CREATING HIDDEN MARKOV MODELS FOR FAST SPEECH BY OPTIMIZED CLUSTERING

R. Faltlhauser, T. Pfau, G. Ruske
Inst. for Human-Machine-Communication, Munich Univ. of Technology (TUM), Munich, Germany
Faltlhauser@ei.tum.de

ABSTRACT

Previous studies have shown that the recognition accuracy often severely degrades at higher speech rates, which can basically be traced back to two main dimensions: acoustic and phonemic. Reasons for this effect can be found in the phonemic field (e.g. elisions) as well as on the acoustic level: with increasing rates of speech the spectral characteristics are changing. A main obstacle in this context is the training data, consisting of only a small fraction of samples, which can be labeled as 'fast'. Therefore, the effects caused by an increased speech rate often cannot be completely covered. To meet this problem, in this paper an optimized clustering process is presented making efficient use of the available data. Our modified mixture splitting algorithm with an incorporated cross-validation step aims at increasing the generalization of Hidden Markov Models, especially with respect to fast speech. Experimental results showed a relative decrease in word error rate of 7.6% for fast speech.

1. INTRODUCTION

The accuracy in speech recognition highly depends on speech rate. Especially for higher speech rates it can be observed that recognition performance significantly degrades [1]. The reasons that lead to this degradation can be attributed to two main areas. Generally, speakers tend to assimilate phonemes or even omit them totally (elisions). Speaking faster, this phonetic problem is more intense. A common way to approach this problem is to incorporate pronunciation variants in the training phase as well as in the recognition phase of a speech recognition system. Moreover, a different weighting for the pronunciation variants may cover this speech rate dependent behaviour. The acoustic dimension causes additional effects, which amplify this degradation. At higher speech rates the vocal tract no longer reaches the final positions for certain phonemes [2], leading, for example, to a centralization of formant frequencies. Hence, the spectral characteristics will vary with speech rate.
The second problem is far more difficult to be treated. Speech data which can be labeled as 'fast' usually amount only a small fraction of the available training material. In standard spontaneous speech databases usually about 10 percent can be marked as fast - depending on the definition of 'fast'. Therefore, the effects caused by increased speech rate often are not adequately covered by the speech databases. Table 1 points out the result of this missing coverage, which is a rather drastic relative increase of 18.5% in word error rate. In this experiment, the testing material was split into 3 sets: slow, medium and fast speech. This paper especially deals with the effects of fast speech. Of course, when speaking (very) slowly, most people tend to hyper-articulate certain words, which may also deteriorate recognition performance [3]. Since hyper-articulation is not considered in this paper, slow speech material is not examined separately.

	total	fast
word error rate	37.8%	44.8 %

Table 1. Word error rates (WER) for entire test material and fast part separately.

The tagging of the training material was carried out by means of a rate-of-speech detector based on the ROS criteria [1]. Further, the models used were trained on the German Verbmobil Task, i.e. 11355 sentences of about 570 speakers. It is obvious, that recognition rate for fast speech only is substantially worse than for the whole testing material. In order to verify these findings, a look at the training score (log likelihood) is quite helpful: it can be considered as an indicator for the modeling accuracy. Breaking down the training material into 3 different subsets and quantifying the training score for the fast speech part yields the following result:

	total	med	fast
avg. score	-50,8	-51,1,	-54,4

Table 2. Decline of average training score (log likelihood).

Table 2 shows significantly worse training scores for the fast part of the training material which are already occuring during the training phase. Considering the obviously lower score in the training, it can be concluded that the difference in recognition performance is basically a problem of modeling. Two major reasons seem to cause this effect:

- Training algorithms (ML) do not represent fast speech (spectra) in placement of gaussians good enough.
- The balance between model size (usually number of gaussians) and available training data is even more delicate for fast speech.

Due to the lack of fast speech data, the effects at higher speech rates (e.g. formant centralization) cannot be adequately modeled by the training algorithm. This fact

however, could be overcome by gathering more samples of fast speech. Since this is not always possible in practice, existing fast speech samples have to be utilized more effectively instead. In the following a cluster algorithm is presented which tries to improve recognition performance by optimizing the generalizing power of models already at the clustering stage. The algorithm's main focus lies in the effective use of crossvalidation data to generate an adequate number of gaussians for each state of the individual Hidden Markov Model.

2. ALGORITHM

Most state-of-the-art recognition systems use Hidden Markov Models to represent the phonetic units. The models differ in structure (SCHMMs, CDHMMs, number of states, possible transitions) and size (number of prototypes). Modeling the statistical properties of the units, i.e. the probability density function (PDF) to describe the feature space is a common characteristic of most structures. Usually, a superposition of so-called prototypes, simple statistical functions such as gaussian densities, is used to achieve this goal. In order to optimize the system properly, the training algorithms (e.g. ML) for these models have to rely on *initial* prototypes. Generating initial prototypes is a typical task for a clustering process. In the following, such a clustering process is presented. The implementation of this algorithm is focused on CDHMMs with mixture gaussian densities, but the concept of the algorithm could be applied for other model structures as well.

In general, the number of gaussians K for each state of the Hidden Markov Model mostly is chosen constant or data dependent in a rather unsatisfactory manner. Typical clustering algorithms take into account only the data to be clustered itself. With enough gaussian densities, data can be modeled ideally, but no generalizing power would be left. The term 'generalization' characterizes the ability of models to recognize previously unseen speech patterns. Generally, a mismatch between available training data and speech data used to evaluate (or more generally: to use) the recognizer can be noticed and has to be taken into account. Every recognizer will be measured by its performance on 'unknown' evaluation data. Therefore, the ultimate goal for the clustering process has to be to produce a set of gaussians which are able to perform optimally on unknown speech data. It is necessary to estimate the quality of the resulting models' ability to cover 'unknown' speech data to reach this aim.

Using crossvalidation data is a common way to rate this capability [4]. Hence, the available training data has to be split into two parts:

- Clustering data
- Crossvalidation data

A closer look at the choice of the crossvalidation data reveals two opposing possibilities:

1. Random selection
2. Preselection

If the crossvalidation data are picked randomly from the entire fund of training samples, the two sets - crossvalidation and the clustering - will represent the *same* source distribution. Hence, an algorithm working on sets created this way will optimize the modeling of underlying source distribution. Another way would be to generate the crossvalidation set by some sort of pre-clustering, for example by speaker or rate of speech. In this case, the two sets could be interpreted as data from two possibly different sources. The second possibility is closer to the concept of 'estimation' - with the drawback, that samples which are used for crossvalidation are completely missing in the modeling process. Both approaches try to get an estimate from this intrinsic dissimilarity between crossvalidation distribution and clustering distribution. If both distributions are alike, it is probable that data from an unknown source will also be within this range of similarity. Considering the dissimilarity to be very high, it is not very sensible to provide the acoustic model with a large number of gaussians. It is favourable to spend less gaussians with increased variances instead. The following algorithm tries to automate this decision and to produce an adequate number of gaussians for the acoustic model.

Basically, the main algorithm is a combination of a standard K-means algorithm to revise the cluster-to-prototype association and a splitting step based on crossvalidation. The idea behind this hierarchical cluster algorithm is to split iteratively normal distributions until an optimal number of gaussians is reached.The main problem is to find this optimal number of gaussians to represent the training material. As mentioned above, the use of crossvalidation data is a possible solution: if the likelihood on the crossvalidation is no longer increasing, an optimum has been found. Therefore, the top-down cluster algorithm starts with a single prototype and iteratively adds new prototypes, which are created by a split of an existing prototype. To ensure a steady enhancement in the representation of the training data (given K prototypes) only S specific prototypes are splitted. In each iteration those prototypes are searched which yield the highest likelihood gain G on the crossvalidation data. In order to get a realistic impression of the modeling the prototypes are evaluated as gaussian densities with mixture coefficients c_k. The algorithm can be outlined as follows:

```
start: K = 1
repeat
  calculate means and variances for
    K gaussians by means of K-Means algorithm
  calculate mixture coefficients
  for all gaussians
    determine optimal splitting direction
    evaluate temporary split: gain G(k)
  j = arg max G(k)  (in case of S=1)
  if G(j) > T
    split gaussian: j
    K -> K+S
until G(j) < T
```

After every splitting iteration the association between clusters and their prototypes is revised, which is done by means of a K-Means algorithm. It minimizes iteratively the distortion measure $D(X,K)$ which applies the common covariance matrix C:

$$d(k,x_j) = (\mathbf{x}_j - \mathbf{m}_k)^T \mathbf{C}^{-1} (\mathbf{x}_j - \mathbf{m}_k)$$

$$D(X,K) = \sum_{j=1}^{P} \min_k d(k,x_j)$$

At the end of this reallocation step newly calculated means and covariance matrices are available. Within this K-Means process diagonal covariance matrices are calculated, which are used for the evaluation as gaussian densities. Only at the end of each reallocation step a full matrix is computed for each cluster. The main reason for calculating a full matrix lies in the subsequent splitting step: here a full covariance matrix is needed to compute an optimal direction for a prototype split. The cluster split should be performed along the main eigen axis of the covariance matrix. It determines the direction in feature space in which the cluster has its widest extension. The main axis can be computed by means of a principle components analysis (PCA): it is the eigenvector corresponding to the largest eigenvalue λ_{max} of the covariance matrix.

$$m_k \rightarrow \begin{cases} m_k + \delta e_{\lambda_{max}} \\ m_k - \delta e_{\lambda_{max}} \end{cases}$$

The given mean vector m_k ist temporarily replaced by the two new mean vectors. Before the split can be evaluated on the crossvalidation data, the position of the two means has to be revised according to the feature vectors belonging to cluster k. For the evaluation purpose each gaussian is weighted with its mixture coefficient c_k:

$$c_k = \sum_{j=1}^{P} p(x_j)p(k|x_j) \approx \frac{1}{P}\sum_{j=1}^{P} p(k|x_j)$$

using a hard assignment:

$$p(k|x) = \begin{cases} 1 & k = argmin_{k^*}\, d(k^*, x) \\ 0 & else \end{cases}$$

At first, each split is done only tentatively to evaluate its performance on the crossvalidation data. Finally those S split(s) are accepted and kept, that yield the highest likelihood gain on the crossvalidation material. To achieve optimal performance with respect to modeling accuracy, here only one prototype is splitted per iteration ($S = 1$). Increasing S would reduce computation time, because fewer iterations are performed, at the expense of worse modeling accuracy. The whole process is iterated until no further likelihood gain is achieved or the maximum gain falls below a given threshold T.

Crossvalidation data is incorporated in two ways: firstly, it determines the algorithm's termination point and secondly it determines the gaussian which is to be splitted. In this algorithm however, the clustering performs a full search, whether a split should be carried out, or not. It is quite obvious that the computational load here is a lot higher than for standard algorithms. But in the end it provides seed prototypes, that are better related to the model.

The algorithm itself is self-terminating. Several parameters strongly influence the resulting number of gaussians:

- the gain threshold T and
- the number of splits per iteration S that are performed.

Keeping T konstant, an increased S ($S > 1$) can produce a higher resulting number of prototypes.

3. RESULTS

3.1. General

First experiments were carried out to test the algorithm (Optimized Clustering, OC) itself and to compare it with a common clustering algorithm (LBG). So the entire data set was used for clustering: approximately 10 million frames (VERBMOBIL spontaneous speech database, CDs 1-12), 44 HMMs with 3 to 4 states, segmentation based on pronunciation variants. For each state the data was randomly split into a crossvalidation set and a set subject to clustering. Therefore, clustering data and crossvalidation describe the same source distribution. To train the models a standard Maximum Likelihood training algorithm was used.

	total	fast	#Prototypes
WER, LBG	37.8 %	44.8 %	8657
WER, OC(1)	36.3 %	43.3 %	5913
WER, OC(2)	35.9 %	41.4 %	8498

Table 3. Word error rates after 4 iterations of ML training, for the total and for the fast material

Table 3 shows a relative decrease in word error rate of 3.3% for the model set OC(1) together with a remarkable decrease in model size of 31.7%. For the in size equivalent model set OC(2) even a relative reduction of 7.6% subject to further adaptation steps is achieved. Different model sizes can be created by changing the termination threshold T. With model set OC(2) we were able to reduce the span between recognition performance on fast sentences and all sentences from 18.5 % relative (LBG) to 15.3%.

Figure 1 shows a sample of the performance (log. likelihood scores) of the OC algorithm on the crossvalidation data during the clustering process. The process is usually iterated until it reaches the saturation region, i.e. it either terminates automatically (no further increase in crossvalidation score) or it can be terminated ($T > 0$) earlier. An earlier termination may be favourable for those phonemes, that have a very low intrinsic variability. In such cases, the high similarity between data and crossvalidation data causes the algorithm to produce a too large number of prototypes. Incorporating further mass constraints in the splitting step is a possible way to meet this side effect.

Figure 1 shows an increase in crossvalidation score with an ascending number of gaussian densities. Due to the fact that the distribution of both data sets, crossvalidation as well as clustering data, are modeled simultaneously, there is a constant increase in likelihood to be observed. An explanation for the improved recognition performance can be found by comparing the distribution of model sizes.

Table 4 shows the number of prototypes produced by our OC algorithm in comparison with a standard mass constrained (number of feature vectors per cluster) LBG algorithm. The main difference lies in the scattering of the model size for the individual phonemes. LBG conspiciously tends to provide particularly the non-speech models with a very high number of gaussians. Mainly the large number of feature vectors available for these models can be held responsible for this effect. In contrast thereto, OC produces rather consistent model sizes for all phonemes (phonemes not listed here are in the same

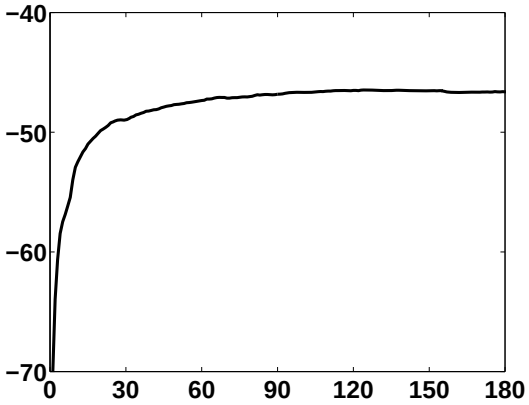

Figure 1. Phoneme /OY/: score (log. likelihood) as a function of the number of prototypes.

	#Prototypes			#Frames
Phoneme	LBG	OC(1)	OC(2)	
/a/	227	125	176	256255
/e:/	169	157	251	188525
/o:/	94	159	230	93304
/y:/	20	127	169	13243
/f/	210	118	178	258732
/j/	78	140	211	71524
/n/	673	141	184	600000
/z/	139	139	211	132901
/s/	590	138	226	600000
<p:>	1122	162	184	600000
<nib>	927	126	134	600000
...	...	...	...	...
TOTAL	8657	5913	8498	

Table 4. Comparison of model size for various phonemes.

range). Moreover, the high intrinsic variability especially of the non-speech model <nib> becomes obvious by the very low increase from 126 to 134 prototypes. Only a small likelihood gain was achieved through further splitting of prototypes. For some phonemes the training data actually used for clustering was limited to 600k feature vectors (200k per state), due to computational restrictions.

3.2. Adaptation to Fast Speech

One of the main questions was how the outcome of clustering processes can be optimized for fast speech without having to increase the training corpus. Hence, the second application was aimed at the introduction of a fast speech part as crossvalidation data. The samples belonging to the fast sentences were preselected and gathered for the crossvalidation corpus.

	total	fast	#Prototypes
WER, OC(3)	40.9 %	42.5 %	5020
WER, OC(4)	38.3 %	43.9 %	5020

Table 5. Word error rates after 4 iterations of ML training, for the total and for the fast material

Both model sets OC(3,4) were identically initialized, i.e. they share the same initial model set produced by our OC algorithm. The main difference lies in the choice of data used for the training algorithm (ML). Set OC(3) was trained by 4 iterations ML only with the fast speech part of the training data, whereas for OC(4) the entire data was used. OC(3) shows a relative reduction in WER of 5.1% for fast speech together with an expected increase in WER for the whole testing material. For model set OC(4) only a slight decrease in word error rate was possible, due to two contradictory adaptations: on the one hand an initialization with respect to fast speech and on the other hand the training (ML) step s with the entire data.

4. CONCLUSION

With our approach we have shown the necessity of spending considerable computing time already at the stage of initial clustering. Training algorithms often only reach poor local minima during the training process. This annoying fact can, at least partially, be avoided if the seed models are already optimized. Of course, this optimization process is far more time consuming than a simple fixed (K prototypes) clustering process. Anyway, this effort has to be only made during the initialization, i.e. as part of the training phase. A further improvement could possibly be made by introducing ML reestimation during clustering to revise the cluster-to-prototype assignment. However, this would mean a further increase in time consumption, but for the training phase computing time is usually not a real problem.
Our OC algorithm performs a heuristic search to find a better optimum. Despite this full search in finding a good split it remains a greedy algorithm. To circumvent this problem algorithms can be applied which aim at finding the global minimum of a distortion measure, such as simulated or deterministic annealing [5, 6]. However, these algorithms tend to be as time consuming.

5. ACKNOWLEDGEMENTS

This work was partially funded by the "Deutsche Forschungsgemeinschaft" (DFG).

REFERENCES

[1] T. Pfau, G. Ruske, "Creating Hidden Markov Models for Fast Speech", Proc. ICSLP 98, paper 255, pp. 205-208

[2] H. Kuwabara, "Acoustic and Perceptual Properties of Phonemes in Continuous Speech as a Function of Speaking Rate", Proc. Eurospeech 97, pp. 1003-1006

[3] H. Soltau, A. Waibel, "On the Influence of Hyperarticulated Speech on Recognition Performance", Proc. ICSLP 98, paper 736, pp. 225-228

[4] T. Kemp, "Data-Driven Codebook Adaptation in Phonetically Tied SCHMMs", Proc. ICASSP 95, pp. 477-479

[5] K. Rose, "Deterministic Annealing for Clustering, Compression, Classification, Regression, and Related Optimization Problems", Proc. IEEE, vol. 86, 1998, pp. 2210-2239

[6] K. Rose, E. Gurewitz, G. C. Fox, "Vector Quantization by Deterministic Annealing", IEEE Trans. Inform. Theory, vol. 38, 1992, pp. 1249-1257