PhpHMM Tool for Generating Speech Recogniser Source Codes Using Web Technologies

R. Krejčí

Abstract

This paper deals with the "phpHMM" software tool, which facilitates the development and optimisation of speech recognition algorithms. This tool is being developed in the Speech Processing Group at the Department of Circuit Theory, CTU in Prague, and it is used to generate the source code of a speech recogniser by means of the PHP scripting language and the MySQL database. The input of the system is a model of speech in a standard HTK format and a list of words to be recognised. The output consists of the source codes and data structures in C programming language, which are then compiled into an executable program. This tool is operated via a web interface.

Keywords: speech recognition, DSP, PHP, MySQL, OMAP, TMS320C674x, ARM.

1 Introduction

An automatic speech recogniser is a computer program consisting of interconnected algorithms whose input is human speech converted from a microphone into digital form, and the output is a text transcription of this speech. The structure of the speech recogniser consists of two main phases: in the first phase, so-called "training" is carried out, resulting in the creation and filling of data structures that describe a speech model. In the second phase of this process, decoding algorithms are developed that provide the speech recognition itself, using the speech models obtained in the training phase.

Since huge amounts of data are needed in order to create the speech recogniser, and huge amounts of data are elaborated, many activities are performed automatically using scripts. This facilitates the work and eliminates the need for repeated manual data processing. This is usually done using the HTK Toolkit [1], with the use of which a complete speech recogniser for the PC platform can be created.

However, when creating a speech recogniser to be run on various hardware platforms, e.g. digital signal processors, no such public tool is available, and thus proprietary software has to be programmed. In this case, the speech models trained using the HTK Toolkit can be utilised, but it is necessary to use totally different algorithms and optimisation methods for their treatment than those used on the PC platform. To test the optimisation methods, it is often necessary to change the data structures and convert their parameters. For this purpose, the Speech Processing Group at the Department of Circuit Theory, CTU in Prague has been developing a *"phpHMM"* tool that facilitates and integrates the development of speech recognition algorithms to alternative hardware platforms.

2 PhpHMM tool

The *PhpHMM* tool is a set of scripts in PHP scripting language [2] using the MySQL database server [3]. This technology has become one of the standards for generating web pages, but it is also useful for generating other texts, such as the source code in any programming language. The basis of the *phpHMM* tool is a class of functions that can be easily included into a superior system written in PHP language. The scripts are run on the server (either on a local computer configured as a server or on a publicly accessible web server), and their output is visible via a graphical user-friendly web interface. The source code of the speech recogniser can consist of a sequence of single steps. The steps will be discussed in the following text.

2.1 Speech model

The result of the training phase of the speech recogniser using HTK Toolkit is a text file in a defined format that describes a general model of speech, created on the basis of the utterances of a training database. The models of speech may have a huge number of different variations, e.g. the type of parametrisation (extraction of speech features), the number of HMM (Hidden Markov Model) states, streams and mixtures, the number of coefficients in each mixture, etc. During recognition, these parameters enter the output probability density function $b(o)$ [4]:

$$b_j(\bar{o}_t) = \prod_{s=1}^{S} \left[\sum_{m=1}^{M_s} c_{jsm} N\left(\bar{o}_{st}; \bar{\mu}_{jsm}, \Sigma_{jsm}\right) \right]^{\gamma_s};$$
$$N\left(\bar{o}_{st}; \bar{\mu}_{jsm}, \Sigma_{jsm}\right) = \frac{1}{\sqrt{(2\pi)^{n_s}\,|\Sigma|}} e^{-\frac{1}{2}(\bar{o}_{st}-\bar{\mu}_{jsm})^T \Sigma_{jsm}^{-1} (\bar{o}_{st}-\bar{\mu}_{jsm})}, \quad (1)$$

```
~h "a"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<MEAN> 39
1.437809e+00 -6.805577e+00 -8.517246e+00 -9.976683e+00 ...
<VARIANCE> 39
2.393653e+01 4.407170e+01 3.864353e+01 4.710320e+01 ...
<GCONST> 1.341746e+02
<STATE> 3
<MEAN> 39
2.916575e+00 -8.322930e+00 -1.077090e+01 -9.984103e+00 ...
<VARIANCE> 39
1.245955e+01 3.486024e+01 3.388573e+01 4.059823e+01 ...
<GCONST> 1.130805e+02
<STATE> 4
<MEAN> 39
4.856239e-01 -1.422903e+00 -6.716645e+00 -3.694754e+00 ...
<VARIANCE> 39
1.848022e+01 2.745304e+01 3.125877e+01 4.468990e+01 ...
<GCONST> 1.222291e+02
<TRANSP> 5
0.000000e+00 1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
0.000000e+00 6.224011e-01 3.775989e-01 0.000000e+00 0.000000e+00
0.000000e+00 0.000000e+00 7.666833e-01 2.333166e-01 0.000000e+00
0.000000e+00 0.000000e+00 0.000000e+00 5.902151e-01 4.097848e-01
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
<ENDHMM>
```

Fig. 1: Example of simple hidden Markov model of "a" phoneme in text form

where S is count of streams, γ_s is stream weight, M_s is count of mixtures in a stream, c_{jsm} is weight of the m-th mixture, $N(\bar{o}; \bar{\mu}, \Sigma)$ is multivariate Gaussian distribution with a vector of mean values $\bar{\mu}$ and a covariance matrix Σ. This function represents the acoustic similarity of the input signal with the reference models of speech units (phonemes).

All these factors enter into the *phpHMM* tool by uploading the text file with the speech model.

2.2 Parsing and storing into the database

After a text file with hidden Markov models is uploaded, it is parsed and converted from text form into data structures in the memory of the server. At the same time, some basic integrity checks of the file are carried out. Then database tables are created in the MySQL database and they are populated with relevant data from the uploaded file. It is convenient to use the server-based (MySQL) database, inter alia, because it enables easy selection of data by means of (even complicated) SQL queries. Selection and processing of data using a server-based database is significantly faster and more comfortable than searching in a text file. For our current experiments, it is advantageous to store the data in a "MEMORY" table type, as this storage allows faster access than the commonly used "MyISAM" type. There are also many techniques for optimizing the performance of the database, such as the use of keys and indexes [4].

2.3 Glossary of words

Our goal is to create a speech recogniser that will be able to handle continuous speech in real time, but currently we are dealing with recognition of individual words and short phrases. In this step, we can simply specify all the words which the recogniser will be able to recognise, either by typing in the text-box, or by uploading a text file. The more words are to be recognised, the greater will be the demands on the recogniser hardware, and hence on optimizing the algorithms.

2.4 Phonetic transcription

In all languages, there are the differences between the written language and the spoken form of speech. This step automatically creates a phonetic transcription of words entered in the previous step. E.g. the Czech word "zpěv" will be rewritten by the transcription "spjef".

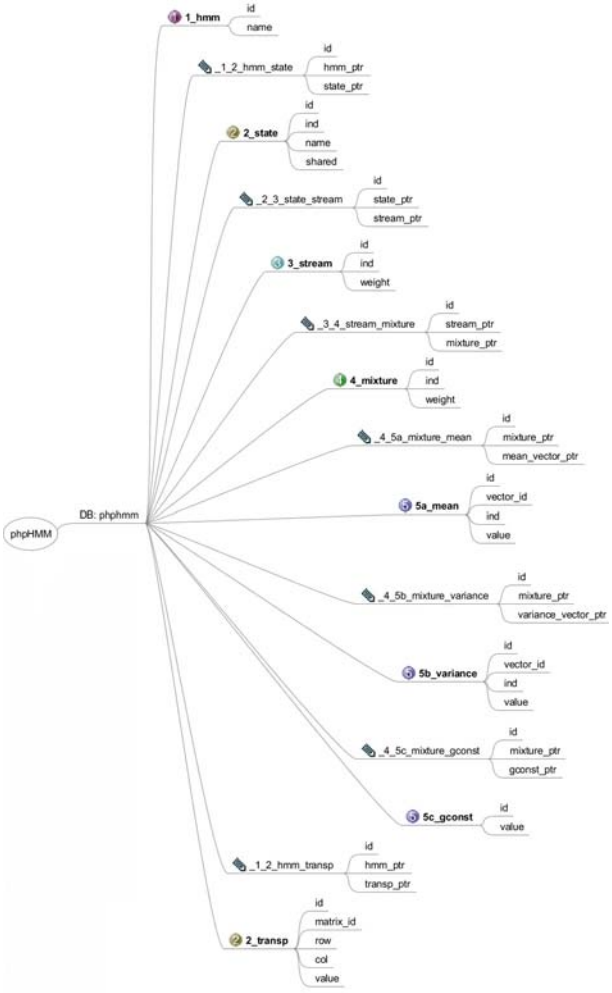

Fig. 2: Graphically expressed database structure for speech model

2.5 Selection of hardware platform

We work on optimizing algorithms of speech recognisers for platforms of multi-core digital signal processors of the TMS320C6000 family from Texas Instruments. The intention of *phpHMM* is to create a general tool for a large number of hardware and software platforms. Currently, this step offers a choice between a "general" platform and the "OMAP-L137" platform. OMAP-L137 is a dual-core heterogenous processor from Texas Instruments with both a 32-bit ARM9 and a TMS320C674x DSP core.

2.6 Selection of optimisation methods

If a speech recogniser is to be run on a system with limited hardware resources, it is necessary to optimize computationally intensive algorithms. In this step, a combination of optimisation methods can be chosen for testing. The optimisation is done at all levels of the design of the speech recogniser — from the layout of the data structures up to modifying the algorithms so that they are performed faster on the chosen hardware platform.

2.7 Creating word models

Depending on the optimisation method, models of the words are created as sequences of states with which the Viterbi algorithm [1] works. For each word, the phoneme models are chained into a sequence of states. E.g. the Czech word "spjef" creates the following sequence of states:

Fig. 3: Sequence of states of word „zpěv“ [spjef]

2.8 Assembling the source code and data structures

The main task of the *phpHMM* tool is to set up the source code and data structures on the basis of the input data, the specification of which has just been described. Depending on the type of parametrisation, the structure of the models and the required optimisations, the system generates the sources of the speech recogniser with the relevant data.

The source code must be generated before it can be programmed for each selection of the hardware platform and optimisation. The source code can be set up very effectively using PHP. The code of the PHP scripting language can be inserted directly into the source code in C. As described in [5], PHP can be used as a preprocessor with many more possibilities than the standard C preprocessor. For example, it can create cycles or compute with goniometric functions. A Hamming window lookup table can be generated as follows:

```
<?php $PI=3.14159; $N=512; ?>
const float hamming_ar[<?php echo $N;?>]={
<?php
for($n=0; $n<$N; $n++){
  $w=0.54 - 0.46 * cos(2*$PI*$n/$N);
  echo "$w,";
}
?>
};
```

The generated code is subsequently compiled by the appropriate compiler. However, this is already beyond the function of the current *phpHMM* tool, although in future it may be possible, after generating the source code, just to run the compiler and get the program in an executable format.

3 Results

Although the *phpHMM* tool is used to generate the entire speech recogniser, in the following text we discuss some examples of using the generated code for faster calculations.

3.1 MFCC optimisations

One of the optimisation methods calculates the results in advance, if all operands are known at compile time. This will avoid counting the same results repeatedly in the recognition process, and it speeds up the calculation.

This so-called "lookup table" method was used to generate the Hamming window coefficients, which are calculated at the beginning of the signal parametrisation by the mel-cepstral coefficients (MFCC) [1]. The parametrisation method during the recognition process never changes, and therefore the Hamming window coefficients do not change. The calculation then reduces to reading the coefficient in a one-dimensional data field.

A part of the parametrisation block of the signal, where speech attributes are extracted from the input signal, is the calculation of the Discrete Cosine Transform (DCT) [6]. Using the standard method for calculating DCT, which is calculated with goniometrical functions, a parametrisation calculation time of approximately 55 ms per segment was achieved at the tested digital signal processor. With the known number of input and output DCT coefficients, which are the constants known at compile-time and do not change during recognition, the concrete cosine results are calculated in advance and stored to the data structure. When running the DCT algorithm in real time, the cosine is (paradoxically) not calculated, but the pre-calculated cosine value is used according to the appropriate arguments. The calculation of the coefficient is thus reduced to reading its value from the pre-calculated table. By this optimisation, calculation time lower than 6 ms was achieved, i.e. approximately a ninefold acceleration.

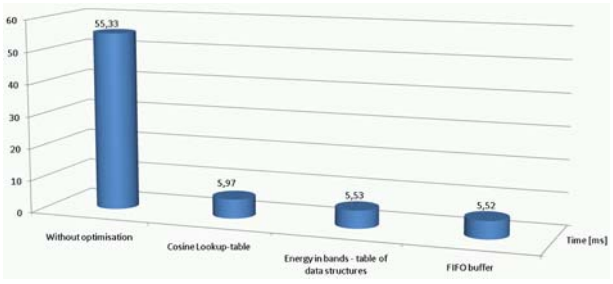

Fig. 4: Computation time vs. optimisation methods for MFCC parametrisation

3.2 Output probability density optimisations

Some of our proposed optimisation methods use transformed parameters, which arise by converting the original model parameters. E.g. a modified algorithm for calculating the output probability density function $b(o)$, based on the type of $A = A + B \times C$ dot product operation ("Multiply and Accumulate" – "MAC"), requires recalculation of the original coefficients by a simple transformation [3]. This transformation is performed while generating the source code, i.e. in compile time. The calculation without optimisations on the dual-core TMS320C74x DSP architecture lasted 1477 ms/segment. After applying appropriate optimisations by recomputing the data structures, the best time of 52 ms/segment was achieved when using the modified MAC algorithm [3].

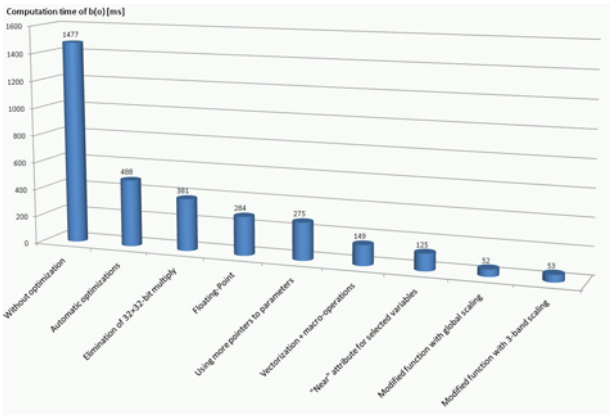

Fig. 5: Computation time vs. optimisation methods of $b(o)$ function

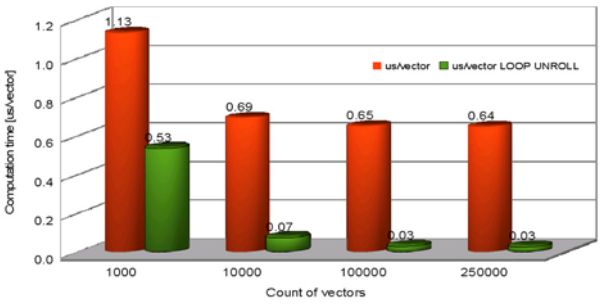

Fig. 6: Computation time of maximum of neighboring values

3.3 Viterbi algorithm optimisation

The Viterbi algorithm, which evaluates the most probable passage through the model, contains a part which compares adjacent values in the vector of the results of previous operations. Various methods have been tried, and the "Loop Unroll" method has proved to be the fastest in this case. The code that was originally performed repeatedly in the cycle is broken down into multiple particular operations without the cycle loop. This will not only reduce the overhead of cycle organisation, but will also provide an opportunity for greater use of the hardware architecture. In our case, instead of 32 passes through the cycle, a sequence of 32 individual operations with

directly addressed operands was created. This loop unrolling led to the possibility to use the "MAX2" instruction of the TMS320C6000 architecture, which is an SIMD (single instruction, multiple data) instruction that simultaneously compares two pairs of 16-bit operands and returns two results. The figure below shows the effectiveness of this optimisation for different numbers of test vectors compared with the best time achieved without using the loop unroll method.

4 Conclusion

The *phpHMM* software tool for developing speech recognition algorithms focuses on applications for Digital Signal Processors. The advantages of this tool include easy comparison of optimisation methods, easily changeable parameters, and a user-friendly graphical environment. It is used for generating source code and data structures tailored to the application.

Acknowledgement

This research was supported by grants GAČR 102/08/0707 "Speech Recognition under Real-World Conditions", GAČR 102/08/H008 "Analysis and modelling of biomedical and speech signals", and by research activity MSM 6840770014 "Perspective Informative and Communications Technicalities Research".

References

[1] Young, S., et al.: *The HTK Book*. Cambridge University Engineering Department, 2006. [online] http://htk.eng.cam.ac.uk/ftp/software/htkbook.pdf.zip.

[2] PHP [online]. 2011 [cit. 2011–03–12]. http:///www.php.net/.

[3] MySQL. The world's most popular open source database [online]. 2011 [cit. 2011–03–12]. http://www.mysql.com/.

[4] Krejčí, R.: Optimization of Computationally Intensive Part of Speech Recognizer. In *19th Czech-German Workshop on Speech Processing* [CD-ROM]. Praha : Institute of Photonics and Electronics AS CR, 2009, p. 22–26. ISBN 978-80-86269-18-4.

[5] Krejčí, R.: Use PHP preprocessor for generating source codes in C programming language. In *Králíky 2010*. Brno : Brno University of Technology, 2010, p. 84–87. ISBN 978-80-214-4139-2.

[6] Uhlíř, J., et al.: *Technologie hlasových komunikací*. Praha : Nakladatelství ČVUT, 2007. 276 p. ISBN 978-80-01-03888-8.

About the author

Robert Krejčí deals with digital signal processing and speech recognition focusing on optimisation of speech recogniser algorithms for systems with limited hardware resources.

Robert Krejčí
E-mail: robert.krejci@centrum.cz
Department of Circuit Theory
Czech Technical University
Technická 2, 166 27 Praha, Czech Republic