# Acoustic model training for speech recognition over mobile networks

Juraj Vojtko, Juraj Kačur, Gregor Rozinaj and Ján Kőrösi

Institute of Telecommunications,
Faculty of Electrical Engineering and Information Technology,
Slovak University of Technology,
Ilkovičova 3, Bratislava 812 19, Slovakia
E-mail: juraj.vojtko@stuba.sk
E-mail: juraj.kacur@stuba.sk
E-mail: gregor.rozinaj@stuba.sk
E-mail: jan.korosi@stuba.sk

**Abstract:** The goal of this article is to provide and present information about the training procedure SpinxTrain and its eligible modifications to get accurate and robust speech recognition in a mobile GSM environment. Some modifications are based on effective preprocessing of input data in combination with the optimal setting of the number of states per model, through the adjustment of the number of tied states or number of Gaussian mixtures. Another source of increased recognition rate is the 'optimal' setting of the speech decoder. As it is a non–linear, mathematically not well tractable task containing both real and integer values, methods of evolution strategies can be successfully used (an 18.6% improvement in WER was observed compared to the original setting). All experiments and results were obtained for the Slovak speech database Mobildat, which contains recordings of 1100 speakers. The Sphinx4 recognition system was used for evaluation of the trained model.

**Keywords:** Sphinx 4; SphinxTrain; ATK; ASR; speech recognition; HMM; mobildat; evolution strategies.

**Reference** to this paper should be made as follows: Rozinaj, G. and Vojtko, J. (xxxx) 'Acoustic model training for speech recognition over mobile networks', *Int. J. Signal and Imaging Systems Engineering*, Vol. x, No. x, pp.xxx–xxx.

**Biographical notes:** Juraj Vojtko, born in 1981, received MSc in telecommunications from the Slovak University of Technology in Bratislava, Slovakia in 2005. Since 2009 he has occupied an assistant professor position at the Institute of Telecommunications at the Faculty of Electrical Engineering and Information Technology of the Slovak University of Technology in Bratislava. He has also worked as developer in commercial segment in the area of communication and information systems since 2002. The field of his research focused on speech processing specifically speech recognition and speaker identification and verification.

Juraj Kačur, born in Bratislava in 1976. Master of Science degree obtained in the year 2000 and PhD in 2005 at the Faculty of Electrical Engineering and Information Technology of the Slovak university of technology (FEI STU) Bratislava. Since 2001, he occupies an assistant professor position at the institute of telecommunication at FEI STU Bratislava. Between years 2000 and 2001, he was with the Slovak Academy of Science, department of speech analysis and synthesis where he participated on several projects. The field of his research activities includes: signal processing, speech processing, speech recognition, speech detection, speaker identification, High order statistic, Wavelet transform, machine learning, ANN and HMM.

Gregor Rozinaj (M'97) received MSc and PhD in telecommunications from Slovak University of Technology, Bratislava, Slovakia in 1981 and 1990, respectively. He has been a lecturer at the Institute of Telecommunications of the Slovak University of Technology since 1981. From 1992–1994, he worked on the research project devoted to speech recognition at Alcatel Research Center in Stuttgart, Germany. From 1994–1996, he was employed as a researcher at the University of Stuttgart, Germany working on a research project for automatic ship control. Since 1997, he has been a Head of the DSP group at the Institute of Telecommunications of the Slovak University of Technology, Bratislava. Since 1998, he has been an Associate Professor at the same institute. He is an author of 3 US and European patents on digital speech recognition and 1 Czechoslovak patent on fast algorithms for DSP.

Ján Kőrösi received MSc in telecommunications from Slovak University of Technology, Bratislava, Slovakia in 2007. During the studies, he worked on several projects with focus on speech recognition. He worked in telecommunications company as DSP/C++ programmer in 2006–2009.At present, he worked as C/C++ programmer for company which process and visualise meteorological data.

## 1 Introduction

Speech applications are becoming more and more popular and very useful in every-day situations. Some of the areas of interest are: speech communication with machines, voice controlled devices, speech data compression, speech analysis, transcription of natural conversation, medical application, etc. Our research is focused on recognition of continual speech. Nowadays speech recognition techniques used for this recognition are hidden Markov models, neural networks, the support vector machine, etc. These systems should be capable of operating on a real time bases, must be speaker-independent and must exhibit high accuracy (Shabtai, 2010). Practical systems (e.g., information retrieval systems in telecommunication networks (Juhár et al., 2007) or public intelligent terminal (Vrabec and Rozinaj, 2007) must support dictionary sizes of at least several thousands of words. These requirements can be currently met by statistical speech modelling using HMM of tied context dependent phonemes with multiple Gaussian mixtures (Kačur and Rozinaj, 2008).

It is known that the best results regarding speech recognition accuracy can be achieved if the training and application environment fit each other. Thus, to secure robust and accurate sets of models, huge speech databases must be collected covering various dialects, ages, conditions, etc. Therefore, the described ASR and the results are related to the speech database gathered over the GSM network MobilDat-SK (Darja et al., 2006). As it was compiled over mobile networks (GSM), this database shows higher degradations of speech quality.

Although the HMM concept has many advantages, there is no analytical solution to the estimation problem of HMM models. Thus, a great effort must be made in the training phase. Despite this, reaching a globally optimal solution is not ensured. Therefore, a mature system must be used, providing many auxiliary tools that can be iterative and selectively applied in various stages of the training. There are more developed systems to do so, but the most widespread ones are HTK (Young, 2005; Cambridge University, 2009) and SHINX (Sphinx Group, 2008; 2011). In the following, we will focus our attention on the open source SPHINX system rather than HTK, despite the fact the HTK system is more advanced and better documented.

The article is organised as follow: in Section 2, we describe the speech database; the system Sphinx and the SphinxTrain tool are listed in Section 3. Section 4 explains training by the SphinxTrain and Section 5 discusses its modifications for our tests. The optimisation of a speech decoder via evolution strategies is outlined in Section 6. Finally, concluding remarks are stated in Section 7.

## 2 Speech database MOBILDAT

The task of accurate recognition is more challenging in adverse environments and requires more steps and more sophisticated handling. Thus, we decided to use the Slovak Mobildat database (Darja et al., 2006), which was recorded over the GSM network and contains various types of cell phones. The concept of the Mobildat database is based on the widely used structure of the Speechdat database, whose many versions were built for several languages using fixed telephone lines (Höge et al., 1999). Further, some reference recognition systems like REFREC 0.96 or MASPER (Lindberg et al., 2000) were designed to cooperate with them. Thus, in the following, we recap some basic facts about the Speechdat database and its differences from the Mobildat.

The Slovak Speechdat database (Cernocky, 2001) consists of 1000 speakers divided into a training set (800) and a testing set (200). Each speaker produced several recordings in a session with durations between 4–8 minutes. There were 48 items spoken per speaker for the SPEECHDAT-SK. The MOBILDAT-SK database consists of 1100 speakers divided into training (880) and testing (220) sets; there are 50 spoken items per speaker. These items were marked and categorised into the following groups: isolated digit (I), digit/number strings (B,C), natural numbers (N), money amounts (M), yes/no questions (Q), dates (D), times (T), application keywords (A), word spitting phrase (E), directory names (O), spellings (L), phonetically rich words (W) and phonetically rich sentences (S, Z). Speech files were A-law coded into 8 bits with no header at 64 kbit/s. Description files were provided for each utterance with the orthographical transcription, but no time alignment was given. Beside the speech, the following nonspeech events were labelled, too: truncated recordings (~), mispronunciation (*), unintelligible speech (**), filed pauses (fil), speaker noise (spk), stationary noise (sta) and intermitted noise (int). In the case of the MOBILDAT-SK, GSM-specific distortion that could afflict the speech parts was also marked, as (%). In the accompanying dictionary, a phonetic transcription allowing multiple pronunciations was provided. Finally, there were 41,739 useable speech recordings in the training portion, containing 51 Slovak phonemes and 10,567 different context-dependent phonemes.

## 3 Sphinx4 and Spinxtrain

### *3.1 SPHINX*

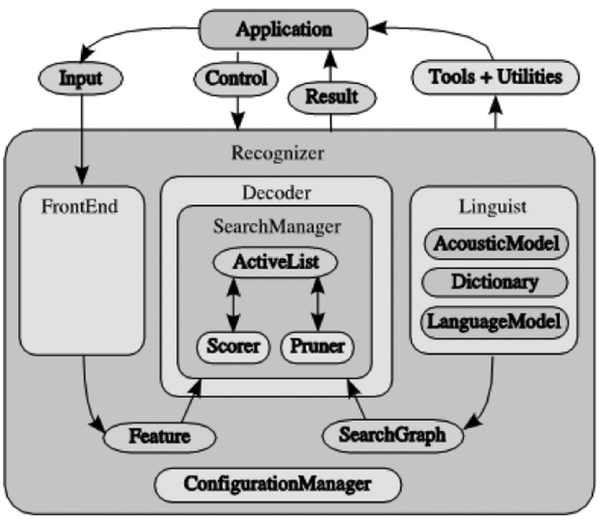
Sphinx4 is an open source speech recognition system written in the JAVA programming language. It was developed as an automatic system for human speech recognition by the Carnegie Mellon University, Sun Microsystems, Hewlett–Packard, Mitsubishi Electric, the University of California

and the Technological Institute from Massachusetts. The first version, programmed in the C language, was introduced in 1987; Sphinx4 is the latest version (Sphinx Group, 2008, 2011; Carnegie Mellon University, 2008). The Java programming language makes it platform independent. It was created to provide flexibility and modularity to perform a wide range of tasks related with the recognition of discrete or continuous speech. The Sphinx4 architecture has a modular design, so that the system programmer can change each module of the system to fit particular demands. The main blocks of Sphinx4 are: Frontend, Decoder and Knowledge base, which contain the lexicon (called dictionary), the acoustic models (HMMs) and the language model. Blocks like Frontend and Decoder are independently replaceable and all the necessary manipulations to do so are carried out by setting the desired configuration features in the configuration file. Next follows a brief description of the most important modules:

- FrontEnd – receives one or more input signals (from a file or microphone) and outputs sequences of attributes (MFCC).
- Linguist – uses the dictionary and structures from the AcousticModel. Then it performs translation and provides results in the form of a SearchGraph.
- Decoder – uses SearchGraph and attributes from both of the modules mentioned and provides results to the application which started the process and controls functionality of the modules.

The next picture on Figure 1 shows the architecture of the system:

**Figure 1** Architecture of the Sphinx4 (Sphinx documentation (Carnegie Mellon University, 2008))




## *3.2 SphinxTrain*

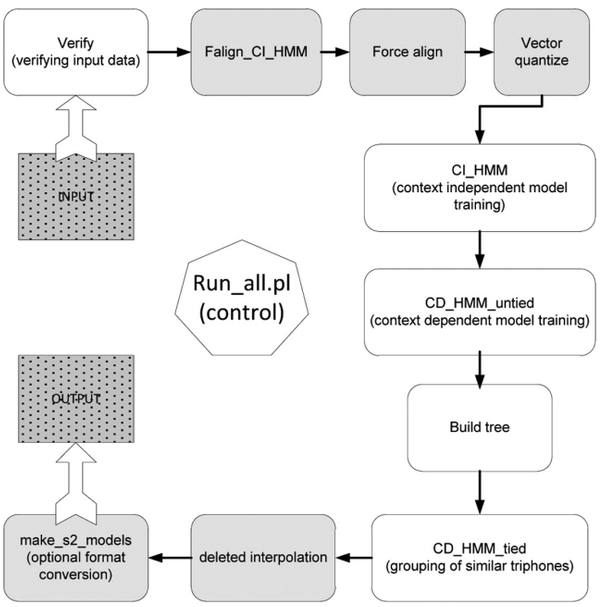
As the recognition is based on statistical speech modelling, each sound unit is modelled by a sequence of states that further model the probability distributions of the observed speech. The process of training is a very complex assignment that plays an important role. This task is performed by a different tool that is common to all Sphinx versions called the SphinxTrain (Carnegie Mellon University, 2000). It is an independent application written in C language, which computes acoustic models.

This package contains source codes of the SphinxTrain tools and control scripts written in Perl, which actually govern the running of the SphinxTrain during the training phase. Training of new models can be started by launching the runall.pl file from the scripts_pl folder. It is a Perl script which launches other scripts during the training process. Generally, the training process outlined by the SphinxTrain gradually undergoes these stages:

- verification of the input data
- statistical analysis of the input data
- vector quantisation (used only for discrete or semicontinuous models)
- training of context independent (CI) HMM models
- training of context dependent (CD) HMM phonemes – all triphones
- generation of language questions and decision trees building that are needed in the tying process of CD phonemes
- pruning of the decision trees and tying the similar states
- training of the CD phonemes' models with the reduced states number (tied triphones)
- training of the tied triphones models with the gradually augmented number of Gaussians.

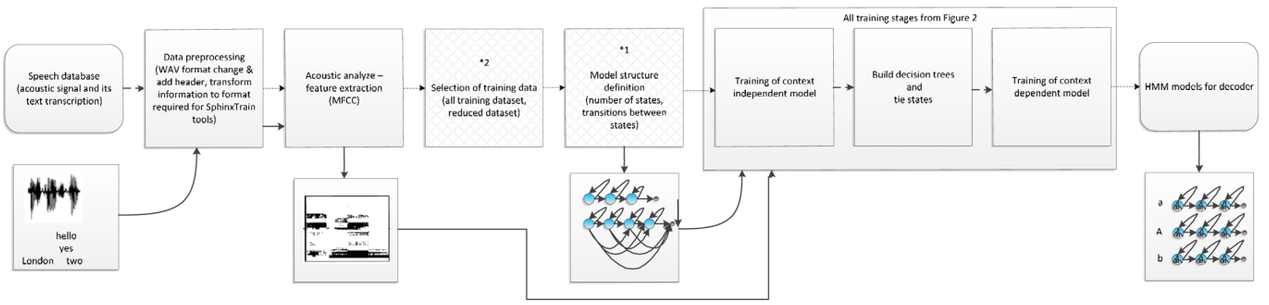
Figure 2a shows the SphinxTrain training process in general; only the stages in white boxes were used for our purpose.

**Figure 2a** Steps in the training process suported by SphinxTrain




In order that the training process would be successfully accomplished, several input data must be carefully prepared

**Figure 2b** A scheme showing steps in our training process with date flow




ahead and presented to the training script. The currently supported speech features are MFCC and PLP parameters and their time derivatives. Apart from the training data (extracted speech), several accompanying files and lists must be generated:

- list of all phonemes
- dictionary (lexicon)
- filler dictionary containing non-speech elements
- orthographical transcription of all training records
- list of all filenames existing in the training set
- SphinxTrain configuration file.

The output of the SphinxTrain consists of several files which together represent acoustic models for CI phonemes, untied CD phonemes with 1 Gaussian mixture and finally, tied CD phonemes with multiple Gaussian pdf per state. However, one of the main drawbacks of the Sphnix system is that during the training phase, all models must exhibit the same structure, regardless of whether they are speech units like phonemes or phrases or even non-speech events. Each model has one terminating, non-emitting state to allow multiple models to be concatenated into more complex models of words or even sentences.

## 4 Training of HMM models on the MOBILDAT-SK using SphinxTrain

The outlined training procedure for the Context Dependent (CD) and Context Independent (CI) models in the SphinxTrain has to be slightly adjusted for the structure of the Mobildat database and the Slovak language. Thus, in the following, a brief description of the training process is provided, together with some statistical information.

In Figure 2b, there is a scheme graphically presenting the phases of the statistical model training from the acoustic signal.

The training was performed on the Slovak Mobildat database to suit the requirements for the models to be successfully employed in GSM networks. The application of the Mobildat to the training procedure defined by SphinxTrain was not as straightforward as it is in the case of Masper or Refrec, which are designed directly for its structure, which is rather complex and distributed (5 CDs). Thus, several modifications had to be done prior to the training, which can be summarised as follows: conversion of SAMPA symbols, generation of the list of all utterances, selection of the training set, collection of transcription information, transformation and storage of data in the format required by the Sphinx tools, removal of records with unusable content, transformation of the Mobildat lexicon, generation of a list of phonemes together with the list of fillers (non-speech events) and finally, adaptation of the configuration file to suit the structure of Mobildat.

During the process, we have created CI models with 1, 2, 4, 8, 16, 32, 64 and 128 Gaussians and CD models with 1, 2, 4, 8, 16, 32, 64 Gaussians PDF, just to verify and compare their performance. There were 51 phonemes and 3 types of filler models determined to model non-speech acoustic artefacts. They are SIL for silence (including the silence at the beginning and the end of recordings), FIL for the filled pauses and SPK for the speaker's noise (Žgank and Maučec, 2010). Non-speech filler INT (intermitted noise) was ignored and mapped into the SIL model, just as was the STA (stationary noise). Records containing incomplete (truncated) or mispronounced words were rejected from the training process, unlike the records contaminated by GSM noise, which were preserved. Finally, the training set consisted of 41,739 records spoken by 880 persons. The dictionary stores the pronunciations of 14,909 words, some of which have more pronunciations. However, the SphinxTraining process can handle only one – the first; the remaining ones can be utilised only by the recogniser.

The initialisation of the models was performed over all training recordings by the so called flat start initialisation. Speech files were described by MFCC parameters and their derivatives, which were divided into three streams for every processed block of the length of 25 m. Thus there were 13 cepstral, 13 delta and 13 delta-delta coefficients, computed from 16 bit linear PCM waveforms. During the training, the Baum-Welch algorithm was used with the convergence ratio set to 0.04 and the maximum and minimum iteration numbers were set at 30 and 3, respectively.

## 5 Optimal settings and functional modification

The main modifications to the original training process are presented by 2 blocks marked by asterisks in the scheme shown in Figure 2b. Block 'Model structure definition' marked by *1 enables the setting of optimal parameters for the models and block 'Selection of training data', marked by

*2, represents the process of choosing the train set. Both the block and the modifications are described below.

### *5.1 Optimal settings of the training process*

Except the settings already mentioned, which were based on the 'common sense' approach, there are several more, which may have a crucial impact on the overall accuracy: the structure of the models, i.e., the number of states and the number of states after the tying process for the CD phonemes. These are language- and database–specific, so their number has to be determined by a series of experiments. In the case of the number of states per model, there are only 2 options regarding the Sphinx systems, 3 or 5 states by default. A modification of the training script permitted us to set other values too, so we tested values from 1–7 states per model. The number of states after tying is both language- and database-specific. The tested ranges were set from 2000–18,000, where the initial guess was based on previous experiments using the HTK system that determined the number of states using different - designer more flexible approach. For this purpose, we needed to set the following variables in SphinxTrain configuration file:

```
$CFG_STATESPERHMM = 5;
$CFG_INITIAL_NUM_DENSITIES_CD = 1;
$CFG_FINAL_NUM_DENSITIES_CD = 128;
$CFG_INITIAL_NUM_DENSITIES_CI = 1;
$CFG_FINAL_NUM_DENSITIES_CI = 64;
$CFG_N_TIED_STATES = 2000;
```

and remove restriction for the number of internal states, modify the scripts slave_convg.pl and norm_and_launchbw.pl and add the script split_gaussians.pl to the phase of context independent model training.

Tables 1–5 show the word recognition rate of application words grouped by number of states per model. Those results are graphically depicted for maximum accuracy per number of model states in Figure 3.

**Table 1** Word recognition rates for CD phonemes and 3 states per model for Application words

| | *Number of Gaussians* | | | | |
|---|---|---|---|---|---|
| *Number of tied states* | 8 | 16 | 32 | 64 | 128 |
| 2000 | 91,23 | 91,41 | 91,7 | 92,42 | 92,43 |
| 5000 | 92,47 | 92,36 | 92,93 | **93,73** | 93,08 |
| 9000 | 93,12 | 92,57 | 93,44 | 93,44 | 92,86 |
| 12,000 | 92,11 | 92,72 | 93,08 | 93,51 | |
| 15,000 | 92,98 | 92,57 | 92,86 | 92,5 | |
| 18,000 | 92,27 | 92,5 | 92,79 | 93,37 | |

**Table 2** Word recognition rates for CD phonemes and 4 states per model for Application words

| | *Number of Gaussians* | | | | |
|---|---|---|---|---|---|
| Number of tied states | 8 | 16 | 32 | 64 | 128 |
| 2000 | 94,03 | 95,2 | 95,66 | 95,53 | 95,53 |
| 5000 | 95,26 | **96,51** | 95,87 | 95,1 | 93,66 |
| 9000 | 94,91 | 96,29 | 96,02 | 94,88 | 92,1 |
| 12,000 | 95,05 | **96,51** | 95,88 | 95,45 | |
| 15,000 | 95,26 | 96,15 | 95,74 | 95,38 | |
| 18,000 | 95,12 | 95,8 | 96,09 | 95,31 | |

**Table 3** Word recognition rates for CD phonemes and 5 states per model for Application words

| | *Number of Gaussians* | | | | |
|---|---|---|---|---|---|
| *Number of tied states* | 8 | 16 | 32 | 64 | 128 |
| 2000 | 94,31 | 95,35 | 95,15 | 95,17 | 95,67 |
| 5000 | 94,53 | 95,79 | 95,95 | 95,89 | 94,4 |
| 9000 | 95,33 | 95,72 | 95,88 | 95,88 | 93,36 |
| 12,000 | 95,33 | 95,79 | 96,03 | 95,89 | |
| 15,000 | 95,33 | 96,15 | 96,17 | 95,88 | |
| 18,000 | 95,55 | **96,79** | 96,3 | 95,8 | |

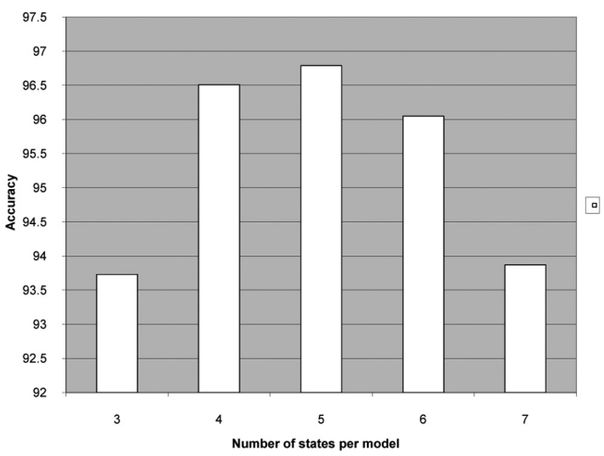
**Table 4** Word recognition rates for CD phonemes and 6 states per model for Application words

| | *Number of Gaussians* | | | | |
|---|---|---|---|---|---|
| *Number of tied states* | 8 | 16 | 32 | 64 | 128 |
| 2000 | 93,14 | 93,88 | 95,05 | 94,84 | 95,22 |
| 5000 | 94,08 | 95,04 | 95,41 | 94,97 | 91,35 |
| 9000 | 94,01 | 95,62 | 95,05 | 94,22 | 86,69 |
| 12,000 | 94,59 | 95,4 | 95,13 | 94,02 | |
| 15,000 | 94,23 | 95,76 | **96,05** | 93,77 | |
| 18,000 | 94,59 | 95,26 | 95,33 | 93,71 | |

**Table 5** Word recognition rates for CD phonemes and 7 states per model for Application words

| | *Number of Gaussians* | | | | |
|---|---|---|---|---|---|
| *Number of tied states* | 8 | 16 | 32 | 64 | 128 |
| 2000 | 90,63 | 91,26 | 91,59 | 92,32 | 90,98 |
| 5000 | 93,15 | 93,08 | 93,09 | 91,42 | 85,85 |
| 9000 | 93,76 | **93,87** | 93,44 | 91,19 | 86,11 |
| 12,000 | 93,67 | 93,27 | 92,91 | 89,34 | |
| 15,000 | 93,6 | 93,36 | 92,39 | 88,63 | |
| 18,000 | 93,73 | 93,49 | 92,23 | 89,7 | |

**Figure 3** Maximum accuracy for all tested number of internal emitting states in a model



### 5.2 Enhanced background model

As we mentioned above, Sphinx has the restriction that all models have to have the same structure ($x$ regular states plus 1 nonemitting state). Sphinx does not allow construction of special models like the T-model. The T-model is used in modelling short pauses between words or spelled syllables, especially when it is not known whether the pause took place or not, as these events are not marked in the description files. To do it automatically during the course of training, the T-models are appended to each word in the dictionary. This trick is not possible in the Sphinx which, by default, automatically assumes the existence of pauses at the beginnings and the ends of recordings. If there are longer pauses in speech, they should be marked. In recordings containing 'continual' speech 'mistakes' caused by long pauses are not relevant on average. But for spelled recordings, this assumption is by far not correct. To remedy this inaccuracy, three options are possible: the first one is to automatically insert a silence model <sil> between any spelled items.

**Original transcription:**

```
...
<s> <spk> jedna päť deväť jedna šesť nula
<s>(b30000c4)
...
<s> k ú <sil> t ypsilon </s>(b3000012)
<s> <spk> e i gé qé í i en a el </s>(b3000013)
...
```

**Modified transcription:**

```
...
<s> <spk> jedna <sil> päť <sil> deväť <sil>
jedna <sil> šesť <sil> nula </s>(b30000c4)
...
<s> k <sil> ú <sil> t <sil> ypsilon </s>(b3000012)
<s> <spk> e <sil> i <sil> gé <sil> qé <sil> í
<sil> i <sil> en <sil> a <sil> el </s> (b3000013)
...
```

the second one is just simply remove recordings with spelled items from the training completely.

**Original transcription:**

```
...
<s> i ako ivan v ako viera e ako emil t ako
tibor a ako alena </s>(b3000011)
<s> k ú <sil> t ypsilon </s>(b3000012)
<s> <spk> e i gé qé í i en a el </s> (b3000013)
<s> <fil> dvetisíc osemsto sedemdesiatjeden
slovenských korún </s>(b30000m1)
<s> jedno euro dvadsať centov </s>(b30000m2)
...
```

**Modified transcription:**

```
...
<s> i ako ivan v ako viera e ako emil t ako
tibor a ako alena </s>(b3000011)
<s> <fil> dvetisíc osemsto sedemdesiatjeden
slovenských korún </s>(b30000m1)
<s> jedno euro dvadsať centov </s>(b30000m2)
...
```

The third one uses the speech decoder to decide on the existence of a particular silence interval using 'inaccurately' trained models from the previous training process. It is more elegant to do it in this way, though it requires higher computational load. However, if the original models are not well trained, the results may not be better (as was observed during some experiments).

In the second approach, fewer models of phonemes would be affected by background noise, but fewer physical realisations of phonemes would be available for the training process. As this is a balance between accuracy and robustness, the correct scenario is a subject for experiments. Thus, four options were tested: the original Sphinxtrain procedure (applied also to spelled recordings), removal of spelled recordings from the training, indiscriminative insertions of SIL models between those spelled items and insertion of SIL models only where the silence intervals were detected by the recognition process (Viterbi decoding). Tables 6 and 7 and Figures 4 and 5 show the results of those four experiments, revealing improvements in the recognition process by all suggested modifications, i.e., by 'repairing mistakes' caused by the existence of longer pauses in spelled utterances and the nonexisting T-model.

**Table 6** Word recognition rates for CI phonemes and 3 states per model for application words

| | *Number of Gaussians* | | | | |
|---|---|---|---|---|---|
| *Modification type* | 8 | 16 | 32 | 64 | 128 |
| *Original procedure* | 76,43 | 79,35 | 79,71 | 81,65 | 82,68 |
| *Removed spelled* | 78,54 | 81,54 | 83,71 | 85,58 | 86,3 |
| *Insert SIL* | 83,48 | 85,49 | 87,87 | 89,46 | 89,75 |
| *Align* | 82,4 | 85,06 | 87,45 | 88,97 | 89,55 |

**Table 7** Word recognition rates for CD phonemes and 3 states per model and 9000 tied states for application words

| | *Number of Gaussians* | | | | |
|---|---|---|---|---|---|
| *Modification type* | 4 | 8 | 16 | 32 | 64 |
| *Original procedure* | 91,81 | 93,12 | 92,57 | 93,44 | 93,44 |
| *Removed spelled* | 93,62 | 93,49 | 92,93 | 93,29 | 93,66 |
| *Insert SIL* | 93,87 | 93,94 | 94,3 | 94,95 | 95,31 |
| *Align* | 94,38 | 94,59 | 94,74 | 95,6 | 94,88 |

**Figure 4** Word recognition rates for Context Independent phonemes and 3 states per model for Application words

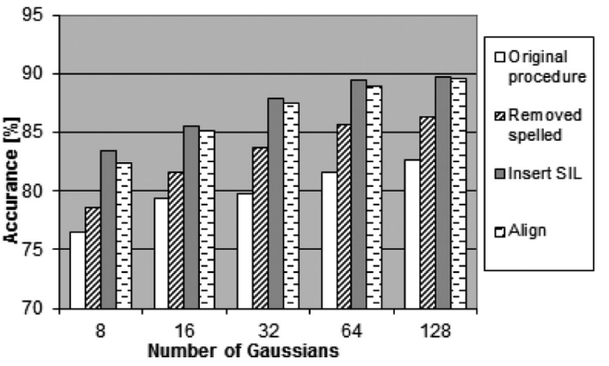
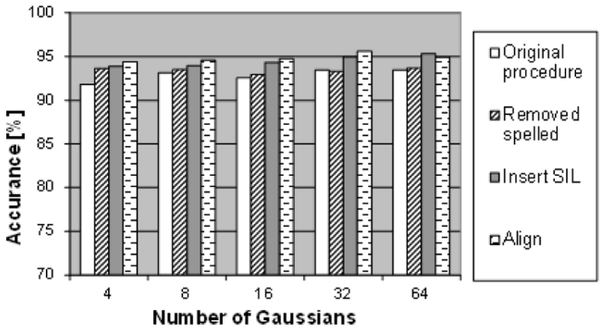


**Figure 5** Word recognition rates for Context dependent phonemes and 3 states per model for Application words




## 6 Optimisation of the speech decoder settings

Except having robust and accurate HMM models from the training phase, there are many variables to be optimally set for the real recognition process (Viterbi back tracking & speech detection). Generally, these parameters are non-linearly tied into an analytically unknown recognition error function that is to be minimised. All features have some physical boundaries with strong dependencies upon each other and can be either real numbers or integers. Thus, one has to solve a multivariable, constrained and non-linear optimisation problem. In mathematics, there are many effective methods and most of them require an analytical knowledge of the objective function. They are based on the gradient of the error function or its estimation, which in our case (unknown function, for which some dimensions are defined only on integer points) is either useless or rather ineffective. Less sophisticated methods that are, however, more suitable for the discussed optimisation problem can be hill climbing (Mitchell, 1997) or direct search algorithms. However, these are sensitive to the initial conditions (may stop at the local minima) and are not guaranteed to search through the whole space. Following this course of deliberation, we decided to apply the evolutional strategies (ES) – a branch of evolutional algorithms (EA), which were originally designed for this kind of optimisation problem (Dianati et al., 2002).

### *6.1 Evolutional strategies for finding global extremes*

Evolutional strategies belong to a group of stochastic non-linear optimisation methods which are based on the natural process of evolution (Mitchell, 1997). They can be applied to a variety of optimisation problems that are not well suited for standard optimisation algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly non-linear. All notions like mutation, recombination, selection, fitness of an individual, generations, parents and offspring are closely related to the process of natural selection.

There are more variations of the basic strategy (Dianati et al., 2002) denoted like: (λ, η) which means that from a population of λ individuals, η offspring will be selected into a new generation (no parent survive), whereas (λ + η) symbolises that η individuals for the new generation will be selected from the set of parents and their offspring, etc. Basic operations are defined as follows.

A mutation represents a process of modification of an individual that slightly differs from its 'original version', i.e., its features-optimised data are slightly changed by a stochastic process like in (1):

$$x' = x + N(0, \delta), \tag{1}$$

where $x'$ is a mutation of the original data vector $x$ that is subject to the optimisation process and $N(0, \delta)$ represents Gaussian distribution with zero mean and standard deviation δ.

Fitness is a measure of eligibility or capability of an individual, i.e., how capable it is of survival. This measure is task-specific and so, a method should be provided to classify each individual that is the subject of optimisation, based on its data.

A recombination is a process of combining several parents who will produce a new child. First, a proper pair (or group) of parents must be chosen, which can be done in several ways. This process can be completely random, or only the best parents according to their fitness are chosen (roulette game). After selecting pairs or groups of parents for each child, a method for their mutual combination must be decided. Again, there are more ways to do so. The most common are: intermediate – an offspring will be a weighted average of its parents, or discrete - some randomly chosen features will be copied from one parent and the remaining ones will be supplied by the other one.

So far, the subject of the evolution was only the optimised vector $x$. It was assumed that settings controlling the evolution such as standard deviation in the mutation phase, the method for selecting parents and the algorithm for passing attributes from parents to the offspring, were constant during all generations. However, these may evolve as well and presumably better respond to the structure and dynamics of the solved problem. Thus, in more sophisticated systems, they become optimised in the same way as the original features. An example of the mutation of the standard deviation is in (2):

$$\delta' = \delta e^{N(0,\delta_0)}, \tag{2}$$

where $\delta_0$ is a constant control measure for standard deviation that is unique for the whole process, $\delta$ is the original deviation and $\delta'$ is the newly mutated one. Next, each optimised feature may have its own dispersion, unless, of course, they are normalised. Then in place of only one value of $\delta$, a vector of standard deviations must be used. Furthermore, optimised elements may be correlated. Thus, instead of vectors of standard deviations, the covariance matrix should be used. For practical reasons rather than mutual correlations of elements, the vector of angles is used (rotation of the diagonal covariance matrix). Then the process of evolution from one generation of individuals to the next one can be formally expressed in general terms as follows (3):

$$P^{(t+1)} = sel_{\mu}^{\mu+\lambda}(\bigcup_{i=1}^{\lambda}\{mut(rec(P^{(t)}))\} \cup P^{(t)}), \tag{3}$$

where $P^{(t)}$ represents individuals in generation $t$ and $sel_{\mu}^{\mu+\lambda}$ determines the operation of selecting $\mu$ individuals for

the new generation from $\mu + \lambda$ individuals in the current generation. Each individual $A$ is then defined by the original data to be optimised $x$, vectors of dispersions $\sigma$ and angles $\alpha$.

The mutation of individuals is performed after the recombination phase in the following steps: mutation of angles, where the standard deviation is empirically about 5°, mutation of dispersions (diagonal covariance matrix) and finally, mutation of the original data vector $x$ based on the newly derived standard deviations and angles. Generally, each part of an individual $A$ can be recombined in a different way; thus, $\vec{\omega} = (\omega_x, \varpi_\sigma, \omega_\alpha)$ represents a vector of possible recombination methods for each part (data, dispersions and angles) and vector $\vec{\rho} = (\rho_x, \rho_\sigma, \rho_\alpha)$ determines the number of parents needed to produce a child. As the optimisation process proceeds, the actual solution is supposed to get to the vicinity of the global extreme and so, the mutations (dispersions) should be naturally decreasing in the time to produce a stable result. However, if the solving problem is hard to describe, but easily verified, it may be better to let the evolution algorithm itself control the standard deviations and methods of recombination, as was previously outlined for the general case.

### 6.2 *The off-line optimisation process of the speech decoding stage*

Once having trained HMM models, the optimisation process for the speech decoder was accomplished using the test portion of the MOBILDAT-SK database, which contains 220 speakers. As our main target is a dialog application, we evaluated records containing looped digits, denoted as B1, which do the best approximation of the pursued task (the unknown number of repetition of words from a dictionary). This ensures the proper setting of the speech detection algorithm that is also involved in a real speech recognition application, as well as several auxiliary grammar parameters. In the tests, we used context-dependent models of phonemes with three states and eight Gaussian mixtures. As the speech features the MFCC, delta and acceleration coefficients were used (39 elements). The models were trained using the train section of the MOBILDAT-SK database (880 speakers) and the MASPER training procedure. To present the real number of variables involved in a decoding task and to highlight their significance, the ATK system (Young, 2004) was used.

In the next stage, we had to choose some of the most significant parameters that we believed can play an important role for the recognition process employed in a real dialog application. It is vital to limit their number to speed up the convergence. It should be noted that playing all eligible records takes 30 minutes and so, assessing the fitness for a single individual in a given generation takes about 30 minutes. The selected parameters for the optimisation process mainly govern the speech detection algorithm the ATK system (Young, 2004; 2005) uses and adjust some grammar-related issues. In Table 8, these features are listed together with their description and 'optimally' set values.

**Table 8** Selected features for the accuracy optimisation of the speech decoding process realised in the ATK system

| *Feature* | *Value* | *Description* |
|---|---|---|
| SILSEQCOUNT | 109 | Number of frames classified as silence to detect the silence |
| SPCSEQCOUNT | 22 | Number of frames classified as speech to detect the beginning of speech |
| SPEECHTHRESH | 9.51 | Energy threshold for the speech detection. It is relative to the estimated silence level. |
| SILENERGY | 10 | Mean energy of silence (background noise) in dB |
| SPCGLCHCOUNT | 3 | Number of frames below the threshold that can be ignored when counting SPCSEQCOUNT |
| SILGLCHCOUNT | 10 | Number of frames above the threshold that can be ignored when counting SILSEQCOUNT |
| SILMARGIN | 17 | Shift the beginning and end of the detected speech given the number of frames. |
| CMNTCONST | 0.9984 | The time constant for cepstral mean adaptation |
| CMNMINFRAMES | 26 | Minimum number of frames after which a cepstral mean adaptation can take place |
| WORDPEN | –81 | Log probability penalisations for a word insertion error |

As all of these parameters have their physical interpretation and cover different ranges of acceptable values, they were transformed to the unified range <0,1> according to (4).

$$fe' = \frac{fe - lower\_limit}{upper\_limit - lower\_limit} \tag{4}$$

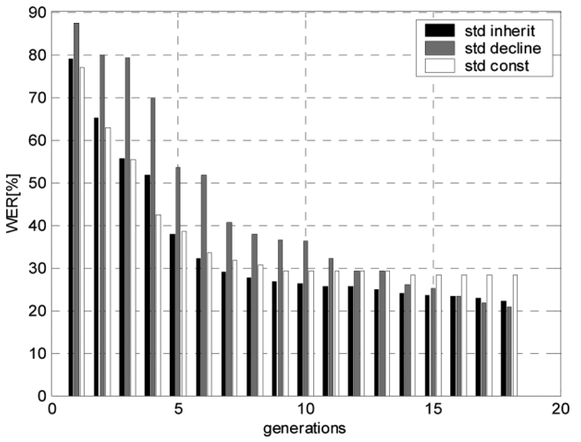
where $fe$ is the original value of the feature, $fe'$ is the transformed feature, *lower_limit* and *upper_limit* are the limit values of a feasible range.

Then, in the process of recombination or mutation, this range had to be checked. If for some features it was not met, the upper or lower limit was substituted, to ensure the physical meaning. Such normalisation enabled us to use only one standard deviation for the vector as a whole, because all features had approximately the same dispersion. Furthermore, we decided to automatically decrease dispersion with the increasing number of generations to reach a stable solution; however, tests with fixed and inherited (optimised) dispersion were executed as well. Because of the time consumption, we tried to limit the number of generations and the number of individuals in each generation. To do so, we neglected some possible correlations among optimised features.

As we were not completely sure of the "ideal" settings of the optimisation process, we tested more versions like: constant, decreasing and optimised deviation, various methods for parents' selection and recombination, number of generations,

etc. In Figure 6, the mean errors in each generation are depicted for the constant deviation (0.3), gradually declining deviation ranging from 0.35–0.02 and the optimised deviation by the evolution process, where in all cases the parents were selected stochastically according to their fitness.

**Figure 6** The mean word error rates per generations for gradually declining, inherited and constant standard deviation




## 7 Conclusion

We suppose there is some theoretical maximal accuracy which can be achieved by using the Mobildat-SK database (HMM models derived from the database) limited to its size, present triphones and conditions. We observed some ambiguity among the tested training parameters mentioned above (number of states per model, number of Gaussians mixtures and number of tied states) in the sense that a similar recognition accuracy can be obtained by different sets of parameters. The modification of one parameter causes a change in others, assuming accuracy should be kept unchanged. We found out that the best results are reached in the interval from 4–5 states per model, 16 and 32 Gaussians for this database. The number of tied states above 5000 has no significant improvement, but it also seems there is a reciprocal proportion in the number of Gaussians mixtures.

The second set of tests that evaluated and compared the original training procedure to its three suggested and implemented modifications proved that the original scenario is the least effective for the MobilDat-SK. From Tables 6 and 7, it can be seen that in most tested cases, the scenario with automatically detected silence intervals based on "incorrectly trained models" using Viterbi alignment for SIL insertion was the most beneficial. However it required almost two runs of the training process, so it is more time consuming. On the other hand, the indiscriminative insertion of SIL models between all spelled items helped to increase the accuracy as well and for some settings, it was even the best option; moreover, this was achieved without adding any extra computational load. Even if this is somehow impractical, the removal of the spelled recordings still gains better averaged results than the original procedure.

**Abbreviations (**Continued)

Although the model training phase is vital for a successful recognition system, the proper adjustment of the decoding process plays an important role as well. It is relevant not only to achieve the declared accuracy usually observed in the offline experiments, which provide better results, but also to speed up the decoding process and to reduce use of time and computational resources. As there are many parameters to be optimised together, which are mutually related either positively or negatively to each other, usually a common sense approach applied to the process of their manual adjustment may be far from the optimum.

By using the above-mentioned off-line method (evolutional strategies) we achieved 18.72% of WER (off-line tests using the standard evaluation procedure defined in MASPER for the looped digits test showed only 1.17% of WER, which shows the huge difference between off-line and on-line experiments), which is about 5% lower than those reached when adjusted manually (generally above 23% of WER). As presented, the utilisation of evolutional strategies brought promising results at low cost (we tested only 98 individuals through 10 dimensional spaces, which is less than 1.6 search points per dimension), in a way similar to what is reported in other areas.

From the tested settings of evolutional strategies (recombination methods, dispersion control), it can be concluded – see Figure 6 – that the gradual decrease of dispersion has brought better final results within the range of tested generations, which follows theoretic expectations. However, this approach showed slightly worse results in the early stages, which may be caused by the higher dispersion, which aimed to cover a wider area at the beginning. In the case of "unlimited" evolution time, the deviation should be kept optimised as well and some kind of life time control introduced, in order to avoid getting stuck in the local minima.

Features like WORDBEAM, MAXBEAM and GENBEAM that were involved in the pruning of the decoding process were not set during the optimisation process. The fitness of each individual was based solely on the recognition accuracy (WER). The above-mentioned parameters introduce some approximations to speed up the computation time, but could cause deterioration in accuracy. To hit the balance, a new fitness must be used utilising the computational load as well. This, however, would be highly dependent on the perplexity of the actual dialog system.

Usually, the incorporation of the Lamarckian evolutional strategy (Mitchell, 1997) can bring even better results, that is, some individuals are additionally trained during the evolution, to reach for a local minimum in their vicinity. Usually this is done by the hill climbing algorithm and such "learned" information is passed to the next generation (this opposes the Darwin theory). Unfortunately, in the case of costly evaluation of the fitness function, this option looses its benefits, as more individuals must be examined, which can be made up for by allowing more generations in the classical evolutional strategy optimisation. Finally, it should be emphasised that the optimisation of the decoder using ES was done offline and independently from the training process for HMM models and focused only on the achieved accuracy.

## Abbreviations

| *Abbreviations* | |
|---|---|
| ASR | Automatic Speech Recognition |
| ATK | API for HTK |
| CD | Context Dependent |
| CDs | Compact Disks |
| CI | Context Independent |
| FIL | FILled pause |
| GSM | Groupe Spécial Mobile (Global system for Mobile Communications) |
| HMM | Hidden Markov Model |
| HTK | Hidden Markov model ToolKit |
| INT | INTermited noise |
| MFCC | Mel-Frequency Cepstral Coefficients |
| ms | milliseconds |
| PCM | Pulse Code Modulation |
| PDF | Probability Density Function |
| PLP | Perceptual Linear Prediction |
| SAMPA | Speech Assessment Methods Phonetic Alphabet |
| SIL | SILence |
| SPK | SPeaKer noise |
| STA | STAtionary noise |
| WER | Word Error Rate |

## Acknowledgement

The authors hereby declare that the article was founded by the grant: FP7-ICT-2011-7-287848.

## References

Cambridge University (2009) *HTK Homepage*, Obtained through the internet: http://htk.eng.cam.ac.uk/

Carnegie Mellon University (2000) *Documentation to the Training Scripts*, Obtained through the internet: http://www.speech.cs.cmu.edu/sphinxman/scriptman1.html

Carnegie Mellon University (2008) *Sphinx-4: A Speech Recognizer Written Entirely in the JavaTM Programming Language*, Obtained through the internet: http://cmusphinx.sourceforge.net/sphinx4/

Cernocky, J. (2001) *Eastern European Speech Databases for Creation of Voice Driven Teleservices*, Obtained through the internet: http://www.fee.vutbr.cz/SPEECHDAT-E/

Darja, S., Rusko, M. and Trnka, M. (2006) 'MOBILDAT-SK - A mobile telephone extension to the SPEECHDAT-E SK telephone speech database in Slovak', *Proceedings of the 11th International Conference Speech and Computer (SPECOM'2006)*, St. Petersburg, pp.449–454.

Dianati, M., Song, I. and Treiber, M. (2002) *An Introduction to Genetic Algorithms and Evolution Strategies*, Technical Report, University of Waterloo, Ontario, Canada.

Höge, H., Draxler, C., Heuvel, H., Johansen, F.T., Sanders, E. and Tropf, H.S. (1999) 'SpeechDat multilingual speech databases for teleservices: across the finish line', *Proc. Europ. Conf. Speech Proc. And Techn. (EUROSPEECH)*. Budapest, Hungary, September 5–9, 1999.

Juhár, J., Čižmár, A., Rusko, M., Trnka, M., Rozinaj, G. and Jarina, R. (2007) 'Voice operated information system in Slovak', *In: Computing and Informatics*, Vol. 26, No. 6, ISSN 1335-9150.

Kačur, J. and Korosi, J. (2007) *An Accuracy Optimization of a Dialog ASR System Utilizing Evolutional Strategies*, ISPA 2007, Sept. 2007, Istanbul, Turkey.

Kačur, J. and Rozinaj, G. (2008) 'Practical issues of building robust HMM models using HTK and SPHINX systems', *In: Speech Recognition, Technologies and Applications*, In-Teh, ISBN 978-953-7619-29-9.

Lindberg, B., Johansen, F.T., Warakagoda, N., Lehtinen, G., Kačič, Z., Žgaňk, A., Elenius, K. and Salvi, G. (2000) 'A noise robust multilingual reference recognizer based on SpeechDat(II)', *In Proceedings of ICSLP 2000*, October 2000, Beijing, China.

Mitchell, T.M. (1997) *Machine Learning,* McGraw-Hill Higher Education, ISBN:0070428077.

Shabtai, N.R. (2010) *Advances in Speech Recognition,* Sciyo, Rijeka, ISBN 978-953-307-097-1.

Sphinx Group at Carnegie Mellon University (2008) *Sphinx-4 TWiki*, Obtained through the internet: http://www.speech.cs.cmu.edu/sphinx/twiki/bin/view/Sphinx4

Sphinx Group at Carnegie Mellon University (2011) *The CMU Sphinx Group Open Source Speech Recognition Engines*, Obtained through the internet: http://cmusphinx.sourceforge.net

Vrabec, J. and Rozinaj, G. (2007) 'IQ KIOSK - INTELLIGENT TERMINAL', *In: 49th International symposium ELMAR-2007 Focused on Mobile Multimedia*, 12–14 September 2007, Zadar, Croatia.

Young, S. (2004) *An Application Toolkit for HTK,v. 1.4.1,* Cambridge University. Obtained through the internet: http://mi.eng.cam.ac.uk/research/dialogue/ATK_Manual.pdf

Young, S. (2005) *The HTK Book (for HTK Version 3.3)*, Obtained through the internet: http://htk.eng.cam.ac.uk/docs/docs.shtml

Žgank, A. and Maučec, M.S. (2010) 'Modelling of filled pauses and onomatopoeias for spontaneous speech recognition', *In: Advances in Speech Recognition,* edited by Shabtai, N.R, Rijeka, Sciyo, ISBN 978-953-307-097-1.