Antonio M. Macías Ojeda

Speaker Diarization

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 17.11.2014

Thesis supervisor:

Prof. Mikko Kurimo

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Antonio M. Macías Ojeda

Title: Speaker Diarization

Date: 17.11.2014 Language: English Number of pages: 16+33

Master's Programme in Machine Learning and Data Mining (MACADAMIA)

Professorship: Speech and Language Processing Code: S-89

Supervisor and advisor: Prof. Mikko Kurimo

In this thesis we document the development of a system to perform *Speaker Diarization*, that is, automatically trying to identify who spoke when in a conversation or any other piece of speech with several speakers. The intended usage is to be able to provide this functionality for broadcast news, with data provided by the Finnish broadcasting company *YLE* under the *Next Media* programme, financed by *TEKES*, the *Finnish Funding Agency for Technology and Innovation*.

Another goal is to produce a system compatible with existing Aalto University speech recognition software, in order to open the door to future improvements and research.

The produced system, a newly implementation of established methods, with the parameters we determined were the best for our use case, obtains a performance that is very close to current stat-of-the-art systems, while still being compatible with the existing speech recognition software of the Aalto University and having a reasonable speed performance. Further improvements to the system are being made as we speech, opening the door to more research options.

Keywords: Speech, Speaker, Recognition, Diarization

A mi familia, por creer en mi cuando yo no podía,
y a mis amigos, mi casa lejos de casa.

To my family, for believing in me when I didn't,
and to my friends, my home away from home.

Antonio M. Macías Ojeda

Otaniemi, 17.11.2014

Preface

It is often said that time flies, and this has indeed been the case with this thesis. I first came to Finland in 2010, and the work this thesis is based on was done in the Speech Group at the Department of Information and Computer Science at Aalto University School of Science during the year 2012. However, work, life, and other issues delayed me and only now, nearly at the end of 2014, the actual thesis book is seen the light, something that at times I thought would never happen.

For these reasons, apart for thanking my family and friends, either here in Finland, in the US where I spent 2013, or back in Spain, without whom I wouldn't have finished, I have to thank my Supervisor Mikko Kurimo, for his immense patience and understanding, even if I was always late to all the deadlines, and the Study Coordinator of my programme, Päivi Koivunen, for her tremendous help with the paperwork and reminding me what's left to do. I also have to thank Ulpu Remes, Ville Turunen, Janne Pylkkönen, Peter Smith, Reima Karhila, Kalle Palomäki, and Jorma Laaksonen, together with the rest of the Speech Group, for their help and guidance in tackling this problem. Also, Hotloo Xiranood was a great company for brainstorming and drinking way too much coffee during that year, and Subhradeep Kayal helped me by being in my same situation and continuing where I left off.

Whatever good is in this thesis I owe it to all of you, and all the mistakes are just a reflection of my own shortcomings.

Thank you all, we finally did it!!

Antonio M. Macías Ojeda

Otaniemi, 17.11.2014

Contents

List of Figures

List of Tables

List of Terms

ASR Automatic Speech Recognition. 3, 8

BIC Bayesian Information Criterion. 11, 14, 15, 21, 22, 24

DER Diarization Error Rate. 17, 18, 21, 24, 25, 27

FFT Fast Fourier Transform. 4

GLR Generalized Likelihood Ratio. 11, 12, 14, 21

GMM Gaussian Mixture Models. 5

HMM Hidden Markov Model. 5

KL2 Symmetric Kullback-Leibler Divergence. 11, 13, 14, 21

LER Letter Error Rate. 7, 8

LVCSR Large-Vocabulary Continuous Speech Recognition. 3, 8

MFCC Mel-frequency cepstral coefficients. 4, 5

MLLT Maximum Likelihood Linear Transform. 5

MTER Missing Turns Error Rate. 18, 21–23

NIST National Institute of Standards and Technology. 17, 21

STT Speech To Text. 3

VAD Voice Activity Detection. 10, 17, 19, 21, 27

VER VAD Error Rate. 17, 18, 20, 21, 27

WER Word Error Rate. 7, 8

1 Introduction

Speech recognition, where a computer system recognizes human speech and acts based on it, by either following directions or just transcribing it to text form, is nowadays a reality. It changes the way we interact with "smart" electronic devices, and the range of applications is vast: from a museum with a stand that can answer questions about the displayed items, to handicapped people navigating internet without the traditional keyboard and mouse. Even smart phones are able to leverage it, enabling us to, for example, ask for directions to our favorite *café* without typing in their tiny screen.

While the general public is getting used to speech recognition and its applications, most people don't know about speaker diarization. In its base form, speaker diarization tries to identify who spoke when in a conversation or any other speech with more than a single speaker. This is useful as a reinforcement of speech recognition for a variety of applications. For example, if we assume perfect speech recognition, we could have perfect transcriptions of all the news of the last year, and it would be possible to do a variety of searches or other information retrieval tasks in this body of text. However, none of them would be able to consistently retrieve everything that a particular president said in an interview, something that would be trivial if his speaking turns where automatically labelled.

Speaker diarization has other usages as a speech recognition support, like meeting or trial transcriptions, or per-speaker speech recognition adaptation to improve the speech recognition quality to name a few. In this thesis, we will be using data gently provided by the Finnish broadcasting company *YLE* under the *Next Media* programme during its third phase, in 2012. The *Next Media* programme is financed by *TEKES*, the *Finnish Funding Agency for Technology and Innovation*, and our main goal is to produce a speaker diarization system capable of being used to help in producing rich transcriptions of broadcast news videos. As such, we will be assuming that there is no speaker overlapping. Our produced system needs to have acceptable performance both in accuracy and speed, because in its intended usage it needs to be able to perform speaker diarization in large amounts of broadcast news videos, in an acceptable amount of time. It does not, however, need to be able to operate online, that is, it does not need to be able to perform the speaker diarization on the fly as the videos are recorded, it will be used afterwards.

Another goal is to produce system compatible with the AaltoASR [5] package, for future developments and optional per-speaker adaptation.

The system developed uses previously existing *Voice Activity Detection* and *Automatic Speech Recognition* systems developed at the Aalto University, so the scope of our contribution is to provide *Speaker Segmentation* and *Speaker Clustering* subsystems, and determine the best parameters for the best performance in our use case. Most of the bibliography in the field focuses on English language data, so while the methods used in this thesis are already established and not innovative, the software implementation and the parameter exploration used to fine-tune these methods and get the best results for our use case are totally new.

2 Speech Recognition

Speech recognition or, more precisely, Automatic Speech Recognition (ASR), refers to automatic systems capable of transcribing human speech to written text, without human intervention. Commonly, this textual output is used for human reading, but since computer systems can handle text much easier than speech, Speech To Text (STT) can also be used as a first step for voice user interfaces or a variety of information retrieval tasks such as searching for particular pieces of information in speech or video data. These tasks are commonly called speech retrieval.

Other commonly related tasks are speaker identification, where the system tries to identify the identity of the speaker, or speaker verification where the system tries to verify that the speaker is who he claims to be.

2.1 Basics

Depending on the particular application, ASR systems approach the problem of transcribing speech to text in different manners. In this overview we will focus on Large-Vocabulary Continuous Speech Recognition (LVCSR) systems, that is, systems that are aimed at being able to transcribe speech uttered in a natural, continuous fashion (as opposed to isolated words) and that can handle a very big vocabulary, generally enough to tackle on general communication and not restricted to particular cases. We will discuss a generic LVCSR system, giving more details on how our particular system differs to that generic one. More details on the particular system we use in section 2.3, and a general overview of recent LVCSR advances can be found in [22].

A general representation of a probabilistic LVCSR system can be seen in figure 1, where the system components are encased in boxes, and its different contributions are represented.

Basically, from the digitized input speech we extract the observed features O, and we need to calculate what are the most probable sequence of words $\hat{W}$, given that observation.

For any given sequence of words W, the probabilities $P(W)$ are given by the language model. The probabilities of observing a set of features given a sequence of words, $P(O|W)$, are in turn given by the acoustic model.

Thus, the process of calculating $\hat{W}$ involves solving:

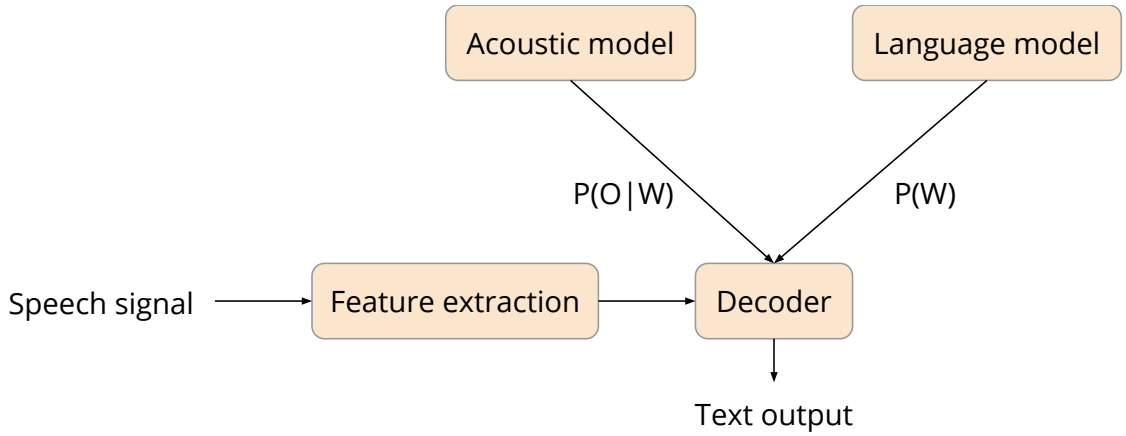

Figure 1: Schematic view of a speech recognition system.

$$\hat{W} = \arg\max_{W} P(W|O) = \arg\max_{W} P(O|W)P(W) \tag{1}$$

Which is calculated applying Bayes rule and based on the observation that the normalization term, $P(O)$, is the same for all W so we can ignore it in the denominator.

2.1.1 Feature extraction

Feature extraction is where the observed features are obtained from the digitized input signal. Since the features are intended for speech recognition, typically we want a representation of the input signal that retains enough information to recognize any speech in it, but discards the rest.

The most common way of representing features with these goals in mind, and the one employed by the recognizer used in this thesis is Mel-frequency cepstral coefficients (MFCC) [4], which will be discussed below.

The statistical properties of the speech signal changes over time but do so relatively slowly, so it is possible to take a small window and assume that they are stationary within it, and it is from these windows that the features are commonly extracted. In the used recognizer, for speech recognition tasks, the windows (called *frames*) are shifted every 8 millisecond (*frame shift*) and have a width of 25 milliseconds (*frame width*), so they are overlapping. If we just cut the signal at the frame borders there might be discontinuities which can be a problem for Fourier analysis, so a common approach is to shrink the values of the signal towards zero at the frame borders. This approach is called *Hamming windowing*.

After fractioning into frames, the short time power spectrum on them is extracted using the Fast Fourier Transform (FFT). This spectrum is sampled in a linear scale for each frequency band, but studies have shown that humans are more sensitive at low frequencies. This produces that two sounds can be, for example, perceived as one being half the pitch of the other with high precision on low frequencies, but

at frequencies over around 1000 Hz the human perception of pitch differences loose accuracy. A unit of pitch called *mel* is defined so that sounds that are perceived as equidistant in pitch are separated by the same number of *mels*. It turns out that at low frequencies, due to the human hearing characteristics mentioned above, the mapping between hertz and *mels* is linear. However, over 1000 Hz, it becomes logarithmic. Thus, the power spectrum is filtered using triangular filters following a non-linear *mel* scale instead of a linear one.

The mel spectrum could be used as a feature representation, but spectral coefficients at different frequency bands are correlated. However, by applying a discrete cosine transform to the log of the mel spectrum we get the aforementioned Mel-frequency cepstral coefficients (MFCC). The different coefficients of this cepstrum representation tend to be uncorrelated, and can then be modelled by Gaussian Mixture Models (GMM). From all the coefficients, only the first 12 are used as features in each frame by the employed system, since those represent the vocal tract shape and thus carry the information needed to recognize the uttered phone in that point in time.

In addition to the first 12 MFCCs, the average power of the frame is added to each frame, and the *delta* and *delta-delta* (first and second time derivatives corresponding to velocity and acceleration respectively) of these 13 features are added as well since they provide useful information for phone recognition, resulting in a 39-dimensional vector.

As useful additions, the system being used globally normalizes mean and variance through a linear transform, and finally applies a Maximum Likelihood Linear Transform (MLLT) to reduce the effect of speaker and environment variations [7].

2.1.2 Acoustic modeling

The acoustic model gives the probabilities of observing a set of features given a sequence of words, *P(O|W)*. To make this possible, it has to be able to map the extracted features to the language valid phonemes. Phonemes do not match a particular feature but rather a sequence of them, with time dependencies, and they have dependencies with the surrounding phonemes as well. The same phoneme can be produced in different ways, with changes of pronunciation, accent, or surrounding phonemes. The most commonly used representation able to cope with these properties is the Hidden Markov Model (HMM).

In Hidden Markov Models, we have an automata composed of states and transition probabilities between different states. The system being modeled is considered a *Markov process*, in which the present state depends only on the previous one, so we do not depend on the full story of states to make future predictions. The particularity of the Hidden Markov Model is that it is not possible to know in which state the system is at any given moment, since the states are hidden, not observable. However, even if the state is not directly visible, output depending on the state is observable and can be used to predict the state.

In speech recognition states represent the phonemes we want to recognize, while the extracted features are the observed features. We wish to train a different Hidden

Markov Model for each *triphone*, that is, each possible phoneme and it's surrounding previous and next phonemes. The states are associated with probability density functions, usually mixtures of Gaussians.

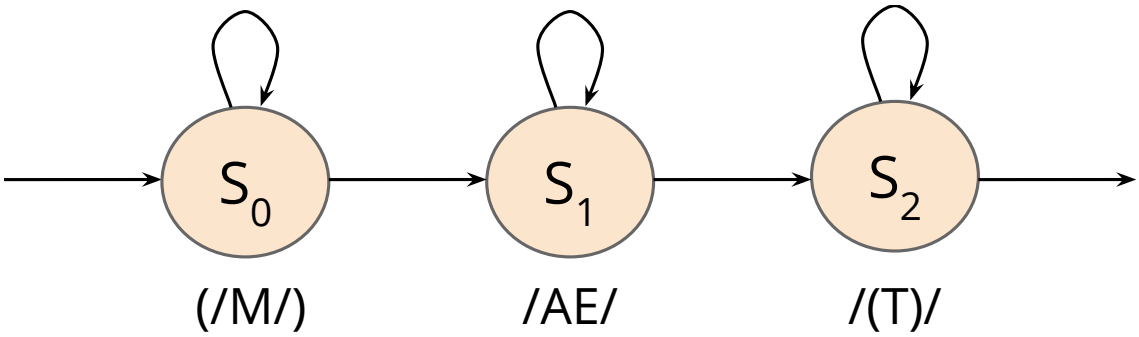

Figure 2: Simplified HMM representing the phoneme /AE/ after /M/ and before /T/, as appearing in the word "mat".

Note that just knowing the uttered phonemes does not equal to knowing how to transcribe them. In some languages, like Finnish or Spanish, most of the phonemes correspond to an unique letter or letter combination, with a few well defined exceptions. This is not true of other languages like English, where different phoneme combinations can be transcribed in different ways. To cope with that, acoustic models are supported by a *lexicon*, which hold lists of words with the phoneme sequences that can be transcribed into those words, and associated probabilities. Some of these words will have only one pronunciation while others will have several, which can depend not only on the language itself but also on local variations and accents. One particularly popular choice for building North American English *lexicons* is to base them in the Carnegie Mellon University Pronunciation Dictionary, as mentioned in [2].

A thorough discussion of Hidden Markov models for speech recognition can be found in [20].

2.1.3 Language modeling

The language model gives the probabilities of any given sequence of words, *P(W)*. It holds information about the words and their possible relations, and this information can be used to determine which word is most probable of the available options, or to discard particularly improbable words.

Since each word's probability depends on the previous words, most systems use an *n-gram* model, that is, the model assumes that the most recent word depends on the $n-1$ preceding words. These probabilities can be obtained by training on a text corpus, computing the relative word frequencies of n-size word sequences. Since getting all possible valid words and sequences of words in the target language is impossible, and even getting all of those appearing in a large enough corpus would make the language model unmanageable, usually only the most frequent of them are added to the language model, while using smoothing the model assigns some probability mass to the unseen or underestimated sequences.

2.1.4 Decoding

The decoder is the final step in process of calculating $\hat{W}$ and solves:

$$\hat{W} = \arg\max_{W} P(W|O) = \arg\max_{W} P(O|W)P(W) \tag{2}$$

As was shown in figure 1. It manages to do that using Bayes rule on the outputs of the previously discussed modules. Since the search space for all possible word sequences is huge, only word sequences that are possible in the target language are explored.

The search space is represented as a graph with each node representing time and each arc representing a word between start and end nodes. This graph is called a *lattice*, and each path through it represents a hypothesis, with the best hypothesis being the path with the higher probability. This enables us to use multiple-pass strategies, where the first pass uses simple acoustic and language models, and more complex models are used in following passes, with smaller lattices.

To further limit the search space and reach acceptable speed performance, other improvements are made, such as *beam pruning*, where tokens that have a probability lower than the highest probability minus a predefined threshold are pruned away, and *histogram pruning*, in which only a predefined number of tokens are kept at every step.

A thorough discussion can be found in most speech recognition textbooks, for example [12, Ch. 9] has been followed in this chapter, and a good introduction can be found in [21, Ch. 2].

2.1.5 Per-Speaker Adaptation

Acoustic models are trained either in clean speech or in variable conditions to make them more reliable. In either case, they can hardly match the exact speaker and ambient conditions of the new speech data in which they are used.

If we have labeled turns in which we know the start and end times of different speaker-ambient conditions on the new speech data, we can use this information to do model-space transformations to better match these conditions. This is a great use case for speaker diarization, in which we want to label not only the who spoke when turns, but also the different ambient conditions. For example, we would like to have a label for speaker "A" with some background music, and another label for the same speaker without this background music. This way, we can transform the acoustic models to better match these new conditions, in order to maximize $P(O|W)$.

2.2 Evaluation Metrics

In order to advance in scientific and technical endeavours it is important to have ways to measure the performance and correctness of the proposed solutions as precisely as possible, in order to know when new developments are an improvement over previous approaches. Speech recognition is not an exception, and the most widely used evaluation metric is the Word Error Rate (WER).

WER is based on the Levenshtein distance [16], also known as edit distance, which measures the minimum amount of editions that has to be performed on one text to transform it into another. These editions are substitutions, insertions and deletions of one unit. For example, if we take letters as the units, "cats" and "cat" has a Levenshtein distance of 1, as a deletion of the "s" in cats (or an insertion of it at the end of "cat") transforms one word into another.

In speech recognition, the most used unit is the word, and this metric is used to compare the output of the system against a reference text, typically a human-made transcription that is considered accurate. Word differences are then considered errors, which can then be classified according to the type of edit that would correct it. Substitution, insertion and deletion errors are thus tracked and sometimes can be reported separately. To calculate the WER then:

$$\text{WER} = \frac{\text{Word errors}}{\text{Total words}} * 100\% \tag{3}$$

While WER is by far the most common evaluation metric, some others can be found in the literature. For example, [18] compares some information retrieval measures against WER for speech recognition performance evaluation.

2.2.1 Use of LER in agglutinative languages

When reporting the speech recognition performance, the metric Letter Error Rate (LER) is preferred to WER in agglutinative languages, such as Finnish. As the previously discussed WER, the LER is based on the *Levenshtein* distance of the output compared to a reference text, but in this case the dissimilarities are counted using letters as a unit instead of words:

$$\text{LER} = \frac{\text{Letter errors}}{\text{Total letters}} * 100\% \tag{4}$$

Finnish is an agglutinative language, where morphemes and suffixes are concatenated together. This can lead to large compound words whose translation would require several words in a non-agglutinative language like English. For example, "kahvin+juoja+lle+kin" in English would translate to "also for a coffee drinker" [21, p. 44]. From this context, if there is an error in some part of the compound word while still correctly transcribing the other parts, reporting the full word as being an error would be too harsh, since the full word carries much more meaning than in a non-agglutinative language and the system could be doing a good job at transcribing most of the spoken message accurately. Thus, for reporting the performance of these systems on agglutinative languages, LER is considered more representative than WER.

From the speech recognition point of view, a deeper discussion of some of the common properties of the agglutinative languages in general and of the Finnish language in particular can be found in [27, p. 45]. Further discussion of WER and LER and some alternatives, mostly centered around the task of speech retrieval, can be found in [27, p. 62].

2.3 Aalto ASR system

In this thesis, we have used a recognition system being developed at Aalto University, with Finnish audio as test data. We have been giving a few details of the Aalto ASR system and how it differs from a generic LVCSR system in the previous sections, wherever applicable, but that was just the tip of the iceberg because we were mentioning them in the context of a general LVCSR introduction.

One thing that takes the Aalto ASR system apart is that it is designed to cope with agglutinative languages in general, and Finnish language in particular. As introduced in the previous section, these languages can have large compound words that carry much more information than the typical word in, for example, a Latin-derived language. This language structure hinders the training of language models in the typical way, since the number of different word forms is levels of magnitude larger than in non-agglutinative languages. To cope with these difficulties, the Aalto ASR uses n-gram models built on word fragments, as opposed to models built on full words.

Using fragments instead of full words affects the decoding process, and at the same time finding a selection of word fragments that is representative enough to model the language but compact enough to enable effective usage of it is not a trivial task, so the system has to use several other improvements to cope with this task. These improvements can also help systems intended to cope with compounding languages, such as German or Swedish, which present similar problems with their big vocabulary.

This is just a bird's eye overview of the Aalto ASR system. For more details, refer to [10], [11], and there is a good description of it in [27, Ch. 4] too.

The mentioned Aalto ASR system as well as the speaker diarization system implemented in this thesis are also available online and can be obtained in [5].

3 Speaker Diarization

Speaker diarization consists of segmenting audio into different speaker turns and identifying which speaker is talking in which turn. This task has several applications, for example it can enable for richer transcriptions than what speech recognition can do it on its own, by labelling the speaker turns and thus enabling information retrieval and searching by speaker identity. It can also be a useful step for doing per-speaker adaptation, with the goal of improving the quality of speech recognition transcriptions [21].

3.1 Basics

While the task of speaker diarization in the general case is well defined, there are many subtleties in solving particular instances of the problem that can change how it can be approached. For example, since we are focusing on broadcast news, we assume that there are no overlapping speakers, while that wouldn't be true if our system were supposed to perform speaker diarization of meeting room or trial recordings, or other environments where some speaker overlapping is bound to happen. Other variables that could affect the performance of the system, and that could or could be not known in advance, are speaker spontaneity, expected turn duration, maximum or minimum number of speakers, quality of the recording, etc. Our system tries to be robust to those variables and assumes no prior knowledge of them, and as such we try to find a setup that can deal with them, although some fine-tuning is possible. These variables and what can we do about them in the constraints of our system are discussed in section 4.

Our system can be divided into four tasks as can be seen in figure 3. First, feature extraction is performed over an audio stream. Second, voice activity detection determines which segments of audio contain speech and which are non-speech only. Afterwards, speaker segmentation ensures that each segment of speech is produced by a single speaker. The last step is to do speaker clustering, labelling each speaker turn of the same speaker with the same label to identify who spoke when.

Good overviews of current state-of-the art in speaker diarization can be found in [25] and [1]. In our case, as mentioned in the introduction, we will focus on broadcast news data, provided by *YLE* for its use in the *Next Media* programme. We consider that there are no overlapping speakers.

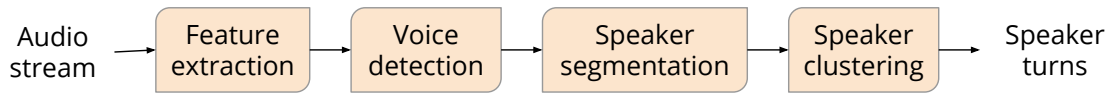

Figure 3: System workflow

3.2 Voice Activity Detection

Voice Activity Detection (VAD) aims to detect which parts of the audio stream contains voice and which parts are non-speech. Taking out the non-speech parts both speeds up and increases the accuracy of the later tasks, but a bad detection could take out audio segments with speech on them, which would lead to errors since those segments would not be considered afterwards. Another important appreciation is that in the middle of normal speech, between words and phrases, there are short silences. If we have a very sensitive voice activity detector and we tag these short pauses as non-speech, the detected speech segments would have too short a duration, sometimes only one word, to be used reliably in next steps. It is thus important to give priority to false positives over false negatives, labelling a segment as speech when in doubt, and also to ignore the short non-speech segments, specially between detected speech.

In broadcast news, non-speech can come from several sources, from music to background noise of all kinds when doing outside interviews. For these reasons a voice activity detection system based on energy levels with a threshold would not work properly, since these levels are very variable. In our system, we use a two class model-based system, with an speech model and a model trained on crowd noise as non-speech model, developed at the Department of Information and Computer Science at Aalto University School of Science. The raw output is the most probable source for each frame, speech or noise.

Since the system uses 125 frames per second, real speech of even a single word will last longer than a frame. To prevent false alarms, we assume that a voice segment will have to last more than a minimum time to be labelled as such. For the same reason, to prevent cutting the segments for wrongly detected non-voice frames or just between words, once we are in a voiced segment there must be a certain amount of consecutive non-voice time to mark the end of the segment. We can also enlarge the detected segments before and after by a small amount of time, to make sure that we don't miss any speech with too tight bounds. The exact values of these times are application dependent, and how these were decided is discussed in section 4.2.1.

Better results could be attained with a non-speech model specifically trained and tailored for the kind of non-speech expected at the broadcast news data available, but since this training is time and resource consuming and we are being conservative about the parameters used for labelling speech and non-speech, the crowd noise model gives an adequate performance, as seen in section 4.2.1.

3.3 Speaker Segmentation

After the segments of speech in the audio stream have been detected and labelled, it is still important to know if they are composed of only one speaker or more. In the voice activity detection phase we ignored small non-speech segments like those between words, which cannot be reliably detected. Furthermore, in broadcast news content we do not consider the case of overlapping speakers. However, it is still possible that some continuous speech segment has more than one speaker, talking in turns. Since we want to label each segment with an unique speaker id, we have to break these multi speaker segments into unique speaker segments before clustering.

For this task, known as speaker segmentation, several methods have been proposed in the literature, and a recent overview can be found in [1]. The developed system can use either a one or multiple passes approach, and either a fixed size sliding window or a growing window to select hypothesized speaker changing points. It can also use a few different metrics to determine if those points are indeed a speaker changing point or if the audio before and after the hipothesized changing point belongs to the same speaker, such as Bayesian Information Criterion (BIC), Generalized Likelihood Ratio (GLR) or Symmetric Kullback-Leibler Divergence (KL2). Further discussion of these options follows.

3.3.1 Bayesian Information Criterion

Bayesian Information Criterion (BIC) [23] is a criterion used to select which of a set of models best fits a determined set of data samples. In parametric models, adding parameters improves the likelihood, but it is possible to incur in overfitting if the model is overly complex. For this reason, BIC introduces a penalty term that gives a bigger penalty to more complex models, aiming to reach a balance between fitness to the data and complexity of the model.

In the case of speaker segmentation, given a speech segment and a hypothesised speaker changing point, there are two possible hypothesis: that all speech comes from only one speaker, or that the speech before the changing point comes from a different speaker than the speech after the changing point. To be more precise, we can consider a segment of consecutive data samples $\{x_0, \ldots, x_n\}$ and we have to choose between two models, one where all x samples are drawn from a multivariate Gaussian process $x \sim N(\mu, \sum)$, where μ is the mean vector and $\sum$ is the full covariance matrix, and another model where the samples before and after the hypothesised changing point, i, are drawn from two separate multivariate Gaussian processes, $\{x_0, \ldots, x_i\} \sim N_1(\mu_1, \sum_1)$ and $\{x_{i+1}, \ldots, x_n\} \sim N_2(\mu_2, \sum_2)$.

The BIC metric to choose from these models is:

$$\Delta BIC = -\frac{N}{2}\log|\sum| + \frac{N_1}{2}\log|\sum_1| + \frac{N_2}{2}\log|\sum_2| + \lambda P \tag{5}$$

Where λ is and adjustable penalty and P is the penalty term weight:

$$P = \frac{1}{2}(d + \frac{1}{2}d(d+1))\log N \tag{6}$$

Where d is the dimension of the space.

In [3], the penalty weight was set as $\lambda = 1$, but other authors have used other weights, for example [26], recommends using $\lambda = 1.3$. All in all, choosing a sensible penalty weight is data dependant, while the decision to prefer the two Gaussians over the single Gaussian model is when $\Delta BIC < 0$. Further discussion on the λ selection and it's effect for speaker segmentation can be found at section 4.2.2.2, and in figure 6.

3.3.2 Generalized Likelihood Ratio

When determining if two sides of a speaker changing point are generated by the same or a different speaker, we can assume that both sides have features that are statistically independent and can be modelled with a multivariate Gaussian distribution. Following [9], we might think of applying a likelihood ratio test to decide between the two hypothesis, because if both sides come from the same Gaussian distribution, we can assume that it is indeed the same speaker. If we take the standard likelihood ratio test and replace the unknown model parameters for their maximum likelihood estimates, we have the Generalized Likelihood Ratio (GLR) test.

The likelihood that both sides of the speaker changing point come from the same distribution is $L_0 = p(X, Y|\Lambda)$ and the likelihood that each side come from a different distribution is $L_1 = p(X|\Lambda_1)p(Y|\Lambda_2)$, with Λ_0 and Λ_1 being the model parameters.

The test criterion is then:

$$R = \frac{\max L_0}{\max L_1} \tag{7}$$

To calculate these maximums we can use:

$$\max p(X|\Lambda) = [(2\pi \mathrm{e})^p |\hat{\sum}|]^{-\frac{1}{2}N} \tag{8}$$

where p is the feature dimension, N is the feature set size and $\hat{\sum}$ is the maximum likelihood of the model's covariance.

Since the GLR is always in the interval $(0, 1)$, we can take its logarithm to define a distance measure from it [9], [21, Ch. 4]. The GLR distance is then:

$$d_{GLR} = -\frac{1}{2}[N \log |S_1| + M \log |S_2| - (N + M) \log |S|] \tag{9}$$

In which N is the number of features in the first segment, before the hypothesized speaker changing point, S_1 is the sample covariance matrix of that segment, M is the number of features of the second segment, after the speaker changing point, and S_2 is the sample covariance matrix of that second segment. S is the sample covariance matrix of the union of both segments.

What would be left is to decide if both segments are from the same or a different speaker using the defined d_{GLR}. For this, we need to calculate the distance at each hypothesized speaker changing point, and declare that there is a speaker change when the distance is over a predefined threshold, which can be found experimentally as discussed in 4.2.2.1.

As defined, our d_{GLR} takes into account the mean and covariance estimates, but [9] suggests that distance measures not depending on mean estimates are less sensitive to environmental conditions and thus we prefer them in our implementation. For that, we replace S with $(NS_1 + MS_2)/(N + M)$. Another important detail, also from [9] is that the GLR depends on the speech segments length, so in order to use the proposed d_{GLR} we have to keep the length of the two segments constant, so this distance measure works best with the fixed sliding window approach discussed in section 3.3.4. Some authors, for example [17] propose modifications to this d_{GLR} so that it can be used with a growing window, and that could be a future improvement for our system.

3.3.3 Symmetric Kullback-Leibler Divergence

The Symmetric Kullback-Leibler Divergence (KL2) is a way of measuring how much two probability densities differ [15]. This enables us to compare the models estimated for the feature sets before and after the hypothesized speaker changing point, in order to conclude that if the models differ over a predefined threshold, they belong to different speakers.

The original *Kullback-Leibler divergence*:

$$KL(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \tag{10}$$

Measures the difference between probability distributions P and Q, and it is not symmetric. A symmetric distance measure, called *Kullback-Leibler distance* for speaker segmentation was defined in [24] as:

$$d_K L = KL(P||Q) + KL(Q||P) \tag{11}$$

We will be using this distance measure in our system, as one of the available metrics to perform speaker segmentation.

3.3.4 Fixed Sliding Window

We have already discussed the simplified case where we have a hypothesised changing point i and want to know if it is indeed a changing point or not. However, the selection of these trial points is not trivial.

Perhaps the simplest way to choose these points is to use a fixed sliding window approach [14]. In this approach, a first segment of audio is selected, of a predefined time duration, and a changing point is hypothesized exactly in the middle of its duration. Then any of the distance metrics discussed before is applied to the first and second part of the segment to determine if both parts belong to the same or different speakers. In any case, this "window" is slided forward by a predefined time in the audio stream, and the process gets repeated (Figure 4).

The segment size and how much to advance in each iteration are defined in our system by the parameters *Window Size* and *Window Step* as can be seen in 4.2. Proper selection of these parameters affects the performance, so they should be

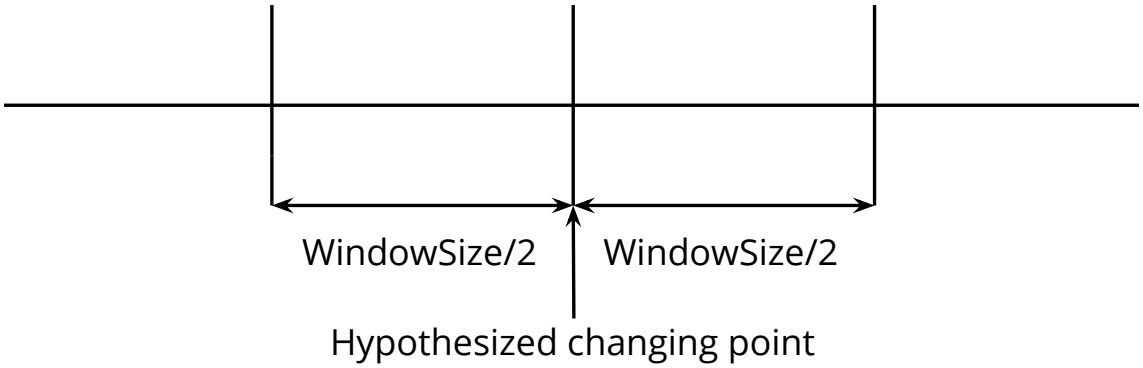

Figure 4: Fixed Window

carefully chosen. The bigger the *Window Size*, the more likely that there is a speaker change in it, but it is also possible that we choose a hypothesized changing where the first segment belongs to one speaker, and the second segment belongs to this speaker and one or more other speakers. In this case, most distance measures will identify this hypothesized point as a real changing point, when it is not. If, however, the *Window Size* is too small, the feature space of both segments could be filled in a way that depends mostly on what is being said rather than who is saying it, but in [8] it is proven that the feature space gets filled in an acoustic and speaker dependent way in just a few seconds, so we just need to make sure our *Window Size* is big enough, but not too big.

The simple nature of the fixed sliding window approach, where the size of both segments are the same, enables us to use any of the considered metrics, BIC, GLR or KL2 with it, something that doesn't work well with the growing window approach, since both segments have variable size. For a comparison of results using fixed sliding window with each different metric, see section 4.2.2.1.

3.3.5 Growing window and BIC segmentation

As discussed above, the proper selection of trial points where to check if there has been an speaker change is important. As opposed to the fixed sliding window already discussed before, in [3] the strategy used was a growing window. We start with a minimum window size, and we move i from start to end to this window, calculating the λBIC on each position. Then if the most negative λBIC is below 0 we determine that there lies a changing point, and the process continues again with a minimum window right after that point. If, however, there was no changing point detected in the window, it was enlarged by adding more data points at the end and performing the whole procedure again.

This process assumes that there is a single changing point present in each window, and for that the initial window is chosen to be deliberately small.

While very accurate when it was first described, this process has several limitations, mainly that it is very slow due to it testing the same hypothesised changing points again and again just adding a bit more of data at the end. For this reason, several heuristic improvements have been made over the baseline algorithm to improve its

speed.

Following the discussion in [26], instead of enlarging the window with a fixed amount of data points, we start enlarging it in small steps to account for quick speaker turns, like in a discussion, and progressively increase the step size until a maximum window size is reach, in order to speed up computations while keeping accuracy. Also, we avoid performing BIC tests at the very borders of the windows, since those represent one Gaussian with very little data.

Apart from the above discussed improvements coming from [26], we perform other optimizations. First, it should be noted that in this growing window approach, when calculating equation (5) after the window gets enlarged, the calculations of $\frac{N_1}{2} \log |\sum_1|$ are repeated from the previous time before enlarging the window. This is because the enlargement only affects the right side of the first hypothesised points, so the float resulting from the left side calculations can be saved to save time in further tests.

Another adjustment where we get a big speed improvement is by changing the way the BIC tests are performed inside the window. Instead of testing every possible changing point, we only consider points spaced by a fixed step distance in a first pass. By keeping this distance step small, for example 0.1 seconds, we try to avoid problems with local minima or skipping a zone of negative BIC distance. After this is done and we have the best candidate, we perform a second pass of BIC computations from the candidate time minus the step, to the candidate time plus the step, in order to fine-tune the best changing point to report.

In our system, the parameters affecting a growing window approach are the minimum *Window Size*, the minimum amount of time we grow the window, defined with *Delta Window Size*, and the maximum a window can grow, defined with *Window Step*, although that name can be misleading. All these parameters can be seen in section 4.2 and results in section 4.2.2.2.

3.4 Speaker Clustering

The last step after speaker segmentation is to perform speaker clustering. This includes labelling each speaker turn of the same speaker with the same label, in order to identify who spoke when. Again, several approaches are available in the literature, and some of the popular ones at the present moment are reviewed in [1].

Since the problem of measuring the similarity between segments is the same as in segmentation, the same metrics as for the segmentation can be used here. If we assume that the previous segmentation has good quality and segments contain only one speaker, we can perform a standard hierarchical clustering on the segments, merging those that are similar as being produced by the same speaker, and stopping when the most similar clusters have their distance over a specified threshold. In the case of BIC, this threshold can be set to 0, so we think it is the best choice because it requires less parameter tuning.

An example could be as follows: After segmentation we have 5 speaker turns, labelled A to E (Figure 5). The two closest speakers to each other by BIC distance are B and C, and thus they are clustered as speaker BC. In the next iteration the

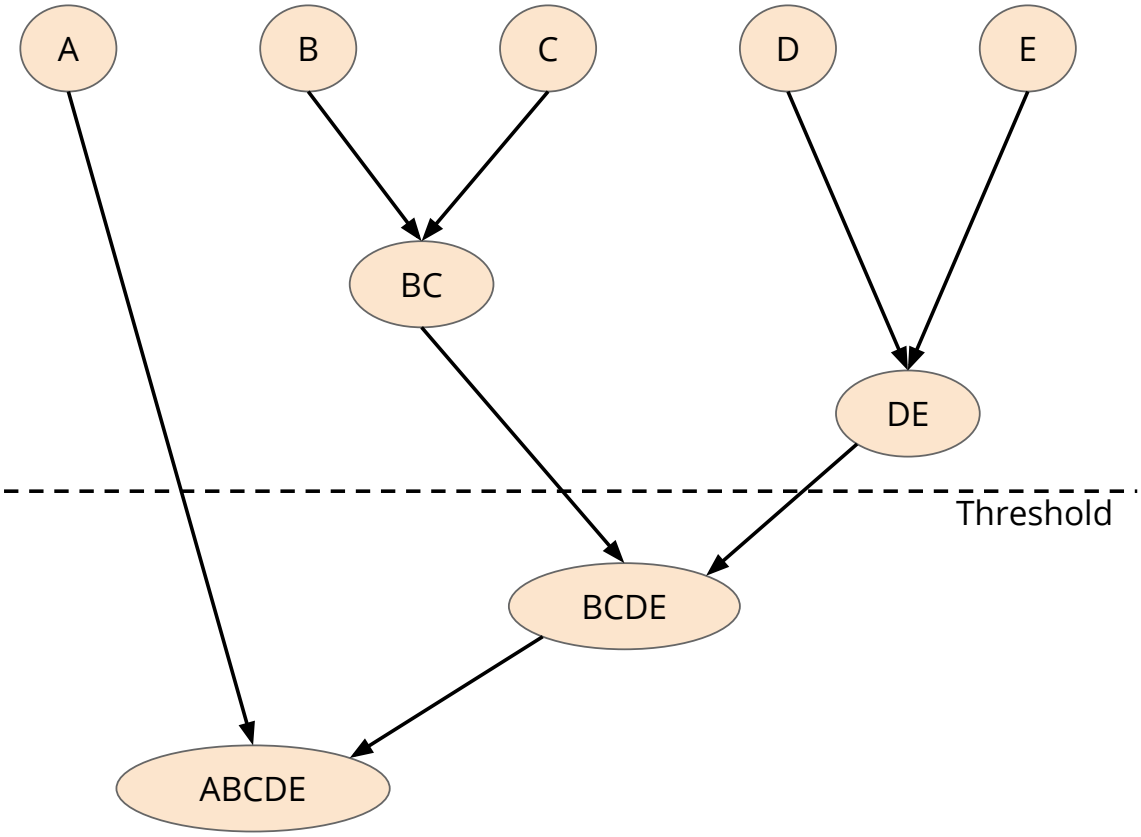

Figure 5: Sample hierarchical clustering

two closest ones are found to be D and E and they become speaker DE. Since the next two closer speakers, BC and DE has a distance bigger than the threshold, the clustering ends here and the system would output that there are three speakers: A, BC and DE. Modifying the λ penalty weight on equation 5 has the effect of altering the threshold.

This approach can fix segmentation errors where one turn is split into two segments, but not when one segment contains more than one speaker turn. Alternative approaches [1] perform clustering with iterative resegmentation to introduce new speaker change points when needed. This process can be slow, since, in general, several iterations are needed. However, this is a more robust approach and a good path for further improvements.

3.5 Evaluation Metrics

While there is abundant literature on how to evaluate the performance of a speech recognition system, some of which is discussed in section 2.2, there is considerably less literature on how to evaluate the performance of the speech diarization task. In this section we will discuss some references and their proposed metrics.

To start with the different evaluation alternatives, we will consider the speaker

diarization task of the *Rich Transcription* evaluation series, performed by the National Institute of Standards and Technology (NIST). These evaluation series started as an effort to evaluate different speech recognition technologies, such as *Speech-To-Text* (STT) and *MetaData Extraction* (MDE) tasks. From 2003 to 2009, one of the evaluated tasks was "Who spoke when", that is, speaker diarization. We will discuss their evaluation metrics as described in [6], and also revised in [19].

The main metric used in the NIST speaker diarization tasks is Diarization Error Rate (DER). This is basically the fraction of speaker time that is not labelled correctly. Secondary metrics, such as the speaker time with the wrong label, the time labeled as silence when there is speech, and the time labelled as speech when there is silence, named *Speaker Error time*, *Missed Speaker Time* and *False Alarm Time* respectively, are absolute times, not a fraction, and thus are only useful to compare different diarization systems when benchmarked in the exact same dataset, as in the NIST evaluation series. Since that is not our case, we do not use them, but we do use DER as our main benchmarking metric.

In our case, with no overlapping speakers, DER can be computed as follows:

$$\mathrm{DER} = \frac{\sum_{Segs}\Big(\mathrm{ST} * \big(\max(\mathrm{IS?}, \mathrm{SD?}) - \mathrm{CD?}\big)\Big)}{\sum_{Segs}(\mathrm{ST} * \mathrm{IS?})} \tag{12}$$

where:

$$\begin{aligned}
\mathrm{Segs} &= \text{Segments},\\
\mathrm{ST} &= \text{Segment Time in seconds},\\
\mathrm{IS?} &= \text{Is Speaker? Is there a real speaker talking? (1, 0)},\\
\mathrm{SD?} &= \text{Speaker Detected? Do we have a label for this speaker? (1, 0)},\\
\mathrm{CD?} &= \text{Correct Speaker? The label is the right one? (1, 0)}
\end{aligned}$$

And the lower, the better.

In addition to this, NIST ignores speech pauses of 0.3 seconds or less. These small pauses are considered to be normal in continuous speech and are an approximation of an uterance boundary pause. Furthermore, when computing the DER, a mismatch between the detected times and human labelled ground truth of 0.25 seconds, both before and after a segment, is considered to be correct. This is to account the difficulties for a human to correctly label the ground truth transcription with exact precision, as well as the subjective or philosophical discussion of when speech exactly begins in word-initial stop consonants [19]. We will take these considerations into account when computing the DER.

In addition to DER, we will be defining other metrics to measure the performance of our subsystems in isolation. To measure the performance of our VAD subsystem, we will define VAD Error Rate (VER). In VAD, there are two kinds of errors, Speech-as-Noise and Noise-as-Speech, and we can define VER as the fraction of incorrectly labeled time, or more precisely:

$$\text{VER} = \frac{\text{SaN} + \text{NaS}}{\text{Total Time}} \tag{13}$$

where:

$$\text{SaN} = \text{Speech as Noise},$$
$$\text{NaS} = \text{Noise as Speech}$$

As with DER, the lower the VER score is, the better. We can see results using this metric in section 4.2.1.

In the case of speaker segmentation, when searching for the best parameters for the subsystem we do not care too much about extra turns because the following clustering phase can conclude that this extra turn belongs to the same speaker as the previous one, thus fixing the issue. Missing turns are, however, a real problem, because the clustering will cluster all that as belonging to the same speaker and that will reflect in the final system performance. For this, we define the Missing Turns Error Rate (MTER) as the fraction of missing turns per total turns:

$$\text{MTER} = \frac{\#\text{ of missed turns}}{\#\text{ of total turns}} \tag{14}$$

And as usual, the lower the better. We can see results using this metric in section 4.2.2.

It is interesting to note that, while errors in voice activity detection are generally undesirable, some kind of errors can be irrelevant or even beneficial depending on the target application of speaker diarization. For example, as mentioned earlier, extra segmentation errors are usually acceptable, because the clustering phase can conclude that both segments belong to the same speaker, thus fixing the issue. An example of desirable errors is when the target application is per-speaker adaptation, to improve a speech recognition system. In this case, it is desirable for our speaker diarization system to separate the same speaker under different ambient conditions, in order to allow the per-speaker adaptation to adapt separately to this speaker with those different ambient conditions, likely resulting in better speech recognition performance, as mentioned in section 2.1.5.

4 Results and Discussion

In this chapter we will discuss what our particular experimental setup is, including the available data, which parameters and thresholds of our system require tuning for the best performance under the expected deployment conditions, and a comparison of our results under different configurations, as well as a comparison of our results with the current state of the art.

4.1 Experimental Setup

For testing our system, intended for speaker diarization of broadcast news, the Finnish broadcasting company *YLE* has gladly provided us with videos of their broadcast news during the year 2012. A full week of these videos was manually annotated with speech versus non-speech segment timings, speaker turn timings, and speaker labels, so that we have a baseline with which to compare our system performance, using the metrics described in section 3.5. The total amount of manually annotated audio is about 2 hours and 20 minutes.

4.2 Parameters and Results

Even though our task is well-defined, to perform speaker diarization in broadcast news, we have implemented a variety of methods to perform this task, as discussed in 3, so we must compare them. Also, each of these methods has different parameters we can fine-tune, a quick overview of them can be seen in table 1. We will discuss the meaning of each of these parameters here, and how to find the best values for optimal performance.

Since our system is composed of several subsystems in a pipeline (Figure 3), we can work to optimize the parameters for each subtask separately, as improvements in each step will lead to better performance in subsequent steps.

4.2.1 Voice Activity Detection

As seen in table 1, there are four parameters that affect our VAD subsystem. Of those, we will optimize the first two, that is, the minimum speech that constitute a turn, and the minimum non-speech that, if present, separates two consecutive turns.

Voice Activity Detection	
Parameter	**Description**
Min. Speech	Minimum speech duration that can constitute a turn.
Min. Non-speech	Minimum silence between speech to cut the segments.
Seg. Before Expansion	Add a padding before each segment.
Seg. After Expansion	Add a padding after each segment.
Speaker Segmentation	
Parameter	**Description**
Method	Fixed sliding, or growing window.
Metric	Can use BIC, GLR or KL2.
Window Size	For fixed windows, or minimum size of a growing one.
Window Step	How much to move or maximally grow the window.
Delta Window Size	Window minimum growing.
Threshold	For GLR and KL2 metrics.
Lambda (λ)	For the BIC metric.

Table 1: Different options and parameters.

To compare the performance of the different values given to these parameters, we will use VER as presented in equation 13.

Voice Activity Detection		
MS	**MNS**	**VER**
0.5s	0.3s	0.301
0.5s	0.5s	0.226
0.5s	0.75s	0.133
0.5s	1.0s	0.095
0.5s	1.25s	0.077
0.5s	**1.5s**	**0.0731**
0.75s	0.3s	0.436
0.75s	0.5s	0.359
0.75s	0.75s	0.217
0.75s	1.0s	0.159
0.75s	1.25s	0.125
0.75s	1.5s	0.120
1.0s	0.3s	0.545
1.0s	0.5s	0.481
1.0s	0.75s	0.350
1.0s	1.0s	0.281
1.0s	1.25s	0.190
1.0s	1.5s	0.180

Table 2: VER for different MS and MNS values, lower is better.

To make sure we have enough speech for a meaningful speech segmentation and

clustering, we explored values of *Minimum Speech* (MS) of 0.5, 0.75 and 1.0 seconds. For *Minimum Non-Speech* (MNS), we started with the NIST standard 0.3 seconds, and then used a range of values from 0.5 to 1.5 seconds, spaced by 0.25 seconds. We also ignored 0.25s differences between the manually annotated baseline scores and the automatically generated ones, following the same arguments as discussed in section 3.5 for DER.

Different VER of our data with these different values can be seen in table 2.

With our VAD subsystem and data, the best results were obtained with a *Minimum Speech* of 0.5 and a *Minimum Non-Speech* of 1.5 seconds, and we will use the VAD output that these values generate as input for final DER calculations.

4.2.2 Speaker Segmentation

For speaker segmentation, there are three methods we will test: growing window with BIC as discussed in section 3.3.5, and fixed sliding window with GLR and KL2 as discussed in section 3.3.4.

In this case, we will use MTER, defined in equation 14 to compare the performance of the different parameters. To isolate from the VAD errors, when fine-tuning these parameters we will start from a manually VAD template, so all the missed turns are a result of segmentation errors, thus enabling us to choose the best performing parameters for our segmentation subsystem.

4.2.2.1 Fixed Sliding Window

For this segmentation approach with metrics GLR or KL2, the biggest parameter selection problem is to find a suitable threshold for the metrics, as this is heavily data dependent.

The other parameters to tune are Window Size (WSz) and Window Step (WS) as presented in section 3.3.4. For Window Size, we deemed reasonable to try with sizes 3, 4 and 5 seconds, and after some testing we are fixing our Window Step at 0.5 seconds, since bigger steps result in lower performance.

For GLR, after some data exploration we concluded that a correct threshold for our data would lie between 2000 and 2500, as can be seen in table 3. Lower than 2000 values would result in even better MTER, but only at the cost of introducing a ton of extra segmentations. While that is not a big problem if clustering can fix it, as discussed in 3.5, if these extra introduced segmentations are too small they result in a slow and wrong clustering too, so we must ensure that the resulting turns have a minimum duration.

For KL2, we found that the threshold values are very low and hard to fine tune, since small threshold changes give very big MTER differences. The best threshold for our data seemed to lie between 10 and 20, as can be seen in table 4. However, the results are quite bad, and lowering the threshold even more, while improving the result, introduces far too many false positives.

With these results in hand, we can recommend GLR over KL2 for fixed sliding window approaches, because the GLR threshold is easier to fine tune for the data.

Speech Segmentation		
MWs	**Threshold**	**MTER**
3.0s	**2000**	**0.219**
3.0s	2250	0.395
3.0s	2500	0.596
4.0s	2000	0.307
4.0s	2250	0.368
4.0s	2500	0.535
5.0s	2000	0.351
5.0s	2250	0.342
5.0s	2500	0.447

Table 3: MTER with fixed sliding window and GLR, lower is better.

Speech Segmentation		
MWs	**Threshold**	**MTER**
3.0s	**10**	**0.316**
3.0s	15	0.509
3.0s	20	0.614
4.0s	10	0.570
4.0s	15	0.659
4.0s	20	0.693
5.0s	10	0.640
5.0s	15	0.693
5.0s	20	0.702

Table 4: MTER with fixed sliding window and KL2, lower is better.

4.2.2.2 Growing Window with BIC

For the growing window approach with BIC metric, the parameters to tune are the Minimum Windows Size (MWS), the Window Step (WS) that defines the maximum window size, the minimum window growing, that we call Delta Window Size (δWS), and the lambda (λ), as can be seen in table 1 and discussed in section 3.3.5. Results can be seen in table 5.

One of the first things we discovered when exploring the parameter space is that δWS is best set at 0.1 seconds, since bigger values can only affect the performance negatively, even though it makes the algorithm slightly faster. As such, in the results table 5 we omit the δWS column and assume that it is set at 0.1 seconds in all entries.

Another discovery is that lower λ values yield better performance. This is not surprising, since our MTER metric just take into account the missing turn, not the extra inserted turns, thus a lower threshold is desirable. The effect of different λ values for fixed MWS of 1.0, WS of 3.0, and δWS of 0.1 seconds can be seen in graph 6. In table 5 λ is fixed at 1.0.

Speech Segmentation		
MWS	**WS**	**MTER**
0.5s	2.0s	0.175
0.5s	3.0s	0.158
0.5s	4.0s	0.193
1.0s	**2.0s**	**0.140**
1.0s	**3.0s**	**0.140**
1.0s	**4.0s**	**0.140**
1.5s	2.0s	0.157
1.5s	3.0s	0.157
1.5s	4.0s	0.157

Table 5: MTER with growing window and BIC, lower is better.

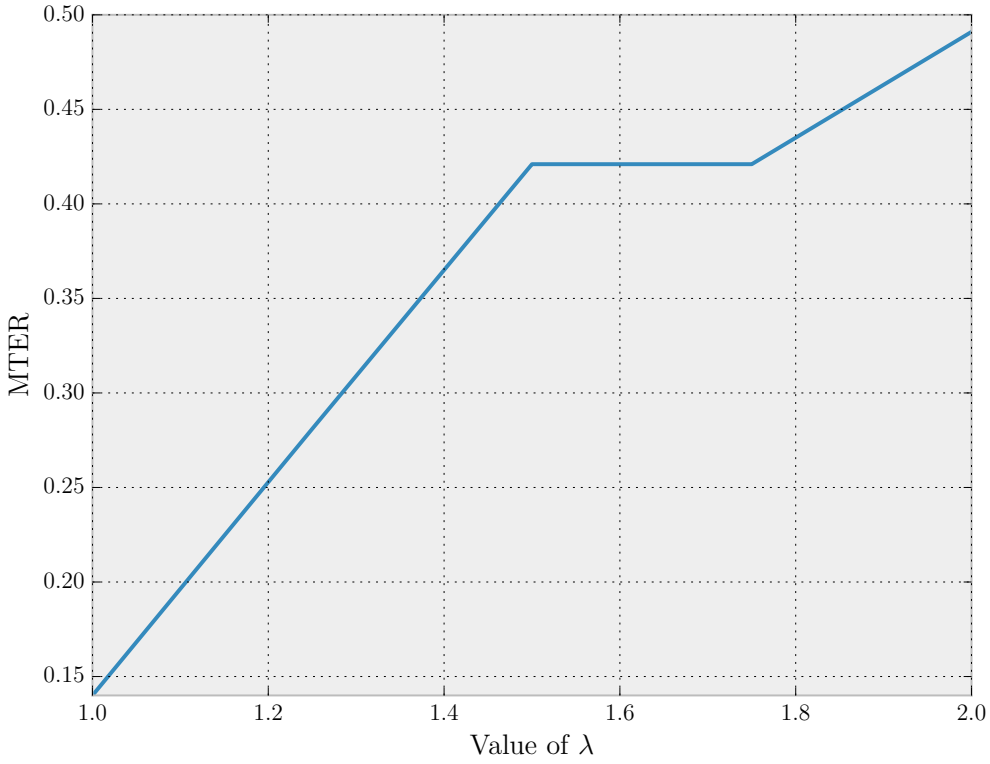

Figure 6: Growing window MTER with different BIC λ values

As seen in the results table 5, a Minimum Window Size (MWS) of 1.0 seconds seems to work best with our data. Smaller values have the peril of making a decision with too little data, and bigger values could skip a decision boundary. As for the maximum window, defined by Window Step (WS), when MWS is set at 1.0 seconds it does not have weight in the resulting MTER, but from the first two rows of the table we can see that it achieves a sweet spot at 3.0 seconds. The explanation is

similar: when the maximum window size is too small we can take decisions with too little data, whereas if it is too big it can take more than one speaking change point and thus make a wrong decision. These values are, nevertheless, data dependent.

The final values that seem to work better to perform speaker segmentation with growing window and a BIC metric in our data are thus a MWS of 1.0 seconds, a WS of 3.0 seconds, a δWS of 0.1 seconds and a λ of 1. These results are also better than any we got with fixed sliding window approaches, so we will use these parameters for our speaker segmentation from now on.

4.2.3 Speaker Clustering

Our final subsystem to test is speaker clustering. Here there is only one parameter to tune, namely the λ of the BIC metric in use. We will output the DER scores with different λ values, first taking manually segmented data, in order to evaluate the clustering subsystem in isolation, and then from the full pipeline as shown in figure 3. In this second case, the parameters used for each subsystem are the best for our data, as determined in each subsystem result section.

Speaker Clustering	
λ	**DER**
1.0	0.146
1.25	**0.117**
1.3	**0.117**
1.35	**0.117**
1.5	0.120
1.75	0.120

Table 6: DER with perfect segmentation

In figure 7 we can see that the best λ for our clustering is in the 1.3 range, as already mentioned in [26]. With this λ, and the best parameters for the previous subsystems, the final DER of our system is 0.15.

An analysis of the kind of errors committed by the system tells us that the very short turns are hard to cluster right, specially with speakers that only appear once. This is because with very short turns the feature space might get filled with characteristics of what is being said or the ambient conditions, and not with characteristics of the speaker itself, as pointed in [8]. However, since these turns are so short, these failures does not increase the total DER by much.

The other frequent error is when the same speaker gets clustered under two or three different labels, under different ambient conditions. This error would be desirable if we were going to perform per-speaker adaptation as mentioned in section 2.1.5, but for rich transcriptions we would like to limit the amount of these cases. A follow up Master's Thesis [13], to be published, hints at the use of multimodal features, such as video facial detection, to further improve these results.

One observed issue with our data is that the DER goes down for a while again at high λ values of over 1.75, before getting worse again. This is an artifact of the

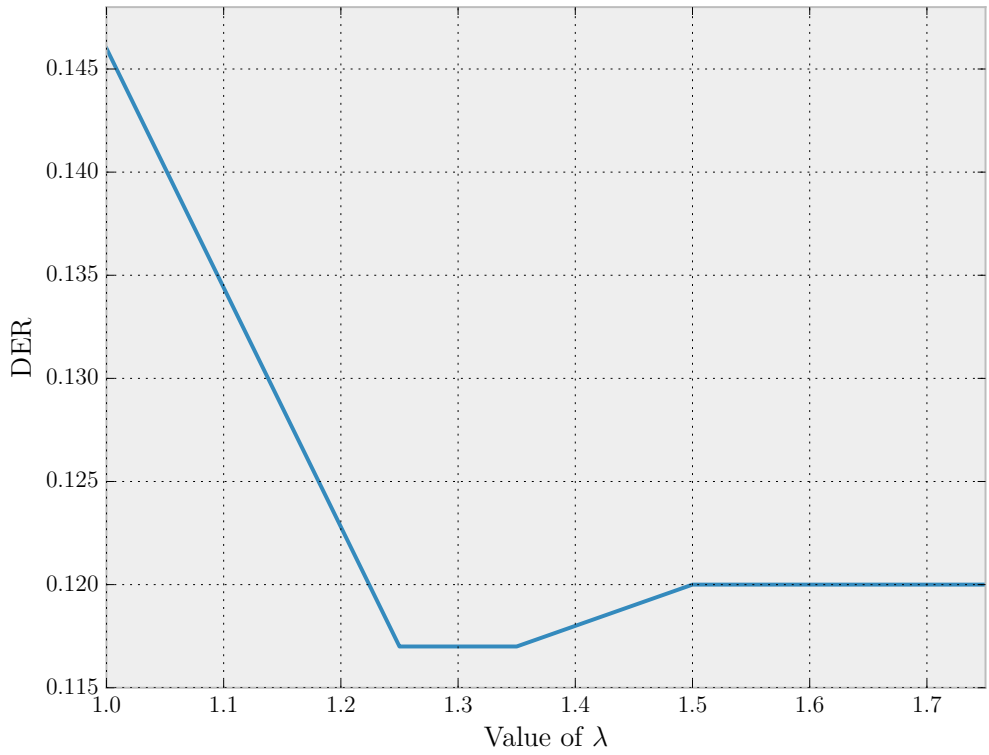

Figure 7: DER with different λ values

data structure: with broadcast news, the anchor speaker has much more speech time than any other speaker, and if we set a λ to a very high degree, the cluster that identifies this anchor speaker aggressively takes turns of other speakers, so even if that is wrong, having all the anchor speaker segments right no matter the background noise or recording conditions means a very low DER, which is a common problem in clustering unbalanced datasets. Since exploding this characteristic is undesirable, we must still conclude that a clustering λ of around 1.3 gives the best results in our system.

5 Conclusions and Future Directions

When starting this Master Thesis, we wanted to make an speaker diarization system suitable for the rich transcription of a big volume of Finnish language broadcast news. We also wanted a system that were compatible with the AaltoASR [5] package, for future developments and optional per-speaker adaptation. We have succeeded in these goals to an acceptable degree, but the produced system has some shortcomings.

The performance of our system, measured with DER, is good but lags behind the state-of-the-art when compared with recent advancements in the field, which often use data in English language and have a typical DER between 7.5% and 11% [1], compared to our system 15%, as seen in section 4.2.3. As mentioned, a follow up Master's Thesis [13], to be published, uses multimodal features, such as video facial detection, to further improve these results. Also each particular subcomponents could be further refined in isolation. For example, the VAD subsystem could see big improvements, since the state-of-the-art has around 2% of missed speech [1], and our system has a VER of around 7% as seen in section 4.2.1.

All in all, we have identified the best parameters for performing speaker diarization of broadcast news in the Finnish language, using popular methods, even when those parameters differ from the recommended for English language data. The produced system is compatible with the current AaltoASR package, so it can also be used for per-speaker adaptation, and is is already being used as a basis for further research.

References

[1] X. Anguera Miro, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals. Speaker diarization: A review of recent research. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(2):356–370, 2012.

[2] Susan Bartlett, Grzegorz Kondrak, and Colin Cherry. On the syllabification of phonemes. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 308–316. Association for Computational Linguistics, 2009.

[3] S. Chen and P. Gopalakrishnan. Speaker, environment and channel change detection and clustering via the bayesian information criterion. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, page 8. Virginia, USA, 1998.

[4] S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, 1980.

[5] Department of Acoustics and Signal Processing, Aalto School of Electrical Engineering. Aalto ASR Public Repository. https://github.com/aalto-speech, 2014. [Online; accessed 17-November-2014].

[6] Jonathan G Fiscus, Jerome Ajot, Martial Michel, and John S Garofolo. *The rich transcription 2006 spring meeting recognition evaluation*. Springer, 2006.

[7] M.J.F. Gales. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer Speech & Language*, 12(2):75–98, 1998.

[8] Herbert Gish and Michael Schmidt. Text-independent speaker identification. *Signal Processing Magazine, IEEE*, 11(4):18–32, 1994.

[9] Herbert Gish, M-H Siu, and Robin Rohlicek. Segregation of speakers for speech recognition and speaker identification. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, pages 873–876. IEEE, 1991.

[10] Teemu Hirsimäki, Mathias Creutz, Vesa Siivola, Mikko Kurimo, Sami Virpioja, and Janne Pylkkönen. Unlimited vocabulary speech recognition with morph

language models applied to Finnish. *Computer Speech & Language*, 20(4):515–541, 2006.

[11] Teemu Hirsimaki, Janne Pylkkonen, and Mikko Kurimo. Importance of high-order n-gram models in morph-based speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(4):724–732, 2009.

[12] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Pearson Prentice Hall, 2008.

[13] Subhradeep Kayal. Multimodal features for speaker diarization. Master's thesis, Aalto University School of Science, 2015. To be published.

[14] Thomas Kemp, Michael Schmidt, Martin Westphal, and Alex Waibel. Strategies for automatic segmentation of audio data. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1423–1426. IEEE, 2000.

[15] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951.

[16] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[17] Daben Liu and Francis Kubala. Fast speaker change detection for broadcast news transcription and indexing. In *Proceedings of the 6 th European Conference on Speech Communication and Technology (EuroSpeech) 3*. Citeseer, 1999.

[18] I. McCowan, D. Moore, J. Dines, D. Gatica-Perez, M. Flynn, P. Wellner, and H. Bourlard. On the use of information retrieval measures for speech recognition evaluation. *IDIAP Research Report*, 04–73, 2005.

[19] National Institute of Standards and Technology (NIST). The 2009 (RT-09) Rich Transcription Meeting Recognition Evaluation Plan. http://www.itl.nist.gov/iad/mig/tests/rt/2009/index.html, 2009. [Online; accessed 30-September-2014].

[20] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[21] Ulpu Remes. Speaker-based segmentation and adaptation in automatic speech recognition. Master's thesis, Helsinki University of Technology, April 2007.

[22] George Saon and Jen-Tzung Chien. Large-vocabulary continuous speech recognition systems: A look at some recent advances. *Signal Processing Magazine, IEEE*, 29(6):18–33, 2012.

[23] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[24] Matthew A Siegler, Uday Jain, Bhiksha Raj, and Richard M Stern. Automatic segmentation, classification and clustering of broadcast news audio. In *Proc. DARPA Broadcast News Workshop*, page 11, 1997.

[25] Sue E Tranter and Douglas A Reynolds. An overview of automatic speaker diarization systems. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(5):1557–1565, 2006.

[26] A. Tritschler and R. Gopinath. Improved speaker segmentation and segments clustering using the bayesian information criterion. In *Proc. Eurospeech*, volume 2, pages 679–682, 1999.

[27] Ville T. Turunen. *Morph-Based Speech Retrieval: Indexing Methods and Evaluations of Unsupervised Morphological Analysis*. PhD thesis, Aalto University School of Science, June 2012.