Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 51(65), Fascicola 2, 2006

VoiceStudio: A HMM-based Tool for Research and Teaching in the Speech Recognition Field

Alexandru Căruntu[1], Gavril Toderean[1], Alina Nica[1]

Abstract – This paper introduces the Romanian speech recognition system *VoiceStudio*. As most state-of-the-art Automatic Speech Recognition (ASR) systems today, it is based on Hidden Markov Models. Although there are numerous toolkits designed for this task, they usually have no visual interface, which means that the student or the researcher needs to spend some considerably amount of time in order to learn their functionality. The system's modular design, together with some implementation issues are pointed out, as well as the future plans of development.

Keywords: speech recognition, Hidden Markov Models, visual interface

I. INTRODUCTION

Being a researcher in the speech recognition field is not easy. Before having some revolutionary ideas that will amaze the whole scientific community you have to understand all those feature extraction algorithms, not to talk about the Hidden Markov Models. The student who learns about speech recognition is in the same position. Although the beauty of the things, that you discover in this time, will make you say, in the end, that the whole journey worth, an easier way to learn will definitely help a lot of people.

A good example in this sense is the HMM toolbox included in the BNT package developed in MATLAB [1]. Since it is not designed for speech related experiments there is no feature extraction algorithms included, nor language or acoustic modeling. Not to forget the considerable time that MATLAB needs to process a large set of data. But the HMM implementation, based on [2], is very useful in understanding both discrete and continuous Hidden Markov Models.

A truly state-of-the-art tool is the HTK [3], developed in C++. It has very good capabilities of feature extraction, as well as HMM training. Its major drawback is the fact that it has no visual interface, all the commands being entered from the command line, many of them having lots of options and parameters to set. Although is an open source tool, the learning curve in understanding the code is significant.

In our approach we tried to combine the strongest points of both toolkits, in an attempt to obtain a strong, flexible, easy to use tool for speech recognition. We used Visual C++ and an object oriented solution for the problem. The goal was to create a good speech recognition engine which runs behind an easy to use interface.

We will present in this paper the results that we have obtained so far, namely the feature extraction and HMM modules. The work is still in progress and soon a language modeling module for Romanian language will be added also. Rest of the paper is organized as follows: a brief outline of the implementation issues, followed by a few experimental results, and finally, conclusions and future plans.

II. ISSUES OF IMPLEMENTATION

This paragraph will give a brief review of the modules which compose our system (Figure 1). When needed, more details will be revealed about the difficulties that we have encountered and the way that we have solved them.

A. *Data preparation*

The first step in the development of any recognizer is data preparation, since speech data is needed both for training and testing. *VoiceStudio* currently supports two file formats, WAV and TIMIT. No recording capabilities are included at this stage of development, so the data must be pre-recorded with another tool.

Of course, before working with any data we need a good matrix library. In our case, we used one developed in our laboratories as a part of a speech analysis tool [4].

A dictionary containing all the words to be recognized is also needed. It must be created by hand by the user, but further implementations will generate it from the sample sentences present in the training data.

B. *Feature extraction*

A detailed explanation of our work in this field can be found in [4]. The feature extraction module described in that paper is included in *VoiceStudio*. The features that we have incorporated in our recognition tool are

[1] Technical University of Cluj-Napoca,
e-mail: Alexandru.Caruntu@com.utcluj.ro

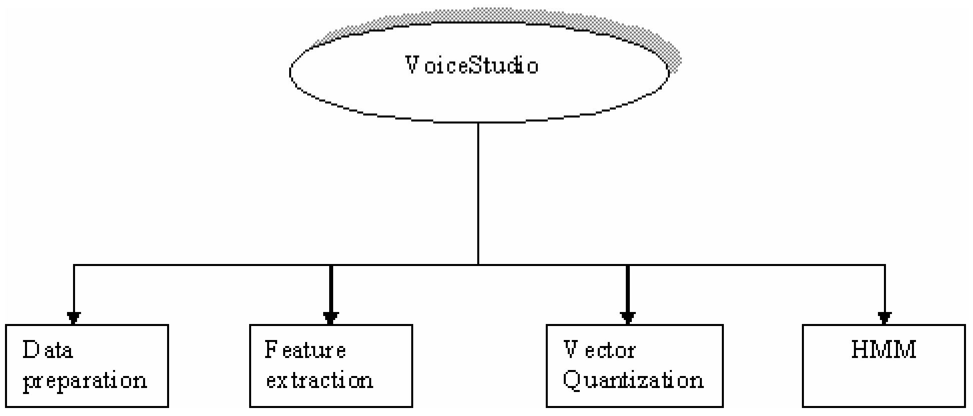


Fig. 1. The architecture of the *VoiceStudio* system

Linear Prediction Coefficients (LPC), Linear Prediction Cepstral Coefficients (LPCC), and Mel-Frequency Cepstrum Coefficients (MFCC).
A discussion is needed regarding the implementation of the last ones. As pointed out in [5], there are several ways of computing the MFCC coefficients:

- The solution proposed in 1980 by Davis and Mermelstein [6]: a filter bank of 20 filters, a sampling frequency of 10 kHz, and a speech bandwidth between 0 and 4600 Hz.
- The implementation from HTK, described in [7]: 24 filters, sampling rate greater than 16 kHz, [0, 8000] Hz speech bandwidth.
- The implementation from Malcom Slaney's Auditory Toolbox for MATLAB [8], which assumes 40 filters in the filter bank, a sampling rate of 16 kHz, and a speech bandwidth between 133 and 6854 Hz.

Several studies [9 – 11] have been done in order to compare the results of these implementations for speech recognition task. Based on them and our point of view regarding the best ratio between their results and the simplicity of implementation, we have chosen the last one for our system.
This implementation uses a filter bank of 40 equal area filters, which cover the frequency range from 133 to 6854 Hz in the following manner [5], [8]:

- The centre frequencies of the first 13 filters are linearly spaced in the range [200, 1000] Hz at 66.67 Hz one from the other.
- The centre frequencies of the last 27 filters are logarithmically spaced between 1071 and 6400 Hz, with a step of 1.0711703 Hz.

The filters in the filter bank are defined as:

$$H_i(k)=\begin{cases} 0, & k<f_{b_{i-1}} \\ \dfrac{2(k-f_{b_{i-1}})}{(f_{b_i}-f_{b_{i-1}})(f_{b_{i+1}}-f_{b_{i-1}})}, & f_{b_{i-1}}\le k\le f_{b_i} \\ \dfrac{2(f_{b_{i+1}}-k)}{(f_{b_{i+1}}-f_{b_i})(f_{b_{i+1}}-f_{b_{i-1}})}, & f_{b_i}\le k\le f_{b_{i+1}} \\ 0, & k>f_{b_{i+1}} \end{cases}, \quad (1)$$

where $i = 1,2,...,M$ stands for the i^{th} filter, f_{b_i} are $M + 2$ boundary points that specify the M filters, and $k = 1,2,...,N$ corresponds to the k^{th} coefficient of the N point FFT [5].
A detailed explanation of the whole procedure can be found in [12]. We will only point out here that the equalization of the area below the filters from equation (1) is due to the term

$$\frac{2}{f_{b_{i+1}}-f_{b_{i-1}}}. \quad (2)$$

Because of (2), the filter bank given by equation (1) is normalized in such a way that the sum of coefficients for every filter equals one.

C. *Vector Quantization*

Two algorithms for Vector Quantization (VQ) were implemented: *Linde – Buzo – Gray (LBG)* and *k-means*. VQ must be used in conjunction with discrete HMMs in order to represent by a single code a vector of features resulted through analysis. For this task any of the two algorithms can be used. Also, we used the latter to initialize some of the parameters of the continuous HMMs.
A detailed explanation of both of the algorithms is beyond the scope of this paper. We only point out that our implementation is based on [13] and [14].

However we encountered a major problem for both of the algorithms: empty clusters.
The principle of VQ technique is to find a set of vectors which describes best a set of data. These vectors are called *centroids*, and they form the *codebook* or the *dictionary*.
Vectors from the dataset are grouped into *clusters* based on their proximity to centroids. As a consequence of this process, empty clusters can result. We implemented a simple procedure in order to solve this problem:

Step 1. *First, we check to see if there are empty clusters.*

Step 2. *If yes, for the corresponding centroid, we try to find out which is the closest vector from dataset.*

Step 3. *We identify the cluster to which belongs the vector that we have found.*

Step 4. *If there are at least 2 vectors in this cluster we move the vector that we found in the empty cluster.*

Step 5. *If not, we go back to Step 2 (otherwise, if we move the vector, we will obtain another empty cluster).*

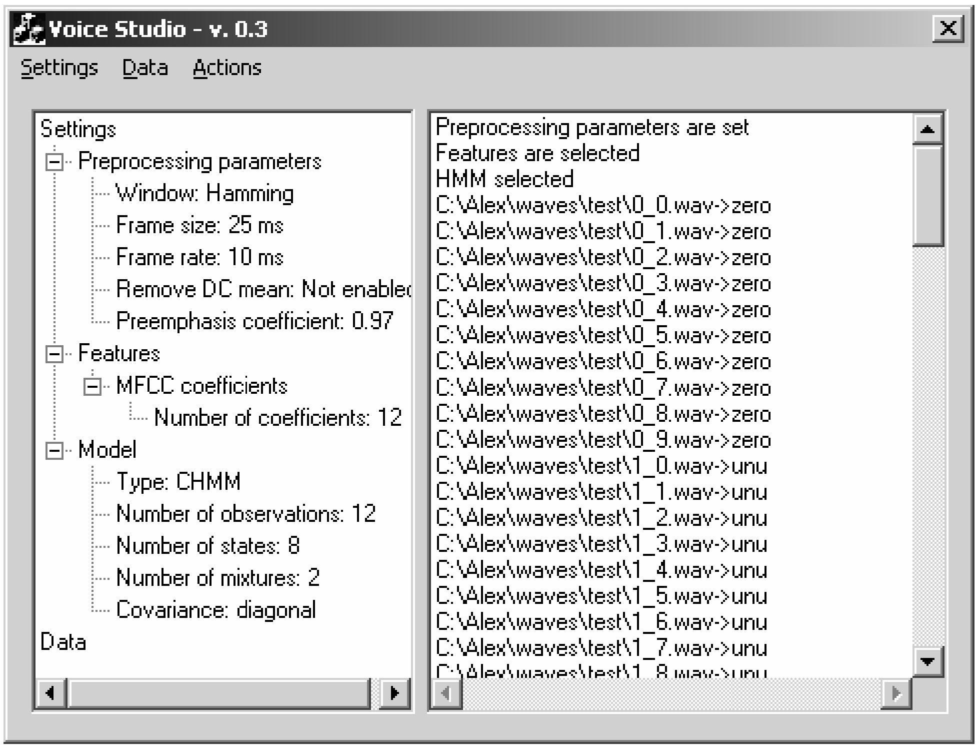
Future developments will take into account improvements of this algorithm in terms of computer efficiency.

D. *Hidden Markov Models*

As stated in [15], modern architectures for ASR systems are mostly software structures which generate a sequence of word hyphoteses from an acoustic signal. Most of the speech recognizers developed nowadays is based on Hidden Markov Models (HMM), because of their capability of best modeling the statistical nature of the speech signals. Since they outperformed all the other speech recognition techniques, we have also chosen them for our speech recognition engine.
A HMM is described by three parameters: the initial state distribution, the state transition probability distribution, and the observation symbol probability distribution (or output probabilities) in a certain state. Each of them is represented by a matrix, which must be estimated. A complete reference of the algorithms used for this task can be found in [2]. Here we will only point out the methods to solve the problems that occur during this process.
Currently *VoiceStudio* supports discrete *(DHMM)* and *continuous (CHMM)* HMMs. The difference between them is given by the nature of the output probabilities, which are distribution functions. If those functions are defined on a finite space, the models are discrete. In this case, the observations are vectors of symbols in a finite alphabet [15]. If distribution functions are defined as probability densities on a continuous observation space, the models are continuous. The most popular approach is to use mixture of Gaussians to characterize the model transitions, so we will need to estimate another three parameters in this case: the weights of the gaussians in the mixture, the means and the covariances.
While means are easily to initialize, using a k-means procedure, the problems arrise when we perform the same operation for covariances. In their case, we have to check if the matrix is singular, and if so, we have to adjust its values. This can be done by adding to the diagonal values a small quantity, until the determinant becomes different than zero.
VoiceStudio supports two kinds of covariance matrices: *full* and *diagonal*. Latter are preferred since they are defined with far less parameters, so we need less acoustic data to train them. Also, they are easier to estimate than full matrices. Ussually, setting a minimum value for their elements solves the singularity problem.
Finally, a problem which appears in any stage of designing a HMM is the small values of the probabilities. Performing multiplying operations over this data can lead to numbers which can not be represented by the computer. The solution is to scale the data with a scalling coefficient so that the values will not be close to zero.

III. EXPERIMENTAL RESULTS

An example of using *VoiceStudio* will be given in this section. As it can be seen in Figure 2, the user must take a few steps in order to perform a speech recognition experiment.
The *Settings* option from the menu is designated for setting up the parameters of the experiment. We divided these parameters into three categories. First of them (*Preprocessing parameters*) is formed by the type of window (Hamming or rectangular), frame size and rate expressed in milliseconds, an option to remove the DC mean from the signal, and an option to pre-emphasize the signal with a certain pre-emphasis coefficient. The second category is called *Features* and deals with the parameters used to represent the speech signal. Here, the user can choose between LPC, LPCC, and MFCC coefficients. Finally, the parameters of Hidden Markov Model are set, under the option *Model*: type of the model (DHMM or CHMM), number of observations, states, and mixtures, and the type of the covariance matrix (full or diagonal).
The *Data* option allows the selection of the location where the data is, and the parameterization of the data files. Also, when selecting the Vector Quantization option, a VQ algorithm (k-means or LBG) can be chosen by the user in order to attach the data to different clusters. This is a required step if we use a DHMM.
The last option, *Actions*, consists of two elements: *Training* and *Testing*. As suggested by their names, they deal with those two phases mandatory in the



Settings
- Preprocessing parameters
  - Window: Hamming
  - Frame size: 25 ms
  - Frame rate: 10 ms
  - Remove DC mean: Not enablec
  - Preemphasis coefficient: 0.97
- Features
  - MFCC coefficients
    - Number of coefficients: 12
- Model
  - Type: CHMM
  - Number of observations: 12
  - Number of states: 8
  - Number of mixtures: 2
  - Covariance: diagonal

Data

```
Preprocessing parameters are set
Features are selected
HMM selected
C:\Alex\waves\test\0_0.wav->zero
C:\Alex\waves\test\0_1.wav->zero
C:\Alex\waves\test\0_2.wav->zero
C:\Alex\waves\test\0_3.wav->zero
C:\Alex\waves\test\0_4.wav->zero
C:\Alex\waves\test\0_5.wav->zero
C:\Alex\waves\test\0_6.wav->zero
C:\Alex\waves\test\0_7.wav->zero
C:\Alex\waves\test\0_8.wav->zero
C:\Alex\waves\test\0_9.wav->zero
C:\Alex\waves\test\1_0.wav->unu
C:\Alex\waves\test\1_1.wav->unu
C:\Alex\waves\test\1_2.wav->unu
C:\Alex\waves\test\1_3.wav->unu
C:\Alex\waves\test\1_4.wav->unu
C:\Alex\waves\test\1_5.wav->unu
C:\Alex\waves\test\1_6.wav->unu
C:\Alex\waves\test\1_7.wav->unu
C:\Alex\waves\test\1_8.wav->unu
```

Fig. 2. The interface of *VoiceStudio*

design of any speech recognition system.

A number of test were performed in order to evaluate the system. Since our main interest was to build a robust HMM library, the task that we have chosen was isolated words recognition. We used for that a database formed by the digits from 0 to 9 in Romanian language. All three kinds of features mentioned before were used both with DHMMs and CHMMs. The results obtained range between 86% and 94% recognition rates, and can be considered satisfactory.

IV. CONCLUSIONS AND FUTURE PLANS

A speech recognition tool based on Hidden Markov Models was presented in this paper. Its major advantage compared with another existing products in this field is the visual interface, which helps a lot the process of research and learning. The problems that we have encountered during implementation, as well as their solutions, were emphasized. The experiments that we have performed showed that our HMM library is accurate and reliable.

Future directions of development will focus on language modeling for Romanian language and on increasing the visual capabilities of our software. Also, improving a number of algorithms that we have implemented, in terms of computer times, will also be taken into consideration.

REFERENCES

[1] "http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html"

[2] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Proc. IEEE, 77(2):257--286, 1989.

[3] "htk.eng.cam.ac.uk/"

[4] A. Căruntu, G. Toderean, and A. Nica, "Software Environment for Speech Processing", WSEAS Transactions on Communications, 4(8): 664 – 671, 2005.

[5] T. Gancev, N. Fakotakis, G. Kokkinakis, "Comparative Evaluation of Various MFCC Implementations on the Speaker Verification Task", Proc. of the 10th International Conference on Speech and Computer, SPECOM 2005, Vol. 1, pp. 191-194, 2005.

[6] S. B. Davis, P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences", IEEE Trans. on Acoustic, Speech and SignalProcessing, 28(4):357–366, 1980.

[7] S. J. Young, J. Odell, D. Ollason, V. Valtchev, P. Woodland, *The HTK Book. Version 2.1*, Department of Engineering, Cambridge University, UK, 1995.

[8] M. Slaney, "Auditory Toolbox. Version 2", Technical Report #1998-010, Interval Research Corporation, 1998.

[9] F. Zheng, G. Zhang, Z. Song, "Comparison of different implementations of MFCC", J. Computer Science & Technology, 16(6):582-589, Sept. 2001.

[10] B. J. Shannon., K. K. Paliwal, "A comparative study of filter bank spacing for speech recognition", Proc. of Microelectronic engineering research conference, Brisbane, Australia, Nov. 2003.

[11] M. D. Skowronski, J. G. Harris, "Exploiting independent filter bandwidth of human factor cepstral coefficients in automatic speech recognition", Journal of the Acoustical Society of America, 116(3):1774–1780, Sept. 2004.

[12] X. Huang, A. Acero, H. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, Prentice-Hall, 2001.

[13] S. Furui, *Digital Speech Processing, Synthesis and Recognition, Second Edition, Revised and Expanded*, Marcel Dekker, Inc., 2001.

[14] G. Toderean, A. Caruntu, *Metode de recunoastere a vorbirii*, Risoprint, 2005.

[15] R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, V. Zue, *Survey of the State of the Art in Human Language Technology*, Technical report, Center for Spoken Language Understanding CSLU, Carnegie Mellon University, Pittsburgh, PA, USA, 1996.