A Phrase Recognition System Intended for Musical Applications

Felipe Espic
University of Miami, Coral Gables, FL, 13343, USA
f.espic@umiami.edu

I. Introduction

Nowadays, musicians and audio technicians are demanding for new technologies that make easier and more efficient the control of electronic musical instruments, effects units, audio routers, mixers and so on. In this regard, the following project is intended to be the first step on a system that recognizes some commands executed by voice, either singing or speaking. This first attempt tries to partially fulfill this task. Specifically, it consists of a MATLAB script that performs the off-line recognition of six spoken commands (phrases or words). The phrases are: "play music", "stop music", "play guitar", "stop guitar", "faster", "slower".

The specific tasks to achieve this goal are: making a signal detection algorithm, a speech recognition stage, determining criteria to make the best decision in terms of phrase recognition avoiding false positives.

Speech recognition is a field that has been developed for several decades. However, there is not an "ideal" system that is capable to recognize the speech of any speaker immersed on any acoustical situation with a certain acceptable error rate. Anyway, this project is focused on recognizing some specific phrases not all of the possibilities of a natural language. In this respect, our problem is delimited.

According to the literature, there are many models to address this task, such as described in [1], [2] and [3]. However, this work is essentially based on the methods presented in the tutorial on Hidden Markov Models (HMMs) for Speech Recognition by Lawrence Rabiner [5], which describes different situations and the techniques to address them.

II. Proposed Method

Basically, the scheme of the proposed algorithm is illustrated in Fig. 1

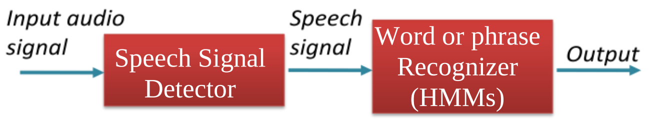

Fig. 1 Scheme of the proposed speech recognition system.

Another simpler scheme was analyzed previously, which consisted only in the phrase recognizer. However, this was dismissed, since the recognizer would have to run the HMMs in every possible combination of starting and ending points. Obviously, it would be really inefficient. Then, the solution consisted on a detector of *useful* signals. The label *useful* is used to define signals that meet certain constraints and therefore being likely one of the spoken phrases listed above. A deeper description of these constraints is presented in section *A.*

The second stage, the recognizer, can take the few chunks of audio previously delimited to run the HMMs and thus choosing the model that has the highest probability.

A. Speech Signal Detector

The specific objective of this stage is to trim and select the chunks of audio that probably correspond to any of the query phrases from the previously recorded audio file. The method mainly consists of three steps (See Fig. 2).

The first stage is a band pass filter whose cutoff frequencies are 200Hz and 5kHz. It is intended to keep the frequency range of the voice rejecting any other frequencies that could come from other undesired sources (noise).

The second step is the computation of the signal envelope. Firstly, the signal is full-wave rectified. Then, it is low pass filtered by convolution with the second half of a Hanning window [7]. The length of the filter is around 30ms in order to simulate the behavior of the human auditory system [8].

The third step is to cut the audio file in few chunks according to some constraints (see Fig. 3):

– The amplitude of every selected chunk has to be higher than a certain threshold. This threshold depends on the root mean square (RMS) of the amplitude of the whole audio file. It ensures that the signal selected is above the background noise floor.

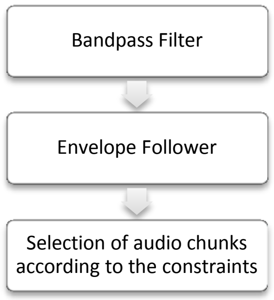

Fig. 2 Signal detector process. The signal is band-passed. Then, its envelope is calculated and according to some constraints, the audio file is trimmed into few audio chunks.

– The dismissed space between two chunks has to last longer than a certain threshold. If it is not achieved, both chunks are joined into only one including the space that previously had been dismissed. It ensures that although some phonemes or the pause between words that belong to the same phrase did not pass the RMS, they pass this step as a unique audio chunk.

– If the chunk is shorter than a defined minimum length, it is dismissed. This threshold is determined by the duration of the phrases that belong to the training set.

– If the chunk is longer than a certain threshold, it is dismissed. This threshold is determined in the same way as above.

With a correct selection of the thresholds, we can ensure that:

– The number of possibilities in which the HMMs can run is dramatically decreased.

– None of the correct audio chunks is dismissed.

B. Building the HMMs

The second stage of the system is the phrase recognizer made up by several HMMs. In order to determine these models, we need to define the feature extraction procedure, number of models, characteristics and number of states, initial prior probabilities, initial transmission probabilities and initial emission probabilities.

Feature Extraction

Firstly, as the same as the signal detector, a band-pass filter is applied to keep the frequency range of the spoken voice, whose pass band range is 200Hz - 5kHz.

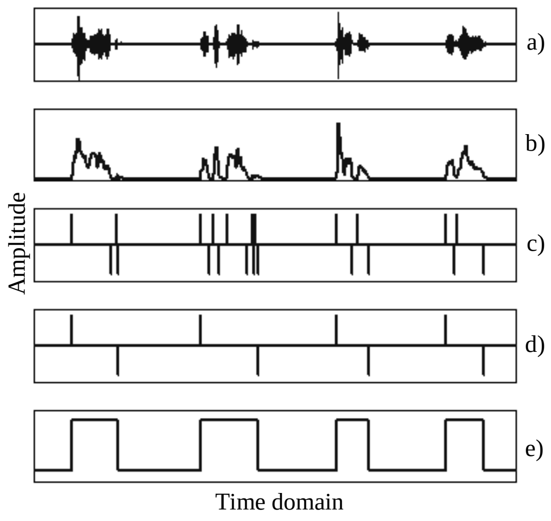

Fig. 3 Example of the two last steps in the Signal Detector algorithm. a) Audio file already-band pass filtered. b) Amplitude envelope. c) Cuts made according to the RMS-related threshold. The upward edges indicate the starting points of the selected chunks. The downward ones indicate the end points. d) The cuts after applying several constraints about the duration. e) Output binary function whose highest values show where the signal is allowed to pass. In this example, finally there are four selected audio chunks.

According to the literature [5] [6], Mel Frequency Cepstral Coefficients (MFCC) are robust features that are widely used for speech recognition applications as well as their first and second derivatives. In this regard, initially, the first nine MFCC plus its first derivative and its second derivative are used as feature vector. Thus, the dimension of this is 24 elements.

$$F = \{mfcc_1, mfcc_2, \dots, mfcc_n, mfcc'_1, mfcc'_2, \dots, mfcc'_{n-1}, mfcc''_1, mfcc''_2, \dots, mfcc''_{n-2}\} \quad (1)$$

Where *n* is the number of MFCC coefficients. The apostrophe means derivative. Since we are using a discrete signal, the correct name for this derivative is *finite difference vector.* The length of this is one less *n,* and the second finite difference vector is one less more. Finally, our feature vector is made up 9 MFCCs, 8 finite differences of MFCC and 7 finite differences of MFCC initially.

The system is frame based, so the feature extraction procedure is performed frame by frame. The duration of each frame is about 20ms and they are taken every 10ms (50% overlap) [6].

Acoustic Model

In order to use HMMs, we have to determine the acoustic units that will be represented by the HMMs and its states. There are many approaches to define them depending on the application. For example, in natural language speech recognition, each HMM can model a phoneme in context or *triphoneme* [6]. In our case, since the phrases to recognize are just a few, each HMM represents one phrase. Besides, each state represents a phoneme as illustrated in Fig. 4. Hence, initially there are 17 states as a total.

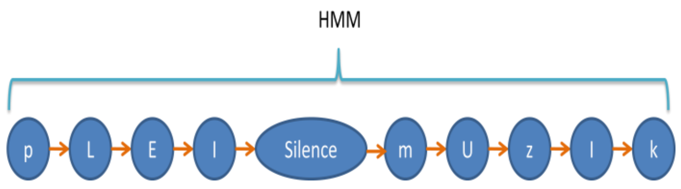

Fig. 4 HMM that represents the phrase "play music". The labels of the states represent the phonemes present in the phrase.

Finally, there are six models with their specific initial prior, initial transmission and initial emission probabilities.

Initial Prior and Transmission Probabilities

The first attempt of defining them was to try to enforce them to match the more obvious paths, although the literature [5] says the contrary. For instance, in the example showed in the Fig. 4, the prior probability of the phoneme *p* may be one, while all of the other ones zero. Besides, the transmission probabilities just allow the connection of two successive states. This approach was dismissed because of several reasons:

– Actually, the phrases do not follow the most obvious or determined paths in several cases.

– Fixing some initial probabilities to zero does not allow the training process to modify them.

– Testing showed that this approach does not carry out acceptable results.

Then, as suggested by [5], uniform initial estimates were applied:

$$P(\omega_i) = {}^{1}/_{c}$$

$$a_{ij} = P\big(\omega_j(t+1)\big|\omega_i\big) = {}^{1}/_{c} \qquad (2)$$

Where c is the number of states, t is the index for the feature vectors in time domain. Thus, $P(\omega_i)$ is the prior probability of the state ω_i and a_{ij} is the transmission probability from the state ω_i to the state ω_j.

Several audio samples were trimmed, extracted and grouped by hand into the 17 phoneme groups' data base. Later, we will see that other automated clustering methods were attempted.

Initial Emission Probabilities

So far, the initial estimates of the HMM parameters have been straightforward. Nevertheless, the initial emission probabilities are not at all. The problems to fulfill this task are as follows:

– Find a method to estimate the multi dimensional and continuous emission probability densities without having any previous knowledge. It is necessary, since the observations are continuous real valued observations (MFCCs).

– Find an algorithm for the re-estimation of the parameters from the HMMs during training process.

– The number of parameters of the selected method could be too high, making the implementation unfeasible. This is the called 'curse of dimensionality'. In the worst case, a data reduction process, like principal component analysis (PCA) may be applied to the feature vectors in order to decrease the dimensionality of them, and as a consequence cutting down the number of parameters of the chosen method.

The first attempt to address this task was by the K-Nearest Neighbor Algorithm (K-NN), which allows estimating multi-dimensional probability densities. However, training involves re-estimation, which could be achieved in this case by discretizing the pdfs and using discrete observation densities techniques for re-estimation. Although it looks very appealing, the discretization involves round-off error, and we can get a too high number of combinations just using a small number of bins per dimension, since the number of dimensions can make the number of bins grows exponentially. For instance, if the dimension of the feature vector is 12 and the number of bins per dimension is 50, the number of all of the different possible quantized vectors are $2.44140625 \times 10^{20}$, which makes this approach not practical.

Gaussian Mixture Models (GMM) are widely used to address this problem, since they are described only by a few parameters and there is a method for its re-estimation from Continuous Observation HMMs. The equation (3) shows the definition of the emission probability density $b_{(\boldsymbol{X})}$ of the *j*th state, where ***x*** is the feature vector to be evaluated. *M* is a predefined number of Gaussians, *N* is a normal distribution (Gaussian), $\boldsymbol{\mu}$ the mean vector and $\sum$ the covariance matrix of the *m*'th Gaussian.

$$b_{j(\boldsymbol{X})} = \sum_{m=1}^{M} \omega_{jm}\, N(\boldsymbol{\mu_m}, \boldsymbol{\Sigma_m}) \qquad (3)$$

Hence, in order to obtain a good initialization, it is necessary to estimate the centers and the covariance matrices of the Gaussians. For this task, the K-Means Algorithm is used. Although there is a Maximum Likelihood (ML) parameter estimation procedure for GMM, sometimes it is not suitable due to non singularities or non analytically solvable equations that appear. The K-Means Algorithm is capable of partitioning the data in clusters and finding their means just doing a few iterations. Then, the means corresponds to the centers of the Gaussians and the covariance matrices are computed as the covariance of the sample vectors. Finally, the weights are calculated as the proportion of samples belonging to a specified cluster meeting the expression (4) in order to ensure stochastic behavior.

$$\sum_{m=1}^{M} \omega_{jm} = 1 \qquad (4)$$

C. *Training*

The prior and transmission probabilities are trained using the standard methods for HMMs explained in [12] using a data base of 900 utterance audio samples.

For the emission probability densities, as mentioned, Maximum Likelihood Estimation does not behave well in certain cases. Also, for the training process it is necessary a procedure that does not consider the data used during building the HMMs, but the model built from that. For those reasons, Expectation Maximization (EM) Algorithm is used. This is capable of estimating the parameters of a new model from an old model by considering new data (training data). It essentially involves two steps. The E-step, which estimates the missing data that is used to build the transmission probability functions. Let us assume that we have a set of unknown data *y* and the observed data *x*. Thus, the whole data set is named *z=(x+y)*. The equation (5) shows that by Bayes rule, it is possible to express $P(z|\theta)$ in terms of the conditional probability $P(y|x,\theta)$, where θ denotes de parameter vector. Then, the expectation of the likelihood is computed (Equation (6)).

$$P(z|\theta) = P(x,y|\theta) = P(y|x,\theta)P(x|\theta) \qquad (5)$$

$$E_{(y|x,\theta)}\left[log\left(P(x,y|\theta)\right)\right] \qquad (6)$$

Then, in the M-step, the likelihood function is maximized using:

$$\theta^{i} = argmax_{\theta}\left\{E_{(y|x,\theta)}\left[log\left(P(x,y|\theta)\right)\right]\right\} \qquad (7)$$

Where *i* is the iteration index. This iteration procedure is performed until a specified number of iteration is reached or when the difference of the results of successive iterations is negligible.

D. *Recognition*

The main scheme is showed in Fig. 5.

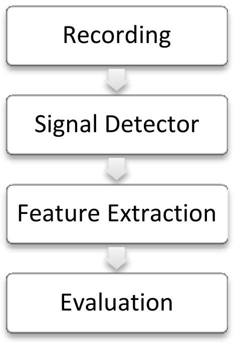

Fig. 5 Main scheme of the recognition system

The Recording process is performed by any standard recording software operating at 16 bits, 44.1kHz. The audio file is saved and a MATLAB script is executed to apply the following steps. The Signal Detector and The Feature Extraction processes were described before. Then, only the Evaluation process will be explained in this section.

Evaluation

For this task, we need to compute the probability of every model λ_i given the observation V^Tset to be recognized. Then, the model that yields the highest probability is assigned to the data.

$$p(\lambda_j|V^T) = \frac{p(V^T|\lambda_j)p(\lambda_j)}{p(V^T)} \tag{8}$$

Where j is the model index and T the length of the observation vector. As we want to compute this probability in order to compare to each other, it is not necessary to compute $p(V^T)$, since it is constant for every model. Besides, the term $p(\lambda_j)$ can be fixed to a constant value as suggested in [12]. Then:

$$p(\lambda_j|V^T) \approx p(V^T|\lambda_j) \tag{9}$$

In order to compute this value, the Log-Likelihood of the testing data is evaluated for each model using the *Forward Algorithm* [12].

On the other hand, if for some observation V^T, there is not any model whose Log-Likelihood is higher than a specified threshold, the sample is dismissed and unclassified. The meaning of this is that the system decides that the observation does not belong to any of the classes (models).

III. Improvements

After some informal tests, it was noticed that the main problem of the recognition system was the false positives. It means that despite applying the Log-Likelihood threshold for the non classification, there are still some samples that are classified as any of the six phrases. Then, some strategies were applied:

– Compute the variance of the Log-Likelihoods for every model. If the variance is below a certain threshold, it means that its assignment to a class is not enough clear. Then, the sample is dismissed.

– Implement an automated clustering algorithm to extract feature vectors for each state in order to build the Emission Probability Densities. So far, this process was made by hand as mentioned in section II.B. The two approaches were k-means and hierarchical clustering. Although, they looked very promising, the performance of the system got worse.

– Use other feature extraction techniques. Relative amplitude RMS values were added as features per state and normalization of feature vectors was implemented. However, none of these attempts were successful.

– Create a new *"Noise"* model which represents an "average" of any utterance that does not belong to any of the classes. Then, if the highest Log-Likelihood of a certain observation corresponds to the *"Noise"* model, the sample is unclassified. Therefore, there are 7 models since now.

The first three strategies did not work as expected, while the last one carried out an important improvement. Therefore, the latter was maintained for the final version.

IV. Testing

The testing data set consists of pre-trimmed (using *Signal Detector*) and pre-labeled audio files stored in such way to make easy any change in the testing data set. It consists of close to 300 utterance audio samples. The same number of samples per model was used. Several number of states and number of features were tested in order to choose the best combination as showed in TABLE I.

Then, the best performance was achieved by using 13 states and 24 features yielding a percentage of accuracy of 83%.

TABLE I. PERCENTAGE OF ACCURACY

Feature Vector Dimension

Number of States	**22**	**23**	**24**	**25**	**26**
8	77.1	78.3	73.5	75.9	72.28
9	78.2	78.6	78.9	75	73.5
10	79.51	79.5	78.3	74.69	74.69
11	78.6	79	81.9	78.1	77.3
12	79.5	78.31	78.9	80.7	79.51
13	80.7	73.5	**83**	78.31	74.69
14	80.7	78.3	78.3	77.1	78.3
15	78.3	77.1	78.6	78.2	77.4
16	74.5	76.2	74.69	77.5	77.4
17	71	74.5	72.2892	77.1	77.1

V. Conclusion

One of the problems to deal with is how to obtain suitable and enough training data. It is really important to be sure which constraints the project has and how to extract (record) the best representative data set.

Every attempt of building and training a system with these characteristics takes several minutes or hours. Therefore, it is really important to be neat in programming and making all the work organized and efficient.

The robustness of the system (83%) is considered good taking into account the kind of input data, which is very sensitive to the speaker, acoustic environment and even, the distance to the microphone. There are some methods in the literature [9] [10] [11] to deal with these issues, but they were not achievable in the frame of this work. They will be considered in further implementations.

The percentage of accuracy is referential and can differ from real world applications. The testing process applied in this work considers the audio files already trimmed and the same number of samples is assigned to each model. However, as mentioned, the main problem is due to *false positives*. The number of samples belonging to the class *"Noise"* is unpredictable in a real world application. This number is able to shift the accuracy quite a lot.

An underlying objective was the application of several topics related to *Pattern Recognition*. It is concluded that several tools such as HMM, GMM, kmeans, hierarchical clustering, and so on, were learned not only from the analysis point of view, but also in their application and implementation.

References

[1] Bahl L.R., Bellegarda J.R., "Multonic Markov word models for large vocabulary continuous speech recognition", IEEE Transactions on Speech and Audio Processing, vol. 1, no. 3, p. 334-344, July 1993.

[2] Mari J.F., Haton J.P., "Automatic word recognition based on second-order hidden Markov models", Proc. of the International Conference on Spoken Language Processing, p. 274-277, Yokohama, Japan, September 1994.

[3] Schukat-Talamazzini E.G., Kuhn T., Niemann H., "Speech recognition for spoken dialog systems", in De Mori R. Niemann H. And Hahnrieder G. (eds.), Progress and Prospects of Speech Research and Technology, Saint Augustin, Germany, 1994.

[4] Y. Zhao,"A Speaker-Independent Continuous Speech Recognition System Using Continuous Mixture Gaussian Density HMM of Phoneme-Sized Units", IEEE transactions on speech and audio processing, Jul 1993.

[5] Rabiner, L.R.; , "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE , vol.77, no.2, pp.257-286, Feb 1989.

[6] Mariani, Joseph. "Spoken Language Processing", John Wiley & Sons, Inc. Hoboken, NJ, USA. 2009.

[7] Scheirer, E.,'Tempo and Beat Analysis of Acoustic Musical Signals', J. Acoust. Soc. Am 103 (1), 588 - 601. 1998.

[8] Perry R. Cook, Music, cognition, and computerized sound: an introduction to psychoacoustics, MIT Press, Cambridge, MA, 1999.

[9] Boll, S. F. "Suppression of Acoustic Noise in Speech Using Spectral Subtraction". IEEE Trans. Acoust. Speech. and Sig. Proc. 2:113--120, 1979.

[10] Acero, A. and Stern, R. M. "Robust Speech Recognition by Normalization of the Acoustic Space". ICASSP-91, pages 893--896, May, 1991.

[11] Acero, A. and Stern, R. M. "Environmental Robustness in Automatic Speech Recognition". ICASSP-90, pages 849--852. April, 1990.

[12] Duda, Richard O., Peter E. Hart, and David G. Stork. "Pattern Classification." 2nd ed. New York: Wiley, 2001.