Diploma Thesis: Data-Driven Speaker and Subword Unit Clustering In Speech Processing

Student: Micha Hersch, Section d'Informatique, EPFL
Under the direction of Prof. Hervé Bourlard, EPFL
and Prof. Nelson Morgan, ICSI

March 19, 2003

Contents

Abstract

In [1], a segment-based automatic clustering algorithm was proposed. In the first part of his work, the algorithm is used for speaker clustering. The effect of its parameters are studied and some modifications to improve its efficiency are presented.
The second part of this report presents an attempt to use this algorithm to automatically derive subword units, and use them in a speech recognition system whose dictionary is automatically generated, and therefore doesn't use any linguistic knowledge of any kind. Two different recognition systems are proposed and tested on the Numbers95 database.

1 Acknowledgments

I would first of all like to thank Prof Hervé Bourlard for making it possible for me to do this research work at the International Computer Science Institute (ICSI) at Berkeley in the framework of the IM2-ICSI agreement. I also thank Prof Nelson Morgan for accepting me in his institute. Moreover, I would like to thank the ICSI researchers for their help and support, in particular Charles Wooters, Barry Chen, Panu Somervuo, Jitendra Ajmera and David Gelbart.
I am also grateful to the Swiss National Center of Competences in Research on Multimodal Information Management (IM2) for providing me with a scholarship.

2 Introduction

Although the science community has been trying to tackle the problem of speech recognition for many decades, and has indeed earned major successes, a lot still remains to be done in this field. But because of its age and popularity, it has become very challenging to come up with new approaches to this problem, which have not been already tried and which can compete with well established traditional approaches.

In [1], a segment-based automatic clustering algorithm was developed to perform speaker clustering and segmentation. Given the unlabeled speech signal of several people talking one after the other, this algorithm can determine how many speakers there are, produce a speaker clustering of the speech signal and therefore tell who is talking when. This algorithm, and the model underlying it, uses very little external knowledge about speech, needs very few parameters and is hence very general. The generality of this algorithm makes it very tempting to try to use it in order to automatically derive subword units. So in the first part of this project the algorithm is examined in the speaker clustering framework, and some improvements are proposed in order to increase its efficiency. Those improvements are necessary in order to conveniently use this algorithm for the subword unit clustering, which is presented in the second part of this work. In this second part a speech model is developed, where all knowledge is data-driven, and hence does not need any linguistic knowledge.

Part I
Speaker Clustering

3 Introduction

The problem of speaker clustering is widely known. It can be expressed in the following way: given an audio recording where several speakers speak one after another, as in a radio show for example, the aim is to determine the number of speakers and for each of them, the times at which they started speaking and at which they stopped. The problem is to find who spoke when.
This information can in itself be interesting, for example when automatically transcribing radio shows, but it is also very useful when doing speech recognition. Indeed, if this information is known, one can then apply speaker adaptation techniques like Vocal Tract Length Normalization (VTLN) which improve the speech recognition. Those techniques adapt the speech model to each speaker, thus yielding better recognition rates.

4 Theoretical Framework

4.1 Speaker Clustering And Model Selection

One common way of guessing the number of speakers in a speech sequence is to have a set of models, each of them modeling a speech sequence involving a different number of speakers. One finds the model that best fits the data (the speech sequence) and one hopes that this model indicates the actual number of speakers in the considered speech sequence. Thus, the speaker clustering problem we are trying to solve can be viewed as a problem of model selection. This is a very classical problem in statistics. The goal is to find a model that best represents a data set. By this we mean not merely a model which maximizes the likelihood of the data, but a model whose complexity mirrors the true underlying complexity of the data. Such a model would not only fit the training data, but would be general enough to also fit data on which it was not trained. As it is always possible to raise the likelihood by choosing a more complex model, there is a trade-off to be reached between the model complexity, i.e. its number of parameters, and the likelihood of the data.

4.2 Bayesian Information Criterion (BIC)

In [2] the following method for choosing the dimension (or complexity) of a model was proposed:

Choose the model j for which $\log ML_j(\boldsymbol{x}^1, \dots, \boldsymbol{x}^n) - \frac{1}{2}k_j \log n$ is largest,

where ML_j is the likelihood according to model j, k_j is the dimension of model j and $(\boldsymbol{x}^1, \dots, \boldsymbol{x}^n)$ is the data set. Although [2] shows that this method is optimal as n reaches infinity, in practical applications the penalty term $\frac{1}{2}k_j \log n$ always needed to be modulated by a constant in order to achieve better results. This constant was to be set empirically, because no theory accounted for it.

4.3 Gaussian Mixture Models (GMM)

A Gaussian mixture model, or a multi-Gaussian is a probability distribution function

$$p(\boldsymbol{x}) = \sum_{m=1}^{M} c_m \cdot g_m(\boldsymbol{x}) \tag{1}$$

where M is the number of mixture components, the c_m are the mixture component weights (and sum to one). A multi-Gaussian is a weighted sum of Gaussian functions g_m, which are determined by their mean μ_m and their covariance matrix Σ_m:

$$g_m(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2} \mid \Sigma_m \mid^{1/2}} \cdot \exp\left\{ -\frac{1}{2}\Big((\boldsymbol{x} - \mu_m)^T \Sigma_m^{-1} (\boldsymbol{x} - \mu_m) \Big) \right\} \tag{2}$$

where d is the dimension of the space.
A useful algorithm for estimating the parameters of a GMM that maximize the likelihood of a set of n data vectors $\{\boldsymbol{x^i}\}$is the Expectation Maximization algorithm. This algorithm works by iteratively updating the parameters according (in the case of diagonal covariance matrices) to the following equations:

$$\begin{aligned}
\mu_m^{new} &= \frac{\sum_{i=1}^{n} P(m \mid \boldsymbol{x^i}, \theta) \cdot \boldsymbol{x^i}}{\sum_{i=1}^{n} P(m \mid \boldsymbol{x^i}, \theta)} && (3) \\
\Sigma_m^{new} &= \frac{\sum_{i=1}^{n} P(m \mid \boldsymbol{x^i}, \theta) \cdot (\boldsymbol{x^i} - \mu_m)^T (\boldsymbol{x^i} - \mu_m)}{\sum_{i=1}^{n} P(m \mid \boldsymbol{x^i}, \theta)} && (4) \\
c_m^{new} &= \frac{1}{n} \sum_{i=1}^{n} P(m \mid \boldsymbol{x^i}, \theta) && (5) \\
& && (6)
\end{aligned}$$

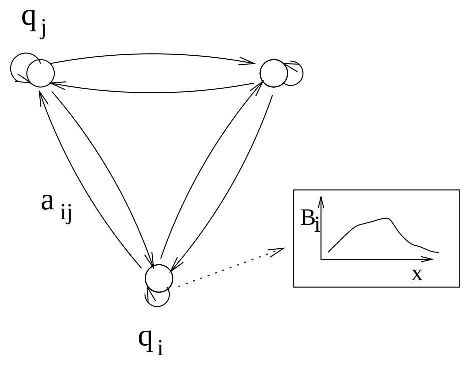
where $\theta = \{\{\mu_m\}, \{\Sigma_m\}, \{c_m\}\}$ is the set of current parameters. The value $P(m \mid \boldsymbol{x^i}, \theta)$ can be computes as

$$P(m \mid \boldsymbol{x^i}, \theta) = \frac{c_m g_m(\boldsymbol{x^i})}{\sum_{j=1}^{M} c_j g_j(\boldsymbol{x^i})} \tag{7}$$

4.4 Hidden Markov Models (HMM)

A Hidden Markov Model can be described as a set of Q states $\{q_i\}$, a transition matrix $A = \{a_{ij}\}$, a set of d-dimensional probability density functions $\{B_i\}$ and an initial state probability vector $\boldsymbol{\rho} = \{\rho_i\}$ with i and j ranging from 1 to Q. The system starts at time 0 in a state q_i with a probability ρ_i. When in a state q_i at time t a d-dimensional vector $\boldsymbol{x^t}$ is emitted according to a probability $B_i(\boldsymbol{x^t})$ and at time $t+1$ the system moves on to state q_j with a probability a_{ij} and so on, generating a sequence of T observation vectors $\boldsymbol{x^t}$. A graphical illustration of a fully connected three state HMM can be found in figure 1.
Two useful algorithms for HMMs are the Baum-Welch and the Viterbi algorithm.

Figure 1: *A fully connected three state HMM. The circles represent the states and the arrows possible transition between states. Each state q_i has an emission probability density function B_i.*



The Baum-Welch algorithm finds the parameters θ of an HMM, $(A, \{B_i\}, \boldsymbol{\rho})$ that maximizes the likelihood of a sequence of observation vectors. The Viterbi algorithm, which is used in the speaker clustering algorithm is described in the following paragraph.

The Viterbi Algorithm The Viterbi algorithm finds the most likely sequence of states given the model and the observations. It consists of computing the following recursive equation over times t and states i:

$$p(q_i^0, \boldsymbol{x^0}) = \rho_i \cdot B_i(\boldsymbol{x^0}) \tag{8}$$

$$p(q_i^t, \boldsymbol{x^1}, \ldots, \boldsymbol{x^t}) = \max_j (p(q_j^{t-1}, \boldsymbol{x^1}, \ldots, \boldsymbol{x^{t-1}}) \cdot B_i(\boldsymbol{x^t}) \cdot a_{ji}) \tag{9}$$

This equation can be computed efficiently using dynamic programming in the log-space, and by keeping track of the j's for each iteration one gets the most likely state sequence.

4.5 Mathematical Model

The algorithm described here avoids the problem of choosing the right model dimension by reducing speaker clustering to a model selection problem, in which all models have the same dimension. The considered model is a first order HMM containing K states (or clusters). Each state represents a speaker, has a minimum duration constraint L, and generates the data according to a Gaussian mixture probability density function (pdf). The transition probabilities from one cluster to another are the same for all clusters and are held fixed. The representation of the resulting model can be seen in figure 2.
On such a model, one can define the merging operation on two clusters, which consists of replacing them with one cluster with a probability density function which is the (weighted) sum of the two disappearing pdfs. The resulting Gaussian mixture contains as many mixture components as the number of components in the two disappearing clusters. Thus, the merging of two clusters does not change the number of parameters in the model. To be more precise, the merging of two clusters

Figure 2: *The model used for speaker clustering, with 4 clusters. Each branch of the star represent a cluster, which is a succession of L (minimum duration) states which all share the same multi-Gaussian probability density function (pdf) as represented with the dotted arrows pointing to the same pdf. The pdfs of different clusters differ and may not have the same number of Gaussian components. From the last state of a cluster $\mathcal{C}_i$, one can reach the beginning of any other cluster with the same probability, or stay in the same cluster with probability a_i.*

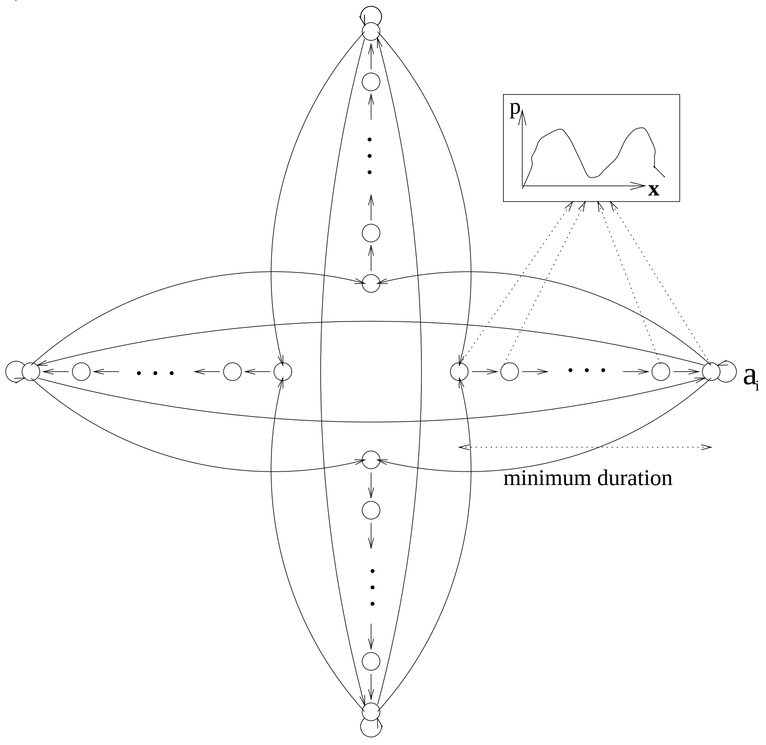
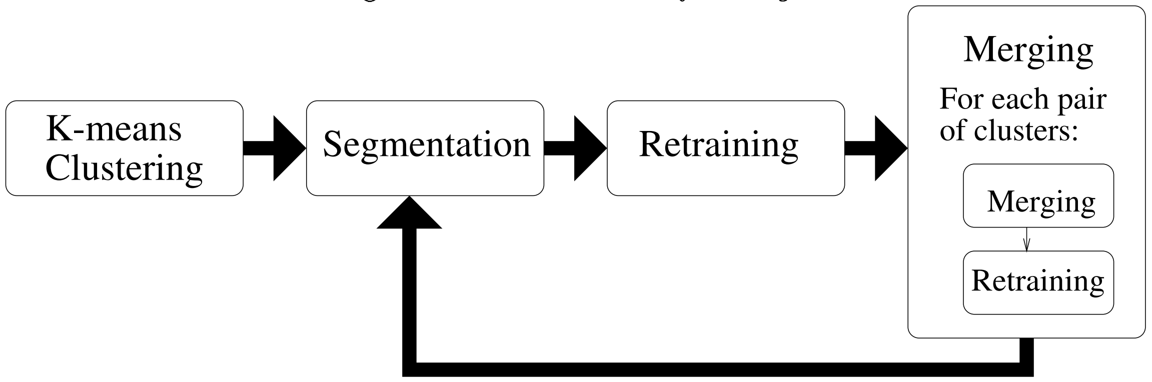


Figure 3: *The structure of the algorithm*



adds one degree of freedom to the model, because as one cluster disappears, the constraint on the weights of its Gaussian mixture components disappears as well. But this approximation will be solved in section 6.4.
As the number of parameters stays (almost) constant, one can use the likelihood criterion in order to decide whether or not to merge two clusters. In other words, given a segmentation $\mathcal{S}$, the clusters $\mathcal{C}_1$ and $\mathcal{C}_2$ with probability density functions p_1 and p_2 respectively are merged into a cluster $\mathcal{C}_3$ with probability density function p_3 only if this raises the likelihood of the data, i.e.

$$P(\boldsymbol{x^1},\ldots,\boldsymbol{x^T} \mid \mathcal{M}_1) \quad < \quad P(\boldsymbol{x^1},\ldots,\boldsymbol{x^T} \mid \mathcal{M}_2) \qquad (10)$$

$$\Leftrightarrow P(D_{\mathcal{S}}(\mathcal{C}_1) \mid \mathcal{C}_1) \cdot P(D_{\mathcal{S}}(\mathcal{C}_2) \mid \mathcal{C}_2) \quad < \quad P(D_{\mathcal{S}}(\mathcal{C}_1), D_{\mathcal{S}}(\mathcal{C}_2) \mid \mathcal{C}_3) \qquad (11)$$

$$\Leftrightarrow \sum_{\boldsymbol{x}\in D_{\mathcal{S}}(\mathcal{C}_1)} \log(P(\boldsymbol{x} \mid p_1)) + \sum_{\boldsymbol{x}\in D_{\mathcal{S}}(\mathcal{C}_2)} \log(P(\boldsymbol{x} \mid p_2)) - \qquad (12)$$

$$\sum_{\boldsymbol{x}\in D_{\mathcal{S}}(\mathcal{C}_1)\cup D_{\mathcal{S}}(\mathcal{C}_2)} \log(P(\boldsymbol{x} \mid p_3)) \quad < \quad 0$$

$$\Leftrightarrow \sum_{\boldsymbol{x}\in D_{\mathcal{S}}(\mathcal{C}_1)} \log(\frac{P(\boldsymbol{x} \mid p_1)}{P(\boldsymbol{x} \mid p_3)}) + \sum_{\boldsymbol{x}\in D_{\mathcal{S}}(\mathcal{C}_2)} \log(\frac{P(\boldsymbol{x} \mid p_2)}{P(\boldsymbol{x} \mid p_3)}) \quad < \quad 0 \qquad (13)$$

where $D_{\mathcal{S}}(\mathcal{C}_i)$ is the data belonging to cluster $\mathcal{C}_i$ according to segmentation $\mathcal{S}$, p_3 is obtained after retraining over $D_{\mathcal{S}}(\mathcal{C}_1) \cup D_{\mathcal{S}}(\mathcal{C}_2)$, $\mathcal{M}_1$ is the model where $\mathcal{C}_1$ and $\mathcal{C}_1$ are kept separated and $\mathcal{M}_2$ is the model where those clusters are merged together. This criterion is equivalent to the BIC criterion, but since the k_j is constant the second term in mathematical expression of section 4.2 can be discarded.

The basic idea of this algorithm is two start with a model containing a large number of clusters and successively merge clusters and retrain the model as long as the likelihood of the data increases.

5 Description

The algorithm, which is summarized in figure 3, consists of the following steps .

1. Cluster the data in K clusters using a standard K-means algorithm.

2. Model each cluster with a trained multi-Gaussian pdf containing M mixture components.

3. Segment the data using the Viterbi algorithm on an HMM topology respecting the minimum duration constraint.

4. For each cluster, retrain its GMM on the features belonging to that cluster according to the current segmentation, using the EM algorithm.

5. For each pair of clusters, merge them (in keeping the number of Gaussian mixture components constant), retrain (with EM) the newly obtained cluster on the features belonging to those two clusters according to the current segmentation and compute the merging score, i.e. the left side of inequality 13.

6. If no merge yields a negative score, end, else keep the merge yielding the best (i.e the lowest) merging score and leave the other clusters unchanged and go to 3.

6 Variations On The Algorithm

The initial version of the algorithm gets pretty slow as the number of initial clusters rises. In this section, new and faster versions of this algorithm are described.

6.1 The "Pick First" Option

This version differs from the original version in that it is not the best of all merges that is performed. Rather, merges are tried in a random order and one performs the first merge that yields a negative score, i.e that satisfies inequality 13. The idea is that it isn't necessary to find the merge which most raises the likelihood, that merges can be performed, that are not the best ones at this point. This way, it isn't necessary to try all $\frac{1}{2}K(K-1)$ merges.

6.2 The "Fast Metric" Option

The idea behind these versions is to find a quickly computable distance or similarity measure that would give a good indication whether two clusters are likely to satisfy inequality 13. Before picking the two clusters to be merged, one computes this measure for all pairs of clusters, sorts them and tries the merges in the resulting order. This way, one hopes to find a pair of clusters that satisfies inequality 13 faster than if the pairs were picked randomly as described in the previous paragraph. Five different measures were studied and are described in the following paragraphs.

6.2.1 Likelihoods Of The Means

The first distance measure is the weighted sum of the log likelihood of the means of the Gaussian mixture of one cluster according to the pdf of the other cluster and symmetrically. More precisely, if clusters $\mathcal{C}_i$ has a multi-Gaussian pdf

$$p_i = \sum_{m=1}^{M_i} c_{i,m} \cdot g_{i,m} \tag{14}$$

where M_i is the number of mixture components in cluster i, the $c_{i,m}$ are the mixture components weights (and sum to one for every i) and $g_{i,m} = \mathcal{N}(\mu_{i,n}, \Sigma_{i,n})$ are the Gaussian mixture components, then the distance between clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ is computed as

$$D_1(\mathcal{C}_i, \mathcal{C}_j) = \sum_{m=1}^{M_i} \log(c_{i,m} \cdot P(\mu_{i,m} \mid p_j)) + \sum_{m=1}^{M_j} \log(c_{j,m} \cdot P(\mu_{j,m} \mid p_i))$$

This distance measure can be understood in he following way. As the aim is to merge clusters that have similar pdfs, i.e. that give high probabilities to the same regions of the feature space. Since the pdfs are estimated on their cluster's segment, one can say that one tries to merge two clusters if the pdf of one of them also well explains the segments belonging to the other cluster. If we understand the means of the Gaussian mixture of one cluster as representants of the data of this cluster (like in vector quantization), this measure gives (if the two clusters have about the same number of points) a very rough approximation of $logP(\boldsymbol{x_i} \mid p_j) + logP(\boldsymbol{x_j} \mid p_i)$, where $\boldsymbol{x_i} \in D_{\mathcal{S}}(\mathcal{C}_i)$ and $\boldsymbol{x_j} \in D_{\mathcal{S}}(\mathcal{C}_j)$, which should be bigger the closer p_i is from p_j.

6.2.2 Volume Of The Squared Difference Of The Pdfs

If two pdfs are alike, the absolute value of there difference will in general be small, and therefore the square of their difference will also be small. Accordingly, the distance measure looks at the volume of the squared difference of the pdfs. More precisely, it is

$$D_2(\mathcal{C}_i, \mathcal{C}_j) = \int (p_i(x) - p_j(x))^2 dx$$

This distance measure, suggested by [3], can be computed pretty efficiently, but some mathematical development, presented in appendix A.1, is needed in order to get the the formula, which is

$$\int (p_i(x) - p_j(x))^2 dx = \frac{1}{(2\pi)^{d/2}} \cdot \Big(\sum_{k,l \in \{i,j\}} \sum_{m_1=1}^{M_k} \sum_{m_2=1}^{M_l} (-1)^{\delta(k,l)} c_{k,m_1} c_{l,m_2} \cdot \tag{15}$$

$$\frac{1}{\mid \Sigma_{k,m_1} + \Sigma_{l,m_2} \mid^{1/2}} \exp\{-\frac{1}{2}(\Sigma_{k,m_1} + \Sigma_{l,m_2})^{-1}(\mu_{k,m_1} - \mu_{l,m_2})^2\}\Big) \tag{16}$$

where $\delta(\cdot,\cdot)$ is the Kronecker's symbol, $c_{\cdot,\cdot}$, $\mu_{\cdot,\cdot}$ and $\Sigma_{\cdot,\cdot}$ are respectively the weights, means, and (diagonal) variances of the mixture components, as described in equation 14.

6.2.3 The Inner Product Of The Pdfs

This distance measure computes the expected probability of a vector drawn from one probability density function according to the other. This can be expressed as:

$$D_3(\mathcal{C}_i, \mathcal{C}_j) = E_{p_j}(p_i(\boldsymbol{x})) \tag{17}$$
$$= \int p_i(\boldsymbol{x}) p_j(\boldsymbol{x}) d\boldsymbol{x} \tag{18}$$
$$= \sum_{m_1=1}^{M_i} \sum_{m_2=1}^{M_j} c_{i,m_1} c_{j,m_2} \cdot \frac{\exp\{-\frac{1}{2}(\Sigma_{k,m_1} + \Sigma_{l,m_2})^{-1}(\mu_{k,m_1} - \mu_{l,m_2})^2\}}{\mid \Sigma_{k,m_1} + \Sigma_{l,m_2} \mid^{1/2}} \tag{19}$$

where $E_f(\cdot)$ denotes the expectation according to probability density function f. The last equality was derived using the mathematical developments presented in A.1.

This expression is symmetric, which suits very well our purpose, and one recognizes the inner product of p_i and p_j (also called the dot product).

6.2.4 The Angle Between The pdfs

The preceding similarity measure, is the inner product of the pdfs. Since the pdfs are L_1 normalized and not L_2 normalized, the similarity between the two same pdf depends on its L_2 norm, which is not exactly what is desired. It would be like using the inner product to compute the similarity between vectors of different lengths. In order to account for this, one can use the angle between the two pdfs as a similarity measure. The cosine of the angle between to elements p_i and p_j of a vector space is defined as

$$D_4(\mathcal{C}_i, \mathcal{C}_j) = \frac{< p_i, p_j >}{(< p_i, p_j > \cdot < p_i, p_j >)^{1/2}} \tag{20}$$
$$= \frac{\int p_i(\boldsymbol{x}) p_j(\boldsymbol{x}) d\boldsymbol{x}}{(\int p_i(\boldsymbol{x})^2 d\boldsymbol{x} \cdot \int p_j(\boldsymbol{x})^2 d\boldsymbol{x})^{1/2}} \tag{21}$$

where the integrals are computed using the results of the previous sections.

6.2.5 Adjacency Of The Segments

Unlike the four preceding distance measures, this one does not look at the pdfs at all. Since we start by over-clustering the data, and because the minimum duration is usually much smaller than the actual expected "true" segment length, a speaker will usually be segmented into many alternating clusters, which (if the algorithm is successful enough) will eventually be merged together. One can therefore think that clusters which often follow one another are more likely to satisfy the merging condition 13. Thus, the third distance is simply the number of times the segments of two clusters are adjacent one to the other.

$$D_5(\mathcal{C}_i, \mathcal{C}_j) = \sum_{l=1}^{L(\mathcal{S})-1} (\delta(C_{\mathcal{S}}(l), \mathcal{C}_i) \cdot \delta(C_{\mathcal{S}}(l+1), \mathcal{C}_j) + \delta(C_{\mathcal{S}}(l), \mathcal{C}_j) \cdot \delta(C_{\mathcal{S}}(l+1), \mathcal{C}_i)$$

where $C_{\mathcal{S}}(l)$ is the cluster of the l^{th} segment of segmentation $\mathcal{S}$ and $L(\mathcal{S})$ its total number of segments.

Table 1: *The number of speakers in each file.*

File	1	2	3	4
Nb of speakers	7	13	15	20

6.3 The "Remembering" Option

Between each realized merge there is a new segmentation and the clusters are accordingly retrained (steps 2 and 3 in paragraph 5). Thus, the cluster change between two "rounds". However, it was noticed that in fact the clusters don't change that much, and that one can take advantage of the results of the previous round of merging scores. If a pair of clusters did not satisfy the merging criterion at the previous iteration it is likely that it will not satisfy it at the current iteration either. Therefore, the unsuccessfully merged pairs of previous iterations are placed at the end of the sorted list, and are only tried after the mergings of all pairs have been tried.

6.4 Learning The Segment Length

The segments of some clusters may tend to be longer than the segments of other clusters. This can reflect the fact that some people like to talk more than other, or that people have a different role in a conversation. In an interview, for example, the interviewed will be likely to talk for longer periods than the interviewer. One possible way to account for this, is to have a flexible self-transition a_i in our model (see figure 2). Those values are learned in the process of the segmentation. At each retraining they are re-estimated according to the following formula:

$$a_i = \frac{\mid D_{\mathcal{S}}(\mathcal{C}_i) \mid - S_{\mathcal{S}}(\mathcal{C}_i) \cdot L}{\mid D_{\mathcal{S}}(\mathcal{C}_i) \mid - S_{\mathcal{S}}(\mathcal{C}_i) \cdot (L-1)} \tag{22}$$

where $\mid D_{\mathcal{S}}(\mathcal{C}_i) \mid$ is the number of feature vectors (frames) belonging to cluster $\mathcal{C}_i$, $S_{\mathcal{S}}(\mathcal{C}_i)$ is the number of its segments according to segmentation $\mathcal{S}$ and L is the minimum duration. This formula is simply the number of time the system remained in the last state of cluster $\mathcal{C}_i$ divided by the number of times it was in that state.

It is interesting to note that by adding this parameter, the number of parameters stays truly constant, because at each merge one a_i disappears, which compensate for the added degree of freedom mentioned in section 4.5.

7 Experiments

7.1 Description

The experiments were done using the same four files as in [4] and [1] in order to have comparable results. Each of those files is a one half hour audio file from Broadcast News. The number of speakers appearing in each file is shown in table 1. The first file also contains a large amount of non-speech data, like music or hand clapping. Eight of the speakers of the second file and two of those of the fourth file talk over

the phone, which filters their acoustic features.
The features are 12 LPCC, for all experiments.

7.2 Evaluation

The same evaluation criteria were used as in [1]. Those criteria are the average cluster and speaker purity proposed in [5]. The average cluster purity (acp) is the probability that given two frames from the same cluster they also belong to the same speaker. Thus, if $S(\boldsymbol{x}^t)$ is the speaker having emitted $\boldsymbol{x}^t$, i.e the speaker talking at time t, $C_{\mathcal{S}}(\boldsymbol{x}^t)$ is the cluster in which x^t is cast by segmentation $\mathcal{S}$, W is the number of speakers, N is the number of cluster, $n_{i,j}$ is the number of feature vectors emitted by speaker j and cast into cluster i then

$$\begin{aligned} acp &= P(S(\boldsymbol{x}^t) = S(\boldsymbol{x}^u) \mid C_{\mathcal{S}}(\boldsymbol{x}^t) = C_{\mathcal{S}}(\boldsymbol{x}^u)) && (23) \\ &= \sum_{j=1}^{W}\sum_{i=1}^{N} P(S(\boldsymbol{x}^t) = i, S(\boldsymbol{x}^u) = i \mid C_{\mathcal{S}}(\boldsymbol{x}^t) = j, C_{\mathcal{S}}(\boldsymbol{x}^u) = j) && (24) \\ &= \sum_{j=1}^{W}\sum_{i=1}^{N} (\frac{n_{i,j}}{n_{\cdot,j}})^2 && (25) \end{aligned}$$

where $n_{\cdot,j} = \sum_i n_{i,j}$ is the number of feature vectors emitted by speaker j.
This measure punishes under-clustering, because in that case many speakers will be attributed the same cluster and the $n_{i,j}$'s will be small compared with $n_{\cdot,j}$ and the sum of the squares is smaller that the square of the sum for positive values. Similarly the average speaker purity *asp* is defined as the probability that two frames belonging to the same speaker will be assigned to the same cluster, that is

$$\begin{aligned} asp &= P(C_{\mathcal{S}}(\boldsymbol{x}^t) = C_{\mathcal{S}}(\boldsymbol{x}^u)) \mid S(\boldsymbol{x}^t) = S(\boldsymbol{x}^u) && (26) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{W} P(C_{\mathcal{S}}(\boldsymbol{x}^t) = j, C_{\mathcal{S}}(\boldsymbol{x}^u) = j \mid S(\boldsymbol{x}^t) = i, S(\boldsymbol{x}^u) = i) && (27) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{W} (\frac{n_{i,j}}{n_{i,\cdot}})^2 && (28) \end{aligned}$$

where $n_{i,\cdot} = \sum_j n_{i,j}$ is the number of feature vectors belonging to cluster i.
This penalizes over-clustering, because in that case many clusters will be attributed to the same speaker. In order to have one single number one defines k as the geometrical average the cluster and speaker purity.

$$k = \sqrt{acp \cdot asp} \qquad (29)$$

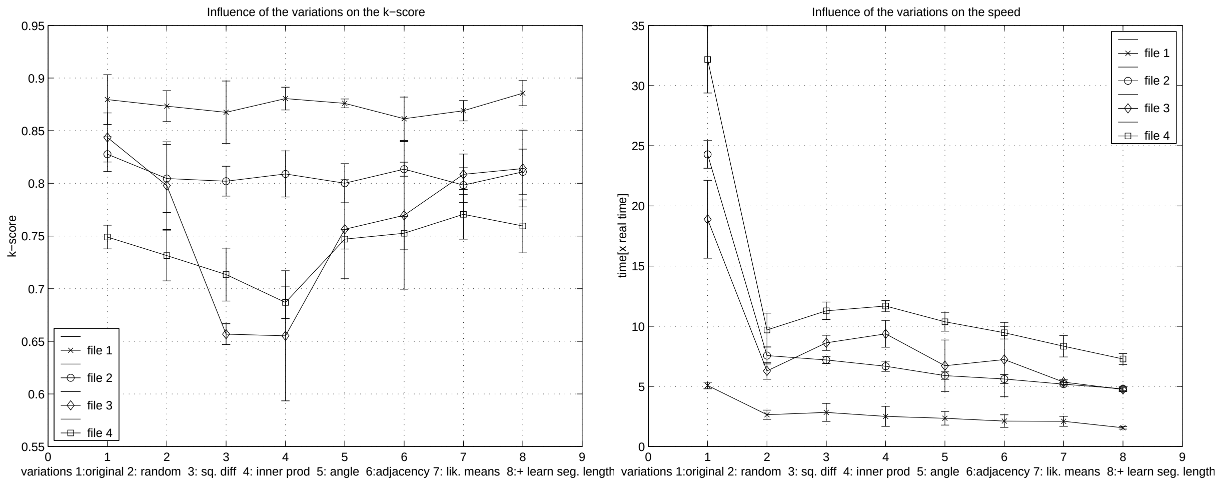
7.3 Results

On the subsequent plots showing the results, the error bars represent the standard deviation. All values are an average over 5 experiments with different initial K-means clustering. All experiments where run on the same computer, a 1024 megahertz sunBlade-1000.

Table 2: *The default parameters for the speaker clustering*

window size[ms]	min.duration[s] L	nb. clusters K	nb. Gaussians/cluster M
25	2	3 x nb. speakers	5

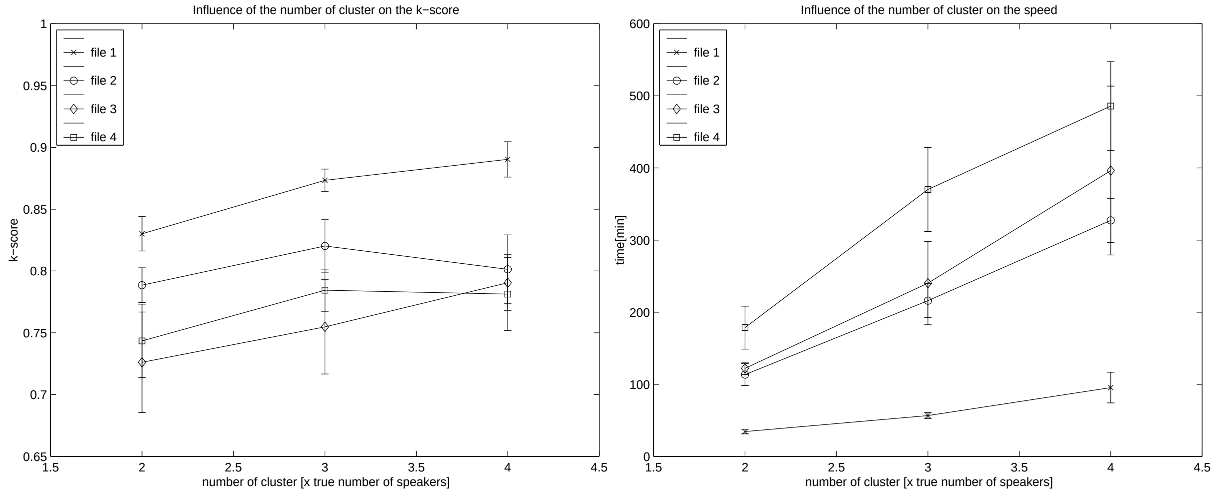
Figure 4: *The effect of the different variations on the efficiency of the algorithm. On the x-axis are the variations (1:"Original Version" , 2: "Pick First", 3: "Volume Of The Squared Difference", 4: "Inner Product Of The Pdfs", 5: "Angle Between The Pdfs", 6:"Adjacency Of The Segment", 7:"Likelihood Of The Means" 8: "Likelihood Of The Means + Learn Segment Length"). The versions 3 to 8 make use of the "Remembering" option. On the y-axis is the k-score (left plot) and the time (right plot). The lines link the results for the same file.*



7.3.1 Influence Of The Algorithm's Variations

All experiments are done using the default parameters of table 2. The aim of those experiments was to see the effects of the variations presented in section 6 on the efficiency of the algorithm. The results are presented in figure 4. The original version (for value 7 on the horizontal axis of the plots) is much slower than the "Pick First" version (value 2 on the horizontal axis) especially if there are a large number of clusters, and the k-score doesn't change significantly. So it does not seem necessary to try all possible merges and select the best one, it is much faster to perform the first acceptable merge that is found.
One can also notice that the only similarity measure that consistently brings an improvement to the speed of the algorithm is the "likelihood of the means" measure (number 7 on the x-axis). The other measures are either almost consistently worth than random like the "volume of the squared difference of the pdfs" (number 3) and the "inner product of the pdfs" (number 4) or are almost the same as random picking (number 2), like the "adjacency" (number 6) and "angle between the pdfs" (number 5). Moreover it seems that the two best similarity measures ("likelihood of the means" and "adjacency") are the measures which are the more heuristic, and which do not integrate some measure over the feature space. The other ones brought a significant decrease in the quality of the clustering for some files. This is

Figure 5: *The influence of the initial number of clusters on the results. On the x-axis is represented the initial number of clusters as a multiple of the true number of speakers. On the left graph the y-axis represents the k-score (section 7.2) and right graph the time for the algorithm to run in minutes.*



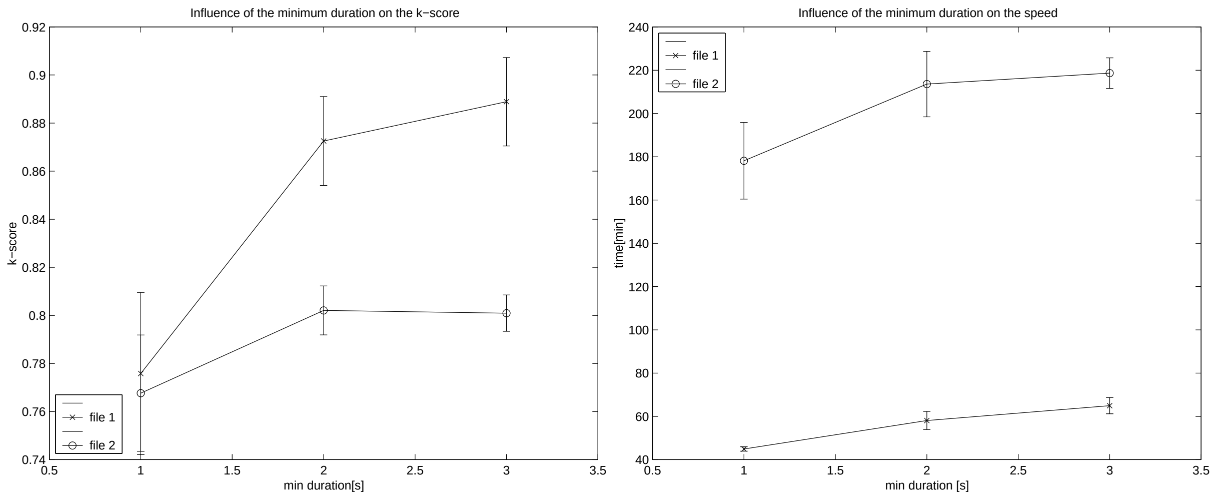
probably due to the fact that there is a discrepancy between the Gaussian mixtures and the actual distribution of the data.
The use of the "Learn Segment Length" option brings an improvement to the speed of the algorithm to all files and an improvement to the clustering score for most of the files. So the modeling of the segment length makes the model a better representation the reality. How this translates into a faster algorithm is not very clear, but could be attributed to the fact that in a better model, the similarity measure is more efficient. Another explanation is that, without this option, since one parameter is added each time two clusters are merged (as one constraint on the weights disappears), the likelihood is more likely to rise. More merges are therefore performed without this option, and the algorithm thus takes more time.
In summary, the modification brought to the algorithm made it from two to four times faster, no loss in the quality of the clustering.

7.3.2 Influence Of The Initial Parameters

All experiments are done using the "adjacency" version except the window size experiments that used the "likelihood of the means" similarity measure. The default parameters are the following ones listed in table 2.

The Initial Number Of Clusters The influence of the initial number of clusters is shown on figure 5. The left graph plots the k-score according to the initial number of clusters (as a multiple of the true number of speakers). One can see (on the left plot) a significant increase when going from twice the true number of speakers to three times the numbers of speakers, but no consistent increase when going from three times to four times the true number of speakers. For half of the files, having more initial clusters increases the quality of the clustering whereas for the other half, it does not. Of course the higher the number of clusters the slower

Figure 6: *The influence of minimum duration on the results. The on the x-axis is represented the minimum duration in seconds. On the left graph the y-axis represents the k-score (section 7.2) and right graph the time for the algorithm to run in minutes.*



the algorithm.

The Minimum Duration The influence of the minimum duration is shown in figure 6. Again there is a consistent increase in the k-score when going from one second minimum duration to two seconds, but not when going from two to three seconds. The choice of the minimum duration is the result of a trade-off between the granularity of the segmentation and the amount of data needed to capture the characteristics of a speaker. Since the needed granularity depends strongly on the file, (if speakers tend to speak for a long time or not) it is not surprising that the optimal minimum duration is different for each file. However, it seems that one can capture enough of the speaker characteristics in two seconds to have good enough results.

The Initial Number Of Gaussian Mixture Components The influence of the initial number of Gaussian mixture components is shown on figure 7. One can see that it is necessary to have enough of them, in order to capture the characteristics of each speaker, lest different speakers will be merged in the same cluster. The quality of the clustering for file 3 sees a very significant increase when having a large number of Gaussians. This suggests that with five Gaussians per cluster, there is not enough parameters to properly model the speech signal of this file. Having a lot of mixture does not affect the quality of the clustering very much, although it makes the algorithm slower. This can be explained by the fact that the risk of over-fitting is not very significant since the training and testing data is the same. However it is sound to assume that if there are really to many Gaussians, the modeling of the clusters derived from a segmentation will be nearly perfect and thus the likelihood could never be risen by any merge, which would stop the algorithm. This would obviously result in an over-clustering of the data.

Figure 7: *The effect of the initial number of Gaussian mixture component per cluster on the efficiency of the algorithm. On the x-axis is the initial number of Gaussian component in each cluster. On the y-axis is the k-score (left plot) and the time (right plot). The lines link the results for the same file.*

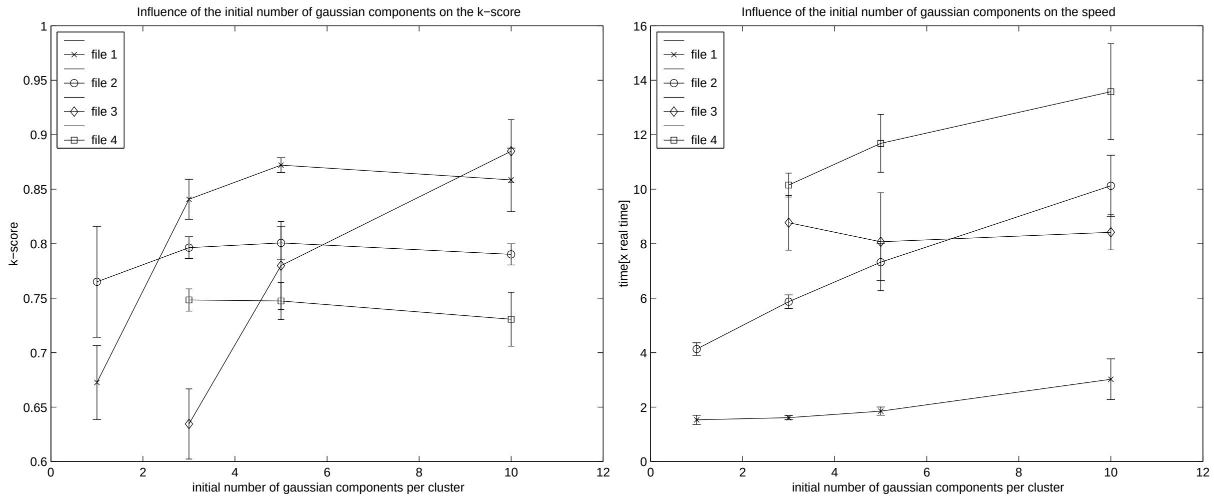
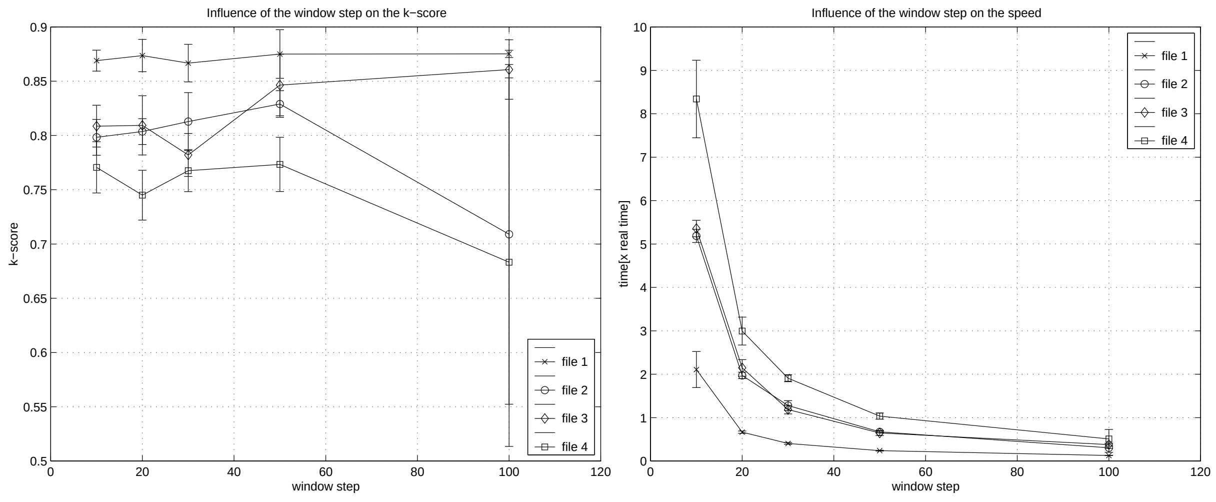


Figure 8: *The effect of the window analysis size and step on the efficiency of the algorithm. On the x-axis is the window step. The window size is always* $2.5 \cdot windowStep$*. On the y-axis is the k-score (left plot) and the time (right plot). The lines link the results for the same file.*



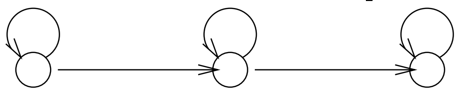
The Window Size Experiments were made to increase the window analysis size, and step size, keeping their ratio constant at 2.5. The idea is that if the step size is bigger, one will have less feature vectors for the same amount of speech. Since the Viterbi algorithm and the EM re-estimation algorithm are both linear with the number of feature vectors, reducing this number will increase the speed of the algorithm. The results presented in figure 8 show that this is in fact the case. In fact, for the fourth file, having the window step increased by five to 50 ms and the window analysis size increased to 125 ms, makes the algorithm about ten times faster, reaching the runtime threshold. Moreover, one sees that there is no loss in the quality of the clustering, on the contrary. This can be explained by the fact that the characteristics that are specific of a speaker are long-term characteristics and that a high temporal resolution is therefore not so important. Those long-term characteristics are best captured by extracting the LPC coefficients over large windows. A large window step, however, limits the granularity of the segmentation. It is interesting to notice that a very large window size of 250 ms yields a sharp decrease in the quality of the clustering for those file that have speakers talking through the phone.

8 Conclusion On The Speaker Clustering

The results presented above show that it was possible to dramatically improve the speed of the algorithm by not picking the best possible merge at each iteration and by computing the LPC coefficients over larger analysis windows and increasing the window step size (i.e. decreasing the frame rate). A further improvement was made possible by using a fast similarity measure to determine the order in which to try the mergings and by modeling the segment length for each cluster. The running time of the algorithm was thus brought (for file 4) from 33 times to 1 time real time with a slight increase in the quality of the clustering. Further improvements in the speed could probably be reached by reducing the amount of retraining at each iteration. As mentioned before, the models of the cluster do not change that much between each iteration, especially at the end of the procedure, when the clusters are fairly well trained. Considerable time is therefore probably waisted on training GMMs that are already trained. It has also been shown that the algorithm is very stable to the initial parameters. The quality of the resulting clustering is little affected by variations of those parameters. One just has to make sure that the the model is complex enough (contains enough Gaussian components) and that the minimum duration is big enough to capture the properties of the speaker and that one starts with enough clusters. And once those basic requirements are met, there is no big loss nor gain in terms of quality of the clustering by changing those parameters. The effect of those changes is merely on the speed of the algorithm. This can be explained by the fact that there is no distinction between training and testing, since the training data is the same as the testing data. Therefore, the risk of over-fitting is very limited.

Although no experiment was made in this work to compare this algorithm with other speaker clustering algorithms, this algorithm was (unofficially) submitted to the NIST dry run evaluation in January 2003. It got an honorable third place out of the seven competitors, despite the fact that it was used in its original version,

Figure 9: *A standard three state phone model.*



without any of the optimizations presented in this work, and without the optimized parameters. This indicates that the algorithm compares well with other state-of-the-art speaker clustering algorithms.

Part II

Automatic Subwords Unit Clustering

9 Introduction

In the second part of this work, another application of this algorithm will be presented, namely for automatic subword unit clustering. Firstly, the standard (phone-based) asynchronous speech recognition system will be briefly presented, and we will show where in such a system subword unit clustering could be used. Then the motivations for using subword unit clustering will be explained.
Two systems using subword unit clustering were experimented, the first one being a simplification of the second one. Those two systems, the models underlying them and their experiments will then be presented, followed by a conclusion, and an idea for future work on this topic.

10 Asynchronous Speech Recognition (ASR)

10.1 Phone-Based ASR

A standard ASR system is made of four main components, the feature extraction, the acoustic model, the dictionary and the language model.
The first component, the feature extraction, takes the sound wave as input and produces a sequence of feature vectors as output. The aim of feature extraction is to produce vectors that contain valuable information about the acoustics of the sound wave, i.e. about what sound was produced.
The second component, the acoustic model, models the generation of those feature vectors by the speaker. Nowadays, the standard acoustic model is based on phones. For a given language, a set of phones is designed by linguists, and each of those phones is represented as a 3 or 5 states HMM as shown on figure 9. The states of this HMM emit the feature vectors according to a multi-Gaussian pdf. This set of HMMs (one for each phone) constitutes the acoustic model.
The third component, the dictionary or word model models the words. In the stan-

dard word model, each word of this language is expressed as a sequence of phones (although some models support more than one sequence for each word). Thus, a given word is represented as the concatenation of the HMMs representing its sequence of phones.
The last component is the language model and models the sequence of words and is generally expressed as a Discrete Markov Chain (a HMM without emission probabilities where the states, the words in our case, are directly observed).

10.2 Motivations For Data-Driven Subword Units-Based ASR

It can be noticed that the system described above heavily relies on the work of linguists who design the phone set and and write the dictionary, i.e decide for each word, what its phone sequence is (or are). This is a serious drawback of the system because such linguistic knowledge is not always available, and one does not know to what extend this knowledge is helpful or right since it is not derived using the same statistical framework that is used for the rest of the system. Rather it is derived using human speech processing which is quite different than computer speech processing. Moreover the quality of this knowledge can be doubted when it comes to conversational speech, which is very different than "right" speech. Furthermore, the history of artificial intelligence shows that, in general, statistical systems trained on a lot of data produce better results than by knowledge-based systems. It would therefore be useful and interesting to build a system that does not use any linguistic knowledge. Since the feature extraction and the language model make no use of linguistic knowledge, those components can be kept as they are. The only components that would need to be changed are the acoustic model and the dictionary. In the following sections an acoustic model and a word model that make no use of linguistic knowledge are proposed in two variations.

11 General Framework

11.1 Core Ideas

The core ideas underlying the two suggested models are the following ones:

1. Instead of using a linguistically defined set of phones, a set of phones (or subword units) will be automatically derived from the data, using the algorithm described in the first part of this work.

2. Each word will be represented as a HMM, where the states represent the position within the word, and emit subword units according to a discrete probability density function.

The advantage of the first idea is that it provides a set of data-driven subword units that can be of different complexity, of different lengths (if the option mentioned in section 6.4 is used), and the size of this set is set automatically. However, there is a fundamental difference between the speaker clustering and the subword unit clustering. Whereas in the speaker clustering there is no distinction between the

training and testing data, this distinction is present in the subword unit clustering since the subword units are derived from a different data than the one on which the recognition is performed. This means that the risk of over-fitting is more crucial in the subword unit clustering and that the performances will likely to be more affected by the choice of the parameters, unlike the speaker clustering whose performances are pretty stable to parameter variations.
Different attempts to perform subword unit-based speech recognition, such as [6], face the problem of how to derive and optimize a subword unit-based dictionary. The advantage of the second idea is that it nicely integrates the dictionary into a well known statistical framework, which is very flexible and allows a very big number of different "pronunciations". One challenge is to define the word HMM topology to be on one hand flexible enough to account for the variations in the pronunciations and acoustic conditions and on the other hand stable enough to keep its discriminative power. A solution to this specific issue is presented in section 16.2.

11.2 Experiments

All the following models where tested on the Numbers95 task. Number95 is a database of numbers ranging from 0 to 100. The training set contains 3233 utterances, and the (disjoint) testing set contains 1227 utterances. Each utterance contains about 4 numbers. It is clean speech, although some laughs, "um" and "uh" occur sometimes. There are 30 words, including the silence word. It is worth noting that no use of the word boundaries was made in any of the experiments, that only the utterances boundaries were known. Moreover, the most basic language model was used, one where all words have an equal probability at any given time. This choice was made in order to isolate the contribution of the acoustic model only, since it is this component that is being studied. The attempt in those experiments was not to compete with the actual state-of-the-art systems that have undergone years of fine-tuning and contain many additional features, but rather to examine the feasibility of the proposed methods in a very simple framework, bearing in mind the simplicity of the experimental systems.

12 A Set Of Subword Units

Before going into the subword unit-based recognition systems, it is worth having a look at a set of subword units and try to understand what the subword unit models are learning and if they seem suitable for the purpose of speech recognition. In order to do this, the hand labeling of phones for the Numbers95 dataset was used. Figure 10 shows the confusion matrices of phones and subword units. On the plot on the left, the distribution of phones among subword units is shown (the rows are normalized). The intensity (darkness) of the squares in position (i, j) indicates the percentage of frames of subword unit i belonging to phone j. The matrix on the right shows the distribution of subword units among phones (the columns are normalized).
Those two matrices are pretty sparse, which means that their entropy is relatively low, i.e that the subword units and the phones contain a lot of information about each other. This is an encouraging fact, since it suggests that subword units contain

Figure 10: *The distribution of phones among subword units (left) and the distribution of subword units among phones (right). The rows represent the subword units and the column represent the phones. All values are between 0 (white) and 1 (black).*

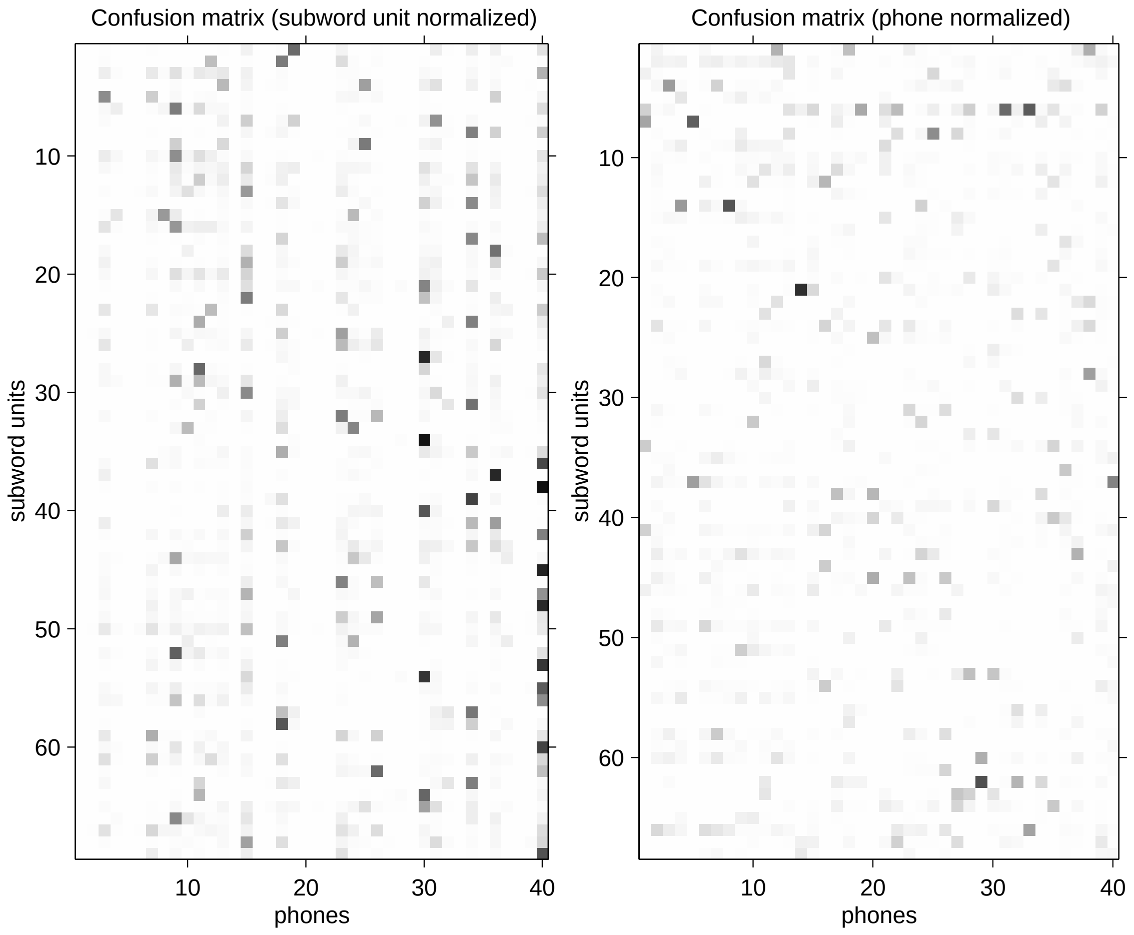
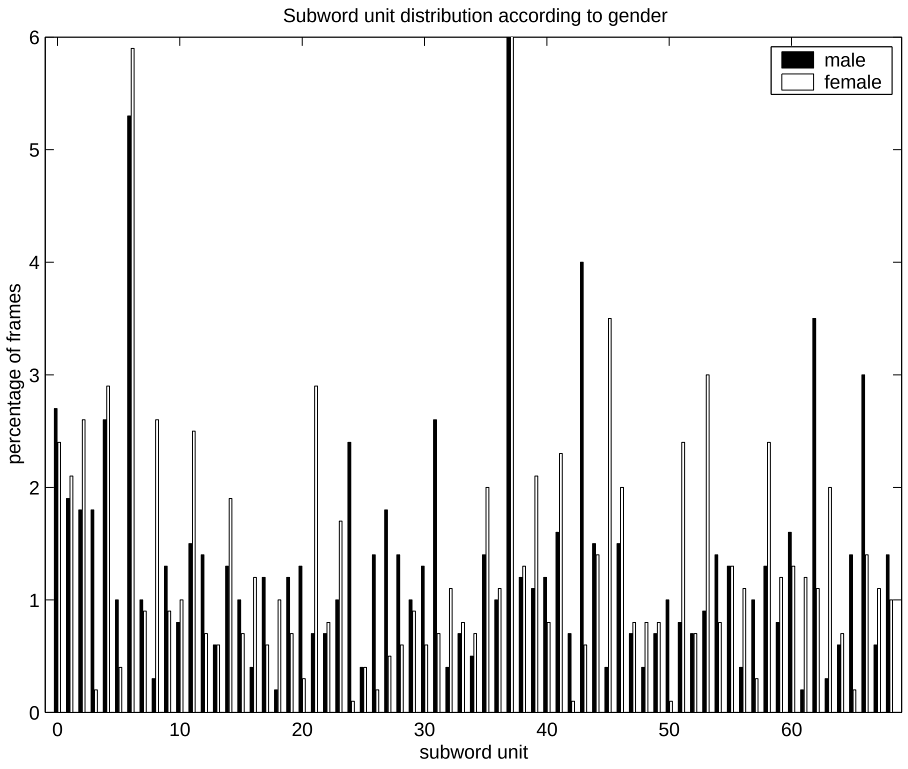


Figure 11: *The distribution of subword units among males (black) and female (white)*



a fair amount of phonetic information. A lot of subword units consist mainly of one, two or three phones, which does not come as a surprise since the minimum duration of 100 milliseconds is a bit higher than the mean duration of a phone. One can also notice that some subword units consist mainly of silence (the last column), and that different subword units consist mainly of the same phone, as indicated by columns containing a lot of black squares on the left matrix. The matrix on the right shows fewer dark squares, which means that phones are usually distributed over many subword units. This suggests that subword units are in way finer grained than phone, which is consistent with the fact that there are more numerous than phones.

Since many subword units express the same phone, especially for vowels, it is sound to assume that the subword unit models learn other characteristics than just the phonetic content. Those characteristics can be speaker dependent, such as the pitch, and through it the gender. In order to test this hypothesis, a the distribution of subword units according to gender was computed. The result is plotted on figure 11. One can see on that plot that men have a different distribution of subword units than women. Since the distribution of phones among men and women are very similar, this shows that the subword units models also catch things that are not the phonetic content. Except the big silence subword unit 37, almost all most significant subword units are gender dependent. Since the subword units are optimizing the likelihood of the data regardless of the transcription, it does not come as very surprising that they capture other properties than just the phonetic content. However, a fair amount of them do capture phonetic characteristics as

Table 3: *Examples of some obviously meaningful subword units. The "percentage" row indicates the percentage of occurrences of the subword unit that overlapped with this phone (or sequence of phones).*

subword unit	4	6	7	8	21
phone(s)	"two"	"wa", "an"	"ev","ayv"	"sih","seh"	"ayn"
percentage	85	49	72	60	36
subword unit	32	37	43	53	57
phone(s)	"zih"	sil	"sih","seh","zih"	"nay"	"owr", "aor"
percentage	73	99.8	66	69	52

can be seen on the table 3, which shows examples of some obviously meaningful subword units. The results presented in this table suggest that a good portion of the subword units capture common sequences of phones and transition between phones.

13 Experiment Evaluation

One challenge that arises when trying new models, is the question what to compare them with and to give some meaning to the results. We could compare our results to a standard basic HMM system that has as many parameters as the experimental model. But then, the standard system makes use of expert knowledge that has been refined over the years, and that the experimental system does not used. So it is to be expected that the experimental system will not produce as good results as the standard system. In order to nonetheless be able to evaluate the performance of our new models, a traditional phone-based system was designed, which has the same complexity as the experimental system and which could be taken as a baseline. This baseline system is described in the following section.

13.1 Baseline System

13.1.1 Description

The aim of this baseline system is to serve as a yardstick for measuring the performances of the experimental systems, i.e. to compare our word and acoustic models to the standard phone-based word and acoustic model. As said before, in order to isolate as much as possible the acoustic and word models and to keep the rest as simple as possible, the baseline system uses the most basic language model, where every word has the same probability. The acoustic model is made of the icsi56 phone set and each phone is represented as a three-state HMM. Each state in the HMM has a multi-Gaussian emission probability density function. The number of Gaussian components was set so that the total number of Gaussian components in the acoustic model would be (roughly) the same in the baseline system and and the experimental system. Since only 26 (including silence) of the 56 phones of the icsi56 phone set appear in the Numbers95 data, each state was therefore assigned $N_g/(26 \cdot 3)$ Gaussian components where N_g is the total number of Gaussian in the experimental system. Each word has one single pronunciation, which is the most

likely pronunciation of the Numbers95-icsi56 dictionary.
The baseline system was implemented using the HMM Toolkit (HTK) developed by [7].

13.1.2 Results

The results are shown on the following table:

Nb of Gaussians		WER
total	per state	
540	6	23.5

A word error rate of 23.5 percent on the recognition of the testing set was obtained using the baseline system. The word error rate (WER) is computed as follow:

$$WER = \frac{N_{deletions} + N_{insertions} + N_{substitutions}}{N_{total}} \tag{30}$$

where N_{xxx} means number of "xxx". This result is worse than what state-of-the-art system can achieve on this task (10 to 12 percent without a language model) but it is still a reasonable result taking into account the simplicity of the model and the small number of parameters. It can therefore serve as a basis for comparisons.

14 Model 1: Subword Units As Features

14.1 The Mathematical Model

This model is a very basic model based on the ideas exposed in section 11.1. The acoustic model is derived from the speaker clustering algorithm described in the first part and initialized according to table 4.
The word model was a HMM, whose topology is represented in figure 12. There is no shared parameters between two word models or within a word model. In order to reflect the fact that some words are longer than others, the number of states in the HMM depends on the word. As a rough approximation each word model $\mathcal{W}_i$ was assigned $Q_i = max(1, length(w_i) - 2)$ states, were Q_i is the number of states in model i, and $length(w_i)$ is the number of letters in the word w_i modeled by $\mathcal{W}_i$. This is probably not the optimal choice, especially when dealing with English, where the orthography of a word as little to do with its pronunciation. The "-2" term is due to the fact that it is better to have a smaller than optimal number of states rather than a bigger than optimal. Since there are no "shortcuts" the danger of having to many states is that one might have more states that subword units, which would give a word a zero probability, since there is no way to skip states in the considered topology.

14.2 Experiments

The experimental system was implemented using HTK. The minimum duration was set to 100 ms in order to have phone sized subword units, and standard mfcc features on a 25 ms window with deltas and acceleration every 10 ms were used.

Figure 12: *The model for one word. In this example, the model is a four states forward HMM. Each state (which represent the position within a word) has a different discrete probability density function over the set of subword units, as represented by the dotted arrows.*

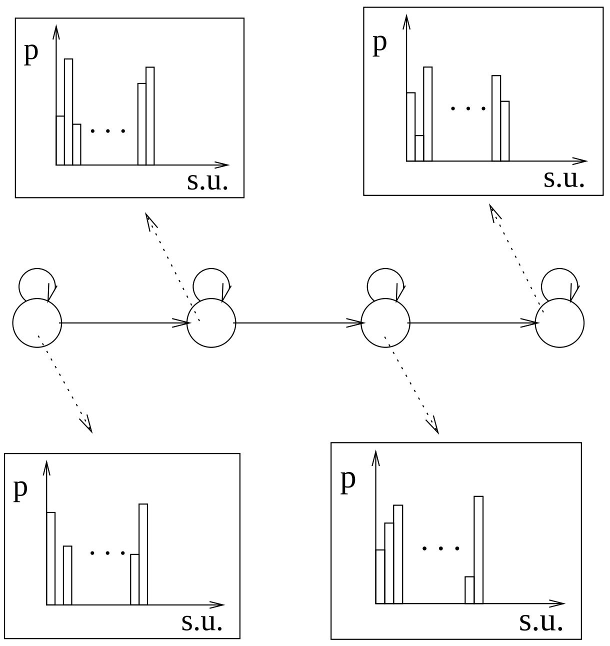


Table 4: *The initialization parameters for the subword unit clustering.*

features	min. duration	Nb initial clusters	Nb. initial Gaussians/cluster
12 mfcc+E+δ + $\delta\delta$	100 ms	150	3
12 mfcc+E+δ + $\delta\delta$	100 ms	150	6

14.2.1 The Training

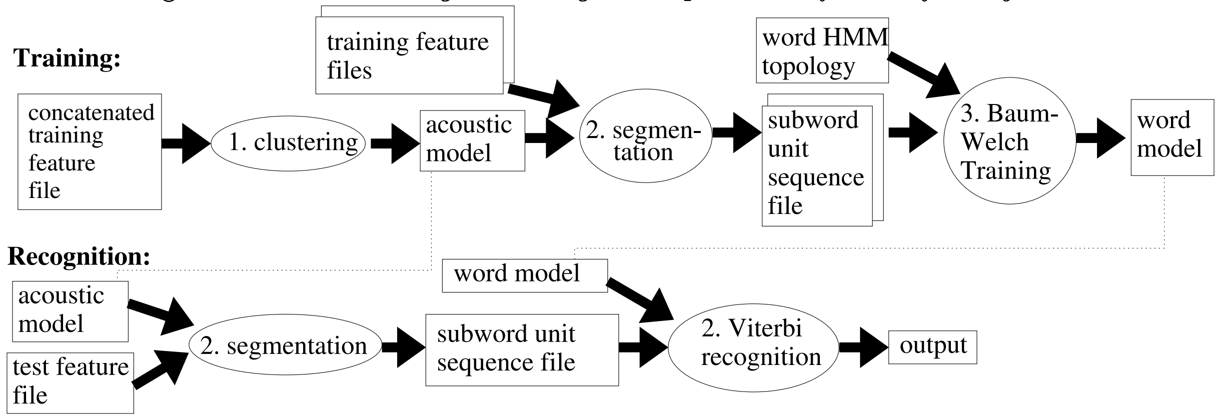
The training is done is three steps as shown on the top of figure 13.

1. The set of subword units is obtained using the clustering algorithm presented in the part I of this work. The initialization parameters were set as described in table 4. All training files (each containing one utterance) were concatenated in one single file in order to perform the clustering.

2. The model resulting from the clustering is then used to find, for each utterance, the most likely sequence of subword unit, using the Viterbi algorithm. Those subword units can be regarded as "new features" for our training data.

3. The word models (the HMMs) are then trained on those sequences of subword unit using the Baum-Welch algorithm. Those models are initialized in a "flat start" manner, where all emissions and all transitions are equal. The Baum-

Figure 13: *The training and recognition procedure for the first system.*



Welch training is performed by calling the HTK "HERest" tool four times.

14.2.2 The Recognition

The recognition is performed by firstly finding, for each testing utterance, the most likely sequence of subword units according to the model obtained during the training, using the Viterbi algorithm. Then, given this sequence of subword units, one finds the most likely sequence of words using again the Viterbi algorithm but this time on the subword units sequence. This procedure is schematically represented on the lower part of figure 13.

14.2.3 Results

The results obtained using the HTK scoring tool are presented in the following table.

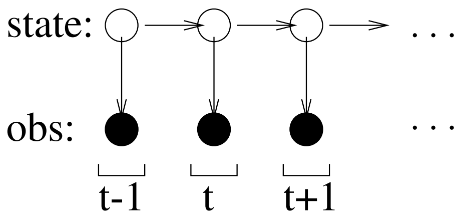
Nb of Gaussians	topology	WER
388	default	46.7
388	hand-designed	43.9
814	default	42.7
814	hand-designed	38.7

The default topology is the one described in section 14.1. For the hand-designed topology, each word is assigned a number of states that seemed more adequate, and transitions that skip one state were allowed.

14.3 Discussion

We see that we obtain a word error rate ranging from about 46% to 38%. This is up to twice as much as the one obtained with the baseline system with a bit more than two third of the number of parameters. Between more than half and less than two thirds of the words are recognized correctly. Although this result can first seem pretty low, it still tells us that the subword units contain information

Figure 14: *An HMM represented as a directed graphical model. The circles represent instances of variables and the arrows represent possible dependencies. The top row of nodes represent the state sequence, and the bottom row represent the observations. The latter are shaded to indicate that those variables are observed. The horizontal arrows represent the transition probabilities, whereas the vertical arrows represent the emission probabilities.*



that is valuable for the recognition, since a majority of the words can be recognized based only on the 1-most probable subword unit sequence. And this, despite the fact that those subword units were derived totally automatically and without any use of the transcription and not in a task oriented manner, as it is the case with regular phone training. Moreover, this result shows that the word model used seems to be adequate since it could be trained, and recognize most of the words on the sole basis of the 1-most probable subword unit sequence, which is a one dimensional feature. One can also observe a 3% decrease of the word error rate when using a better word topology and a further 5% when using more Gaussian components.
This model makes a clear separation between the deriving of the subword units and the word model. First the acoustic model is optimized, and then the word model is optimized. A hard decision is made on the sequence of subword units without regards to the word models. It is clear that this, while computationally efficient, is far from optimal in terms of likelihood maximization. A better model would learn the subword unit, taking the recognition task into account, and thus allow some interaction between the acoustic and the word models, and optimize both of them jointly. This is done in the second model.

15 Model 2: Joining Words And Acoustics

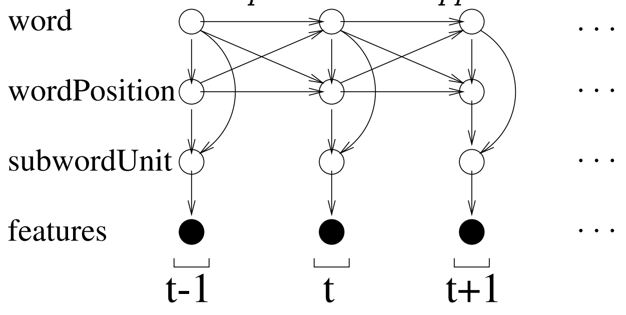
15.1 Theoretical Framework

As explained in section 14.3, the training and testing of the first model is not optimal in terms of likelihood maximization. The proper training and testing are derived using graphical models. In this section, a very brief introduction to graphical models is presented, which should allow the reader unfamiliar with the topic to follow the discussion. A more thorough review of this subject can be found in [8].

15.1.1 Directed Graphical Models

A directed graphical model is an acyclic graph, where each node represent a variable, and arrows between nodes represent possible dependencies between variables.

Figure 15: *The (simplified) directed graphical model representing a speech sequence. At the top level are the discrete "word" variables. Below are the discrete "wordPostion" variables, which represent the states of the word HMM shown in figure 12. Those hidden variables emit the hidden "subwordUnit" variables, which in turn emit the observed "features" according to a multi-Gaussian pdf. The real model has to take the minimum duration constraint into account and is presented in appendix B.*



Variables can be observed (in which case they are shaded) or hidden. A representation of an HMM as a directed graphical model is shown in figure 14.
Two very useful algorithms for graphical models are the Expectation-Maximization (EM) algorithm and the Max-Sum algorithm. The EM algorithm (of which the Baum-Welch algorithm is a special case for HMMs) optimizes the parameters of a graphical model with regard to the likelihood of the observed variables, that is it finds

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \, l(\theta, x), \tag{31}$$

where θ is the set of parameters of the graphical model, x is the set of all observed variables, and $l(\theta, x)$ is the log-likelihood of the observed data given θ, i.e.

$$l(\theta, x) = \log p(x \mid \theta) = \log \sum_z p(x, z \mid \theta), \tag{32}$$

where z is the set of all hidden variables.
The Max-Sum algorithm finds the most probable configuration of the hidden variable, given the observed variables and the model parameters, that is it finds

$$\hat{z} = \underset{z}{\operatorname{argmax}} \, p(z \mid x, \theta). \tag{33}$$

15.1.2 The Mathematical Model

The word model used our speech recognition system is basically the same as the one used in section 14.1 but the models of the subword units are integrated into it. The result is a kind of two level HMM, as represented in (simplified) figure 15. At the top level is the sequence of word variables. Those variables indicate to which word the corresponding feature vector belongs to, that is in which HMM is the system at time t. At a lower level are the word position variables. They indicate what state of the HMM produces the current subword unit that produces the feature vector. So they tell if the system is in the beginning, the middle or the end of a word since

this is what the states model. One level lower are the subword unit variables. They tell what subword unit is currently uttered, what is acoustically pronounced. On the lowest level are the observed feature vectors.
In order to fully describe the model, one has to explain the dependencies between the various variables, which are represented by arrows in figure 15. First, a word variable depends on the previous word and the previous word position, as a new word can only appear if the preceding word has been completed, i.e. if the previous word position was the last of its word. The word position variables depend on the previous word and the word position (which indicate in what state of what HMM the system was at time $t-1$), and on the current word. The subword unit variable depend on the current word and the current word position which indicates by what state of what HMM the subword unit is generated. The feature vectors depend only on the current subword unit. Apart from the dependency between the feature vectors and the subword units, which is realized through a multi-Gaussian probability density function for each subword units, all the dependencies can be realized through conditional probability tables, where zeros are forced on some places to ensure the right word HMM topology and that each word must be completed from its starting state to its ending state.

As mentioned before, figure 15 shows a simplified version of the model, as the minimum subword unit duration constraint is not represented on it. The real graphical models used for training and testing are represented in figure 16 of appendix B.

15.2 Experiments

The experimental system was implemented using the Graphical Model Toolkit (GMTK) developed by [9].

15.2.1 The Training

The training was performed in two steps. Firstly, the set of subword units was derived by using the clustering algorithm described in part I, on the concatenated training set, as explained in the first step of section 14.2.1. In fact, the same set of subword units was used in order to facilitate the comparisons. Then the graphical model presented in appendix B was initialized with this set of subword units (i.e. of multi-Gaussian pdfs), and the word HMMs were initialized as in the previous experiment, i.e. each state with a uniform discrete pdf on the subword units, and all allowed transition probabilities were set equal. The number of Gaussian mixture components remained unchanged during the whole training. The training was performed using the "gmtkEMTrain" tool with a pruning beam width of 50 for efficiency purposes. Since the graphical model used is fairly complex, the GMTK software was excessively slow and it was materially not possible to train with a more adequate pruning beam width (typically 1000 or more). It took about five minutes to perform one EM iteration on each utterance on a 900 MHz sun station. This made it impossible to perform more than 3 EM iterations, which is hardly enough.

15.2.2 The Recognition

The recognition was performed using the "gmtkViterbi" tool. For the same reasons the same very high pruning was set. It also took about five minutes to decode each utterance.

15.2.3 Results

The results appear in the following table:

Nb of Gaussians	topology	WER
388	default	49.9

15.3 Discussion

Although this second system should theoretically be better than the first, the word error rate is 3% worse. This is probably due to the very high computational pruning applied when using GMTK. Because of the slowness of the software, the experiment could only be run once, without being able to try different initializations, or a higher number of EM iterations. Thus, this result should rather be taken as an upper bound for the word error rate. The first system with similar constraints (3 EM iterations and high pruning) would yield significantly poorer results, and it is therefore safe to assume that this system could perform better than the first system.

16 Improving The Model

16.1 Tying The Word Models

In most (if not all) languages there is an intermediate level between the words and the "sounds" (phones, syllables, subword units), like affixes, endings and so on. One way to account for this fact is to tie together states representing the same element of different word HMMs, i.e. to have states of different word HMMs share the same parameters. This would reduce the number of parameters in the model, probably allowing for more robust training, and would match the intuition that there is something common in the pronunciation of the ending "ation" in "optimization" and "pronunciation". Those matching could be done on a orthographic basis and would not need linguistic knowledge. One could just identify that two words having a group of, say, three or more consecutive letters in common, should share some parameters.
The numbers95 database is appropriate for such an experiment, as the"teen"'s are shared among many numbers, and many digits appear as digit (like six), in the teen's (sixteen) and the ty's (sixty).

16.1.1 Experiments

The tying of the parameters was experimented using the first model described in section 14.1. The experiments were performed with the parameters described in table 4, but the states (or group of states) modeling sequence of same consecutive three or more letters shared the same emission probability distribution over the set of subword units. Thus, the word fragment “thir”, “four”,”fif”,“six”, ”seven”, “eight”, “nine” and “teen” were each assigned a number of states (between one and three).

16.1.2 Results

The results are presented in the following table:

Tying	Nb of Gaussians	Topology	WER
no	388	hand-designed	43.9
yes	388	hand-designed	41.2
no	814	hand-designed	38.7
yes	814	hand-designed	38.8

One observes a 2.7% improvement of the word error rate for the model containing fewer parameters, but a very slight increase for the model with more parameters. This suggests that the tying of parameters only helps when there are not so many of them.

16.2 “Weighted States” HMMs

In the default word model used so far, each word has a minimal number of subword units, which is the number of states in its HMM since it is not allowed to skip states. This, with the minimum duration constraint, constitutes a drawback of this model, since it imposes *ipso facto* a minimum duration constraint on the words. Such a minimum duration on words might not be wishful, as it would prevent adaptation to fast speech, making it impossible even to train on such speech. Another drawback of this model is, that there is no clear way of choosing the number of state for each word. An easy thumb rule based on the number of letters in the word was proposed, but this is certainly not satisfying for serious applications. In the next paragraph, a model (called “weighted states HMM”) is proposed, which would offer a solution to those two problems. It is however beyond the scope of this work to implement and test this model, so its justification is largely based on intuition.

A “weighted state HMM” is a HMM with a particular topology. Here are its characteristics:

1. Each state q_i has a weight w_i.
2. The transition a_{ij} between states q_i and q_j is entirely determined by the set of weights $\{w_k\}$:

$$a_{ij} = \begin{cases} \frac{w_j}{\sum_{k=i}^{Q} w_k} & \text{if } i \leq j, \\ 0 & \text{otherwise} \end{cases} \qquad (34)$$

So this is a feed-forward HMM with self-loops, in which the probability to go to one state is always proportional to the weight of the arrival state. Thus if a state is less visited during training, its weight becomes lower and the probability to reach that state from whatever other state will lower as well. Eventually, if a state is never visited, it’s weight will reach zero which means that the state disappears

as all transitions to that state will be zero. So the weight would represent the importance of a state in the pronunciation of a word, and one might think that states representing the position of accented syllables in a word would have higher weights than other states. So when people talk fast, they only pronounce some of the syllables of a word, usually the accented ones, and this would be equivalent to skipping states with low weights in the weighted states HMM. This model might also solve the problem of choosing the right number of states, as unnecessary states will just disappear when their weight reaches zero. One has to choose enough states, as states can only disappear but not emerge (although one could figure out a splitting mechanism for states too).
The training of such a mode could be derived from the EM algorithm.

17 Conclusion On The Subword Unit Clustering

The above described experiments show that is it possible to perform subword unit-based speech recognition that makes no use of linguistic knowledge. The common problem of deriving a subword unit-based dictionary has been solved in an elegant way, that nicely integrates the dictionary into the statistical framework. This "soft" dictionary allows the modeling of word portions by tying the parameters of word HMM states. But a mapping between phones and subword unit distributions should be found in order to be able to add totally new word to the dictionary without having to train the word model.
It is important to bear in mind that the optimizing criterion for the initialization of the subword unit, which is the likelihood of the data given the acoustic model (i.e. $P(\{\boldsymbol{x}^t\} \mid \mathcal{A})$ if $\mathcal{A}$ is the acoustic model), is different from the optimizing criterion for the EM training algorithm, which is the likelihood of the data given the acoustic model, the word model and the transcription (i.e. $P(\{\boldsymbol{x}^t\} \mid \mathcal{A}, \mathcal{W}, \mathcal{T})$ where $\mathcal{W}$ and $\mathcal{T}$ are the word model and the transcription respectively). The result of this, is that the set of subword units does not necessarily capture characteristics that are relevant for speech recognition as the results of section 12 show. Would one want to train a recognition system in the truly same framework as the speaker clustering system, one would have to retrain the whole system, that is the acoustic and the word model, at each iteration of the clustering algorithm and merge two subword units if (after retraining of $\mathcal{A}$ and $\mathcal{W}$ using the EM algorithm and somehow keeping the number of parameters constant) it raises $P(\{\boldsymbol{x}^t\} \mid \mathcal{A}, \mathcal{W}, \mathcal{T})$. This would imply many retrainings of an entire speech recognition system at each iteration, which seems prohibitive bearing in mind and the time needed for the training of one such system on a 33 word vocabulary.

If the recognition rate does not yet compare with actual state-of-the-art systems, one can still recognize at least 60% of the words and there is a lot of room for improvement. Beside a proper EM training on the graphical model (with reasonable pruning, and more iterations), one could probably find a better minimum duration constraint, a better initialization of the subword units (for example by taking the transcription and the word model into account), a better word HMM topology (as the weighted state HMM) and a better number of parameters. Thus subword unit-based speech recognition, and more particularly the statistical framework (the graphical model) presented here, certainly has the potential to achieve recognition rates that would compete with phone-based recognition systems.

18 General Conclusion

The data-driven clustering framework presented in this work is a very simple and powerful idea. It is based on performing agglomerative clustering while keeping the number of parameters constant and thus allows the use of a maximum likelihood criterion. But this algorithm can also be seen as an iterative model selection procedure. At each iteration, different models are compared and one of them is selected. This framework was first successfully applied to speaker clustering. It could also be applied to subword unit clustering. There are however, a fundamental differences between the speaker and the subword unit clustering.
The first difference is that for the speaker clustering there is no distinction between training and testing data, since the data serves for training and for testing. This makes the algorithm very little sensitive to the initialization of the parameters as the risk of over-fitting is less relevant. This is obviously not the case for the subword unit clustering, as those subword units will be used when performing recognition on different data than the one on which they were trained. This makes the algorithm more sensitive to the initialization parameters which should be more carefully chosen.
The second and crucial difference is that in the case of speaker clustering, one can retrain and select the entire model at each iteration. This is feasible thanks to the simplicity of the model used (a small HMM with a minimum duration constraint). For the subword unit clustering, on the other hand, the retraining and the selection is done only on one part of the model, namely the acoustic model, and independently of the rest of the model. This is may be good enough for a first initialization step, but it is certainly not the best way of getting subword units optimized for the speech recognition task. Such an optimal set of subword units could probably not be derived without using the entire model, that is the word and the acoustic models, as well as the transcription.
In summary, good results were obtained by naively using the speaker clustering algorithm, with different parameters for performing subword units. But those results could probably improve significantly by using this framework in a more intelligent manner, that would take into account the aim and the particularities of subword unit clustering, which are quite different from those of speaker clustering.

A Mathematical Proofs

A.1 Formula for the 2nd distance measure

Following is the proof of formula 15, as proposed by [3]. We are trying to compute $\int (p_i(x) - p_j(x))^2 dx$ where $p_i(x)$ and $p_j(x)$ are multi-Gaussians as described in equation 14. So we can write

$$\begin{aligned}
& \int (p_i(x) - p_j(x))^2 dx \\
= \;& \int \Big(\textstyle\sum_{m=1}^{M_i} c_{i,m} g_{i,m}(x) - \sum_{m=1}^{M_j} c_{j,m} g_{j,m}(x) \Big)^2 dx \\
= \;& \int \Big(\textstyle\sum_{m_1=1}^{M_i} \sum_{m_2=1}^{M_i} c_{i,m_1} c_{i,m_2} g_{i,m_1}(x) g_{i,m_2}(x) + \sum_{m_1=1}^{M_j} \sum_{m_2=1}^{M_j} c_{j,m_1} c_{j,m_2} g_{j,m_1}(x) g_{j,m_2}(x) \\
& -2 \textstyle\sum_{m_1=1}^{M_i} \sum_{m_2=1}^{M_j} c_{i,m_1} c_{j,m_2} g_{i,m_1}(x) g_{j,m_2}(x) \Big) dx \\
= \;& \int \Big(\textstyle\sum_{k,l \in \{i,j\}} \sum_{m_1=1}^{M_k} \sum_{m_2=1}^{M_l} (-1)^{\delta(k,l)} c_{k,m_1} c_{l,m_2} g_{k,m_1}(x) g_{l,m_2}(x) \Big) dx \\
= \;& \textstyle\sum_{k,l \in \{i,j\}} \sum_{m_1=1}^{M_k} \sum_{m_2=1}^{M_l} (-1)^{\delta(k,l)} c_{k,m_1} c_{l,m_2} \int g_{k,m_1}(x) g_{l,m_2}(x) dx
\end{aligned}$$

One must therefore compute the integral of the product of two Gaussians. It is known that such a product is also a Gaussian function:

$$\begin{aligned}
& g_1(x) \cdot g_2(x) \\
= \;& \tfrac{1}{(2\pi)^{d/2} |\Sigma_1|^{1/2}} \cdot \exp\Big\{ -\tfrac{1}{2} \Big((x-\mu_1)^T \Sigma_1^{-1} (x-\mu_1) \Big) \Big\} \cdot \\
& \tfrac{1}{(2\pi)^{d/2} |\Sigma_2|^{1/2}} \cdot \exp\Big\{ -\tfrac{1}{2} \Big((x-\mu_2)^T \Sigma_2^{-1} (x-\mu_2) \Big) \Big\} \\
= \;& \underbrace{\frac{1}{(2\pi)^d \cdot (\mid \Sigma_1 \mid \cdot \mid \Sigma_2 \mid)^{1/2}}}_{A} \cdot \exp\Big\{ -\tfrac{1}{2} \Big((x-\mu_1)^T \Sigma_1^{-1} (x-\mu_1) + (x-\mu_2)^T \Sigma_2^{-1} (x-\mu_2) \Big) \Big\} \\
= \;& A \cdot \exp \left\{ -\tfrac{1}{2} \Big(x^T \underbrace{\left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)}_{\Sigma_3^{-1}} x - 2x^T \underbrace{\left(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2\right)}_{\Sigma_3^{-1}\mu_3} + \mu_1^T \Sigma_1^{-1} \mu_1 + \mu_2^T \Sigma_2^{-1} \mu_2 \Big) \right\}
\end{aligned}$$

Since we are dealing with diagonal covariance matrices, the order of the matrix multiplication has no importance. So the product of two normalized Gaussians is a non-normalized Gaussian with covariance

$$\Sigma_3 = \left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)^{-1}$$

and mean

$$\mu_3 = \Sigma_3 \left(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2\right).$$

The (L_1) norm of this Gaussian can be computed as

$$\begin{aligned}
& c_3 \\
= \;& \tfrac{g_1(x) \cdot g_2(x)}{g_3(x)} \\
= \;& \frac{A \exp\Big\{ -\frac{1}{2} \Big(x^T \Sigma_3^{-1} x - 2x^T \Sigma_3^{-1} \mu_3 + \mu_1^T \Sigma_1^{-1} \mu_1 + \mu_2^T \Sigma_2^{-1} \mu_2 \Big) \Big\}}{(2\pi)^{-d/2} |\Sigma_3|^{-1/2} \exp\Big\{ -\frac{1}{2} \Big(x^T \Sigma_3^{-1} x - 2x^T \Sigma_3^{-1} \mu_3 + \mu_3^T \Sigma_3^{-1} \mu_3 \Big) \Big\}}
\end{aligned}$$

This turns out, in case of diagonal Gaussian, to be equal to the following:

$$\int g_{k,m_1}(x)g_{l,m_2}(x)dx = \frac{1}{\mid \Sigma_{k,m_1} + \Sigma_{l,m_2} \mid^{1/2}} \exp\{-\frac{1}{2}(\Sigma_{k,m_1} + \Sigma_{l,m_2})^{-1}(\mu_{k,m_1} - \mu_{l,m_2})^2\} \tag{35}$$

The formula is proved by inserting 35 into A.1.

B The GMTK Model

The real graphical model used for the training of the speech recognition system is shown in figure 16. This graphical model express the same structure as the simplified one represented on figure 15, but some variables are added in order to account for the minimum duration constraint, and to keep the whole coherent. The variables are presented in the following table 5.

For the decoding, the same graphical model was used, with the trained parameters, but the variables "wordCounter" and "skipSil", were suppressed since the role of those variables is to insert the knowledge derived from the transcription into the model. Since this knowledge is obviously not available when performing the recognition, those varaibles have to be removed.

The "emissConstraint" Variable If a given occurrence of a subword unit is more than twice the minimum duration long, it can be seen as one or as two occurrences of this subword unit. In order remove this ambiguity, it was decided to allow a subword unit to be split into two only if those (now) two subword units are emitted by different states. That is, to same subword units can only follow one another if there is a state transition between them. This rule is enforced by the observed binary "EmissConstraint" variable, which is true if the rule is observed. This variable is observed as being true for every frame, thus enforcing that rule.

Figure 16: *The graphical model used for training with GMTK. Auxiliary variables were added to account for the minimum duration constraint, and to ensure the coherence of the whole model. The dashed lines represent "switching parents"*

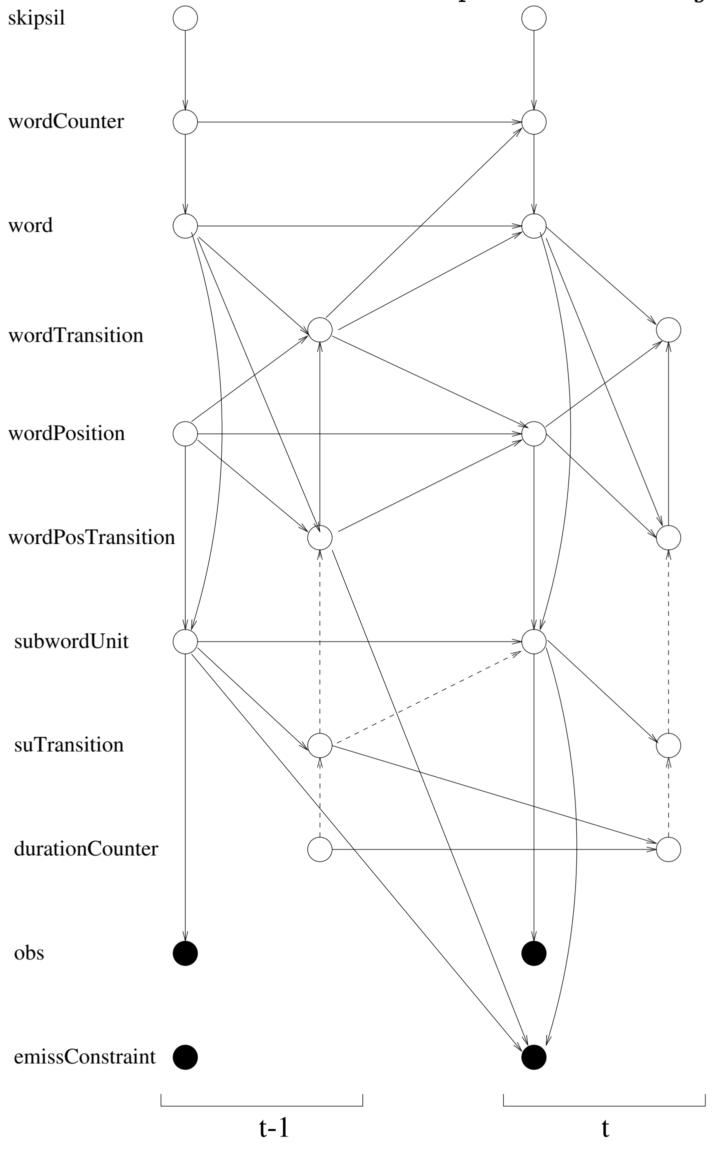


Table 5: *The list of variables used for the GMTK training. The* variableName(-1) *refers to the* variableName *of the previous frame (i.e. at time $t-1$).*

Variable	Function	Value
skipSil	skipped silence indicator	according to a conditional probability table
wordCounter	counts the word in an utterance	wordCounter(-1) if wordTransition = 0 wordCounter(-1)+1 if wordTransition=1 and skipSil=0 wordCounter(-1)+2 if wordTransition=1 and skipSil=1
word	the word is being said	matched from wordCounter for each utterance
wordTransition	if last frame of a word	1 if wordPosTransition = 1 and last wordPosition of this word, 0 otherwise
wordPosition	current state of the the current word HMM	wordPosition(-1) if wordPosTransition = 0 wordPosition(-1)+1 if wordTransition(-1) = 1 0 if wordTransition = 1
wordPos-Transition	if there is a state transition	0 if suTransition = 0, according to a probability table if suTransition = 1
subwordUnit	the current subword unit	subwordUnit(-1) if suTransition = 0, according to a conditionnal probability table otherwise
suTransition	if there is a subword unit transition	0 if durationCount ¿ min. duration, according to a probalility $(1 - a_{subwordUnit})$ otherwise
durationCounter	counts the subword unit duration	min. duration if durationCounter(-1) = min. duration and suTransition(-1) = 0, durationCounter(-1)+1 if suTransition(-1) = 0, 0 if suTransition(-1) = 1
obs	the feature vector	according to a multi-Gaussian pdf
emissConstraint	remove ambiguity	0 if subwordUnit(-1) = subwordUnit and wordPosTransition = 0, 1 otherwise

C Used Notation

$\Sigma_{i,m}$: covariance matrix of Gaussian function$g_{i,m}$;
$\delta(x, y)$: Kronecker's symbol, (1 if $x = y$, 0 otherwise);
$\mu_{i,m}$: mean of Gaussian function $g_{i,m}$;
θ: parameters of a model;
$\mathcal{A}$: $\mathcal{C}_i$: cluster i;
$\mathcal{M}_i$: model i;
$\mathcal{L}$: language model;
$\mathcal{S}$: segmentation;
$\mathcal{T}$: transcription;
$\mathcal{W}$: word model, or dictionary;
$\mathcal{W}_i$:model for word w_i;
A: transition matrix;
B_i: emission pdf of state q_i;
$C_{\mathcal{S}}(l)$: cluster of the l^{th} segment of segmentation $\mathcal{S}$;
$D_{\mathcal{S}}(\mathcal{C}_i)$: set of all feature vectors belonging to cluster $\mathcal{C}_i$ according to segmentation $\mathcal{S}$;
K: initial number of clusters when initializing the clustering algorithm;
L: minimum segment length (minimum duration);
N: number of clusters in the model;
N_g: total number of Gaussian components in the model;
$S(\boldsymbol{x^t})$: speaker emitting feature vector $\boldsymbol{x^t}$:
T: length of the time sequence;
W: true number of speakers in an audio file;
a_i: self-transition of last state of cluster i;
a_{ij}: transition probability from state q_i to q_j;
$c_{i,m}$: m^{th} weight of Gaussian mixturep_i;
d: dimension of the feature vectors;
$g_{i,m}$: m^{th} Gaussian of Gaussian mixturep_i;
k: speaker clustering evaluation criteria (see equation 29);
p_i: multi-Gaussian probability density function of cluster i;
q_i: state i of a HMM;
t: time index;
u: time index;
w_i: word i;
$\boldsymbol{x^t}$: feature vector emitted at time t;
acp: average cluster purity (see equation 23);
asp: average speaker purity (see equation 26);
$length(w_i)$: number of letters in word w_i;

References

[1] I. Lapidot J. Ajmera, H. Bourlard. Improved unknown-multiple speaker clustering using hmm. Technical Report IDIAP-RR 02-23, Institut Dalle Molle D'Intelligence Artificielle Perceptive (IDIAP), Martigny, Switzerland, September 2002.

[2] G. Schwartz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[3] Panu Somervuo. private conversation, January 2003.

[4] I. Lapidot J. Ajmera, H. Bourlard and I. McCowan. Unknown-multiple speaker clustering using hmm. In *International Conference on Spoken Language Processing*, 2002.

[5] M. Schmidt A. Solomonoff, A. Mielke and H Gish. Clustering speakers by their voices. In *IEEE International Conference on Accoustics, Speech and Signal Processing*, pages 757–760, 1998.

[6] B. Raj R. Singh and R.M. Stern. Automatic generation of subword units for speech recognition systems. *IEEE Transactions On Speech And Audio Processing*, 10(2):89–99, February 2002.

[7] Phil Woodland et al., 2001. http://htk.eng.cam.ac.uk/.

[8] Michel I. Jordan. An introduction to probabilistic graphical models. Lecture notes, not yet published.

[9] J. Bilmes and G. Zweig, 2002. http://ssli.ee.washington.edu/~bilmes/gmtk/.