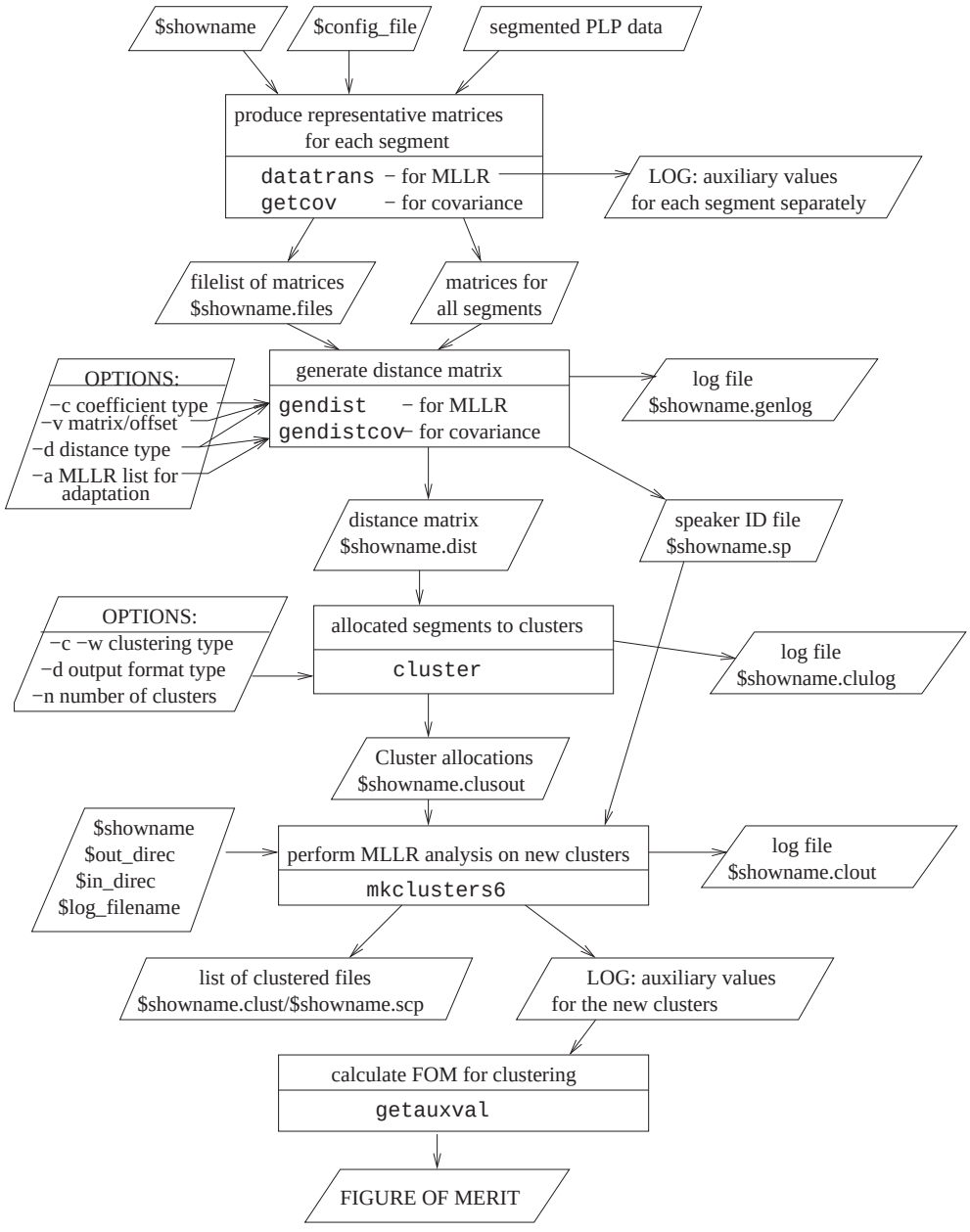

Figure 14: Overall Implementation of the System

5.6 Preliminary Results

The results for forming two, three and four clusters on the `a960610` show are illustrated in figure 15. More detailed results can be found in appendices C, Dand E.

Since the FOM is expected to lie between -7173364 and -7226296 for ease of readability the score reported is the (rounded) second to the fifth numbers in this value. This gives scores with a possible range of 1734 to 2263 for the two-cluster case, where the *minimum* value now represents the best score.

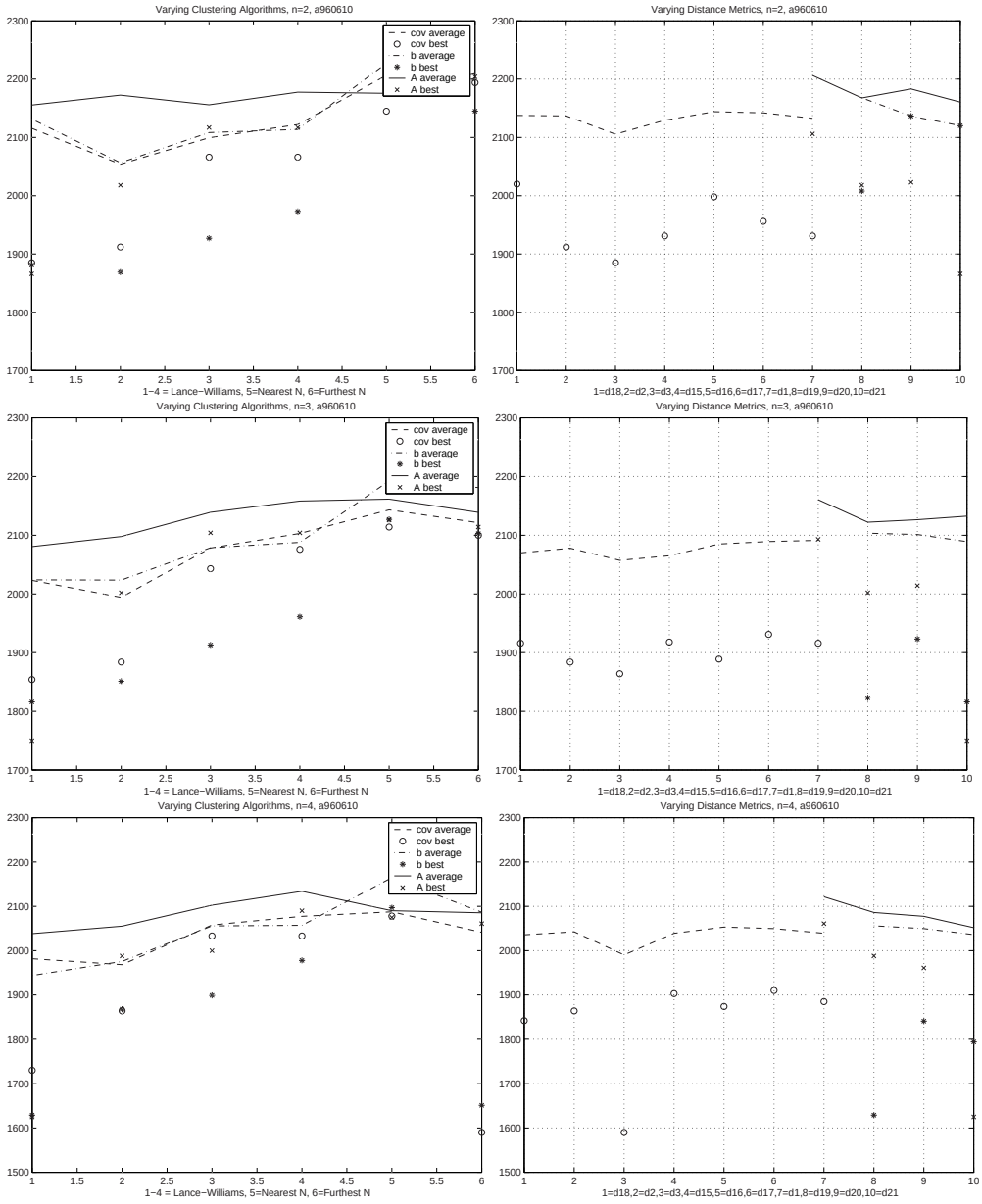

Figure 15: Results for 2,3 and 4-cluster case on a960610 show

5.7 Conclusions

The results show that Lance-Williams clustering with $\beta = -0.25$ (LW2) or $\beta = -0.5$ (LW1) give the best performance for this data. Nearest neighbour clustering performs the worst due to a phenomenon called *chaining*. This describes the tendency to group together relatively distinct clusters which have a series of intermediates between them. This makes the method very sensitive to noise and thus causes poor performance.

Furthest neighbour clustering does not suffer from this problem and has also been shown to be less sensitive to particular types of observation error [4, p71] than nearest neighbour. This is reflected in better performance in the furthest neighbour case, especially for 4 clusters.

Both neighbourhood methods however, are not discriminative in that they only consider the minimum of a certain distance between the groups. Lance-Williams formulae (LW1 and LW2), take all the distances between all segments into account, joining clusters which are close to each other but also further away from the remaining ones. This accounts for the generally better performance of these methods.

The covariance-based methods generally have a better average performance than the MLLR-based methods, although the best individual cases often came from the latter. Bearing in mind the poor performance of the offset vector in the speaker-identification task in section 3 it is perhaps surprising to note that it seems to perform better than the A-matrix case on this data. However, when the numbers are examined, [14] it is apparent that adding delta coefficients to the feature vector improves performance in the case of the covariance and offset-vector representations, but generally decreases performance for the A-matrix case. Since the average is calculated across the different feature vectors this has the effect of relatively decreasing the performance of the A-matrix case. It is also noted that adding acceleration coefficients does not produce an improvement in any case.

There is little to chose between the distance measures for the covariance-based cases with d3 (A-log(G)-1) performing best, although the angular separation appears to have a slight advantage in the MLLR-based methods.

The clustering decisions and final groupings for the best cases of 3 and 4 clusters are given in figures 16 and 17 respectively.

The 3-cluster case originates from Lance-Williams clustering with $\beta = -0.5$ (LW 1) and the angular separation (d21) on the static A-matrix. The tree structure shows the tendency of Lance-Williams clustering to group the data into a few large *plateaus* i.e. to keep adding to the same cluster on each clustering step. This is due to the discriminatory way the clustering works, with a similar distance being obtained from all data in a group which is quite far from the rest of the data. By noting when the clustering switches between plateaus an indication of when that cluster has finished growing can be obtained. This shows that all the segments of that speaker are likely to have been found. This can then be used to help determine the number of speakers in the sound-track.

The 4-cluster case originates from furthest neighbour clustering using d3 (A-log(G)-1) on the static covariance matrix. This shows no tendency to have plateaus since once a group has been combined it is represented by its furthest distance to the other clusters and hence the focus of clustering is likely to switch on each step. This makes determining a stopping criterion more difficult than for the previous case.

Both these tree-diagrams illustrate the ability of the methods to pick out speakers, with only a few segments being obviously mis-classified.

[14] see appendices C, D and E for details.

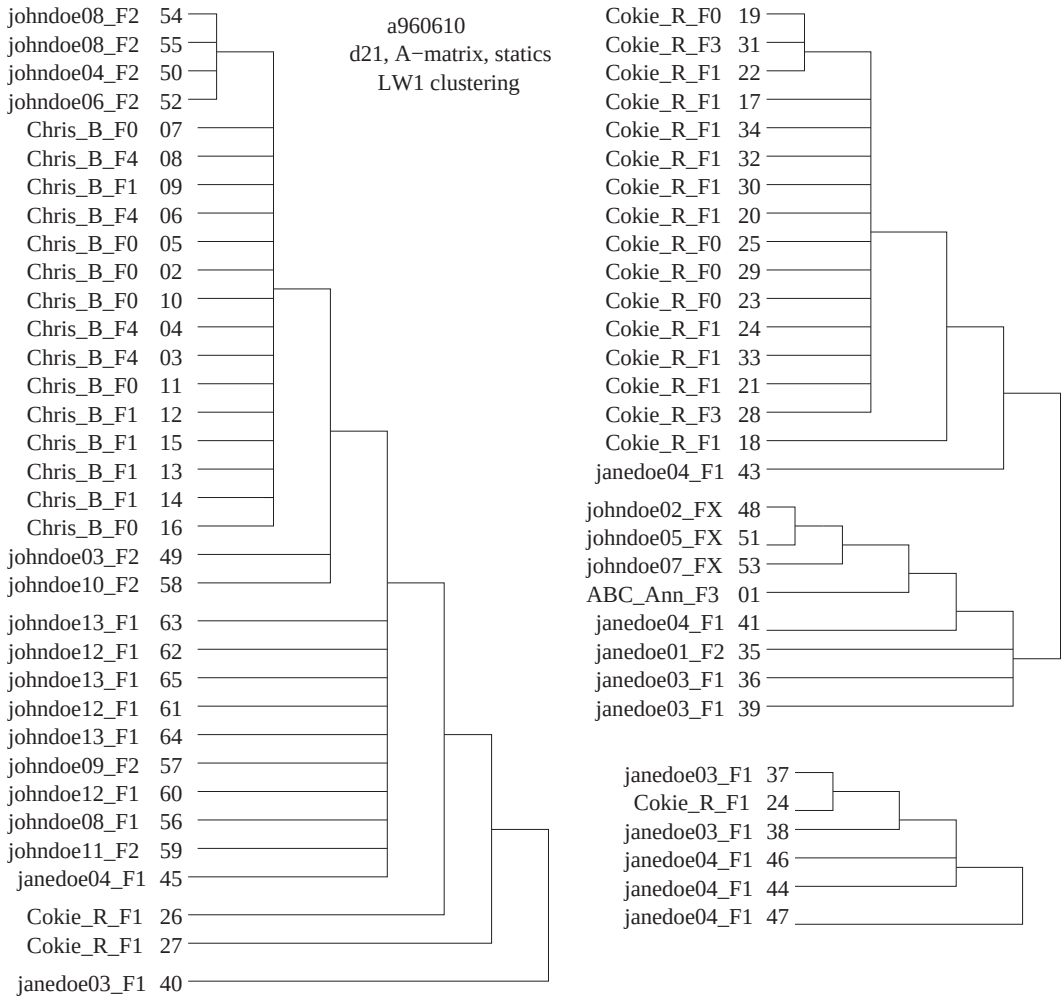

Figure 16: Clusters formed from A-matrix, Lance-Williams 1 and d21

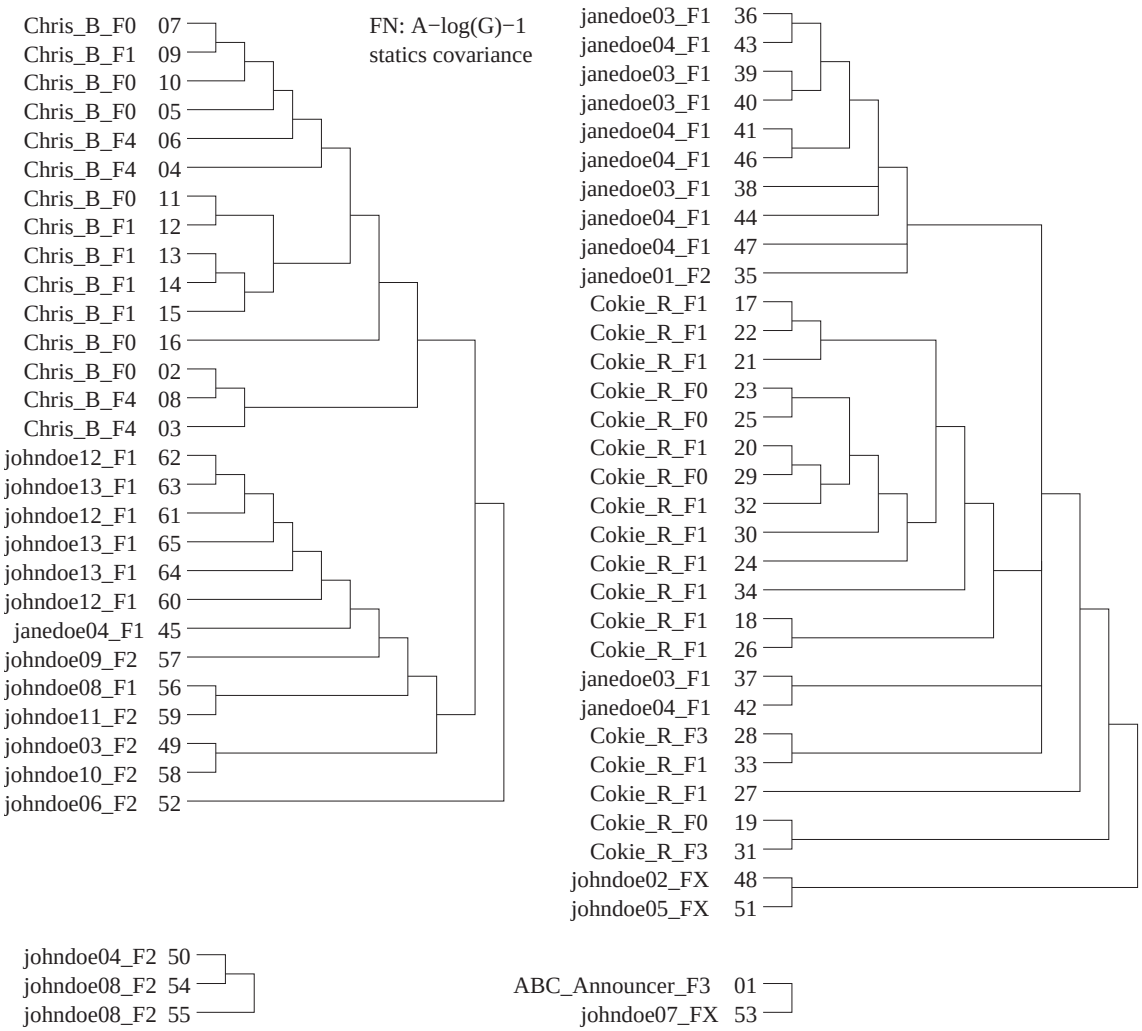

Figure 17: Clusters formed from Covariance Matrix, Furthest Neighbour and d3

6 Improving the System

6.1 Symmetrising A - Using AA'

The MLLR transform matrix is not symmetric. An option, `-y` was added to the `gendist` function to allow this A matrix to be symmetrised before generating the distance matrix. The allowable substitutions were $(A + A^T)$, (AA^T) and $(A^T A)$.

Since most of the distance metrics implemented for this system work in an elementwise fashion, $(A + A^T)$ was not tried as it would duplicate previous results. (AA^T) was tried however, as this represents the result of the transforming an identity covariance matrix.

$$\Sigma_y = AIA^T$$

A summary of the results is given in table 15 and illustrated in figure 18. More detailed results can be found in appendix F. These results indicate that performance can indeed be improved slightly by replacing A by AA^T before calculating the distance matrix.

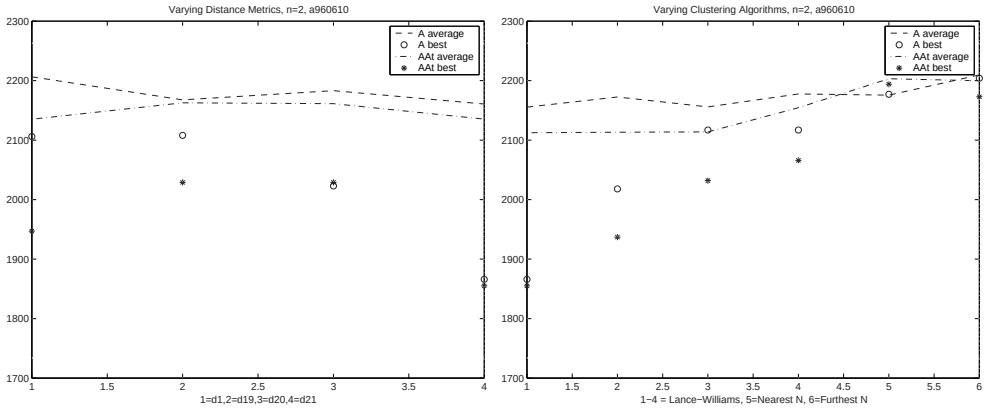

Figure 18: Results for 2-cluster case on a960610 show using A and AA'

	A-matrix		AA^T matrix	
Coefficient	**Average**	**Best Score**	**Average**	**Best Score**
Statics alone (S)	2169.92	1866 (d21,LW1)	2139.01	1855 (d21,LW1)
Statics and Deltas (SD)	2186.59	2106 (d1,LW1)	2151.88	1947 (d1,LW1)
Statics, Deltas and Accels (SDA)	2181.84	2018 (d19,LW2)	2156.09	1935 (d21,LW1)

Clustering Type	**Average**	**Best Score**	**Average**	**Best Score**
Lance Williams, β=-0.5 (LW1)	2155.50	1866 (d21,S)	2112.42	1855 (d21,S)
Lance Williams, β=-0.25 (LW2)	2172.50	2018 (d19,SDA)	2113.25	1937 (d21,SDA)
Lance Williams, Nearest Neigh. (LW4)	2155.84	2117 (many)	2113.92	2032 (d20,SDA)
Lance Williams, Furthest Neigh. (LW8)	2177.67	2117 (many)	2151.42	2066 (many)
Nearest Neighbour (NN)	2175.58	2177 (d1,SDA)	2203.17	2194 (d1,S)
Furthest Neighbour (FN)	2209.58	2204 (many)	2199.75	2173 (d1,S)

Distance Metric	**Average**	**Best Score**	**Average**	**Best Score**
log(A/H) (d1)	2206.39	2106(LW1,SD)	2135.11	1947 (LW1,SD)
Euclidean Elementwise (d19)	2167.61	2018 (LW2,SDA)	2162.67	2029 (many)
City Elementwise (d20)	2183.22	2023 (LW2,S)	2161.22	2029 (LW2,SD)
Angular Elementwise (d21)	2160.56	1866 (LW1,S)	2135.28	1855 (LW1,S)

Table 15: Using AA^T on the 2-cluster problem

6.2 Symmetrising the Distance Measures

It was noted in table 5 that although the distance measures d2 (log(A/G)), d3 (A-logG-1), d15 (log(AA/GH)) and d16 (A-logH-G) work well on classifying speakers, they are not symmetric. This means that the order in which the segments are presented to the system will affect the results. This is not a desirable property. Furthermore, it has been shown [2] that symmetrisation of these distance metrics increases performance on a speaker-identification task.

Several methods of symmetrising the distances are possible. In the linear domain, they use the formula

$$d_{sym}(X,Y) = \rho_{mn} d(X,Y) + \rho_{nm} d(Y,X)$$

$$\rho_{mn} + \rho_{nm} = 1$$

The simplest case is to average the values from the alternate orderings by using

$$\rho_{mn} = \rho_{nm} = 0.5$$

For example, when applied to the Arithmetic-Geometric mean this produces a value equivalent to the Arithmetic-Harmonic mean

$$d_{AH}(X,Y) = log(A/H) = log(A/G) + log(G/H) = d_{AG}(X,Y) + d_{AG}(Y,X)$$

More sophisticatedly, the values of ρ can be used to represent the weighting, or associated confidence of the information in that segment. Such a scheme can use

$$\rho_{mn} = \frac{M}{M+N} \qquad \text{scheme a}$$

or

$$\rho_{mn} = \frac{\sqrt{M}}{\sqrt{M}+\sqrt{N}} \qquad \text{scheme b}$$

Bimbot and Mathan report more success with scheme b than scheme a which in turn was more successful than the simple averaging case. [2] They also report more efficient symmetrisation by using

$$d_{sym}(X,Y) = [d(X,Y)]^{\rho_{mn}} [d(Y,X)]^{\rho_{nm}}$$

This is roughly equivalent to performing the previous symmetrisation in the log domain.

A `-x` option was added to `gendistcov` to allow this symmetrising to be used when calculating the distance matrix. A summary of the results are given in table 16 and illustrated in figure 19. The full results are given in appendix G.

These results show that although a small gain in performance is obtained for the case of log(AA/GH) and A-logG-H, the effect is not very significant, and symmetrisation of the distance metrics was not thought to offer significant improvement over the original implementation.

Distance Metric	**Average**					**Best Score**
	Original	lina	linb	loga	logb	
Divergence (d18)	2137.67					2010 (LW2)
log(A/H) (d1)	2132.72					1931 (LW2,SD)
log(A/G) (d2)	2136.83	2133.0	2150.1	2144.7	2134.6	1912 (LW2,SD,nosym)
A-logG-1 (d3)	2105.50	2157.8	2155.0	2117.5	2163.3	1885 (LW1,SDA,nosym)
log(AA/GH) (d15)	2129.28	2111.4	2124.6	2106.5	2110.9	1902 (LW1,SDA,allsym)
A-logG-H (d16)	2144.06	2136.7	2154.3	2122.2	2116.3	1890 (LW2,SD,loga)
Bhat (d17)	2142.12					1956 (LW1,S)

Table 16: Results for Symmetrising the Distance Measures with 2 clusters

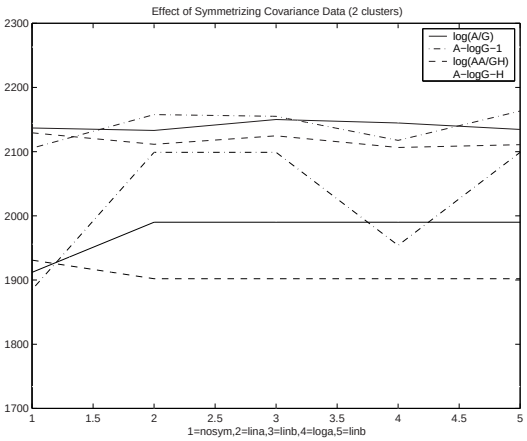

Figure 19: Results of Symmetrising Distance metrics

6.3 Adding Occupancy Counts

In the case where there is one extreme outlier in the data, many of the clustering algorithms would retain this outlier as an individual and group all the other segments into one large cluster. Although this may still technically form the "best" clusters, it obviously does not give good speaker identification, and is unlikely to offer an auxiliary-function FOM much greater than the case of a single cluster.

To overcome this problem, it is possible to use an *occupancy count* to set a minimum size of each cluster. This has the added advantage of ensuring that each cluster has sufficient data to enable satisfactory MLLR transforming subsequently.

The user specifies the maximum number of clusters which are acceptable. From this base level, clustering is continued until all clusters have reached the minimum occupancy count. This means that the exact number of clusters is not known in advance. This is also potentially an advantage since the number of speakers in the soundtrack will in general be unknown.

If the same standard clustering procedure were used throughout, then for the case of one outlier whose size was less than the minimum occupancy count, one large cluster would be formed. This is obviously undesirable. To prevent this happening the way of combining clusters should be changed once the base number of clusters has been reached. Rather than joining the closest groups, the tactic switches to combining the smallest cluster to its closest neighbour. This ensures the minimum occupancy count on all clusters will be reached relatively quickly. A flowchart explaining this procedure is given in figure 20.

6.4 Non-hierarchical methods

All the clustering schemes discussed in this report are hierarchical. This means that exactly two groups are joined on every step. The advantage of this method is that once the dendogram has been generated, the partioning of an arbitrary number of clusters, n, can be extracted with no further work. Also, by setting a threshold on the value used to combine clusters, a stopping criterion can be implemented for the case where the number of speakers in the sound-track is unknown.

Hierarchical methods however, suffer from the drawback that the groupings made are only locally optimal and the decisions are irreversible so once an incorrect decision has been made, a full recovery is never possible.

An alternative to this approach is to use non-hierarchical methods, which specify the desired

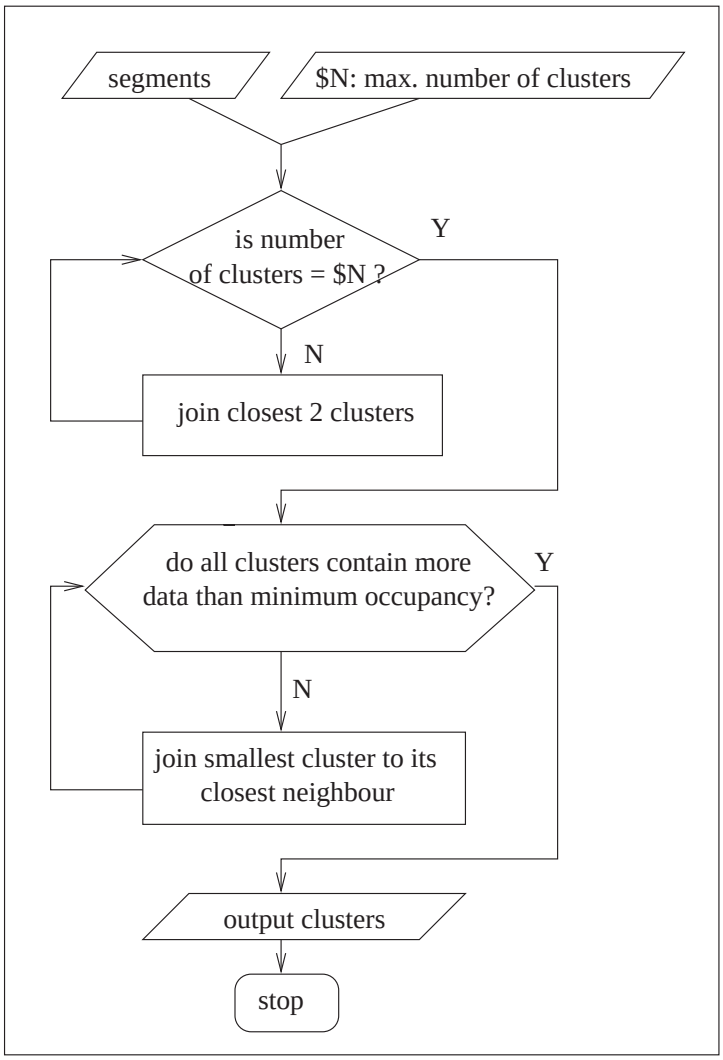

Figure 20: Implementing a Minimum Occupancy Count

numbers of clusters beforehand. The clustering process then must be re-calculated for each distinct value of n, making it significantly more computationally expensive for the case when the number of desired clusters is initially unknown, but the problem of irreversible local decisions is irradicated.

One example of such an approach is K-means clustering [5, p218], where the desired number of clusters are formed (for example at random initially) and subsequently the segments move to the clusters with the closest centroid until a stable solution is reached. Whilst this is not guaranteed to be a global maximum, it does not make hard local decisions at an early stage which cannot be undone.

6.5 Maximisation of Auxiliary Function Directly

Further improvements could be made if the auxiliary function could be maximised directly. One potential method of doing this is to cluster the data initially as before, generating the MLLR transform for each cluster. The log likelihood of each segment is then calculated after the application of each cluster transform in turn. The segment is then allocated to the cluster which maximises this likelihood. This guarantees the auxiliary function will either increase or stay constant during this step.

This process is repeated for all segments. If any of the segments have changed cluster, the new MLLR transform is calculated, again guaranteeing a monotonic increase in the auxiliary function, and the process is repeated. [15] This algorithm is illustrated in figure 21.

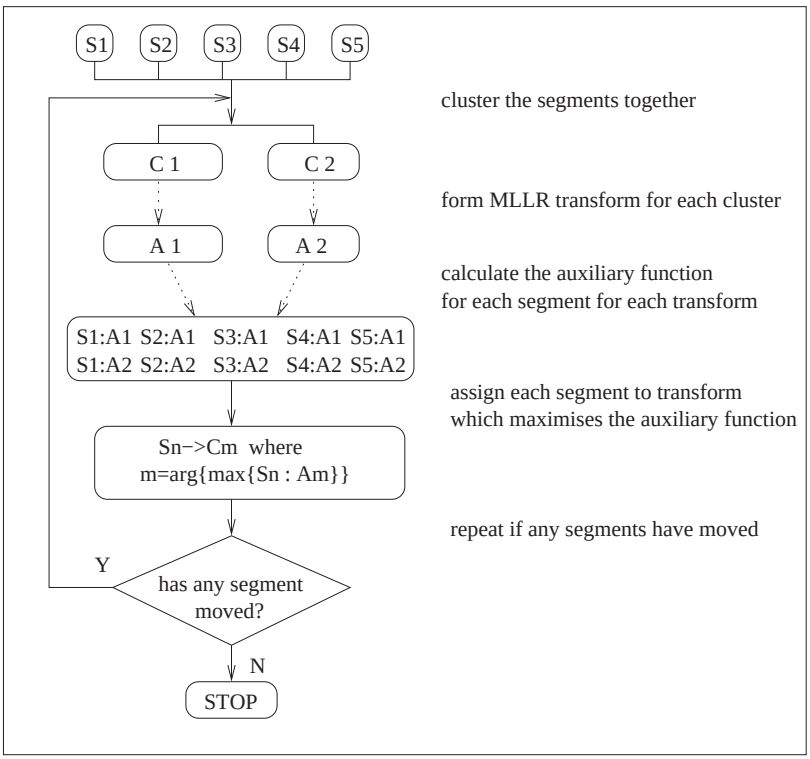

Figure 21: Direct Maximisation of the Auxiliary Function

Unfortunately time constraints do not permit this algorithm to be implemented, but it is hoped significant increase in performance could be made with this method.

[15] This algorithm was suggested by my supervisor, Phil Woodland

7 Conclusions

This project set out to investigate speaker tracking. It was shown that this could be broken down into two problems, namely segmentation and labelling. The latter was the focus of this work since given a successful labelling scheme, the segmentation problem is solvable by using a sliding window over the data.

The labelling problem was shown to break down into three main areas. Firstly a mathematical representation of each segment must be found which holds sufficient information to be able to identify the speaker of the segment. Secondly some measure of "closeness" between these representations must be defined. This is done by specifying a *distance metric* between the representations of the segments. Thirdly a method of joining segments which are in some sense closer to each other than to the remaining ones must be given. This is done with the implementation of a *clustering* algorithm.

A method of evaluating the success of the speaker tracking system has to be defined. Initially experiments were run on a speaker identification problem. This assigned one segment in three to be a *test* segment and the remainder to be *references*. Performance was then evaluated by noting the proportion of the test set which was correctly identified.

Section 2 showed that the covariance matrix of the data in the segment was a sufficient mathematical representation of the speaker to allow correct identification, with over 90% success in the case of the Gaussian divergence, the geometric-harmonic sphericity and the arithmetic-harmonic sphericity distance metrics. By noting trends in the results a further distance metric, $\log(A^2/\mathrm{GH})$, was defined, which gave a 12% reduction in mis-classification rate.

Section 3 used the MLLR transform matrix as the basic segment representation. This again proved to offer recognition rates of over 90% using the data-based transform for the case of element-based distance metrics.

The ability to reduce the speaker-specific information in the data by applying the data-based MLLR transform to the covariance of the segments was noted and the inverse-transform was applied in an attempt to exaggerate speaker-dependent properties. This had little effect on the results, giving the same maximum recognition rate as the un-transformed case.

Section 4 described several hierarchical clustering schemes. These methods either join (*agglomerative*) or split (*divisive*) segments in an irreversible binary fashion. Whilst not necessarily providing as good a performance as a fixed-number non-hierarchical scheme such as K-means, they only have to be run once and allow a stopping criterion to be implemented for the case of an unknown number of speakers in the soundtrack.

Several experiments were run with different numbers of segments with two and three speakers on broadcast news data. These confirmed the ability of the methods to form clusters which represented the speakers in the soundtrack.

Section 5 described the implementation of the clustering system on a larger scale, making use of all the results from the previous sections. A new measure of performance was defined based on the likelihood of the data after MLLR transformation given a particular alignment. This was shown to be a good indication of speaker-split.

Several experiments were run with the different representations, different distance metrics and different algorithms on the `a960610` show of the 1996 Broadcast News database. These showed the best performance was obtained with the discriminative Lance-Williams scheme, with furthest neighbour clustering outperforming nearest neighbour due to the problems of *chaining* with the latter.

Tree diagrams for the best case of three and four clusters were presented showing the ordering of clustering. These clearly indicated the difference in strategy between Lance-Williams and fur-

thest neighbour clustering. The former tends to grow clusters into *plateaus* which can be used to identify speaker changes, whereas the latter tends to group them in a more binary fashion. The speaker groups could clearly be seen in both cases.

Some improvements to the system were presented in section 6. The symmetrisation of the MLLR matrix was found to improve performance, whereas symmetrising the distance metrics with covariance data did not make a significant difference.

More improvements were suggested which are left as further work to do on this project. These include adding a minimum occupancy count and direct maximisation of the auxiliary function figure of merit. Incorporating the idea of non-hierarchical schemes in making the clustering decisions reversible by allowing them to move around, could also improve performance.

In summary, this project has demonstrated the validity of using covariance and MLLR matrices as speaker-specific representations for the segments, shown the success of many clustering methods on a small scale, implemented a large-scale clustering program which has been shown to produce good speaker splits, defined a new figure of merit for evaluating clusters and suggested ways of improving the overall performance of the system.

A Distance Measures Used

$$
\begin{aligned}
d1 &= \log\left(\frac{A}{H}\right) \\
d2 &= \log\left(\frac{A}{G}\right) \\
d3 &= A - \log(G) - 1 \\
d4 &= \frac{1}{D}\sum_{i=1}^{D}|\lambda_i - 1| \\
d5 &= \frac{1}{D}\sum_{i=1}^{D}|\min(\lambda_i, \frac{1}{\lambda_i}) - 1| \\
d6 &= \max_{i=1:D}|\max(\lambda_i, \frac{1}{\lambda_i}) - 1| \\
d7 &= \min_{i=1:D}|\min(\lambda_i, \frac{1}{\lambda_i}) - 1| \\
d8 &= \frac{1}{D}\sum_{i=1}^{D}(\lambda_i - \frac{1}{\lambda_i})^2 \\
d9 &= \left(\prod_{i=1}^{D}|\max(\lambda_i, \frac{1}{\lambda_i}) - 1|\right)^{1/D} \\
d10 &= \left(\prod_{i=1}^{D}|\min(\lambda_i, \frac{1}{\lambda_i}) - 1|\right)^{1/D} \\
d11 &= \frac{1}{D}\sum_{i=1}^{D}\lambda_i = A \\
d12 &= \sqrt[D]{\prod_{i=1}^{D}\lambda_i} = G \\
d13 &= D\left(\sum_{i=1}^{D}\frac{1}{\lambda_i}\right)^{-1} = H \\
d14 &= \log\left(\frac{A}{GH}\right) \\
d15 &= \log\left(\frac{A^2}{GH}\right) \\
d16 &= A - \log(G) - H \\
d17 &= \frac{1}{8}(\mu_x - \mu_y)^T\left(\frac{X+Y}{2}\right)^{-1}(\mu_x - \mu_y) + \frac{1}{2}\ln\left(\frac{|(X+Y)/2|}{|X|^{\frac{1}{2}}|Y|^{\frac{1}{2}}}\right) \\
d18 &= \frac{1}{2}tr(X^{-1}Y + Y^{-1}X - 2I) + \frac{1}{2}(\mu_x - \mu_y)^T(X^{-1} + Y^{-1})(\mu_x - \mu_y) \\
d19 &= \sum_{i=1}^{D}\sum_{j=1}^{D}(x_{ij} - y_{ij})^2 \\
d20 &= \sum_{i=1}^{D}\sum_{j=1}^{D}|x_{ij} - y_{ij}| \\
d21 &= 1 - \frac{\sum_{i=1}^{D}\sum_{j=1}^{D}x_{ij}y_{ij}}{\left(\sum_{i=1}^{D}\sum_{j=1}^{D}x_{ij}^2\sum_{i=1}^{D}\sum_{j=1}^{D}y_{ij}^2\right)^{1/2}} \\
d22 &= \sigma^+(Y - X)
\end{aligned}
$$

$$d23 \quad = \quad \frac{1}{D}\sum_{i=1}^{D}(\mu_{yi} - \mu_{xi})^2$$

B Initial Results in Speaker Clustering

B.1 Experiment 1: Simple 2-Speaker Case

The segments for this experiment are:

```
1 2 3   Ted_Koppel_F0
4 5 6   a960521_janedoe001_F2
```

Statistical Merging

	Concatenation		Centroid		Median	
Merge	min	2ndmin	min	2ndmin	min	2ndmin
13	0.22	0.26	0.22	0.26	0.22	0.26
123	0.23	0.47	0.23	0.47	0.23	0.47
45	0.47	0.50	0.47	0.50	0.47	0.50
456	0.43	0.63	0.42	0.62	0.41	0.64
123456	0.55		0.55		0.58	

Neighbourhood Schemes

Nearest		Furthest		Mutual		Group Average		
merge	min	merge	min	merge	min	merge	min	2ndmin
13	0.22	13	0.22	13	2	13	0.22	0.26
123	0.26	123	0.33	45	2	123	0.30	0.47
45	0.47	45	0.47	123	3	45	0.47	0.50
456	0.50	456	0.54	456	3	456	0.52	0.79
123456	0.70	123456	0.96	123456	6	123456	0.82	

Ward's method

2-D on means			3-D on covariances		
merge	min	2ndmin	merge	min	2ndmin
13	0.25	0.33	12	11.00	13.45
123	0.77	1.40	123	28.36	42.10
45	1.92	2.44	56	59.46	67.66
456	4.25	13.49	456	105.09	176.87
123456	20.35		123456	243.10	

Lance-Williams' Recurrence

Nearest Neighbour			Furthest Neighbour			$\beta = -0.25$			$\beta = -0.5$		
merge	max	2ndmax	merge	max	2ndmax	merge	max	2ndmax	merge	max	2ndmax
45	2.88	2.85	45	2.86	2.85	45	3.30	3.26	13	3.65	3.59
456	2.35	1.80	456	2.35	1.80	456	2.81	2.33	123	3.86	3.03
23	1.01	0.96	23	1.01	0.96	13	1.42	1.39	45	2.28	2.23
123	0.70	0.22	123	0.70	0.22	123	1.21	0.65	456	1.95	1.09

Polythetic divisive clustering

```
>> [group1,group2]=split_cluster(dist_ahm)
group1=
     5     4     6
group2=
     1     2     3
```

B.2 Experiment 2: Adding More Segments

The segments for this experiment are:

	F0	FX	F2	F4
Ted_Koppel	1 2 3	4	5	6
a960521_janedoe001		10	7 8 9	

The following classifications were the only ones which clustered the segments incorrectly. (Only the smallest cluster is given for ease of readability).

Distance	d18	d17	d1	d17	d3	d15
Lance Will. $\beta = -0.5$	1 2 3 4 5		5	2 3 5 6	3 5	2 3 5 6
Lance Will. $\beta = -0.25$	5	4	5	5		5
LW. Nearest Neigh	1 2 3 6	1 2 3 6	1 2 3 6	1 2 3 6	1 2 3	1 2 3 6
LW. Furthest Neigh	5 10	5	3 9	2 3	7 9	3 9

B.3 Experiment 3 : 3-Speaker Problem

The segments for this experiment are:

```
1 2 3                  Ted_Koppel_F0
4 5 6 7                Ted_Koppel_FX
8                      Ted_Koppel_F2
9                      Ted_Koppel_F4
10 11 12               a960521_janedoe001_F2
13                     a960521_janedoe001_FX
14 15 16 17 18 19 20   a960521_johndoe007_FX
```

Each clustering method is traced back to the point where an incorrect decision was made.

B.3.1 Statistical Merging Techniques

	Concatenation, Centroid and Median Clustering
two groups	1 2 3 4 5 6 7 8 9 14 15 16 17 18 19 20/10 11 12 13
three groups	1 2 3 4 5 6 7 8 9 14 15 16 17 18 19 20/10 12 13/11
four groups	1 2 3 4 5 6 7 8 9 14 15 16 17 18 19 20/ 10 13/11/12
five groups	1 2 3 4 5 6 7 8 9/14 15 16 17 18 19 20/10 13/11/12

B.3.2 Neighbourhood Methods

	Nearest, Furthest, Mutual and Group Average
two groups	1 2 3 4 5 6 7 8 9 14 15 16 17 18 19 20/10 11 12 13
three groups	1 2 3 4 5 6 7 8 9/14 15 16 17 18 19 20/10 11 12 13

B.3.3 Wards Method

	Ward 2D (means)
two groups	1 2 3 5 7 8 9 10 11 12 13 14 15 16 17 18 19 20/4 6
three groups	1 2 3 5 7 8 9 10 11/12 13 14 15 16 17 18 20/4 6
...	
eight groups	1 2 3 5 7 8 9/14 15 16 17 18 19/4 6/10/11/12/13/20

	Ward 3D (covariance)
two groups	1 2 3 4 5 6 7 9 /8 10 11 12 13 14 15 16 17 18 20
three groups	1 2 3 4 5 6 7 9/10 11 12 13 14 15 16 17 18 19 20/8
...	
six	1 2 3 4 5 6 7 9/8/10 11 12 13/14 15 16 18/17 20/19

B.3.4 Lance Williams Recurrence

	Lance Williams, $\beta = -0.5$
two groups	1 2 3 4 5 6 7 8 9 14 15 16 17 18 19 20/10 11 12 13
three groups	1 2 3 4 5 6 7 8 9 14 15 16 17 18 20/10 11 12 13/19
...	
ten groups	1 2 3 4 5 6 7 8 9/10 11 12 13/19/16/14/15/17/18/20

	Lance Williams, $\beta = -0.25$
two groups	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 20/19
three groups	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17 18 20/19/16
...	
seventeen	10 11 12 13/19/16/14/15/18/17/20/2/4/6/7/5/9/1/3/8

	Lance Williams, nearest neighbour
two groups	1 2 3 4 5 6 7 8 9 10 11 12 13 16 18 19 20/14 15 17
three groups	1 2 3 4 5 6 7 8 9 10 11 12 13 16 18 19 20/14 15/17
...	
seventeen	10 11 12 13/14/15/17/18/16/19/20/2/1/5/4/6/7/9/3/8

	Lance Williams, furthest neighbour
two groups	1 2 3 4 5 6 7 8 9 10 11 13 14 15 17 18 19 20/12 16
three groups	1 2 3 4 5 6 7 8 9 10 11 13 14 15 17 18 19 20/12/16
...	
first merge	8 11

B.3.5 Polythetic Divisive Clustering

```
>> [splinter,rest]=split_cluster(dist_ahm)
splinter =
    11  10  13  12          % these are janedoe001

for i=1:length(rest)        % make new distance metric of
  for j=1:length(rest)      % largest cluster
    new_dist(i,j)=dist_ahm(rest(i),rest(j));
  end
end

>> [splinter,rest]=split_cluster(new_dist)
splinter =
     8   1   3   9   5   2   7   4   6 % Ted_Koppel

rest =
    10  11  12  13  14  15  16        % johndoe007
```

C Results for Making 2 clusters on a960610 show

C.1 Results on Covariance Data

STATICS ALONE

clustering type	Div.	log(A/H)	log(A/G)	A-logG-1	log(AA/GH)	A-logG-H	Bhat.	**FOM**
LanceW 1	2172	2127	2087	2052	2209	2093	1956	**2099.43**
LanceW 2	2140	2107	2107	2028	2107	2195	2117	**2114.43**
LanceW 4	2066	2099	2099	2147	2099	2099	2178	**2112.43**
LanceW 8	2156	2099	2099	2147	2099	2099	2178	**2125.29**
NN	2204	2204	2204	2194	2204	2204	2204	**2202.57**
FN	2204	2194	2194	2194	2194	2204	2204	**2198.29**
FOM	**2157.00**	**2138.33**	**2131.67**	**2127.00**	**2152.00**	**2149.00**	**2139.50**	**2142.07**

STATICS AND DELTAS

LanceW 1	2171	2078	2164	2064	2151	2174	2009	**2115.86**
LanceW 2	2010	1931	1912	2009	1931	1998	2217	**2001.14**
LanceW 4	2066	2099	2099	2066	2099	2099	2066	**2084.86**
LanceW 8	2156	2099	2099	2066	2099	2099	2117	**2105.00**
NN	2204	2204	2228	2228	2204	2204	2228	**2214.29**
FN	2204	2194	2194	2194	2194	2204	2204	**2198.29**
FOM	**2135.17**	**2100.83**	**2116.00**	**2104.50**	**2113.00**	**2129.67**	**2140.17**	**2119.91**

STATICS, DELTAS AND ACCELS

LanceW 1	2085	2215	2206	1885	2156	2216	2159	**2131.71**
LanceW 2	2010	1985	2151	2088	1985	2109	1997	**2046.43**
LanceW 4	2066	2178	2099	2099	2099	2099	2066	**2100.86**
LanceW 8	2156	2178	2099	2099	2099	2099	2223	**2136.14**
NN	2204	2204	2228	2145	2204	2204	2228	**2202.43**
FN	2204	2194	2194	2194	2194	2194	2204	**2196.86**
FOM	**2120.83**	**2159.00**	**2162.83**	**2085.00**	**2122.83**	**2153.50**	**2146.17**	**2135.74**

C.2 Results from MLLR transform Matrices

STATICS	**Offset Vector**				**Using Block Diagonal A only**				
clustering type	Euclid	City	Angle	**FOM**	Euclid	City	Angle	AHM	**FOM**
LanceW 1	2121	2237	2137	**2165.00**	2175	2171	1866	2189	**2100.25**
LanceW 2	2195	2090	2122	**2135.67**	2208	2023	2209	2220	**2165.00**
LanceW 4	2153	2122	2191	**2155.33**	2117	2153	2117	2222	**2152.25**
LanceW 8	2153	2152	2191	**2165.33**	2117	2240	2117	2222	**2174.00**
NN	2236	2236	2185	**2219.00**	2204	2204	2204	2239	**2212.75**
FN	2238	2236	2200	**2224.67**	2204	2204	2228	2225	**2215.25**
FOM	**2182.67**	**2178.83**	**2171.00**	**2177.50**	**2170.83**	**2165.83**	**2123.50**	**2219.50**	**2169.92**

STATICS AND DELTAS

LanceW 1	2222	2012	1881	**2038.33**	2221	2204	2176	2106	**2176.75**
LanceW 2	2025	2016	1979	**2006.67**	2230	2152	2208	2218	**2202.00**
LanceW 4	2154	1927	2151	**2077.33**	2117	2172	2117	2232	**2159.50**
LanceW 8	2154	2135	2201	**2163.33**	2117	2205	2117	2232	**2167.75**
NN	2236	2236	2222	**2231.33**	2204	2204	2204	2215	**2206.75**
FN	2236	2236	2145	**2205.67**	2204	2204	2204	2215	**2206.75**
FOM	**2171.17**	**2093.67**	**2096.50**	**2120.45**	**2182.17**	**2190.17**	**2171.00**	**2203.00**	**2186.59**

STATICS, DELTAS AND ACCELS

LanceW 1	2241	2218	2114	**2191.00**	2196	2167	2229	2166	**2189.50**
LanceW 2	2025	2188	1869	**2027.33**	2018	2208	2206	2170	**2150.50**
LanceW 4	2153	1973	2151	**2092.33**	2117	2140	2140	2226	**2155.76**
LanceW 8	2008	1973	2057	**2012.67**	2160	2239	2140	2226	**2191.25**
NN	2236	2236	2222	**2231.33**	2204	2204	2204	2177	**2197.25**
FN	2236	2236	2145	**2205.67**	2204	2204	2204	2215	**2206.75**
FOM	**2149.83**	**2137.33**	**2093.00**	**2126.72**	**2149.83**	**2193.67**	**2187.17**	**2196.67**	**2181.84**

D Results for Making 3 clusters on a960610 show

D.1 Results on Covariance Data

STATICS ALONE

clustering type	Div.	log(A/H)	log(A/G)	A-logG-1	log(AA/GH)	A-logG-H	Bhat.	**FOM**
LanceW 1	1854	2102	2040	1991	2003	1987	1931	**1986.86**
LanceW 2	2096	2025	2025	2014	2025	2025	2072	**2040.29**
LanceW 4	2043	2077	2076	2135	2076	2076	2168	**2093.00**
LanceW 8	2142	2077	2076	2135	2076	2076	2168	**2107.14**
NN	2169	2169	2169	2169	2169	2169	2169	**2169.00**
FN	2144	2169	2100	2100	2104	2109	2144	**2124.29**
FOM	**2074.67**	**2103.17**	**2081.00**	**2075.50**	**2076.33**	**2073.67**	**2108.67**	**2086.76**

STATICS AND DELTAS

LanceW 1	1887	2041	2140	2044	1975	2158	1941	**2026.57**
LanceW 2	1993	1916	1884	1991	1918	1889	2011	**1943.14**
LanceW 4	2043	2076	2076	2043	2076	2076	2043	**2061.86**
LanceW 8	2142	2076	2076	2043	2076	2076	2104	**2084.71**
NN	2169	2114	2116	2116	2114	2114	2169	**2130.29**
FN	2144	2169	2103	2100	2104	2114	2144	**2125.43**
FOM	**2063.00**	**2065.33**	**2065.83**	**2056.17**	**2043.83**	**2071.17**	**2068.67**	**2062.00**

STATICS, DELTAS AND ACCELS

LanceW 1	1942	2110	2160	1864	2111	2198	2008	**2056.14**
LanceW 2	1993	1968	1992	2009	1968	2095	1969	**1999.14**
LanceW 4	2043	2168	2076	2076	2077	2076	2043	**2079.86**
LanceW 8	2142	2168	2076	2076	2076	2076	2211	**2117.86**
NN	2169	2114	2116	2116	2114	2114	2169	**2130.29**
FN	2144	2104	2103	2100	2103	2104	2144	**2114.57**
FOM	**2072.17**	**2105.33**	**2087.17**	**2040.17**	**2074.83**	**2110.50**	**2090.67**	**2082.98**

D.2 Results from MLLR transform Matrices

STATICS	**Offset Vector**				**Using Block Diagonal A only**				
clustering type	Euclid	City	Angle	**FOM**	Euclid	City	Angle	AHM	**FOM**
LanceW 1	1823	2208	2118	**2049.67**	2138	2079	1750	2151	**2029.50**
LanceW 2	2128	2058	2101	**2095.67**	2129	2007	2129	2194	**2114.75**
LanceW 4	2107	2101	2171	**2126.33**	2104	2141	2104	2197	**2136.50**
LanceW 8	2107	2107	2171	**2128.33**	2104	2219	2104	2197	**2156.00**
NN	2211	2211	2153	**2191.67**	2169	2169	2169	2191	**2174.50**
FN	2211	2211	2159	**2193.67**	2169	2169	2138	2201	**2169.25**
FOM	**2097.83**	**2149.33**	**2145.5**	**2130.89**	**2135.50**	**2130.67**	**2065.67**	**2188.50**	**2130.09**

STATICS AND DELTAS

LanceW 1	1929	1923	1816	**1889.33**	2111	2014	2149	2093	**2091.75**
LanceW 2	2004	1973	1958	**1978.33**	2035	2079	2172	2110	**2099.00**
LanceW 4	2107	1913	2136	**2052.00**	2104	2160	2104	2203	**2142.75**
LanceW 8	2107	2121	2188	**2138.67**	2104	2196	2104	2203	**2151.75**
NN	2211	2211	2187	**2203.00**	2169	2169	2169	2126	**2158.25**
FN	2211	2211	2104	**2175.33**	2169	2114	2114	2126	**2130.75**
FOM	**2094.83**	**2058.67**	**2064.83**	**2072.78**	**2115.33**	**2122.00**	**2135.33**	**2143.50**	**2129.04**

STATICS, DELTAS AND ACCELS

LanceW 1	2213	2096	2090	**2133.00**	2159	2113	2096	2114	**2120.50**
LanceW 2	2004	2135	1851	**1996.67**	2002	2066	2113	2138	**2079.75**
LanceW 4	2076	1961	2136	**2057.67**	2104	2126	2126	2197	**2138.25**
LanceW 8	1992	1961	2037	**1996.67**	2147	2198	2126	2197	**2167.00**
NN	2211	2211	2127	**2183.00**	2169	2144	2169	2126	**2152.00**
FN	2211	2211	2103	**2175.00**	2114	2114	2114	2126	**2117.00**
FOM	**2117.83**	**2095.83**	**2057.33**	**2090.33**	**2115.83**	**2126.83**	**2196.67**	**2149.67**	**2129.08**

E Results for Making 4 clusters on a960610 show

E.1 Results on Covariance Data

STATICS ALONE

clustering type	Div.	log(A/H)	log(A/G)	A-logG-1	log(AA/GH)	A-logG-H	Bhat.	**FOM**
LanceW 1	1842	1964	2017	1971	1990	1964	1910	**1951.14**
LanceW 2	2058	2012	2011	1964	2011	2012	2058	**2018.00**
LanceW 4	2033	2066	2066	2125	2066	2066	2033	**2065.00**
LanceW 8	2090	2066	2066	2125	2066	2066	2066	**2077.86**
NN	2109	2079	2078	2079	2079	2079	2109	**2087.43**
FN	2109	2075	2008	1590	2079	2075	2109	**2006.43**
FOM	**2040.17**	**2043.67**	**2041.00**	**1975.67**	**2048.50**	**2043.67**	**2047.50**	**2034.31**

STATICS AND DELTAS

LanceW 1	1844	2018	2121	1730	1960	2144	1921	**1962.57**
LanceW 2	1979	1885	1864	1978	1903	1874	1992	**1925.00**
LanceW 4	2033	2066	2066	2033	2066	2066	2033	**2051.86**
LanceW 8	2090	2066	2066	2033	2066	2066	2090	**2068.14**
NN	2109	2079	2079	2079	2079	2079	2109	**2087.57**
FN	2109	2075	2007	2059	2079	2054	2109	**2070.29**
FOM	**2027.33**	**2031.50**	**2033.83**	**1985.33**	**2025.50**	**2047.17**	**2042.33**	**2027.57**

STATICS, DELTAS AND ACCELS

LanceW 1	1918	2082	2146	1845	2089	2175	1967	**2031.71**
LanceW 2	1979	1954	1951	1995	1954	1948	1955	**1962.29**
LanceW 4	2033	2066	2066	2066	2066	2066	2033	**2056.57**
LanceW 8	2090	2066	2066	2066	2066	2066	2182	**2086.00**
NN	2109	2079	2079	2079	2079	2079	2109	**2087.57**
FN	2109	2008	2008	2008	2008	2079	2109	**2047.00**
FOM	**2039.67**	**2042.50**	**2052.67**	**2009.83**	**2043.67**	**2068.83**	**2059.17**	**2045.19**

E.2 Results from MLLR transform Matrices

STATICS	**Offset Vector**				**Using Block Diagonal A only**				
clustering type	Euclid	City	Angle	**FOM**	Euclid	City	Angle	AHM	**FOM**
LanceW 1	1629	2082	2092	**1934.33**	2121	2059	1625	2101	**1976.52**
LanceW 2	2038	1955	2087	**2026.67**	2098	1995	2098	2174	**2091.25**
LanceW 4	2094	2087	2155	**2112.00**	2090	2130	2090	2166	**2119.00**
LanceW 8	2094	2094	2155	**2114.33**	2090	2202	2090	2166	**2137.00**
NN	2188	2188	2128	**2168.00**	2079	2079	2079	2100	**2084.25**
FN	2175	2188	2064	**2142.33**	2079	2079	2079	2172	**2102.25**
FOM	**2036.33**	**2099.00**	**2113.50**	**2082.94**	**2092.83**	**2090.67**	**2010.17**	**2146.50**	**2085.04**

STATICS AND DELTAS

LanceW 1	1902	1902	1794	**1866.00**	2095	1961	2100	2062	**2054.50**
LanceW 2	1979	1868	1921	**1922.67**	2017	2007	1995	2088	**2026.75**
LanceW 4	2017	1899	2122	**2012.67**	2090	2000	2090	2168	**2087.00**
LanceW 8	2016	2039	2174	**2076.33**	2090	2179	2090	2169	**2132.00**
NN	2188	2188	2128	**2168.00**	2109	2109	2079	2075	**2093.00**
FN	2142	2187	1651	**1993.33**	2079	2079	2079	2061	**2074.50**
FOM	**2040.67**	**2013.83**	**1965.00**	**2006.50**	**2080.00**	**2055.83**	**2072.17**	**2103.83**	**2077.96**

STATICS, DELTAS AND ACCELS

LanceW 1	2195	1841	2059	**2031.67**	2111	2081	2061	2083	**2084.00**
LanceW 2	1979	2119	1836	**1978.00**	1988	2052	2039	2112	**2047.75**
LanceW 4	2065	1942	2122	**2043.00**	2090	2058	2090	2168	**2101.75**
LanceW 8	1978	1942	2022	**1980.67**	2137	2135	2090	2168	**2132.50**
NN	2188	2188	2097	**2157.67**	2109	2109	2079	2075	**2093.00**
FN	2142	2188	2043	**2124.33**	2079	2079	2079	2079	**2079.00**
FOM	**2091.17**	**2036.67**	**2029.83**	**2052.56**	**2085.67**	**2085.67**	**2073.00**	**2114.17**	**2089.63**

F Results from using AA^T with 2 clusters

STATICS ALONE

	Distance				
clustering type	Euclid	City	Angle	AHM	FOM
LanceW 1	2029	2041	1855	2231	2046.50
LanceW 2	2091	2102	2230	2041	2116.00
LanceW 4	2160	2197	2134	2066	2139.25
LanceW 8	2153	2195	2134	2066	2137.00
NN	2204	2204	2204	2194	2201.50
FN	2204	2204	2194	2173	2193.75
FOM	2140.17	2157.17	2125.17	2128.50	2139.01

STATICS AND DELTAS

	Distance				
clustering type	Euclid	City	Angle	AHM	FOM
LanceW 1	2236	2183	2206	1947	2143.00
LanceW 2	2029	2029	2230	2017	2076.25
LanceW 4	2066	2122	2140	2178	2126.50
LanceW 8	2117	2205	2140	2178	2160.00
NN	2204	2204	2204	2204	2204.00
FN	2204	2204	2204	2194	2201.50
FOM	2142.67	2157.83	2187.33	2119.67	2151.88

STATICS, DELTAS AND ACCELS

	Distance				
clustering type	Euclid	City	Angle	AHM	FOM
LanceW 1	2231	2231	1935	2194	2147.75
LanceW 2	2298	2146	1937	2209	2147.50
LanceW 4	2066	2032	2140	2066	2076.00
LanceW 8	2228	2195	2140	2066	2157.25
NN	2204	2204	2204	2204	2204.00
FN	2204	2204	2204	2204	2204.00
FOM	2205.17	2168.67	2093.33	2157.17	2156.09

STATICS ALONE

	Distance																FOM
	log(A/G)				A-logG-1				log(AA/GH)				A-logG-H				
	lina	linb	loga	logb	lina	linb	loga	logb	lina	linb	loga	logb	lina	linb	loga	logb	
LW 1	2089	2089	2209	2209	2204	2209	1993	2204	2209	2209	2127	2127	2063	1891	2148	1891	2116.9
LW 2	2107	2107	2107	2107	2107	2107	2107	2107	2107	2107	2107	2107	1889	2105	1989	2147	2088.4
LW 4	2099	2178	2099	2099	2099	2099	2178	2148	2099	2099	2099	2099	2148	2155	2148	2095	2121.3
LW 8	2099	2178	2099	2099	2130	2099	2099	2148	2099	2099	2099	2099	2148	2155	2148	2093	2118.2
NN	2204	2204	2204	2204	2204	2204	2204	2204	2204	2204	2204	2204	2194	2194	2194	2194	2201.5
FN	2204	2204	2194	2194	2204	2204	2194	2194	2194	2194	2194	2194	2194	2194	2194	2194	2196.5
FOM	2133.7	2160.0	2155.3	2152.0	2158.0	2153.7	2129.2	2167.5	2152.0	2152.0	2138.3	2138.3	2106.0	2115.7	2136.8	2102.3	2140.7

STATICS AND DELTAS

	Distance																FOM
	log(A/G)				A-logG-1)				log(AA/GH)				A-logG-H)				
	lina	linb	loga	logb	lina	linb	loga	logb	lina	linb	loga	logb	lina	linb	loga	logb	
LW 1	2215	2150	2078	2078	2164	2192	2003	2152	2078	2078	2078	2078	2128	2142	2128	2117	2116.2
LW 2	2026	2027	2026	2026	2144	2081	2009	2208	1937	1937	1931	1931	2098	2079	1890	2028	2023.6
LW 4	2099	2178	2099	2099	2099	2099	2099	2099	2099	2178	2099	2099	2140	2160	2066	2086	2112.4
LW 8	2099	2178	2099	2099	2099	2099	2130	2099	2099	2178	2099	2099	2140	2160	2066	2086	2114.3
NN	2204	2204	2204	2204	2204	2204	2204	2204	2204	2204	2204	2204	2194	2194	2194	2194	2201.5
FN	2204	2204	2204	2194	2204	2204	2194	2194	2194	2194	2194	2194	2194	2194	2194	2194	2197.1
FOM	2141.2	2156.8	2118.3	2116.7	2152.3	2146.5	2106.5	2159.3	2101.8	2128.2	2100.8	2100.8	2149.0	2154.8	2089.7	2117.5	2127.5

STATICS DELTAS AND ACCELS

	Distance																FOM
	log(A/G)				A-logG-1				log(AA/GH)				A-logG-H)				
	lina	linb	loga	logb	lina	linb	loga	logb	lina	linb	loga	logb	lina	linb	loga	logb	
LW 1	2159	2215	2219	2225	2231	2231	1954	2231	1902	1902	1902	1902	2232	2232	2094	2135	2110.4
LW 2	1990	1990	1990	1990	2151	2161	2151	2151	1985	1985	1985	1985	2179	2179	2209	2103	2074.0
LW 4	2099	2099	2178	2099	2099	2099	2099	2099	2099	2099	2099	2099	2066	2178	2099	2099	2106.8
LW 8	2099	2099	2178	2099	2099	2099	2099	2099	2099	2178	2099	2178	2066	2178	2099	2099	2116.7
NN	2204	2204	2204	2204	2204	2204	2204	2204	2204	2204	2204	2204	2194	2194	2145	2145	2195.4
FN	2194	2194	2194	2194	2194	2194	2194	2194	2194	2194	2194	2194	2194	2194	2194	2194	2194
FOM	2124.2	2133.5	2160.5	2135.2	2163.0	2164.7	2116.8	2163.0	2080.5	2093.7	2080.5	2093.7	2155.2	2192.5	2140.0	2129.2	2132.9

H Data Used

The frequency of occurrence of each speaker and condition in all the shows used for this project is given in the tables.

a960521						
speaker name	F0	F1	F2	F3	F4	FX
ABC_NLI_Announcer				1		
Ted_Koppel	12		4	1	4	12
a960521_janedoe001			3			1
a960521_janedoe002						1
a960521_janedoe003						1
a960521_johndoe002			6			
a960521_johndoe003			1			
a960521_johndoe004			1			
a960521_johndoe005			3			1
a960521_johndoe006			1			
a960521_johndoe007						9
a960521_johndoe008		5				
a960521_johndoe009			1			
a960522						
ABC_NLI_Announcer				1		1
Ted_Koppel	8		1	1	14	3
a960522_janedoe003						1
a960522_janedoe004			1			
a960522_janedoe005						1
a960522_janedoe006						1
a960522_janedoe008			1			
a960522_janedoe009						1
a960522_janedoe010			1			
a960522_johndoe001						1
a960522_johndoe003			1			
a960522_johndoe004			1			
a960522_johndoe007			2			1
a960522_johndoe008						2
a960522_johndoe009			1			
a960522_johndoe010			1			
a960522_johndoe011			1			
a960522_johndoe013						1
a960522_johndoe014			1			
a960522_johndoe015			2			
a960522_johndoe016			1			
a960522_johndoe017						1
a960522_johndoe018			1			
a960522_johndoe019						1
a960522_johndoe020			2			
a960522_johndoe021			1			
a960522_johndoe022						1
a960522_johndoe023			1			
a960522_johndoe024			1			1
a960522_johndoe025			1			
a960522_johndoe026						1
a960522_johndoe028						1

a960528						
speaker name	F0	F1	F2	F3	F4	FX
ABC_NLI_Announcer				1		
Dave_Marish	6				7	
Ted_Koppel	6	16			1	
a960528_janedoe002		6				
a960528_johndoe001						1
a960528_johndoe002		1				
a960528_johndoe003		1				
a960528_johndoe004			1			
a960528_johndoe005						1
a960528_johndoe006			1			
a960528_johndoe007			2			
a960528_johndoe008						1
a960528_johndoe009			1			
a960528_johndoe010			2			
a960528_johndoe011		1				
a960528_johndoe012						1
a960528_johndoe014		4				
a960528_johndoe015		8				
a960604						
Michael_Gillen	7				24	1
Ted_Koppel	5	2		3	1	
a960604_anon001				1		
a960604_anon003						1
a960604_anon004			1			5
a960604_anon007						1
a960604_anon008						1
a960604_anon010						1
a960604_anon011			4			7
a960604_anon012						2
a960604_anon013			2			2
a960604_anon015						1
a960604_anon016			3			3
a960604_anon017		7				
a960625						
Adalah_Al-Jabah						10
Bill_Clinton			2			2
Britt_Hume			3			
David_Baldwin						6
David_Ensor			3			
Phil_Wilcox		2				
Ray_Madis		4				
Secretary_Perry			1			1
Ted_Koppel	23	2	1	3		
a960625_anon001				1		
a960625_anon007				1		

a960610						
speaker name	F0	F1	F2	F3	F4	FX
ABC_NLI_Announcer				1		
Chris_Beary	6	5			4	
Cokie_Roberts	4	12		2		
a960610_janedoe001			1			
a960610_janedoe003		5				
a960610_janedoe004		7				
a960610_johndoe002						1
a960610_johndoe003			1			
a960610_johndoe004			1			
a960610_johndoe005						1
a960610_johndoe006			1			
a960610_johndoe007						1
a960610_johndoe008		1	2			
a960610_johndoe009			1			
a960610_johndoe010			1			
a960610_johndoe011			1			
a960610_johndoe011			1			
a960610_johndoe012		3				
a960610_johndoe013		3				

a960621						
ABC_NLI_Announcer				2		
Mike_Von_Fremd	3		1		6	
Ted_Koppel	4	11		1	1	
a960621_janedoe002			1			
a960621_janedoe003			1			
a960621_johndoe001			1			1
a960621_johndoe002			7			1
a960621_johndoe004		3				
a960621_johndoe005		3				
a960621_johndoe006		4				

a960624						
Brian_Ross	3		1		17	2
Mary_Schiavo			2			
a960624_anon002			3			
a960624_anon005			1			
a960624_anon009			1			
a960624_anon015		7				

a960626						
speaker name	F0	F1	F2	F3	F4	FX
ABC_NLI_Announcer				1		
Chris_Beary	2				11	
Chris_Wallace	9	5			1	1
Michelle_McQueen	3			5		
a960626_anon002						1
a960626_anon003						1
a960626_anon005						1
a960626_anon006						1
a960626_anon007			4			
a960626_anon008						2
a960626_anon014						1
a960626_anon015			1			
a960626_anon016						1
a960626_anon017			1			1
a960626_anon018			1			
a960626_anon018			1			
a960626_anon021			1			
a960626_anon024			3			
a960626_anon025						1
a960626_anon026			1			
a960626_anon027			1			
a960626_anon028			1			
a960626_anon029			3			
a960626_anon031			1			
a960626_anon036			1			1
a960626_anon038			1			
a960626_anon039			1			
a960626_anon040			4			1

Chris_Wallace	3	13		1	1	1
a960624_anon001				1		1
a960624_anon003			2			1
a960624_anon006			4			2
a960624_anon014		7				

I Software Written

A fuller listing of the main software written for this project is submitted in a separate volume.

I.1 Classification of Testers and Reference Speakers

I.1.1 tcsh scripts

`dostuff $showname`

Generates all the covariance and A-matrix data for all of the shows. Writes the information into a MATLAB-readable form. Calls `getcov, newcovtomat, datatrans` and `atomat`

`getcov $showname`

Makes an .scp file of all the segments in the show of more than 5 seconds duration. Runs `HCompV -m` in `HTK` [18] to generate the inverse-covariance and mean for each segment. Stores this information in a model file for each segment Lists all model files into $showname.files

`datatrans $showname`

As above, but runs `HERest` in `HTK` to generate the MLLR transpose for each segment. The config file set in the script determines whether model-based or data-based transforms are generated.

I.1.2 C programs

```
USAGE: newcovtomat [OPTIONS] -f filelist

 Option                                      Default

 -l fn   Write debugging info to file fn     stdout
 -o fn   Write matlab output to file fn      stdout
 -s fn   Write Speaker IDs to file fn        stdout
 -T N    Set Trace Level to N                  3
 -a fn   filelist of A matrices for adapting NONE
 -i      Try using inverse A-adaptation       FALSE
```

Reads in the upper triangular half of the inverse covariance matrix given in the segment files listed in `filelist`. Assigns every third one to be a test segment and the others to be references.

If the `-a` option is set, the transpose of the A-matrices in the following filelist are read in for adaptation, with the `-i` flag inverse adaptation is used instead.

Writes the identities of the speakers of the reference/test segments to the file specified by `-s` and writes the covariance, inverse of the covariance, mean and number of frames of each segment in MATLAB-readable form to the file specified by `-o`.

Allows different amounts of debugging information to be reported using `-T` and written to the log file specified by `-l`.

```
USAGE: atomat [OPTIONS] -f filelist

 Option                                          Default

 -l fn   Write debugging info to file fn         stdout
 -o fn   Write matlab output to file fn          stdout
 -s fn   Write Speaker IDs to file fn            stdout
 -T N    Set Trace Level to N                      3
 -y N    Type of Symmetrizing to use               1
           1=none, 2=A+A', 4=AA', 8=A'A
```

Reads in the transpose of the A-matrix from the segment files listed in `filelist`. Assigns every third one to be a test segment and the others to be references.

Allows the transpose of the A-matrix to be replaced by A+A', AA' or A'A if having a symmetrical matrix is important, by setting the `-y` option.

Writes the (transposed) A-matrix, the inverse of this and the offset vector of each segment to the file specified by `-o` in MATLAB-readable format.

`-s`, `-T` and `-l` options are as for `newcovtomat`.

I.1.3 MATLAB Programs

`multi_off1.m`

Reads in the covariance or A-matrix data. Calculates distance matrices between all the testers and all the reference speakers for all the distances given in appendix A.

Calls `show_class` to find the minimum distance in the matrices and assigns each tester to each closest reference speaker. Writes the output to an ascii file

`show_class.m`

Given a tester/reference distance matrix, finds the minimum and second minimum value, classifies the tester as the closest reference speaker and produces an associated confidence of the log of the ratio of these two values.

`bhatrow.m, div2row.m`

Finds the Bhattacharyya distance and Gaussian divergence between two distributions assuming the mean is a row vector.

I.1.4 PERL Scripts

`evaldist.prl`

Takes the speaker ID file generated from `dostuff` and the classification file generated from `multi_off1.m` and calculates how many of the classifications for each distance metric have been assigned to the correct speaker and/or background condition.

Counts the number of unseen speakers and conditions. Calculates the percentage accuracy based on the ratio of the number of correct classifications to the number of segments which could have been correctly classified from the reference set. Outputs the results as a LaTeXtable.

I.2 Simple Clustering Procedures

I.2.1 Distance Measures

aagh(cov1,invcov1,cov2,invcov2)	log(AA/GH)
agm(invcov1,cov2)	log(A/G)
ahm(cov1,invcov1,cov2,invcov2)	log(A/H)
lhr(invcov1,cov2)	A-log(G)-1
bhat(mean1,cov1,mean2,cov2)	Bhattacharyya
div2(mean1,cov1,invcov1,mean2,cov2,invcov2)	Divergence

I.2.2 Combining distributions

[num,mu,cov]=concat_stats(n1,mu1,S1,n2,mu2,S2)	for Concatenation Clustering
[num,mu,cov]=weight_stats(n1,mu1,S1,n2,mu2,S2)	for Centroid Clustering
[num,mu,cov]=average_stats(n1,mu1,S1,n2,mu2,S2)	for Median Clustering
[newdist]=lw_comb(distki,distkj,distij)	Calculate Lance-Williams Distance

I.2.3 Clustering Procedures

`Dist` is the input distance matrix.
`link` is an array which contains the identity of the "parent" of each individual/cluster.
`minval` is an array of the critical clustering values.

[link,minval]=fneigh(Dist)	Furthest Neighbour
[link,minval]=nneigh(Dist)	Nearest Neighbour
[link,minval]=mutneigh(Dist)	Mutual Nearest Neighbour
[link,minval]=group_av(Dist)	Group Averaging
[link,minval]=lancew(Dist,NUM_DESIRED)	Lance Williams
cluster_con_msms(type,dist_fun)	dist_fun :== bhat, type:==concat|weight|average
cluster_con_sisi(type,dist_fun)	dist_fun :== ahm|aagh
cluster_con_is(type,dist_fun)	dist_fun :== agm|lhr
cluster_con_msimsi(type,dist_fun)	dist_fun :== div2
ward_2D	Wards clustering on vectors
ward_3D	Wards clustering on matrices
[splinter,rest]=split_cluster(Dist)	polythetic divisive clustering

Note the type of Lance-Williams clustering used is set by changing the parameters in `lw_comb`.

I.2.4 Utility Functions

[matrix]=array2lmat(array,length)	return a lower-triangular matrix with values from array
[smallest,greatest]=order2(a,b)	order 2 numbers
[truth]=ismember(number,array)	checks if `number` is in `array`
[value,index]=secmin(vector)	returns second minimum value in an array
get_dists	get the relevant distance matrix from the data
[max,row,col]=max_lmat(matrix)	find maximum of Lower-Triangular matrix
[max,row,col]=min_lmat(matrix)	find maximum of Lower-Triangular matrix
[max,row,col]=min_mat_nod(matrix)	find maximum of matrix ignoring diagonal

I.2.5 Looking at cluster structure

[array]=an_link(link)	show clustering decisions
show_tree2(array,num_orig,minval)	Show Tree diagram and critical value
show_tree3(array,num_orig,minval,secval)	Show Tree diagram, critical value and next best value.

I.3 Full-scale Implementation

I.3.1 tcsh scripts

`handsplit $string`

Assigns all the files in `C/clust_data/a960610.sp` which contain the string `$string` to cluster 1 and all the others to cluster 2. writes output to a file clusout.$string Can be used to form speaker or condition clusters.

`handsplit2 $str1 $str2`

As for `handsplit` but allows two strings to be used to assign segments to cluster 1 (logically ORed). For example: "`handsplit2 Cokie jane`" to give gender-split

`mkclustershand $showname $string $outdirec`

Uses `HERest` in `HTK` to calculate the auxiliary function values of the clusters defined in clusout.$string and places the output in $outdirec/$showname.spec/$string.2cls

`gethand`

Contains the showname and a list of query strings with which
to call `handsplit` and `mkclustershand`.

`mkclusters6 $showname $showdir $clusfile $outfile $direc`

Reads clusters from $clusfile, speaker id from $direc$showdir.sp makes soft link to PLP information. Runs `HERest` to calculated the MLLR transforms and outputs results to $outfile

`getauxval $resfile`

Calculate the auxiliary-based FOM for $resfile

`getauxall $direc`
Calculate the auxiliary-based FOM for all relevant files in $direc

`eval_all_cov_fix $showname`
Calculate the auxiliary-based FOM based on covariance data for all options specified in the script.

`eval_all_data_fix $showname`

As above, but for data-based MLLR transform data.

`eval_most_aat $showname`

As above, but for AA^T instead of A.

`eval_all_cov_sym $showname`

As above but for symmetric options on covariance data

I.3.2 C programs

```
USAGE: gendist [OPTIONS] -f filelist

 Option                                      Default

 -l fn   Write debugging info to file fn     stdout
 -o fn   Write distance metric to file fn    stdout
 -s fn   Write Speaker IDs to file fn        stdout
 -T N    Set Trace Level to N                  3
 -d N    Type of Distance Metric to use        1
     1=Euclid, 2=City, 4=Ang, 16=Ahm
 -c N    Type of coefficients to use           1
     1=Statics, 2=Deltas, 4=Accels
 -v N    Type of Data to Use                   1
     1=matrices only, 2=mean only, 3=all
 -y N    Type of Symmetrizing to use           1
     1=none, 2=A+A', 4=AA', 8=A'A
```

Generate distance matrix based on MLLR transforms

```
USAGE: gendistcov [OPTIONS] -f filelist

 Option                                      Default
 -l fn   Write debugging info to file fn     stdout
 -o fn   Write distance metric to file fn    stdout
 -s fn   Write Speaker IDs to file fn        stdout
 -T N    Set Trace Level to N                  3
 -d N    Type of Distance Metric to use        1
       1=Div, 2=AHM, 4=AGM, 8=AlogG1, 16=aagh,
       32=AlogGH, 64=Bhat
 -x N    Type of Symmetry option on distance   1
       1=none, 2=lina, 4=loga, 8=linb, 16=logb
 -a fn   filelist of A matrices for adapting
```

Generate distance matrix based on covariance data

```
USAGE: cluster [OPTIONS] -f distfile

 Option                                     Default
 -o fn   Write output to file fn            stdout
 -l fn   Write debugging info to file fn    stdout
 -n N    set number of final clusters to N    2
 -T N    Set Trace Level to N                 3
 -c N    Set Clustering Type to N             1
          1=LW, 2=NN, 4=FN
 -w N    Set Lance-Williams type to N         1
          1=b=-0.5 2=b=-0.25 4=NN 8=FN
 -d N    Set display value to N               8
      1=clusters 2=links 4=family 8=standard
```

Cluster the segments using the generated distance matrix.

`structures.h`

Contains the global constants, structures, enums, and typedefs used in all the programs in this section.

I.3.3 Matrix and I/O Library

All functions are preceded by `extern` and the routines `ludcmp, lubksb, InvertMatrix` were adapted from algorithms in [19].

```
void     Quit(char *message,...); /* exit program */
char*    Chop(char *string); /* chop \n if it trails string */

IVector  InitIVector (int length);
DVector  InitDVector (int length);
TrVector InitTrVector (int length);
Matrix   InitMatrix(int num_rows, int num_cols);
Block3   InitBlock3(int num_params);
B3Array  InitB3Array(int length);
MVISet   InitMVISet(int dimen,int nframes);
MVIArray InitMVIArray(int length);

void     PrintIVector(FILE*, IVector);
void     PrintDVector(FILE*, DVector);
void     PrintTrVector(FILE*, TrVector);
void     PrintMatrix(FILE*, Matrix);
void     PrintBlock3(FILE*, Block3);
void     PrintB3Array(FILE*, B3Array);
void     PrintDistances(FILE*, char *filename, Matrix);
void     PrintMVISet(FILE*, MVISet);

void     FreeIVector(IVector);
void     FreeDVector(DVector);
void     FreeTrVector(TrVector);
void     FreeMatrix(Matrix);
void     FreeBlock3(Block3);
void     FreeB3Array(B3Array);

/* extracting subsets etc */
DVector  SubsetDVec(DVector array, int start, int end);
Matrix   SubsetMatrix(Matrix mx, int rowa, int cola, int rowb, int colb);
DVector  ConcatDVec(DVector, DVector);
Matrix   SuperMatrix2(Matrix mx11, Matrix mx12, Matrix mx21, Matrix mx22);

/* special matrices */
void     ZeroMatrix(Matrix);
Matrix   EyeMatrix (int dimen);
Matrix   CopyMatrix(Matrix Orig);         /* returns new matrix copy*/
void     CopyMatrixFT(Matrix orig, Matrix copy); /* overwrites copy */
void     SymMatrixAdd(Matrix);    /* A + A' */
void     SymMatrixMult(Matrix);   /* AA' */
void     SymMatrixTrMult(Matrix); /* A'A */

/* Boolean Tests */
Boolean  IsSquare(Matrix);
Boolean  IsSymmetric(Matrix);
Boolean  IsMember(IVector array, int value);

/* arithmetic Matrix Operations */
Matrix   ScMultMatrix(float, Matrix);
Matrix   MultMatrix(Matrix, Matrix);
DVector  MultMatVec(Matrix, DVector);
Matrix   AddMatrix(Matrix, Matrix);
Matrix   SubMatrix(Matrix, Matrix);
```

```
Matrix   TransposeMatrix(Matrix);
Matrix   InvertMatrix(Matrix);
double   DetMatrix(Matrix);
double   TraceMatrix(Matrix);
double   TraceMatrixProd(Matrix, Matrix);
double   SumSqMatrix(Matrix);

/* used for inverse */
void     ludcmp(Matrix mx,int *index, double *d);
void     lubksb(Matrix mx,int *index, double b[]);

/* min max operations */
Element  MaxDifMatrix(Matrix, Matrix);
Element  MinMatrix(Matrix);
Element  MinMatrixNoD(Matrix);
Element  MaxMatrixNoD(Matrix); /* no diagonal */
Element  MinLMat(Matrix); /* elements below diagonal */
Element  MaxLMat(Matrix);

/* special means */
double   GeoMeanMatProd(Matrix y, Matrix invx);
double   ArithMeanMatProd(Matrix y, Matrix invx);
double   HarmMeanMatProd(Matrix invy, Matrix x);

/* distance metrics on Matrices */
double   EuclidMat(Matrix, Matrix); /* element wise */
double   CityMat(Matrix, Matrix);   /* element wise */
double   AngMat(Matrix, Matrix);
double   AhmMat(Matrix mx1, Matrix mx2, Matrix invmx1, Matrix invmx2);

/* asymmetric distance measures on Matrices */
double   AgmMat(Matrix x, Matrix y, Matrix invx, Matrix invy);
double   AaghMat(Matrix, Matrix, Matrix, Matrix);
double   AlogG1Mat(Matrix, Matrix, Matrix, Matrix);
double   AlogGHMat(Matrix, Matrix, Matrix, Matrix);

/* corrected symmetric distance measures on Matrices
   dist_f can be AgmMat, AaghMat, AlogG1Mat, AlogGHMat */
double SymDistMat(Matrix x, Matrix y, Matrix invx, Matrix invy,
                  int numx, int numy, SymDType method,
                  double (*dist_f)(Matrix, Matrix, Matrix, Matrix));

/* arithmetic vector operations */
DVector  SubDVector(DVector, DVector);
DVector  AddDVector(DVector, DVector);
double   DotProdDVec(DVector, DVector);
Matrix   DyadProdDVec(DVector, DVector);
double   MaxDifDVector(DVector, DVector);

/* distance metrics on Vectors */
double   EuclidVec(DVector, DVector);
double   CityVec(DVector, DVector);
double   AngVec(DVector, DVector);

/* arithmetic scalar operations */
double   max2(double, double);
double   absval(double);
```

References

[1] Bimbot F. & Mathan L.
Text-Free Speaker Recognition using an Arithmetic Harmonic Sphericity Measure.
Proc. Eurospeech, 1993, Vol. 1. pp 169-172

[2] Bimbot F. & Mathan L.
Second-Order Statistical Measures for Text-Independent Speaker Identification.
ECSA Workshop on Automatic Speaker Recognition, Identification and Verification, 1994, pp 51-54

[3] Cook G.
Data Selection and Model Combination in Connectionist Speech Recognition.
Chapter 5, Ensembles for Speaker Adaptation
PhD Thesis, Department of Engineering, University of Cambridge, UK, January 1997.

[4] Everitt B.
Cluster Analysis.
Haltsted Press, New York, 1980

[5] Fukunaga, K.
Introduction to Statistical Pattern Recognition.
Boston; London: Academic Press, 1990

[6] Gales M. & Woodland P.
Mean and Variance Adaptation within the MLLR Framework.
Computer Speech and Language, 1996, pp249-264

[7] Gales M.
Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition.
Technical Report CUED/F-INFENG/TR291 May 1997
`ftp://svr-ftp.eng.cam.ac.uk/pub/reports/gales_tr291.ps.gz`

[8] Gish H.
Robust Discrimination in Automatic Speaker Identification
Proc ICASSP 1990, Vol. 1, pp 289-292

[9] Gish H., Siu M., & Rohlicek R.
Segregation of Speakers for Speech Recognition and Speaker Identification.
Proc. ICASSP 1991, Vol. 2, pp 873-876

[10] Heck L. & Sankar A.
Acoustic Clustering and Adaptation for Improved Speech Recognition.
1997 DARPA Speech Recognition Workshop

[11] Hermansky H.
Perceptual Linear Prediction (PLP) Analysis for Speech.
Journal Acoustic Society of America, Vol. 87, pp 1738-1752

[12] Jin H. & Schwartz R.
Automatic Speaker Clustering.
1997 DARPA Speech Recognition Workshop

[13] Leggetter C. & Woodland P.
Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models.
Computer Speech and Language, 1995, pp 171-185

[14] Leggetter C. & Woodland P.
Flexible Speaker Adaptation for Large Vocabulary Speech Recognition.
Proc. Eurospeech 1995, Vol. 2, pp 1155-1158

[15] Liu G.
Introduction to Combinatorial Mathematics.
McGraw Hill, 1968

[16] Mayer D.
Segmentation/Labelling of Unrestricted Audio.
MPhil Thesis 1997, Department of Engineering, University of Cambridge, UK

[17] Neumeyer L., Sankar A. & Digalakis V.
A Comparative Study of Speaker Adaptation Techniques.
Proc. Eurospeech 1995, Vol. 2, pp 1127-1130

[18] Odell J., Ollason D., Woodland P., Young S. & Jansen J.
The HTK Book for HTK V2.0.
Cambridge University, Cambridge, UK, 1995

[19] Press W., Flannery P., Teukolsky S. & Vetterling W.
Numerical Recipes in C.
Cambridge University Press, 1988

[20] Pye D. & Woodland P.
Experiments in Speaker Normalisation and Adaptation for Large Vocabulary Speech Recognition.
Proc. ICASSP 1997, Vol. 2 pp 1047-1050

[21] Smith G.
Speaker Recognition.
Fourth Year Project Report 1997, Department of Engineering, University of Cambridge, UK

[22] Therrien C.
Decision Estimation and Classification.
J. Wiley and Sons, New York, 1989

[23] Woodland P., Gales M., Pye D. & Young S.
Broadcast News Transcription using HTK.
Proc. ICASSP 1997, Vol. 2 pp 719-722