Karlsruhe Institute of Technology

A Study of Distance Measures for Clustering Generalized Polyphones

Student Research Paper
by

Lucas Czech

at the
Institute for Anthropomatics,
Interactive Systems Labs (ISL),
Karlsruhe Institute of Technology (KIT),
Germany

First Reviewer:	Prof. Dr. Alexander Waibel
Second Reviewer:	Dr. Sebastian Stüker

June 15 – November 30, 2012

Zusammenfassung

In dieser Arbeit untersuchen wir die Auswirkungen von verschiedenen Distanzmaßen zwischen Gaussverteilungen und Gauss-Mixturen auf die Qualität eines Cluster-Algorithmus für Polyphone. Auf einem voll-kontinuierlichen System testen wir dazu die *Euklidische Distanz*, die *Kullback-Leibler-Divergenz*, die *Erweiterte Mahalanobis-Distanz* und die *Bhattacharyya-Distanz* mit einem divisiven Clustering.

Spracherkennung mit großem Vokabular und kontinuierlicher Sprache erfordet viele Trainingsdaten und sowohl im Training als auch in der eigentlichen Anwendung große Rechenkapazitäten. Daher werden an vielen Stellen Vereinfachungen und Annäherungen genutzt, um Leistung einzusparen. Eine solche Reduktion ist die Verwendung von generalisierten Polyphon-Klassen anstelle von einzelnen Polyphonen; um diese Klassen zu erstellen, wird ein Cluster-Algorithmus verwendet, dessen Aufgabe es ist, ähnliche Polyphone zusammenzufassen.

Ein beliebtes Distanzmaß, welches diese Ähnlichkeit misst, ist die sogenannte Entropie-Distanz. Sie wird direkt auf den Mixturgewichten der Gauss-Mixturen berechnet, ist daher auf semi-kontinuierlichen Modellen leicht zu berechnen und stellt auf diesen ein anschauliches und gut interpretierbares Maß dar.

Aufbauend auf der Entropie-Distanz als Bezugspunkt untersuchen wir die Euklidische Distanz, die Kullback-Leibler-Divergenz, die Erweiterte Mahalanobis-Distanz und die Bhattacharyya-Distanz hinsichtlich ihrer Diskriminierung von Polyphonen. Da diese Distanzen direkt auf den Gaussverteilungen (bzw. in unserer Anwendung ebenso auf Gaussmixturen) operieren, wird hierzu ein voll-kontinuierliches System verwendet.

Für jedes Distanzmaß trainieren wir anschließend ein System, welches den aus dem Clustering hervorgehenden Entscheidungsbaum nutzt. Dieses System wird auf unabhängigen Test-Daten in Form der resultierenden Wortfehlerraten evaluiert.

Abstract

In this work we analyze the effects of different distance measures between Gaussian distributions and Gaussian mixtures on the quality of a clustering algorithm for polyphones. Using a fully-continuous system, we test the *Euclidian distance*, the *Kullback–Leibler divergence*, the *Extended Mahalanobis distance* and the *Bhattacharyya distance* with a divisive clustering.

Speech recognition with large vocabulary and continuous speech necessitates much training data and huge computing capacities in the training as well as in the actual utilization. Therefore, often simplifications and approximations are used to save effort. One such reduction is the use of generalized polyphone classes instead of single polyphones; to create these classes, we use a clustering algorithm, whose task it is to combine similar polyphones.

A common distance measure, which measures these similarities, is the so–called entropy distance. It is directly calculated on the mixture weights of the Gaussian mixtures, thus it is easily computed and represents a descriptive and well interpretable measure.

Based on the entropy distance as benchmark, we analyze the Euclidian distance, the Kullback–Leibler divergence, the Extended Mahalanobis distance and the Bhattacharyya distance in regard to their discrimination of polyphones. As these distances directly operate on the Gaussians (respectively also on the Gaussian mixtures in our application), we use a fully-continuous system.

For each distance measure we then train a system, which uses the decision tree resulting from the clustering. This system is evaluated on independent test data in form of the resulting Word Error Rate.

Contents

1. Introduction

Computing power has increased almost exponentially over the last few decades and there is no end of this trend in sight. Almost all branches of research have profited by this, and still will – also, many achievements in science would be impossible without powerful computers.

Automatic Speech Recognition (ASR) is one of these tasks that is only feasible because of fast processors and large memories. Although humans understand speech without major problems even under bad conditions, it is still error-prone to translate speech to text automatically by a machine. Partly this is because speech recognition still has to cope with various limitations constraining the complexity of the solutions and thus reducing recognition accuracy. Two of these limitations are the motivation for this work: limited data and limited computing power.

On the one hand, the ASR system needs to be trained, which means it estimates its parameters based on samples of speech data. To train all the acoustic models reliably, there need to be at least some samples for each of them. As these sounds are modeled in their particular context of surrounding sounds (called *polyphones*, see Section 2.1), there are millions of possible combinations. Creating a speech database to cover all of them is hardly achievable. The data needs to be labeled, i. e. annotated with a transcription of the spoken language in written form, which has to be done manually to a certain extent, and thus is too expensive. The quote

> "There is no data like more data"[1]

brings this to the point.

On the other hand, even if there was a speech database covering all polyphones with sufficient acoustic evidence, computing power would be the bottleneck in many real-time scenarios.

A possibility to overcome these limitations is the clustering of different polyphones that are similar with respect to a specific distance metric. This means the acoustic model treats "similar" sounds as one, allowing to end up with an arbitrary number of polyphone classes instead of millions of single polyphones.

[1]According to [Jeli05], this comment was made by Bob Mercer at Arden House, 1985.

1.1 Objective

In this work we investigate the effects of different acoustic distance metrics on the quality of the clustering. This means we try out measures which determine a distance between two acoustic units in order to define which sounds are "similar" to each other and which are not. Our main objective is to improve the Word Error Rate (WER) of the ASR system by creating polyphone classes that best fit the given speech data.

1.2 Structure

This work consists of a theoretical part which explains the foundations and concepts and a practical part which describes their usage and effects in a real speech recognition system.

Chapter 2 begins with an introduction into some basic terms of speech recognition and clustering in general. In Chapter 3 we describe the task in more detail and present existing approaches to its solution. Finally, Chapter 4 explains the distance measures and other algorithms used in this work.

After this, in Chapter 5 we implement an ASR system and evaluate the effects of the presented distance measures in Chapter 6. Chapter 7 resumes the work and presents issues for further investigations.

1.3 Related Work

In speech recognition, cluster algorithms are used to reduce the number of acoustic models – mainly hierarchical clustering is used for this. The two major ways to do this are bottom up [IKHv00] and top down [FiRo97] algorithms. There also exist hybrids or mix forms of these two base algorithms [SiRS99].

All hierarchical clustering algorithms have in common that they need a distance metric in order to decide how to build the cluster tree. As finding an appropriate distance metric between phonemes is also a common problem in multilingual acoustic modeling, there has been previous research on this:

One field of particular interest for many current research projects is the adaption of acoustic models from one language to another. Mainly for under-resourced languages it is difficult to establish an acoustic model without having much speech data to train the models well [LeBe05].

One approach is to create an automatic cross-language phoneme mapping from a well-trained source language with enough acoustic evidence for training to the target language [LeBe09]. The quality of this mapping depends primarily on the distance measure used to estimate the similarity between source and target phonemes, thus there has been a lot of effort in trying out different measures to get a good mapping [SoBo01].

There are also proposals to use the confusability of two phonemes to measure their distance [LeBS06]. Another approach for a distance measure is to directly compare the Hidden Markov Models (HMMs) the acoustic model is based on [MoTr06, AnHe04].

2. Foundations of Acoustic Models

This chapter describes and defines the basic terms used throughout this work. At first, we describe the acoustic units phonemes, phones and polyphones and their representation in the form of Gaussian Mixture Models, as well as codebooks and distributions. Finally, the basic algorithms for clustering are introduced.

2.1 Phonemes, Phones and Polyphones

A *phoneme* is the cognitive abstraction of a sound representing the smallest structural unit that distinguishes meanings of words[2].

In contrast, a *phone* is the smallest identifiable unit in speech and relates to an instance of a phoneme in an actual utterance[2]. Spoken language can be seen as a continuous series of phones, which therefore build the basic units of our acoustic model.

For example, the words “ceiling” and “sealing” consist of distinct *phonemes*; however their pronunciation is almost identical, which means their *phones* are the same. Linguists and phonologists distinguish precisely between the terms *phoneme* and *phone*. From a more technical point of view, they can easily be mixed up, as the goal of ASR is to find words consisting of phonemes from a given stream of phones. In this work we tried to use them appropriately.

Lastly, a *polyphone* is a phoneme modeled in its particular context of surrounding phonemes. If e. g. the word “hello” is represented as the phoneme series H E L O, the phoneme E has a left context H and a right context of L O, resulting in the polyphone E(H|L O). E is then called the *center phoneme* of the polyphone. The introduction of polyphones to ASR brought about an improvement in recognition accuracy, as taking the context of a phone into account benefits from co-articulation effects in spoken language.

About 50 different phonemes is a typical quantity for the English language, as for example listed in Appendix A. Given these, this can easily sum up to millions of polyphones.

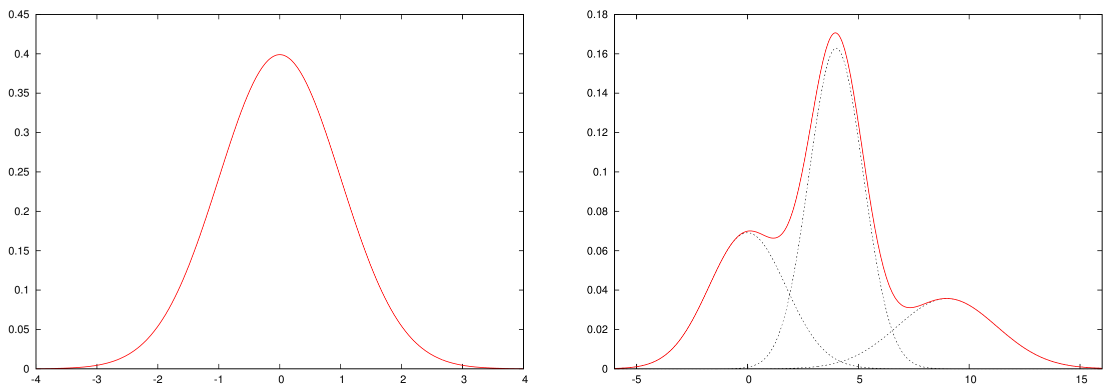
[2] cf. http://www.voxforge.org/home/docs/faq/faq/what-is-the-difference-between-a-phone-and-a-phoneme [Online; accessed September 10,2012]

2.2 Gaussian Mixture Models

A single multivariate (=multidimensional) *Gaussian distribution* $\mathcal{N}$ with dimension N, mean vector (centroid) μ and covariance matrix Σ is defined as

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^N \ \det(\Sigma)}} \cdot \exp\left(-\frac{1}{2}(x-\mu)^{\mathrm{T}}\ \Sigma^{-1}\ (x-\mu)\right) \tag{2.1}$$

This distribution is also known as Normal Distribution (e. g. Figure 2.1(a)).



(a) Gaussian with $\mu = 0$ and $\Sigma = 1$ (b) Gaussian Mixture of three Gaussians

Figure 2.1: Gaussian and Gaussian Mixture Distribution

It can be used for a mixture distribution, which accordingly is called a *Gaussian Mixture Model* (GMM) (e. g. Figure 2.1(b)). This mixture distribution is a linear combination of M single Gaussians with coefficients ω_m, that are called *mixture weights* or simply *weights*. As a GMM itself is a distribution, the ω_m must hold $\sum_{m=1}^{M} \omega_m = 1$ and $0 \le \omega_m \le 1\ \forall m \in \{1 \dots M\}$. Then the GMM Γ is defined as

$$\Gamma(x) = \sum_{m=1}^{M} \omega_m \cdot \mathcal{N}(x|\mu_m, \Sigma_m) \tag{2.2}$$

Gaussian mixtures therefore fulfill the Universal Approximation Theorem, which means that they can be used to model any probability distribution. In the case of speech recognition, they are commonly used to represent the acoustic units of speech.

2.3 Codebooks and Distributions

A set of Gaussians $\mathcal{N}_m$ is called a *codebook* and the set of related mixture weights ω_m its *distribution*. Figure 2.2 shows an example of three polyphones modeled with distributions and codebooks.

Having the Gaussians separated from their weights brings some flexibility into their working:

- A *fully continuous system* uses one codebook per distribution and thus is the most accurate approach, but it needs most training data.

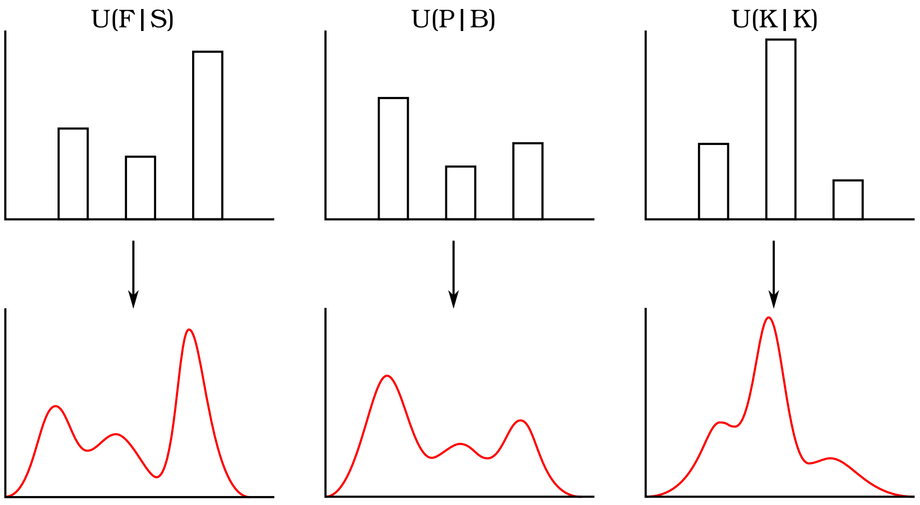


Figure 2.2: Fully continuous system: Three polyphones modeled by distributions and codebooks.

- A *semi-continuous system* instead shares a codebook between several distributions and therefore has less parameters to estimate.

These are two common possibilities to tie codebooks and distributions, which try to find a trade-off between robust training and smooth modeling.

The task of estimating the parameters μ_m, Σ_m and ω_m of the GMMs is called training. It is executed with an expectation-maximization-algorithm, which allows to use an arbitrary number of Gaussians and weights, so we can use it for each grade of continuity.

2.4 Clustering

This section gives a short introduction into clustering algorithms in general. Chapter 4 then gives a detailed description of all the procedures involved in our specific algorithm.

Clustering is the task of dividing a set of objects into subsets (called *classes*) so that a class contains objects similar to each other (with respect to a metric) while objects of different classes are dissimilar. A cluster analysis is a very common tasks in information theory and thus there is no shortage of clustering algorithms; this section however is limited to an introduction of *hierarchical* clustering, as we used these types of algorithms in this work only.

There are two general strategies for hierarchical clustering:

Agglomerative: All objects are initially in separated classes; similar classes are then merged to new classes until the number of classes falls below a certain stop criterion. This is a bottom up approach; Figure 2.3(a) shows a visualization of the process.

Divisive: The objects are initially in one big class which gets divided into smaller classes until a desired number of classes is reached. This approach works top down and is what we used in this work. Figure 2.3(b) shows the process.

In both cases, there are also refinements of the stop criteria, which for example ensure to have reasonable distances between the classes or which make sure to always have enough training samples for all classes.

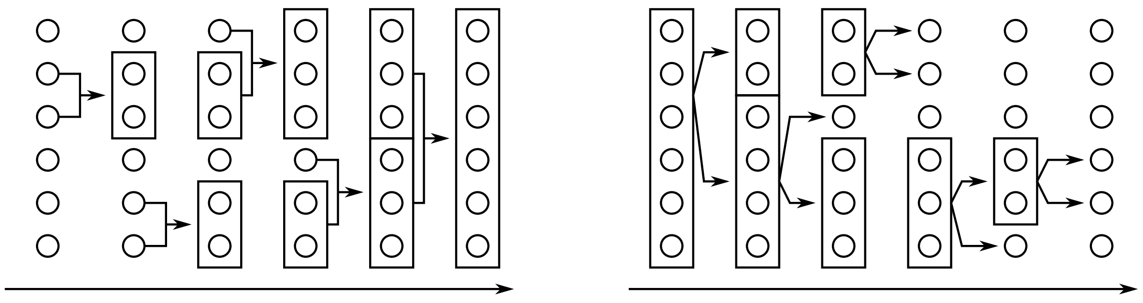

(a) Agglomerative clustering (b) Divisive clustering

Figure 2.3: Visualization of clustering processes

Algorithm 1 gives a basic recursive definition of divisive clustering. Given a class containing all objects, it returns the root node of a tree which represents a hierarchy of the clustered objects.

```
input  : Class of objects
output: Tree of Classes
1 if Class only has one object then
2     return Class
3 end
4 Split Class into n classes $Class_1$ to $Class_n$;
5 for $i = 1 \rightarrow n$ do
6     $TreeNode_i$ = DivisiveCluster($Class_i$);
7 end
8 Create TreeNode with children $TreeNode_1$ to $TreeNode_n$;
9 return TreeNode
```

Algorithm 1: Divisive Clustering

In this general definition, the Split-Command in line 4 neither defines the number of produced classes per split nor the method how to select the objects for each new class. In order to make this a working algorithm for ASR, we will define both properties in Section 4.3.

The objects in the classes of our clustering algorithm are polyphones; the algorithm then clusters similar polyphones into the same class.

3. Analysis

In this chapter, we first analyze the problem background, which at the same time gives further motivation why clustering is necessary in ASR. Then we describe existing approaches to solve the problem and their flaws.

3.1 Problem Background

Modeling phonemes in their particular context in form of polyphones improves the recognition accuracy, because it takes co-articulation effects into account: A phoneme is pronounced differently depending on its surrounding phonemes. Polyphones can be seen as new special phonemes for each slightly different variation of the pronunciation of one base-phoneme (the center-phoneme).

Depending on the width of the context, there are different types of polyphones. Two important types are *triphones* and *quinphones*: The former ones use a context width of ± 1, so they consider the phoneme before and the one after the center phoneme. The latter ones use a width of ± 2, which means, these polyphones use two phonemes before and two after the center phoneme.

Using quinphones on a phoneme set of 50 phonemes leads to a theoretical count of $50^5 = 312{,}500{,}000$ polyphones. Most of them will never appear in the actual training data, but as mentioned in Section 2.1, the number of polyphones in a typical speech database can easily exceed a million.

Figure 3.1 plots the number of polyphones that have a certain count of training samples in the database against this count. The upper left part of the diagram illustrates that the majority of the polyphones only have very few samples in the training data – this is most clearly seen at the topmost data point: about 100,000 polyphones only have one sample in the database. On the other hand, the bottom right part of the diagram shows that there are only few polyphones that have 1,000 or more samples.

This leads to a problem: the majority of the polyphones have not enough speech samples available for a robust and reliable training. Also, the computation during the recognition process takes too long if millions of polyphones are involved.

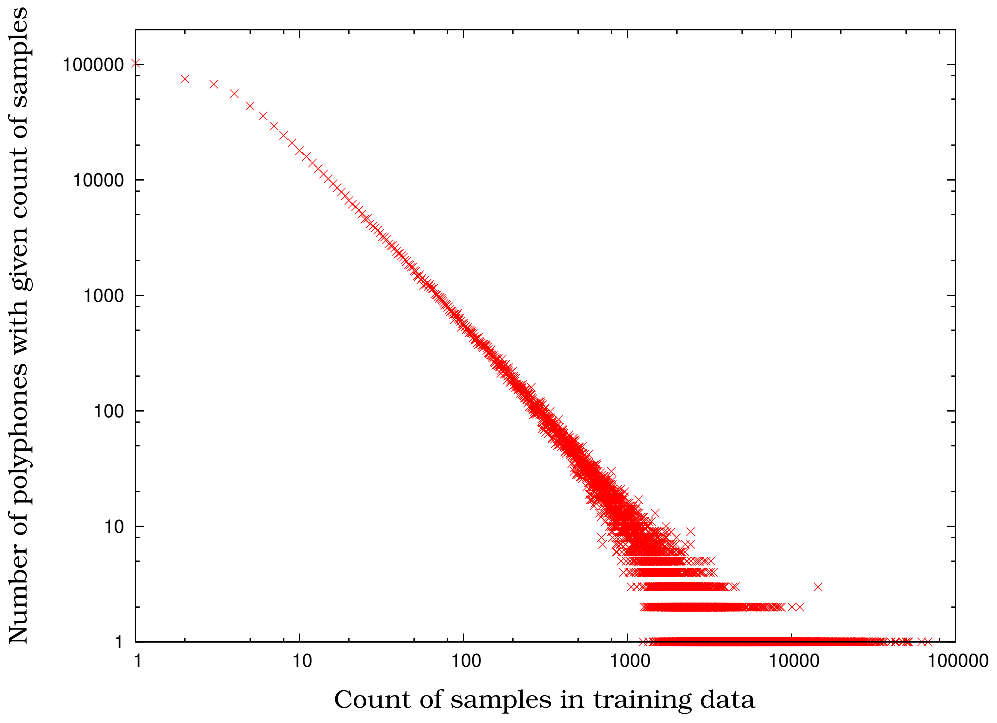


Figure 3.1: How many polyphones have how many training samples?

These are two of the reasons why in speech recognition often clustering algorithms are used to create an arbitrary number of classes of similar polyphones and thus reduce the millions of polyphones to a desired number of classes. The objective of this work is to try out different distance measures that determine which polyphones are "similar" and thus belong into one class and which not.

3.2 Existing Approaches

The clustering algorithm as described in Section 2.4 needs to calculate the distance between polyphone classes as a measure of their similarity. This is done several times in each split step (line 4) – depending on the specific settings, in our experiments one run of the clustering needs to calculate a total of 1.5–2.6 million distances.

This necessitates to restrict the complexity of the models in order to be processable with the available computing power. A common approach is to use semi-continuous acoustic models, where several distributions share one codebook, as introduced in Section 2.3. This means that the single acoustic units, e. g. polyphones, use the same codebook, but differ in the mixture weights of the Gaussians. Figure 3.2 shows an example of three distributions sharing one codebook.

Using semi-continuous models offers a good compromise between few parameters and smooth modeling: As most of the parameters are in the means and covariance matrices of the Gaussians, sharing them reduces their number drastically, which makes the training more robust – but the feature space is still continuous as it is when using fully-continuous models.

One advantage of semi-continuous models concerning computing performance is the simplicity of calculations on the distribution weights: During the clustering there is no need to evaluate the emission probabilities of the Gaussian mixtures. A distance

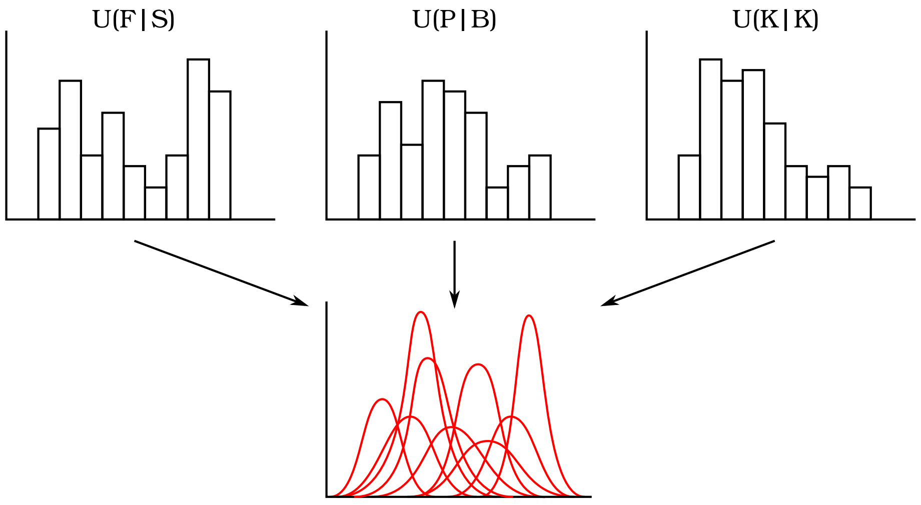


Figure 3.2: Semi-continuous system: Three polyphones modeled by distributions over one codebook.

measure that benefits from that is the *entropy distance*, which works on the distribution weights only: it measures the distance as the entropy gain of the distribution weights when the the classes are split.

The entropy H of a discrete distribution f [Shan48] is defined as

$$H(f) = -\sum_{i=1}^{k} f(i) \log_2 f(i) \tag{3.1}$$

The entropy is a measure for the unpredictability of the information content of a random variable. A high value represents a high uncertainty – with the maximum reached at equal distribution and the minimum (0) reached for certain events.

To use this as a distance on distribution weights, let C_1 and C_2 denote two polyphone classes and f_{C_1}, f_{C_2} be their discrete mixture weights distribution, defined as $f_C(m) = \omega_{C,m}$. The counts of training samples for the classes be n_1 and n_2, which means there are n-many samples in the speech database that belong to the classes.

For the actual distance measure, we need the distribution of the union $C_1 \cup C_2$ of the two classes[3]:

$$f_{C_1 \cup C_2}(m) = \frac{n_1 \cdot f_{C_1}(m) \ + \ n_2 \cdot f_{C_2}(m)}{n_1 + n_2} \tag{3.2}$$

Then we can use the simple entropy distance

$$d_{\text{SimEntr}}(C_1, C_2) = H(f_{C_1 \cup C_2}) \ - \ \frac{1}{2} \cdot H(f_{C_1}) \ - \ \frac{1}{2} \cdot H(f_{C_2}) \tag{3.3}$$

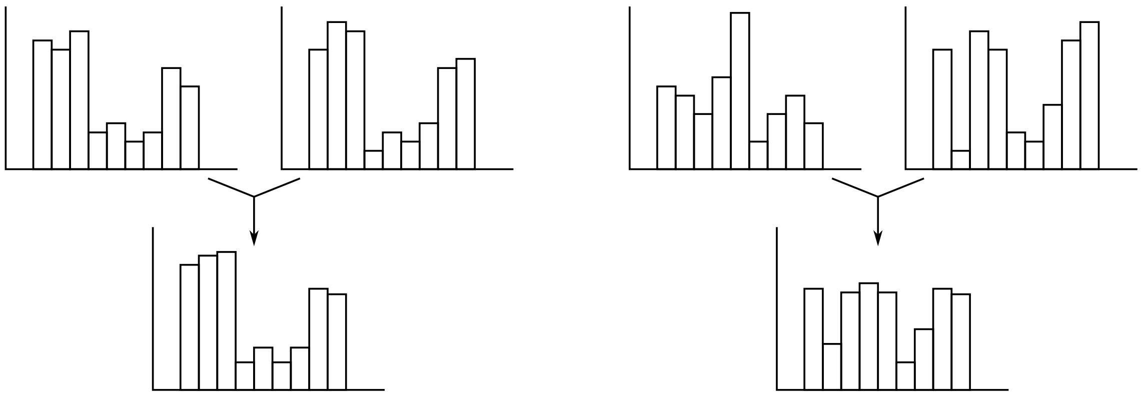
or the weighted entropy distance

[3]This union is actually the same polyphone class that was split into C_1 and C_2 during the clustering.

$$d_{\text{WeiEntr}}(C_1, C_2) = (n_1 + n_2) \cdot H(f_{C_1 \cup C_2}) \; - \; n_1 \cdot H(f_{C_1}) \; - \; n_2 \cdot H(f_{C_2}) \tag{3.4}$$

to measure the distance between C_1 and C_2.

The motivation to use the entropy as a distance measure is visualized in Figure 3.3: Joining two similar distributions results in a distribution which is similar to both (Figure 3.3(a)), so their entropy values also are similar and thus the distance is small; joining distinct distributions leads to a different distribution (Figure 3.3(b)), which is closer to an equal distribution and so the entropy value is higher than the single entropy values of the input distributions, which makes the distance between them bigger.



(a) Two similar distributions (b) Two distinct distributions

Figure 3.3: Distributions and their union

As mentioned, this approach only takes the distribution weights of classes defined over the same codebook into consideration. This makes it easy to calculate and thus it is a good distance measure for the clustering algorithm when computation power is limited. In Chapter 4 we instead present a clustering algorithm which works on fully-continuous models and introduces new helper codebooks for the classes during the clustering and therefore uses more of the available information.

4. Concepts

This chapter describes the functions and techniques used in this study as parts of the clustering algorithm. The distance measures are the subject of investigation of this work, so they are described first. The concepts of merging models and splitting with questions follow, before finally the clustering algorithm is described, which depends on both concepts.

4.1 Distance Measures

As mentioned in Section 3.2, this study tries out new distance measures for the clustering algorithm that are based on codebook distances instead of the distribution weights only. This implies the use of a fully continuous acoustic model, because in order to use codebooks as the basis to measure polyphone distances, they have to be distinct.

This section discusses the distance measures used in this work, following [Stü09]:

- the Euclidian distance
- the Kullback-Leibler divergence
- the Extended Mahalanobis metric
- the Bhattacharyya metric

First, we will describe variants of the measures that operate on two Gaussians and afterward we suggest a method for the application of the measures to GMMs with an arbitrary number of Gaussians.

For the following sections let two multivariate Gaussian distributions g_1 and g_2 have the dimension N, both given as $g_i = \mathcal{N}(\mu_i, \Sigma_i)$ with means μ_i and covariances Σ_i.

4.1.1 Euclidian Distance

The *Euclidian distance* measures the distance between two Gaussians by simply calculating the distance between their means μ_i as given by the Pythagorean theorem:

$$d_{\mathrm{Eucl}}(P, Q) = \sqrt{(\mu_1 - \mu_2)(\mu_1 - \mu_2)^{\mathrm{T}}} \tag{4.1}$$

It solely relies on the means and does not take the covariances into account, which leaves out some of the available information.

4.1.2 Kullback-Leibler Divergence

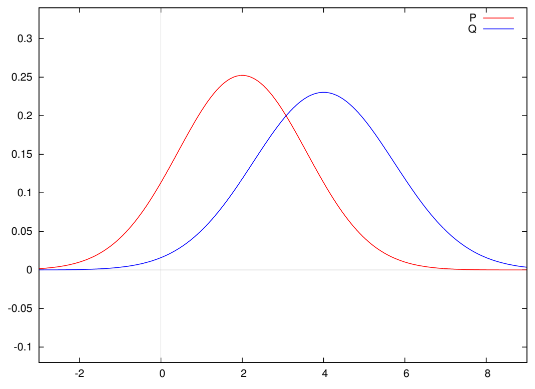
The *Kullback–Leibler* (KL) divergence is a measure for the dissimilarity of two probability distributions. For distributions P and Q over a discrete random variable X [Runn07], it is:

$$d_{\mathrm{KL,Base}}(P, Q) = \sum_{x \in X} P(x) \cdot \log_2 \frac{P(x)}{Q(x)} \tag{4.2}$$

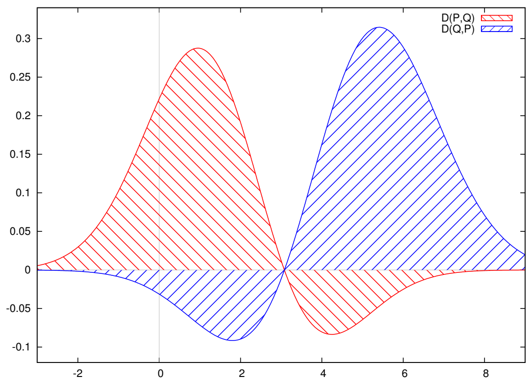
In a typical application, P represents a theoretical, exact probability distribution whereas Q is an approximation model to describe P. It can then be interpreted as the number of bits wasted when using a code based on Q to encode observations that actually result from P.

However, it is neither symmetric nor does it obey the triangle inequality and thus is not a metric in this form.

Figure 4.1 illustrates the Kullback–Leibler divergence for two Gaussian distributions: Figure 4.1(a) shows the distributions, while the area under Figure 4.1(b) symbolizes the value of their distance. The asymmetry is clearly visible.



(a) Two normal distributions P and Q



(b) KL divergence between P and Q

Figure 4.1: Illustration of the Kullback–Leibler divergence

In order to use it as a distance function, we combine the Kullback-Leibler divergence between P and Q with the divergence between Q and P:

$$d_{\mathrm{KL,Sym}}(P, Q) = d_{\mathrm{KL,Base}}(P, Q) + d_{\mathrm{KL,Base}}(Q, P) \tag{4.3}$$

Furthermore, the use of Gaussians as distributions simplifies the Kullback-Leibler measure [LoSa01] to

$$d_{\mathrm{KL}}(g_1, g_2) = \frac{\Sigma_1}{\Sigma_2} + \frac{\Sigma_2}{\Sigma_1} + (\mu_2 - \mu_1)^2 \cdot \left(\frac{1}{\Sigma_1} + \frac{1}{\Sigma_2}\right) \tag{4.4}$$

This is the form of the KL distance we use for the experiments.

4.1.3 Extended Mahalanobis Metric

The *Mahalanobis* distance measures the distance between a multidimensional vector x and a set X of samples that result from a distribution with mean μ and covariance matrix Σ:

$$d_{\mathrm{Mah,Base}}(x, X) = \sqrt{(x-\mu)^{\mathrm{T}}\ \Sigma^{-1}\ (x-\mu)} \tag{4.5}$$

It is an extension to the Euclidian distance[4] that takes the covariance into account.

If we see Σ as an ellipsoid, the Mahalanobis distance can be interpreted as the distance of the vector x to the mean μ with respect to the width of the ellipsoid in the direction between them. Figure 4.2 shows an illustration of the Mahalanobis distance: The points on each of the ellipses given by Σ have the same distance to the center μ.

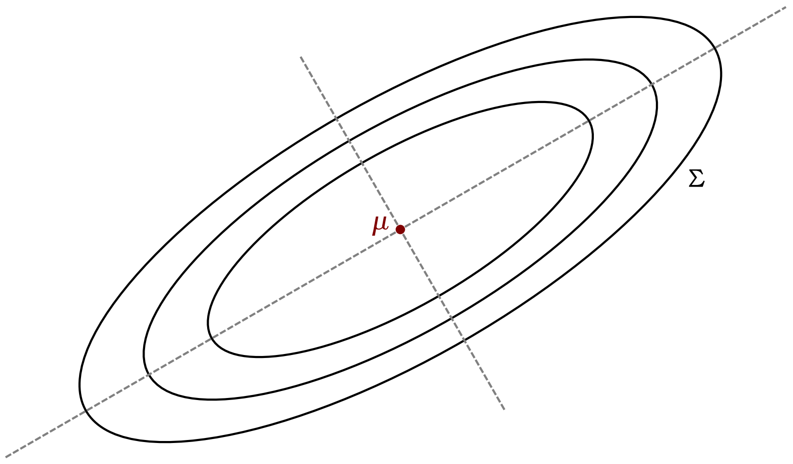


Figure 4.2: Illustration of the Mahalanobis distance

The *Extended Mahalanobis* distance [BoNA10] adopts this to Gaussians by using the means μ_i and the sum of the covariances Σ_i as parameters:

$$d_{\mathrm{Mah}}(g_1, g_2) = \sqrt{(\mu_2-\mu_1)^{\mathrm{T}}\ (\Sigma_1+\Sigma_2)^{-1}\ (\mu_2-\mu_1)} \tag{4.6}$$

However, as the covariances are first combined into one, this metric also does not take full advantage of the available information.

[4]If Σ is the identity, the Mahalanobis distance in fact becomes the Euclidian distance.

4.1.4 Bhattacharyya Metric

The *Bhattacharyya* distance metric is a common distance between two probability distributions. For distributions P_1 and P_2 over a discrete random variable X [LiTz00], it is:

$$d_{\text{Bha,Base}}(P_1, P_2) = -\ln\left(\sum_{x \in X} \sqrt{P_1(x) P_2(x)}\right) \tag{4.7}$$

It is symmetric but does not necessarily obey the triangle inequality.

For multivariate Gaussian distributions [SoBo01, RiTJ00, MaBa96] it is defined as

$$\begin{aligned} d_{\text{Bha,Full}}(g_1, g_2) = \frac{1}{8}(\mu_2 - \mu_1)^{\mathrm{T}} \left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1} (\mu_2 - \mu_1) \\ + \frac{1}{2} \ln \frac{\det(\Sigma_1 + \Sigma_2)}{2\sqrt{\det(\Sigma_1) \cdot \det(\Sigma_2)}} \end{aligned} \tag{4.8}$$

Note that the first term of this resembles Equation 4.6 in Section 4.1.3.

Using diagonal covariance matrices [LiTz00] simplifies this to

$$d_{\text{Bha,Diag}}(g_1, g_2) = \frac{1}{4} \sum_{i=1}^{N} \frac{|\mu_{2,i} - \mu_{1,i}|^2}{\sigma_{1,i}^2 + \sigma_{2,i}^2} + \frac{1}{2} \sum_{i=1}^{N} \ln \frac{\sigma_{1,i}^2 + \sigma_{2,i}^2}{2\sqrt{\sigma_{1,i}^2 \cdot \sigma_{2,i}^2}} \tag{4.9}$$

where the variance σ^2 denotes the diagonal elements of the matrices Σ.

4.1.5 Application to GMMs

All the above measures are defined to calculate the distance of two Gaussians from each other. One possibility to apply the measures to GMMs is to find a mapping between the single Gaussians of the two GMMs and compute the distance on pairs of mapping Gaussians (e. g. [GoGG03, GoAr05]).

In this work we map all components of the GMMs pairwise and calculate the distance as the sum of the distances, weighted with the distribution values ω_i of the Gaussians:

$$d_{\text{GMMs}}(\Gamma_1, \Gamma_2) = \sum_{g_1 \in \Gamma_1} \sum_{g_2 \in \Gamma_2} \omega_1 \cdot \omega_2 \cdot d(g_1, g_2) \tag{4.10}$$

where the different measures defined above are used as distance function $d(g_1, g_2)$ in the experiments. This means, we calculate the distance between each Gaussian of the first mixture and each Gaussian of the second mixture and add up these $|\Gamma_1| \cdot |\Gamma_2|$ distances using the ω_i as weights.

It is however disputable whether this method is suitable to describe the similarity between mixture models. We therefore run our experiments on GMMs as well as on single Gaussian models. As we will see in Chapter 6, the systems using mixture models do not score as good as systems with only one Gaussian due to this flaw.

4.2 Merging Models

For the clustering algorithm we will need to merge a set of GMMs into a single new auxiliary GMM: The distance measures operate on single Gaussians and single GMMs respectively, but not on sets of them. In the clustering however we will have classes of polyphones that contain a set of GMMs each. We will need to calculate the distance between two of those classes, so this section presents methods to merge all GMMs of one class into a single one.

Figure 4.3 shows how three Gaussian mixture models in a polyphone class are merged into a helper model representing the whole class.

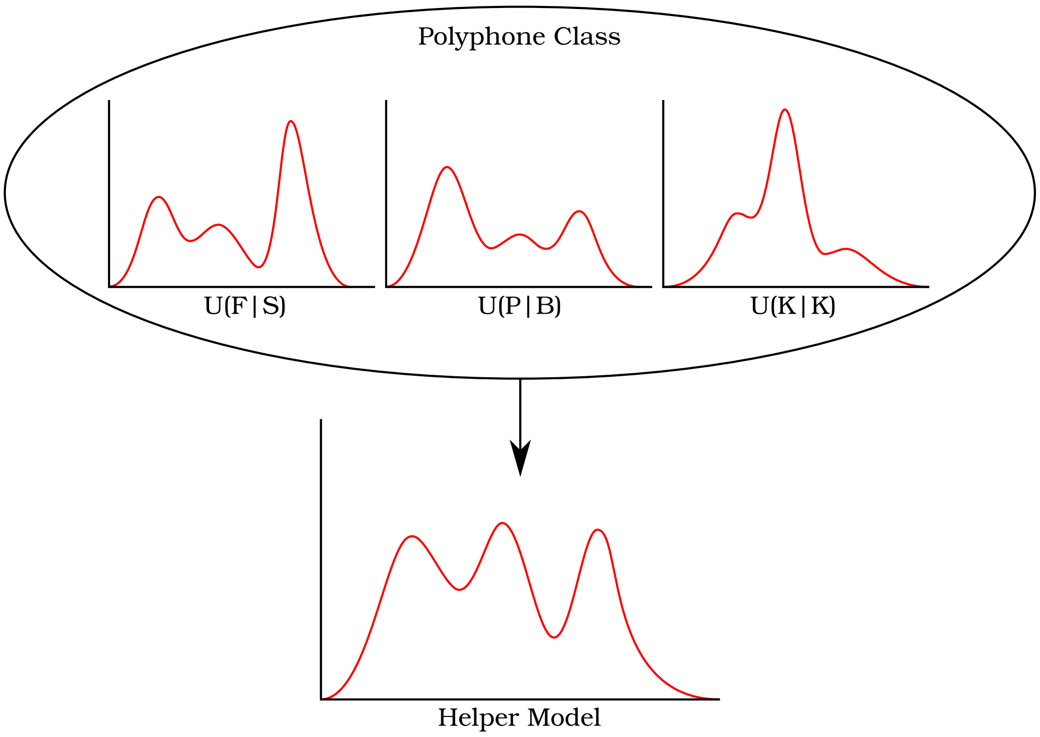


Figure 4.3: Merging Gaussians

Let a polyphone class contain a set $G = \{\Gamma_1, \dots \Gamma_m\}$ of m GMMs for its polyphones and let every Γ_i be estimated on a set of training samples X_i. The task is then to obtain the parameters of a new auxiliary GMM Γ_h that most likely emits the unified samples $X_h = \bigcup_{i=1}^{m} X_i$. The following two subsections will present methods for this depending on the number of Gaussians in the GMMs.

This newly created GMM Γ_h is merely a temporary helper used to calculate the distance between two polyphone classes. It will be used in line 8 of Algorithm 2 and discarded immediately after calculating the distance.

In every cluster step we will need to calculate many distances, and as the classes change in each step, there is no possibility for caching mechanisms. This necessitates an efficient method for merging the models.

4.2.1 Merging Single Gaussians

If the GMMs Γ_i of the polyphone class contain only one Gaussian g_i each[5], there is a closed form to calculate the Gaussian resulting from merging them. It is sufficient to describe how to merge two Gaussians g_1 and g_2, as this process can be repeated successively for each element of the polyphone class then.

We will use the following computational formula for the variance σ^2 of a set X of random variables with $n = |X|$ elements and mean $\mu = \frac{1}{n}\sum_{x\in X} x$:

$$\begin{aligned}\sigma^2 &= \frac{1}{n}\sum_{x\in X}(x-\mu)^2 \\ &= \frac{1}{n}\left(\sum_{x\in X} x^2 - n\mu^2\right) \\ &= \frac{1}{n}\sum_{x\in X} x^2 \;-\; \mu^2 \qquad (4.11)\end{aligned}$$

Now let the two Gaussians g_1 and g_2 as in Section 4.1 be estimated from training sample data sets X_1 and X_2 with counts $n_1 = |X_1|$ and $n_2 = |X_2|$.

Based on these we want to create a new helper Gaussian $g_h = \mathcal{N}(\mu_h, \Sigma_h)$ that most likely emits the data $X_h = X_1 \cup X_2$ of both given Gaussians, thus having $n_h = n_1 + n_2$ training samples.

The mean is simply a weighted sum of the given means:

$$\mu_h = \frac{n_1}{n_h}\mu_1 + \frac{n_2}{n_h}\mu_2 \qquad (4.12)$$

We get the diagonal covariance matrix Σ_h by calculating the variances σ_h^2 for each dimension:

$$\begin{aligned}\sigma_h^2 &= \frac{1}{n_h}\cdot\sum_{x_h\in X_h}(x_h-\mu_h)^2 \qquad (4.13) \\ &= \frac{1}{n_h}\cdot\sum_{x_h\in X_h} x_h^2 \;-\; \mu_h^2 \\ &= \frac{1}{n_h}\cdot\sum_{x_1\in X_1} x_1^2 \;+\; \frac{1}{n_h}\cdot\sum_{x_2\in X_2} x_2^2 \;-\; \mu_h^2 \\ &= \frac{1}{n_h}\left(n_1\sigma_1^2 + n_1\mu_1^2 + n_2\sigma_2^2 + n_2\mu_2^2 - n_h\mu_h^2\right) \qquad (4.14)\end{aligned}$$

[5]This is called a *monogaussian model*, which actually is not really a *mixture* model.

Equation 4.13 is the default formula for variance given the sample data X_h; by applying 4.11 twice, we get 4.14. This form only uses variables that are given by g_1 and g_2, so we can directly calculate the new variance σ_h^2 without using the sample data.

We now have calculated the parameters μ_h and Σ_h of g_h from the two original Gaussians without using their sample data, which allows a reduction of needed computing resources.

4.2.2 Merging GMMs

If the GMMs however have more than one Gaussian each, there is no closed analytic solution known to us to derive a common Gaussian mixture that maximizes the emission probability for the union of the given set of GMMs. To obtain the helper GMM Γ_h in this case, we need to train it on the combined sample data $X_h = \bigcup_{i=1}^{m} X_i$ of all polyphones in the class.

Our training data contains about 56 million sample vectors and during the clustering there are about 2 ± 0.5 million distances to calculate – while for each of them at least two helper models have to be trained. To use a full training for this is not achievable, so we use a simplified algorithm:

Given the set $G = \{\Gamma_1, \dots \Gamma_m\}$ of m Gaussians mixtures and the sample data $X = \{X_1, \dots X_m\}$ on which they were originally estimated, we get the helper model Γ_h in these steps:

1. Initialize the helper model with values (means, covariances, mixture weights) from a random model $\Gamma_i \in G$. This avoids an expensive run of the k-means algorithm, while still being a reasonable initialization for the given data.
2. Run an expectation-maximization-algorithm on a subset of X. We cannot use all the available data here for performance reasons, so we choose to take a maximum of 5.000 random samples. This is sufficient for a reliable estimation, but still small enough for fast computation.
3. Repeat step 2 for a total of 4 iterations in order to converge the parameters. We use a different random subset in each repetition to maximize the number of samples seen during this training.

This returns the helper GMM, albeit it is more expensive then the direct calculation for single Gaussians.

4.3 Splitting with Questions

The clustering algorithm works divisive, which means it splits classes into smaller classes. As stated in Section 2.4, we need to provide information on how to execute these splits. This is the subject of this section.

The split method for polyphone classes is based on questions concerning the polyphones. Questions can ask for specific phonemes of the polyphone's context, for example 0=T asks whether the center phoneme is a T, or -1=B asks if the first phoneme in the left context of the polyphone is a B. The answer to a question can either be "yes" or "no"[6], which divides a polyphone class into two new classes concerning the question and thus results in a split.

Figure 4.4 shows an example of some polyphones divided into two classes based on a question: All polyphones with a vowel on the right are classified in the "yes"-branch, the others in the "no"-branch.

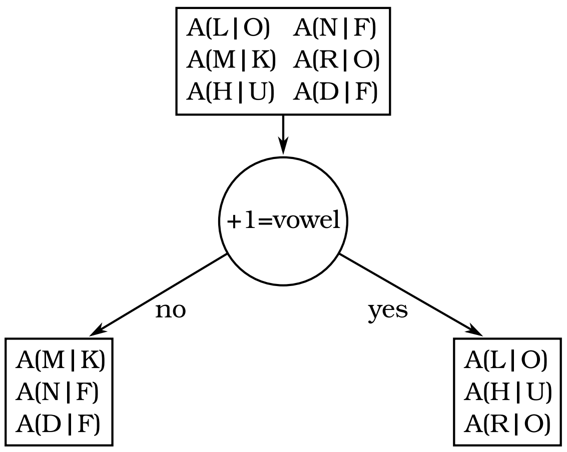


Figure 4.4: Splitting with Questions

We now can determine the distance between these two polyphone classes in two steps:

1. Merge the models of the polyphones in each class to a single model per class using the methods described in Section 4.2.
2. Now we have one model for the "yes"-class and one model for the "no"-class and can calculate their distance as in Section 4.1.

This distance can be interpreted as a score of how well the splitting is: the bigger the distance, the more the question is suitable for splitting. If the distance between two classes is small, this means they are similar to each other with respect to the question that created them, and thus the question is not a good candidate for a split. However if the distance is big, the classes are distinct and the question provides a good discrimination for a split.

Equipped with this, we can find the question that scores best to split a given polyphone class with Algorithm 2. In line 8, we use the two steps from above to calculate the distance between two classes.

Line 5 uses the function $|\cdot|$, which returns the total number of training samples $|C| = \sum_{X \in C} |X|$ for the polyphones of the class C. The condition in this line provides a criterion to prevent splits whose resulting classes have to few training samples to be trained reliably: Only if both resulting classes have a total of more

[6]Also, "unknown" is a possible answer, for example if the question asks for a context wider then the given context of the polyphone: -3=F is not applicable to the polyphone E(B|R). We avoided this answer by using a padding phoneme that fills gaps and returns "no" to every question.

```
input  : $PolyphoneClass$, $QuestionSet$
output: The $Question$ that best splits $PolyphoneClass$ and its $Distance$
1  $BestDistance = 0$;
2  $BestQuestion =$ NULL;
3  foreach $Question$ in $QuestionSet$ do
4      Split $PolyphoneClass$ into two classes $C_{yes}$ and $C_{no}$ according to $Question$;
5      if $|C_{yes}| < SplitMinCount$ or $|C_{no}| < SplitMinCount$ then
6          continue
7      end
8      Calculate $Distance$ between $C_{yes}$ and $C_{no}$;
9      if $Distance > BestDistance$ then
10         $BestDistance = Distance$;
11         $BestQuestion = Question$;
12     end
13 end
14 return $BestQuestion, BestDistance$
```

Algorithm 2: Find Question

than *SplitMinCount* training samples, this split is considered. Section 5.2 will go into detail about this parameter.

The essential choice before starting this algorithm is the selection of a suitable set of questions: They aim is to split a class of polyphones into two classes that sound “different”, so a good question distinguishes different sounds. Instead of asking for specific phonemes as introduced in the first paragraphs of this section, we therefore use questions about groups of similar phonemes.

For example, -1=voiced asks if the phoneme on the left involves vibration of the vocal cords when pronounced[7] or the question +1=plosive asks if the next phoneme is pronounced with blocking the vocal tract[8] – which means, both ask for specific properties of the sounds to find similarities. Appendix B lists all groups of phonemes used to build the question set.

4.4 Clustering Algorithm

The clustering algorithm is the core of our experiment's calculations: its goal is to find classes of polyphones that are similar to each other with regard to a distance metric. It uses all the methods described above and returns a tree of questions with polyphone classes as leafs; each leaf contains then a class of similar polyphones. These kind of trees are called *classification and regression trees* (CART)[9] as described in [BFOS84].

In opposition to other clustering methods like agglomerative clustering, the use of decision trees ensures that also polyphones that occur in the testing data but not in

[7]These are the vowels and some other phonemes like N or M.

[8]For example, the phonemes K, P and T.

[9]The original CARTs however use only the entropy distance to find the best split.

the training data are assigned to the correct polyphone class: When walking down the tree, even for unseen polyphones all questions can be answered unambiguously.

Algorithm 3 presents the essence of the steps involved.

```
input  : PolyphoneClass, QuestionSet
output: SplitTree of Questions and Classes
1  Add PolyphoneClass as root to new SplitTree;
2  (BestQuestion, BestDistance) = FindQuestion(PolyphoneClass, QuestionSet);
3  Add (PolyphoneClass, BestQuestion, BestDistance) to new SplitList;
4  while SplitList not empty do
5      (Class, Question, Distance) = Dequeue first element of SplitList;
6      if $|Class| < TreeMinCount$ then
7          continue
8      end
9      Split Class into two classes $C_{yes}$ and $C_{no}$ according to Question;
10     Replace Class in SplitTree by split node containing Question;
11     Attach children $C_{yes}$ and $C_{no}$ to this node;
12     (BestQuestion, BestDistance) = FindQuestion($C_{yes}$, QuestionSet);
13     Add ($C_{yes}$, BestQuestion, BestDistance) to SplitList, sorted by Distance;
14     (BestQuestion, BestDistance) = FindQuestion($C_{no}$, QuestionSet);
15     Add ($C_{no}$, BestQuestion, BestDistance) to SplitList, sorted by Distance;
16 end
17 return SplitTree
```

Algorithm 3: Cluster

The algorithm works top-down, which means it starts with one big polyphone class and splits it into smaller classes. In the initialization (lines 1 to 3), we create the *SplitTree*, which will be the algorithm's result, and use the given *PolyphoneClass* with all polyphones as the initial root node. For this class we then find the best scoring question and use it to initialize the *SplitList* of possible splits, which is sorted by the distance between the classes resulting from a split.

As the *SplitList* is always sorted, we ensure that the main loop (lines 4 to 16) performs the best possible next split in each iteration: We split a class (line 9), replace its node in the *SplitTree* by the according question (line 10) and for both classes resulting from the split now find best questions and add them to the *SplitList* (lines 12 to 15).

Line 6 is a criterion that applies if the polyphone class has too few samples in the speech database to be trained reliably. This means that a polyphone class is not further divided if the total number of samples is below $TreeMinCount$. The function $|\cdot|$ again returns the total number of training samples $|C| = \sum_{X \in C} |X|$ for the polyphones of the class C. Section 5.2 will further explain this min count.

The resulting decision tree of the algorithm contains questions on every branch, which split the original polyphone class, and the leafs are the polyphone classes. If these classes are used as representatives of polyphones, they are called *generalized polyphones*. A generalized polyphone thus is a phoneme which is modeled in the context of its class (given by the questions of the split tree), instead of being modeled in a specific context of surrounding phonemes.

Figure 4.5 shows a part of a decision tree.

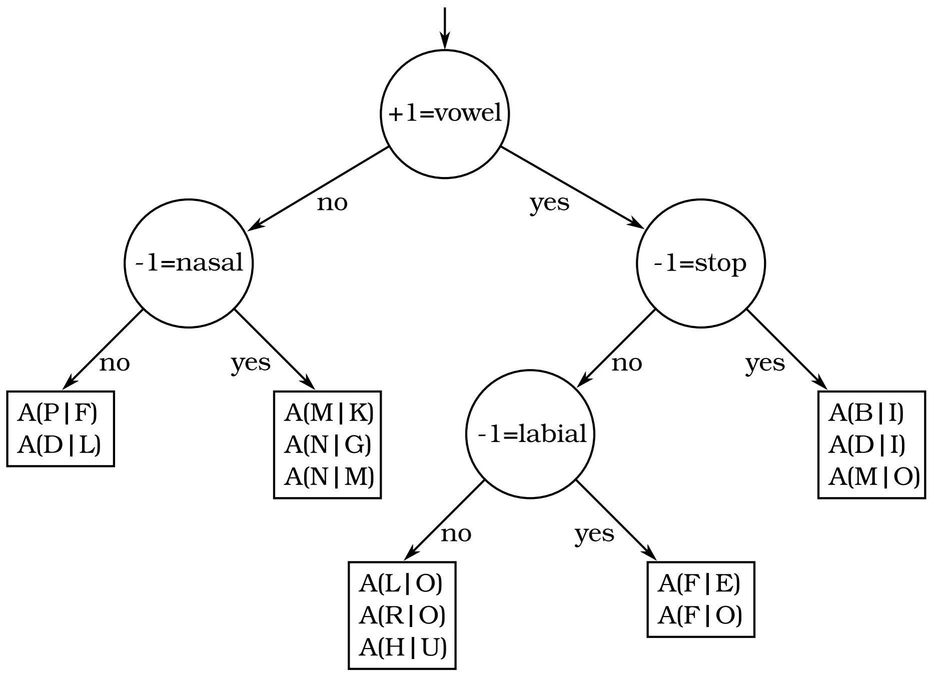


Figure 4.5: Part of a Split-Tree after Clustering

The classification resulting from our clustering algorithm is then used to train and evaluate ASR systems. Details on that are subject of Chapter 5.

5. Implementation

This chapter covers the practical realization of the concepts in order to test their performance in an ASR system. First, we describe the environment of the experiments and the system setup. After a description of the settings used to parametrize the algorithms, the experiment setup lists the steps involved to evaluate our distance measures.

5.1 System Setup

The implementation of the system as presented in Chapter 4 is build upon the Janus Recognition Toolkit (JRTk) version 5.1.1. This ASR system is developed since 1993 at the Interactive Systems Labs, which is based at Karlsruhe Institute of Technology, Germany, and at Carnegie Mellon University, USA. The system is similar to the one described in [FiRo97].

The environment used for this work provides diagonal covariance matrices instead of full covariance matrices – that means only the diagonal elements are different from 0 and represent the variances σ^2 of the Gaussians. As a consequence the codebooks are only able to represent uncorrelated features, which makes some of the calculations easier while keeping the modeling relatively smooth.

The used speech data consisted of audio recordings in English. We use broadcast news and broadcast conversations from the Quaero Project[10]. The advantage of using broadcast data is the cleanness of the speech: news anchormen are trained for clear pronunciation and the channel is mostly undisturbed by noise, which both makes the recognition process easier and more reliable.

The data is divided into two parts:

1. A *training database*, consisting of about 187 hours of speech, is used as basis for the clustering algorithm as well as for the training of the final system.
2. A *testing database*, consisting of almost 4 hours of speech, is then used to test this system and to evaluate and score it.

[10] cf. http://quaero.org [Online; accessed October 1, 2012]

The evaluation is carried out by quantifying the *Word Error Rate* (WER) of the different systems for each set of parameters as described in the next section.

5.2 Settings

There are several parameters that influence the clustering algorithm. The following listing describes the parameters and explains the decisions on which values to use for them:

Distance measure: Trying out the different distance measures as presented in Section 4.1 is the objective of this work – we run the experiments using the Euclidian distance, the Kullback-Leibler distance, the Extended Mahalanobis distance and the Bhattacharyya distance.

Number of Gaussians: The helper models as introduced in Section 4.2 cope with an arbitrary number of Gaussians to model the GMMs for polyphone classes. Therefore, this parameter determines how many Gaussians are used for the models and helpers to calculate the class distances. Incrementing this number makes the modeling of the acoustic features smoother – but at the same time the GMM distances loose significance, as mentioned in Section 4.1.5.

Maximum number of splits: This number determines how many different splits are created during the clustering and thus is the number of final classes that result from the algorithm. One has to find a tradeoff for the number of these classes: On the one hand, the higher their number, the more accurate a system can detect different polyphones (provided that enough training data is available for each of them). On the other hand, if their number is small, each single class accumulates more sample data and thus can be training more reliably.

Split Min Count: This stop criterion is used in line 5 of Algorithm 2 in Section 4.3. It ensures that only those splits are considered as possible that result in two classes with enough sample data to train both reliably.

Treenode Min Count: In line 6 of Algorithm 3 in Section 4.4 we use this count as a criterion to decide when the clustering algorithm should not further split a polyphone class. It assures that classes that are already too small for splitting (they cannot be trained reliably) are left out in further clustering.

The difference between the two min counts is small, as both determine a threshold when to not further split a class – so the system could work with only one of them. But still both influence the algorithm at different points: While the Split Min Count decides which splits are possible for a class, the Treenode Min Count applies even before a class is split.

5.3 Experiment Setup

Given the system as described in the previous sections, the following steps are performed for each set of parameters in order to run and evaluate the clustering with different distance measures:

1. **Extract samples:** The first step is to generate samples for all polyphones from the speech database. As we use a fully continuous acoustic model in the clustering algorithm, we need to extract one set of samples for each of the almost 800,000 polyphones separately.

2. **Train the codebooks:** Before the cluster algorithm can calculate distances between the codebooks of the single polyphones, they have to be trained on the given data. For this purpose we initialize them with a k-means algorithm and run four Expectation-Maximization iterations.

3. **Run the cluster algorithm:** The core of the experiments is the actual clustering, where the concepts from Chapter 4 are applied. The result of this is a decision tree which describes the polyphone classes.

4. **Train the system:** This decision tree is then used to build new codebooks and distributions for the classes (*generalized polyphones*) resulting from the clustering; using them, an ASR system is trained on the training database.

5. **Test and score the system:** This system finally is evaluated on the testing database and the outcome in form of the WER is measured.

Figure 5.1 shows two exemplary generalized polyphones as they are used in step 4. The dashed lines are the single Gaussians used in the codebook of each polyphone; the continuous lines show the weighted mixture of the Gaussians and therefore represent the probability density of the complete polyphone.

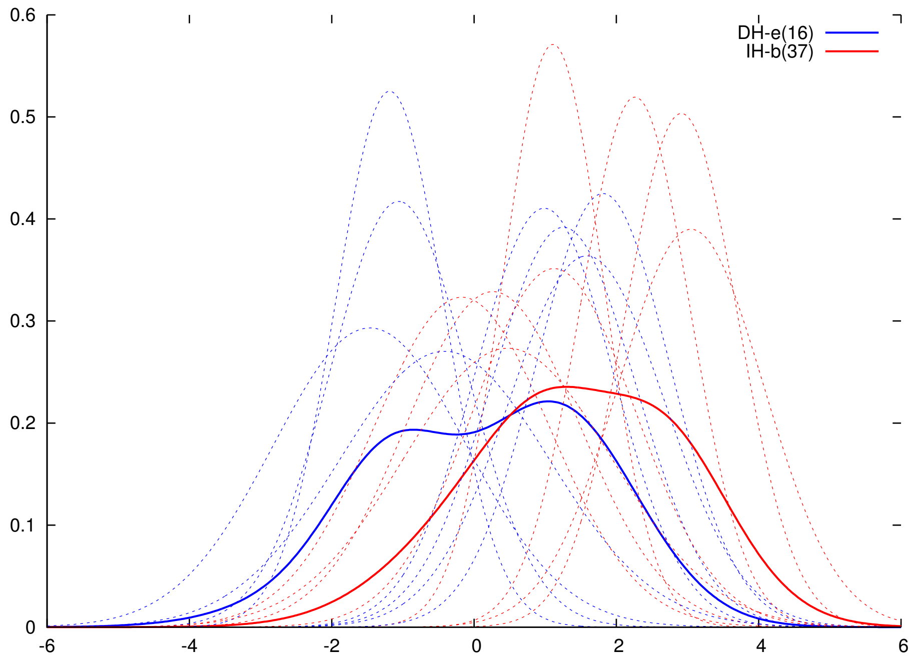


Figure 5.1: Generalized Polyphones

In order to keep the calculations efficient, steps 2–3 are run on each center phoneme separately. Their results are then joined by generating a new decision tree combining the splits sorted by their distance before starting with steps 4 and 5.

With this setup, we are able to measure the ability of the distance measures for polyphone clustering by comparing the resulting Word Error Rates. This evaluation is the subject of Chapter 6.

6. Evaluation

This chapter discusses the results of our experiments with different distance measures and compares them to the entropy distance as described in Section 3.2. As we use the same training and testing data for all the experiments, they are directly comparable. The benchmark is evaluated in form of the *Word Error Rate* (WER) – the entropy distance reached a WER of 28.1%. Our goal is to get below this value.

As described in Section 5.2, there are various parameters to configure our system. For each set of parameters, at least steps 3–5 of Section 5.3 have to be executed individually[11], which takes about one week on our computer cluster. Thus, evaluating all permutations of the parameters is not feasible, which is why we set some parameters to established values from experience with similar systems.

The settings of our experiment are as follows: The maximum number of splits is set to 6000, which is a good compromise to have enough, but still well trainable, models for discrimination; the split min count is set to 1000, the treenode min count to 2000. Again, these values allow the clustering algorithm to create a significant number of classes on the one hand, and on the other hand make sure to have sufficient data for a reliable training of each model.

Using these fixed parameters, we now can modify the number of Gaussians and run evaluations for each distance measure. Table 6.1 shows the Word Error Rates using 1, 4, 8, 16 and 32 Gaussians for the helper models. For better visualization, Figure 6.1 shows the same results as a graph.

As easily can be seen, none of our results outperforms the entropy distance directly. This seems to have mainly two reasons:

On the one hand, helper models with only one Gaussian have the least number of parameters to estimate, as they only need one mean vector and one covariance matrix per model, which thus is robust. The more Gaussians are used for the helpers, the fewer training data is available for each of them – this is one reason why the WER constantly increases when using more Gaussians. Also, using only one Gaussian

[11]Step 1 (Sample Extraction) is only done once, as all following steps use the same samples; step 2 (Codebook Training) needs to be done once for each number of Gaussians.

	Number of Gaussians				
	1	**4**	**8**	**16**	**32**
Euclidian	30.3	33.0	33.7	34.2	34.6
Kullback-Leibler	30.1	33.6	34.1	35.3	35.5
Mahalanobis	30.2	33.3	34.2	34.4	35.1
Bhattacharyya	30.3	33.4	34.0	35.0	35.8

Table 6.1: Word Error Rates for different number of Gaussians

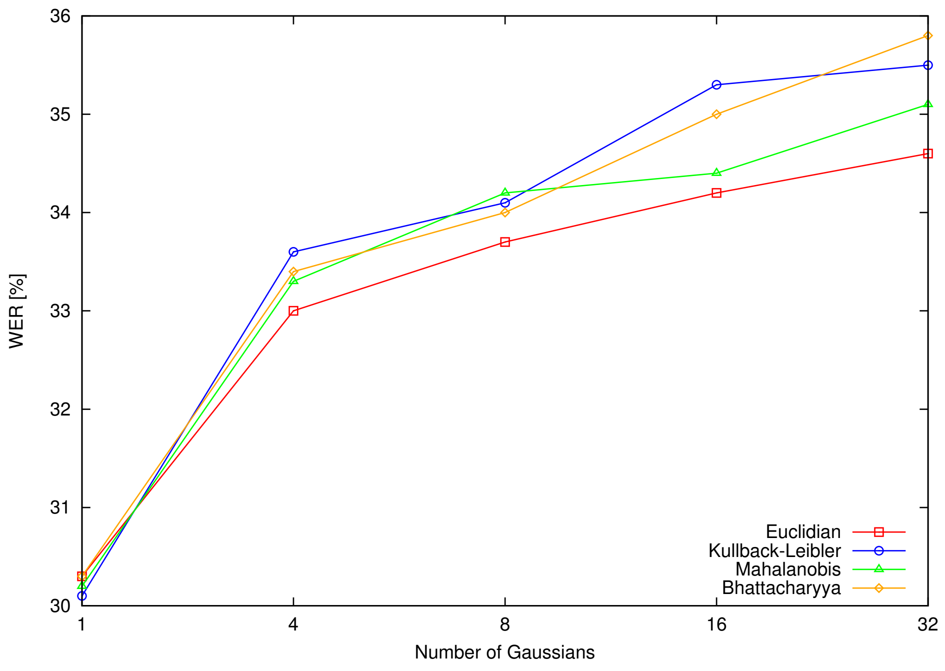


Figure 6.1: Word Error Rates for different number of Gaussians

for the helper model allows to use the simple merging of Gaussians for the helpers as presented in Section 4.2.1, which makes the successive training of helper models unnecessary.

On the other hand, the method to calculate the distance between GMMs as presented in Section 4.1.5 has its flaws: We calculate a weighted sum of the distances between all pairs of Gaussians of the two GMMs. This method does not take the actual distribution into consideration, but rather picks its single elements one after another as representatives for the whole – which in fact they are not. Summing these elements up still does not incorporate the distribution as a whole.

Furthermore, it is striking that the WER mainly depends on the number of Gaussians, but not much on the distance measure itself. This is another indicator that the weak point lies in the general method for calculating the GMM distance, but not in the single Gaussian measures.

It is interesting that the Euclidian distance performs best for all Gaussian mixtures: as it leaves out the covariances, it does not consider all the information of the Gaussians, so one would expect it to score less well. This indicates that the means of the Gaussians are more reliably trained than the covariances. Perhaps this could be solved by using more training data.

The described results and weaknesses all lead to one conclusion: The distance measures between single Gaussians may be established and broadly accepted by the research community, but using them on Gaussian mixtures does not lead to the desired results, at least not with the method we used. This is due to the fact that our method does not represent a reasonable distance between GMMs. Chapter 7 will present two approaches for further investigation on distance measures that instead directly work on Gaussian mixtures.

7. Conclusion and Outlook

In this work we presented different acoustic distance metrics and evaluated their effects on the quality of a divisive polyphone clustering algorithm using fully continuous acoustic models. We therefore compared the Euclidian distance, the Kullback–Leibler divergence, the Extended Mahalanobis distance and the Bhattacharyya distance to the entropy distance.

As Chapter 6 showed, the clustering resulted best in terms of the Word Error Rates in case of single Gaussian helper models – but scored significantly worse if the number of Gaussians was increased.

This can possibly be attributed to the fact that the helper models have more parameters to estimate while still having a constant amount of training data available. Moreover, our method to calculate the distance between GMMs has its disadvantages and may not be suitable to represent a reasonable distance between two GMMs.

For further investigations, the following two approaches may therefore be interesting:

1. Using a different method for the calculation of the distance between GMMs: We calculated it as the weighted sum of all pairwise distances between the single Gaussians of the GMMs. Another approach is to find pairs of minimal distances and only use them.

This means, we first calculate all pairwise distances between the Gaussians, then find the pair with minimal distance, the one with the second smallest distance (excluding both Gaussians from the first pair), and so on, and add up only these distances to get the total distance of the GMMs.

2. Using a distance measure, that does not use single Gaussians but rather measures the distance between GMMs as a whole. A good distance between two polyphones determines how differently the probability mass is distributed; therefore, an interesting measure for further investigations is to use the volume between the two distributions as a distance as showed in Figure 7.1(a).

The figure shows two GMMs and the (2-dimensional) volume between them. This volume represents a distance: If the distributions are similar to each other, they

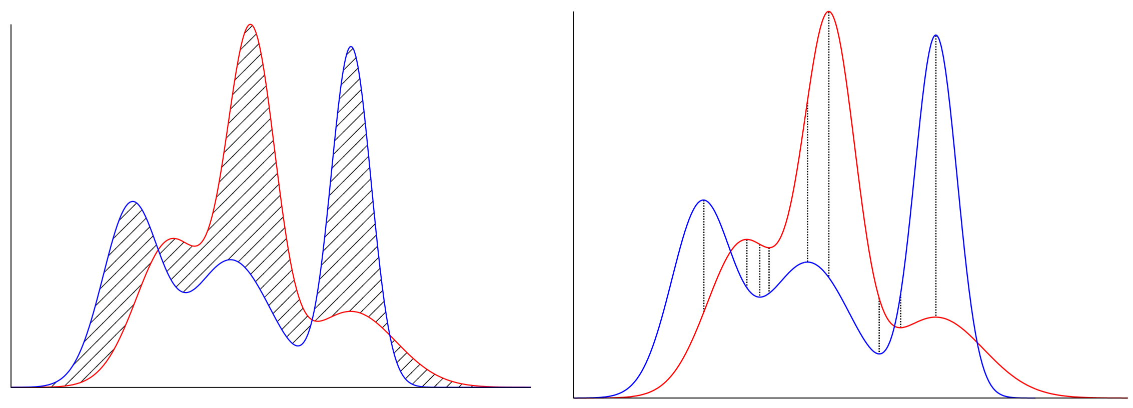

(a) Volume between two Gaussian mixtures (b) Sample points to estimate the volume

Figure 7.1: Volume between GMMs as a distance measure

largely overlap – so the distance is small. However, if they are distinct, the volume gets bigger. The maximal distance between two distributions is the sum of their single volumes, if they do not overlap at all[12].

This distance can be expressed as

$$d_{\mathrm{vol}}(\Gamma_1, \Gamma_2) = \int \left|\Gamma_1(x) - \Gamma_2(x)\right| \,\mathrm{d}x \tag{7.1}$$

As the Gaussian integral is involved in an analytical solution to this, it may be easier to use an estimation of the distance instead. One possibility is to use a set Ω of sample points and measure the distance only at these points:

$$d_{\mathrm{est}}(\Gamma_1, \Gamma_2) = \sum_{x \in \Omega} \left|\Gamma_1(x) - \Gamma_2(x)\right| \tag{7.2}$$

The sample set Ω can either be random or deterministic. Figure 7.1(b) uses the means μ of all involved Gaussians as a deterministic set of samples to visualize this estimation. Additionally to the means, one could include the inflection points and their "multiples" $\mu \pm n \cdot \sigma$, $n \in \mathbb{N}$ (in each dimension) in the set Ω in order to get more samples and thus a closer estimation.

Summing the results up, we can conclude that our experiments did not yield the desired results, but still offer potential for further research. As this chapter showed, there are still refinements and concrete approaches to exploit the outcome of polyphone clustering. Future research can possibly outrun the benchmark of the entropy distance by using the full potential of fully continuous acoustic models.

[12] Actually, there still will be an infinitesimal overlapping, as Gaussian distributions never reach zero.

A. Phonemes

The following listing presents the phonemes used in our system along with examples for them.

Phoneme	**Example Words**
AA	**a**rm, **a**rticle
AE	**a**venue, **a**xe
AH	**a**bout, **a**bove
AO	**awe**some, f**o**rce
AW	b**ou**nce, d**ow**n
AX	**a**ccount, **a**lert
AXR	capt**ure**, lit**er**
AY	m**i**ke, ps**y**cho
B	**b**rain, a**b**out
CH	**ch**ain, **ch**icken
D	**d**evelopment, **d**estiny
DH	**th**e, **th**ank
EH	**e**rror, **e**xcellent
ER	v**er**sus, t**er**m
EY	w**ei**ght, t**a**ke
F	**f**ilter, **f**lag
G	**g**old, **g**un
HH	**h**ack, **h**ammer
IH	h**i**story, **i**mage
IX	**i**llusion, **i**ntensive, ...**i**ng
IY	jeweller**y**, magaz**i**ne, ...t**y**
JH	ma**j**or, mer**ge**
K	mi**c**ro, **k**ill
L	**l**ong, **l**ife
M	**m**an, **m**anual
N	**n**ovel, **n**ice
NG	la**ng**uage, ba**nk**, ...i**ng**

Phoneme	**Example Words**
OW	b**o**ld, c**o**de
OY	app**oi**ntment, depl**oy**
P	**P**ittsburgh, **p**arty
R	**r**eason, **r**ecord
S	**s**enior, **s**etup
SH	**sh**ield, **sh**ort
T	**t**ime, **t**oday
TH	**th**umb, **th**eatre
UH	w**ou**ld, l**oo**k
UW	y**ou**, l**oo**se
V	o**v**er, pro**v**ider
W	q**u**een, **w**ay
XL	ab**le**, ang**le**
XM	rhyth**m**, touris**m**
XN	certai**n**, butto**n**
Y	**y**oung, **y**ear
Z	advi**ce**, i**s**, ...**s**
ZH	mea**s**ure, u**s**ual

B. Phoneme Sets

This listing shows the sets of phonemes used as questions in the clustering algorithm in Section 4.3.

When splitting a class of polyphones, a question concerning a phoneme set asks for each polyphone in the class if a specific phoneme of it is part of that phoneme set. For example "-1=LABIAL" asks, if the phoneme on the left of the center phoneme is part of the labial set. This can return *yes* or *no* and therefore splits the class into two new classes.

Phoneme Set	**Elements**
PHONES	PAD IY IH EH AE IX AX AH UW UH AO AA EY AY OY AW OW L R Y W ER AXR M N NG CH JH DH B D G P T K Z ZH V F TH S SH HH XL XM XN SIL GARBAGE +FILLER+ +BREATH+ +HUMAN+ +LAUGH+ +NOISE+
HUMANSND	IY IH EH AE IX AX AH UW UH AO AA EY AY OY AW OW L R Y W ER AXR M N NG CH JH DH B D G P T K Z ZH V F TH S SH HH XL XM XN
VOLATILE	AO EY AY OY AW OW L R Y W ER AXR M N NG CH JH DH B D G P T K Z ZH V F TH S SH HH XL XM XN
NOISES	GARBAGE +BREATH+ +FILLER+ +HUMAN+ +LAUGH+ +NOISE+
FILLERS	+FILLER+
BREATH	+BREATH+
HUMAN	+HUMAN+
LAUGH	+LAUGH+
NOISE	+NOISE+
NOISE2	+BREATH+ +FILLER+ +HUMAN+ +LAUGH+ +NOISE+
NOISE3	+BREATH+ +HUMAN+ +LAUGH+ +NOISE+
HUMANS	+BREATH+ +HUMAN+ +LAUGH+

Phoneme Set	**Elements**
SILENCES	SIL
CONSONANT	P B F V TH DH T D S Z SH ZH CH JH K G HH M N NG R Y W L ER AXR XL XM XN
CONSONANTAL	P B F V TH DH T D S Z SH ZH CH JH K G HH M N NG XL XM XN
OBSTRUENT	P B F V TH DH T D S Z SH ZH CH JH K G
SONORANT	M N NG R Y W L ER AXR XL XM XN
SYLLABIC	AY OY EY IY AW OW EH IH AO AE AA AH UW UH IX AX ER AXR XL XM XN
VOWEL	AY OY EY IY AW OW EH IH AO AE AA AH UW UH IX AX
DIPHTHONG	AY OY EY AW OW
CARDVOWEL	IY IH EH AE AA AH AO UH UW IX AX
VOICED	B D G JH V DH Z ZH M N NG W R Y L ER AY OY EY IY AW OW EH IH AO AE AA AH UW UH AXR IX AX XL XM XN
UNVOICED	P F TH T S SH CH K
CONTINUANT	F TH S SH V DH Z ZH W R Y L ER XL
DEL-REL	CH JH
LATERAL	L XL
ANTERIOR	P T B D F TH S SH V DH Z ZH M N W Y L XM XN
CORONAL	T D CH JH TH S SH DH Z ZH N L R XL XN
APICAL	T D N
HIGH-CONS	K G NG W Y
BACK-CONS	K G NG W
LABIALIZED	R W ER AXR
STRIDENT	CH JH F S SH V Z ZH
SIBILANT	S SH Z ZH CH JH
BILABIAL	P B M W
LABIODENTAL	F V
LABIAL	P B M W F V
INTERDENTAL	TH DH
ALVEOLAR-RIDGE	T D N S Z L
ALVEOPALATAL	SH ZH CH JH
ALVEOLAR	T D N S Z L SH ZH CH JH
RETROFLEX	R ER AXR
PALATAL	Y
VELAR	K G NG W
GLOTTAL	HH
ASPIRATED	HH
STOP	P B T D K G M N NG
PLOSIVE	P B T D K G
NASAL	M N NG XM XN
FRICATIVE	F V TH DH S Z SH ZH HH
AFFRICATE	CH JH

Phoneme Set	**Elements**
APPROXIMANT	R L Y W
LAB-PL	P B
ALV-PL	T D
VEL-PL	K G
VLS-PL	P T K
VCD-PL	B D G
LAB-FR	F V
DNT-FR	TH DH
ALV-FR	SH ZH
VLS-FR	F TH SH
VCD-FR	V DH ZH
ROUND	AO OW UH UW OY AW
HIGH-VOW	IY IH UH UW IX
MID-VOW	EH AH AX
LOW-VOW	AA AE AO
FRONT-VOW	IY IH EH AE
CENTRAL-VOW	AH AX IX
BACK-VOW	AA AO UH UW
TENSE-VOW	IY UW AE
LAX-VOW	IH AA EH AH UH
ROUND-VOW	AO UH UW
REDUCED-VOW	IX AX
REDUCED-CON	AXR
REDUCED	IX AX AXR
LH-DIP	AY AW
MH-DIP	OY OW EY
BF-DIP	AY OY AW OW
Y-DIP	AY OY EY
W-DIP	AW OW
ROUND-DIP	OY AW OW
LIQUID-GLIDE	L R W Y
W-GLIDE	UW AW OW W
LIQUID	L R
LW	L W
Y-GLIDE	IY AY EY OY Y
LQGL-BACK	L R W
X-LMN	XL XM XN

Bibliography

[AnHe04] J. Anguita and J. Hernando. Inter-Phone and Inter-Word Distances for Confusability Prediction in Speech Recognition. In *Natural Language Processing*, Volume 33. Spanish Society for Natural Language Processing, September 2004.

[BFOS84] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC. 1984.

[BoNA10] A. Boudlal, B. Nsiri and D. Aboutajdine. Modeling of Video Sequences by Gaussian Mixture: Application in Motion Estimation by Block Matching Method. *EURASIP Journal on Advances in Signal Processing* 2010(1), 2010.

[FiRo97] M. Finke and I. Rogina. Wide context acoustic modeling in read vs. spontaneous speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 1997. (ICASSP 1997).*, Volume 3, April 1997, p. 1743–1746.

[GoAr05] J. Goldberger and H. Aronowitz. A Distance Measure Between GMMs Based on the Unscented Transform and its Application to Speaker Recognition. In *Proceedings of Interspeech 2005.*, September 2005.

[GoGG03] J. Goldberger, S. Gordon and H. Greenspan. An efficient image similarity measure based on approximations of KL-divergence between two gaussian mixtures. In *Ninth IEEE International Conference on Computer Vision, 2003. Proceedings.*, Volume 1, October 2003, p. 487–493.

[IKHv00] B. Imperl, Z. Kačič, B. Horvat and A. Žgank. Agglomerative vs. tree-based clustering for the definition of multilingual set of triphones. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000. Proceedings. (ICASSP 2000).*, Volume 3, 2000, p. 1273–1276.

[Jeli05] F. Jelinek. Some of my Best Friends are Linguists. *Language Resources and Evaluation* 1(39), 2005, p. 25–34.

[LeBe05] V.-B. Le and L. Besacier. First Steps in Fast Acoustic Modeling for a New Target Language: Application to Vietnamese. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP 2005).*, Volume 1, March 18-23, 2005, p. 821–824.

[LeBe09] V.-B. Le and L. Besacier. Automatic Speech Recognition for Under-Resourced Languages: Application to Vietnamese Language. *IEEE Transactions on Audio, Speech, and Language Processing* 17(8), November 2009, p. 1471–1482.

[LeBS06] V.-B. Le, L. Besacier and T. Schultz. Acoustic-Phonetic Unit Similarities For Context Dependent Acoustic Model Portability. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2006. Proceedings. (ICASSP 2006).*, Volume 1, May 2006, p. I.

[LiTz00] S. Liapis and G. Tziritas. Image Retrieval by Colour and Texture Using Chromaticity Histograms and Wavelet Frames. In R. Laurini (Ed.), *Advances in Visual Information Systems*, Volume 1929 of *Lecture Notes in Computer Science*, p. 49–61. Springer Berlin/Heidelberg, 2000.

[LoSa01] B. Logan and A. Salomon. A music similarity function based on signal analysis. In *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*, August 2001, p. 745–748.

[MaBa96] B. Mak and E. Barnard. Phone clustering using the Bhattacharyya distance. In *Fourth International Conference on Spoken Language, 1996. Proceedings. (ICSLP 1996).*, Volume 4, October 1996, p. 2005–2008.

[MoTr06] M. Mohammad and W. Tranter. Comparing Distance Measures for Hidden Markov Models. In *Proceedings of the IEEE SoutheastCon, 2006.*, April 2006, p. 256–260.

[RiTJ00] L. Rigazio, B. Tsakam and J.-C. Junqua. An optimal Bhattacharyya centroid algorithm for Gaussian clustering with applications in automatic speech recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000. Proceedings. (ICASSP 2000).*, Volume 3, 2000, p. 1599–1602.

[Runn07] A. R. Runnalls. Kullback-Leibler Approach to Gaussian Mixture Reduction. *IEEE Transactions on Aerospace and Electronic Systems* 43(3), July 2007, p. 989–999.

[Shan48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal* Volume 27, 1948, p. 379–423.

[SiRS99] R. Singh, B. Raj and R. Stern. Automatic clustering and generation of contextual questions for tied states in hidden Markov models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 1999. Proceedings.*, Volume 1, March 1999, p. 117–120.

[SoBo01] J. J. Sooful and E. C. Botha. An acoustic distance measure for automatic cross language phoneme mapping. In *Proceedings of the Twelfth Annual Symposium of the Pattern Recognition Association of South Africa.* PRASA, 2001.

[Stü09] S. Stüker. *Acoustic Modelling for Under-Resourced Languages.* Dissertation, Universität Fridericiana zu Karlsruhe (TH), 2009.