

SPEAKER VERIFICATION EVALUATION REPORT

*Scientific evaluation of speaker verification technologies
on behalf of Australian Government*

Document No: SVE Test Report Version 2.0

Project ID: RFP-SV-026f

Customer: Telstra Corporation

COPYRIGHT RESERVED

The information and design as detailed in this document is the property of the University of Canberra. This document must not be reproduced, copied or communicated to any third party, nor be used for any purpose other than that stated in the particular inquiry, order or contract with which it is issued, without the written permission of University of Canberra. The reservation of copyright in this document extends from each date appearing thereon and in respect of the subject matter as it appeared at that relevant date.

©University of Canberra 2005

EXECUTIVE SUMMARY

Authenticating identity in Government services is critical to providing secure yet convenient services to the general public. As part of the enhanced authentication framework being developed by certain Australian Government agencies, the Australian Government has been interested to understand the capabilities of commercial voice authentication for providing authentication functions in telephone and other on-line services.

In December 2004, the Australian Government engaged Telstra Corporation to conduct a commercial and scientific evaluation of available speaker verification technologies. The results of Telstra's commercial evaluation are not publicly available, however to support Australian Government's intention to make public the results of the scientific evaluation, Telstra commissioned the University of Canberra to perform an open and independent scientific evaluation of four selected verification technologies. This report reflects the results of the scientific evaluation only, and does not make any claim for commercial deployment performance.

The University's evaluation undertook to determine the performance of speaker verification engines against a speech database in order to establish different engines' suitability for use as an authentication mechanism for Australian telephone services. Specifically, the Australian Government was interested in gaining information on the performance and resilience of the engines to real-world scenarios, including mobile telephone environments and noise scenarios. Their interest was to ascertain which engines, either working in isolation or in combination, would perform best for enhanced proof of identity authentication in telephone and on-line services.

The four engines submitted for evaluation were:

1. Scansoft with SpeechWorks Speaker verification SDK pro 3.0 (Scansoft)
2. Nuance with NVP 2.0.1 (Nuance)
3. Persay with VocalPassword Build 5.0.5.0 (Persay)
4. KAZ with SpVerServer version1.3 (KAZ)

Nuance and Persay engines were evaluated in text-dependent mode only. Scansoft's engine was evaluated in both text-dependent and text-independent modes, whilst the KAZ engine was evaluated in text-independent mode only. Adaptation capabilities were not tested.

All engines were set-up in an evaluation environment that enabled direct comparison of the engines and integrated speech processing (Emu) and biometric evaluation (Performix) tools to run and analyse various test cases.

Three test cases were set-up, involving around 300 speakers from the speech database: counting 1-to-9; saying names; and saying long sequences of speech created by concatenating single utterances. These long samples were used for text-independent testing, where the engine verified callers on voice quality alone, irrespective of what the caller was saying. Speech samples were processed through mobile telephone channels to create mobile telephone versions as well as mixed with noise (white, office, city and shopping centre noise samples) to simulate noise conditions found in live deployments.

Benchmark text dependent tests (using Counting 1to9 speech) showed Nuance and Persay significantly out-performs Scansoft, returning an EER (equal error rate) measure of 0.86% and 1.55% respectively compared to Scansoft at 4.27%. Furthermore, Nuance and Persay showed high levels of robustness (that is insensitivity) in both mobile and noisy conditions, with Nuance and Persay showing around three times less sensitive to mobile and noise conditions than Scansoft for the same test conditions.

Engine performance was much more even in name tests with Nuance, Scansoft, and Persay returning EER benchmark scores of 4.49%, 5.27% and 5.33%, respectively. Tests using noisy speech showed no appreciable difference in robustness between the engines. However, tests using mobile telephone speech did show Nuance and Persay to be marginally more robust than Scansoft.

Name tests were also configured to simulate "shared secret" authentication where the enrolled speaker (client), is tested saying the wrong name. These tests showed that all engines provide similar levels of performance and were able to discriminate in a situation where a legitimate caller speaks the wrong answer to a shared secret question.

A series of text-dependent tests were also performed on longitudinal and sibling speech data. All engines shows similar sensitivity in the longitudinal tests, with engines exhibiting increases in the EER

performance of between 3% and 6% for counting 1to9 and around 1% variation in ERR performance on names. Tests of sibling speech showed that Scansoft exhibited significantly higher levels of sensitivity to this condition compared to Nuance and Persay with an increase in EER of around 6% over the baseline, compared to 1.3% for Nuance and Persay on the same speech data.

A series of enrolment tests were also performed. These included reducing enrolment from three samples (the recommended number of enrolment samples) to two, and also to a single sample to look at how reduced enrolment impacts on EER performance. These tests showed Nuance and Scansoft exhibited only marginal reduction in performance, around 0.25% and 0.4% increase in EER, when reducing enrolment from three to two samples. However, a reduction was produced when enrolment was reduced from two to a single utterance (around 1.1% for Nuance and almost 3% for Scansoft). Persay's engine actually showed slightly improved EER performance as the enrolment was reduced, indicating that Persay's engine may be insensitive to the number of samples used in the enrolment procedure.

Tests were also conducted to measure sensitivity to enrolment using mobile speech data. These tests showed that all engines showed some degradation in performance when enrolled using mobile speech data. However Nuance and Persay was significantly more robust than Scansoft, with both engines registering 1.5% and 1% increase in EER, compared to Scansoft where the EER performance increased 3.5% over their respective baseline performances.

Two engines were involved in the text-independent tests, Scansoft and KAZ. Test cases were set up using the same 300 speakers as in the text-dependent tests, but with enrolment and verification speech samples produced by concatenating speech items from the database. Baseline tests showed Scansoft and KAZ, to perform very differently, with Scansoft returning an EER performance of between 7.2% and 5.7% compared to KAZ 0.38% and 0.47%. Testing with mobile speech data also showed KAZ to be almost totally insensitive to mobile communications, with an increase of in EER of only 0.13% compared to Scansoft's increase of 5.6% over and above its baseline performance. KAZ also maintains this highly robust behaviour in noisy conditions, registering results that were at least an order of magnitude better than Scansoft. Only under the most extreme noise conditions did the Scansoft and KAZ engines register equivalent performances.

1.1 Key Findings

Finding 1

Text dependent evaluation involving Counting 1to9 (Test Case 1-1): Nuance and Persay engines clearly showed stronger performance in terms of baseline EER performance as well as showing higher levels of robustness in mobile communications and noise conditions.

Finding 2

Text dependent evaluation involving Name verification (Test Case 4-1, 2, 2A): all engines performed to a similar level of performance; however Nuance and Persay showed higher levels of robustness in mobile communications and noise conditions for Name verification.

Finding 3

Text dependent evaluation involving Name verification (Test Case 4-2, 2A): tests show that all engines would have an ability to discriminate between a client and an impostor saying the right answer to a "shared secret" question – indicating all engines are suitable for implementing this function.

Finding 4

Enrolment reductions: Nuance and Scansoft both registered decreases in performance as the enrolment was reduced. This test indicates that two repetitions would be the minimum number of repetitions required for adequate enrolment. The test indicates that a single utterance is inadequate for reliable and robust verification.

Finding 5

Mobile telephone enrolments: Nuance and Persay exhibited the similar high levels of robust performance.

Finding 6

Text-independent tests: Baseline tests showed KAZ performed significantly better than Scansoft, with KAZ returning an EER performance of between 0.38% and 0.47% compared to Scansoft's EER of between 7.16% and 5.69% for the same speech data. The difference was greater than an order of magnitude difference in performance.

Table of Contents

1 EXECUTIVE SUMMARY	2
<i>1.1 Key Findings</i>	3
2 EVALUATION ENVIRONMENT AND PROCESS	6
<i>2.1 Introduction</i>	6
<i>2.2 Overview of the Core Engine Evaluation Environment</i>	6
<i>2.3 Speech and Noise Database</i>	8
2.3.1 A8 Speech Data	9
2.3.2 Speech Database Materials	9
2.3.3 Noise Database Material	11
<i>2.4 Speech Signal Processing and Tools</i>	12
2.4.1 Emu Speech Database Tool and Snack Speech Signal Processing Software	12
2.4.2 Performix Biometric Analysis Tool	13
2.4.3 Speech Database Audit	14
<i>2.5 Processing Speech Samples for Evaluation</i>	15
2.5.1 Creating Noisy Speech Samples	15
2.5.2 Creating Mobile Telephone Speech Samples	16
2.5.3 Creating Speech Samples for Text-Independent Evaluations	17
<i>2.6 Evaluation Environment and Set-up</i>	18
<i>2.7 Summary of Evaluation Environment and Processes</i>	19
3 EVALUATION TEST CASES	20
<i>3.1 Description and Designations</i>	20
<i>3.2 Text-Dependent Test Cases</i>	20
<i>3.3 Text-Independent Test Cases</i>	22
4 EVALUATION PROCEDURES – TEST CASE SET-UP	23
<i>4.1 Introduction</i>	23
<i>4.2 “m-script” and “t-script” Formats</i>	23
<i>4.3 Process for Producing M-scripts and T-Scripts</i>	24
5 EVALUATION RESULTS	27
<i>5.1 Introduction</i>	27
<i>5.2 Text Dependent Test Cases</i>	27
5.2.1 Test Case 1-1 Counting 1to9	27
5.2.1.1 Baseline Results	27
5.2.1.2 Observations	29
5.2.1.3 Mobile Telephone Performance	29
5.2.1.4 Observations	30
5.2.1.5 Noise Performance	30
5.2.1.6 Observations	32
5.2.2 Test Case 4-1 – Name Verification	32
5.2.2.1 Baseline Results	32

5.2.2.2	Observations	33
5.2.2.3	Mobile Telephone Performance	34
5.2.2.4	Observations	35
5.2.2.5	Noise Performance	35
5.2.2.6	Observations	36
5.2.3	Test Case 4-2 and 4-2A – Name Verification	38
5.2.3.1	Observations	38
5.2.4	Test Case 3-1 Longitudinal Speech Data “Counting 1to9” and “Names”	38
5.2.4.1	Observations	39
5.2.5	Test Case 7-1 and 7-2 - Sibling Tests	40
5.2.5.1	Observations	40
5.2.6	Test case 9-1 and 9-2 - Reduced Enrolment Tests	41
5.2.6.1	Observations	42
5.2.7	Test case 12-1 and 12-2 - Mobile Telephone Enrolment	43
5.2.7.1	Observations	44
5.3	<i>Text Independent Test Case</i>	45
5.3.1	Baseline Performance	45
5.3.1.1	Observations	45
5.3.2	Mobile Telephone Performance	45
5.3.2.1	Observations	46
5.3.3	Noise Performance	46
5.3.3.1	Observations	48
5.4	<i>Volumetric Stress Tests</i>	49
5.4.1	Observations	50
6	FINDINGS AND CONCLUSIONS	51
6.1	<i>Text Dependent Evaluations</i>	51
6.1.1	Finding from Test Case 1-1	51
6.1.2	Finding from Test Case 4-1, 4-2 and 4-2A	52
6.1.3	Finding from Test Case 9-1 and 9-2 - Reduced Enrolment	53
6.1.4	Finding from Test Case 12-1 and 12-2 – Mobile Telephone Enrolment	53
6.1.5	Finding from Text Independent Evaluations	54
7	Vendor Feedback	55
7.1	<i>Introduction</i>	55
7.2	<i>Feedback from Scansoft</i>	56
7.3	<i>Feedback from Persay</i>	58
7.4	<i>Feedback from KAZ</i>	60
8	Glossary	62
9	References	64
10	Acknowledgements	65

2 EVALUATION ENVIRONMENT AND PROCESS

2.1 Introduction

In July 2004 the Australian Government collected a speech database of 567 speakers for Government employees. The database collection was designed to support the evaluation of speaker verification engines and included two parts, the collection of enrolment speech samples and verification samples. The enrolment part comprised the speaker saying three repeats of the same speech information. The verification part was collected separately from the enrolment part and comprised a single sample of the same speech material. The database enables separate speech samples to be used for enrolment and verification, thus providing an accurate assessment of the performance and the speaker verification engines.

In March 2005 a subset of the original group of speakers were contacted and invited to record the same speech material originally collected in July 2004. The resulting speech material collected was used in the longitudinal evaluations.

Also during this time, siblings, including twins, were identified within Australian Government employee database and invited to participate in the speech database collection. Siblings recorded both the enrolment and verification speech samples as in the original collection. This material was then used to evaluate the speaker verification engines' performance to speech data recorded by siblings, including twins.

The speech database was used to confirm the *relative* performance of speaker verification engines. This involved determining the *baseline* performance for each engine and then examining how the performance varies against the baseline result.

The focus of the evaluation was to examine *relative* behaviour of the engines under different noise conditions and telecommunications environments. These included quiet, white noise, office, city and shop noise conditions in both landline and mobile telecommunications environments.

The evaluation specifically referred to evaluating a *verification function*, not an *identification process*. As a consequence, the evaluation focuses on measuring the performance of the engine in correctly accepting legitimate individuals (clients) and correctly rejecting impostors (one-to-one process). The evaluation did not evaluate performance of the engines in correctly identifying an individual against the full database of voice templates (one-to-many process).

2.2 Overview of the Core Engine Evaluation Environment

Figure 1 shows the overall evaluation environment.

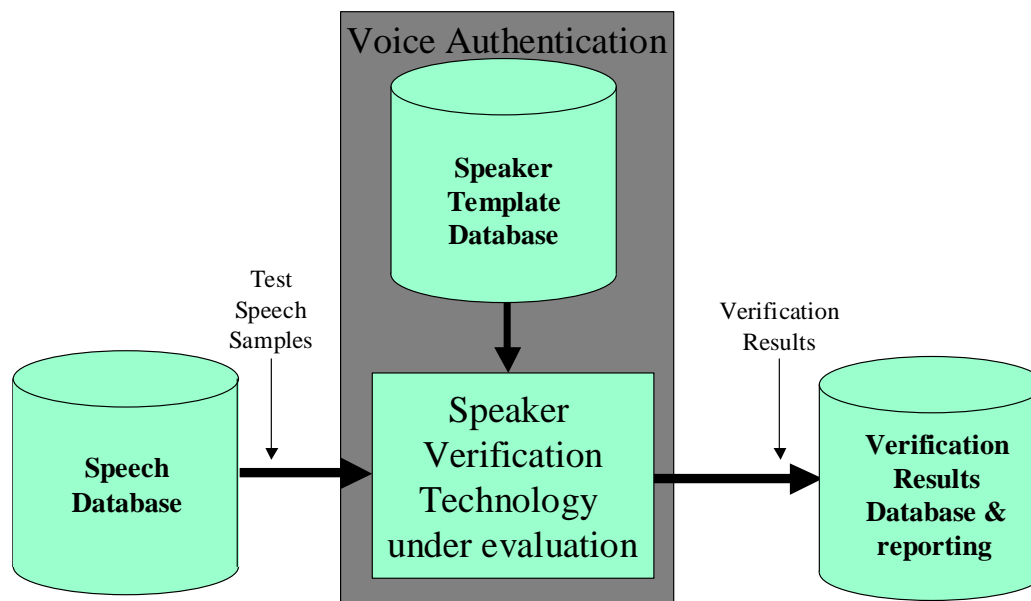


Figure 1: Verification Engine Evaluation Environment

There were three components to the evaluation environment:

1. The speech database
2. The verification engine under evaluation (along with its speaker template database);
3. The verification evaluation results database and reporting.

The evaluation process was conducted in two parts. The first part created the voice templates (enrolment), whilst the second part compared a selected speech file against a nominated voice template to determine verification performance.

During enrolment, enrolment speech samples were selected from the speech database and presented to each of the verification engines to create the speaker voice templates. The standard enrolment process used all three enrolment samples from each speaker and the enrolment process was repeated for each individual in the speech database. However, during reduced enrolment evaluations, the number of speech samples from each speaker used in the enrolment process was reduced from three, to two and then to a single speech utterance. During the mobile enrolment evaluations, the enrolment speech data was processed through the mobile telephone processor, before being made available for enrolment.

The evaluation process involved extracting a verification speech sample from the speech database and presenting this to the engine for processing. The engine processes this speech sample, comparing it to the selected speaker template (the claimed identity) and generating a numerical verification result, which is recorded in the results database. This process is repeated for all speech samples in the database to obtain an overall measure of the performance of the engine under evaluation.

Two standard verification configurations were established:

1. a full evaluation process which involved testing each model (or speaker template) against the client and impostors comprising every other speaker enrolled in the system
2. a reduced evaluation process which involved testing each model (or speaker template) against the client and a limited set of impostors selected from the enrolled speakers of the same gender.

Verification performance was initially calibrated using the full evaluation process. However, given the processing time required to complete the full evaluation, the reduced process was used to determine the relative performance of the engines under evaluation.

Evaluations were administered under centralised software control using scripts, referred to as “m-scripts” and “t-scripts”. The “m-script” was used during the enrolment process and specified the model reference and the speech file used to create that model. The “t-script” was used during the evaluation process and specified the model and the speech file against which that model was compared. Both these scripts allow the engine to report the results of enrolment and verification, as well as any errors that may occur during enrolment and verification.

Through this process, an accurate profile of the performance of the speaker verification engine in the different operating conditions was generated. The “likelihood scores” generated by the verification engine and initially logged in the “t-scripts” were extracted from these scripts and imported into the results database and Performix evaluation tool to allow detailed analysis of the engines’ performance.

The process allowed the false-accept rate (FAR) and the false reject rate (FRR) characteristics of each of the engines operating in each of the evaluation conditions to be produced and graphically displayed. The process also determined the threshold settings at which the FAR is equal to the FRR, that value being known as the equal-error rate (EER) for the different evaluation environments and conditions. This measure (EER) was then used as a raw performance measure for each of the engines in the evaluation. The lower the EER measure the better the engine at discriminating between clients and impostors in those environments and conditions.

2.3 Speech and Noise Database

The speech database was designed to examine the performance of speaker verification under different scenarios or operating conditions. In particular, the speech database was designed to replicate the noise and telecommunications conditions that are expected in real-world deployments.

Specifically, the objective of the speech database component is to create conditions that stress the engine and potentially trigger high levels of false accepts and false rejects. Specifically the speech database was set-up to examine the degradation in the overall performance of the engine as measured by the Equal Error Rate (EER) in different environments and operating conditions.

Figure 2 shows the structure of the speech database, which comprises four elements:

1. A speech database collected by the Australian Government containing utterances in different telephone environments (the A8 Database)
2. A database of noise scenarios, which represent various noise conditions expected in live deployments (the Noise Database)
3. The speech database processing tool (Emu)
4. The mobile & Cordless telephone channel processing (mobile processing)

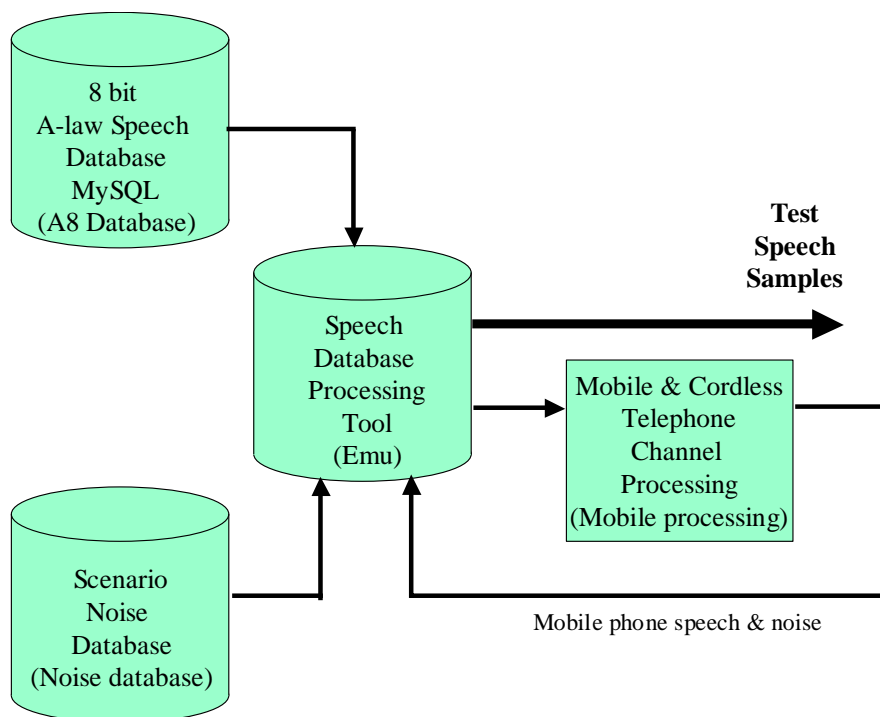


Figure 2. Speech database for speaker verification evaluation

2.3.1 A8 Speech Data

The A8-database contains the raw speech waveforms recorded from the telephone network and information about the speaker and the telephone environment in which the data was recorded.

This information is stored in a MySQL database which allows speech data and speaker demographics to be analysed in the evaluation process.

The speech data is recorded and stored as A8 data, that is A-law companded 8 bit speech data, the standard format used in digital voice communication in the Australian telephone network.

2.3.2 Speech Database Materials

The A8 speech database contains the recordings of each speaker speaking seven items of information. These items are listed in table 1 and they comprise the speaker saying their unique customers reference number (CRN), the speaker saying a name, the speaker counting from 1 to 9, the speaker saying a date of birth, the speaker saying a friend's name. The speaker saying their PIN and the speaker saying a location. Speakers were also recorded saying a money amount, but that was not including in the evaluation speech database.

Database Items	Speech Database Utterances
Item 1	The speaker saying their Customer Reference Number (CRN)
Item 2	The speaker saying a name
Item 3	The speaker counting 1 to 9
Item 4	The speaker's date of birth
Item 5	The speaker saying a friend's name
Item 6	The speaker saying their PIN
Item 7	The speaker saying their location

Table 1. Speech database structure

Whilst each speaker in the speech database recorded a unique CRN, speech items 2, 4, 5, 6 and 7 (that is name, date of birth, friend's name, Pin and location) contain only 11 different responses as documented in table 2.

This design allowed various impostor-testing strategies, including enrolling with names and testing against impostors saying the same name or different names. Using the friend's name item, it is also possible to design a test where a client could be configured as an impostor saying the wrong name.

The speech database material was split into two parts - the enrolment speech data and the verification speech data.

The enrolment speech data simulated an enrolment process and contains three repetitions of each speech item recorded, all in the same telephone call. The verification speech materials comprise the same speakers saying the same speech items, but only once. Some speakers recorded several versions of the verification speech database materials, either to simulate several verification runs, or in some cases because they had made a mistake in the original verification recording.

The original enrolment and verification speech materials were recorded in July 2004. Selected speakers were contacted in March 2005 during the evaluation procedures and invited to record a further set of speech items as they had in the original verification utterances. By enrolling using the original enrolment set recorded in July 2004 and then verifying against the speech material collected in March 2005, engine performance against longitudinal speech data could be determined.

Also during March 2005 siblings, including twins, were identified amongst Australian Government employees and invited to record an enrolment and verification set to enable evaluation of sibling speech data. Around 30 sibling pairs (60 sibling individuals) were identified and their speech material, along with their demographic information was loaded into the speech database.

Script No.	Name (Item 2)	Date of Birth (Item 4)	Friend's Name (Item 5)	PIN (Item 6)	Location (Item 7)
1	Tom Baker	15/8/70	Len Rowland	4709	Brisbane
2	Jane Low	1/7/82	Rebecca Jones	5123	Sydney
3	Mario Ferris	3/1/58	George Baxter	9051	Adelaide
4	Alex Bunberry	9/2/61	Helmut Heinz	8494	Melbourne
5	Peter Wright	3/3/79	Joseph Memo	7690	Hobart
6	Lynda Cook	4/11/76	Julia Como	1650	Perth
7	Annette Riverstone	30/12/67	Kathy Lam	2060	Alice Springs
8	Jack Samuel	25/9/64	Andrew Filler	3418	Darwin
9	Sophia Norton	16/10/81	Melinda Childs	6277	Canberra
10	Helen Alba	18/5/63	Brendan Jacks	1560	Flinders Island
11	Jamie Lee	16/11/74	Rebecca Cole	6734	Mittagong

Table 2. Speaker Responses for Items listed in table 1.

2.3.3 Noise Database Material

In order to simulate telephone speech originating from different noisy environments, the speech files were also mixed with four different recorded noise signals. These were:

1. White noise (similar to the noise from an untuned AM/FM radio receiver)
2. Office noise
3. City street noise
4. Shopping centre noise

White noise was generated using a random number generator which forms part of the Snack signal processing package available within Emu. Office noise was recorded in a university office and contained some general background office noise and air conditioner noise. City noise was recorded at Woden bus terminus and included cars and bus traffic. Shop noise was recorded in Woden Plaza shopping centre and included some piped music and people/children speaking.

Speech files were mixed with noise files at two different signal-to-noise ratios.

All sets of speech data files and their corresponding metadata descriptions were managed using a speech database management and processing tool (Emu).

A period of approximately 30 minutes of each noise scenario was recorded, loaded into the speech database and made available for mixing with speech signals to simulate real-world noise conditions. In the case of shopping centre noise multi-speaker environments, including child speech, were recorded.

Figure 4 shows the waveform and spectrographic analysis of the noise materials used in the evaluations. These were produced using the Emu speech database processing tool.

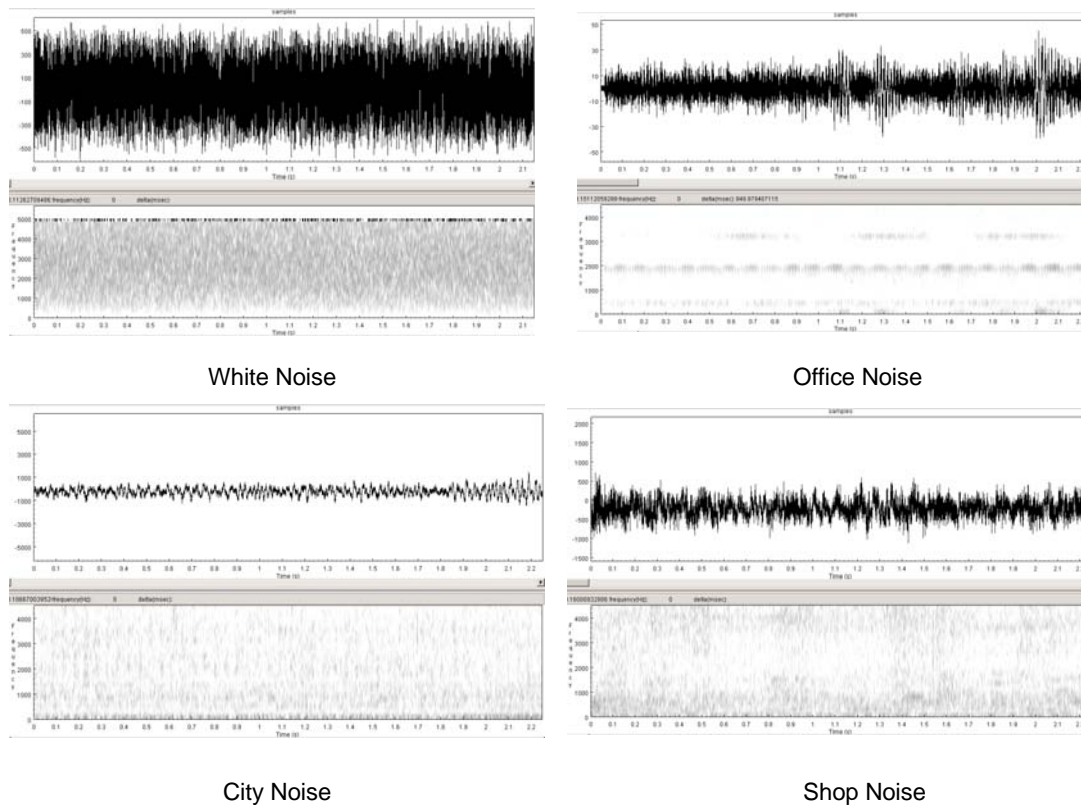


Figure 5. Noise Materials

2.4 Speech Signal Processing and Tools

2.4.1 Emu Speech Database Tool and Snack Speech Signal Processing Software

Emu was developed by Macquarie University in Sydney specifically for speech technology R&D. It has been used extensively by the speech science community for setting up speech analysis experiments and in speech engine research. It comprises a collection of software tools for the creation, manipulation and analysis of speech databases. It contains a hierarchical labeller and speech analysis toolkit (spectrographic tools) which is used for editing, labelling and processing speech databases. Emu integrates “Snack”, a speech signal processing toolkit developed at KTH Speech Laboratories in Stockholm (Sweden). Emu is designed for use with scripting languages (Tcl/Tk or Python) to implement a wide range of signal processing functions, including voice I/O, waveform scaling and mixing, and visualisation functions, such as waveform and spectrographic displays.

Figure 3 shows a screen shot of the Emu analysis tool.

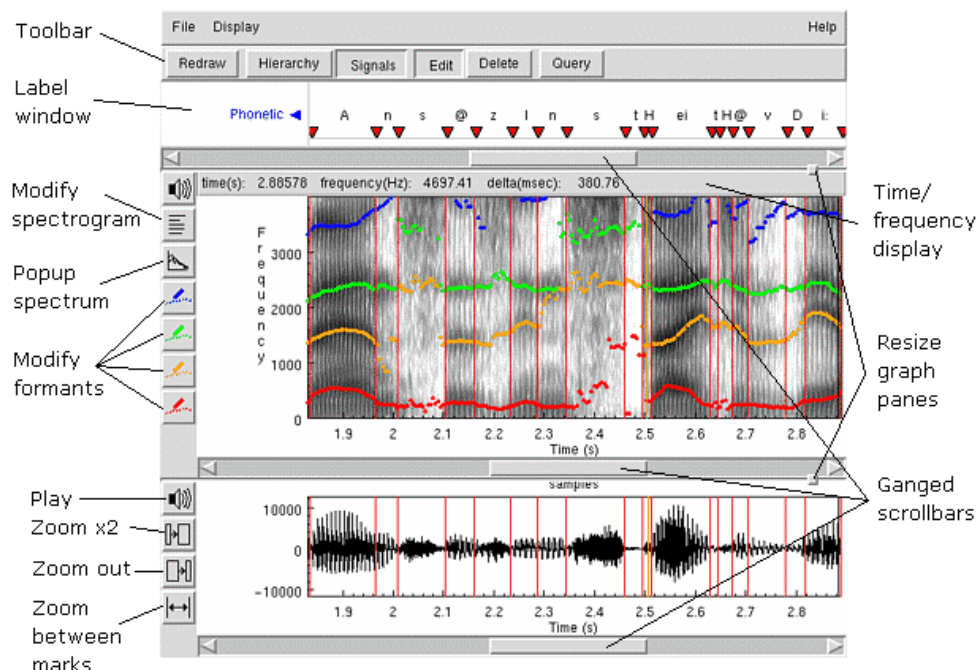


Figure 3. The Emu Speech Processing Tool

2.4.2 Performix Biometric Analysis Tool

Performix is an analysis tool developed by Biometix Pty Ltd for the evaluation and testing of biometric technologies. The tool was originally developed for evaluation of face recognition technologies used in the Smartgate border control system used at Australian international airports.

The tool allows for the probability score output from one or more biometric systems to be imported and various analysis procedures to be applied. This includes generating false accept rate (FAR), false reject rate (FRR) characteristics, equal error rate (EER) analysis, cumulative match curves (CMC), probability density function histogram analysis (HIST) and ROC curves (Receiver Operating Characteristic), curves that provide an analysis of the trade-off between FAR and FRR characteristics. These features provide a comprehensive analysis of the performance of biometric systems as well as providing information on how business rules may be applied to optimise the performance of authentication systems based on the speaker verification engines being evaluated.

The original face recognition tool was modified by Biometix to meet the specific requirements of the speaker verification evaluation. Modifications were made to the reporting function to allow direct comparison between the responses of different engines to various evaluation scenarios. The tool's "import function" was modified to handle the "t-script" format developed for the evaluation, including modifications to process failure-to-acquire (FTA) and failure-to-enrol (FTE) cases reported by the engines.

A scatter graphical analysis feature was also added. This allows the results of two or three verification processes or engines to be displayed as a two or three dimensional scatter diagram. Points on the scatter diagram could also be tied back to demographic information stored in the original MySQL database, allowing a graphic inspection of the evaluation results against certain demographic information, specifically gender and age.

Figure 4 shows a screen shot of the Performix tool used in the evaluation. This screen shot showing the FAR and FRR characteristics plotted for three engines and the equal-error-rate (EER) performance for each of the engines.

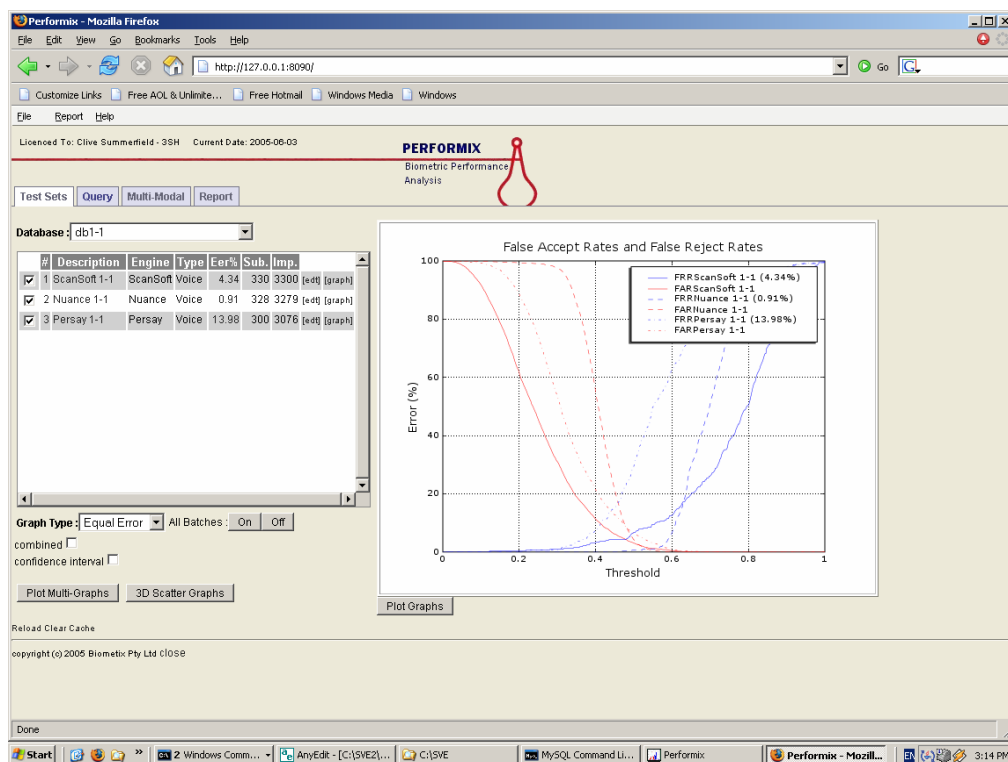


Figure 4. Performix Biometric Evaluation Tool

The EER performance reported by Performix was used as the foundation of the evaluation. The EER result reported by Performix was then imported into a spreadsheet and the results in the spreadsheet were used to report on EER performance described in this report.

2.4.3 Speech Database Audit

Prior to commencing evaluations an audit of the speech database was undertaken. The audit used the Emu tool to select each of the speech samples from the database to listen and inspect each speech sample to ensure that it complies with the description and specification required in the evaluation process.

Where the speech sample was either wrong, of insufficient quality or had a fault such as being prematurely truncated, the sample was marked as “bad” in the speech database. If the speech sample complied with the description and specification, it was marked “good”.

Once complete, this process then allows the total number of speakers with a complete inventory of “good” speech samples to be determined. Out of the original 567 speakers, around 330 speakers were identified with complete inventories of “good” speech samples.

These speakers were selected for the evaluation.

2.5 Processing Speech Samples for Evaluation

Prior to running an evaluation, speech data had to be processed, formatted and made available to the verification engines for the evaluation.

This process was performed by Snack scripts that were produced to extract the speech information from the database, convert the A8 speech data to a linear data format and store them in flat file structures for submission to the speaker verification engines for processing.

Where required the speech processing function also included mixing noise and converting the speech signal or the mixture of speech and noise into mobile telephone for evaluation.

These two processes were performed by specially prepared Snack and Visual Basic (VB) programs that performed these functions.

2.5.1 Creating Noisy Speech Samples

Snack scripts were developed to mix speech file and noise samples to produce noisy speech samples for the evaluation test cases. The mixing process involved measuring the power in a sample of noise material and in the selected speech material. This information was used to scale the noise and speech samples and then adding the scaled waveforms together to create speech at the required signal-to-noise ratio.

The evaluation test cases stipulated noise at 0 db and -20 db. That is speech and noise at the same power level (0 db) and signals where the noise was 100 times the power of the speech signal.

As example of the process for city noise can be seen in Figure 5. This shows Emu displays of the city noise and speech with a signal-to-noise ratio of 0 dB, (equal power of speech and noise) and of -20 dB, (noise power is 100 times speech power).

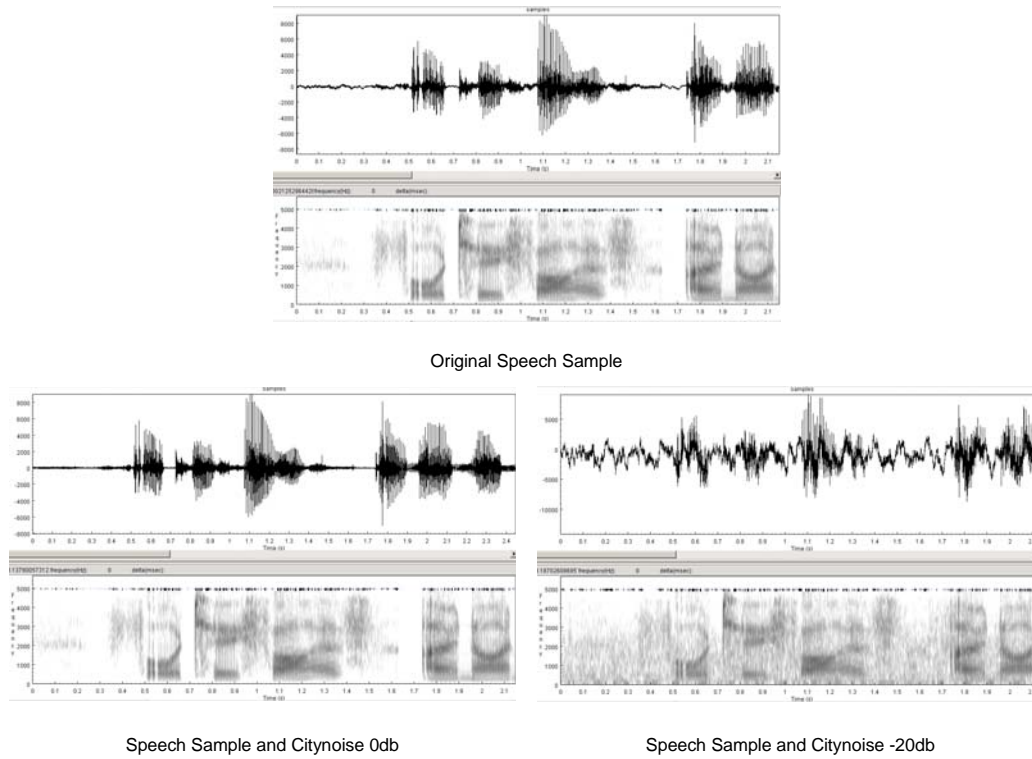


Figure 6. Process for creating Noisy Speech Samples

2.5.2 Creating Mobile Telephone Speech Samples

Mobile and cordless telephone speech was simulated by processing the speech and noise material through suitable software codec.

Figure 6 shows the process for simulating the mobile telephone channel. It contains three sections:

1. An encoder algorithm (Adaptive Multi-Rate), which converts the original high quality speech data to a low bit rate format used in mobile telephone communications
2. The communications channel simulator, which simulates the communications channel between the handset and the base station and allows the bit error rate to be adjusted to simulate the various degrees of noise in the communication link
3. The decoder algorithm, related to the original encoder algorithm, which converts the low bit rate digital data back into a speech signal for use in the evaluation of the speaker verification engines.

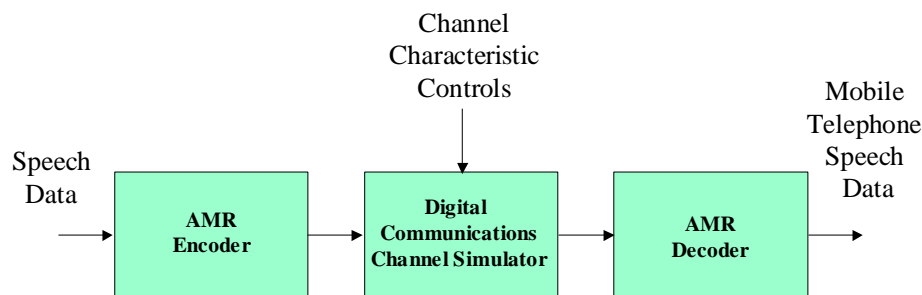


Figure 7. Mobile Telephone Processing (using example AMR algorithm)

The process allows the impact of the mobile coding algorithms on verification performance to be independently examined. It also provides an environment for assessing the impact of the communications channel on speaker verification in mobile telephone environments.

A software AMR mobile telephone simulator was obtained from Sony/Erricsson. This codec simulated G3 mobile telephone environments offering 9 different coding standards ranging from 12.2 kb/s to 4.75kb/s.

The evaluations tested three codec bit rates:

1. A high quality 12.2kb/s data rate codec
2. The current GSM standard 6.7kb/s data rate codec
3. A low bit rate 4.75kb/s codec, which is likely to be introduced by the carriers as a part of dynamic bit rate control to handle high congestion trunks (as opposed to the current response of dropping the call).

A Visual Basic (VB) program was developed to operate the codec and process the speech signal at the selected data rates. A Snack script was run to select and format the speech signal into a linear format for AMR processing. Where required the noise mixing process was also run to add noise at the required level. Once the speech data was converted to the appropriate format the VB program was run to generate the mobile telephone versions of the speech materials at the required data rate required for the evaluation.

All evaluations assumed an error free communications channel. The evaluations only report the impact of different AMR data rates on engine performance. Furthermore, noisy as well as original clean speech samples were passed through the AMR codec software to simulate the impact of noisy environment combined with the mobile communications encoding, a common scenario expected in deployed environments.

2.5.3 Creating Speech Samples for Text-Independent Evaluations

The speech database was set-up to contain single utterances containing specific speech data used for the evaluation. This is ideal for evaluation of text dependent engines that require the same utterance to be enrolled as used in the verification process. In this mode, the utterances can be selected from the speech database and used directly in the evaluation process.

However, a different strategy was required for evaluation of text-independent engines. For text independent engines, enrolment speech files were created by concatenating all speech items from the enrolment group. Similarly, the verification speech files were created by concatenating speech items from the verification set. A single speech file was created for enrolment and for verification.

Mobile telephone versions of the concatenated speech files were produced by processing each of the concatenated speech files through the mobile telephone codec software in the same way as for the text

dependent evaluations. Noise samples were similarly produced by mixing the appropriately scaled noise material with the concatenated speech files to create the noisy speech files used to evaluate the text-independent verification engines.

2.6 Evaluation Environment and Set-up

Figure 7 shows the configuration of the speaker verification evaluation environment.

Each engine was configured on a separate PC and networked to the server systems. Speech data were selected from the server and supplied to each of the speaker verification engines operating independently of each other.

During the enrolment processing, copies of enrolment speech data were supplied to each of the speaker verification engines and the enrolment process initiated for each of the engines independently of one another. The speaker templates produced by each engine as a result of the enrolment process were then stored locally on that PC.

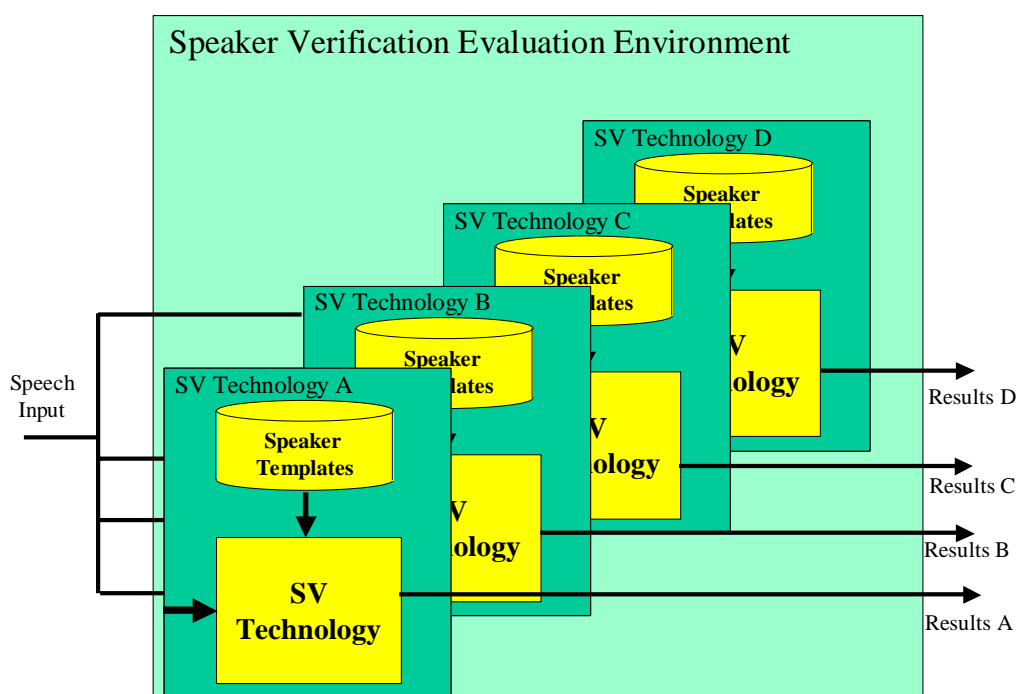


Figure 8. Speaker Verification Evaluation Environment

During the evaluation process, speech data was copied from the server to each of the verification engines separately and the results of the verification process stored locally. Once complete the results were transferred to the server systems and loaded into the Performix biometric evaluation tool for processing and analysis.

Figure 11 shows the hardware system set-up used to implement the evaluation environment.

Hardware comprises two Unisys ES3120 database servers running Microsoft Windows 2003, networked via a D-link switching hub with separate PCs supplied by the engine vendors running the speaker verification engine.

The servers were configured with the speech database (as shown in Figure 2), the results database, Emu and the Performix biometric reporting/evaluation software.

The server ran the Snack and VB scripts to select and prepare speech samples.

M and t-scripts produced to control the enrolment and verification engines were produced on the server and transferred to each verification PC prior to operating the process. Speech samples extracted from the speech database were submitted to each of the engines under evaluation via the network connection. Results generated by the verification engines were transmitted back to the server, via the network at the completion of the evaluation and loaded into the Performix tool for analysis.

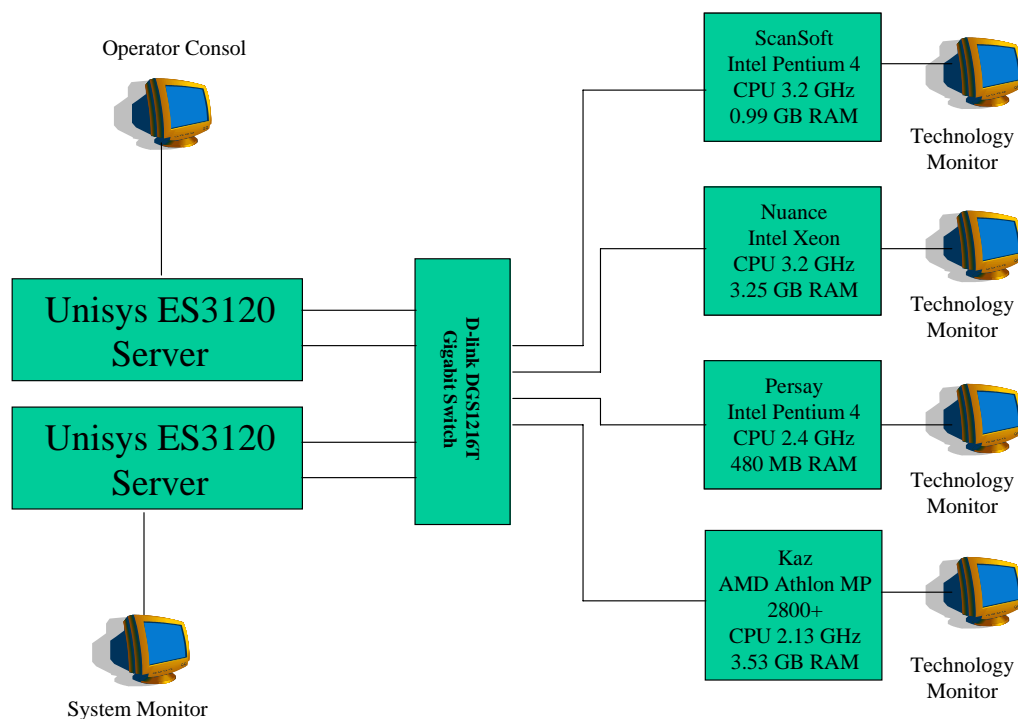


Figure 11. Evaluation hardware set-up

2.7 Summary of Evaluation Environment and Processes

This section describes the environment and processes used to conduct the evaluation. The environment comprises a speech database stored in MySQL format, noise materials and speech signal processing and mobile telephone simulation software to select, process and format speech and noise samples for enrolment and verification. The environment also supported a modified version of the Performix biometric evaluation tool that was used to analyse verification performance and measure equal-error-rate (EER) performance for each of the verification engines involved in the evaluation.

The next section describes how selected Test Cases were set-up, administered and performance measured using this environment.

3 EVALUATION TEST CASES

3.1 Description and Designations

A suite of tests cases was prepared to evaluate the speaker verification engines' performance across a number of operating conditions and configurations. These included testing and evaluating the performance of each of the speaker verification engines in different telecommunication environments (i.e. landline and mobile telecommunication environment) and different noise conditions and scenarios.

Two separate test case suites were prepared, one to evaluate text-dependent engines, the other to evaluate text-independent engines.

In each test-case, baseline performance was established. Once established, the performance of the engines was reported with reference to the baseline.

Baseline performance was established by doing a full impostor analysis. That is, processing every model against the client and all other enrolled clients posing as impostors.

Based on selecting around 300 "good" speakers from the database, the baseline tests involved enrolling 300 speakers and then testing each enrolled template against the client and 299 impostors. This test is referred to as a "300x300" test as it requires 300x300 (90,000) verification processes to be performed to complete the test case.

Test cases involving 90,000 verification processes can take many hours (in some cases many days) of CPU to complete and are impractical to administer in a large evaluation study. The 300x300 test case size is a highly redundant configuration with an over supply of impostor test conditions. Performance of the engines can be adequately determined by considerably reducing the impostor testing. For regular evaluation, the test case size was reduced to 300x11, involving testing against the client plus 10 randomly selected impostors of the same gender. In this case the number of verification processes was reduced from 90,000 to 3,300 (300x11) tests enabling (most) engines to complete the test-cases in 10-20 minutes.

The design of the test cases involved performing a single 300x300 test-case and the corresponding 300x11 test-case to obtain the baseline performance to check the consistency between these two test case configurations. Once this was complete testing for noise, mobile, sibling and other conditions was performed on the reduced test case configuration and referenced back to the baseline result. (In the case of siblings, because of the limited number available, these test cases are configured as a 60x60 to obtain baseline sibling performance, and then 60x2 where each model is only tested against their sibling.)

3.2 Text-Dependent Test Cases

Twelve text-dependent test conditions were identified to evaluate specific performance attributed of the engines for specific Government applications, not all of which are reported in this document. Table 3 summaries the test case conditions and numbering arrangements for the text-dependent tests. Some test case conditions specify only a single test case, whilst other specified a number of different test-cases. Test case condition 1 only has a single test case that is evaluating the technology for speakers counting 1to9 (test case 1-1). However, test condition 4 specified 3 test cases (4-1, 4-2 and 4-2A) where impostors said the same name, a different name and where the client is configured as an impostor saying a different name to that enrolled.

Test Case Designation	Description	Test Size
1-1	Speech Database Item 3 - 1to9 Landline and mobile telephone tests White noise; office noise; city noise and shopping centre noise	300x300 300x11
3-1	Speech Database Item 3 - 1to9 Longitudinal speech data Landline and mobile telephone tests White noise; office noise; city noise and shopping centre noise	300x11
4-1	Speech Database Item 2 – Name Landline and mobile telephone tests White noise; office noise; city noise and shopping centre noise	300x300 300x11
4-2	Speech Database Item 2 – Name and Item 5 - Friend’s Name Landline and mobile telephone tests White noise	300x11
4-2A	Speech Database Item 2 – Name and Item 5 - Friend’s Name Landline and mobile telephone tests White noise	300x11
6-1	Speech Database Item 2 – Name longitudinal speech data Landline and mobile telephone tests White noise;	300x11
7-1	Speech Database Item 3 – 1to9 sibling speech data Landline	60x60
7-2	Speech Database Item 3 – 1to9 sibling speech data Landline	60x2
9-1	Speech Database Item 3 - 1to9 Reduced Enrolment (2 samples) Landline	300x11
9-2	Speech Database Item 3 - 1to9 Reduced Enrolment (single sample) Landline	300x300 300x11
12-1	Speech Database Item 3 - 1to9 Mobile Enrolment Landline	300x11
12-2	Speech Database Item 3 - 1to9 Mobile Enrolment Mobile	300x11

Table 3. Test Cases and Designations

A number of other test cases were also included to check consistency of the evaluations results. These included tests to confirm if any engine adaptation/optimisation processes were active and test with small number of live mobile and cordless phone speech data sample to contrast with simulated mobile telephone speech data. Adaptation testing showed no adaptation was taking place within the algorithms. Also there was good correspondence between results of the small amount of real mobile telephone speech

data in the speech database when compared against the simulated mobile telephone environment used in the evaluation.

3.3 Text-Independent Test Cases

Given that text independent engines do not process content, only voice quality, the number of test conditions is significantly reduced.

Only two test cases were established from text-independent evaluations, designated test case 1-1 and test case 2-1.

Test case 1-1 replicated the test cases for text dependent test case 1-1, but using concatenated speech files made up of all items of speech from the database, not just counting 1to9 as in the text dependent test conditions.

Test case 1-2 evaluated the enrolment based on reduced speech material. This test was performed by only enrolling limited items and speech and measuring the FTE (Failure to Enrol).

4 EVALUATION PROCEDURES – TEST CASE SET-UP

4.1 Introduction

This section describes the process for setting up and administering the test cases described in Section 5.

The evaluation involved two stages:

1. Enrolment
2. Verification

The enrolment process was defined and controlled through the production of a script file called an m-script. This script defines the model (or template) to be produced for a given client and the speech data files used to produce that model or template.

The testing of the verification function for a specific test case is defined by the t-script. This defines a client model, created using the m-script, and the speech files to be verified against that model. The format of the t-script also allows the verification engine to return the results of the verification function. At the end of the verification process, the output t-script is imported into Performix to compute the False-Accept and False-Reject rates and the Equal-Error Rate for the engine in that test condition.

4.2 “m-script” and “t-script” Formats

Table 4 shows the format of the m-script which controls the enrolment procedure.

The script defines the model name and the speech files used by the verification engine to create that model. The speech files include the relative pathname and the label defining the type of utterance or speech item.

The format enables vendors to return a “Failure to Enrol” error (FTE) against the specified speech file in the event that the engine returns such an error. the format also allows the verification engine to append an optional explanation relating to the error, for example “sample_too_short”.

Similarly, the verification process and test-case is controlled by the t-script. An example t-script is shown in Table 5. The script defines the model used in the verification process and the speech file against which the verification function is performed. Like the m-scripts, this file format also includes a definition of the speech item.

The result produced by the verification engine is then inserted after the speech item label. This information is then used by Performix to plot the False Accept/False Reject characteristics for each of the engines in the evaluation and return the Equal-Error-rate performance for that each test case.

Like the m-script, provision is made in the format to enable the engines to report Failure to Acquire errors, writing a FTA in the output script, with an optional explanatory comment, such as “sample too small”.

Through this process, m-script and t-script files can be produced which implement each of the test cases described in Section 5.

m0011
pathname\w00123456.wav CustomerReferenceNumber
pathname\w00654321.wav CustomerReferenceNumber
m0013
pathname\w00135642.wav CustomerName
pathname\w00246531.wav CustomerName
m0099
pathname\w00147852.wav Counting 1to9
pathname\w00258741.wav Counting 1to9
m0109
pathname\w00147852.wav DateofBirth
pathname\w00258741.wav DateofBirth
m0299
pathname\w00147852.wav FriendsName
pathname\w00258741.wav FriendsName
m0399
pathname\w00147852.wav PIN
pathname\w00258741.wav PIN
m0499
pathname\w00147852.wav Location
pathname\w00258741.wav Location

Table 4. Sample m-script defining enrolment

m0011	pathname\w00125690.wav CustomerReferenceNumber	0.000000e+000
m0013	pathname\w00096521.wav CustomerName	0.000000e+000
m0099	pathname\w00122566.wav Counting 1to9	0.000000e+000
m0109	pathname\w00125690.wav DateofBirth	0.000000e+000
m0299	pathname\w00096521.wav FriendsName	0.000000e+000
m0399	pathname\w00122566.wav PIN	0.000000e+000
m0499	pathname\w00125690.wav Location	0.000000e+000

Table 5. Sample t-script defining enrolment

4.3 Process for Producing M-scripts and T-Scripts

The m-scripts and t-scripts were produced using a custom developed software program that extracts the appropriate speech data from the data, produces the corresponding m and t-scripts and applies these to the engines under evaluation.

Appendix A gives an example of the software program produced to generate an m-script for the name test (test case 4-1).

The software program commences by extracting the required speech data from the speech database and confirming that the speech samples have been verified and comply with the test case conditions. The speech data is copied into wav files and made ready for administration of the evaluation process. In the case of evaluations using mobile or noisy speech data, a separate VB (Visual Basic) script is run to extract the required speech data and generate the mobile or noise versions as required. At the same time the

program also extracts the meta data containing the demographic information about the speaker and makes this available to Performix in order to be able to map demographic information against the verification result.

The program then proceeds to compile the required m-script (or t-script). Once compiled, the script is submitted to each of the engines and the enrolment/verification process on each engine initiated. The program monitors all engines to detect when each engine finishes.

Once complete, the program reports the process completion and the t-scripts generated are imported into performance for FAR/FRR and EER analysis.

Tables 6 (a) and (b) show an extract from an m-script and a t-script for part of the 1-1 Counting 1to9 Test Case, respectively.

There are two parts to each script, the input script which is used by the engine to enrol and verify speech data and the output scripts which contains error codes and explanations and in the case of t-scripts the verification results returned by the engine for each verification process performed.

In this example, the engine reported two FTE (Failure to Enrol) in the output m-scripts against speech files 471091201-1-3-1.wav and 471091201-1-3-2.wav marking them both as “invalid_utterance”.

Similarly, table 6 (b) shows extracts from the input and output t-scripts used to perform test case 1-1. The output t-script contains the “likelihood scores” returned by the engine in response to the verification test. This is the verification score assigned by the engine relating to the verification of the specified speech file compared against the specified model (created using the m-script file). This information is imported into Performix to produce the FAR and FRR characteristics for each of the engines and to computer the EER performance.

In the example shown, the output t-scripts in table 6 (b) also demonstrates the format when the engine returns an error. In this case the engine under evaluation returned an FTA (Failure to Acquire) error against speech file 424056714-2-3-1.wav with the explanatory label “invalid_utterance”, appended to the record. This enables speech file 424056714-2-3-1.wav and any characteristics of the speech file to be noted.

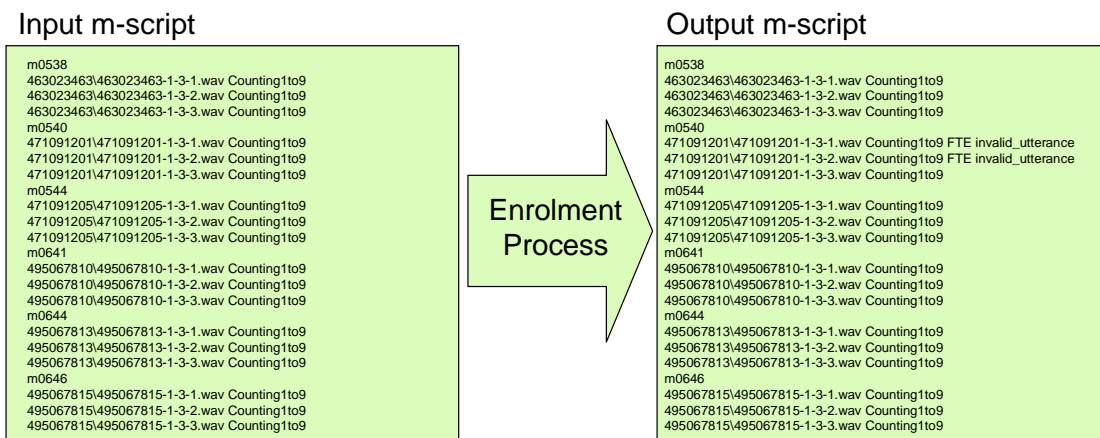


Table 6 (a) Extracts from Input and Output m-scripts (Test Case 1-1 Counting 1to9)

Input t-script

```

m0650 424056708\424056708-2-3-1.wav Counting1to9 0.000000e+000
m0650 424056709\424056709-2-3-1.wav Counting1to9 0.000000e+000
m0650 424056711\424056711-2-3-1.wav Counting1to9 0.000000e+000
m0650 424056714\424056714-2-3-1.wav Counting1to9 0.000000e+000
m0650 424056715\424056715-2-3-1.wav Counting1to9 0.000000e+000
m0650 424056719\424056719-2-3-1.wav Counting1to9 0.000000e+000
m0650 424056720\424056720-2-3-1.wav Counting1to9 0.000000e+000
m0650 425091201\425091201-2-3-1.wav Counting1to9 0.000000e+000
m0650 425091203\425091203-2-3-1.wav Counting1to9 0.000000e+000
m0650 425091204\425091204-2-3-1.wav Counting1to9 0.000000e+000
m0650 428091206\428091206-2-3-1.wav Counting1to9 0.000000e+000
m0650 428091208\428091208-2-3-1.wav Counting1to9 0.000000e+000
m0650 428091209\428091209-2-3-1.wav Counting1to9 0.000000e+000
m0650 428091212\428091212-2-3-1.wav Counting1to9 0.000000e+000
m0650 428091213\428091213-2-3-1.wav Counting1to9 0.000000e+000
m0650 428091216\428091216-2-3-1.wav Counting1to9 0.000000e+000
m0650 428091224\428091224-2-3-1.wav Counting1to9 0.000000e+000
m0650 428091227\428091227-2-3-1.wav Counting1to9 0.000000e+000
m0650 428091229\428091229-2-3-1.wav Counting1to9 0.000000e+000
    
```

Output t-script

```

m0650 424056708\424056708-2-3-1.wav Counting1to9 -1.0558
m0650 424056709\424056709-2-3-1.wav Counting1to9 -1.9424
m0650 424056711\424056711-2-3-1.wav Counting1to9 -1.9716
m0650 424056714\424056714-2-3-1.wav Counting1to9 +0.0000 FTA invalid_utterance
m0650 424056715\424056715-2-3-1.wav Counting1to9 -2.2666
m0650 424056719\424056719-2-3-1.wav Counting1to9 -1.6032
m0650 424056720\424056720-2-3-1.wav Counting1to9 -1.5959
m0650 425091201\425091201-2-3-1.wav Counting1to9 -2.7652
m0650 425091203\425091203-2-3-1.wav Counting1to9 -2.2440
m0650 425091204\425091204-2-3-1.wav Counting1to9 -1.3247
m0650 428091206\428091206-2-3-1.wav Counting1to9 -1.8753
m0650 428091208\428091208-2-3-1.wav Counting1to9 -1.2089
m0650 428091209\428091209-2-3-1.wav Counting1to9 -1.7174
m0650 428091212\428091212-2-3-1.wav Counting1to9 -0.8851
m0650 428091213\428091213-2-3-1.wav Counting1to9 -1.6592
m0650 428091216\428091216-2-3-1.wav Counting1to9 -1.5013
m0650 428091224\428091224-2-3-1.wav Counting1to9 -3.7879
m0650 428091227\428091227-2-3-1.wav Counting1to9 -1.0742
m0650 428091229\428091229-2-3-1.wav Counting1to9 -3.5152
    
```

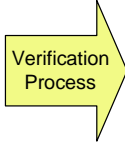


Table 6 (b) Extracts from Input and Output t-scripts (Test Case 1-1 Counting 1to9)

5 EVALUATION RESULTS

5.1 Introduction

The evaluation results are listed in this section under two headings:

- Text-dependent tests
- Text-independent tests

Results of the text-dependent evaluations reported using the test case designation listed in table 1 above.

As there are only two test cases (1-1 and 2-1) for text-independent evaluation, the results are reported under a single heading “Text-Independent Test Case”

5.2 Text Dependent Test Cases

5.2.1 Test Case 1-1 Counting 1to9

5.2.1.1 Baseline Results

Base line results for each of the vendor’s engines are given in Figure 12.

Two baseline results were produced: one for a full 300 by 300 test (Table 1); the other for a standard 300 by 11 test (Table2). The 300 by 300 is a highly redundant where each enrolled client is tested against every other client. This test provides a complete examination of the performance of the various engines, based on the full speech database available for the test. This involved enrolment 300 identities, verifying against themselves and 299 impostors. In total, this involved processing some 90,000 speech files (that is, 300 clients x (1 client + 299 impostors). Given the hardware, this took a considerable amount of time, ranging from a few hours for some of the faster engines to a few days for the slower ones.

Exhaustive impostor testing is actually not necessary to obtain an accurate determination of an engine’s false accept performance. A 300 by 11 test, involving the same 300 identities being enrolled and tested against their own identity plus 10 randomly selected impostors of the same gender was devised. This involved computing verification results for 3,300 speech samples which given the performance of the hardware provided a tractable solution to the limited computation bandwidth available for the evaluation. This 300 by 11 test was established as a standard test configuration for each of the telecommunications and noise testing scenarios.

Theoretically, both tests should give very similar results. To qualify this, a full 300 by 300 test was performed to create a baseline, followed by the reduced test of 300 by 11 to confirm correspondence.

Figure 12 shows the EER performance of Scansoft, Nuance and Persay on the 300 by 300 test, whilst Figure 13 shows the EER performance of the same engines on the 300 by 11 test.

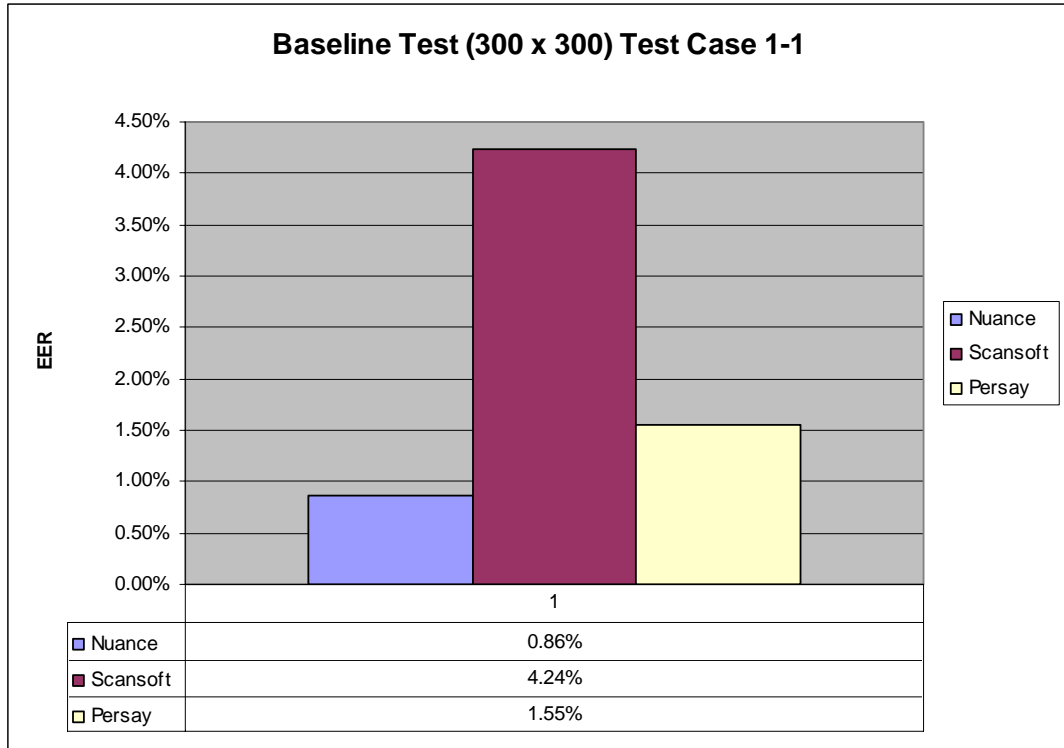


Figure 12. Base line Test 300 by 300 speech samples

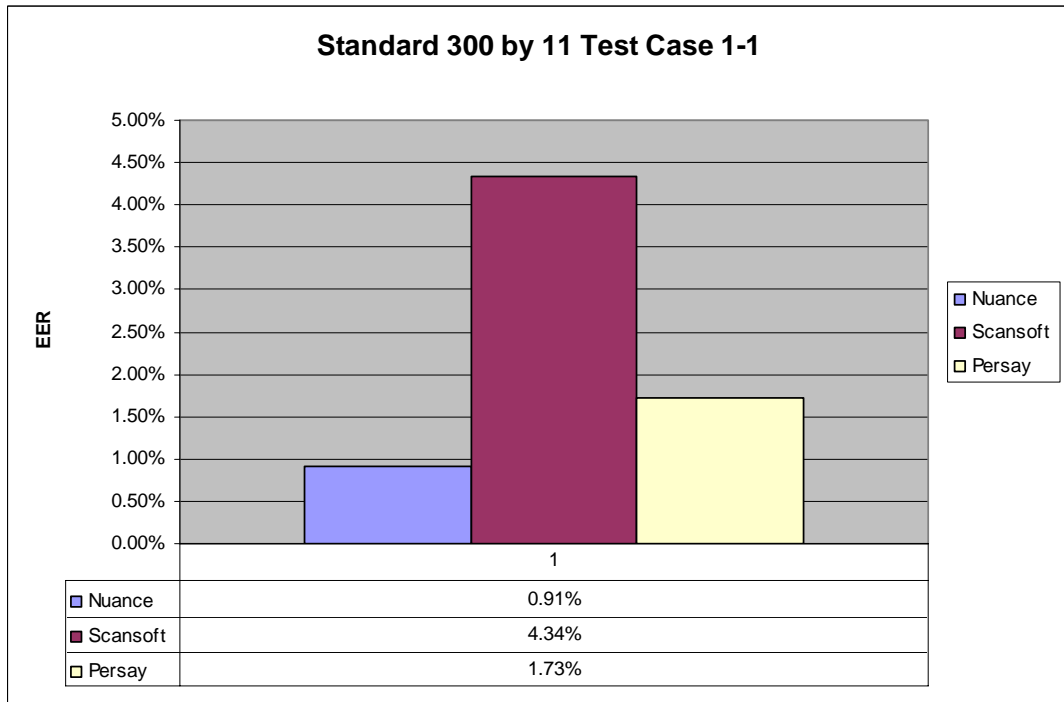


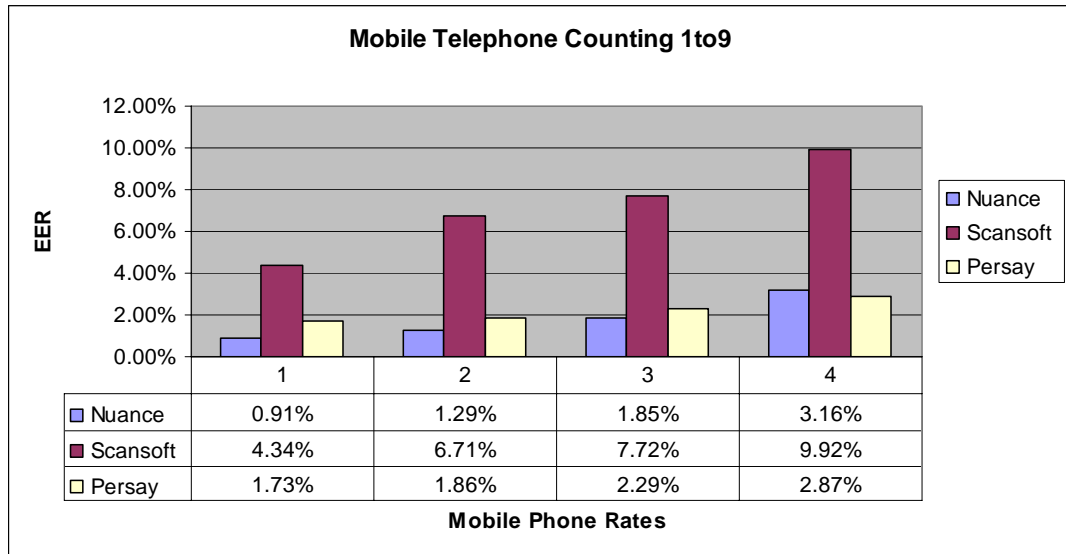
Figure 13. 300 by 11 Speech Samples

5.2.1.2 Observations

In this examination, Scansoft, Nuance and Persay all showed good correspondence between the full 300 by 300 test and the 300 by 11 test, with Scansoft and Nuance EER performance increasing by 0.1% and 0.05%, respectively. In the case of Persay the EER measure increased from 1.55% to 1.72%, (0.17%).

5.2.1.3 Mobile Telephone Performance

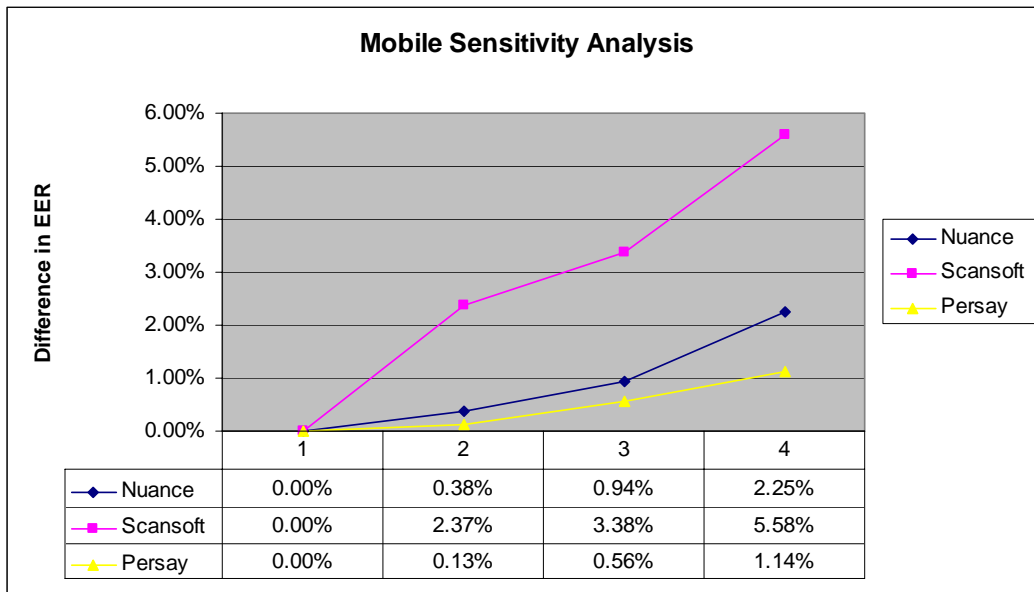
Figure 14 shows the performance of each of the Scansoft, Nuance and Persay engines as the mobile telephone encoding rate was lowered from 12.2 kb/s, 6.7 kb/s to 4.75 kb/s. This was compared against the base rate computed above for land line, which is at a data rate of 64 kb/s.



1=64 kb/s 2=12.2 kb/s 3=6.7 kb/s 4=4.75 kb/s

Figure 14. Performance in Mobile Telephone Communications Networks

All engines showed an increase in the EER performance as the data rate using in mobile telephone networks was decreased. Figure 15 shows the relative performance of each of the engines in this test as the communications data rate is decreased.



1=64 kb/s 2=12.2 kb/s 3=6.7 kb/s 4=4.75 kb/s

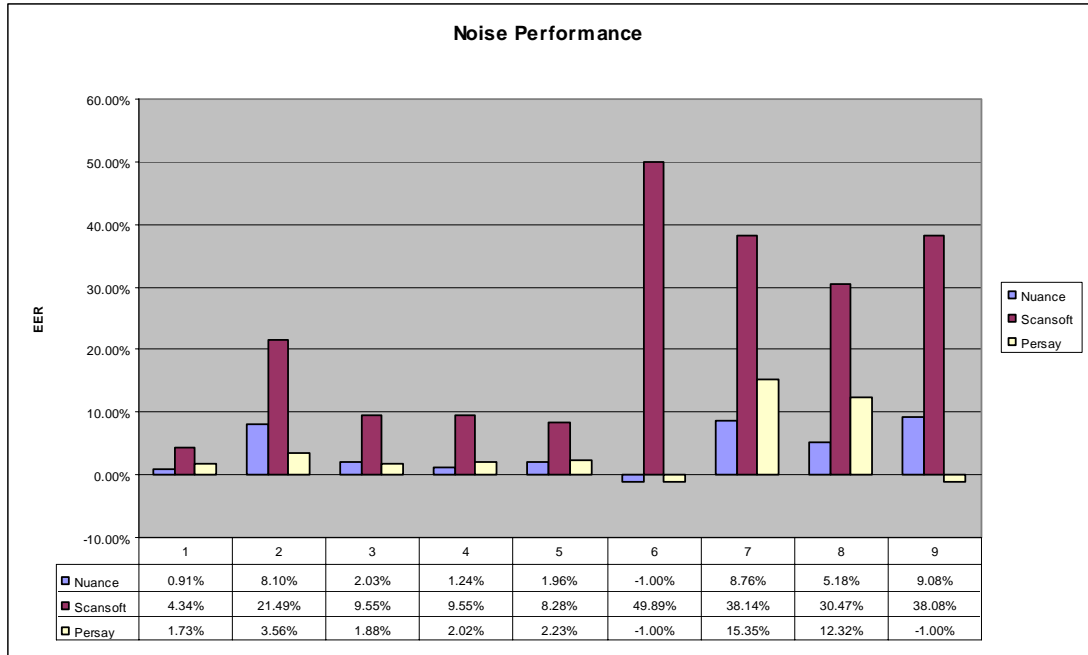
Figure 15. Relative Performance in Mobile Telephone Communications Networks

5.2.1.4 Observations

All engines showed an increase in EER performance as the speech data rate is decreased in mobile telephone environments. Nuance and Persay engines shows considerably more robustness in this environment with an increase of 2.1% and 1.54% in the EER performance between the baseline performance and the lowest data rate of 4.75kb/s, compared to Scansoft engine that showed increases of 4.6% and 5.6% respectively. In addition, Persay showed considerable robustness when moving from a landline to mobile network with only 0.14% increase in EER compared to Nuance and Scansoft who showed a jump of 0.38% and 2.37% in their EER performance in this test.

5.2.1.5 Noise Performance

Figure 16 shows the performance of each of the Scansoft, Nuance and Persay engines in different noise scenarios (white noise, office noise, city noise and shop noise) and as the noise is increased from 0 dB to 20 dB relative to the baseline performance.



1 = Baseline Performance

2 = 0dB White Noise 3 = 0dB Office Noise 4 = 0dB City Noise 5 = 0dB Shop noise

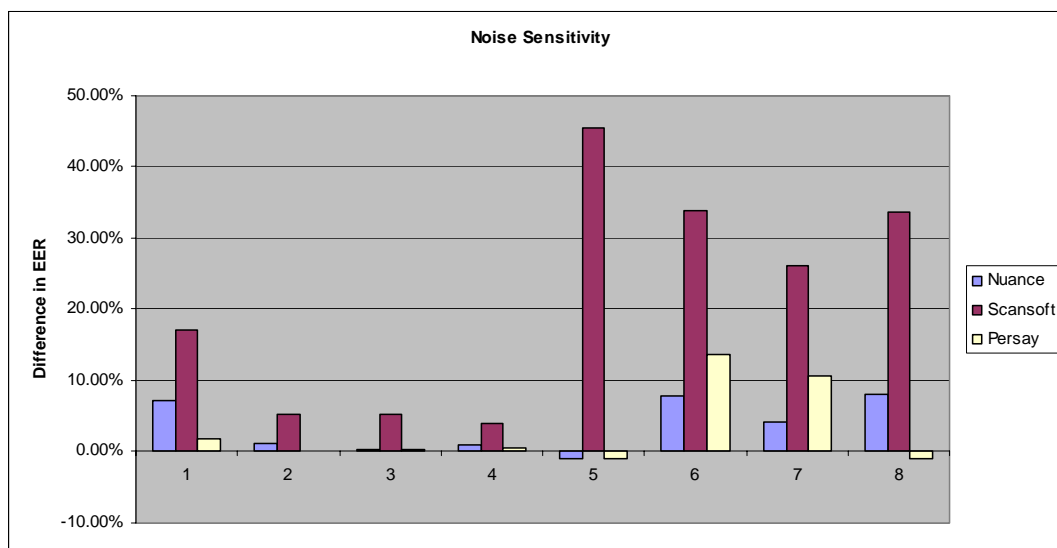
6 = -20dB White Noise 7 = -20dB Office Noise 8 = -20dB City Noise 9 = -20dB Shop noise

(-1.00% indicates FTA returned)

Figure 16. Noise Performance

All engines showed an increase in the EER with increasing noise. However, the EER performance varied considerably for different engines and noise conditions. Nuance and Persay showed considerable robustness over the different noise conditions, when compared against Scansoft.

Figure 17 shows the sensitivity, that is the difference in the EER relative to the baseline performance, as the signal-to-noise ratio is decreased from 0 dB to -20 dB.



1 = 0dB White Noise 3 = 0dB Office Noise 5 = 0dB City Noise 7 = 0dB Shop noise
 2 = -20dB White Noise 4 = -20dB Office Noise 6 = -20dB City Noise 8 = -20dB Shop noise
 (-1.00% indicates FTA returned)

Figure 17. Noise Sensitivity Analysis

5.2.1.6 Observations

All engines displayed an increase in EER performance and some sensitivity to the signal-to-noise ratio but to differing degrees. An analysis of the noise sensitivity shows that both Nuance and Persay were more robust to noise than Scansoft, particularly in the city noise environment, where Nuance and Persay showed a similar level of noise robustness. At high noise levels (-20 dB), specifically in the white noise test, both Nuance and Persay rejected all utterances submitted, returning a FTA (Fail to Acquire). This demonstrates that at high noise levels both engines reject the input signal as outside its processing parameters, whilst the Scansoft engine appeared not to perform any of these tests and returned a potentially poor quality verification score, irrespective of the condition of the input signal. Persay also returned FTA's in the -20dB shop noise test.

Different engines show quite different sensitivities to the different noise scenarios. Whilst white noise was the most problematic for all engines, the Scansoft, Nuance and Persay engines showed significantly different sensitivities to office, city and shop noise scenarios. This is most likely attributed to the various engines' ability to discriminate between speech signals and non-speech signals for the scenario environments. Nuance and Persay had more robust performances in noise scenarios compared to the performance of Scansoft.

5.2.2 Test Case 4-1 – Name Verification

5.2.2.1 Baseline Results

Base line results were produced for condition 1 (different speakers, same name). Two baseline results were produced: one for a full 300 by 300 test (column 1 in figure 18); the other for a standard 300 by 11 test (column 2 in figure 18). As with the 1to9 counting test, the 300 by 300 tests provided a complete examination of the performance of the different engines. This involved enrolment of 300 enrolled identities, and verifying against those identities and 299 impostors for each client. In all, these tests processed some 90,000 speech files which took considerable time for the engines to compute. A 300 by 11 test, involving the same 300 identities being enrolled and testing against their own identity plus 10 randomly selected impostors each saying the same name as the client. This test involved computing verification results for 3,300 speech samples which given the performance of the hardware, provided a methodology that could be computed with the timeframe of the experiment. This 300 by 11test was established as a standard test methodology for each of the telecommunications and noise testing scenarios.

Figure 18 shows the EER performance of Scansoft, Nuance and Persay on the 300 by 300 test (results 1) and the 300 by 11 test (results 2).

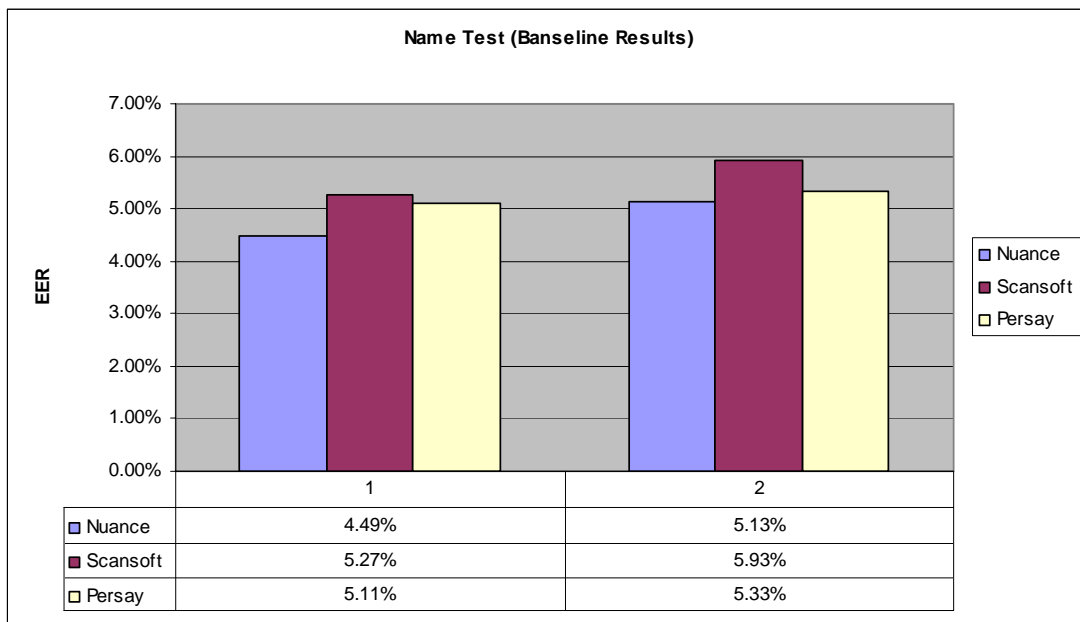


Figure 18. Baseline Tests 300 by 300 (1) and 300 by 11 (2) name speech samples

5.2.2.2 Observations

In this test all engines were much closer in performance with all showing comparable results.

It is also interesting to compare the EER performance of each of the engines for name verification against the corresponding 1to9 counting verification test (1-1 series). Table 7 provides a summary of this comparison:-

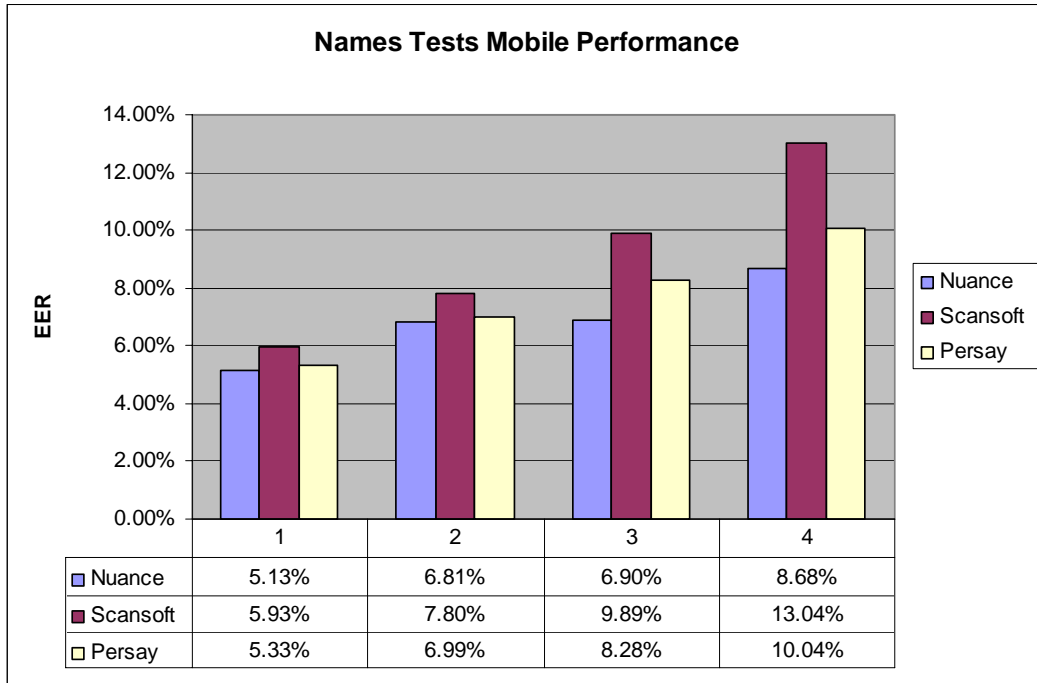
Engine	300by 300 Counting	300by11 Counting	300by300 Names	300by11 Names
Scansoft	4.24%	4.34%	5.27%	5.93%
Nuance	0.86%	0.91%	4.49%	5.13%
Persay	1.55%	1.72%	5.11%	5.33%
Scansoft			+1.03%	+1.59%
Nuance			+3.63%	+4.22%
Persay			+3.56%	+3.61%

Table 7. Comparison of performance of verification engines on counting 1to9 and name verification

This comparison shows that whilst Scansoft’s performance was relatively stable when moving from counting to names, Nuance’s and Persay’s performance fell considerably.

5.2.2.3 Mobile Telephone Performance

Figure 19 shows the performance of each of the Scansoft, Nuance and Persay engines as for names test condition 1, (different speaker; same name) as the mobile telephone encoding rate was lowered from 12.2 kb/s, 6.7 kb/s to 4.75 kb/s. This was compared against the baseline performance for land line computed above, which is at a data rate of 64 kb/s.



1=64 kb/s 2=12.2 kb/s 3=6.7 kb/s 4=4.75 kb/s

Figure 19. Mobile telephone sensitivity tests

Figure 20 shows the sensitivity of the engines as the mobile communications data rate is decreased relative to the baseline performance.

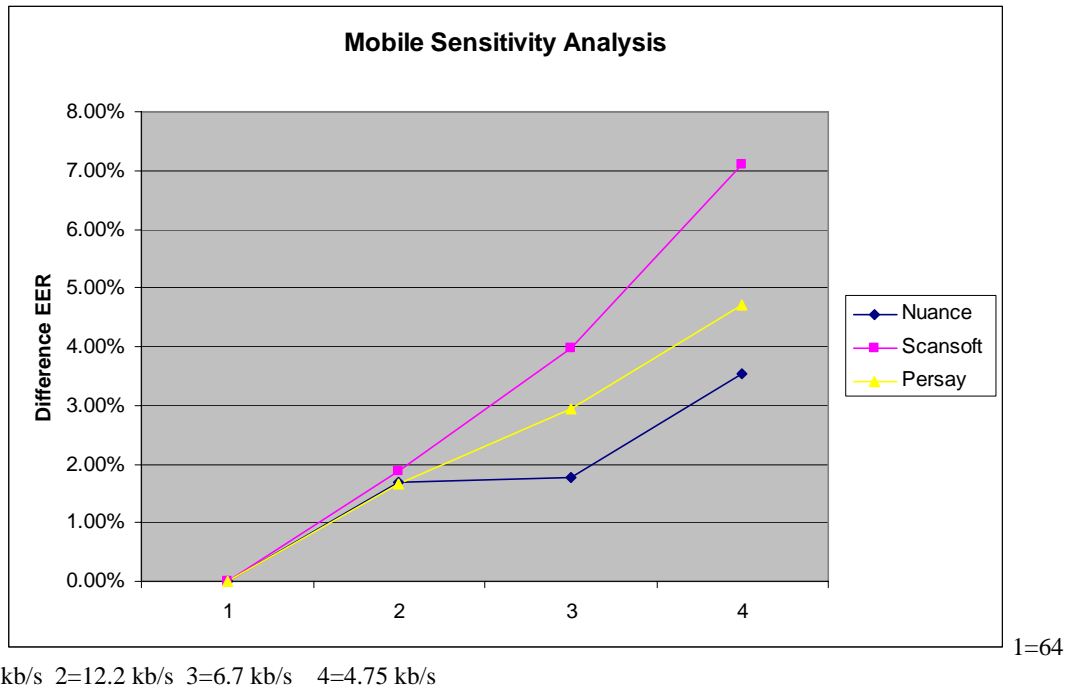


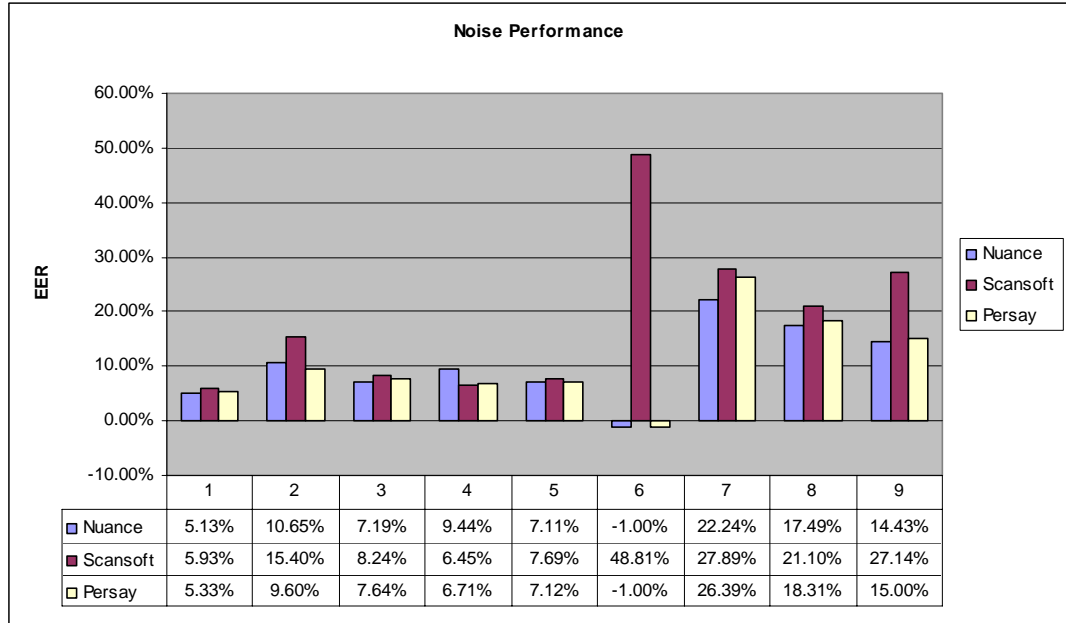
Figure 20. Mobile Telephone Sensitivity Analysis

5.2.2.4 Observations

The analysis shows that all engines exhibit increased EER performance as the mobile telephone data rate is reduced. Nuance and Persay showed more robustness in mobile telephone environments and the performance was relatively stable for higher mobile telephone rates when compared to Scansoft. Whilst the actual increases in EER are somewhat higher than in the counting 1to9, the robustness characteristic reflects that exhibited in the counting 1to9 tests.

5.2.2.5 Noise Performance

Figure 21 shows the performance of each of the Scansoft, Nuance and Persay engines in different noise scenarios (white noise, office noise, city noise and shop noise) and as the noise is increased from 0 dB to 20 dB relative to the baseline performance.



1 = Baseline Performance

2 = 0dB White Noise 3 = 0dB Office Noise 4 = 0dB City Noise 5 = 0dB Shop noise

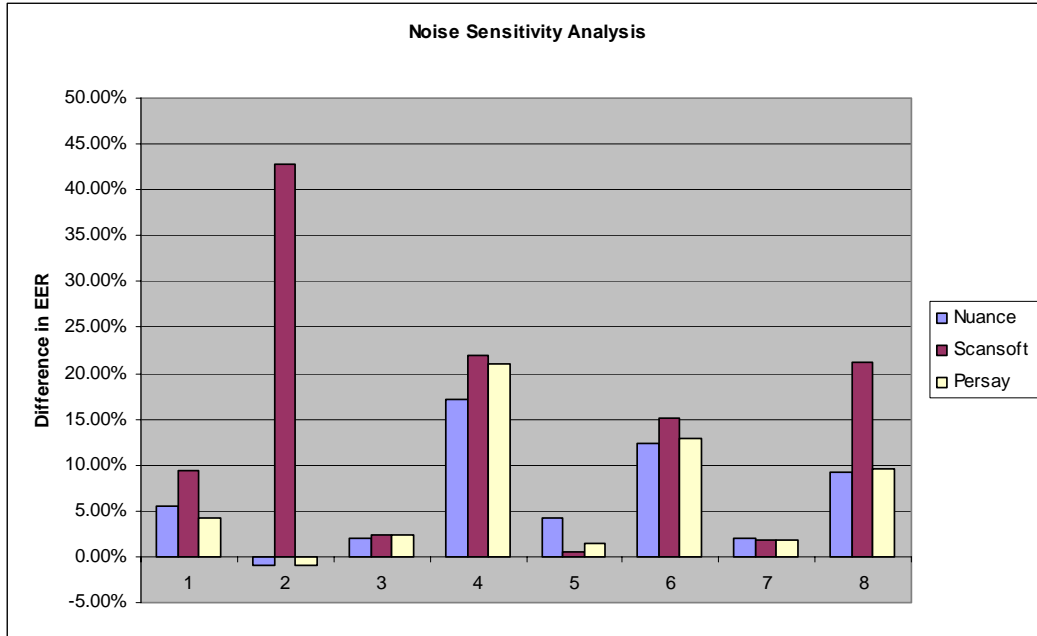
6 = -20dB White Noise 7 = -20dB Office Noise 8 = -20dB City Noise 9 = -20dB Shop noise

(-1.00% indicates FTA returned)

Figure 21. Noise Performance for name verification

5.2.2.6 Observations

All engines showed an increase in the EER with increasing noise. Unlike the counting 1to9 test, the EER performance does not vary widely across the different engines. Again, Nuance and Persay do show marginally higher levels of robustness over the different noise conditions, compared against Scansoft, but not to the same degree as in the counting 1to9 tests. Figure 22 shows the sensitivity, that is the difference in the EER relative to the baseline performance; as the signal-to-noise is decreased from 0 dB to -20 dB.



1 = 0dB White Noise 3 = 0dB Office Noise 5 = 0dB City Noise 7 = 0dB Shop noise
 2 = -20dB White Noise 4 = -20dB Office Noise 6 = -20dB City Noise 8 = -20dB Shop noise
 (-1.00% indicates FTA returned)

Figure 22. Noise Sensitivity Analysis

5.2.3 Test Case 4-2 and 4-2A – Name Verification

Using condition 1 for names in section 7.2.2 above as a baseline, the performance of the engines was examined for condition 2 and 3 name verification. Conditions 2 and 3 are defined as:

Condition 2: Impostors say different names to those enrolled (different speakers; and different names)

Condition 3: Impostors as the enrolled speakers say different names to those enrolled (same speaker, different name)

Figure 23 shows the performance of Scansoft, Nuance and Persay.

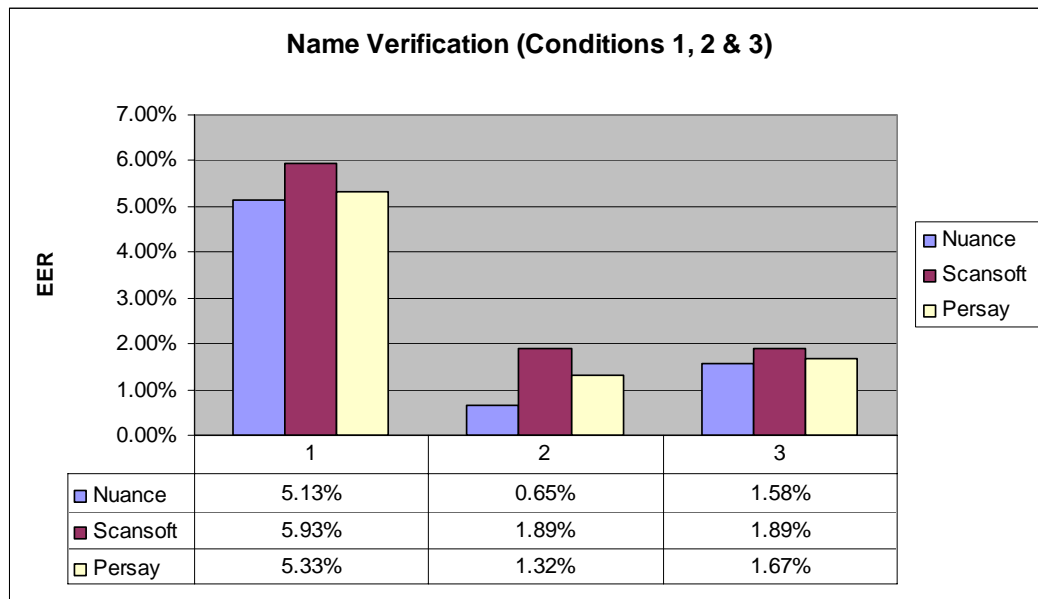


Figure 23. Performance for Name Test Conditions 1, 2 and 3.

5.2.3.1 Observations

All engines showed considerable improvement in EER performance from test condition 1 to test condition 2 where impostors were saying different names to that of the clients, with Nuance showing the strongest discrimination between client and impostor for all the engines tested. Test condition 3 also showed that all the engines were able to discriminate when the client was specified as the impostor but said a different name from that enrolled by that identity. This shows that all the engines in this test exhibited an ability to discriminate when a client says a different name to that enrolled. Indeed this test showed that the engines had increased performance in discriminating when a client says the “wrong name” when compared to an impostor saying the “correct name”.

5.2.4 Test Case 3-1 Longitudinal Speech Data “Counting 1to9” and “Names”

Two longitudinal tests were performed, one “counting 1to9” and saying “Names”.

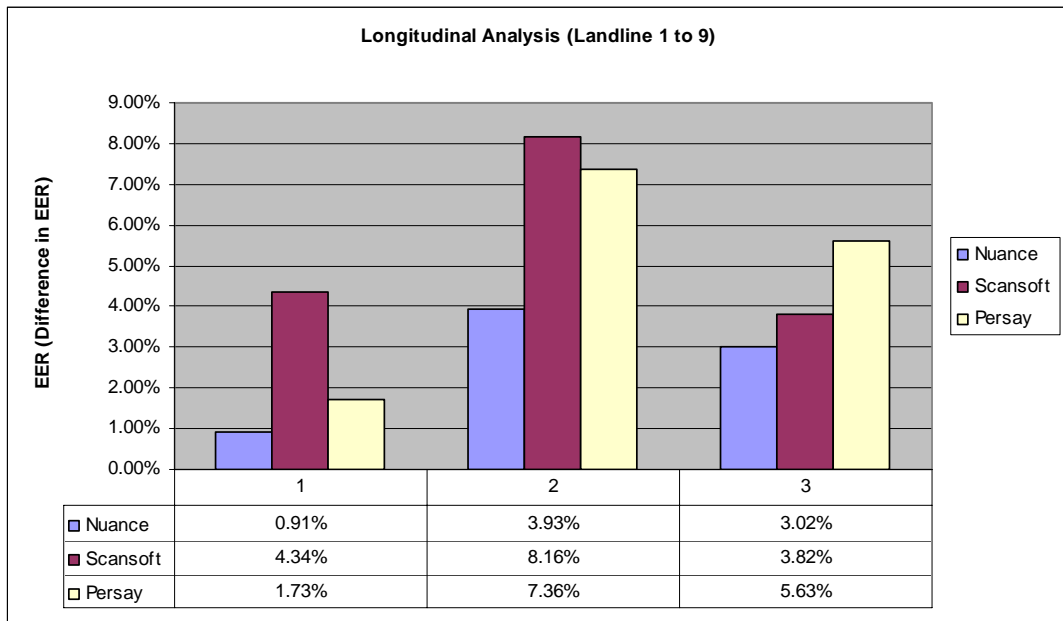


Figure 24. Longitudinal Test Results for landline data, Counting 1 to 9

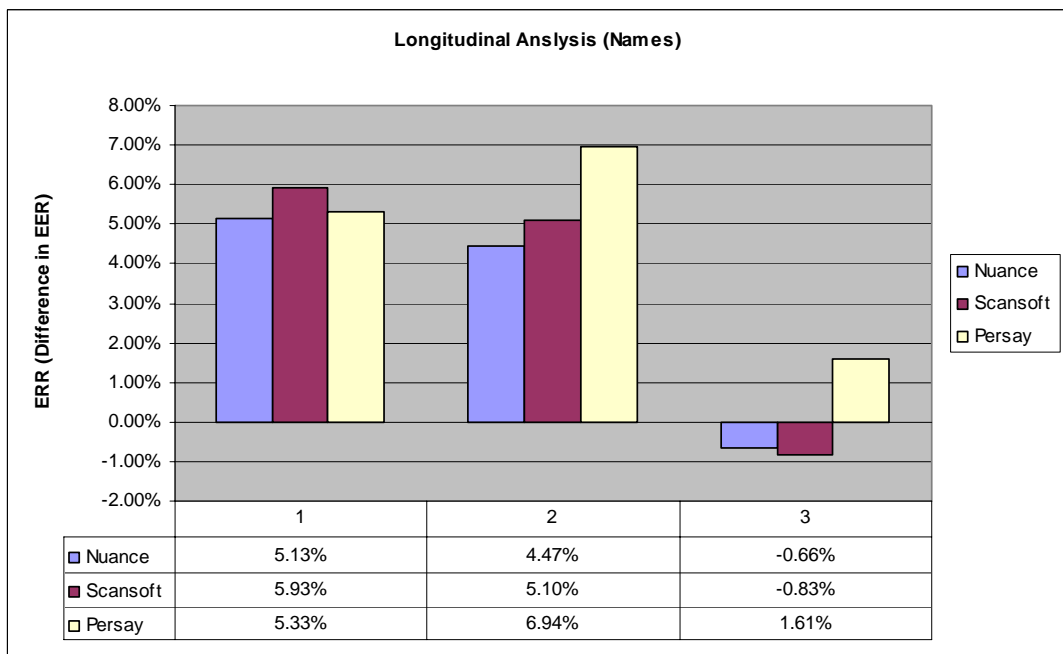


Figure 25. Longitudinal Test Results for landline data, Names

5.2.4.1 Observations

Longitudinal testing for counting 1 to 9, graphed in Figure 24, showed an increase in EER performance of all the engines tested with Nuance, Scansoft and Persay registering 3.02%, 3.82% and 5.63% respectively. In the case of Names tests, shown in Figure 25, the EER of Nuance and Scansoft actually declined slightly, whilst Persay increased marginally by 1.59%. This test shows only slight change in the EER performance of the engines over time for name verification, suggesting name verification is more stable over time than counting 1 to 9 verification.

Test Case 7-1 and 7-2 - Sibling Tests

Figure 26 shows the results of the sibling tests for counting 1to9. This test included only 77 clients. Condition 1 was with all sibling subjects enrolled and all also selected as impostors (eg 77 enrolled, 1 client + 76 impostors authenticating) which then acted as the baseline for this test. Condition 2 tested all enrolled (condition 1) and only their sibling as the impostor (77enrolled, 1 client + 1 sibling impostor authenticating), and the difference was then derived (Condition 3). This then gave the sibling performance. All sibling tests were performed on landline recorded speech data.

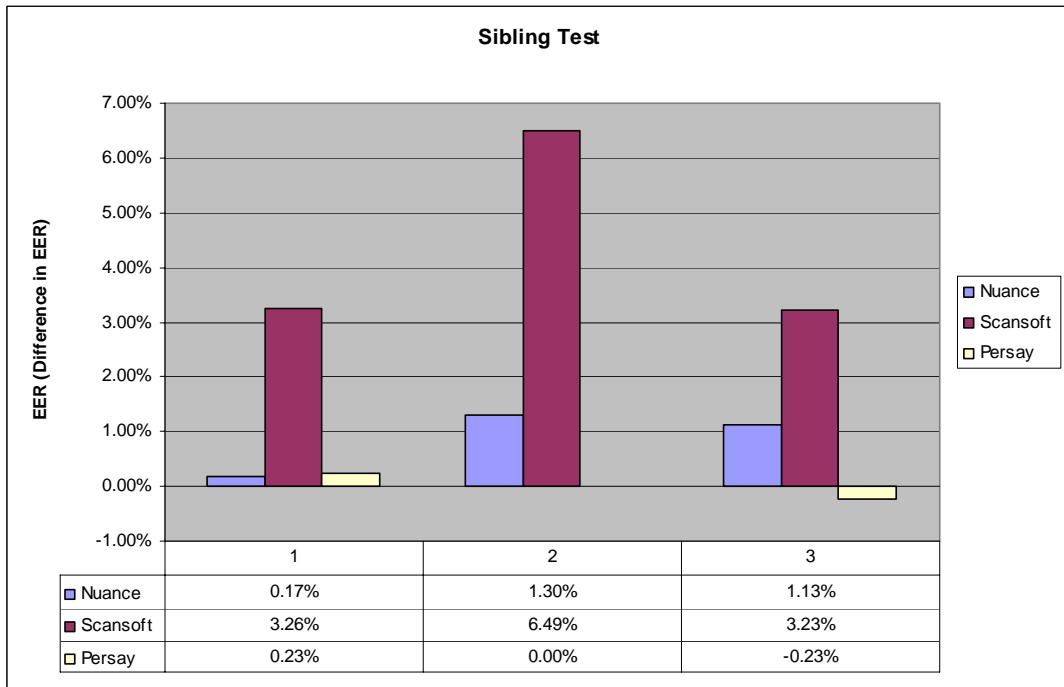


Figure 26. Sibling Test.

5.2.5.1 Observations

All engines displayed some level of sensitivity to siblings, that is, Nuance and Scansoft engines performed poorer on discriminating between sibling speakers, than discriminating between unrelated individuals. However, the sensitivity was not particularly significant, with Persay and Nuance showing the most robust performance in this test.

5.2.6 Test case 9-1 and 9-2 - Reduced Enrolment Tests

There are three enrolment tests performed. These were: -

Condition 2: Enrolment using two samples

Condition 1: Enrolment using a single speech sample

Condition 3: Baseline test using three enrolments

Figure 27 shows the performance of the engines for each of these conditions.

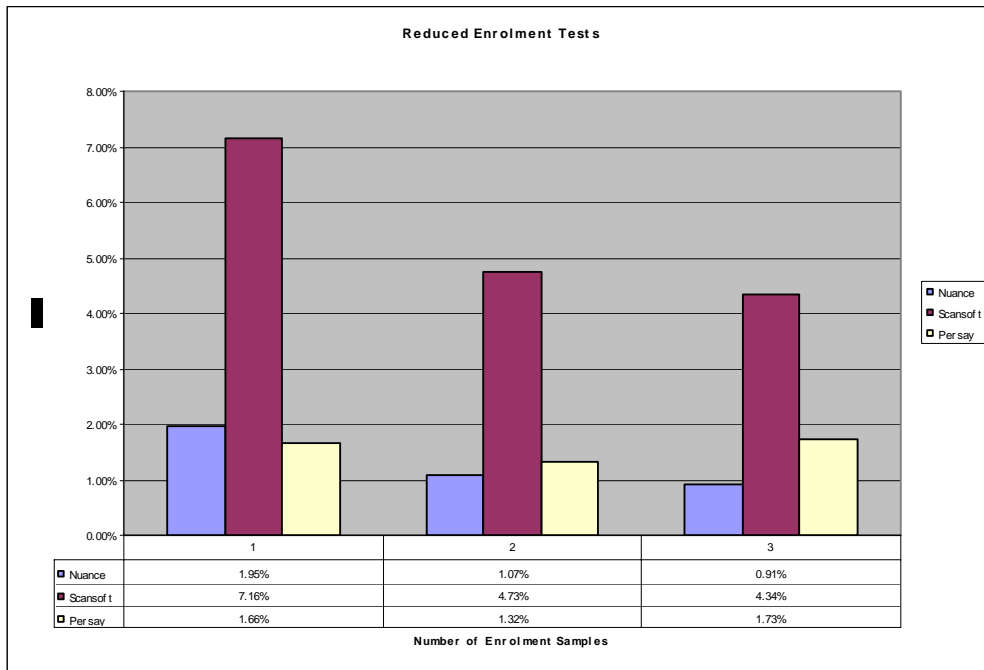


Figure 27. Performance of the engines under reduced enrolment conditions.

Figure 28. shows the relative performance of the engines as the enrolment is reduced from three samples (the baseline) to two to a single sample.

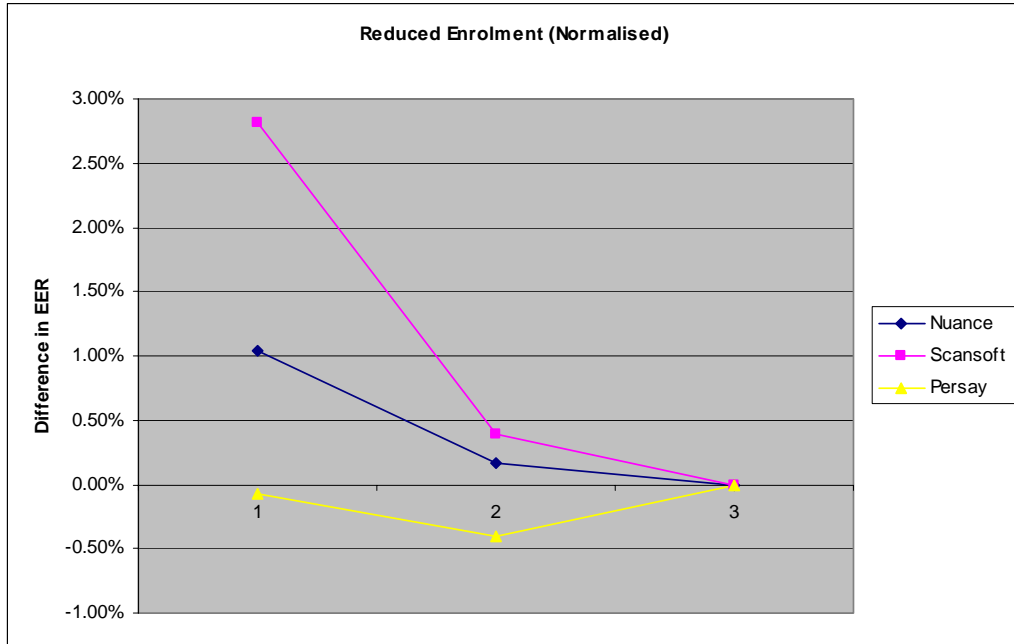


Figure 28. Relative Performance as the Enrolment is Reduced from three samples to two to a single sample

5.2.6.1 Observations

In this test both Nuance and Scansoft show a degradation in performance as the number of speech samples used in the enrolment is reduced. Reducing enrolment from three to two samples increases EER by between 0.25% and 0.75% for Nuance and Scansoft respectively. Reducing enrolment to a single sample shows a significant degradation in performance with EER increasing between 1% and 2.9% for Nuance and Scansoft, respectively, relative to baseline performance. Persay’s performance showed slight improvement with reduced enrolment suggesting that Persay’s engine may be insensitive to the number of samples used in the enrolment procedure.

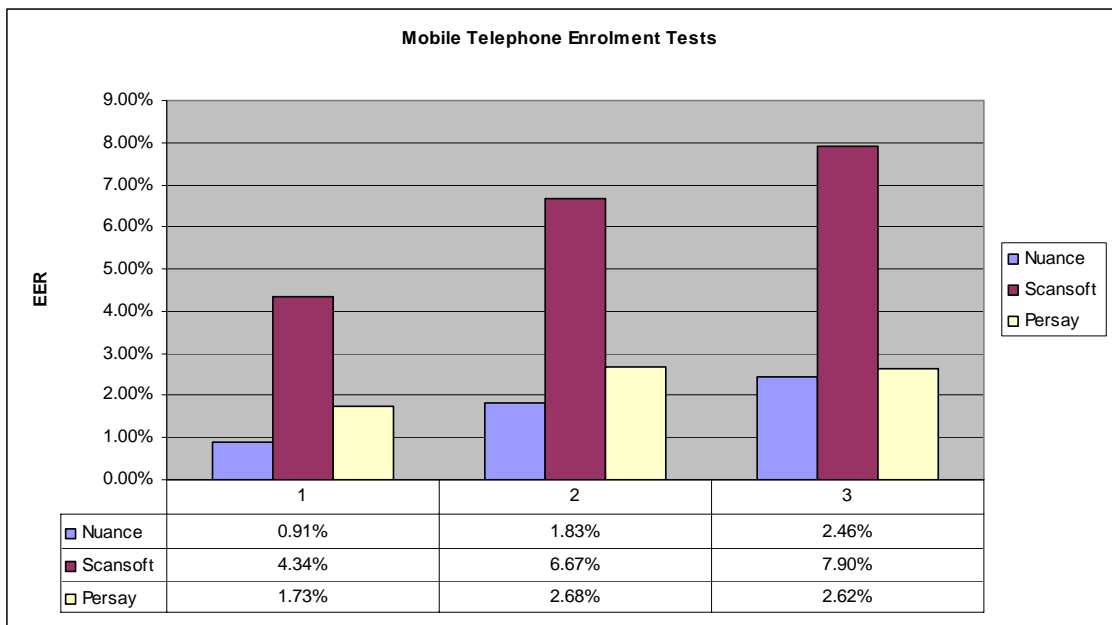
5.2.7 Test case 12-1 and 12-2 - Mobile Telephone Enrolment

Two mobile engine tests were performed: -

Condition 1: Enrol with 6.7 Kb/s mobile telephone codec speech data and tested against 64 Kb/s landline speech data

Condition 2: Enrol with 6.7 Kb/s mobile telephone codec speech data and tested against 6.7 Kb/s mobile telephone codec speech

Figure 29 shows the performance of the engines in the two test condition compared against the baseline test condition (enrol landline: test landline).



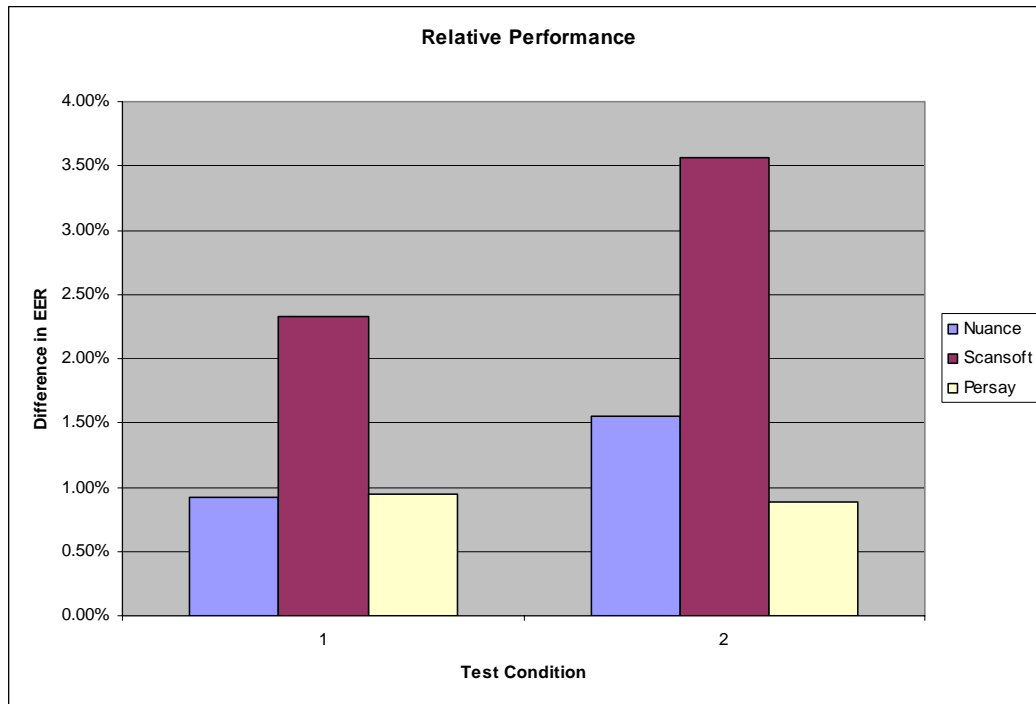
1= Baseline Condition - Enrol landline: test landline

2= Condition 1: Enrol mobile: test landline

3= Condition 2: Enrol mobile: test mobile

Figure 29. Performance of the engines for mobile telephone enrolled speech data

Figure 30 details the performance of the engines relative to the baseline performance.



Test Condition 1 = EER degradation in performance when tested in landline telephone environment

Test Condition 2= EER degradation in performance when tested in mobile telephone environment

Figure 30. Relative performance of the engines for mobile telephone enrolment

5.2.7.1 Observations

All engines showed some level of degradation in performance when enrolled with mobile telephone speech data. Degradation compared to baseline performance were in the region of 1% for Nuance and Persay, and 2.25% for Scansoft (Condition 1).

Degradation compared to baseline performance was also observed when enrolled using mobile telephone speech data and tested in the same communications environment (Condition 2). In this condition, Nuance’s and Scansoft engines’ performance degraded by about 1.5% and 3.5%, respectively whilst the Persay engine performed at about the same level in both test conditions.

5.3 Text Independent Test Case

5.3.1 Baseline Performance

A subset of the test performance on the text-dependent engines was performed on the text-independent systems.

Figure 31 shows the baseline performance of the two text-independent engines tested, Scansoft and KAZ for the 300 by 11 test (condition 1) and the 300 by 300 test (Condition 2).

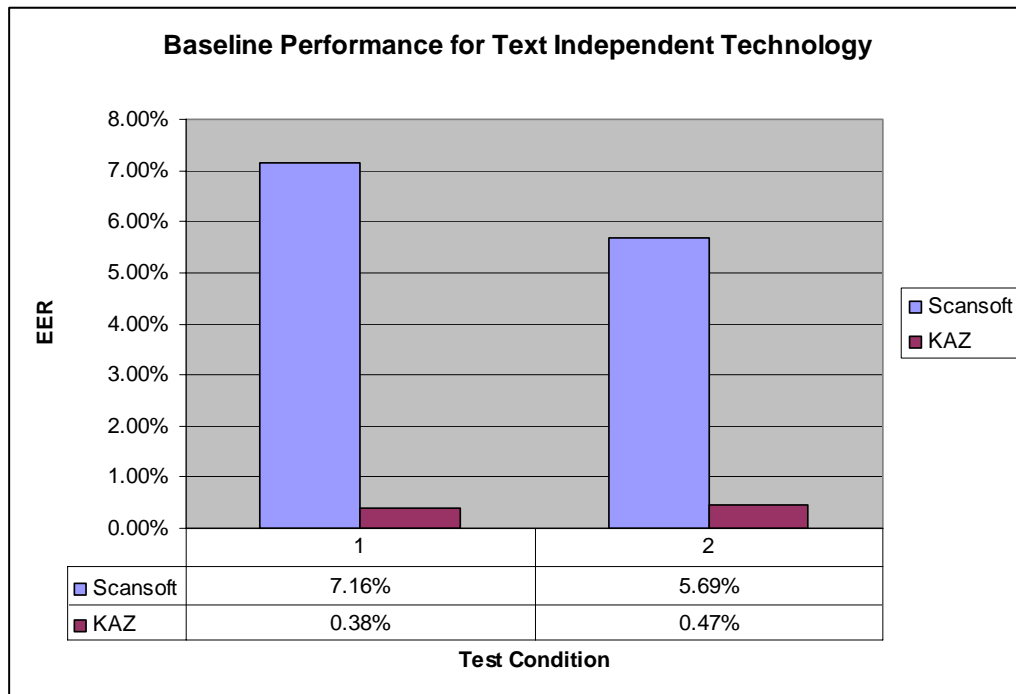


Figure 31. Baseline performance for text-independent engines.

5.3.1.1 Observations

This test shows that KAZ performed considerably better than Scansoft on text-independent test data. KAZ shows some improvement in the standard 300 by 11 test over the full 300 by 300 test, whilst the Scansoft engine showed some deterioration in the standard 300 by 11 test over the full 300 by 300 test.

5.3.2 Mobile Telephone Performance

Figure 32 shows the performance of the text-independent engines for mobile telephone speech data, where condition 1 in the baseline performance (at a speech data rate of 64 kb/s) and conditions 2, 3 & 4 are reduced mobile telephone rates; where 2=12.2 kb/s 3=6.7 kb/s 4=4.75 kb/s.

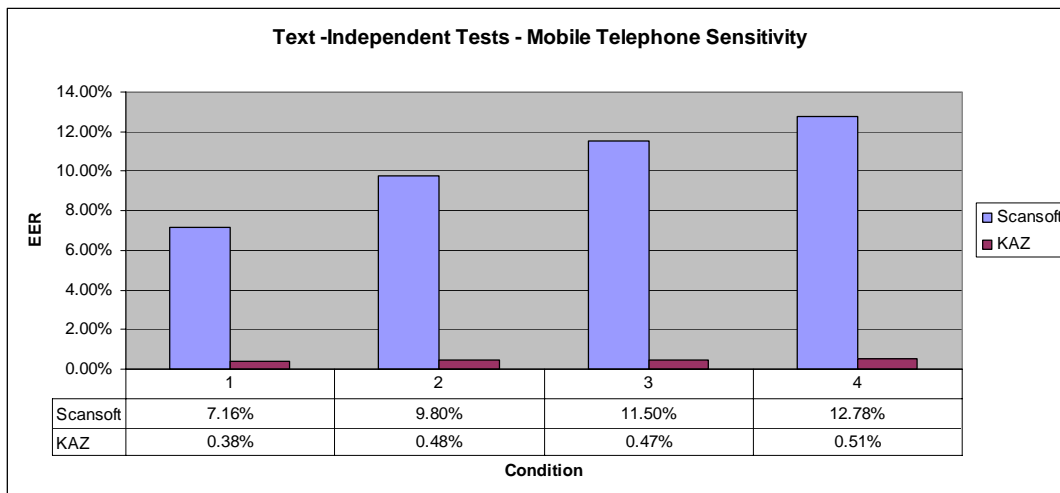


Figure 32. Performance of text-independent engines for mobile telephone speech data

Figure 33 shows the increase in EER performance relative to the baseline performance as the mobile telephone speech data rate is reduced.

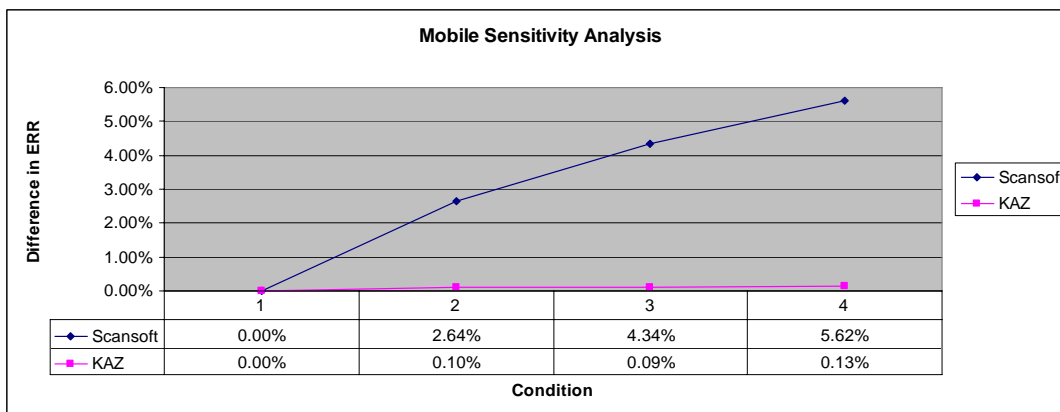


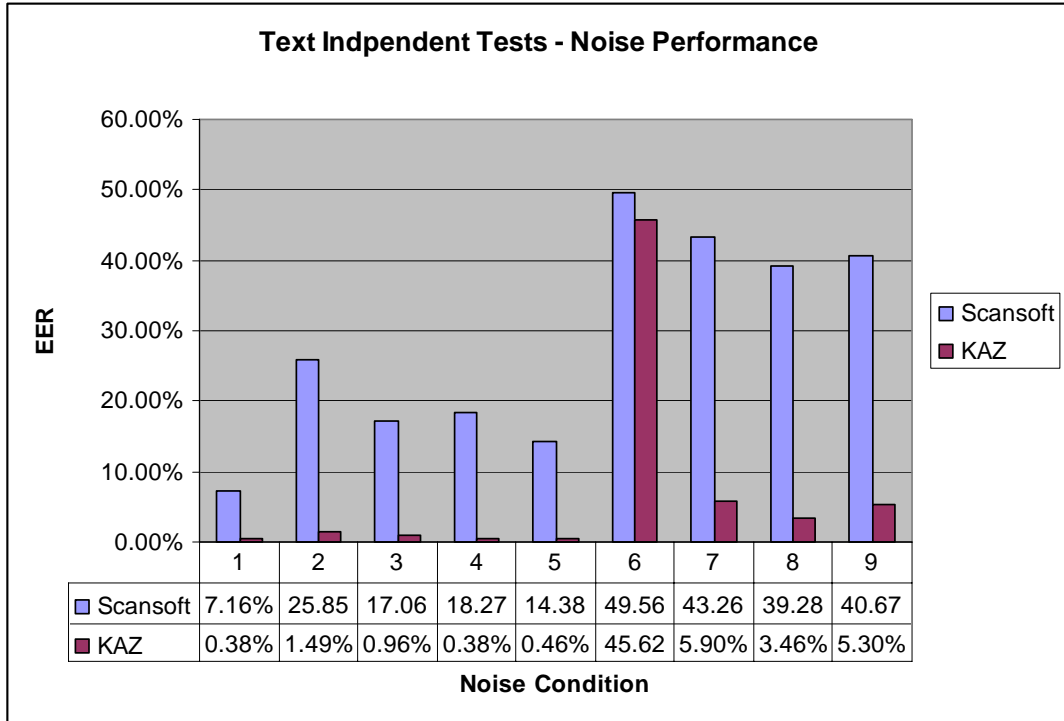
Figure 33. Increase in EER performance for mobile telephone speech data relative to the baseline performance.

5.3.2.1 Observations

The KAZ engine is almost completely insensitive to mobile telephone encoding rates, with an increase in EER performance of only 0.13% at the lowest data rate compared to 5.6% increase in EER for the Scansoft engine over and above the baseline performance of the engine.

5.3.3 Noise Performance

Figure 34 shows the performance of the text-independent engines for different noise levels and scenarios for Scansoft and KAZ engines.



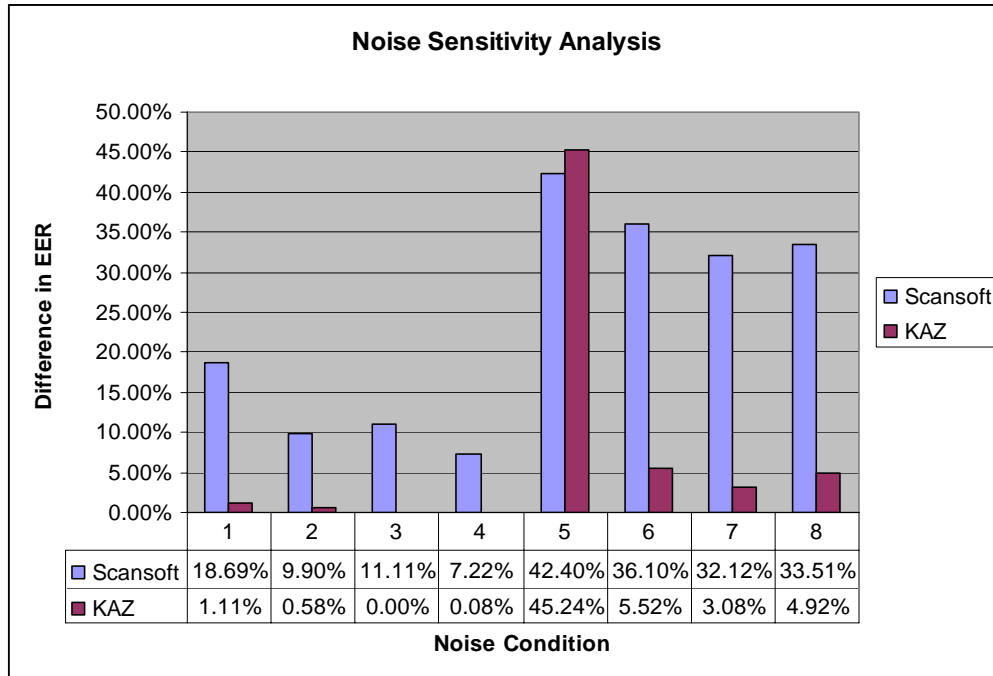
1 = Baseline Performance

2 = 0dB White Noise 3 = 0dB Office Noise 4 = 0dB City Noise 5 = 0dB Shop noise

6 = -20dB White Noise 7 = -20dB Office Noise 8 = -20dB City Noise 9 = -20dB Shop noise

Figure 34. Noise Performance of Text-independent engines

Figure 35 provides an analysis of the sensitivity of the engines to different noise levels and scenarios.



1 = 0dB White Noise 2 = 0dB Office Noise 3 = 0dB City Noise 4 = 0dB Shop noise
 5 = -20dB White Noise 6 = -20dB Office Noise 7 = -20dB City Noise 8 = -20dB Shop noise

Figure 35. Text Independent Noise Sensitivity Analysis.

5.3.3.1 Observations

These tests demonstrate that the KAZ engine was considerably more robust to noise conditions than the Scansoft engine, showing almost no sensitivity to noise scenarios at 0 dB and, apart from white noise condition, exhibiting a low level of sensitivity in high noise conditions (-20 dB signal to noise ratio).

Both engines exhibit very high EER performance at -20 dB white noise suggesting that both engines fail in a vulnerable mode (as opposed to fail safe mode).

5.4 Volumetric Stress Tests

Table 8 shows the average timing per enrolment and verification for Scansoft, Nuance and Persay for the Counting 1to9 evaluation. In the case of Scansoft and Nuance enrolment and verification timing remained relative static for all 1-1 test cases. In the case of Persay, the timings varied depending on the test condition.

Engine	Enrolment (Seconds)	Verification (Seconds)	CPU Speed/Memory
Scansoft	1.9	0.15	3.2GHz / 0.99GBytes
Nuance	0.45	0.14	3.2 GHz / 3.25 G Bytes
Persay	3.7 to 5.7	0.25 to 0.5	2.4 GHz / 480 M Bytes

Table 8. Volumetric Stress testing for Counting 1to9 Evaluation

Similarly, Table 9 shows the average timing for enrolment and verification for Scansoft, Nuance and Persay for the Name test, test case 4-1.

Engine	Enrolment (Seconds)	Verification (Seconds)	CPU Speed/Memory
Scansoft	0.2	0.01 to 0.06	3.2GHz / 0.99GBytes
Nuance	0.1	0.04	3.2 GHz / 3.25 G Bytes
Persay	3.0 to 4.5	0.15	2.4 GHz / 480 M Bytes

Table 9. Volumetric Stress testing for Name Evaluation

Table 10 shows the average timing for enrolment and verification for Scansoft and KAZ engines for the text-independent evaluation.

Engine	Enrolment (Seconds)	Verification (Seconds)	CPU Speed/Memory
Scansoft	4.2 to 8.0	0.12 to 0.38	3.2GHz / 0.99GBytes
KAZ	65.0 to 120.0	2.0 to 5.0	2.13 GHz / 3.53 Gbytes

Table 10. Volumetric Stress testing for Text Independent Engine

5.4.1 Observations

Volumetric tests showed little difference in the speed performance of Nuance and Scansoft engines. In the evaluation set-up Persay ran slightly slower than both Nuance and Scansoft. However, the processor for Persay was around 30% slower and memory was considerably smaller. On balance Persay had similar volumetric performance to Nuance and Scansoft given the hardware specifications.

However, KAZ was considerably slower than all other engines tested, being between 13 and 15 times slower than other engines. Whilst this engine operated on a slower processor, even taking this into account the KAZ technology was an order of magnitude slower than all other verification engines tested.

6 FINDINGS AND CONCLUSIONS

6.1 Text Dependent Evaluations

6.1.1 Finding from Test Case 1-1

Test case 1-1 (Counting 1to9) clearly showed differentiated performance between the vendors' engines, both in terms of baseline performance and robustness to mobile communications channels and noise scenarios.

Finding 1.1

Text dependent evaluation involving Counting 1to9 (Test Case 1-1): Nuance's and Persay's engines equivalent performance in terms of baseline performance as well as showing equivalent levels of robustness in mobile communications and noise conditions compared with Scansoft's engine.

In very high noise conditions some engines simply processed the speech data and returned a value, whilst others returned a high number of FTA's (Failure to Acquires). In these conditions, the engines showed almost no ability to discriminate between clients and impostors. Engines that fail to return a FTA under high noise conditions are potentially vulnerable to noise attacks.

Finding 1.2

Text dependent evaluation involving Counting 1to9 (Test Case 1-1) in high noise condition: Nuance and Persay engines showed a "fail-safe" characteristic whilst the Scansoft engine showed a "fail-vulnerable" characteristic.

Whilst some engines showed higher levels of noise robustness than others, all engines showed better performance in scenario noise compared to performance in white noise. This implies that the performance against white noise represents a worst case condition for all speaker verification engines in the evaluation.

Finding 1.3

All engines perform poorest in white noise conditions compared to scenario noise demonstrating that white noise is a worst case condition for all engines involved in this evaluation.

Comparison between baseline performance of the engines returned for the 1-1 300 by 300 and the 1-1 300 by 11 test cases shows that EER performance of all engines remained relatively stable.

Finding 1.4

All engines showed stable performance between the 1-1 300 by 300 and the 1-1 300 by 11 test cases.

6.1.2 Finding from Test Case 4-1, 4-2 and 4-2A

Test Cases 4-1, 4-2 and 4-2A (Name verification) showed much closer performance between the vendors' engines. However, Nuance and Persay did show stronger robustness to mobile communications channels. Nuance and Persay did show equivalent performance in noise scenarios for Name verification in all name tests performed.

Finding 2.1

Text dependent evaluation involving Name verification (Test Case 4-1, 2, 2A): all engines performed to a similar level of performance. Nuance and Persay showed higher levels of robustness in mobile communications and noise conditions for Name verification.

All text dependent engines tested showed an ability to discriminate in the situation when an impostor says the right answer to a "shared secret" question, based purely on the difference in the voice quality of the speaker.

Finding 2.2

Text dependent evaluation involving Name verification (Test Case 4-2, 2A): all engines would have an ability to discriminate between a client and an impostor saying the right answer to a "shared secret" question – indicating that all engines are suitable for implementing this function.

Finding 2.3

All engines showed increased performance on Test Case 4-2 (different speaker, different name) compared to Test Case 4-1 (different speaker; same name) indicating that all engines have an increased ability to discriminate in the situation when an impostor says the wrong information to a "shared secret" question.

Test Case 4-2A (Name verification: same speaker, different name) showed that all text-dependent engines have an ability to discriminate between the client speaking information that is different from the enrolled information. This result indicated that these engines would be suitable to implement a "shared secret" solution where the engines could be used to discriminate where an enrolled speaker provides the wrong answer to a shared secret question.

Finding 2.4

The results of the Test Case 4-2A, (same speaker, different name) indicated that all text-dependent engines tested are able to discriminate when an enrolled speaker says a different name to the one originally enrolled. This indicates that all engines could be suitable for implementation of a "shared secret" solution when the correctly enrolled speaker provides the wrong answer to a "shared secret" question.

6.1.3 Finding from Test Case 9-1 and 9-2 - Reduced Enrolment

The results of reduced enrolment shows mixed results. Nuance and Scansoft produced reduced performance as the number of enrolment samples were reduced from 3 to 2 and to a single sample, whilst Persay's engine showed increased performance with a smaller number of enrolments. This result indicates that the enrolment process used in the Persay engine may potentially be unstable in the evaluation environment.

Taking the Nuance and Scansoft results, both engines showed a small reduction in performance when reducing the enrolment from three to two repetitions, with a much higher reduction in performance when the enrolment was reduced to a single utterance. Nuance demonstrated higher levels of robustness (lower level of reduction in performance) compared to Scansoft as enrolment was reduced.

Finding 3.1

Nuance and Scansoft both showed decreases in performance as the enrolment was reduced. The decrease was lower when reducing enrolment from three repetitions to two compared to a reduction from two repetitions to a single utterance. This test indicates that two repetitions would be the minimum number of repetitions required for adequate enrolment. The test indicates that a single utterance is inadequate for robust verification.

Finding 3.2

Nuance appeared to show more robust performance than Scansoft for reduced enrolment.

Finding 3.3

The Persay engine appeared to return a result that provides evidence that the enrolment process in the evaluation environment is unstable and not optimised in the evaluation environment.

6.1.4 Finding from Test Case 12-1 and 12-2 – Mobile Telephone Enrolment

Results from enrolment using processed mobile telephone samples show that the performance of all engines decreased when enrolled in this environment, although the engines evaluated showed considerable variation in robustness in this condition.

Finding 4.1

Nuance and Persay demonstrated similar levels of robustness for mobile telephone enrolment, followed by Scansoft. This result is consistent in both the landlines and mobile verification situations.

6.1.5 Finding from Text Independent Evaluations

Two engines were tested in text-independent mode, Scansoft and KAZ.

Finding 5.1

Baseline tests showed that KAZ performed significantly better than Scansoft, with KAZ returning an EER performance of between 0.38% and 0.47% compared to Scansoft's EER performance of between 7.16% and 5.69% for the same speech data, greater than an order of magnitude difference in performance.

Finding 5.2

Tests with noisy speech data and mobile telephone speech data confirmed that KAZ maintained this performance over the noise range and mobile telephone conditions tested.

Finding 5.3

Tests with mobile telephone speech data showed that the KAZ engine EER performance increased only 0.13% between landline and lowest data rate mobile telephone speech compared to Scansoft where the EER performance increased 5.62% across the same data rate.

7 Vendor Feedback

7.1 Introduction

As part of the evaluation process, all vendors were given an opportunity to provide written comment and feedback on the evaluation results. Written feedback was obtained from Scansoft, Persay and KAZ. Nuance provided only verbal feedback relating to the applicability of their technology for text-independent applications.

Vendors' written feedback is reproduced here verbatim without comment:

7.2 Feedback from Scansoft

Telstra Speaker Verification Response to Evaluation Results

SpeechWorks[®] solutions
from **ScanSoft[®]**

Summary of Response

The recently conducted Telstra Speaker Verification Evaluation shows that the ScanSoft "SpeechSecure" product performs in a reliable and robust fashion in a multitude of conditions. The results of the evaluation shows that the ScanSoft verification engine is a secure and deployable product capable of fulfilling the requirements of network based speaker verification when used in an Australian environment.

The SpeechSecure engine also caters for both text dependent and independent applications and provides a high level of performance and accuracy when used in either application. This good result was achieved despite the verification engine not being accompanied by the optional Automatic Speech Recogniser which may be used to aid overall verification performance, especially involving numbers.

KEY POINTS

Below are some salient features of ScanSoft's SpeechSecure product that contribute to the product's performance in the trial.

ScanSoft SpeechSecure Text Dependent Analysis

- ◇ **ASR Engine Not Required** – The SpeechSecure engine as tested does not need to be coupled with the ASR engine reducing costs and providing non language specific operation. It may be noted that coupling with the optional ASR engine may help provide better performance of the system especially when verifying with numbers.
- ◇ **Standards Architecture Compliance**. - VoiceXML compliant engine may be interfaced using the industry standard SOAP protocol.
- ◇ **Flexible Architecture Options** - Flexible architecture plus non-reliance on an ASR engine allows for a geographical and language independent distribution of the technology. Pass-phrase applications do not have to be language dependant.
- ◇ **New version of Speech Secure available**. - Version 3.1 of the SpeechSecure engine just released has improved performance by 15% for text dependant applications.
- ◇ **Tuning of Anti Speaker Database**. - Data from the US used to model anti-speak database – performance improvements to be gained by using local environment specific data gained over time.

Text Independent Analysis

- ◇ **Commercially Deployable**. SpeechSecure may be used for either text dependent or independent applications sharing a common database. Text independent systems have been deployed and are in successful operation.
- ◇ **New version of Speech Secure available**. Version 3.1 of SpeechSecure shows 25% relative improvement in the error rate on text independent applications.

Detailed Response

The Telstra Speaker Verification trial was designed to evaluate the performance of various Speaker Verification systems that are available on the Australian market. ScanSoft's SpeechSecure speaker verification system has many successful deployments globally and was one of the products nominated for the evaluation.

The results show the ScanSoft technology as performing robustly and consistently in both text dependent and independent operation, however we believe there are a few factors that prevented our technology from achieving the highest possible score.

It should be noted is that the evaluation was conducted on an older version of the Speech Secure engine (Version 3.0). Subsequent to the completion of the evaluation, a new version of the engine (Version 3.1) has been released resulting in a 15% relative improvement in Equal Error Rates for text dependent applications and 25% improvement in text independent applications coupled with superior performance in noisy conditions. ScanSoft is committed to the constant enhancement of its products through improvement of the core technology based on feedback gathered from performance in the field.

Further to this, an anti-speaker database is utilised by the engine's Neural Tree Network to help discern one voice from another. Although ScanSoft believes this mechanism is 'best of breed' for this component of the verification engine – it does require a certain amount of 'local data' to establish the highest performance and reliability. The default anti-speaker database has data that was collected solely in the U.S. whereas an anti-speaker database collected in Australia would contain influence from the telephone networks in addition to subtle differences in the phonetic content of people's voices. This local data would improve the performance of the system under local conditions helping to improve the EER figure.

SpeechSecure has been designed to provide a speech based biometric in a multitude of applications. Its open architecture and standards based design allow speech secure to be integrated into a range of speech network platforms. Integration may be performed standalone, client – server or via SOAP based protocol. There is also an Open Speech Dialogue Module for VoiceXML Gateways. The decoupling from the recogniser provides versatility in the form of multi-lingual and multi-regional applications using the same engine and database. Further, the web based architecture and independence from any ASR engine allows SpeechSecure to be installed in a distributed fashion accessing a single secure database.

In relation to the Text Independent results, the ScanSoft engine is a practical, market ready text independent solution. The other text independent engine tested is presently only a prototype system running at an order of magnitude slower than the ScanSoft engine and is not yet market ready.

Finally, ScanSoft is one of the two vendors that support text independent verification out of the four suppliers tested. ScanSoft's layered technology approach which incorporates Dynamic Time Warping, Hidden Markov Modelling and Neural Tree Networking, composing a core technology which is suitable across a broad range of applications. ScanSoft's SpeechSecure is a technology that stands alone in being able to provide a robust and secure technology that is ASR independent and provides both text dependent and independent operation.

7.3 Feedback from Persay

October 11, 2005

Persay Comments to the final report (version 2.0)

The evaluation described in this report focused on the raw verification performance over a pre-prepared database. Real-time verification in an operational environment requires some additional measures that were not evaluated here. Following is a brief description of some of Persay's VocalPassword™ 5.0 features, which are crucial for a successful implementation of a speaker verification application in an operational environment:

- Enrollment consistency – testing that all enrollment sentences are valid and consistent allows the system to prompt the speaker for additional enrollment repetitions, if necessary, while he or she are still online, thus minimizing the failure-to-enroll rate.
- Multiple background models with optimal selection – allowing the system to work in multi-language, multi-accent environments, by using multiple background models simultaneously and optimally selecting the best matching background model for each speaker.
- Advanced scoring technique – allowing the calling application to distinguish between valid speakers that said the wrong pass-phrase, and impostors that said the correct pass-phrase, without using any language-dependent speech recognition engine.
- Multiple verification repetitions – allowing the system to prompt the speaker for additional repetitions of the same test phrase or for a new repetition of a fall-back pass-phrase, based on the results of the first verification attempt. This can significantly improve the verification performance, as well as the overall security level of the system.
- Playback detection – allowing detection of playbacks of previous interactions with the system.

Additionally, Persay's research team has made significant progress since the evaluation system was installed at the University of Canberra. Specifically, verification performance was improved since then following the introduction of extended feature vectors and advanced classification techniques. Currently, the research team focuses on improving the verification performance under multiple-channel conditions. The improved-performance verification engine is expected to be integrated in Persay's next release of VocalPassword™.

Persay VocalPassword™ 5.0 utilizes a comprehensive text-dependent biometric speaker verification platform that enables verification and identification of individuals using a simple spoken pass phrase. Totally language and accent independent, VocalPassword™ uses voice biometrics to ensure that speakers are who they claim to be, securing access to remote services, telephony and web applications, eliminating identity fraud and enhancing customer experience.

VocalPassword 5.0 offers unsurpassed accuracy and performance, ease of integration, security, and smooth deployment through its new algorithms, unique design, features and tools. Based on .Net technology, this innovative, scalable, and robust platform exposes Web Service (SOAP) based APIs delivering speaker verification, administration, and management services.

VocalPassword™ was selected by leading financial services, telecom operators, and security organizations, as well as IVR/voice platform vendors and system integrators worldwide, as the speaker verification platform of their choice.

7.4 Feedback from KAZ



Addendum Comments to University of Canberra Centrelink Report

Executive Summary

The KAZ Speaker Verification Engine (KAZ SVE) exhibited the best EER performance of all products tested in this trial. The initial trial also produced concerns about the amount of time and data required to process an authentication instance.

KAZ has completed a re-engineering of the KAZ Speaker Verification Engine since the assessment presented in this report was completed.

Motivated by requests from Centrelink and Telstra to understand how this new version of the KAZ SVE performed under the same conditions as the original trial, the tests were re-run within the same environment at the University of Canberra using the same test methodology and the same data.

This addendum reports on the results from the second iteration of tests conducted in September 2005.

Initial Trial Results

Summary of results contained in this report from University of Canberra.

Positive performance results

- Best EER performance across tested noise conditions for both text dependent and text independent products (see Figures 16, 21, and 34 of the Speaker Verification Evaluation Report).
- Best EER performance across all measured mobile telephone encoding rates for both text dependent and text independent products (see Figures 14, 19, and 31 of the Speaker Verification Evaluation Report).
- Best EER baseline performance for both text dependent and text independent products (see Figures 12, 13, and 30 of the Speaker Verification Evaluation Report).

Negative performance results

- The amount of processing time required to authenticate speech files, and
- the amount of data used in authentication to achieve the levels of accuracy previously mentioned. Concerns were expressed that, while the KAZ technology had the best accuracy outcomes, the amount of speech used to achieve this would make deployment of the technology impractical from a human factors perspective.

Initial Trial Analysis and Resulting Actions

Upon analysis of the test results, two factors were identified as contributing to the observed slow processing speed and the large amount of data employed.

- Contributing fact one – the KAZ SVE's executable performance was too slow and inefficient, and

- contributing fact two - the use of large amounts of speech data utilised during the authentication runs were a result of test design decisions and not because of any explicit processing requirements from the KAZ verification technology.

Having said this, the question still remained about speed and accuracy performance with smaller and more commercially acceptable amounts of speech data. Centrelink was specifically interested in performance data in the range of five to seven seconds of speech for authentication purposes.

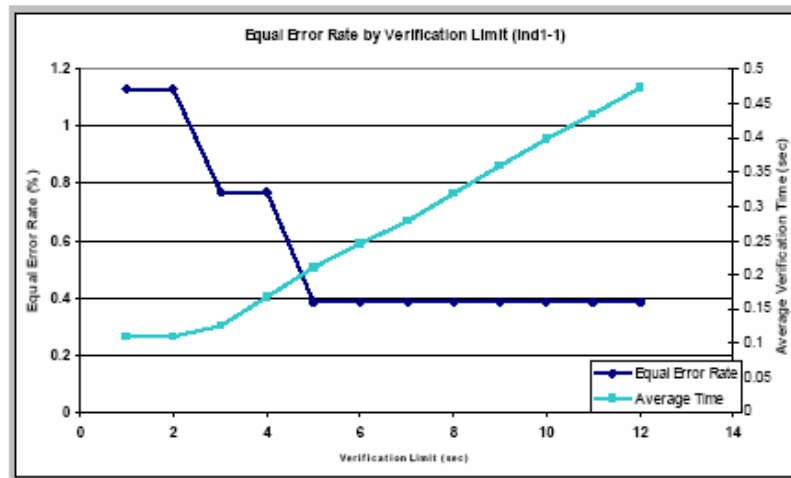
A second run of the verification tests were performed. Analysis of the test runs were produced for speech data amounts ranging from one to twelve seconds in one second increments.

The remainder of this addendum will report on the results of these tests.

Results from Rerun of Trial

All tests were run with 30 seconds of enrolment data.

Authentication Trials: KAZ SVE Baseline EER and Timing information for 1 second to 12 seconds of speech data



Summary of Results: KAZ SVE V2.0 retained most accurate EER authentication performance in comparison to the other tested products. This is true for the 3 seconds to 12 seconds of speech data range.

KAZ SVE V2.0 reduces processing time from a range of 2.0 - 5.0 seconds per authentication (See Table 10) to within the range of 0.12 seconds to 0.47 seconds. The more data used in the authentication run, the more time is required for processing.

Conclusion

As a result of the re-engineering of the KAZ SVE, both performance limitations shown in KAZ SVE version 1.1 were addressed successfully. The KAZ SVE Version 2.0 performs at comparable speeds to the other trialled verification engines while retaining its best performer for accuracy ranking.

This was true for baseline data trials, the data bandwidth tests as well as for tested noise conditions.

8 Glossary

A8 means an 8-bit, 8-kHz digital encoding of the speech signal using the A-law companding standard used in the Australian telephone network.

CCIR means the Centre for Communications Interface Research, Edinburgh University

Client means a legitimate user who has enrolled in the system; the client's voice has been used to create that client's voice model or template

CSO means a Customer Service Officer

Equal-Error Rate (EER) means the results from a threshold setting of the system where the false-accept rate equals the false-reject rate.

False-Accept Rate (FAR) means the error rate at which impostors are incorrectly accepted as a claimed speaker by the system. The FAR is defined as: the number of impostors accepted divided by the total number of speakers tested.

False-Reject Rate (FRR) means the error rate at which legitimate speakers (clients) are incorrectly rejected by the system. The FRR is defined as: the number of clients rejected divided by the total number of speakers tested.

Failure To Acquire (FTA) means that a particular speech file cannot be used to obtain a verification result against a given client model, e.g. the noise level is too high.

Failure To Enrol (FTE) means that a particular set of speech files cannot be used to compute a client model, e.g. the amount of speech data is insufficient for enrolment.

IBG means the International Biometric Group, New York, USA

IVR means Interactive Voice Response (system) using touch-tone input to enable callers to select spoken menu items.

Likelihood Score means the probability score produced at the output of a verification engine which is the engine's means that the speech sample is from the same speaker as a given speaker template.

m-script (short for model script) means a computer script, which specifies all the voice files that are used to create the voice model of a given client

NCBS means the National Centre for Biometric Studies of the University of Canberra

NLSR means natural language speech recognition– that is a system that recognises and processes naturally spoken information.

Open dialogue means an NLSR module that accepts an unstructured spoken input that it processes to obtain information about caller requests or requirements. For example:

NLSR: “Welcome to Service – How can we help you today?”

Caller: “I've recently moved and I'm calling about my “*change of address*””

NLSR: “Please confirm that you want to notify us of your “*change of address*?””

Caller: “Yes, that's right”

Call is directed to the “*Change of address*” application

POI means Proof of Identity

ROI means Return on Investment – the period of time required for cost benefits from the deployment of the system to exceed the price paid.

Speaker Template means a statistical template of information representing a person's voice characteristics that is used in a Voice Authentication (VA) system to confirm a caller's claimed identity.

SV means Speaker Verification (cf VA)

t-script (short for test script) means a computer script, which specifies one client model and one voice file; the voice file is to be tested against the client model and the numerical result of that test is then stored in the t-script.

UC means the University of Canberra

VA means Voice Authentication – a engine that processes callers' voice characteristics and compares them against a speaker template stored in a database to confirm a caller's claimed identity (often also referred to as Speaker Verification engine).

9 **References**

- a. Project Contracts
- b. Meeting of the 7th January 2005.
- c. Telstra Project plan of the 13th January 2005
- d. Emu User Manual
- e. Performix User Manual

10 Acknowledgements

The author wished to express his sincere thanks to the following for their invaluable contributions to this evaluation study and preparation of this report.

Professor Michael Wagner, University of Canberra School of Information Sciences and Engineering

Dr. Ted Dunstone, CEO of Biometix Pty Ltd and author of Performix

Mr. Ross Summerfield, of the Australian Government

Mr. Fergal Murphy, Project Manager Telstra Corporation

Mr. Joel Moss, Research Assistant and the person responsible for implementing and operating the evaluation process.

My thanks also to the very kind contributions from Nuance, Scansoft, Persay and KAZ, the technology participants, and to Unisys Australia Pty Ltd for their very kind donation of the server systems.

Finally, my thanks to all those within the Australian Government and Telstra Corporation who made financial support for this study possible.

Dr Clive Summerfield

Director

National Centre for Biometric Studies

University of Canberra

2005

Appendix A - Example Program for m-Script Production

Example program to extract speech data from the speech database, generate m-scripts and submit scripts to verification engine for evaluation.

```

package require mysqlc1
package require snack

# generate scripts and send to vendors, copy result to performix

proc checkgoodid { testid } {

    # check if all samples for passed speaker have been evaluated

    global sql
    global db

    set sql "SELECT COUNT(QUALITY) FROM recordings WHERE QUALITY LIKE \"%good%\" AND QUALITY NOT LIKE \"%quiet%\" AND
    UNIQUESCRIPTIDENTIFIER = $testid AND ITEM = 2 AND CALLNUMBER < 3"

    set n [mysql::sel $db $sql]

    if {$n > 0} {

        set line [mysql::fetch $db]
        set temp [lindex $line 0]

        if {$temp == 4} {
            return "all"
        } else {
            return "$temp"
        }
    } else {
        return "error"
    }
}

proc checklandline { testid } {

    # check if all samples for passed speaker have been evaluated

    global sql
    global db

    set sql "SELECT i.UNIQUESCRIPTIDENTIFIER FROM recinformation i, recordings r WHERE i.UNIQUESCRIPTIDENTIFIER = r.UNIQUESCRIPTIDENTIFIER AND
    i.CALLNUMBER = r.CALLNUMBER AND PHONETYPE = 1 AND r.UNIQUESCRIPTIDENTIFIER = $testid AND r.CALLNUMBER < 3"

    set n [mysql::sel $db $sql]

    if {$n > 0} {

        if {$n == 32} {
            return "landline"
        } else {
            return "mixed"
        }
    } else {
        return "none"
    }
}

proc getgender { id } {

    # get gender for id

    global db
    global sql

    set sql "SELECT GENDER FROM customer WHERE UNIQUESCRIPTIDENTIFIER = $id"

```

```

set n [mysql::sel $db $sql]
if {$n > 0} {
    set line [mysql::fetch $db]
    set gender [lindex $line 0]
    if {$gender != ""} {
        return $gender
    } else {
        set sql "SELECT QUALITY FROM recordings WHERE UNIQUESCRIPTIDENTIFIER = Sid AND CALLNUMBER = 1 AND ITEM = 1 AND SUBITEM = 1 AND
QUALITY LIKE \"%female%\""
        set n [mysql::sel $db $sql]
        if {$n > 0} {
            set line [mysql::fetch $db]
            return "F"
        } else {
            return "M"
        }
    }
} else {
    return "unknown"
}
}

proc getscrip { id } {
    # get the scrip number for the given id
    global sql
    global db
    set sql " SELECT SCRIPTNOISSUED FROM customer WHERE UNIQUESCRIPTIDENTIFIER = $id "
    set n [mysql::sel $db $sql]

    if {$n > 0} {
        set line [mysql::fetch $db]
        set scrip [lindex $line 0]
        if {$scrip != 12} {
            return $scrip
        } else {
            return 3
        }
    } else {
        set sql " SELECT QUALITY FROM recordings WHERE UNIQUESCRIPTIDENTIFIER = Sid AND QUALITY LIKE \"%scrip%\""
        set n [mysql::sel $db $sql]
        if {$n > 0} {
            set status [mysql::fetch $db]
            set first [string first "scrip" $status]
            if {$first > -1} {
                incr first 7
                set scripnum [string range $status $first [expr $first + 1]]
                set lastchar [string range $scripnum end end]

                if {![regexp {[0-9]+} $lastchar]} {
                    set mylen [string length $scripnum]
                    incr mylen -2
                    set scripnum [string range $scripnum 0 $mylen]
                }
            }
            return $scripnum
        }
    }
}

```

```

    } else {
        return "unknown"
    }
} else {
    return "unknown"
}
}
}

proc getmodelid { id } {
    # get the model id associated with this unique identifier

    global db
    global sql

    set sql "SELECT MODELID FROM customer WHERE UNIQUESCRIPTIDENTIFIER = $id"
    set n [mysql::sel $db $sql]
    if {$n > 0} {
        set line [mysql::fetch $db]
        set model [lindex $line 0]
        if {$model != ""} {
            return $model
        }
    }
    return "unknown"
}

#####
# Create test 4.1 - landline

set testname "4-1"
set testtext "landline: CustomerName"

puts "-----"
puts "running: test $testname - $testtext"
puts "The time is: [clock format [clock seconds] -format %H:%M:%S]"
puts "-----"

set performixdir "c:/program files/performix/data/"
set performixbatchdir "c:/program files/performix/"
set scansoftdir "c:/sve/speakerdb/scansoft/"
set nuancedir "c:/sve/speakerdb/nuance/"
set kazdir "c:/sve/speakerdb/kaz/"
set persaydir "c:/sve/speakerdb/persay/"

set scanfilename "$testname-enrolscan.txt"
set nuancefilename "$testname-enrolnuance.txt"
set persayfilename "$testname-enrolpersay.txt"
set kazfilename "$testname-enrolkaz.txt"
set mapfilename "$testname-mapping.txt"
set genfilename "$testname-gender.txt"
set metafilename "$testname-meta.txt"

set scanverif "$testname-verifscan.txt"
set nuanceverif "$testname-verifnuance.txt"
set persayverif "$testname-verifpersay.txt"

```

```

set kazverif "Stestname-verifikaz.txt"

#####
# Generate script

expr srand(12345)

puts "creating enrolment file"

# get list of ids

set db [mysql::connect -db spkver3 -user root -password password]

set sql "SELECT DISTINCT UNIQUESCRIPTIDENTIFIER FROM recordings WHERE UNIQUESCRIPTIDENTIFIER < 600000000 AND UNIQUESCRIPTIDENTIFIER
NOT IN (SELECT DISTINCT UNIQUESCRIPTIDENTIFIER FROM recordings WHERE QUALITY LIKE \"%different speaker%\")"

set n [mysql::sel $db $sql]

for {set i 0} {$i < $n} {incr i} {

    lappend ids [mysql::fetch $db]

}

# get list of good ids

set length2 0

for {set i 0} {$i < $n} {incr i} {

    set allres [checkgoodid [lindex $ids $i]]

    if {$allres == "all"} {

        lappend ids2 [lindex $ids $i]

        incr length2

    }

}

# get list of good ids on landline

set numgood 0

for {set i 0} {$i < $length2} {incr i} {

    set linerres [checklandline [lindex $ids2 $i]]

    if {$linerres == "landline"} {

        lappend goodids [lindex $ids2 $i]

        incr numgood

    }

}

# create 10 lists of callers, one per script

set scriptlist(M) [list 0 0 0 0 0 0 0 0 0 0]
set scriptlist(F) [list 0 0 0 0 0 0 0 0 0 0]

for {set i 0} {$i < $numgood} {incr i} {

    set curid [lindex $goodids $i]

    set script [getscript $curid]

    set gender [getgender $curid]

    set curlist [lindex $scriptlist($gender) $script]

    if {$curlist == 0} {

        set curlist $curid

    } else {

        lappend curlist $curid

    }

    lset scriptlist($gender) $script $curlist

}

```

```

puts "made list of callers (total of $numgood)"
puts "creating script files"

# create script files

set scanfile [open $scanfilename "w"]
set nuancefile [open $nuancefilename "w"]
set persayfile [open $persayfilename "w"]
set kazfile [open $kazfilename "w"]
set mapfile [open $mapfilename "w"]
set genfile [open $genfilename "w"]
set metafile [open $metafilename "w"]
puts $metafile "MODEL UNIQUEID GENDER AUSBORN AGE"

set path ""
set pathn ""
#set pathn "y:\speakerdb\wavs4\"

for {set i 0} {$i < $numgood} {incr i} {
    set curid [lindex $goodids $i]

    set filename31 "$path$curid\$curid-1-2-1.wav"
    set filename32 "$path$curid\$curid-1-2-2.wav"
    set filename33 "$path$curid\$curid-1-2-3.wav"

    # output to enrol files

    # model number - all
    set modelid [getmodelid $curid]
    puts $scanfile $modelid
    puts $nuancefile $modelid
    puts $persayfile $modelid
    puts $kazfile $modelid

    # scansoft
    set tag ""

    puts $scanfile "$filename31 $tag"
    puts $scanfile "$filename32 $tag"
    puts $scanfile "$filename33 $tag"

    # kaz
    puts $kazfile "$filename31 $tag"
    puts $kazfile "$filename32 $tag"
    puts $kazfile "$filename33 $tag"

    # nuance
    set tag "CustomerName"

    puts $nuancefile "$pathn$filename31 $tag"
    puts $nuancefile "$pathn$filename32 $tag"
    puts $nuancefile "$pathn$filename33 $tag"

```

```

# persay
puts $persayfile "$filename31 Stag"
puts $persayfile "$filename32 Stag"
puts $persayfile "$filename33 Stag"

# output to gender file
set gender [getgender $curid]
puts $genfile "$modelid $gender"

# output meta data
set sql "SELECT AUSTRALIANBORN, DESCRIPTION FROM customer c, agerange a WHERE c.AGERANGE = a.item AND UNIQUESCRIPTIDENTIFIER = $curid"
set n [mysql::sel $db $sql]
if {$n > 0} {
  set line [mysql::fetch $db]
  set ausborn [lindex $line 0]
  set age [lindex $line 1]
} else {
  set ausborn " "
  set age " "
}
puts $metafile "$modelid $curid $gender $ausborn $age"

# output to mapping file
set script [getscript $curid]
set callnum 2
puts $mapfile "$modelid $path$curid\\$curid-$callnum-2-1.wav Stag $script"
}

close $scanfile
close $nuancefile
close $kazfile
close $persayfile
close $genfile
close $metafile
close $mapfile

# copy meta and mapping file to performix dir
file copy -force $mapfilename "$performixdir"
puts "copied $mapfilename to $performixdir"
file copy -force $metafilename "$performixdir"
puts "copied $metafilename to $performixdir"

# copy enrol files to vendor dirs
file copy -force $scanfilename $scansofdir
puts "copied $scanfilename to $scansofdir"
file copy -force $nuancefilename $nuancedir
puts "copied $nuancefilename to $nuancedir"
file copy -force $persayfilename $persaydir
puts "copied $persayfilename to $persaydir"
#file copy -force $kazfilename $kazdir
#puts "copied $kazfilename to $kazdir"
#file copy -force $genfilename $kazdir
#puts "copied $genfilename to $kazdir"

```

```
puts "finished generating enrolment scripts"
puts "-----"
puts "The time is: [clock format [clock seconds] -format %H:%M:%S]"

#####
# send enrol request to vendors

set numexpectedresults 0

# scansoft
if {1 == 1} {
  puts ""
  incr numexpectedresults
  puts "connecting to scansoft"
  set server "137.92.104.18"
  set sockChan [socket $server 9000]
  puts "connected to scansoft"

  puts $sockChan "enrol"
  puts $sockChan "$scanfilename"
  puts "sent enrol request"

  close $sockChan

  puts "disconnected from scansoft"
}

# nuance
if {1 == 1} {
  puts ""
  incr numexpectedresults
  puts "connecting to nuance"
  set server "137.92.104.21"
  set sockChan [socket $server 9000]
  puts "connected to nuance"

  puts $sockChan "enrol"
  puts $sockChan "$nuancefilename"
  puts "sent enrol request"

  close $sockChan

  puts "disconnected from nuance"
}

# persay
if {1 == 1} {
  puts ""
  incr numexpectedresults
  puts "connecting to persay"
  set server "137.92.104.17"
  set sockChan [socket $server 9000]
  puts "connected to persay"
```

```

puts $sockChan "enrol"
puts $sockChan "$persayfilename"
puts "sent enrol request"

close $sockChan

puts "disconnected from persay"
}

# wait for results
set numresults 0

proc Server {channel clientaddr clientport} {
    global scansoftdir
    global scanfilename
    global scanverif
    global nuancedir
    global nuancefilename
    global nuanceverif
    global kazdir
    global kazfilename
    global kazverif
    global persaydir
    global persayfilename
    global persayverif
    global performixdir
    global numresults
    global state
    global numexpectedresults

    puts "Connection from $clientaddr registered"

    # get result
    gets $channel vendor
    gets $channel result
    puts "received connection from $vendor"
    puts "received: $result result"
    puts "The time is: [clock format [clock seconds] -format %H:%M:%S]"

    # disconnect
    close $channel
    puts "disconnected from $clientaddr"

    if {$vendor == "scansoft"} {
        if {$result == "completed"} {
            # copy file to performix dir
            file copy -force "$scansoftdir$scanfilename" "$performixdir"
            puts "copied $scanfilename to $performixdir"
        } else {
            puts "bad result received from scansoft"
        }
    }
    incr numresults
} elseif {$vendor == "nuance"} {

```

```
if {$Result == "completed"} {
    # copy file to performix dir
    file copy -force "$nuancedir$nuancefilename" "$performixdir"
    puts "copied $nuancefilename to $performixdir"
} else {
    puts "bad result received from nuance"
}
}
incr numresults
) elseif {$Vendor == "persay"} {
    if {$Result == "completed"} {
        # copy file to performix dir
        file copy -force "$persaydir$persayfilename" "$performixdir"
        puts "copied $persayfilename to $performixdir"
    } else {
        puts "bad result received from persay"
    }
}
incr numresults
}

if {$Numresults == $Numexpectedresults} { set state accepted }
# wait for next result
}

puts "waiting for vendors..."
puts ""
set mysock [socket -server Server 9001]
fconfigure $mysock
if {$Numexpectedresults > 0} {
    vwait state
}

puts "-----"
puts "received results from all vendors"
puts "-----"
puts "finished"
puts "The time is: [clock format [clock seconds] -format %H:%M:%S]"
puts "-----"
```