Completely Unsupervised Phone Segmentation, Clustering and Classification

Master Intelligent Technology
Extended Research Paper

By

L.G.W. Taverne (Leo), s4340000
Dr. L.F.M. ten Bosch (Louis)
Prof. M.A. Larson (Martha)

September 2022

ABSTRACT

Automatic speech recognition (ASR) often relies at least partly on supervised learning. However, labelled data sets are not available for all tasks. For many types of ASR training, the availability of phone-level annotation of audio would be of substantial help. This paper presents and in part implements a completely unsupervised network for segmenting and classifying phones from a speech signal. A novel unsupervised phone segmentation method based on clustering is presented and achieved an accuracy of 80%. This paper also presents different clustering methods, and the results are discussed, looking at both individual phones and broad phone groups. We look at the advantages and disadvantages of using a clustering algorithm as a pre-step to unsupervised classification. We discuss the performance in detail and look at the effect of phone context. An unsupervised method to classifying the resulting cluster indices to phone labels is discussed. This method uses a generative adversarial network (GAN) that works without using parallel data. Finally, we discuss most recent advancements in GANs that can potentially be of use in this classification problem.

Table of Contents

1 Introduction

1.1 Aim

This project aims to design, implement and discuss a computational framework that exclusively uses unsupervised training for automatically recognizing phones from an audio corpus. This framework for phone recognition consists of three main steps, which are phone segmentation, phone clustering and phone classification. For the framework to be completely unsupervised, it is important that all these steps happen without supervision, so without the use of labelled speech data. Figure 1 shows a visualization of this framework.

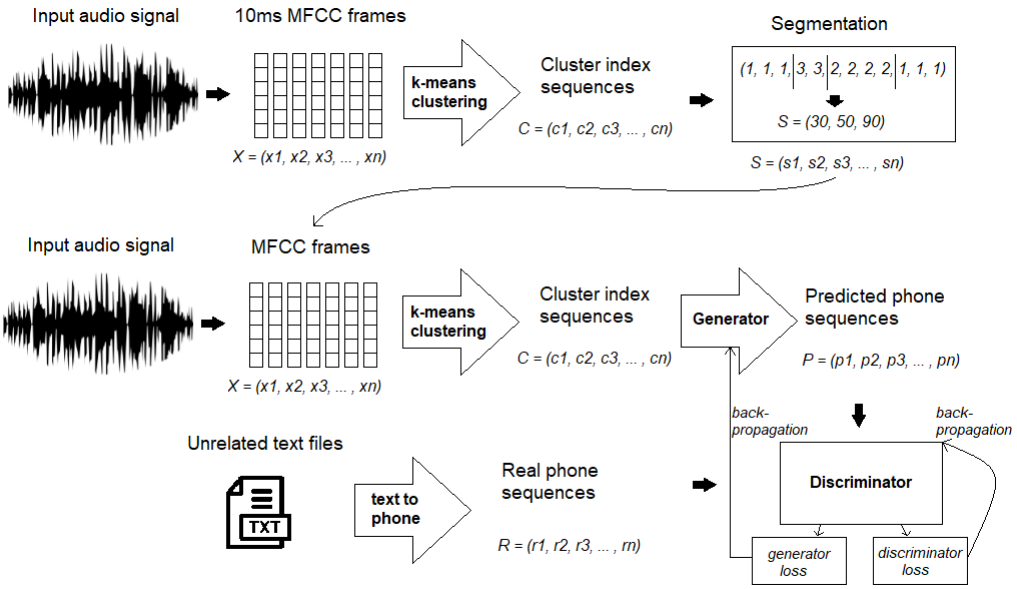

Figure 1: Visualization of the complete framework.

Next to the automatic recognition of words and sentences, the automatic classification of phones (i.e., speech sounds) is one of the major tasks in the research field of automatic speech recognition (ASR). Within ASR, unsupervised learning has been a key challenge for decades, since reliable and robust unsupervised approaches would greatly simplify training methods. Unsupervised training methods enormously broaden the availability of audio data material, which is especially interesting for the development of ASR systems in the case of under-resourced languages. In the past, research in ASR has often focused more on supervised learning methods, which have been the default for many decades. Generally, supervised algorithm still perform a lot better than unsupervised algorithms, however, the gap is slowly closing [1]. This thesis presents our computational framework using unsupervised

methods. We will show that this framework is able to automatically segment, cluster and classify phones from speech.

Our work is centered around one main research question. This question is how accurately can a completely unsupervised framework potentially recognize phones from a speech signal? To answer this question, we need to look at the three main parts that the framework is comprised of individually. The first important step here is the automatic segmentation of speech by hypothesizing potential phone-phone boundaries. So, the first sub-question is how accurately can phones (or phone-like units) be automatically segmented from speech data in an unsupervised way? To answer this question, we designed and implemented a new segmentation algorithm that can automatically segment phones from speech data without supervision. This segmentation algorithm uses k-means clustering on mel-frequency cepstral coefficients (MFCCs) of speech portions to find phone boundaries. Using k-means clustering, a cluster index is assigned to each 10ms speech portion of a speech file. The segmentation algorithm then postulates hard phone boundaries at the locations where cluster indices change. We will test the performance of this algorithm and compare it to other phone segmentation algorithms in the literature.

The second step is the clustering of the audio stretches resulting from the first step. This step involves the task of dividing the population of individual phones into a certain number of groups. The segmented stretches will then be transformed into sequences of cluster indices, where each sequence represents the consecutive phones in a sentence that is spoken. The second sub-question, which is related to this step, is how adequate or less adequate can phones be clustered without supervision? To answer this question, we also implemented an algorithm for clustering phones. We will use the MFCCs of phone recordings as input to a k-means clustering algorithm. We will compare clustering performance using different methods of data pre-processing. We will for example look at how windowing influences performance. Furthermore, we will also look at how well the clustering algorithm can distinguish broad phonetic groups, besides individual phones. Through this we can also learn how well a cluster sequence captures the overall structure of a sentence. The aim is to use this information to decide whether clustering is an adequate pre-step to classification.

The last step of our framework is the classification of the phones. That is, mapping the sequences of cluster indices to phone labels. This step will be dealt with theoretically by discussing possible methods using a Generative Adversarial Network (GAN). In a GAN, two neural networks are contesting with each other to generate more accurate predictions. Solving this problem using a GAN is inspired by a paper by Liu et al., which to our knowledge

is the only paper that attempts to perform the classification step using only unsupervised training [2]. The idea is that a GAN will then be able to generate phone label sequences based on cluster index sequences. The original plan in this thesis was to also implement this GAN-based network that is able to map the cluster indices to phone labels without parallel data. However, this turned out to take too much time to also implement for this project completely. Section A of the appendix describes our work that was carried out to attempt to implement the GAN. Instead, we will discuss the GAN theoretically and look at the latest research advances related to GANs and theorize about how these can be used for phone classification. Our third sub-question, which is related to this step, is could a GAN be used for classifying phones without the use of parallel training data? Since it is a relatively new and experimental method for classification, it will be informative to learn more about it. We will look at what a GAN is, how it can be used for classification and the data it needs for unsupervised training. We will also look at the strengths and potential weaknesses of a GAN in this context.

Main Question	How accurately can a completely unsupervised framework potentially recognize phones from a speech signal?
Sub-question 1	How accurately can phones be automatically segmented from speech data in an unsupervised way?
Sub-question 2	How adequate or less adequate can phones be clustered without supervision?
Sub-question 3	Could a GAN be used for classifying phones without the use of parallel training data?

Table 1: Research Questions

So, the overall aim of this research is to obtain a deeper understanding of how unsupervised phone recognition works and performs. We will do this by answering multiple research questions shown in table 1 that will help us in answering the main research question. The goal is to look at this from a more scientific perspective instead of just optimizing model parameters. We want to analyze the performance of the segmentation algorithm and the clustering algorithm on the level of individual phones and general phone classes. We will look at a new method of unsupervised classification using a GAN and analyze whether clustering is an adequate pre-step.

1.2 Motivation

Most ASR techniques use algorithms that rely at least partly on supervised learning and use large audio databases for the training process of acoustic models. For languages like English, for which there are immense audio corpora available, this nowadays results in high accuracy ASR systems in many conditions. Some state-of-the-art supervised ASR systems will be discussed in section 2.1. However, research on developing methods that rely on completely unsupervised training is still limited, for example in detecting and classifying phones from audio. Developing unsupervised methods for recognizing phones can be important for analyzing languages for which there is limited data available. The availability of a phone-level annotation of audio is very important for many types of ASR learning. There are many languages spoken by smaller populations from which there are no large corpora of data available, which means there is limited data that can be used for training. The same can be true for extinct languages or specific dialects. So, for low-resourced languages, unsupervised training can be very important for language analysis and the development of ASR systems and other speech-based information systems.

Other areas of interest for unsupervised systems include automated large-scale phonetic analyses of pathological speech or non-normal speech production in general. Recently, there has been increased interest in using machine learning algorithms within the study of speech pathology. These algorithms need to be able to handle challenges like annotator subjectivity and data sparsity [3]. These challenges can be addressed by unsupervised training methods, as these methods do not require annotated data. In a broader sense, unsupervised methods have already been shown to be able to accurately detect anomalous sounds in real sound environments [4].

There is also another, more cognitive rationale underlying the use of unsupervised methods for the development of audio decoding systems. By delving deeper into unsupervised learning, one may learn more about how humans acquire knowledge about language. This acquisition process is usually multimodal, and it is believed to be largely unsupervised. Although humans use information from multiple sensory modalities, there is no direct feedback needed after errors are made in order to help young infants understand what has been spoken. With humans, the mere exposure to languages causes the brain to make sense of the abstractions and generalizations of those languages [5]. This method, of automatically learning a set of symbolic units and mapping them, can also be extended beyond recognizing phones. Other problems like object recognition could also benefit from more research on unsupervised training techniques [6]. In these domains, research is often also more focused on supervised learning, whilst there are many low-resource

scenarios for which supervised learning is not an option or not preferred. Another example is automatic machine translation for rare languages. Some studies have shown that with low-resource and rare languages, unsupervised translation still performs poorly [7]. Therefore, analyzing new methods for unsupervised learning can be very valuable.

We also want to learn whether clustering can be an adequate method to represent phones prior to labelling. We opted for representing phones as a cluster index instead of more complex vector representations as it greatly reduces the dimensionality of the data. This is beneficial when using a GAN for labelling, as GANs work highly inefficient in high dimensional spaces [8]. However, the downside of reducing the dimensionality of the data can be that information is lost. Clustering simplifies the problem to a sequence-to-sequence translation problem. In our research we would like to find how much information about phones a clustering algorithm can capture. We will look at how adequate phonotactic information can be captured by a clustering algorithm. Past research has shown that clustering algorithms can be used to discover patterns in speech signals, which can be used to acquire an inventory of lexical units [9]. We will also analyze the ability of the cluster algorithm to distinguish broad phonetic groups. Learning about this will be important in determining whether clustering is an adequate pre-step to classification. By analyzing this we also learn more about the strengths and weaknesses of the framework and where the framework can potentially be improved. So, this research should also result in a deeper understanding of the inner workings of an unsupervised system and its performance in relation to phone recognition. Information about whether the clustering algorithm is able to distinguish general phone groups from each other and whether it is able to retain information about phonetic structure and patterns within sentences is essential. This is especially important if we want to use the output of the clustering algorithm as input for a GAN, since the GAN will use phonotactic information in order to generate possible phone label sequences.

2 Background

In this section, we will discuss some past literature and give some background information related to unsupervised phone recognition. In the first subsection, we will discuss the performance of some current state-of-the-art ASR algorithms. We will also look at the difference between phone segmentation, phone classification and phone recognition. In the second subsection, we will give some more background on unsupervised ASR and its importance. In the third subsection, we will give some background on GANs and how they can be used for unsupervised classification.

2.1 Supervised Phone Recognition

Humans have the ability to automatically process vast amounts of unlabeled speech from an early age onwards. We are able to learn and recognize phones, detect word boundaries, and make sense of new words. In the field of ASR, tremendous advances have been made over the last couple of decades. These days ASR technology is widely available to most people and its accuracy keeps improving. In certain scenarios, ASR technology is already close to being on par with human speech recognition [10]. Speech recognition consists of multiple hierarchical levels that can be individually evaluated. The lowest level consists of individual speech sounds, called phones, which we will focus on in this research. Other levels include words, phrases and sentences. Phones are the actual individual sounds you can hear in an utterance. The mental representations of sounds in a word are called phonemes. The difference between phones and phonemes can be explained by looking at the words "matter" and "madder". These words are composed of different phonemes /d/ and /t/, however, in American English, both words can be pronounced with the phone [d]. Phonemes can be specific to a certain language, while phones are not specific to any language. In our research, we will be looking at the phonetic transcriptions of sentences provided by TIMIT.

State-of-the-art supervised models can reach a word level accuracy of up to 94.4% on voice search tasks [11] using an LSTM with several optimizations. This means 95% of the words can be automatically recognized from an audio recording. For spoken digit recognition, supervised algorithms have even reached an accuracy of 99.6% [12]. Correctly recognizing phones, however, is a different and more difficult task. Phone recognition only reaches an accuracy of up to roughly 80% when using supervised methods. Seltzer et al. achieved this performance by using deep neural networks and multi-task learning [13]. Their network performs both the primary classification task and some secondary tasks. Secondary tasks based on the prediction of left and right phonetic context showed the best results. We will therefore be looking at phonetic context and how it can be used for our own implementation.

Phone recognition consists mainly of two sub-problems, which are phone segmentation and phone classification. Phone classification is the problem where you want to label a speech signal that has already been segmented, whereas with phone recognition you do not know the phone boundaries. With phone classification the accuracy can reach about 85% using an LSTM network [14]. Supervised phone segmentation achieved a 93% agreement within 20ms on the TIMIT corpus [15]. Human annotators achieved a similar average agreement on various corpora including TIMIT [16].

An overview comparing the performances achieved for digit, word and phone recognition can be found in table 2. Good phone recognition and classification are very important in many applications. For example, large vocabulary ASR models rely on the performance of a phone recognizer. Other applications that use phone recognition include language recognition, speaker identification and music identification [17]. Therefore, it is an important aspect of ASR to conduct more research on. In this subsection, we only discussed the performances of supervised methods, next we will discuss unsupervised phone recognition.

Paper	Training	Level	Accuracy
Li, Jinyu	Supervised	Digit Recognition	99.6%
Chiu et al.	Supervised	Word Recognition	94.4%
Yuan et al.	Supervised	Phone Segmentation	94%
Michalek et al.	Supervised	Phone Classification	85%
Seltzer et al.	Supervised	Phone Recognition	79.8%
Yue et al.	Unsupervised	Phone Segmentation	83%
Abdullah et al.	Unsupervised	Phone Segmentation	80.3%
Liu et al.	Unsupervised	Phone Classification	36%

Table 2: Overview of performance achieved by different ASR methods

2.2 Unsupervised Phone Recognition

Research is also being done on developing unsupervised methods, so learning from unlabelled data only. Most of these methods, however, only attempt to automatically organize data in distinct groups. These groups then remain unlabelled. In other words, clustering the phones into distinct groups without adding the corresponding labels to them. It still proves to be much harder to also map these phones accurately to real phones when no parallel data is used. Nevertheless, an interesting method has been proposed to achieve this. In a paper we mentioned earlier, the authors proposed a method using a Generative Adversarial Network (GAN) to automatically discover and map phones to real phonemes [2]. This method achieved an accuracy of 36% in detecting the correct phones directly from speech. This result shows promise, but there is also much room for improvement. Interestingly, the same performance could be reached with only 0.1% of the data by using a supervised method. Also, the authors did not manage to automatically segment the audio into individual phones and instead used manually annotated phone boundaries. In our research, we would like further explore the idea of using adversarial learning for phone classification and see how it can be improved. We also want to implement an algorithm for automatic phone segmentation without supervision.

In other research successful unsupervised methods have been developed for the automatic segmentation of phones. An overview is given of the performance of different unsupervised segmentation algorithms by Scharenborg et al. [18]. On TIMIT speech data, detection rates of different automatic segmentation methods range between roughly 68% and 74% of boundaries without over-segmentation. There is over-segmentation when an algorithm postulates more boundaries than there are according to the TIMIT data, which needs to be prevented. In very recent studies from Interspeech self-supervised methods have even achieved a precision of up to 83% on the TIMIT data set, however, over-segmentation rates are not always reported [19]. Another paper showed promise in using k-means clustering as a pre-step to segmentation [20]. The authors use k-means clustering to divide speech segments into vowel regions and consonant regions and segment between the two regions. The research shows promise with a reported accuracy of 80%. The authors however did not report how the accuracy was defined and they only used a limited data set of 100 words. This paper does however motivate us to use clustering as a pre-step to segmentation for our own implementation. We want to improve upon this research by experimenting with more than two clusters and using MFCCs as input data. We will also be conducting the experiments on a larger data set (TIMIT) with sentences instead of isolated words.

So, it has been shown that it is possible to automatically detect phone boundaries without supervision with decent accuracy. Nevertheless, supervised methods still perform a lot better with up to 94% accuracy [15], so there is still room for improvement. There is still a long way to go for unsupervised methods to get close to the performance of supervised methods. Therefore, it is important to still attempt to develop and improve unsupervised methods. There are many scenarios in which supervised methods are simply not feasible, for example, because there is little or no parallel training data available. This is especially the case with ASR-related problems, where there is generally a lot of data available in English but not necessarily in other languages. There are many low-resource languages and dialects for which unsupervised learning is the only option. Therefore, unsupervised learning is an important area to conduct further research on and potentially improve existing methods.

2.3 Generative Adversarial Networks

Maybe the most challenging part of this problem is to classify the cluster indices to phone labels without the use of parallel data. So, we need to translate sequences of cluster indices to sequences of phone labels. Recently, there have been impressive advances in unsupervised translation tasks. Conneau et

al. managed to achieve an accuracy of up to 83.3% in word translation tasks without the use of parallel corpora, which is on par with supervised methods on the same tasks [21]. They use adversarial learning to align embeddings of the source language with embeddings of the target language. This creates a latent space from which embeddings can be decoded to both the source and target language. The success of adversarial learning in translation tasks motivates us to also use adversarial learning for our cluster index to phone label translation problem. A method to do this using adversarial learning is already described by Liu et al. [2]. They propose using a generative adversarial network that generates phone labels from cluster indices. GANs are a relatively new form of unsupervised machine learning, first designed by Ian Goodfellow et al. in 2014 [22]. A GAN is composed of two separate neural networks, a generator and a discriminator. These two networks compete, where the generator attempts to create something and make it look as real as possible, and the discriminator tries to tell fake from real apart. In other words, the goal of the generator is to maximize the classification error of the discriminator, and the goal of the discriminator is to minimize the classification error. So, the weights in both networks are updated accordingly to achieve this goal. The two networks are trained simultaneously, where the aim is that the generator can eventually generate examples that are impossible to tell apart from the real input. Figure 2 shows a visualization of a GAN. GANs have been very successful recently in various tasks, for example in generating photo-realistic images of people and animals that are nearly impossible to distinguish from real [23].

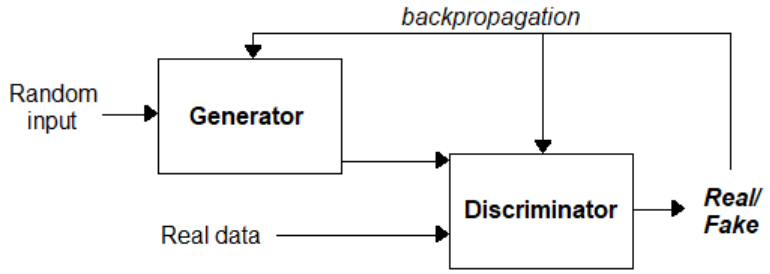

Figure 2: Visualization of a Generative Adversarial Network

When using a GAN for phone classification we do not have to use parallel training data, therefore the process remains unsupervised. We cannot use pairs of the input audio files and the transcribed phones, since training would then be supervised. Consequently, there are no direct pairs of cluster indices and phone labels for training. Instead, a separate corpus of data can be used as the target data for training, a corpus that only consists of phone label sequences. By using separate data the training remains unsupervised.

This separate data set can be any collection of English texts, which can be translated into sequences of phones by using a lexicon. Lui et al. for example used text sentences extracted from online news [2]. The criteria are that the texts are in English and contain overlapping lexicon with the sentences in the TIMIT data set. The goal is then to deduce which clusters belong to which phone labels from the structure of unrelated sequences of phone labels. The GAN uses phonotactic information to infer the phone labels from a sequence of cluster indices. So, the idea of solving this problem using a GAN, is because GANs can work without the use of directly labelled data. GANs are also very robust against misaligned input data [24]. Since in this case the input data is first automatically segmented and then clustered it can contain a lot of errors. So, a robust network is important. This will all be discussed in more detail in section 5.3.

Putting this to practice, however, was not possible during this research due to time limitations. The initial attempts to implement the network were very time-consuming and we decided to first focus on segmentation and clustering (see appendix, section A). Instead, the GAN will be discussed theoretically in the discussion section. We will also look at the newest advancements in GANs and discuss whether these are suitable in the context of phone mapping. This will then pave the way for future research where such a network can be implemented.

3 Methods

3.1 Data

The experiments will be conducted using English sentence-long audio fragments as input. The speech data will be extracted from the TIMIT Speech Corpus [25]. This data set consists of recordings of 630 speakers in 8 major American-English dialects. Each speaker reads ten phonetically rich sentences. All sentences are manually transcribed at phone level. For each sentence, the phone boundaries have been manually decided and verified and a corresponding phone symbol was added.

Phone	Count	Pct	Phone	Count	Pct
closure	32711	13.6	dx	3649	1.5
ih/ix	18347	7.6	p	3545	1.5
silence	15943	6.6	zh/sh	3259	1.4
n/en/nx	11874	4.9	ay	3242	1.3
s	10114	4.2	uw/ux	3213	1.3
iy	9663	4.0	f	3128	1.3
l/el	9451	3.9	ey	3088	1.3
r	9064	3.8	b	3067	1.3
ax/ah/ax-h	8634	3.6	ow	2913	1.2
ao/aa	8293	3.4	hh/hv	2836	1.2
er/axr	7636	3.2	g	2772	1.1
k	6488	2.7	v	2704	1.1
t	5899	2.4	y	2349	1.0
m/em	5600	2.3	ng/eng	1787	0.7
ae	5404	2.2	jh	1581	0.7
eh	5293	2.2	ch	1081	0.4
z	5046	2.1	th	1018	0.4
q	4834	2.0	oy	947	0.4
d	4793	2.0	aw	945	0.4
w	4379	1.8	uh	756	0.3
dh	3879	1.6			

Table 3: Phone distribution of train and test data combined

There are 61 phone symbols present in the TIMIT database, these are mapped down to the 41 individual phones shown in table 3. Several allophones are folded together according to common procedure [26]. This is done to prevent evaluation from being hurt by linguistically non-significant variants of phones belonging to different categories. Also, all closures (bcl,pcl,dcl,tcl,gcl and kcl) and silences (epi, pau, h#) are folded together. These phone labels and the start and finish time from each phone will be used for evaluation.

Broad Group	Group	Phones
Vowel	Vowel	iy,ih,eh,ey,ae,aa,aw,ay,ah,ao,oy, ow,uh,uw,ux,er,ax,ix,axr,ax-h
	Semivowel	l,r,w,y,hh,hv,el
Closure	Closure	bcl,dcl,gcl,pcl,tcl,kcl
Fricative	Fricative	s,sh,z,zh,f,th,v,dh
	Affricate	jh,ch
Nasal	Nasal	m,n,ng,em,en,eng,nx
Silence	Silence	pau,epi,h#
Stop	Stop	b,d,g,p,t,k,dx,q

Table 4: Phone groups and broad phonetic groups

The analysis will also focus on the performance within broader phone groups. The subdivision of broad phone groups is shown in table 4. These broad phonetic groups are based on TIMIT documentation and common subdivisions in past research [27].

3.2 Unsupervised Phone Segmentation

To detect the boundaries between individual phones a new method based on k-means clustering is developed. In this step we are not clustering phones yet as a pre-step for classification, now we are using a clustering algorithm as a pre-step for segmentation. The method is based on clustering small segments of the audio files and placing the boundaries at places where the cluster changes. The algorithm postulates hard phone boundaries between adjacent speech segments that have different cluster indices. Figure 3 shows a simplified visualization of this process and algorithm 1 shows the pseudo-code. The idea is that a clustering algorithm is quite good at distinguishing general groups of phones from each other [28]. So, whenever a short audio segment is part of a certain phone, the algorithm will cluster it with the matching phone group. When using a low number of clusters, the algorithm should be able to distinguish for example vowels, consonants and silences from each other with decent accuracy. This means that whenever one phone changes to another phone, and they are in different groups, the cluster will change at that point. When using a too large number of clusters we expect the cluster indices to change at too many places causing over-segmentation. We will experiment with different numbers of clusters to find the optimal number. However, a possible weakness could be that the algorithm does not detect boundaries between two adjacent phones that belong to the same general phone group.

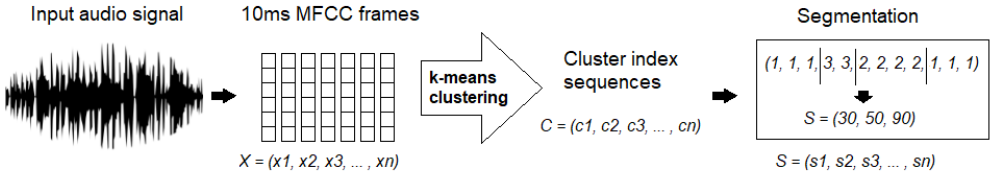

Figure 3: Visualization of the segmentation process.

Algorithm 1 Segmentation Algorithm

for sentence in data set **do**
  Split sentence in 20ms partitions at interval of 10ms.
  Create an MFCC with 13 coefficients for each partition.
  Perform k-means clustering on MFCCs
  $C = (c_1, c_2, ...c_n)$ are cluster indices for each consecutive partition
  Time in ms from last split $t = 0ms$
  Last cluster $c_{last} = -1$
  for c_i in C **do**
    if $i > 0$ **then**
      $c_{last} = c_{i-1}$
    end if
    if $t > 160ms$ **then**
      add boundary at current position
      $t = 0ms$
    end if
    if $c_{last} \neq c_i$ & $t > 20ms$ **then**
      add boundary at current position
      $t = 0ms$
    end if
    $t+ = 10ms$
  end for
end for

The input data to the clustering algorithm are the Mel-frequency Cepstral Coefficients (MFCCs) of the audio file. More specifically, an MFCC with 13 coefficients is made with a window size of 20ms at intervals of 10ms. This means there is an overlap of 10ms with each consecutive MFCC frame. Then each MFCC frame is clustered using the k-means clustering algorithm. This results in a sequence of cluster indices at an interval of 10ms spacing. Within this cluster sequence, a boundary is postulated at each point where a cluster is different compared to the cluster before it. If there was already a boundary detected at less than 20ms difference, then it will not create another boundary, because phones usually do not take less than 20ms. Another rule is that a

boundary will automatically be created whenever there is a segment of more than 160ms without a boundary because phones usually do not last longer than this. Using phone duration in the decision process for automatic speech segmentation has been proven to increase performance in multiple papers. Stober et al. first concluded that incorporating hypotheses about segment duration significantly increases the quality of automatically placed segment boundaries [29]. Scharenborg et al. concluded that one of the main problems in unsupervised segmentation algorithms is related to segment duration [18]. Algorithms have difficulties in dealing with segments that differ in duration. They propose to use an approach that also keeps track of the time since the last hypothesized boundary and to use this to only insert a boundary when the right conditions are met. This method was recommended for future research, and we therefore opted for this strategy in our approach.

Next, an important part is the performance evaluation. There need to be pre-defined rules on when a boundary is considered correct. As described by Scharenborg et al. and many others, it is not straightforward to compare the performances of different unsupervised speech segmentation algorithms. One of the reasons is that the true phone boundary is often debatable, as people often place the boundaries at slightly different positions. Several papers show that speech is annotated differently by different annotators. In an experiment using a German and French speech corpus, research shows that there is a mean deviation of 7.6ms between annotators [30]. However, 93% of the boundaries were agreed upon within a window of ±20ms. In another paper, the authors showed that, even when the same annotator is asked to annotate the same speech fragments after a three-month interval, the results will be different [31]. In this case, 96% of the boundaries were within a window of ±20ms. So, although there is no hard agreement on the exact placement of phone boundaries, there is usually an operational definition of using a tolerance window of ±20ms. Therefore, the evaluation will also be based on a tolerance window of ±20ms. This window is also used in most research concerning phone segmentation. The performance will then be computed using the following formula:

$$\text{CDR} = \frac{\text{nr\_correct\_boundaries}}{\text{total\_nr\_boundaries}} \times 100 \tag{1}$$

This number is called the correct detection rate (CDR), which is the percentage of boundaries that are detected within the tolerance window. For the number of correct boundaries, only one predicted boundary is counted as correct for each ground truth boundary. The total number of boundaries is the total number of boundaries in the ground truth. One flaw, however, when

only using the CDR as a performance indicator is that it will automatically be very high when the algorithm simply detects boundaries everywhere. Therefore, another value needs to be considered that makes sure that the algorithm does not over-generate phone boundaries at every place. This value is usually called the over-segmentation rate (OSR):

$$\text{OSR} = \left(\frac{\text{nr\_found\_boundaries}}{\text{nr\_true\_boundaries}} - 1\right) \times 100 \tag{2}$$

This number shows the percentage of over- or under-segmentation compared to the number of boundaries in the ground truth. When negative, this number indicates under-segmentation, meaning the algorithm found fewer boundaries than present in the data. The goal is to get this number as close to zero as possible and especially not too high.

3.3 Phone Clustering

The next step is clustering the resulting segments as a pre-step to classification. In order to classify the phones in an unsupervised manner the phones first need to be clustered. We will be using k-means clustering because it works in linear time complexity and can therefore handle large data sets with many dimensions in a time-efficient way. Other cluster methods like hierarchical clustering would not be suitable for the large data sets with many dimensions we are working with. With clustering, the most important aspect is how the input data is prepared. There are many ways in which data can be extracted from a speech signal and presented to an algorithm. Windowing is also very important, so the length of a speech segment that is used as input to the clustering algorithm to represent each phone. Two main techniques are tested and will be described here, the sliding window technique and the averaging technique. Figure 4 shows a simplified visualization of the segmentation process.

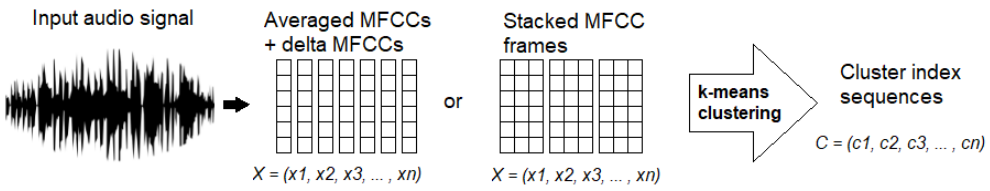

Figure 4: Visualisation of clustering using the sliding window technique.

For the first method, a sliding window is used. First, the entire speech signal is transformed into 10ms MFCC frames. In this case, each frame is represented by a vector containing 13 coefficients, together representing the spectral characteristics of a 10ms part of speech. A number of consecutive frames within a certain window appended together represents a phone. The input data to the clustering algorithm consists of these windows of MFCC frames. We will look at different frame windows to represent phones and measure performance using the evaluation methods described later in this section. Deciding what windowing to use will be essential for this method. We will also look at the effect of using a different number of clusters. Furthermore, we will look at a-symmetric configurations of frames, to discern whether there is a difference in the importance of information closer to the onset compared to the end of a phone. This will all give us a better idea of how context influences performance.

The second method works in a slightly different way, as the extraction of features is done differently. We call this method the averaging technique. With this method, the speech signal is first split into individual phones according to the phone boundaries provided by TIMIT. Boundaries provided by TIMIT are still used to simplify the process, but the boundaries provided by the automatic segmentation algorithm can be used in the future. After segmenting the phones, for each phone, an MFCC frame is created that is averaged over the entire phone length. In its simplest form, each phone is then only represented by a vector containing one averaged MFCC frame. However, we also want to test the added benefit of adding an averaged delta MFCC and delta-delta MFCC to the input vector. This means the first difference coefficients and the acceleration coefficients are also added to the input data. These coefficients give more information about how the speech signal changes and could therefore increase performance.

We will be looking at how well the individual phones described in table 3 are clustered using these methods. Then, using the best performing method, we will also be looking at how well the broad phone groups described in table 4 can be clustered. We will also create a clustering tree by splitting the data set using the clustering algorithm. This clustering tree will be a good visualization of how certain phones and phone groups are related to each other according to the clustering algorithm. The tree will show how similar or dissimilar phones are by means of where they are positioned in the tree.

Next, we will discuss the evaluation of the clustering algorithm. Audio segments are not assigned with phone labels yet, instead, each segment is assigned with a cluster index. Therefore, it is not immediately evident whether a clustering technique is adequate. A common method for cluster evaluation is computing the purity. The purity is computed by assigning each

cluster with the class that is most frequent in the cluster and then counting the percentage of correctly classified items. Equation 3 shows the formula for purity with N as the number of data points, k the number of clusters and t_j is a classification with the max count for cluster c_i

$$Purity = \frac{1}{N} \sum_{i=1}^{k} max_j |c_i \cap t_j| \tag{3}$$

However, the number of clusters can differ a lot from the number of phone classes. When using more clusters, the purity automatically increases, so it does not always represent cluster quality correctly. A better way to measure the agreement between two label assignments on the same data set is the Adjusted Mutual Info Score (AMIS) [32]. This is the Mutual Info Score (MIS), but adjusted for randomness. For two label assignments U and V, the MIS is defined as:

$$MIS(U, V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N|U_i \cap V_j|}{|U_i||V_j|} \tag{4}$$

The entropy is defined as:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \tag{5}$$

And the AMIS is defined as follows (with H denoting the entropy and $E(MIS)$ denoting the expected MIS):

$$AMIS(U, V) = \frac{MIS(U, V) - E(MIS(U, V))}{avg(H(U), H(V)) - E(MIS(U, V))} \tag{6}$$

The AMIS will be 1 with a perfect labelling and 0 when all classes are split among different clusters as would be expected by chance. This metric is ideal in this situation because it is independent of the absolute values of the labels. So, even if clusters are labelled differently than the actual phone labels, this will not affect the score. The score is also independent of the number of clusters used, so it is an accurate representation of cluster quality. This makes AMIS also a good score to determine if clustering is a good

pre-step to classification. In other words, determining whether the cluster index sequences can be adequate input to a GAN. This is because a high AMIS means the cluster indices capture the overall phonetic structure of the sentences. This phonotactic information is very important since it is used by the GAN to determine the correct phone labels.

A second evaluation method that will be used is with the help of a supervised classification algorithm as shown in figure 5. The goal in this paper is to use an unsupervised algorithm for classification, but a supervised algorithm can be used for evaluation purposes. Training the cluster index sequences with the ground truth phone labels using a supervised algorithm is very easy to implement. If clustering works as an adequate pre-step to a supervised algorithm, it indicates that the cluster sequences contain information from which the phone labels can be deduced. This means that an unsupervised algorithm can potentially also use this information from the cluster sequences.

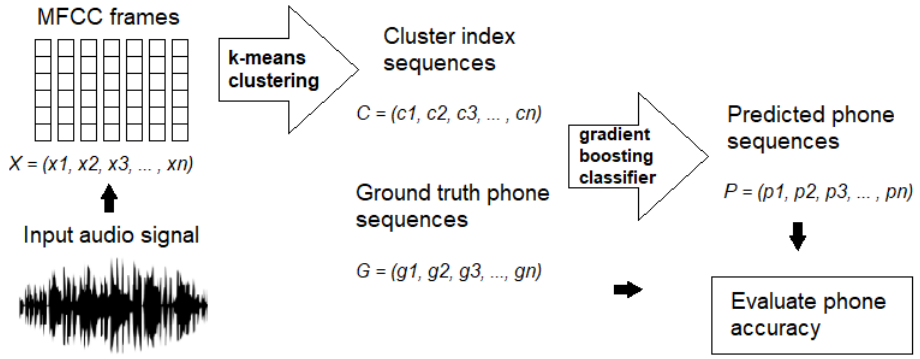

Figure 5: Evaluation using a supervised classifier after clustering.

The cluster indices can be trained on the phone labels provided by TIMIT. This can be done by matching a window of cluster indices with the corresponding phone label in the middle of the window. The supervised algorithm predicts the phone labels from a window of cluster indices. A larger window will mean more context is used to predict the label. The accuracy of the supervised algorithm can then be used for evaluation. This will give an upper bound to the possible classification accuracy when using the cluster indices as input data. It is taken as an upper bound assuming an unsupervised algorithm does not perform higher than a supervised algorithm. As for the supervised algorithm, a gradient boosting classifier is used because it has high accuracy, and works with categorical data as input. So the cluster index sequences can be used as input data without further pre-processing. We can then look at the percentage of correctly predicted phone labels. We will

match the TIMIT labels with different windows of cluster indices to discern the effect of using more context.

3.4 Unsupervised Phone Mapping

As mentioned before, the last step will not be implemented, we will instead look at it theoretically. We will discuss how a GAN could be used to map cluster indices to phone labels, by learning from a separate corpus of phone sequences. We will also focus on a newer variation of a GAN called seqGAN [33]. This new variation shows many improvements compared to a conventional GAN in the context of sequence generation. In the discussion section, we will discuss how seqGAN works and why it could be used for generating phone sequences.

4 Segmentation and Clustering Results

4.1 Unsupervised Segmentation

Table 5 shows the correct detection rate, mean deviation in milliseconds, and over-segmentation rate when using a different number of clusters in the segmentation algorithm. The CDR is based on a tolerance window of $\pm$20ms. The table shows the performance of the segmentation algorithm when using two clusters and up to a hundred clusters. As a reference, the first method is done without using a clustering algorithm and shows performance if boundaries were placed randomly. With this method a boundary is simply placed at intervals of 70ms, starting at 70ms, 140ms, etc.

Method	Mean Deviation	CDR	OSR
Fixed interval	21.5ms	49.2%	-0.9%
2 Clusters	34.55ms	52.3%	-32.8%
3 Clusters	20.2ms	69.6%	-16.4%
4 Clusters	16.3ms	76.5%	-7.0%
5 Clusters	14.6ms	80.0%	0.6%
6 Clusters	13.6ms	82.1%	6.0%
7 Clusters	12.9ms	83.6%	10.7%
8 Clusters	12.4ms	84.7%	14.5%
9 Clusters	12.0ms	85.6%	17.9%
10 Clusters	11.7ms	86.5%	20.7%
20 Clusters	10.1ms	90.5%	40.5%
50 Clusters	8.7ms	94.9%	73.4%
100 Clusters	8.1ms	97.5%	105.6%

Table 5: Segmentation performance using different numbers of clusters in the segmentation algorithm.

The next table shows the method with 5 clusters from table 5, but now with different tolerance windows.

Tolerance Window	±5ms	±10ms	±20ms	±30ms	±40ms
CDR	29.3%	53.5%	80.0%	87.4%	91.4%

Table 6: Segmentation performance using different tolerance windows

The next three tables show all the phone transitions in terms of the phonetic groups they belong to according to the TIMIT labels. The subdivision of individual phones within the phonetic groups can be found in table 4 in section 3.1, along with more information about the phone groups. Table 7 shows the total number of transitions in the test data set. These numbers can be useful as they show how many times certain transitions appear in the data set in proportion to other transitions. The row determines the first phone and the column the second phone. For example from the table can be deduced that there are 6045 transitions from a semivowel/glide to a vowel in TIMIT. Table 8 shows how many of these transitions are correctly segmented by the segmentation algorithm at a tolerance window of ±20ms. We use k-means clustering with five clusters, which has a CDR of 80% on the entire test set. Table 9 shows the corresponding percentages of correctly detected transitions for each group. From these tables, we can determine the detection rates of boundaries adjacent to different phone groups, and the number of occurrences in the data set.

	Vowel	SV/Gl	Nasal	Fric	Affr	Stop	Clos	Sil
Vowel	1497	3731	3969	4477	0	1451	5211	575
SV/Glide	6045	387	154	360	0	112	623	141
Nasal	2213	400	79	662	14	254	1187	295
Fricative	4211	595	200	307	1	98	1331	981
Affricate	436	17	13	23	0	4	69	69
Stop	6195	1839	146	565	0	87	51	293
Closure	98	208	219	667	582	6455	8	253
Silence	216	645	324	663	34	715	10	0

Table 7: Total number of phone transitions where the row shows the preceding phone and the column the succeeding phone.

	Vowel	SV/Gl	Nasal	Fric	Affr	Stop	Clos	Sil
Vowel	502	1451	3370	4176	0	1091	5091	350
SV/Glide	3257	240	126	338	0	72	598	89
Nasal	1879	182	12	527	13	208	781	155
Fricative	3957	514	173	162	0	94	1232	815
Affricate	411	10	13	4	0	4	66	50
Stop	5305	1560	123	404	0	81	46	185
Closure	81	153	177	595	554	6232	6	73
Silence	208	556	285	620	34	703	7	0

Table 8: Number of correctly detected phone transitions where the row shows the preceding phone and the column the succeeding phone.

	Vowel	SV/Gl	Nasal	Fric	Affr	Stop	Clos	Sil
Vowel	33.5	38.9	84.9	93.3	-	75.2	97.7	60.9
SV/Glide	53.9	62.0	81.8	93.9	-	64.3	96.0	63.1
Nasal	84.9	45.5	15.2	79.6	92.9	81.9	65.8	52.5
Fricative	94.0	86.4	86.5	52.8	0.0	95.9	92.6	83.1
Affricate	94.3	58.8	100.0	17.4	-	100.0	95.7	72.5
Stop	85.6	84.8	84.2	71.5	-	93.1	90.2	63.1
Closure	82.7	73.6	80.8	89.2	95.2	96.5	75.0	28.9
Silence	96.3	86.2	88.0	93.5	100.0	98.3	70.0	-

Table 9: Percentage of correctly detected phone transitions where the row shows the preceding phone and the column the succeeding phone.

4.2 Clustering

The performance of the clustering algorithm is first tested using the sliding window technique. In table 10 the results are shown when using a sliding

window of 10ms, where a 13 coefficient MFCC frame is created for every 10ms. Multiple frames can then be stacked and used as input for the k-means clustering algorithm. In this test 41 clusters are used, the same number as phones used for evaluation. The table shows the adjusted mutual info score (AMIS) when using a different number of frames as input. The formula for the AMIS can be found in equation 6 in section 3.3. So the higher the AMIS, the better the phones are clustered compared to the TIMIT labels. An AMIS of 1 would be a perfect clustering, an AMIS of 0 would be expected with a random clustering and the AMIS can be negative if the clustering is worse than random. Frames are taken from the middle of every phone as a reference point. For example, the 4-1-2 formation means the input vector consists of the 10ms frame from the middle of the phone, plus the four frames preceding the middle frame and two frames following the middle frame. So, table 10 shows the clustering performance when using different frames in relation to the target phone as input to the clustering algorithm.

Frames	Vector Length	AMIS
1-1-1	39	0.4004
2-1-2	65	0.4118
3-1-3	91	0.4031
4-1-2	91	0.3916
2-1-4	91	0.4025
4-1-4	117	0.3934
5-1-5	143	0.3734
8-1-8	221	0.3377

Table 10: Adjusted mutual info score using different frames as input.

Table 11 shows the accuracy when using different windows of cluster indices as input for a gradient boosting classifier. In this example, the cluster algorithm is set to 41 different clusters and the input consists of five MFCC frames per phone in 2-1-2 formation. A visualization of this process is shown in figure 6, where the window is set to 3. The accuracy is determined by using the cluster indices as input for a gradient boosting classifier and training them using the phone labels provided by TIMIT. This is a supervised classification algorithm used instead of a GAN for evaluation purposes. It shows the accuracy when using different window sizes for vectors extracted from the cluster sequences. When using a longer vector, more context is taken into consideration in order to determine the phone label. For example, a window of five means that a sequence of five cluster indices is matched to each phone label in the training process.

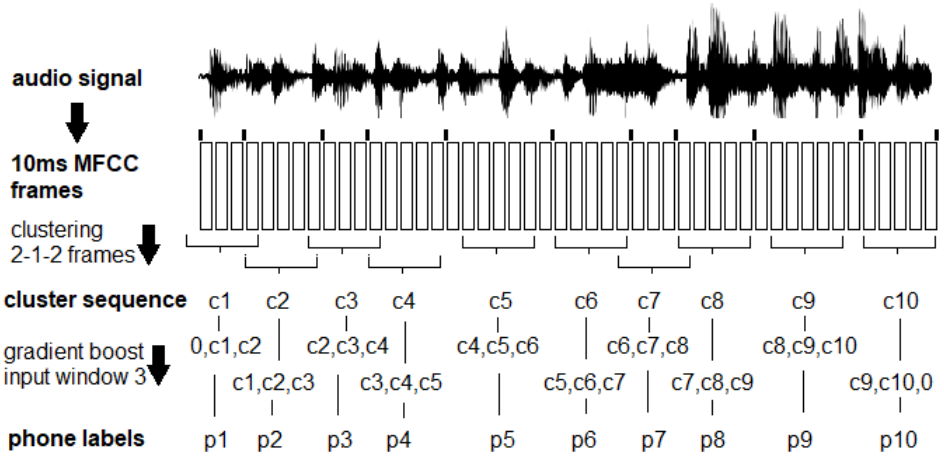

Figure 6: Visualization when using 2-1-2 frame formation as input for the clustering algorithm and an input window of 3 for the gradient boosting classifier.

Window	Accuracy
1	0.3973
3	0.5168
5	0.5336
7	0.5388
9	0.5422
11	0.5434
13	0.5466

Table 11: Accuracy when using different window lengths from the cluster sequence.

Table 12 shows the performance versus the number of clusters using five MFCC frames as input. Besides the AMIS it also shows an accuracy score determined by the gradient boosting classifier again. Training of the boosting classifier is done by matching TIMIT labels with a window of seven cluster indices.

Clusters	AMIS	Accuracy
30	0.4104	0.5315
41	0.4118	0.5388
50	0.4064	0.5352
64	0.4051	0.5324
96	0.4032	0.5374
128	0.3987	0.5330

Table 12: Adjusted mutual info score using a different number of clusters.

There is another method that can be used instead of a sliding window. We call this method the averaging technique. Instead of using multiple 10ms windows as input, here only one window is used with the length of the phone, according to TIMIT data. So one averaged MFCC frame is used over the entire phone length. For this experiment, the TIMIT data for phone start- and endpoints are still used, but in future research, the start- and endpoints from the automatic segmentation algorithm can be used. We first want to test how well the clustering algorithm works when using accurate input data, which also makes it easier to evaluate.

Table 13 shows performance versus different forms of data as input. On the first row, only an MFCC is used, this MFCC has 13 coefficients, which are averaged over the entire phone length. The second row, shows one MFCC also, but with 21 coefficients. In the third row, a value for the duration of the phone is added to the input vector. On the next row, a delta MFCC is added to the input vector. This means the first difference coefficients are also added to the input data. In the fifth row, a delta-delta MFCC is added, this means the acceleration coefficients are also added to the input data. Finally, in the last row duration is added again. The accuracy is again determined by using a boosting classifier and training on TIMIT labels matched with a window of seven cluster indices. In appendix B a cluster-phone cross-tabulation can be found using the data from the last row as input.

Input Data	Vector Length	AMIS	Accuracy
MFCC	13	0.3909	0.5177
MFCC	21	0.3975	0.5219
MFCC, Duration	14	0.4009	0.5425
MFCC, Delta	26	0.4485	0.5635
MFCC, Delta, Delta^2	39	0.4859	0.5845
MFCC, Delta, Delta^2, Duration	40	0.4861	0.5841

Table 13: Clustering performance using different input data.

The following tables (14 and 15) show how phones are split by k-means clustering when splitting each cluster in half at every level. Phones of which more than 50% are within the resulting cluster are shown. The input vector consists of an MFCC, delta MFCC,delta-delta MFCC and duration.

<table>
<tr><td colspan="4">v,f,dh,th,z,s,s,zh,sh,jh,ch,b,p,t,g,k,hh,hv,closure,silence</td></tr>
<tr><td colspan="2">f,th,z,s,zh,sh,jh,ch,p,d,t,g,k</td><td colspan="2">closure,silence</td></tr>
<tr><td>p,d,t,k</td><td>z,s,zh,sh,jh,ch</td><td>closure</td><td>silence</td></tr>
</table>

Table 14: Clustering tree right branch.

<table>
<tr><td colspan="4">iy,ih,ix,eh,ae,ax,ah,uw,ux,uh,ao,aa,ey,ay,oy,aw,ow,er,axr,l,el,r,w,y,m,em,n,en,nx,ng,eng,dx,q</td></tr>
<tr><td colspan="2">ih,ix,eh,ae,ax,ah,uh,ao,aa,ey,ay,oy,aw,ow,er,axr,r</td><td colspan="2">iy,uw,ux,l,el,w,y,m,em,n,en,nx,ng,eng</td></tr>
<tr><td>ao,aa,oy,aw,ow,er,axr,r</td><td>eh,ae,ey,ay</td><td>l,el,w,m,em,n,en,nx,ng,eng</td><td>iy,uw,ux,y</td></tr>
</table>

Table 15: Clustering tree left branch.

Table 16 shows how individual phones are clustered using 41 clusters. The input vector consists of a value for phone duration, an MFCC, delta MFCC and delta-delta MFCC. For each phone, the table shows the main cluster percentage and how dominant the phone is in the main cluster. The main cluster for each phone is the cluster to which the largest number of that phone is designated to. So for example for the zh/sh phone this means that 81% of this phone is in the same cluster, and within this main cluster, 43.1% is the zh/sh phone.

Phone	MCP	CDP	Phone	MCP	CDP
zh/sh	81.0	43.1	ae	30.8	24.9
ch	78.1	13.8	iy	28.8	42.5
f	71.6	31.7	l/el	28.2	43.6
w	69.7	44.4	d	27.9	34.5
y	55.6	20	hh/hv	27.2	10.9
th	55.1	7.9	v	25.7	10.9
jh	54.0	13.9	er/axr	25.1	33.7
t	52.6	39.3	ay	25.0	12.8
silence	50.8	81.8	aw	24.4	3.9
s	49.5	59.3	ao/aa	23.9	32.7
z	49.1	29.3	oy	23.8	3.7
b	48.6	43.7	uw/ux	22.5	10.4
p	47.2	26	ow	22.1	10.5
g	42.6	18.4	eh	21.0	16.6
ng/eng	41.5	11.2	uh	20.1	3.8
dx	40.8	28	ey	19.1	8.5
n/en/nx	35.9	64.4	dh	16.2	16.3
k	34.7	28.5	q	14.2	8
m/em	34.4	30.3	ih/ix	13.6	35.7
closure	32.9	92.2	ax/ah/ax-h	13.4	19.6
r	31.9	50.9			

Table 16: Main cluster percentage and cluster dominance percentage for each phone.

Next, we will look at how well the averaging technique works on clustering phone groups. The input vector again consists of a value for phone duration, an MFCC, delta MFCC and delta-delta MFCC. We will be looking at the six broad phone groups as previously defined in sector 3.1, table 4. Table 17 shows the clustering performance in terms of AMIS and accuracy after using a gradient boosting classifier. It shows the performance when using a variable number of clusters with the k-means clustering algorithm.

Clusters	AMIS	Accuracy
5	0.4986	0.8299
6	0.5281	0.8512
7	0.5130	0.8621
8	0.5331	0.8704
9	0.5160	0.8627
10	0.5043	0.8658

Table 17: AMIS and accuracy with different clusters on broad phone groups.

The next two tables show the results in more detail when using k-means clustering with eight clusters. Table 18 shows a cross-tabulation of the eight clusters and the six broad phone groups.

Cluster	1	2	3	4	5	6	7	8
Vowels	563	209	33694	39184	955	24966	13	6869
Closures	431	9169	45	39	23	34	20368	2602
Fricatives	24229	1199	87	135	2785	260	448	2667
Nasals	6	91	529	701	462	525	127	16820
Silences	110	15163	18	38	12	7	215	380
Stops	3754	1715	1116	1479	21340	1131	849	3663

Table 18: Cross tabulation of broad phone group versus cluster.

Table 19 shows the classification report after using the gradient boosting classifier. For every phone group, it shows the precision, recall and f1-score.

Group	precision	recall	f1-score	support
Vowels	0.93	0.95	0.94	28733
Closures	0.91	0.92	0.91	8490
Fricatives	0.87	0.77	0.82	8355
Nasals	0.60	0.79	0.68	5104
Silences	0.96	0.88	0.92	4287
Stops	0.81	0.72	0.76	9176
Accuracy			0.87	64145

Table 19: Classification report after using a gradient boosting classifier.

5 Discussion

5.1 Phone Segmentation

The segmentation algorithm was tested using different numbers of clusters as can be seen in table 5. The algorithm performs best with five clusters, whilst still maintaining a low over-segmentation rate. When using six clusters or more, the algorithm starts over-segmenting a lot, which is not desirable in most cases. In figure 7, which is made using the data points in table 5, the trade-off between the CDR and OSR is visualized. In order to reach a detection rate of 97.5%, over-segmentation exceeds 100%. The segmentation algorithm could however still be useful when using more than five clusters, as over-segmentation could be addressed by downstream procedures. With an over-segmentation rate of less than 1%, the segmentation algorithm detects up to 80% of the phone boundaries. As mentioned in the background sector,

other methods mostly reach a detection rate of up to 74% [18]. There are also some papers that report detection rates of up to 83%, however, these do not report over-segmentation rates [19]. Abdullah et al. first showed that k-means clustering with two clusters could be used as a tool for unsupervised phone segmentation [20]. The authors reached a similar accuracy of 80% on a small data set with single words. We now showed that our segmentation algorithm based on k-means clustering is also successful at segmenting full sentences on a large data set. Furthermore, we showed that using more than two clusters creates an interesting dynamic between the CDR and OSR. It can be said that this algorithm performs at least on par with other current models and might be a slight improvement when taking into consideration over-segmentation. This is a reasonable performance considering the algorithm works completely unsupervised. No data is needed except for the speech signal that needs to be segmented. The algorithm also works relatively simple, as no complex speech embeddings need to be generated.

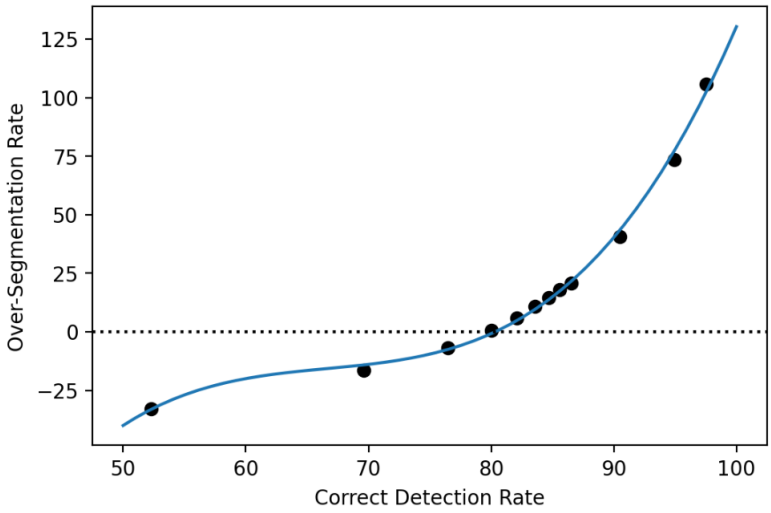

Figure 7: Correct detection rate versus over-segmentation rate plot.

When looking at transitions from different phone groups to each other individually, big differences in performance present themselves. Table 7 shows that the highest numbers of transitions are respectively in the groups closure → stop, stop → vowel, semivowel/glide → vowel, vowel → closure and vowel → fricative. Looking at table 9, the percentage of correctly detected transitions can be found for each group. Most of these big groups perform well, except for semivowel/glide → vowel, so in absolute numbers, there is much room for improvement here. The five groups with the most undetected transitions in absolute numbers can be seen in table 20.

Boundary	Misses	Percentage Correctly Detected
SV/Glide → Vowel	2803	53.9
Vowel → SV/Glide	2281	38.9
Vowel → Vowel	996	33.5
Stop → Vowel	898	85.6
Vowel → Nasal	611	84.9

Table 20: Highest number of undetected boundaries in absolute numbers.

The table shows that in absolute numbers of transitions between semivowels/glides and vowels show the most misses, in both directions. Focusing on improving these groups can therefore increase the overall CDR the most. The results also show that the other transitions that remain undetected a lot are always adjacent to vowels. So, the segmentation algorithm especially has difficulties in consistently detecting the onset points and end points of vowels. A paper by A. Kumar et al. suggests several improvements in detecting these points [34]. They suggest using several temporal and spectral characteristics of the speech signal as input data besides the MFCCs. Insights from this paper could be used in the future for improving the segmentation algorithm.

5.2 Phone Clustering

Next, we will discuss the performance of the different cluster techniques. Table 21 gives a summary of the most important results.

Data	Output	AMIS	Accuracy
Sliding Window MFCC	Individual Phones	0.4118	0.5388
Avg MFCC	Individual Phones	0.3909	0.5177
Avg MFCC, Δ, Δ^2, Dur	Individual Phones	0.4861	0.5841
Avg MFCC, Δ, Δ^2, Dur	Broad Phone Groups	0.5331	0.8704

Table 21: Clustering performance summary

5.2.1 Sliding Window Technique

First, we will discuss the performance of the clustering algorithm using the sliding window technique, which can be seen at the top of table 21. As can be seen from table 10 the algorithm performs the best in terms of AMIS using frames in 2-1-2 formation. This formation means that for each phone, a 10ms frame in the middle of that phone is taken, plus the two 10ms frames preceding and the two 10ms frames following the middle frame. The middle frame is chosen by subtracting the starting time from the end time of the phone and then choosing the frame which contains the resulting time point.

This means each phone is represented by a vector of 5 frames with 13 coefficients each. So, the input vector contains 65 parameters after stacking the frames.

The results indicate that the best performance is achieved when using frames representing a total of 50ms. A duration of 50ms is slightly shorter than the average length of a phone. When using more frames, the performance quickly drops, especially after using data of more than 100ms per phone. Using more than 65 parameters probably causes the model to be too complex, which leads to overfitting. The clustering algorithm might find clusters that do not generalize with new data. The overfitting could be countered by making sure there is not too much noise in the input vectors. When using more frames, the input vector might also contain data from the phones preceding and succeeding the target phone. This can be seen as added noise that causes the clustering algorithm to not see what the target phone is.

Another interesting point is that the formation 2-1-4 performs slightly better than the formation 4-1-2. This might indicate that information closer to the offset of a phone has more value than information close to the onset of a phone, which could be explained by the flexibility of the classical idea that speech production is primarily a result of pre-planning [35]. They suggest that the scope of advance planning narrows with an increased cognitive load. However, the difference in performance is not significant enough to draw serious conclusions. When looking at table 12, we can see the number of clusters has no real effect on accuracy. Performance is slightly better when using 41 clusters compared to other numbers, but the difference is not significant. It shows that the number of clusters used is not the most important factor, and the algorithm will work similar using different settings.

Table 11 shows that accuracy increases when using a wider window of cluster indices as input to the gradient boosting classifier. Especially the step from a window of one compared to a window of three phones results in a big increase in accuracy. This can be explained by the fact that phones can vary depending on the context, especially by the directly preceding and succeeding phone. Triphone-based acoustic models have shown to perform better in ASR and phone recognition tasks than monophone-based models [36]. This means that the cluster numbers representing the phone directly preceding and succeeding the target phone are very important in determining the label of the target phone. From table 11 can be seen that accuracy still slightly increases when using a window of more than 11. So besides directly adjacent phones, context further away can also benefit performance.

Consequently, we can conclude that context is very important when classifying phones. This is very important information if we want to use a

GAN for the classifying task, because a GAN uses context information to generate phone labels. This also pertains to one of the strengths of using a clustering algorithm. A clustering algorithm reduces the data representing a phone to one number. The dimensionality of the data is significantly reduced, resulting in a simpler index-to-label mapping problem. So, when using these cluster indices as input for a GAN, it makes it much less computationally expensive and complex to use more context. However, when using a gradient boosting classifier, a wider window does quite drastically increase the running time of the algorithm. Using a window wider than nine has very little effect on accuracy relative to the extra running time it takes.

5.2.2 Averaging Technique

Now we will discuss the performance of the clustering algorithm when using averaged MFCC frames as input. We tested input vectors containing only MFCC frames, delta MFCC frames and delta-delta MFCC frames. Also, the benefit of using the duration of the phone was tested, by adding a value for the duration to the input vector. An overview of these results can be found in table 13. Accuracy increased significantly from roughly 52% to 58% when adding a delta MFCC and a delta-delta MFCC to the input data. A delta MFCC improves performance by over 4% and a delta-delta MFCC by an additional 2%. From these results, we can conclude that a delta and delta-delta MFCC together are important in improving performance. This makes sense because a regular MFCC only gives static values extracted from each phone. Meaning a regular MFCC only contains static amplitude values in the frequency domain. When these MFCCs are averaged over a phone, all information about the temporal change is lost. With speech, it is more informative to also analyze the overall shape of the data. Knowing how the signal changes over time within a phone is essential. Therefore, adding information about the change in the audio signal is important for improving performance. A delta MFCC adds information about how much the frequency coefficients change over the duration of the phone. The delta-delta MFCC then also adds information about how fast the coefficients change over the duration of the phone. This information is shown to be very important in representing the dynamic within the data. Although adding a delta and delta-delta MFCC is not new, these results do reaffirm their importance. Another interesting detail from the results in table 13 is the effect of adding a value for the duration to the input vector. Adding an extra value for the duration to the input vector does increase performance when only using an MFCC. However, it has no significant added benefit when using the first and second derivatives of the MFCC. It is likely that the temporal information present in the delta and delta-delta MFCC cause the duration value to be redundant.

The results indicate that over 58% of phone labels can be deduced from cluster sequences when using a supervised algorithm with these cluster sequences as input. This is achieved by using the averaged MFCC technique, whereas the sliding window technique achieves a maximum accuracy of about 54%. The performance of 58% can be interpreted as a potential upper bound to the performance of a GAN, assuming an unsupervised algorithm does not perform better than a supervised algorithm. This also gives a rough answer to the research question of how accurately a completely unsupervised framework can potentially recognize and map phones from speech. The results signify that the clustering algorithm produces cluster sequences that still hold a reasonable amount of information about the phones. However, an upper bound of 58% accuracy means there is also room for improvement in the clustering algorithm. It indicates that there probably is a lot of information loss occurring when phones are represented as cluster indices. Liu et al. achieved a similar upper bound accuracy of 59% for their clustering algorithm [2]. They however did not implement a successful algorithm for automatically segmenting phones, they instead used the phone boundaries provided by TIMIT. Whereas we were able to develop and implement a new segmentation algorithm that performs well. Liu et al. also used more computationally complex Audio2Vec embeddings as input data. These embeddings are generated using an RNN encoder and an RNN decoder. So, our results show that using MFCCs as input data could also be an adequate method.

5.2.3 Phone Groups

Using the best-performing clustering method, we can now look at how phones are clustered by the algorithm relative to each other. Figure 8 can be made using the data from table 14 and 15. These tables show the phones with a more than 50% dominance in each cluster. The data is split into two clusters, and then the resulting clusters are each split into two clusters again, the resulting four clusters are each split into two clusters once more. Figure 8 shows the phone groups which are predominately present in each branch representing a cluster in the resulting tree. This gives a good visualization of how phones are grouped by the clustering algorithm.

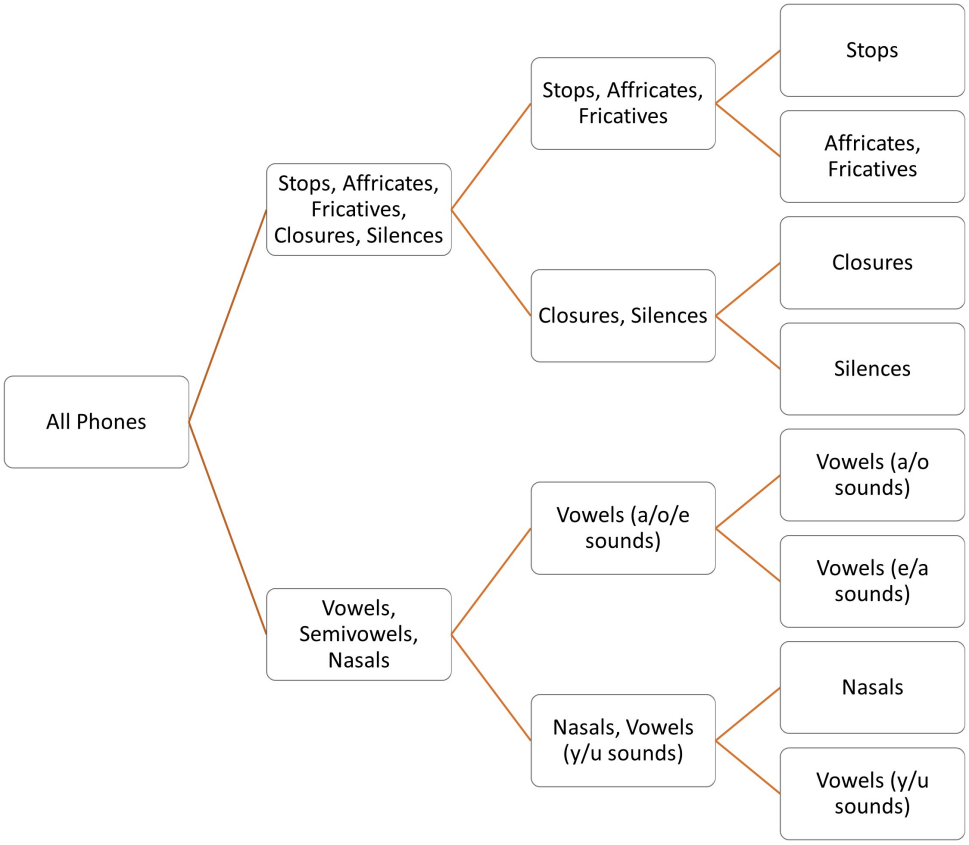

Figure 8: Clustering tree showing dominant phone groups in each cluster after dividing the data into two clusters on each level.

On the first level of the tree, vowels and nasals are clearly split from all other phone groups. Then on the right branch stops, affricates and fricatives are split from closures and silences. On the left branch, vowels are separated into two groups and nasals are concentrated in one branch. There are however also some phones that do not have a majority in any cluster anymore at this level. From table 14 and 15 can be deduced that v,dh,b,dx,hh,hv and q sounds are not clearly separated into one cluster. This means these sounds are the hardest for the clustering algorithm to separate from the rest. Another thing that stands out from figure 8 is that semivowels and glides are never clearly grouped into one cluster, so the algorithm cannot clearly separate this group from other groups. For example, one branch contains the l/el and w sounds, another branch the y sound, another branch the r sound and another branch the hh/hv sounds. Affricates and fricatives, however, are

always clustered together. So, these two groups are easy to separate from all other groups, but not from each other. Vowels are sub-divided into multiple groups, a/o sounds are separated from e/a sounds and y/u sounds. All in all, we can conclude that unsupervised clustering does result in a division of phones that has a clear phonetic reality. The clusters do correspond with the broad phonetic classes as described by Holmes and Holmes [37]. Broad phonetic classes like stops, fricatives, vowels and nasals are clustered together reasonably well when using a limited number of clusters. It is important to show that unsupervised clustering has a phonetic reality since it supports the idea of using it before the labelling process.

In table 17 we looked at how well the clustering algorithm can separate broad phone groups, and how many clusters should be used for the best performance. From this table, we can see that the algorithm works best with eight clusters in terms of AMIS. When using gradient boosting to classify the cluster labels into broad phone groups it achieves an accuracy of 87%. So, the clustering algorithm seems to be quite able to distinguish broad phone groups. Looking at table 18 we can find a cross-tabulation of the eight resulting clusters and their corresponding broad phone groups. We can see that there is a dominant phone group in every cluster. Cluster 1 contains mostly fricatives and cluster 2 mostly silences. Clusters 3,4 and 6 are dominated by vowels. Cluster 5 contains mostly stops, cluster 7 closures and cluster 8 nasals. These results correlate well with the clustering tree. In table 19 we can see a classification report after using the gradient boosting classifier. It shows how well the classifier was able to classify the individual phone groups. We can see that nasals and stops are the most difficult to cluster and classify. Vowels perform the best in terms of f1-score. So, it is interesting to note that although vowels are the most difficult group to segment, they are however the easiest to cluster and classify correctly.

It is also interesting to look at the performance of individual phones separately, to see which phones are largely clustered together and which ones are not. This gives us more information about where the clustering algorithm performs adequate or less adequate. Table 16 shows for each phone the percentage that is allocated to the majority cluster, the main cluster percentage (MCP). So, the MCP indicates how well the algorithm is at clustering the same individual phones together. The table also shows the cluster dominance percentage (CDP), which indicates how well the same individual phones can be distinguished from other phones. The results indicate that individual phones belonging to affricates, fricatives and stops are mostly clustered together with values between roughly 45% and 80%. Nasals are clustered together between roughly 34% and 42%. Notably, vowels are clustered least consistently in the same clusters. For almost all vowels less than 30% are allocated to the same cluster, whereas for some consonants more than 70%

are clustered together. This indicates that the clustering algorithm performs relatively poor at grouping the same individual vowels together. So again, future research should focus on vowels when attempting to improve accuracy on individual phones. However, as we discussed in the previous paragraph, the clustering algorithm does perform well at distinguishing vowels from other broad phone groups. Overall, we see a low CDP for most phones, indicating that the clustering algorithm performs poorly at distinguishing different individual phones from each other. CDP is only high for closures and silences. So, the results show that the clustering algorithm is relatively good at clustering broad phone groups, but less capable of clustering individual phones correctly.

5.3 Phone Mapping

As mentioned before in section 3.4, the next step is to map the clusters to actual phone labels, without using parallel data. Although an implementation of this step was not achieved during this research, this section will be used to lay out a theory on possible methods to continue from the last two steps.

5.3.1 Adversarial Learning

An interesting method to address this problem is by using adversarial learning. As mentioned before, methods using adversarial learning have been very successful at various translation tasks. Liu et al. showed that a GAN could be used to translate sequences of cluster indices to sequences of phone labels. By first segmenting audio files on the phone level and clustering these segments the problem is made into a discrete sequence-to-sequence translation problem. A GAN could then be used to generate a realistic phone sequence from an unlabeled cluster sequence. A visualization of this framework is shown in figure 9. On the discriminator side, the input consists of sequences of phone labels extracted from unrelated texts, so no parallel data. These texts can be extracted from any open source text data sets. These texts can then be converted into phone sequences. This can be done using tools like toPhonetics or Text2Phonetics [38] [39]. These applications translate English texts into phonetic transcriptions using the International Phonetic Alphabet. The generator receives the sequences of cluster indices as input. In the earlier quoted paper by Liu et al., the authors attempt to solve the problem using this method. The assumption here is that if the generated sequence is indistinguishable from a real phone sequence, it is automatically the correct sequence in most scenarios. So, the algorithm will predict phone labels based on context patterns derived from the cluster sequences. The context patterns will be learned and recognized from unrelated texts, which are transformed into phone sequences. The results from table 11 showed that context is important in predicting phone labels. These results give reason

to believe that an unsupervised GAN might perform relatively well at the task.

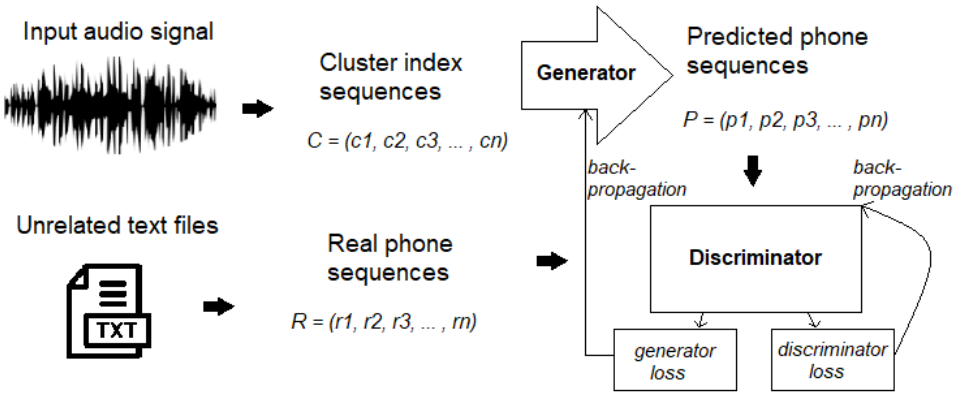

Figure 9: Visualization of phone recognition using a GAN.

The most important aspect of a GAN is that it can work completely unsupervised. The unsupervised nature of GANs is one of the main reasons why this architecture was considered for solving the phone classification problem. Another important reason to use a GAN for solving a sequence mapping problem is the robustness of the network. GAN-based networks have shown to be very robust against misalignment errors in input data with sequence-to-sequence translation [24]. This is a very important property in the context of mapping phones, because errors can be made during the initial phone segmentation process, causing misalignment errors with the input data. The automatic segmentation algorithm we introduced has a maximum boundary detection rate of 80%, so 20% of the boundaries are misaligned. So, it is important that the algorithm can deal with these input vectors containing errors in order to perform well.

Conventionally, GANs also have their limitations when it comes to discrete sequence-to-sequence translation. Past research concluded that it is still unclear how to employ adversarial training accurately on discrete probabilistic models [8]. Training on discrete data is difficult because the generator is guided by the discriminator to slightly make the output more realistic with each iteration. With discrete symbols, this slight change usually does not result in an actual change of symbols, resulting in the network not learning anything. Another problem is that the discriminator can normally only give a score on how real the input is based on the entire sequence. Since the score for a partial sequence is not necessarily indicative of a possible score for the completed sequence.

5.3.2 Sequential GAN

These problems regarding sequence-to-sequence translation have since been addressed by considering sequence generation as a sequential process, in a variation of a GAN called seqGAN [33]. SeqGAN works using a real-world data set of sequences comprised of discrete values from a certain vocabulary. In the context of phones, it could be trained by using a data set of phone sequences that is comprised of a vocabulary of phones. This data set is used to pre-train the generator G using maximum likelihood estimation. The generator then generates a sequence of phones $P_1 : T = (p_1, \ldots, p_t, \ldots p_T), p_t \in V$ using the phone vocabulary V with reinforcement learning. At every time step t there is a current state s of produced tokens in a sequence, and an action a which determines the next token to choose in the sequence. The generator uses stochastic policy model $G(p_t|P_1 : t-1)$, where the goal is to choose the next phone p_t that has the highest expected end reward. These possible rewards for a complete sequence are decided using the discriminator model D. These rewards are based on how similar a sequence is compared to a sequence from the real data set. A higher likelihood that a sequence would fool the discriminator, will result in a higher reward. A problem, however, is that the discriminator can only give a reward for a finished sentence. A high reward for a partial sequence does not necessarily result in a high reward for the completed sequence. This is similar to for example chess where short-term success sometimes needs to be ignored in order to achieve long-term success. This problem is solved by using Monte Carlo search with a roll-out policy to sample the remaining phones after the current state. So, using a roll-out policy, the sequence of phones will be completed N times to find the action with the highest expected end reward. The discriminator is used to evaluate each sequence by computing rewards. The evaluation can then be used as feedback to further improve the generative model. This consequently solves the difficulty of differentiating sequences of discrete symbols.

More recently another variation called stepGAN has further improved this architecture and works in a less computationally intensive way [40]. Consequently, a GAN-based architecture is a promising method to be used for unsupervised phone mapping. Combining the architecture of a GAN designed for phone mapping and newer variations like seqGAN and stepGAN could result in increased performance. So, future research should focus on implementing this GAN-based architecture while taking inspiration from new models like seqGAN and stepGAN. Using GANs to solve a sequence-to-sequence translation problem is still a relatively new concept that could benefit from more experimentation. Combining a GAN classifier with the automatic segmentation algorithm and clustering algorithm will then result in a complete model that only uses unsupervised learning. From more experimentation we can learn how adequate a completely unsupervised network

can perform. Therefore, we suggest future research to focus on GANs and especially the newest advancements like seqGAN and stepGAN to complete the phone mapping.

So, we encourage future research to use the knowledge we acquired about unsupervised segmentation and clustering. The segmentation and clustering implementations we developed could be combined with a GAN based on the research described in this section. This would then result in a network (like in figure 9) that can transform an audio file into the corresponding phone labels without ever using labelled data.

6 Conclusions

In this research, we introduced a novel algorithm for automatically detecting phone boundaries using k-means clustering. It is a relatively simple technique that performs at least on par with existing techniques when using the TIMIT data set. There is however room for improvement when detecting the boundaries of vowels. This could possibly be improved in future work by adding more temporal and spectral features to the input vector. We also analyzed the performance of k-means clustering for grouping phones using different data preparation techniques. We showed that using an averaged MFCC with delta MFCC and delta-delta MFCC as input generated the best results. Again, clustering performance on vowels was much lower compared to consonants. However, the cluster index sequences still contain a lot of information about the structure of a sentence. We also showed that the clustering algorithm is good at distinguishing general phonetic classes from each other. Mapping the sequences using a supervised algorithm achieved a 58% accuracy on individual phones and an 87% accuracy on broad phone groups. So, although using a clustering algorithm has the advantage of reducing the dimensionality of the data, the process also results in some information loss, especially at the level of individual phones. Using the generated cluster sequences could be viable as input for a GAN, however, information about individual phones might be limited. Unsupervised classification using a GAN would probably be more successful when solely looking at broad phone groups. We also showed that the newest variations of GANs display a lot of improvements in sequence generation. Methods like seqGAN could be used to translate cluster sequences to phone labels with improved performance.

To summarize, in this research we laid the foundations for a completely unsupervised framework for phone segmentation and recognition. We successfully implemented an unsupervised phone segmentation algorithm and used clustering to generate cluster index sequences that can be used by a

GAN. We analyzed segmentation and clustering performance on the level of individual phones and broad phone group to identify where the algorithms can still be improved. Our segmentation and clustering implementations can be combined with the proposed GAN framework to create a fully unsupervised network that segments, clusters and classifies phones automatically from speech. The resulting phone-level annotation of audio can then be of use for many types of ASR training.

References

[1] H. Aldarmaki, A. Ullah, S. Ram, and N. Zaki, "Unsupervised automatic speech recognition: A review," *Speech Communication*, 2022.

[2] D.-R. Liu, K.-Y. Chen, H.-y. Lee, and L.-s. Lee, "Completely unsupervised phoneme recognition by adversarially learning mapping relationships from audio embeddings," *arXiv preprint arXiv:1804.00316*, 2018.

[3] R. Gupta, T. Chaspari, J. Kim, N. Kumar, D. Bone, and S. Narayanan, "Pathological speech processing: State-of-the-art, current challenges, and future directions," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 6470–6474.

[4] Y. Koizumi, S. Saito, H. Uematsu, Y. Kawachi, and N. Harada, "Unsupervised detection of anomalous sound based on deep learning and the neyman–pearson lemma," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 1, pp. 212–224, 2018.

[5] C. Biemann, "Unsupervised and knowledge-free natural language processing in the structure discovery paradigm." in *Ausgezeichnete Informatikdissertationen*, 2007, pp. 31–40.

[6] E. Crawford and J. Pineau, "Spatially invariant unsupervised object detection with convolutional neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3412–3420.

[7] X. Garcia, A. Siddhant, O. Firat, and A. P. Parikh, "Harnessing multilinguality in unsupervised machine translation for rare languages," *arXiv preprint arXiv:2009.11201*, 2020.

[8] F. Huszár, "How (not) to train your generative model: Scheduled sampling, likelihood, adversary?" *arXiv preprint arXiv:1511.05101*, 2015.

[9] A. S. Park and J. R. Glass, "Unsupervised pattern discovery in speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 186–197, 2007.

[10] C. Spille, B. Kollmeier, and B. T. Meyer, "Comparing human and automatic speech recognition in simple and complex acoustic scenes," *Computer Speech & Language*, vol. 52, pp. 123–140, 2018.

[11] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," in *2018 IEEE*

International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018, pp. 4774–4778.

[12] J. Li, "Soft margin estimation for automatic speech recognition," Ph.D. dissertation, Georgia Institute of Technology, 2008.

[13] M. L. Seltzer and J. Droppo, "Multi-task learning in deep neural networks for improved phoneme recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 6965–6969.

[14] J. Michalek and J. Vaněk, "A survey of recent dnn architectures on the timit phone recognition task," in *International Conference on Text, Speech, and Dialogue*. Springer, 2018, pp. 436–444.

[15] J. Yuan, N. Ryant, M. Liberman, A. Stolcke, V. Mitra, and W. Wang, "Automatic phonetic segmentation using boundary models." in *Interspeech*, 2013, pp. 2306–2310.

[16] J.-P. Hosom, "Speaker-independent phoneme alignment using transition-dependent states," *Speech Communication*, vol. 51, no. 4, pp. 352–368, 2009.

[17] C. Lopes and F. Perdigao, "Phone recognition on the timit database," *Speech Technologies/Book*, vol. 1, pp. 285–302, 2011.

[18] O. Scharenborg, V. Wan, and M. Ernestus, "Unsupervised speech segmentation: An analysis of the hypothesized phone boundaries," *The Journal of the Acoustical Society of America*, vol. 127, no. 2, pp. 1084–1095, 2010.

[19] X. Yue and H. Li, "Phonetically motivated self-supervised speech representation learning," *Proc. Interspeech 2021*, pp. 746–750, 2021.

[20] A. Abdullah and E. Jasem, "Phonemes based speech word segmentation using k-means," *International Journal of Engineering Sciences Paradigms a nd Researches (IJESPR)*, vol. 29, pp. 23–27, 05 2016.

[21] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou, "Word translation without parallel data," *arXiv preprint arXiv:1710.04087*, 2017.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[23] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image

super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[24] T. Kaneko, H. Kameoka, K. Hiramatsu, and K. Kashino, "Sequence-to-sequence voice conversion with similarity metric learned using generative adversarial networks." in *INTERSPEECH*, vol. 2017, 2017, pp. 1283–1287.

[25] J. S. Garofolo, "Timit acoustic phonetic continuous speech corpus," *Linguistic Data Consortium, 1993*, 1993.

[26] K.-F. Lee and H.-W. Hon, "Speaker-independent phone recognition using hidden markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 11, pp. 1641–1648, 1989.

[27] K. Kaewtip and A. Alwan, "A hierarchical classification framework for phonemes and broad phonetic groups (bpgs): a discriminative template-based approach," in *2019 23rd International Computer Science and Engineering Conference (ICSEC)*. IEEE, pp. 18–23.

[28] P. B. de Mareüil, C. Corredor-Ardoy, and M. Adda-Decker, "Multilingual automatic phoneme clustering," in *Proc. 14th International Congress of Phonetic Sciences*, 1999, pp. 1209–1212.

[29] K. Stöber and W. Hess, "Additional use of phoneme duration hypotheses in automatic speech segmentation," in *Fifth International Conference on Spoken Language Processing*, 1998.

[30] O. Mella, D. Fohr, and A. Bonneau, "Inter-annotator agreement for a speech corpus pronounced by french and german language learners," in *Workshop on Speech and Language Technology in Education*, 2015.

[31] K. Kvale and A. K. Foldvik, "Manual segmentation and labelling of continuous speech," in *Phonetics and Phonology of Speaking Styles*, 1991.

[32] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *The Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.

[33] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[34] A. Kumar, S. Shahnawazuddin, and G. Pradhan, "Improvements in the detection of vowel onset and offset points in a speech sequence," *Circuits, systems, and signal processing*, vol. 36, no. 6, pp. 2315–2340, 2017.

[35] V. Wagner, J. D. Jescheniak, and H. Schriefers, "On the flexibility of grammatical advance planning during sentence production: Effects of cognitive load on multiple lexical access." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 36, no. 2, p. 423, 2010.

[36] R. Schwartz, Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul, "Context-dependent modeling for acoustic-phonetic recognition of continuous speech," in *ICASSP'85. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 10. IEEE, 1985, pp. 1205–1208.

[37] J. Holmes and W. Holmes, *Speech synthesis and recognition*. CRC press, 2002.

[38] "tophonetics," https://tophonetics.com/.

[39] "Text2phonetics," http://www.photransedit.com/online/text2phonetics.aspx.

[40] Y.-L. Tuan and H.-Y. Lee, "Improving conditional sequence generative adversarial networks by stepwise evaluation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 4, pp. 788–798, 2019.

[41] Y. Qiao, N. Shimomura, and N. Minematsu, "Unsupervised optimal phoneme segmentation: Objectives, algorithm and comparisons," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 3989–3992.

A First Appendix - Time Table

Time table during research.

1/8/21 - 5/9/21
Write proposal draft / literature research

6/9/21 - 15/9/21
Finalize proposal

16/9/21 - 22/9/21
Create code to import and read TIMIT data. Get to know data, write something about data.

23/9/21 - 6/10/21
More research in automatically segmenting sentences into individual phones using existing techniques. Trying methods from different papers with no success [41] [18]. Looked at simpler versions using the change in amplitude in MFCC frames.

7/10/21 - 27/10/21
Create code for clustering algorithm, implement k-means clustering and researching other methods. Test the clustering algorithm with different methods of data pre-processing. Averaged MFCC with delta and delta MFCC was developed and tested.

28/10/21 - 25/11/21
Prepared the input data for a GAN, which are cluster index sequences from the clustering algorithm. The real phone sequences were extracted from TIMIT first for testing, the plan was to later extract phone sequences from unrelated texts. Many attempts at implementing and understanding the GAN. Primarily using the github by Liu et. al. (`https://github.com/gary083/GAN_mapping_relationship`) [2]. Tried to run separate parts of the code and get it to work with different data. Also looked at the seqGAN and stepGAN papers [33] [40], and also looked at some other unsupervised sequence-to-sequence mapping solutions. Network turned out to be too complicated to completely understand and use without it taking a very long time, so we decided to drop the implementation of the GAN and focus on the clustering and segmentation, while still dedicating a theoretical part to the GAN network. Also did some writing on the GAN.

28/10/21 - 25/11/21
Instead of using a unsupervised GAN for the mapping problem, we attempted to solve it using some supervised algorithms. Experimented with methods

like a feed forward network, LSTM network and gradient boosting network. The gradient boosting classifier turned out to work the best for the task, so we decided to use this for evaluation of the clustering algorithm as well.

26/11/21 - 30/1/22
Focusing on phone segmentation again and came up with a segmentation method using clustering. Theorized that since the clustering algorithm showed to be able to distinguish general phone classes from each other that it could also be used to find the phone boundaries. Tried clustering small segments of speech and looked whether the cluster change close to phone boundaries. Segmentation based on clustering turned out to work, so we further developed it. Did some research on how to improve the algorithm and added a variable for the last found boundary based on other research [29] [18]. Looked at evaluation methods for boundary detection.

31/1/22 - 20/2/22
Looked in more detail at the clustering process. Now also implemented the sliding window approach, to get a better idea on how context influences performance. Created a clustering tree to look at how the clustering algorithm behaves with a varying k. Also looked at how the sliding window method compared to the averaged MFCC method. Also looked at a-symmetric sliding windows. Writed about the findings.

21/2/22 - 6/3/22
Looking at how the segmentation and clustering algorithm perform on individual phones. Looking at how the segmentation algorithm performs on transitions to and from different phone classes. Preparing the experiments. Writing about evaluation techniques.

7/3/22 - 3/4/22
Doing the experiments on TIMIT train + test set combined and documenting the results. Evaluating the results and writing them in the result section. Explaining the results in the result section.

4/4/22 - 6/22
Writing the rest of the thesis, creating visualisations and tables to aid writing. Revise writing multiple times and more literature research to support findings.

7/22 - 9/26 Rewriting and expanding the thesis. Adding results about broad phone groups.

B Second Appendix - Phone Cluster Diagonal

true	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
iy	2781	1255	0	56	2	0	0	35	2	239	0	68	0	1	1	2	2628	0	45	0	60
ih, ix	991	2487	0	169	14	0	1	244	52	1378	2	912	12	14	11	0	1659	0	564	1	766
eh	27	556	0	5	0	0	0	118	73	988	0	1112	1	0	1	0	178	0	869	0	126
ae	18	119	0	1	0	0	0	13	43	1573	0	1664	0	0	0	0	74	0	1491	0	12
ax,ah,ax-	17	59	1	41	74	0	0	104	640	388	8	557	30	104	95	4	15	2	634	9	1121
uw,ux	227	723	0	85	1	0	0	52	29	74	0	50	0	66	118	1	398	0	27	0	80
uh	0	62	0	1	0	0	0	7	22	33	0	26	0	22	4	0	9	0	45	0	152
ao, aa	0	8	0	0	0	0	0	51	1840	109	0	322	0	494	4	0	5	0	630	0	39
ey	156	589	0	8	0	0	0	18	4	493	0	281	0	1	0	0	564	0	228	0	42
ay	2	12	0	0	0	0	0	16	302	487	0	607	0	0	0	0	3	0	810	0	178
oy	0	7	0	0	0	0	0	6	174	28	0	64	0	20	2	0	8	0	84	0	19
aw	0	1	0	0	0	0	0	0	174	68	0	130	0	0	0	0	0	0	165	0	1
ow	0	9	0	1	0	0	0	21	332	69	0	143	0	118	21	0	6	0	266	0	119
er, axr	7	425	0	99	0	0	0	1752	82	178	0	245	0	28	13	0	39	0	80	0	91
l,el	14	24	0	13	17	0	0	20	816	88	0	39	11	2144	409	8	39	0	82	0	673
r	9	149	0	31	12	0	0	1896	242	118	0	210	11	119	227	3	21	0	141	0	118
w	1	4	0	13	14	1	1	1	40	3	0	2	2	3054	170	19	0	0	5	0	43
y	1307	156	0	54	2	0	22	1	0	4	0	8	0	10	6	22	403	0	0	0	1
m,em	20	8	3	449	8	1	17	17	16	9	0	1	5	58	1927	142	12	0	6	0	16
n,en,nx	66	54	6	2026	38	1	7	34	70	81	0	78	7	90	1374	182	58	0	22	0	127
ng,eng	33	8	0	477	0	0	0	2	8	3	0	0	0	38	302	17	24	0	1	0	9
v	8	4	4	261	33	3	24	13	9	3	4	1	8	41	695	181	2	14	1	2	22
f	0	0	2	7	200	13	13	0	0	1	62	0	46	0	0	3	0	8	0	253	0
dh	17	5	22	60	127	89	617	0	5	15	24	2	166	13	149	205	7	18	1	0	34
th	0	0	8	1	30	43	37	0	0	2	86	0	10	0	0	10	0	41	0	21	0
z	1	1	0	5	0	0	3	0	0	2	1777	0	1	0	1	0	0	0	0	285	3
s	0	0	0	0	0	0	0	0	0	0	4287	0	0	0	0	1	0	0	0	579	0
zh,sh	5	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	2640	0
jh	11	0	0	0	0	0	70	0	0	0	9	0	1	0	0	1	0	0	0	853	1
ch	0	0	0	0	0	0	3	0	0	0	15	0	0	0	0	0	0	0	0	844	0
b	2	0	0	0	2	8	626	0	0	0	0	0	197	55	14	32	0	1	0	0	6
p	0	0	3	0	9	16	454	0	0	1	1	0	1672	4	0	2	0	3	0	1	0
d	42	8	0	17	14	11	903	0	0	4	12	0	153	1	15	61	2	11	0	45	3
dx	140	168	0	469	0	0	1	30	52	65	0	60	2	4	178	88	48	14	13	0	69
t	8	1	2	0	82	38	627	0	0	0	102	0	324	1	3	14	0	14	0	340	0
g	10	4	1	28	6	4	356	1	0	1	2	0	1180	23	20	38	1	6	0	6	0
k	0	1	5	4	35	47	328	0	2	10	5	0	1974	20	0	4	1	12	0	108	0
hh,hv	367	14	0	102	195	5	130	1	30	88	3	18	289	21	54	13	194	0	27	22	1
closure	16	4	4070	1099	575	1415	0	13	0	4	57	0	1	40	328	5581	4	10767	1	109	10
q	233	41	19	279	67	106	197	34	226	111	0	81	318	268	202	117	169	83	86	0	71
silence	1	0	231	42	1548	8102	0	1	0	0	1	0	4	8	14	235	1	62	0	3	0

Figure 10: Phone-cluster diagonal cluster 0-20

true	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
iy	120	11	3	0	6	0	138	101	12	1	20	0	2042	2	4	27	0	0	0	1
ih, ix	2136	80	59	28	337	1	2199	1684	56	13	234	0	2121	19	7	56	4	0	0	36
eh	76	7	99	4	412	0	88	124	19	0	204	0	189	0	1	0	0	0	0	16
ae	5	1	23	0	238	0	2	5	3	0	52	0	62	2	2	0	0	0	0	1
ax,ah,ax-	536	36	721	6	1156	0	469	963	512	13	100	4	32	36	33	22	19	1	5	67
uw,ux	96	22	43	0	76	0	94	245	106	0	50	0	493	0	0	25	0	0	0	32
uh	28	1	52	0	38	0	69	57	83	0	19	0	20	0	0	0	0	0	0	6
ao, aa	1	1	1986	0	1293	0	2	15	1262	0	146	0	0	0	0	0	0	0	0	85
ey	27	1	5	0	29	0	46	8	7	0	20	0	557	2	0	0	0	0	0	2
ay	7	0	227	0	531	0	9	0	4	0	34	0	9	0	0	1	0	0	0	3
oy	1	0	225	0	139	0	1	0	155	0	8	0	5	0	0	0	0	0	0	1
aw	0	1	148	0	231	0	0	0	21	0	4	0	0	0	0	0	0	0	0	1
ow	15	2	573	0	538	0	0	15	644	0	16	0	4	0	0	0	0	0	0	1
er, axr	253	76	41	0	189	0	147	334	19	4	1410	0	187	16	0	3	0	0	0	1918
l,el	20	148	1396	3	161	5	9	101	2663	187	46	0	13	9	9	62	0	7	0	215
r	53	86	94	2	310	3	18	445	58	98	1610	0	81	2	2	4	0	0	0	2891
w	3	3	159	0	8	3	0	84	297	338	17	0	2	3	0	28	0	3	0	58
y	0	9	0	32	0	3	6	40	6	33	4	0	173	3	11	25	1	4	0	3
m,em	47	310	17	0	8	33	1	18	14	1408	31	8	16	47	3	893	0	3	0	28
n,en,nx	217	1711	51	0	56	38	17	130	46	622	46	119	83	100	17	4265	0	8	0	27
ng,eng	8	39	3	0	1	0	1	2	9	0	3	3	32	21	1	741	0	0	0	1
v	64	270	8	12	4	24	1	5	11	21	8	44	4	127	680	20	7	5	4	52
f	2	5	0	106	0	2	0	0	0	0	0	4	0	107	2240	1	44	2	7	0
dh	12	284	3	280	3	202	5	9	9	314	5	52	2	82	342	52	7	629	4	7
th	0	1	0	50	0	2	0	0	0	0	0	2	0	77	561	0	19	12	5	0
z	3	12	1	14	0	0	0	0	0	0	0	1	0	5	450	3	2478	0	0	0
s	0	0	0	23	0	0	0	0	0	0	0	0	0	4	215	0	5005	0	0	0
zh,sh	0	6	0	35	0	0	0	0	0	0	0	0	0	0	146	0	391	0	0	0
jh	0	2	0	369	0	7	0	0	0	0	0	0	0	0	90	0	158	3	6	0
ch	0	0	0	108	0	0	0	0	0	0	0	0	0	0	8	0	103	0	0	0
b	0	7	0	30	1	1492	2	0	3	141	0	0	2	7	9	1	0	383	40	6
p	0	0	0	629	1	96	4	0	0	11	0	0	0	33	48	0	0	437	118	2
d	4	28	0	450	0	1036	3	3	1	104	0	14	3	77	251	1	15	1335	163	3
dx	13	1489	3	0	63	3	0	12	12	1	27	223	92	3	133	70	0	0	1	103
t	2	3	0	3103	0	58	10	5	0	5	0	4	0	68	250	0	94	434	307	0
g	8	18	0	154	0	349	0	1	2	85	3	7	0	23	104	4	1	194	109	23
k	3	1	2	2252	0	4	3	2	1	14	0	2	1	135	187	2	9	210	1103	1
hh,hv	0	87	8	165	4	2	0	29	8	23	9	0	19	94	770	26	8	5	2	3
closure	71	357	0	2	0	17	1	4	4	49	5	6186	5	1238	304	228	83	1	49	13
q	20	197	126	37	60	35	4	124	59	167	66	11	86	688	121	52	0	194	1	78
silence	3	8	0	1	0	0	0	7	0	2	0	22	1	5552	75	13	1	0	5	0

Figure 11: Phone-cluster diagonal cluster 21-40