VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ MLUVČÍHO VE VOIP PROSTŘEDÍ
SPEAKER RECOGNITION IN THE VOIP ENVIRONMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN REMEŠ
AUTHOR

VEDOUCÍ PRÁCE — Ing. OLDŘICH PLCHOT
SUPERVISOR

BRNO 2014

Abstrakt

Tato práce popisuje použití systémů pro rozpoznávání mluvčího v prostředí VoIP, úspěšnost systému a přístupy k jejímu zlepšení. Popisuje architekturu těchto systémů, metriky pro vyhodnocení jejich úspěšnosti a klíčové komponenty VoIP z hlediska rozpoznávání mluvčího. Je zde popsáno vytvoření simulace VoIP prostředí, úspěšnost systému je vyhodnocena na datech pocházejících z různých druhů VoIP prostředí a výsledky jsou demostrovány. Adaptace a kalibrace systému je provedena a jejich přínosy zhodnoceny.

Abstract

This work describes using speaker recognition systems in the VoIP environment, system performance and approaches to improving it. System architecture, evaluation metrics and VoIP technology key components from the view of speaker recognition are described. VoIP environment simulation is described. Speaker recognition system's performance is evaluated on data sets from various kinds of VoIP environments and the results are demonstrated. System adaptation and calibration is performed and their benefits are discussed.

Klíčová slova

SRE, rozpoznávání mluvčího, VoIP, síť, chyby sítě, kodeky, přesnost SRE, EER, DCF, adaptace, kalibrace

Keywords

SRE, speaker recognition, VoIP, network, network errors, codecs, SRE performance, EER, DCF, adaptation, calibration

Citace

Jan Remeš: Speaker Recognition in the VoIP Environment, bakalářská práce, Brno, FIT VUT v Brně, 2014

Speaker Recognition in the VoIP Environment

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Oldřicha Plchota. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . .
Jan Remeš
May 21, 2014

Poděkování

I would like to thank my supervisor, Olda, for countless advices and guidance during my work on the thesis.

Contents

1 Introduction **2**

2 Speaker Recognition **3**
- 2.1 Score . . . 3
- 2.2 SRE system . . . 4
  - 2.2.1 UBM . . . 5
  - 2.2.2 i-vectors . . . 6
  - 2.2.3 PLDA . . . 6
- 2.3 Adaptation . . . 7
- 2.4 Calibration . . . 8
- 2.5 System accuracy evaluation . . . 8

3 Voice over IP (VoIP) **12**
- 3.1 VoIP call . . . 12
- 3.2 Encoding . . . 12
- 3.3 VOIP codecs . . . 14
  - 3.3.1 Silence transmission . . . 14
  - 3.3.2 ITU-T G.711 . . . 14
  - 3.3.3 ITU-T G.723.1 . . . 14
  - 3.3.4 ITU-T G.729 . . . 15
  - 3.3.5 GSM AMR . . . 15
- 3.4 Networking . . . 15

4 Evaluations & Experiments **18**
- 4.1 SRE system . . . 18
- 4.2 Audio data set . . . 19
- 4.3 VoIP and network simulation . . . 20
- 4.4 Performance Evaluation . . . 21
- 4.5 Adaptation Impact . . . 23
- 4.6 Calibration Impact . . . 29

5 Conclusion **30**

Chapter 1

Introduction

Speaker recognition (SRE) is a technology widely used in today's world. It finds its use in bank and financial sector (remote authentication in general), security applications, and audio data indexing. The quality of SRE technology, in terms of percentage of wrong trials (false-positives and false-negatives referred to as *misses*) is of great importance to everyone using this technology. There are many software products for running SRE, and each may provide different results. All of them, however, depend strictly on the quality of the data provided.

In many applications, the data of interest come from a phone call (be it lawful interception, remote authentication or call history search). As more and more organizations and people tend to use *VoIP technology* (so-called „Internet-telephony“) instead of the classical phones, there is a need for SRE technology to adapt to these conditions.

This work describes current state of SRE technology, VoIP technology and problems arising from it for SRE, and metrics for evaluating SRE systems' performance. SRE system performance is measured for both 'ordinary' data and their VoIP counterparts. Issues leading to their difference are described and possible solutions are suggested. Results of conducted experiments are provided.

In chapter 2, the SRE system and metrics designed to evaluate its performance are described. Chapter 3 deals with the VoIP technology, describes VoIP codecs and features with focus on properties, which may influence SRE performance. In chapter 4, software setup and data used to conduct experiments are described and the results of the experiments are shown. Chapter 5 brings the summary of the work and suggests possible ways of further extensions.

Chapter 2

Speaker Recognition

Speaker recognition (SRE) is a classification process, where automatic software systems extract information from audio recordings and identify their speakers. As described by [4], the process may be viewed either as an *identification* task, where the system's goal is to determine, to whom in a given set of speakers given recording belongs; or as a *verification* task, where two recordings are given and the system must decide, whether they have been uttered by the same speaker, or not. For the identification task, the set of potential speakers may either be closed (the recording must have been uttered by a speaker from the set; or open, where previous condition does not hold. The fact, that the speaker may not be present in the considered set (which is true for most of the applications), can make the identification task results difficult to interpret. The identification task, however, can be reinterpreted as a series of verification tasks for each speaker in the considered set. This fact allows SRE systems to provide the verification task capabilities only without removing their ability to perform the identification. The tasks are presented to the system in the form of *trials*. A trial is defined by pair of recordings. Each trial is either *target* (both recordings were uttered by the same speaker) or *nontarget* (each recording was uttered by different speaker).

2.1 Score

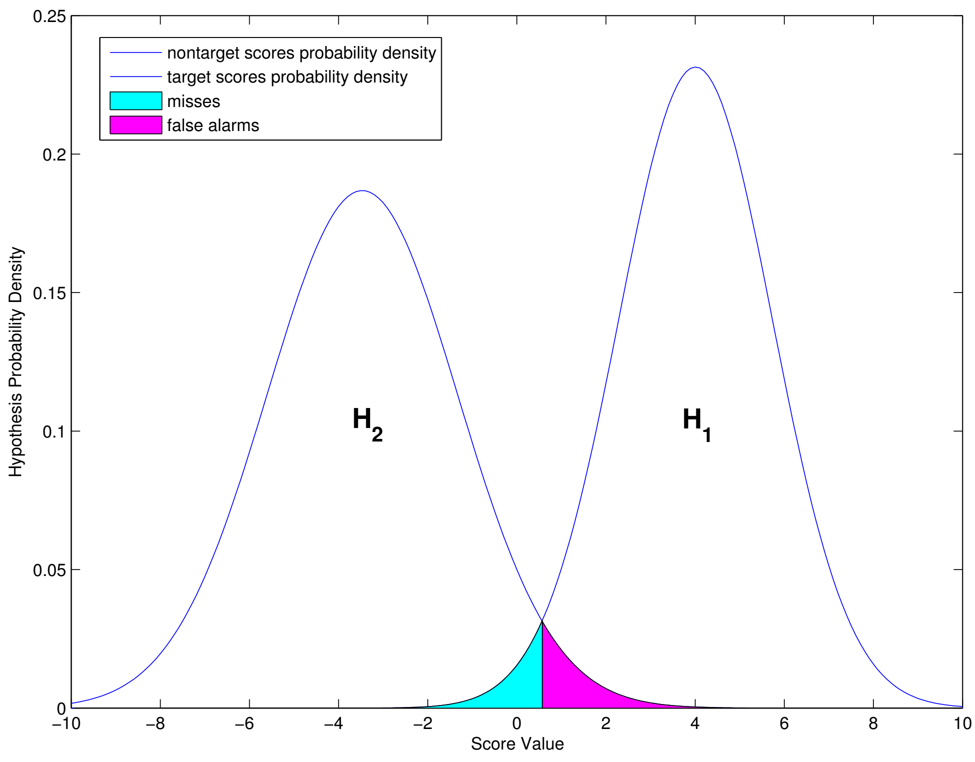
SRE systems return their decision in the form of numerical value called score. The higher the score, the more likely is the trial target. There is a threshold value (usually zero) defined for the system. Trials with score above the threshold are considered target, the ones with score below the threshold are considered nontarget. The system may even produce the threshold value when it cannot reach the decision (e. g. recordings contain too little speech to be processed). With threshold set to zero, system's decision may be simply reached by examining score sign; the absolute value of the score then indicates system's „certainty“ over the result.

Usually, the score is system's log-likelihood ratio (LLR) of two hypotheses:

- H_1....both recordings were uttered by the same speaker (target hypothesis)
- H_2....each recording was uttered by different speaker (nontarget hypothesis)

The score can be calculated by 2.1. Note, that when both hypotheses are equally likely, the fraction will have a value of 1, resulting for score $= log(1) = 0$.

Figure 2.1: Target and nontarget score distributions



$$H_1 = \text{target hypothesis} \tag{2.1a}$$

$$H_2 = \text{nontarget hypothesis} \tag{2.1b}$$

$$\text{score} = log(\frac{p(H_1)}{p(H_2)}) \tag{2.1c}$$

Following section describes an SRE system and its components.

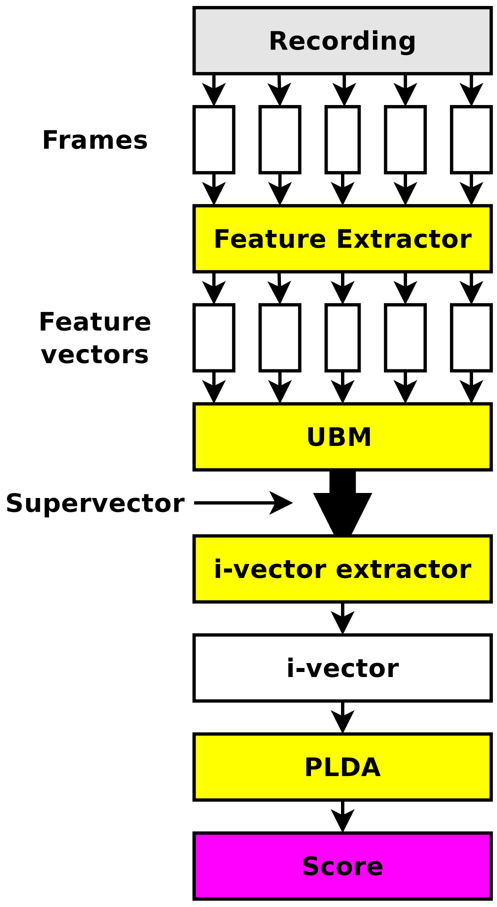
2.2 SRE system

An SRE system is understood as a software system, which can take two waveform files as input and is supposed to yield a decision about whether both provided recordings were uttered by the same speaker. In this section, the recognition process and used techniques will be described. SRE system scheme is depicted in 2.2.

First, each recording is split into frames of 20 ms length (frames are overlapping by 10 ms, see [2]). Following operations are run per-frame.

The system needs to decide, whether a frame contains actual speech. This technique is known as VAD (Voice Activity Detection). There are two typical approaches to VAD. First of them is energy-based detection, where the log-spectrum of the signal is thresholded or directly classified, second is use of higher level classifiers, such as phoneme recognizer

Figure 2.2: SRE system scheme



(as described in [2]), which transform the signal into sequence of speech and silence frames. Frames not containing speech are discarded, the rest of them are processed further.

For each frame, system computes so-called *features*. Todays systems use Mel-frequency cepstral coefficients (MFCCs) and their dynamic coefficients called deltas and double-deltas. Deltas are obtained by differentiating adjacent MFCC vectors (numerical approximation of their derivative). The formula is given at 2.2, equation taken from [7]. Double-deltas are computed using the same formula from deltas. This step results in a sequence of low-dimensional (60 dimensions) feature vectors.

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta * (c_{t+\theta} - c_{t-\theta})}{2 * \sum_{\theta=1}^{\Theta} \theta^2} \tag{2.2}$$

2.2.1 UBM

Universal Background Model (UBM) is SRE system's statistical model (GMM - Gaussian Mixture Model) to represent the distribution of feature vectors in the acoustic space. It is represented by many[1] components, which are multidimensional (the dimensionality must be equal to feature vector dimensionality) normal probability distributions. For each feature vector from the recording, the degree of its contribution to each component may be evaluated, thus allowing to convert arbitrarily long sequence of feature vectors to the fixed-size vector of the Baum–Welch statistics[2] collected with the given UBM.

2.2.2 i-vectors

The recording, after being split into frames, derived features from and evaluated through UBM, is represented by one vector of numbers. This vector's dimensionality is given by

$$|\boldsymbol{v}| = C_{\mathrm{UBM}} * |\boldsymbol{f}| \tag{2.3}$$

where C_{UBM} denotes count of UBM components and $|\boldsymbol{f}|$ denotes feature vector size. With 2048 UBM components and 60-dimensional feature vector, the resulting vector has more than 120 000 dimensions and contains redundant information. In order to extract as much diversity as possible from such vectors with significantly reducing their dimensionality, *i-vectors* are used. The origin of the i-vectors is described in [2]. The process of generating them may be viewed as lossy compression and may be taken as low-dimensional [3] vector representing the recording in low-dimensional acoustic subspace.

2.2.3 PLDA

The remaining part in the system is converting the pair of i-vectors to the score (called *i-vector scoring*. In order to compensate i-vectors' different size and bias, usually mean normalization (2.4a) and L2 normalization (2.4b) are performed. We denote the original i-vector as $\boldsymbol{x}$, i-vectors' mean as $\boldsymbol{\mu}$, mean-normalized i-vector as $\boldsymbol{x_n}$. $|\boldsymbol{x}|_2$ represents the L2-normalized vector, and x_r denotes single elements of the i-vector.

$$\boldsymbol{x_n} = \boldsymbol{x} - \boldsymbol{\mu} \tag{2.4a}$$

$$|\boldsymbol{x}|_2 = \sqrt{\sum_{r=1}^{N} |x_r|^2} \tag{2.4b}$$

Scoring with PLDA (Probabilistic Linear Discriminant Analysis) understands the i-vector as $\boldsymbol{\phi} = \boldsymbol{\mu} + \boldsymbol{V}\boldsymbol{y} + \boldsymbol{U}\boldsymbol{x} + \boldsymbol{\epsilon}$, where μ is i-vector mean (zero if normalized), U and V are matrices derived (see 2.5) from system's within-class (Σ_{wc}) and across-class (Σ_{ac}) covariance matrices respectively, y and x are hidden variables describing the speaker and the channel respectively and ϵ is the residual variability. The portion $\boldsymbol{\mu} + \boldsymbol{V}\boldsymbol{y}$ describes the speaker, the latter $(\boldsymbol{U}\boldsymbol{x} + \boldsymbol{\epsilon})$ describes the channel.

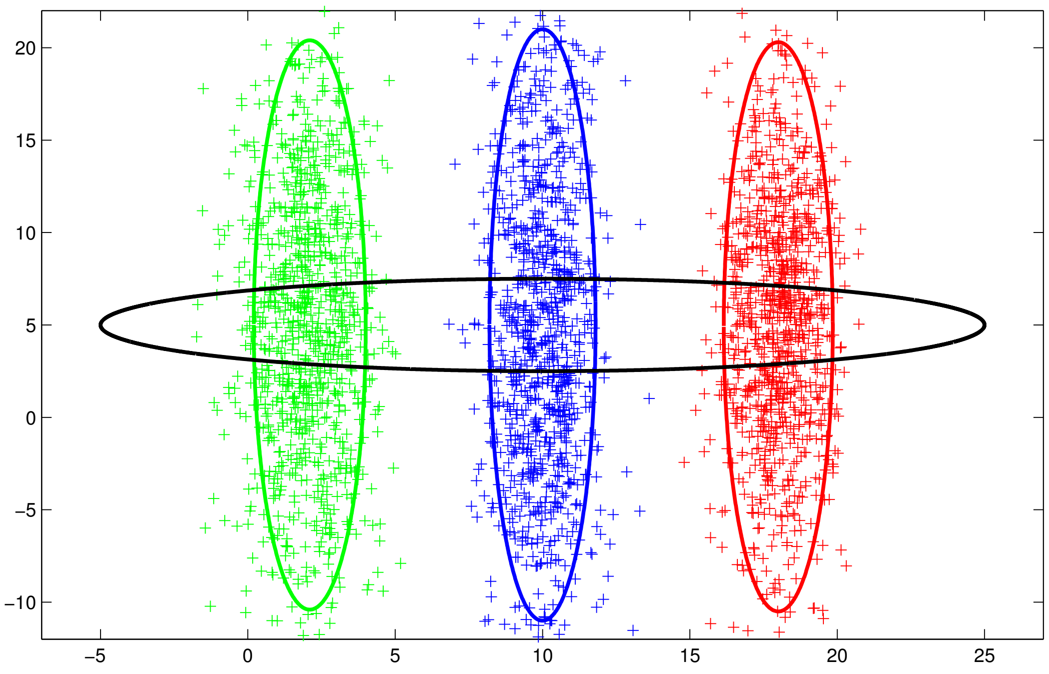
$$\begin{gathered} \Sigma_{ac} = UU' \\ \Sigma_{wc} = VV' \end{gathered} \tag{2.5}$$

The covariance matrices are subject to adaptation (see 2.3).

As stated by [2], with the above assumptions the SRE task may be reinterpreted as whether the $\boldsymbol{y}$ might be the same for both i-vectors. The score can be calculated as (taken from [2])

[1] 2048 in used system
[2] see [3, p. 2] for definition
[3] 600 in used system

Figure 2.3: PLDA intra-speaker and inter-speaker distributions



$$s(\boldsymbol{\phi}_1, \boldsymbol{\phi}_2) = \frac{\int p(\boldsymbol{\phi}_1|\boldsymbol{y})p(\boldsymbol{\phi}_2|\boldsymbol{y})p(\boldsymbol{y})\,\mathrm{d}\boldsymbol{y}}{p(\boldsymbol{\phi}_1)p(\boldsymbol{\phi}_2)}\,, \tag{2.6}$$

where $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ denote compared i-vectors and $\boldsymbol{y}$ denotes certain speaker factor.

2.3 Adaptation

Each recording (or a waveform) can be understood as a mix of two signals. The signal actually produced by the speaker, and the environmental influence referred to as a *channel*. The channel consists of the background noise, microphone quality effects, transmission signal changes, etc. The channel effect decreases SRE performance as it includes information not related to speaker identification to the signal. The SRE system can to some level compensate the channel, if it was provided data with that channel during training. When an SRE system is to be used on another type of data, that it was trained on, it is advisable to run system adaptation to eliminate the channel's distortion.

When an SRE system is being developed, several of its components (UBM, i-vector extractor, PLDA) must be provided with *train data*. UBM and i-vector extractor can be trained with unlabeled data; PLDA, however, requires significant amount of labeled recordings (labeled recordings are recordings with known speaker) to be trained. The train data, however, may not come from the same domain as the application data. In order to adjust themselves to new conditions, SRE systems may allow their users to retrain their components (change PLDA covariance matrices) on labeled data from the target domain.

This process is called SRE system **adaptation**.[4] Although it is possible to train a system on target data directly, there may not be enough labeled data available, as SRE system training requires several thousands of recordings. For adaptation, few hundreds of recordings may be sufficient. The impact of the count of adaptation recordings to system's performance improvement is evaluated by experiments.

I will use the term „adaptation“ to refer to PLDA adaptation only, as it is the only kind of adaptation provided by used SRE system (Phonexia production SID system). [5]

2.4 Calibration

For this work, SRE system calibration is considered as the process of finding its optimal[6] threshold value and shifting it to zero. For each application, different requirements may be laid upon the system in terms of desired percentage of misses or false alarms. Also, when trained on data from certain domain and run (evaluated) on data from another domain, the system may return higher or lower scores. Even a system, which is well trained and adapted to the target domain, will provide bad results, if not calibrated.

Calibration requires several dozens of labeled recordings, being much less demanding in this way than training a new system and adaptation. The process of calibration consists of running the system on known trials and trying to find a threshold value, which is optimal for desired application.

SRE system calibration provides two numerical values, scale and shift. For well calibrated system, if produced score is scaled and shifted by calibration values, the optimal threshold will be zero. The figure 2.4 depicts a calibrated system, where the false-alarm cost

2.5 System accuracy evaluation

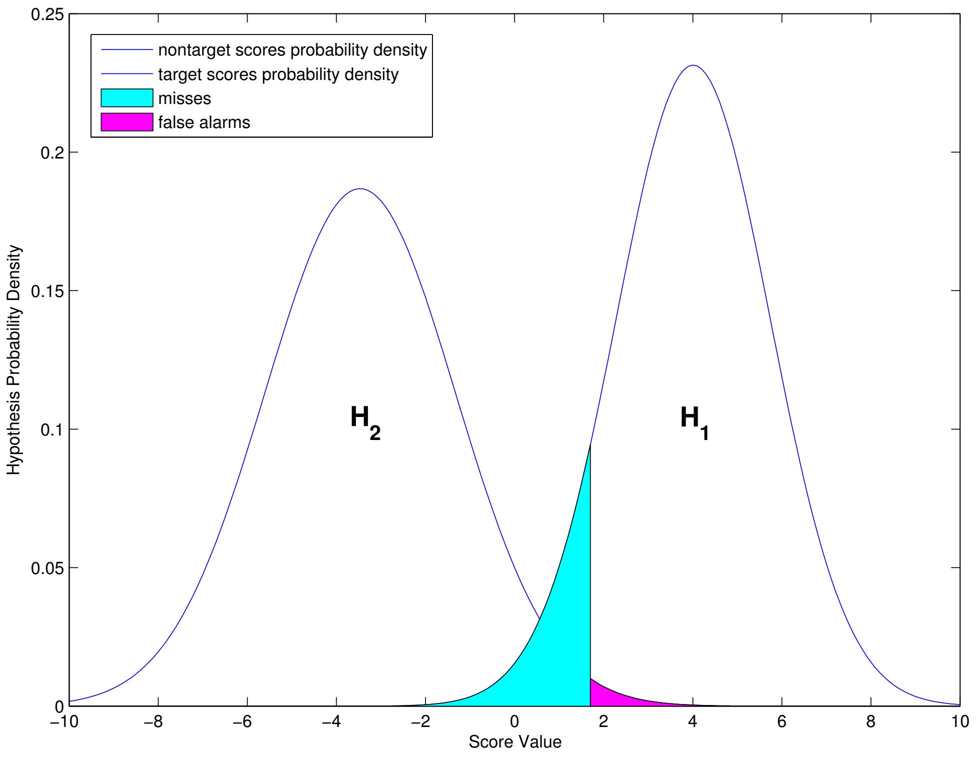
In order to evaluate system's performance and quality, several metrics are used. All of them expect the system to be run on a labeled set of data. The dataset consists of

- list of speakers
- list of trials with results

The system runs the trials from the list and provides its results. Comparison to the correct result from the dataset list provides lists of scores produced by the system for both target and nontarget trials. The overall quality of the system (disregarding calibration) is given by separability of those two lists. Multiple system's *operating points* can be evaluated by setting the decision value to such, that provides given percentage of false alarms / misses (e.g. to get 10% FA operating point, a value, which classifies 10 percent of trials from target list as nontargets, is chosen). Interpolation of all operating points determines a function to describe system's quality. This function's graphical representation is called DET (Detection Error Tradeoff) graph, and it is often used to graphically represent system's quality at given operating points. Examples of DET curves are depicted in figure 2.5. Because of their low resolution in the area of interest and rather bad ability to visualize operating points

[4]The Phonexia SRE system used in the experiments, allows only this kind of adaptation
[5]Development SRE systems may provide other ways for adaptation
[6]for the given metric

Figure 2.4: Nontarget and target score distributions with threshold moved to certain value due to the calibration.



and differences between single DET curves, linear axes are discouraged. Instead, probit[7] function is used for the axes, providing high resolution in the area of interest and linearizing the DET curve.

For more exact (numerical) evaluation, either an operating point may be chosen, or a specific value called EER may be calculated. EER (Equal Error Rate) is the operating point, where the probability of false alarm and miss are equal. Let us denote arbitrary threshold values x_k. The threshold value for EER shall be denoted by v. $P_{FA}(x_k)$ and $P_{miss}(x_k)$ denote false alarm probability and miss probability for the selected threshold value x_k, respectively.

$$x_1 > x_2 => P_{FA}(x_1) < P_{FA}(x_2) \wedge P_{miss}(x_1) > P_{miss}(x_2) \tag{2.7a}$$
$$v = x_n | P_{FA}(x_n) = P_{miss}(x_n) \tag{2.7b}$$
$$EER = P_{FA}(v) \tag{2.7c}$$

Both DET curve and EER only describe system's ability to separate target and non-target trials. They do not use system's hard decision; instead they take into account all possible threshold values. In order to represent system's performance with current threshold, *Detection Cost Function - DCF* metric was established. We can obtain its value using [8]

$$C_{\text{Det}} = C_{\text{Miss}} \times P_{\text{Miss}|\text{Target}} \times P_{\text{Target}} + C_{\text{FA}} \times P_{\text{FA}|\text{NonTarget}} \times (1 - P_{\text{Target}}) \tag{2.8}$$

C_{Miss} and C_{FA} are miss and false-alarm costs, respectively. In some applications, one type of error may be more negative than other type. For basic evaluations, both can be set to 1. P_{Target} denotes prior probability of target trial and $P_{\text{Miss}|\text{Target}}$ and $P_{\text{FA}|\text{NonTarget}}$ are conditional probabilities of miss occurring for target trial and false alarm occurring on nontarget trial, respectively. Target prior and detection error costs are set for desired application, conditional probabilities arise from system's performance.

DCF's value is usually normalized by

$$C_{\text{Norm}} = C_{\text{Det}} / C_{\text{Default}} \,, \tag{2.9}$$

where C_{Default} is the best cost that could be reached without observing the data, that is the cost received by either claiming all trials target or claiming all trials nontarget.

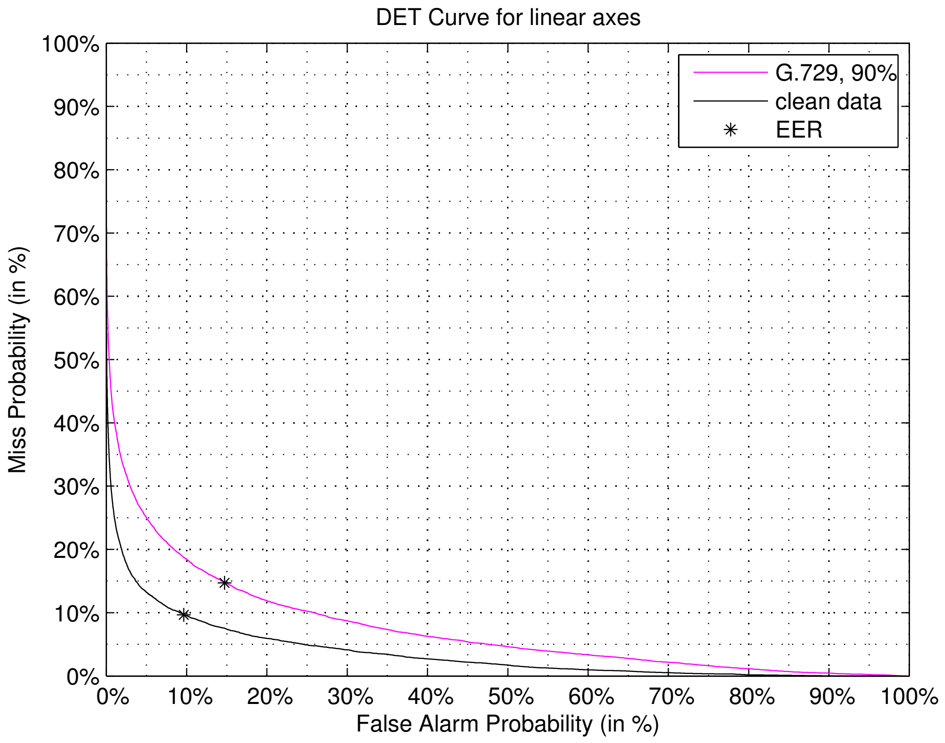
$$C_{\text{Default}} = min \begin{Bmatrix} C_{\text{Miss}} \times P_{\text{Target}}, \\ C_{\text{FA}} \times (1 - P_{\text{Target}}) \end{Bmatrix} \tag{2.10}$$

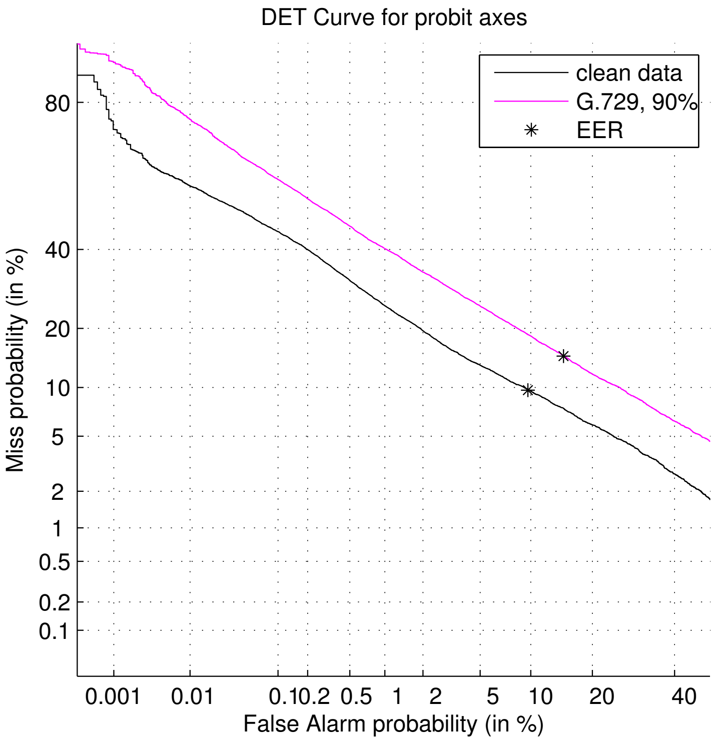
For a system, usually two DCF values are calculated. *actDCF* is DCF value for current system's hard decisions, taking zero as threshold value. DCF value for theoretically optimally calibrated system is called *minDCF*. Note that minDCF $\leq$ actDCF and quotient $\frac{\text{actDCF}}{\text{minDCF}}$ can be used to estimate system's calibration.

[7]inverse cumulative distribution function of normal probability distribution

[8]Formulas taken from [1]

Figure 2.5: Examples of DET curves for both linear and probit axes





Chapter 3

Voice over IP (VoIP)

Voice over IP, a technology for communicating over computer networks, is widely used throughout the world. Companies utilize its ability for central management and reduced need for cabling, end users choose it for its cost. VoIP devices allow their users to communicate by encoding audio signals produced by them to network packets and sending them to the other participant over the IP network (usually Internet).

Unlike the classical telephony network, which is *circuit-switched*, IP networks are *packet-switched*. This fact allows multiple logical connections to share one physical connection, but it brings quality problems. In the classical telephony network, the physical link is dedicated to one call at a time. This allows the analog voice signal data to be transmitted as-is. VoIP technology, where a media is shared among many users, needs to search ways to reduce required *bandwidth* (amount of data transferred in a time unit).

3.1 VoIP call

When a VoIP user wants to communicate, he uses his VoIP phone and dials the other one's number. Before the communication starts, the caller's phone needs to do two things:

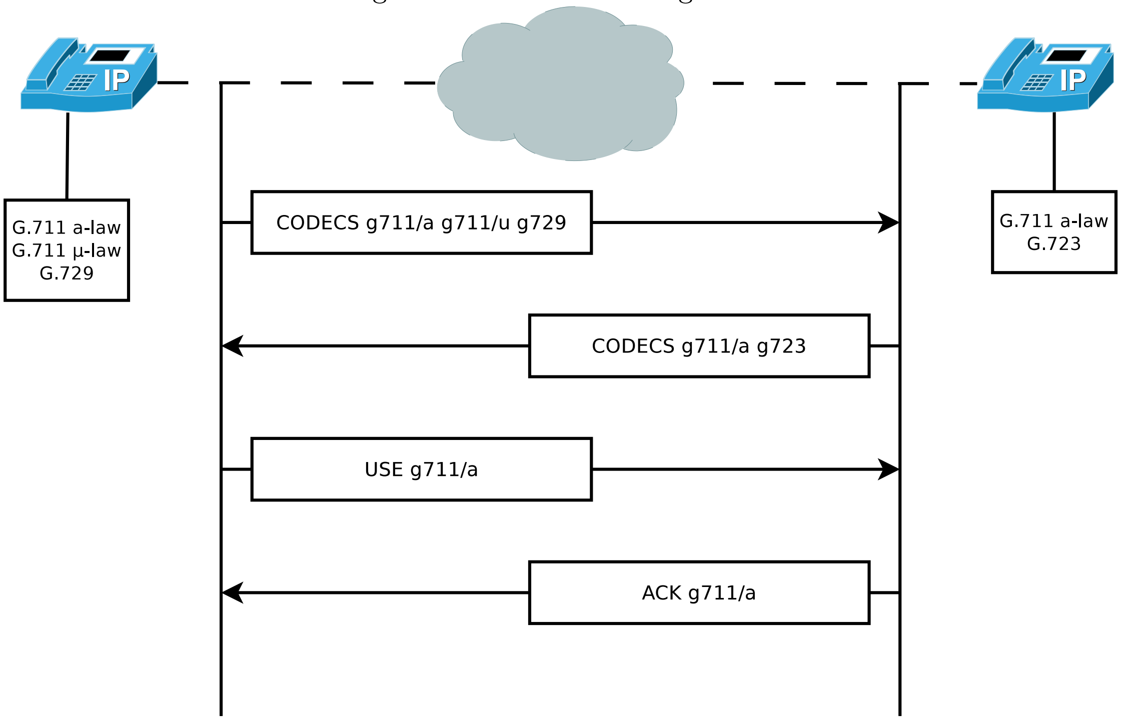
- locate the callee's phone on the network
- negotiate transferred voice data format

As both devices need to transmit and receive digital data and interpret it as voice, they need to negotiate the data format, along with codec (see 3.3) used to encode it. There are several internationally standardized codecs (described in 3.3) and each device may support a different set. Both devices inform each other about their format capabilities and negotiate a common format best suiting their requirements. This process is called *codec negotiation* and is depicted on figure 3.1. When codec negotiation is finished, devices start transmitting data and communicate.

3.2 Encoding

In order to represent a continuous audio signal in digital environment (such as computers or VoIP phones), the signal needs to be encoded in a digital form. Usually, a sequence of numbers representing signal's actual amplitude is used. To achieve this, two operations must be run on the signal: *sampling* and *quantization*.

Figure 3.1: VoIP codec negotiation



Sampling is the process of converting continuous audio signal into the digital form. This is achieved by periodically evaluating original signal's instantaneous amplitude. The signal may be sampled at various frequencies, but, according to the Nyquist theorem (see [6]), sampling frequency must be at least double of sampled signal's frequency (maximal frequency in case of compound signals) in order to allow perfect signal reconstruction.

$$f_{\text{sampling}} \geq 2 * f_{\text{signal}} \tag{3.1}$$

The minimal requested frequency is called *Nyquist frequency*. Sampling a signal at frequency below Nyquist frequency results in signal aliasing. Low-pass filters are used to eliminate high frequencies from the signal before sampling is done to avoid aliasing. Human voice is an audio signal with most of the energy within 400 – 3500 Hz range. Subsequently, in VoIP applications, which aim to encode human voice, signal is usually sampled at 8 kHz. There are so-called „wideband“ encoders sampling at 16 kHz to receive better quality, but in this work, only 8 kHz sampled audio is used.

Quantization is the process of constraining real value of signal's amplitude to limited discrete range. Size of the range determines digital signal's quality - „smoothness“. In computer environment, data sizes are usually chosen to fit whole bytes. For audio, usually 16bit (2byte) sample size is used. This gives 65536 possible values of amplitude while preserving low sample size.

As a result, required bandwidth for described audio is 128 kbps (8 000 samples / second * 16 bits / samples).

$$\text{Bandwidth} = \text{sample rate} * \text{sample size} \tag{3.2}$$

3.3 VOIP codecs

The word *c*odec stands for coder-decoder. Codecs are pairs of algorithms designed for encoding and decoding data from/to desired formats. The VoIP technology utilizes codecs for encoding audio data to a representation suitable for being transferred via packet-switched networks. Each VoIP device is equipped with several codecs - they define the device's support for audio data formats. This section will describe VoIP codecs' requirements, benefits and limitations. *N*ote: I will use the word 'codec' for VoIP codecs.

VoIP codecs are designed to meet two contradictory requirements: audio quality and low bandwidth. There are various codecs, each combining those requirements differently, therefore suitable for different environments. Audio quality for the codec is often measured by MOS[1] metrics, which scores codec on 1–5 scale, where 5 means excellent audio and 1 means very bad quality.

There are two basic codec implementations. Waveform coders compress the audio signal and transmit it as such, vocoders derive speech parameters and spectral characteristics and transmit those only, which can result in bandwidth save.

3.3.1 Silence transmission

When two people talk to each other on the phone, usually one of them is silent, when the other one speaks. Therefore, there is a lot of silence in the signal, which need not be transmitted. Several codecs allow to transmit special SID (Silence Insertion Descriptor) packet instead of the actual signal. As the SID packet can be very short, the required bandwidth can be significantly decreased.

For the hearing side, absolute silence is uncomfortable, as one can always hear at least background noise, even though the talker is silent. To emulate this, codecs use module called CNG (Comfort Noise Generator) to play low-level noise to the hearing side when SID packets are received.

3.3.2 ITU-T G.711

G.711 is a waveform coder with 64 kbit/s bandwidth and MOS 4.11. It uses 12 most significant bits of each speech sample and converts those into 8-bit logarithmic scale. With human hearing being logarithmic as well, the codec saves one third of the bandwidth without significant audio quality decrease. Its bandwidth determines it to be used for high-speed networks and/or where audio quality is crucial. Several enhancement were introduced, including wideband version of the codec.

Two different encodings are used with G.711. *μ-law* encoding is used primarily in USA and Japan. It is optimized for better audio quality. The *A-law* is computationally simpler form of encoding used in the rest of the world. Both encodings show similar characteristics, subsequently only more common *A-law* encoding is used in this work.

3.3.3 ITU-T G.723.1

G.723.1 is a multi-rate vocoder with two available operating modes, 5,3 kbit/s with MOS 3.62 and 6,3 kbit/s with MOS 3.9. It provides very good performance in low-bandwidth spectrum of codecs, so it is usually used in conditions, where very low bandwidth is a requirement. In this work, 6,3 kbit/s mode is used.

[1]See [5] for closer information

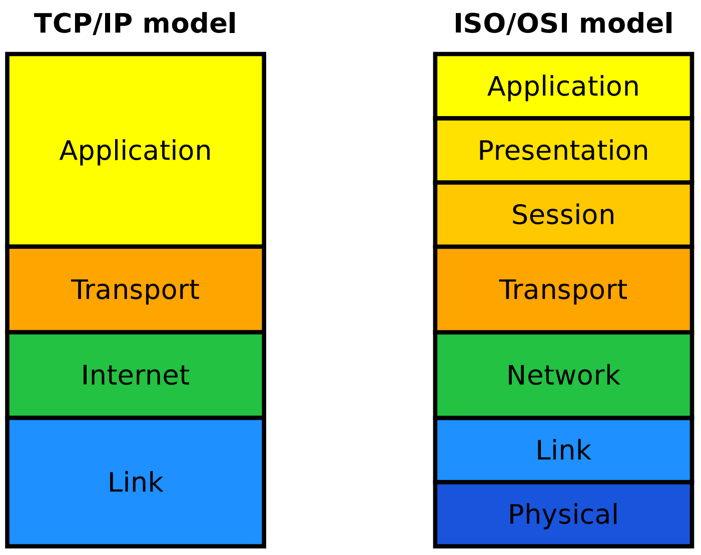


Figure 3.2: TCP/IP and ISO/OSI network models

3.3.4 ITU-T G.729

G.729 is code-excited linear prediction (CS-ACELP) vocoder with 8 kbit/s bandwidth and MOS of 3.92. There are many extensions; ITU-T has defined 10 annexes (Annex A - Annex J) and two appendices, which define other transmission rates (6,4 kbit/s and 11,8 kbit/s), floating point implementation or DTX (discontinuous transmission, see 3.3.1). G.729 offers slightly better audio quality then G.723.1 with small trade in bandwidth, it is therefore broadly used as low-bandwidth codec. In this work, ITU-T G.729 codec without any annexes or appendices is considered.

3.3.5 GSM AMR

AMR is multi-rate vocoder, operating at various bandwidths from 4,75 to 12,2 kbit/s. It was developed for mobile phone networks, but for its performance in low bandwidths, it is utilized in low-bandwidth VoIP applications. In this work, AMR codec operating at 4,7 kbit/s (with MOS 2.59) is considered.

3.4 Networking

Most of today's network devices implement the TCP/IP network model (a simplified version of the ISO/OSI model), see 3.4. This model divides network protocols and services to layers, where each layer is based on the services of the lower one and provides services to the upper one.

In TCP/IP stack, protocols of the transport layer are responsible for delivering data to specific application. Two protocols, called TCP[2] and UDP[3] are commonly used.

TCP protocol implements connection-oriented communication. This means, that communicating processes have to establish a connection, transfer data and close it. This type of connection provides reliable connection — TCP protocol ensures all packets are received

[2]Transmission Control Protocol
[3]User Datagram Protocol

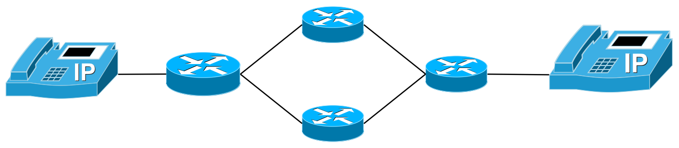


Figure 3.3: VoIP network example

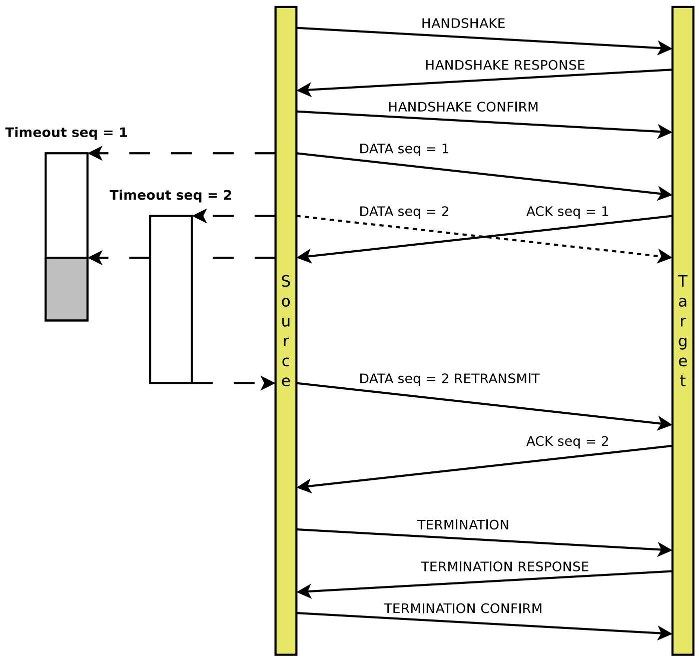
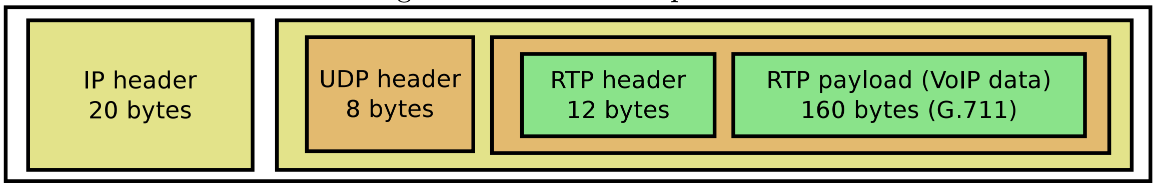


Figure 3.4: TCP communication example

and they are correctly ordered. Lower layers ensure packet transmission and routing, but since the packets can take various paths (see figure 3.3) and they may get lost, receiving side is not guaranteed to have received all the packets. TCP protocol utilizes sequence numbers encoded in packets to deal with packet reordering and response packets from target to source confirming accepting the data. When a packet is sent, its sender starts a timer for awaiting the response (acknowledgment) packet confirming, that given packet was received. If the acknowledgment is not received back in time (timer expires), the packet is considered lost and it is resent. Figure 3.4 shows simplified example of TCP communication.

TCP protocol has some drawbacks, though. The need for acknowledgment packets and big size of TCP header (20 bytes compared to 8-byte UDP header) increase bandwidth requirements (although TCP acknowledgment may be included in data packets sent by the other side). Transmission of VoIP data requires more bandwidth than used codec's bandwidth. This is cause by need to encapsulate VoIP data to network protocols (see figure 3.5. However, the bandwidth increase is not the biggest issue in VoIP. The biggest problem is the delay. TCP provides data in correct order. That means, if a packet is missing, no data is delivered to the application (VoIP process), it is rather stored by receiving side's TCP layer until missing packet is retransmitted and received; data from both packets are provided to the VoIP application after that. This behaviour would result in VoIP calls being interrupted (when packet was not received and sender's timer has not expired yet). For this reason, UDP protocol is used in VoIP applications. UDP protocol was designed

Figure 3.5: Packet encapsulation



Payload of each protocol contains header and payload of higher level protocol

for applications which do not require reliable connection or prefer real-time delivery over ensuring everything was delivered. It only provides checksum to ensure packet was not damaged.

Utilizing UDP enables VoIP transmission to keep the real-time requirements, but bring s problems in form of possibly missing or reordered packets. The RTP[4] protocol used for VoIP at the application level uses sequence numbers, which provides VoIP application with information about missing or reordered packets, but leaves the application to process the data and choose action without waiting or all preceding packets.

When a packet is missing during VoIP call, the receiving side VoIP application cannot wait for its retransmission as it needs to play data contained in it in real-time. It may deal with this situation by either playing silence or generated noise instead of the audio from missing packet, or try to extrapolate the waveform from previous packets (e.g. repeat the last accepted packet).

For SRE, however, the audio, that is played for the missing packet, has no relevance; either way it has been generated by software, not produced by actual human speech and therefore contains no valuable information. In this work I assume, that the receiver ignores missing packets and considers two subsequently accepted packets to be subsequent; this may result in output audio being shorter than input audio.

Wrong order of packets is not considered. There is no need for the SRE to be done real-time, so the packets may be reorganized before passed to the decoder.

[4]Real-time Transport Protocol

Chapter 4

Evaluations & Experiments

The aim of this work is to evaluate chosen system's performance on VoIP data, to analyze aspects, that may decrease system's performance and to discuss approaches of eliminating those aspects. In this chapter, I will describe used SRE system and its interface, used data set, VoIP traffic simulation environment and results of conducted experiments.

For the experiments, MathWorks MATLAB software was used, along with the BOSARIS Toolkit[1] and non-public evaluation scripts by Ing. Oldřich Plchot for computing the evaluation metrics.

4.1 SRE system

Throughout this work, Phonexia SID system[2] is used, particularly version 2.4.0, command-line interface for Linux 64-bit. The SID system provides two executable files: `vpcompare` and `vpextract`. `vpextract` can be used for creating so-called *voice-prints*, which are file-formatted i-vectors (described in section 2.2). `vpcompare` can be used for comparing voice-prints and producing score (PLDA). It provides ways to conduct system adaptation (creation of so-called *model* — set of parameters, which modify internal PLDA covariance matrices) and allows to specify adaptation constant to set the weight of newly trained model against built-in one.

The `vpextract` program accepts a single file, list of files or a directory as an input and provides single file or directory of files (voice-prints) as an output. The input files are audio files in RAW format, 16-bit signed linear encoded PCM. I decided to use directory to directory, as I wanted voice-prints created from all audio files (audio data will be described in section 4.2). `vpcompare` accepts (as input) two filenames, two lists of files or two directories (voice-prints in either case). In the first case, it compares given files and produces single score. The latter cases are considered 'enroll list (directory) to test list (directory)', all comparisons for Cartesian product of the lists are run and evaluated. The result is written into a matrix file. This file will be referred to as SCOREFILE, it is subject to further processing and evaluation.

For adaptation, `vpcompare` accepts list of training voice-prints with speaker information (either by dividing them into subfolders or by supplying a file with speaker name in first column and voice-print name in second one. Adaptation provides results in the form of

[1]https://sites.google.com/site/bosaristoolkit
[2]http://phonexia.cz/technologies/sid

„model directory“. This directory may be used in subsequent `vpcompare` calls - this is equivalent to using adapted system.

4.2 Audio data set

I have used NIST SRE 2010 audio data set. This data set contains 14270 audio files from both male and female speakers. NIST has defined several *evaluation conditions.* A condition is a triplet of enroll files, test files and trial definitions. For each condition, there are only some trials defined along with their true result. Subsequently, one can evaluate system's performance by having it score trials defined by the condition and compare those results to trial definitions.

I have chosen *condition 2* for its sufficient size and non-difficult evaluation environment. This condition consists of interview recordings, where the speakers were recorded using different set of microphones. Subsequently, the data is rather clean, without distortions caused by telephone transmission. As SRE system's modifications (adaptation and calibration) cannot be evaluated correctly on the same data it was trained on, there was a need to extract rather small held out *development set* for training calibration and adaptation and leave the rest of the files as *evaluation set* to perform system evaluation on. The selection was random with keeping several restrictions.

According to Phonexia SID developers, several dozen speakers with multiple recordings (preferably >10 recordings per speaker) are needed to train system adaptation, 129 speakers (limited to speakers with >10 recordings) were chosen for the development set. There are 2 223 recordings uttered by those speakers, 1 508 of them are 3 minutes long, the rest are 8 minutes. 1 166 of them were uttered by female speakers, 1 057 by male. In some cases, development set was further cut in order to simulate very small amount of labeled data available for system modifications. There are 658 enroll recordings and 1 889 test recordings[3], trial definitions for the development set contains 188 546 trials (3 994 target and 184 552 nontarget trials).

The evaluation set consists of 6 157 recordings uttered by 394 speakers. 3 332 recordings were uttered by female speakers and 2 825 were uttered by male speakers. 3 949 of recordings are 3 minutes long, 2 208 are 8 minutes long. The evaluation set has 1 733 enroll recordings, 5 297 test recordings and 1 536 503 trials in trial definitions (10 609 target and 1 526 066 nontarget). Brief statistics are shown in table 4.1.

The trial definitions for both sets were created as a subset of condition 2 trial definitions in the following way. Let $D = (E_D, T_D, r_D)$ denote a trial definition, where E_D denotes its enroll segment (recording), T_D denotes test segment and r_D denotes trial result. Let us further denote set of development segments as S_{Dev}, set of evaluation segments as S_{Eval} and the whole condition 2 trial definitions as Φ. Resulting development and evaluation trial definitions will be denoted as Φ_{Dev} and Φ_{Eval}, respectively. Then

$$\Phi_{Dev} = \{D | D \in \Phi, E_D \in S_{Dev}, T_D \in S_{Dev}\} \tag{4.1}$$

$$\Phi_{Eval} = \{D | D \in \Phi, E_D \in S_{Eval}, T_D \in S_{Eval}\} \tag{4.2}$$

[3]A recording may be in both enroll and test lists

	Development Set	**Evaluation Set**
Count of recordings	2 223	6 157
Count of speakers	129	394
Enroll recordings	658	1 733
Test recordings	1 889	5 297
Target trials	3 994	10 609
Nontarget trials	184 552	1 526 066

Table 4.1: Basic information about development and evaluation sets

Codec	**Options**	**Bitrate**	**Download URL**
G.711	A-law	64 kbps	http://www.itu.int/rec/T-REC-G.191/
G.729	No Annexes	8 kbps	http://www.itu.int/rec/T-REC-G.729/
G.723.1	Annex A	6.3 kbps	http://www.itu.int/rec/T-REC-G.723.1/
AMR	MR475	4.75 kbps	http://www.3gpp.org/DynaReport/26073.htm

Table 4.2: List of codecs with parameters and modes used in the experiments

4.3 VoIP and network simulation

In order to evaluate influence of VoIP factors on system's performance, VoIP traffic simulating environment was created. VoIP simulation consists of

- encoding audio files with selected codec
- transferring encoded data over (lossy) network
- decoding received data with selected codec

ITU-T codecs were obtained from ITU-T web pages. AMR codec was obtained from 3GPP, version Rel-8. For download web page, codec settings and bitrates (referred to as codec bandwidths previously), see table 4.2.

All of the codecs provide reference implementation of both encoder and decoder. All provide the same function — with proper parameters, input file name and output file name, they encode input audio to a bitstream file or decode the bitstream file to audio. The bitstream file contains audio data, that would be sent over network in real VoIP application (although most of the codecs create bigger files, e.g. by encoding each bit of the bitstream into a word (two bytes)).

The network is considered to drop packets by random. Bursts of dropped packets may be observed when routers along the path start to drop packets massively due to network congestion or when the packets are rerouted by errors in the routing tables.

Instead of separating bitstream files into packets and sending them over the network, which would be set to discard some of them, I chose to simulate network errors directly on the data. Since packet corruption is not considered, the packet may only be received and valid, or not received at all. All of the codecs generate fixed-size portions of bitstream files — those portions are considered to be packets to real application. Subsequently, network error simulation is performed by cutting out these portions from the bitstream file. Packets can be cut either by random, in bursts of predefined size or by time marks defined by external file.

Condition Name	Network Quality	Burst size
clean	100%	N/A
95	95%	1
90	90%	1
90_burst5	90%	5
90_burst20	90%	20
80	80%	1
80_burst5	80%	5

Table 4.3: List of used network conditions

Cutting is done by random, keeping preset network quality (percentage of packets received). For burst cutting, packet count is computed from file length and packet length, proper count of discarded packets is computed from preset network quality and discarded packet indexes are chosen by random. Time marks of cut packets can be exported to a file in any mode, using following syntax

```
STARTTIME1 ENDTIME1
STARTTIME2 ENDTIME2
...
...
```

where `STARTTIMEx` and `ENDTIMEx` are integer time marks in milliseconds for the beginning and end of `x`-th cut packet (or packet burst). Time marks are derived from cut packet's index in file and duration of packets (duration of audio information they contain). The duration for specific codec was derived by inspection of codec code (by examining the count of samples read in the process of creating a single packet), it ranges from 10 to 30 milliseconds.

Exporting time marks into external file allowed me to cut out packets representing the same time portion of audio files for different codecs, even though the packets were of different duration. This was done in order to eliminate possible differences between codecs produced by having them process different random portions of files.

Encoding all the audio files with all (4) the codecs, subsequent processing of the encoded data with network simulator (`cutter`) using several (7) conditions (described in table 4.3) and their decoding with their codec produced 28 audio files sets, further referred to as sets. They will be denoted `<codec>.<condition> (g711.clean)` or `<CODEC>, <QUALITY> (G.711, 100%)`.

4.4 Performance Evaluation

One of the goals of this work was to determine VoIP distortions' impact on SRE system's performance. Using tools and setup described above, untouched SRE system was used to evaluate the same set of data under several VoIP conditions. From the results, EERs are shown in table 4.5 and minDCF statistics are shown in table 4.6. For DCF computing, „old“ SRE parameters (described in [1, Table 3]) were used. For clean data, all collected statistics are in table 4.4

These results indicate, that used codec and compression has far greater importance for SRE performance, than percentage of data missing. Conditions with bursts of missing

Metric	**Value**	**Description**
miss10	9.21%	Miss percentage at 10% FA rate
fa2p5	17.76%	FA percentage at 2.5% miss rate
actDCF	3.3527	Actual DCF (hard decisions)
minDCF	0.3487	Minimal possible DCF (see 2.5)
EER	9.64%	Equal Error Rate (see 2.5)

Table 4.4: SRE performance metrics for clean data

	clean	**95**	**90**	**90_burst5**	**90_burst20**	**80**	**80_burst5**
G.711	10.06%	10.10%	10.28%	10.65%	10.37%	10.93%	11.05%
G.729	13.90%	14.33%	14.72%	14.38%	14.10%	16.12%	15.08%
G.723.1	15.52%	15.71%	16.47%	16.11%	15.89%	17.74%	16.92%
AMR	18.13%	18.82%	18.95%	18.84%	18.80%	20.99%	19.64%

Table 4.5: EERs of unmodified (original) SRE system for all network conditions

	clean	**95**	**90**	**90_burst5**	**90_burst20**	**80**	**80_burst5**
G.711	0.7032	0.6965	0.6982	0.7062	0.7204	0.7003	0.7191
G.729	0.8098	0.8351	0.8472	0.8184	0.8225	0.8871	0.8510
G.723.1	0.8747	0.8758	0.8922	0.8854	0.8809	0.9154	0.8978
AMR	0.9153	0.9247	0.9311	0.9278	0.9217	0.9475	0.9342

Table 4.6: minDCFs of unmodified (original) SRE system for all network conditions

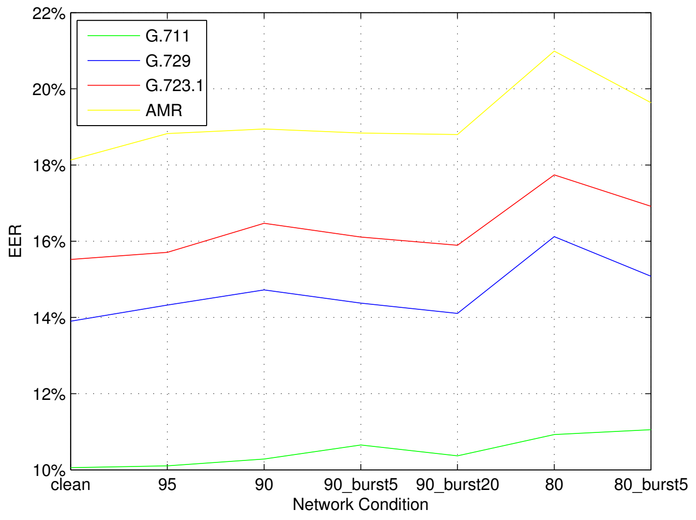


Figure 4.1: Comparison of EERs for used codecs (default system)

packets show worse results, than their random counterparts for waveform codec G.711. For other codecs (all vocoders), they show slightly better results.

Next section will describe SRE system adaptation and its impact on the performance improvement. In order for the system to provide good results, system calibration should be performed after the system is well-adapted.

4.5 Adaptation Impact

The system adaptation was trained on each VoIP condition data set. As the training provides model directories (described in 4.1), a condition may be evaluated using system adapted on another condition. This will be referred to as „cross-adaptation“.

The table 4.7 shows EERS for both untouched system and system adapted on full development set for the given condition.

	G.711		**G.729**		**G.723.1**		**AMR**	
	def.	adp.	def.	adp.	def.	adp.	def.	adp.
clean	10.06%	8.26%	13.90%	10.24%	15.52%	10.20%	18.13%	11.94%
95	10.10%	8.19%	14.33%	10.44%	15.71%	10.63%	18.82%	12.64%
90	10.28%	8.21%	14.72%	10.42%	16.47%	10.75%	18.95%	12.32%
90_burst5	10.65%	8.44%	14.38%	10.64%	16.11%	10.69%	18.84%	12.71%
90_burst20	10.37%	8.58%	14.10%	10.31%	15.89%	10.49%	18.80%	12.47%
80	10.93%	8.62%	16.12%	11.36%	17.74%	11.55%	20.99%	13.76%
80_burst5	11.05%	9.04%	15.08%	11.00%	16.92%	11.16%	19.64%	13.21%

Table 4.7: EERs for both unmodified (default, def.) and fully adapted (adp.) system, AC = 0.9

Adaptations with AC (adaptation constants, see 4.1) from 0.3 to 0.95 were run on selected conditions `g711.80`, `amr.clean` and `g729.90_burst5` to find its optimal value. The results (EERs) of those runs are in table 4.8. The best results were achieved with the constant 0.9 for most of the conditions and that value was subsequently used for all conditions.

Adaptation const.	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**0.95**
g729.90_burst5	11.29%	11.05%	10.89%	10.75%	10.64%	10.70%
g711.80	9.12%	8.96%	8.83%	8.69%	8.62%	8.72%
amr.clean	13.11%	12.77%	12.49%	12.19%	11.94%	11.92%

Table 4.8: EERs of chosen conditions for different values of the adaptation constant

The adaptation causes the SRE performance to improve significantly (over 7% EER at amr.80 condition. However, these results were achieved with adaptation run on full development set - that means 2 223 labeled recordings. In many situations, however, such amount of labeled recordings may not be available. Following limited development sets were created by randomly selecting desired count of speakers and selecting all recordings uttered by those speakers (approx. 17 recordings per speaker)

- adp20 with 20 speakers and 352 recordings,
- adp40 with 40 speakers and 711 recordings,
- adp80 with 80 speakers and 1386 recordings.

The optimal adaptation constant has been searched (using the same procedure as for the AC for the full development set, except changing the range to 0.2 - 0.9) for all of the limited sets. Chosen conditions EERs for various values of AC for `adp80` are depicted on figure 4.2. ACs chosen for the limited development sets are in table 4.9.

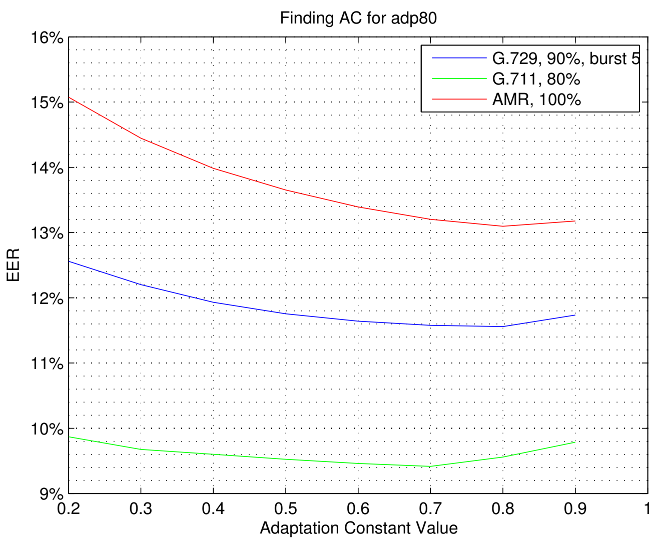


Figure 4.2: EERs for chosen conditions in searching the optimal value of AC for `adp80`

Development Set	Chosen AC
adp20	0.5
adp40	0.6
adp80	0.75

Table 4.9: Chosen ACs for limited development sets

The results of evaluating the performance of the SRE system adapted with the limited development sets are shown in tables 4.10, 4.11, 4.12 and 4.13. Comparison of improvements for each network condition and limited development set can be viewed at figures 4.3 (G.711) and 4.4 (AMR). The graphical representation of the performance is given at figures 4.3 and ??.

	none	adp20	adp40	adp80	full
clean	10.06%	9.59%	9.28%	8.90%	8.26%
95	10.10%	9.60%	9.31%	8.84%	8.19%
90	10.28%	9.76%	9.38%	8.80%	8.22%
90_burst5	10.65%	10.10%	9.69%	9.25%	8.44%
90_burst20	10.37%	9.95%	9.60%	9.16%	8.58%
80	10.93%	10.50%	9.95%	9.46%	8.62%
80_burst5	11.05%	10.47%	10.04%	9.60%	9.04%

Table 4.10: G.711 EERs for all adaptation sets, including original system's performance (column 'none')

The results indicate, that the improvement of the performance is strongly dependent on the size of the adaptation set. However, even a relatively small set (`adp20`) was able to improve system's performance for as much as 3.5% for AMR, 80% condition. Generally, the improvement is greater for the conditions, which show generally worse results (0.42% difference between non-adapted and adp20-adapted g711.90_burst20 compared to 3.78%

	none	**adp20**	**adp40**	**adp80**	**full**
clean	13.90%	12.23%	11.85%	11.18%	10.24%
95	14.33%	12.50%	11.98%	11.17%	10.44%
90	14.72%	13.00%	12.24%	11.27%	10.42%
90_burst5	14.38%	12.90%	12.23%	11.58%	10.64%
90_burst20	14.10%	12.78%	12.14%	11.24%	10.31%
80	16.12%	13.99%	13.05%	12.41%	11.36%
80_burst5	15.08%	13.35%	12.88%	11.93%	11.00%

Table 4.11: G.729 EERs for all adaptation sets, including original system's performance (column 'none')

	none	**adp20**	**adp40**	**adp80**	**full**
clean	15.52%	13.02%	12.12%	11.28%	10.21%
95	15.71%	13.35%	12.45%	11.85%	10.63%
90	16.47%	13.97%	12.93%	12.02%	10.75%
90_burst5	16.11%	13.81%	12.88%	11.91%	10.69%
90_burst20	15.89%	13.57%	12.49%	11.64%	10.49%
80	17.74%	14.85%	13.96%	13.00%	11.55%
80_burst5	16.92%	14.51%	13.44%	12.44%	11.16%

Table 4.12: G.723.1 EERs for all adaptation sets, including original system's performance (column 'none')

between the same adaptation sets for amr.80.

In all previous adaptation experiments, systems were evaluated on the data from the condition their adaptation was trained on. However, when there are not enough data from the target domain or the codec or network condition of the adaptation data are unknown, the system can only be adapted on data from the condition other than the target. Results of evaluations of this kind of adaptation is shown in figures 4.5, 4.6 and 4.7. Models trained from full development set were used.

The results indicate that, at least with large enough development set, performance improves significantly even for other codecs. The network condition used for cross-adaptation is of lesser importance than the codec, although cross-adaptation from models trained on bad network conditions (80, 80_burst5) show generally better results. When evaluating on one of good network conditions, the system cross-adapted on bad network condition can even outperform the system adapted on the evaluated condition. G.711 models provide only slight improvement for the tested G.729 and AMR conditions, other combinations give the improvement rather close to the classic adaptation.

	none	adp20	adp40	adp80	full
clean	18.13%	15.04%	14.07%	13.15%	11.94%
95	18.82%	15.67%	14.88%	13.78%	12.64%
90	18.95%	15.76%	14.75%	13.58%	12.32%
90_burst5	18.84%	15.76%	14.96%	13.77%	12.71%
90_burst20	18.80%	15.52%	14.65%	13.51%	12.47%
80	20.99%	17.21%	16.33%	15.21%	13.76%
80_burst5	19.64%	16.51%	15.66%	14.45%	13.21%

Table 4.13: AMR EERs for all adaptation sets, including original system's performance (column 'none')

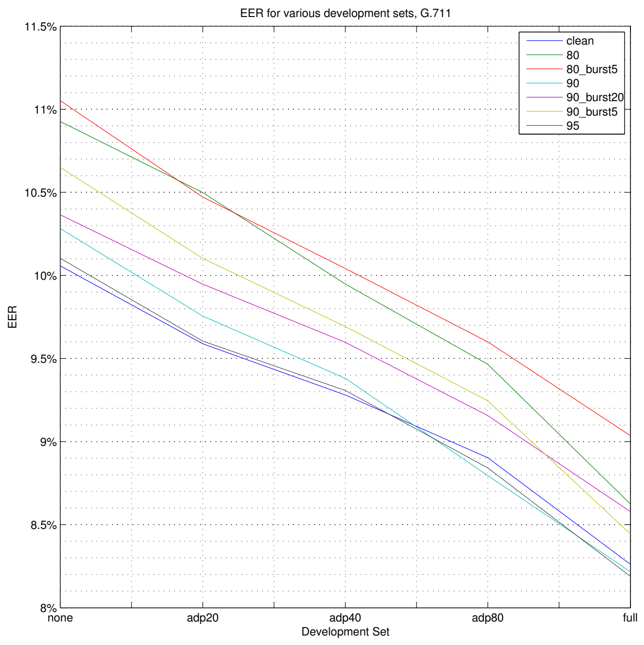


Figure 4.3: SRE system performances for G.711 with various network conditions and adaptation sets

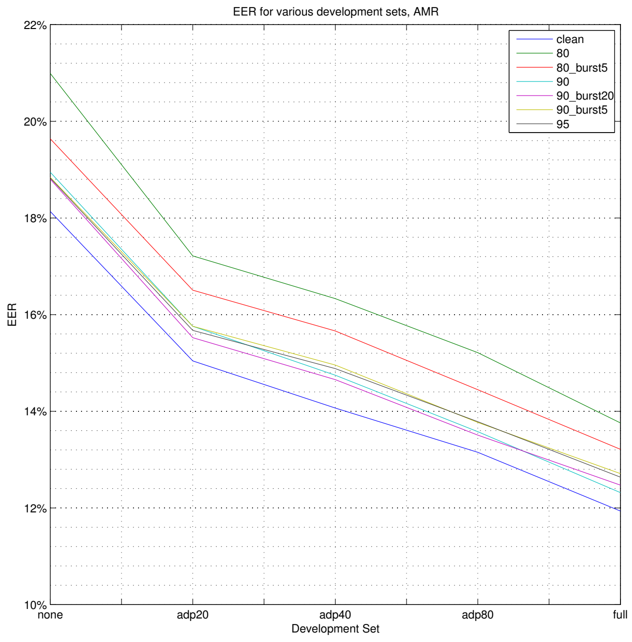


Figure 4.4: SRE system performances for AMR with various network conditions and adaptation sets

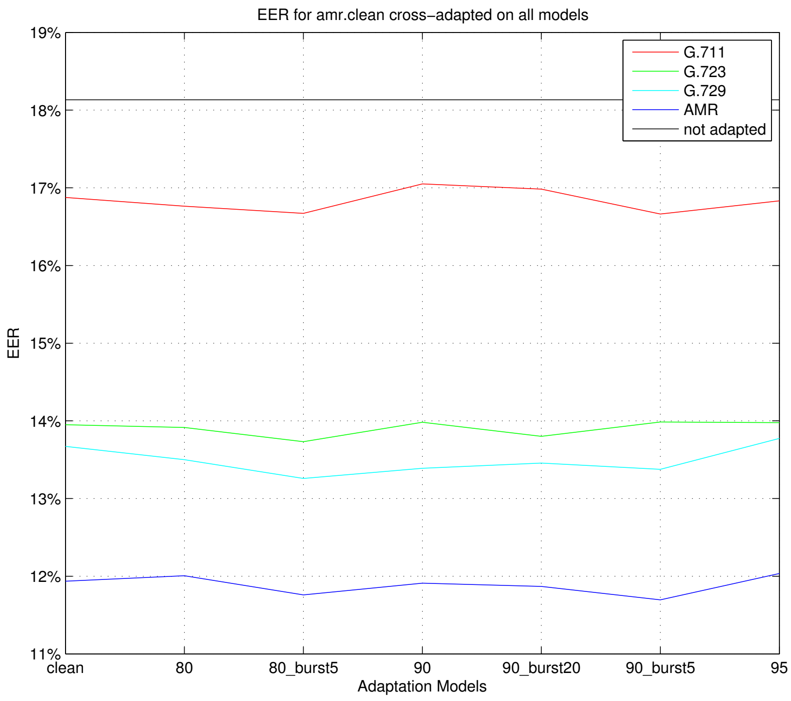


Figure 4.5: EERs of system evaluated on amr.clean when cross-adapted for all conditions

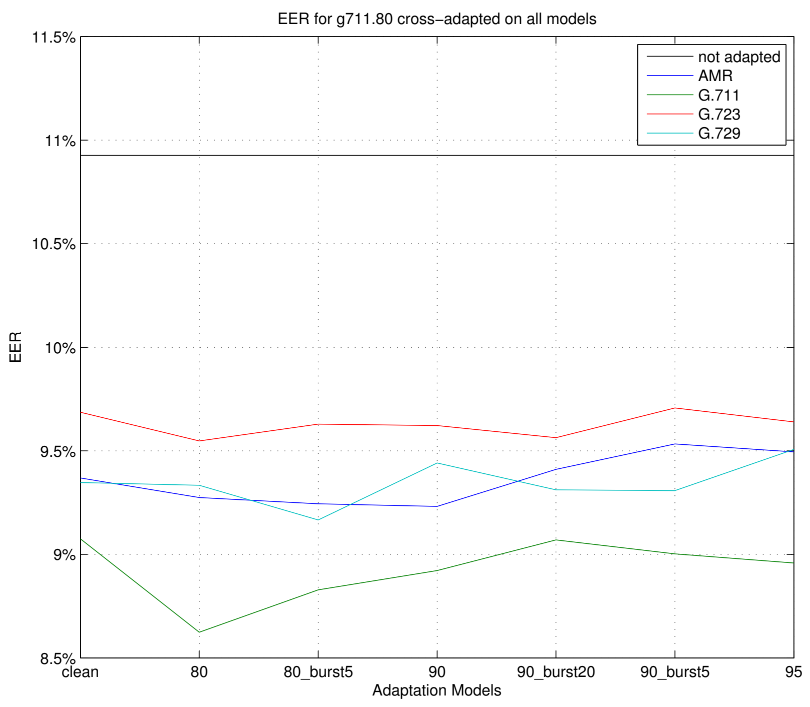


Figure 4.6: EERs of system evaluated on g711.80 when cross-adapted for all conditions

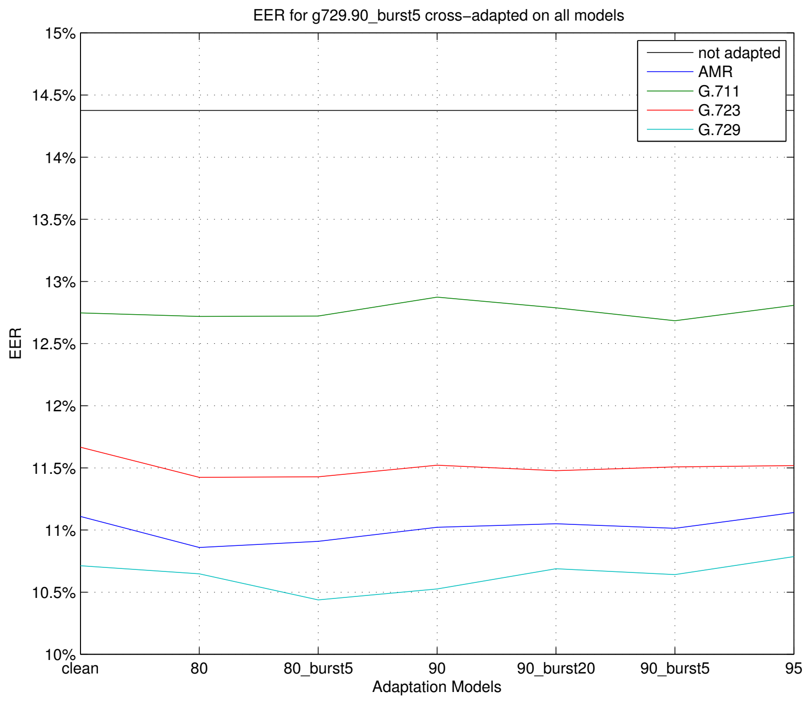


Figure 4.7: EERs of system evaluated on g729.90_burst5 when cross-adapted for all conditions

4.6 Calibration Impact

The experiments in the previous section showed the potential in system adaptation to increase its ability to separate target and nontarget trials. Without proper calibration, though, results provided by such a system are of little value. In this section, experiments will be conducted for chosen conditions `amr.clean`, `g729.90_burst5` and `g711.80`. Unmodified (non-adapted) system will be used.

The calibration will be designed to optimize the actDCF metric. Let me remind the formula for computing DCF.

$$C_{Det} = C_{Miss} \times P_{Miss|Target} \times P_{Target} + C_{FA} \times P_{FA|NonTarget} \times (1 - P_{Target}) \tag{4.3}$$

The costs (C_{Miss} and C_{FA}) and the prior probability of target trial (P_{Target}, also denoted „prior“) are the parameters of target VoIP application, the conditional probabilities are obtained from the system running the evaluation. Parameters used in the experiments are NIST „old“ parameters (see [1]) and are given in table 4.14. Note, that the values of actDCF and minDCF are normalized (see section 2.5 for explanation).

P_{Target}	C_{Miss}	C_{FA}
0.01	10	1

Table 4.14: Application parameters for the calibration

For running and evaluating the calibration, I have created scripts to train the calibration parameters, apply them to the score files and evaluate actDCF and minDCF values for the score files.

In table 4.15, counts and probabilities of misses and false-alarms are shown. Note, that the uncalibrated system yields over a million of wrong decisions (`amr.clean`) in million-and-a-half trials.

	Misses	$P_{Miss\|Tgt}$	False Alarms	$P_{FA\|Non}$	actDCF	minDCF
`amr.clean`	279	2.67%	1 184 072	77.59%	7.7081	0.6019
`g711.80`	290	2.78%	680 008	45.08%	4.4911	0.3835
`g729.90_burst5`	394	3.78%	869 723	56.99%	5.6799	0.4863

Table 4.15: Misses, False Alarms and DCFs for uncalibrated system

After the system has been properly calibrated, those numbers change greatly, see table 4.16. There are 50 times less wrong decisions in the calibrated system and the actDCF metric has decreased 10 times.

	Misses	$P_{Miss\|Tgt}$	False Alarms	$P_{FA\|Non}$	actDCF	shift
`amr.clean`	5 177	49.60%	16 417	1.08%	0.6025	-8.53
`g711.80`	3 232	30.97%	11 592	0.76%	0.3849	-6.49
`g729.90_burst5`	4 128	39.55%	14 427	0.95%	0.4891	-7.05

Table 4.16: Misses, False Alarms, actDCF and trained threshold shift for calibrated system

Chapter 5

Conclusion

In this work, I acquainted myself with the SRE systems architecture, their usage and the ways to modify them. I explored the VoIP technology and described codecs and features used in the VoIP. I have also studied and described the metrics used to evaluate SRE system's performance.

In order to simulate VoIP environment, I have created tools to generate VoIP audio data. These tools were subsequently used to prepare multiple data sets simulating various VoIP traffic to evaluate the SRE system on. I acquainted myself with the production SRE system developed by Phonexia and used it, along with the prepared data sets, to evaluate VoIP influence on the system's performance.

The experiments conducted has shown, that the performance of SRE system on the VoIP data depends greatly on the codec used to encode the data. The data from the codecs with lower bitrate (especially G.723.1 and AMR) result in worse system performance, than high bitrate (G.711) codec data. The fact, that some data gets lost during the network transmission does further decrease the performance, but the degradation of performance is much smaller than the degradation, when more compressing codec is used. The influence of packets getting lost in bursts decreases the performance of system for waveform codec G.711, vocoders (G.729, G.723.1 and AMR) perform slightly better.

The performance of the SRE system can be significantly improved by running system adaptation. The adaptation requires significant amount of labeled data, the size of it affects achieved performance improvement. The experiments show, that the data used for training the adaptation should be originated from the same codec and possibly even the same (or worse) network condition; however, systems adapted on large amount of data from similar codecs show great performance improvement, too. The necessity of running system calibration was showed.

I find the greatest benefit in the results of limited development sets adaptation and cross-adaptation, which can provide those interested with information about performance improvement, various types and sizes of labeled data may provide.

Further work on the topic may include training the whole SRE system directly on VoIP data, fusion of such systems or exploring the possibility to train the system directly on vocoder transmitted data instead of the audio. Another related field is the examination of signal distortions, possibly introduced when a packet was missing and two non-adjacent signal portions became adjacent.

Bibliography

[1] The NIST year 2010 speaker recognition evaluation plan [online]. http://www.itl.nist.gov/iad/mig/tests/sre/2010/NIST_SRE10_evalplan.r0.pdf, 2010. [cit. 2014-05-15].

[2] Ondřej Glembek. *Optimization of Gaussian Mixture Subspace Models and related scoring algorithms in speaker verification.* PhD thesis, 2012.

[3] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel. A study of interspeaker variability in speaker verification. *IEEE Transactions on Audio Speech and Language Processing*, 2008.

[4] Jan Silovský. *Generativní a diskriminativní klasifikátory v úlohách textově nezávislého rozpoznávání a diarizace mluvčích.* PhD thesis, 2011.

[5] Wikipedia. Mean opinion score — Wikipedia, the free encyclopedia [online]. http://en.wikipedia.org/w/index.php?title=Mean_opinion_score&oldid=599554337, 2014. [cit. 2014-05-12].

[6] Wikipedia. Nyquist–Shannon sampling theorem — Wikipedia, the free encyclopedia [online]. http://en.wikipedia.org/w/index.php?title=Nyquist%E2%80%93Shannon_sampling_theorem&oldid=605293132, 2014. [cit. 2014-05-11].

[7] Steve Young, Gunnar Evermann, Dan Kershaw, Gareth Moore, Julian Odell, Dave Ollason, Valtcho Valtchev, and Phil Woodland. *The HTK book*, volume 2. Entropic Cambridge Research Laboratory Cambridge, 1997.