JOINT TRAINING OF CODEBOOKS AND ACOUSTIC MODELS IN AUTOMATIC SPEECH RECOGNITION USING SEMI-CONTINUOUS HMMS

Albino Nogueiras, Mónica Caballero, José B. Mariño

TALP Research Center
Universitat Politècnica de Catalunya (UPC). Barcelona, Spain.
{albino,monica,canton}@gps.tsc.upc.edu

RESUMEN

In this paper, three different techniques for building semi-continuous HMM based speech recognisers are compared: the classical one, using Euclidean generated codebooks and independently trained acoustic models; jointly reestimating the codebooks and models obtained with the classical method; and jointly creating codebooks and models growing their size from one centroid to the desired number of them. The way this growth may be done is carefully addressed, focusing on the selection of the splitting direction and the way splitting is implemented. Results in a large vocabulary task show the efficiency of the approach, with noticeable improvements both in accuracy and CPU consumption. Moreover, this scheme enables the use of the concatenation of features, avoiding the independence assumption usually needed in semi-continuous HMM modelling, and leading to further improvements in accuracy and CPU.

1. INTRODUCTION

Along the rest of this Introduction, an overview of semi-continuous Hidden Markov Models (HMMs), and a description of the experimental environment can be found. Follows, on Section 2, a description and comparison of the three alternatives for training codebooks and acoustic models considered. Finally, on Section 3 some conclusions are sketched.

1.1. Semi-continuous HMMs

An HMM is a collection of *states*. Each frame of voice can be in one and just one of the states at any time. Each HMM is formed of two different parts: a *transition matrix*, along with the *initial distribution*, and a set of *emission probability functions*. The transition matrix of an S-state HMM is an $S \times S$ matrix. Each element of the matrix represents the probability of moving from one of the states to another. The initial distribution represents the probability of being at each of the states at the beginning of the utterance. The transition matrix and initial distribution are common to all kind of HMMs, and are fundamental in the time warping capabilities of the method. Nevertheless, their exact value has shown little influence on the final results.

Each state in the model also contains an emission probability function that provides the probability with which this state generates any frame. There are several ways of defining emission probability functions, but the most usual way is by means of mixtures of Gaussian densities, trained with the expectation maximisation algorithm. Yet two alternatives are possible: continuous HMMs, and semi-continuous HMMs.

In continuous HMMs each state is modelled with a mixture of private Gaussians with diagonal covariance. As the number of Gaussians rapidly grows when the number of units and/or states grows, it is usual to tie together groups of them. This means that several states of several units share some of the Gaussian distributions, but not the mixture weights, which are still private. In semi-continuous HMMs all the Gaussians are shared, and form a vector quantifier. For each frame, the probability density of all the Gaussians is calculated, and a score vector is formed out of the highest of them. The probability of this frame being in a given state is equal to the sum-product of the score vector that represents the frame, and the mixture vector that represents the state.

Being $o = o_1 o_2 ... o_T$ an utterance of length T, λ the acoustic model corresponding to this utterance, we define $\mathcal{G}^k(o_t)$ as the value of the k^{th} Gaussian of the codebook for frame o_t, and c_{sk} the mixture weight of this Gaussian at state s. The likelihood of o_t at state s is given, for semi-continuous HMMs, by:

$$P(o_t/q_t = s) = \sum_{k=1}^{K} c_{sk} \mathcal{G}^k(o_t) \qquad (1)$$

The mixture weights c_{sk} are usually trained using the expectation maximisation algorithm. It holds that the likelihood is maximised at each iteration if the weights are updated with the value of the ratio between the expected number of times at state s and observing symbol k, and the expected number of times at state s. Defining the contribution of frame o_t to the k^{th} symbol at state s, $C_{sk}(o_t)$, as:

$$C_{sk}(o_t) = \alpha_t(s)\frac{c_{sk}\mathcal{G}^k(o_t)}{\sum_{k'} c_{sk'}\mathcal{G}^k(o_t)}\beta_t(s) \qquad (2)$$

Where $\alpha_t(s)$ is the *forward probability*, equal to the probability of o_t at state s given the model and all the frames of o from the beginning of the utterance until the current one: $\alpha_t(s) = P(o_1 o_2 \cdots o_{t-1} o_t, q_t = s/\lambda)$; and $\beta_t(s)$ is the *backward probability*, equal to the probability of o_t at state s given the model and all the following frames until the end of the utterance: $\beta_t(s) = P(o_{t+1} o_{t+2} \cdots o_T, q_t = s/\lambda)$. We have:

$$\bar{c}_{sk} = \frac{\sum_{t=1}^{T} C_{sk}(o_t)}{\sum_{k=1}^{K}\sum_{t=1}^{T} C_{sk}(o_t)} \qquad (3)$$

1.2. System Overview

Along the rest of the paper some alternatives of codebook and acoustic models training will be tested. The experimental framework within which this will be done is the recognition of a large vocabulary continuous speech task using RAMSES [1], the speech recognition system developed at UPC. Its main features are:

- Speech is windowed every 15ms with 30ms frame length. Each frame is parameterised with 12 Mel-frequency cepstral coefficients (MFCC) and energy, plus their first and second derivatives. Mean subtraction is applied to static parameters. A liftering window of length 22 is applied to the cepstral coefficients.
- Spectral parameters are quantified to 512 diagonal Gaussian centroids; energy ones are quantified to 128. Only the 32 highest scores are considered for spectral parameters; 8 in the case of energy ones.
- Demiphones are used as acoustic unit [2]. Each of them represents half phoneme with its closest neighbour explicitly considered. Contexts across words are not. 1000 demiphones are selected using a minimum entropy decision tree.
- Each demiphone is modelled with three states. The first and second states are possible entries to the model, and the second and third states are possible exits from it.
- Several pruning configurations are tested at recognition time. On the final results, only those leading to the highest accuracy at each level of CPU consumption are kept.

1.3. The Recognition Task

The task used is the recognition of free speech telephonic dialogues in a tourism information retrieval semantic domain. Both the training and the test material used were recorded inside the scope of the European Commission funded project LC-STAR [3], which is devoted to collecting lexica and corpora for automatic speech-to-speech translation.

The corpus is composed of 211 dialogues (422 different speakers), of which 16 define the standard test material. The remaining 195 dialogues are used for training the acoustic models. In mean, each dialogue lasts 9 minutes and is composed of 45 utterances, each of about 30 words. In total, 8418 utterances (29h45m, 230,000 words) were used for training, and 1040 utterances (3h40m, 28,000 words) were used as test material. Neither segmentation nor silence detection were carried by hand.

The objective of the task is recognising the words pronounced during the tourism information retrieval dialogues. The vocabulary of the task is composed of 7466 words, and the grammar perplexity is around 70. Both vocabulary and grammar were obtained from the training material. The test material presents an out-of-vocabulary rate over 1 %. For the results presented in this paper, the margins for a 95 % confidence in the error rates is around half a point.

2. THE DIFFERENT ALTERNATIVES

2.1. The Classical Approach: Using Euclidean Codebooks

In their classical implementation, codebooks and HMMs are trained in two phases: first we generate the codebooks, and then the HMMs are trained using them. The generation of the codebooks is done using the Lloyd algorithm in order to minimise the mean Euclidean distortion. The Gaussians of the final codebook are modelled with the means and variances of the clusters obtained in this way. During all the process of construction of the codebook each frame is quantified discretely, i.e. only to the closest centroid. At training and recognition times each frame is quantified to all the centroids in the codebook although, as just the highest are usually relevant, the lowest are usually discarded in order to alleviate the CPU load.

This approach has several advantages. Mainly its robustness, simplicity and low CPU consumption. Its main drawback is that it relies heavily on the Euclidean distance defined on the parameter space. This means that this space must be designed in a way that this distance is meaningful in terms of phonetic similarity. For instance, Euclidean distance is sensitive to the individual variance of each component in the feature vector, but this variance does not necessarily account for phonetic information, it may just account for a greater magnitude of this component respect to the rest of the vector. For instance, liftering of the cepstral coefficients leads to improvements of more than half a point in the experiments herein presented, and it is just a scaling of the cepstral coefficients, that should be neutralised by the Gaussian modelling.

A way to improve the effectiveness of the Euclidean

codebook modelling is to linearly transform the feature space in such a way that differences between components really reflect phonetic differences. Several strategies have been proposed—PCA, LDA, etc.—but either their implementation is too difficult, or it is not clear the benefits we may get. Actually, any linear transformation of the space should be almost completely voided by the Gaussian mixture modelling. If we used full covariance Gaussians, such a transformation would lead to new means and variances, but the probabilities given to each point of the space would be the same. In the case of using diagonal covariance Gaussians, the linear transformation also changes the component independence assumption, but this change should not have a remarkable effect if convenient estimation of the Gaussians is done. It is not the case if Euclidean codebooks are used, because they assume covariance equal to the identity.

The error rate achieved with this strategy at different CPU consumption levels is plotted on top of Fig. 1 (**classic**). This kind of system was soon superseded because of the mentioned limitations. We include it in this paper because it presents some benefits that make it interesting for certain applications. For instance, this kind of training is useful when different tasks are added to the system, and we do not want to train again all the rest of tasks. The use of a common, blind codebook enables us to simply train the mixture weights of the task specific models, without affecting the rest of them. In the rest of systems studied in this paper, the addition of a new task represents training again the acoustic models of the rest of tasks. Besides, the CPU consumption of the training phase is much lower than for the rest of alternatives—using Euclidean codebooks, the whole system used in this paper is trained in less than one day; the best results presented in this paper needed almost three weeks—.

2.2. The Baseline: Joint Re-Estimation of Codebooks and HMMs.

A first approach to mitigate the *blind* nature of the Euclidean distance based quantisation used in the classical system is the joint re-estimation of codebooks and HMMs. In this case, an initial codebook is used to train initial acoustic models as in the classical system, but then another optimisation process is performed where both the acoustic models and the Gaussians are re-estimated using expectation maximisation. The mixture weights re-estimation is performed in the same way as before. The formulae for reestimating the means and variances of the Gaussians are, using the contributions defined in (2):

$$\bar{\mu}_k = \frac{\sum_{s=1}^{S}\sum_{t=1}^{T} C_{sk}(o_t)\cdot o_t}{\sum_{s=1}^{S}\sum_{t=1}^{T} C_{sk}(o_t)} \quad (4)$$

$$\bar{\sigma}_k^2 = \frac{\sum_{s=1}^{S}\sum_{t=1}^{T} C_{sk}(o_t)\cdot (o_t-\mu_k)(o_t-\mu_k)^t}{\sum_{s=1}^{S}\sum_{t=1}^{T} C_{sk}(o_t)} \quad (5)$$

Equation (5) provides the full covariance matrix of each Gaussian distribution, in the experiments done in this paper, as well as in most of usual speech recognition systems, just the diagonal components of the covariance are considered, what is equivalent to supposing component independence.

Results using joint re-estimation of codebooks and HMMs are better than the ones achieved using the classical method in all cases (see Fig. 1, **joint**). Error rate decreases at all CPU levels by about two points, and it is clear that the system performs faster than the classical one.

Although the used of re-trained Gaussians, with means and variances adjusted to the shape of the feature space, should make the system immune to any scaling of the components, the use of liftering carries on to have a noticeable effect, of about half a point in the error rate. We think that this is due to the difficulty of moving the centroids from their original place. Somehow, the system cannot forget the initial codebook.

2.3. The alternatives: HTK/Julius

The above mentioned baseline system is the one that we have been using at UPC as a reference in our research work during the last times. It has been used in the recognition of a variety of tasks (from isolated words to continuous speech), of recording conditions (from recording studio to noisy GSM environments), and of languages (some 8 different languages, plus several dialectal variants of Spanish). In many of these tasks, parallel independent teams carried the same task using alternative recognition systems (HTK/Julius, Sphinx, Janus, etc.) Although direct comparison is very difficult, results were in general quite similar, particularly when CPU consumption limitations appeared. For instance, an independent team carried out the recognition of the task presented in this paper, with similar parameterisation and language model, but using HTK for building the acoustic models [4], and Julius for performing the recognition [5]. Their goal was to minimise the error rate at around 1'5 times real time on a Pentium Xeon at 3GHz.

The best result, 44'5 % at 1'53 times real time, was achieved when 4,000 context dependent triphones were modelled with 32 tied Gaussians per state. This result is also plotted in Fig. 1 (**HTK/julius**), and falls between the results of the classical system and our baseline [6].

2.4. Joint Growth of Codebooks and HMMs.

An alternative to first building an Euclidean codebook, then training initial acoustic models, and finally reestimating both codebook and models together is to grow Codebooks and HMMs together. The idea is similar to the Lloyd algorithm, and has been applied in the training of continuous HMMs:

1. An initial one centroid codebook is generated.

2. Models and codebook are trained using the expectation maximisation algorithm.

3. Each centroid in the codebook, as well as the corresponding weights in the models, is splitted into two.

4. The process is iterated from step 2 until the desired codebook size is reached.

The key question is how to split codebooks and models. In our work we divide the problem in two parts: selecting the direction of splitting, and performing the splitting itself.

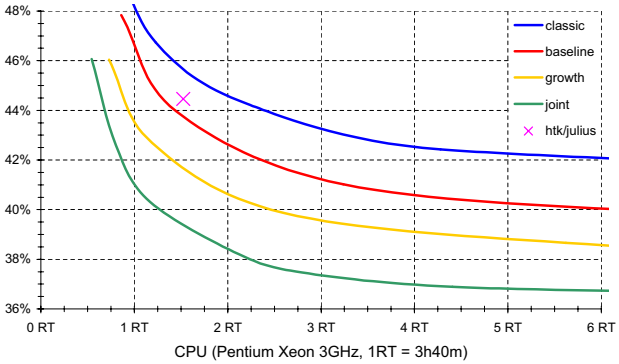

Figura 1. Word error rate in the recognition of the LC-STAR Spanish tourism retrieval task versus the CPU consumption.

2.4.1. *Selection of the splitting direction*

The main objective of the splitting is to generate two new centroids out of each previous one. The usual way of doing this is selecting the direction of splitting and moving forward and backward from the mean of the old centroid along this direction. In the classical Lloyd algorithm the direction of splitting is random, because the Euclidean discrete quantisation is very robust and this initialisation is not much relevant. In the joint growth of codebooks and HMMs we wish the split to be as effective as possible, while minimising the perturbation of the models at each step. A bad election of the splitting direction is readily passed to the HMMs weights, and the expectation maximisation algorithm may not be able to move the centroids to convenient positions.

The addressed problem is that of finding the direction of splitting that provides the maximum gain of information. Different criteria may be defined to find this direction [7]. A convenient way is selecting the direction where the separation between classes is maximised. In our case, the different classes can be defined by the states of the HMMs. In a space where the variance is equal in all the directions, the variance of the means of the classes is indicative of their separability using linear classification.

For each centroid, two different covariance matrices can be built using the contributions defined in (2): the overall covariance, $\mathbf{G}_k$ is the covariance matrix of all the contributions of each centroid; and the interclass covariance, $\mathbf{I}_k$, the covariance of the means of each class:

$$\begin{aligned} \mu_k &= \frac{\sum_{s=1}^{S}\sum_{t=1}^{T} C_{sk}(o_t)\cdot o_t}{\sum_{s=1}^{S}\sum_{t=1}^{T} C_{sk}(o_t)} \\ \mathbf{G}_k &= \frac{\sum_{s=1}^{S}\sum_{t=1}^{T} C_{sk}(o_t)\cdot (o_t-\mu_k)(o_t-\mu_k)^t}{\sum_{s=1}^{S}\sum_{t=1}^{T} C_{sk}(o_t)} \end{aligned} \quad (6)$$

$$\begin{aligned} \mu_{sk} &= \frac{\sum_{t=1}^{T} C_{sk}(o_t)\cdot o_t}{\sum_{t=1}^{T} C_{sk}(o_t)} \\ f_{sk} &= \sum_{t=1}^{T} C_{sk}(o_t) \\ \mathbf{I}_k &= \frac{\sum_{s=1}^{S} f_{sk}\cdot(\mu_{sk}-\mu_k)(\mu_{sk}-\mu_k)^t}{\sum_{s=1}^{S} f_{sk}} \end{aligned} \quad (7)$$

The product $\mathbf{G}_k^{-1/2T}\,\mathbf{I}_k\,\mathbf{G}_k^{-1/2}$, is equal to the covariance of the means in a variance-normalised space, so it reflects how much of the total variance is due to the separation between classes. In each direction, its value is comprised between zero and one, meaning zero that none of the variance in this direction is due to the separation between classes, and being close to one if most of the variance is due to it. The most effective linear split of the cluster, in the sense of reducing this variance, is following the hyperplane normal to the eigenvector of highest eigenvalue.

Notice that this product is insensitive to any linear transformation applied to the whole feature space, because it will be appear in all the terms of the product and cancel out. This includes any scaling of the coefficients, such as liftering, principal component analysis and linear discriminant analysis.

2.4.2. *Performing the splitting*

Chosen the direction of splitting, say $\mathbf{d}_k^*$, performing it is not a trivial task. In previous experiments centroids were splitted by generating two new centroids whose variance and mixture weights were the same as before, and the means were the previous one moved a certain amount forward and backward following the selected splitting direction. The problem was that, if a small movement was done, codebooks and models were too similar before and after the splitting, and the method converged to too close

centroids with little discrimination capabilities. On the contrary, if the movement was large, models and codebooks were too much disturbed at each step, and even convergence was compromised.

In order to maximally separate the centroids without perturbing too much codebooks and models, an additional step was introduced in the training procedure. In this step the contributions needed in (3) and (4) for reestimating the parameters of the grown codebooks and models are calculated as before, using the original size system. The difference is that the contribution is assigned to one of the two new centroids using linear classification along the splitting direction. In this way the separation between the two new centroids is maximised because each contribution is assigned to one or the other. At the same time, the system is minimally perturbed because the contributions are calculated using the original parameters, and the splitting of one cluster does not interfere with the rest of clusters.

$$\begin{aligned} \eta_k &= \mathbf{d}_k^{*t}\mu_k \\ \delta_{2k-1}(o_t) &= \begin{cases} 1 & \mathbf{d}_k^{*t}o_t > \eta_k \\ 0 & \text{otherwise} \end{cases} \\ \delta_{2k}(o_t) &= 1-\delta_{2k-1}(o_t) \end{aligned} \tag{8}$$

$$\begin{aligned} \bar{c}_{s2k} &= \frac{\sum_{t=1}^{T}\delta_{2k}(o_t)C_{sk}(o_t)}{\sum_{k=1}^{K}\sum_{t=1}^{T}\delta_{2k}(o_t)C_{sk}(o_t)} \\ \bar{\mu}_{2k} &= \frac{\sum_{s=1}^{S}\sum_{t=1}^{T}\delta_{2k}(o_t)C_{sk}(o_t)\cdot o_t}{\sum_{s=1}^{S}\sum_{t=1}^{T}\delta_{2k}(o_t)C_{sk}(o_t)} \\ \bar{\sigma}^2_{2k} &= \frac{\sum_{s=1}^{S}\sum_{t=1}^{T}\delta_{2k}(o_t)C_{sk}(o_t)\cdot (o_t-\mu_k)^t(o_t-\mu_k)}{\sum_{s=1}^{S}\sum_{t=1}^{T}\delta_{2k}(o_t)C_{sk}(o_t)} \end{aligned} \tag{9}$$

The formulae for $\bar{c}_{s2k-1}$, $\bar{\mu}_{2k-1}$, and $\bar{\sigma}^2_{2k-1}$, are similar as above for $\bar{c}_{s2k}$, $\bar{\mu}_{2k}$, and $\bar{\sigma}^2_{2k}$, except that $\delta_{2k-1}(o_t)$ should be used instead of $\delta_{2k}(o_t)$.

2.4.3. Experimental results using joint growth

The system built jointly growing codebooks and models achieved the lowest error rate of all the tested so far, reducing in more than one and a half points the error rate of the baseline, while also reducing even more the CPU consumption (see Fig. 1, **growth**). As expected, the system is now completely immune to liftering. Moreover, the result is almost identical whether linear discriminant analysis is used or not—we do not dispose of results using LDA on the previously described systems—.

2.5. Training the Whole Feature Vector Using Joint Growth of Codebooks and HMMs.

The fact that joint training of models and codebooks is able to increase the separability of the acoustic classes—as reflected by the improvement in both accuracy and speed—, and that it is immune to linear transformations of the feature space, encouraged us to undertake a rather difficult task: training semi-continuous models of high dimension features, namely the concatenation of the different energy and spectral features into one vector. The main implication of using this kind of feature is that different elemental features are no longer supposed to be independent.

Historically, using the whole vector with semi-continuous HMMs was impossible, or almost, because constructing a blind quantifier is increasingly difficult as the number of components grows. For instance, in order to ensure that each component in a vector of size P is split into at least two clusters, the size of the codebook should be 2^P. This is 4096 for a typical cepstral vector of size 12, but grows up to 500×10^9 when the whole vector of 39 components is used.

This problem does not appear in continuous HMMs. because they do not use a blind codebook, but phonetically guided mixtures instead. The same holds for the herein proposed algorithm—certainly, our system is very similar to a continuous system with all the Gaussians tied—. The results achieved when the whole vector was added to the former six ones are plotted on bottom of Fig. 1 (**whole**). As it stands out, it is not only possible to train such kind of information, but it still improves significantly both accuracy and speed. This is the best result we have ever achieved in this task using RAMSES or any other system.

3. DISCUSSION

The results presented in this paper clearly show that the classical ways of training semi-continuous HMMs can be improved by jointly growing models and codebooks from scratch. This technique also enables us to use high dimension feature vectors such as the concatenation of elementary features. The high performance obtained in this way is not only interesting from a recognition performance point of view. It is also interesting because the fact that a codebook of such dimension can be built opens a lot of possibilities in several other fields of speech processing such as text to speech synthesis or speech coding.

4. BIBLIOGRAFÍA

[1] A. Bonafonte et alter, "RAMSES: el sistema de reconocimiento del habla continua y gran vocabulario desarrollado por la UPC," in *VIII Jornadas de Telecom I+D*, 1998.

[2] J.B. Mariño et alter, "The demiphone: a new contextual subword unit for continuous speech recognition," *Speech Communication*, vol. 32, no. 3, pp. 187–197, October 2000.

[3] "LC-STAR: Lexicon and corpora for speech to speech translation," IST-2001-32216. http://www.lc-star.com.

[4] S. Young et alter, *The HTK Book 3.2*, http://htk.eng.cam.ac.uk, 2002.

[5] A. Lee et alter, "Julius: An open source real-time large vocabulary recognition machine," in *Proceedings of EUROSPEECH*, 2001, pp. 1691–1694.

[6] F.J. Punzano, *Sistema de Reconocimiento del Habla de Gran Vocabulario Basado en Julius. Entrenamiento mediante HTK*, Universitat Politècnica de Catalunya, 2006.

[7] T. Parsons, *Voice and Speech Processing*, McGraw-Hill, 1986.