Speaker Diarization Error Analysis Using Oracle Components

Marijn Huijbregts, David A. van Leeuwen, and Chuck Wooters

***Abstract*—In this paper, we describe an analysis of our speaker diarization system based on a series of oracle experiments. In this analysis, each system component is substituted by an oracle component that uses the reference transcripts to perform flawlessly. By placing the original components back into the system one at a time, either in a top-down or bottom-up manner, the performance of each individual system component is measured. The analysis approach can be applied to any speaker diarization system that consists of a concatenation of separate components. Our experimental findings are relevant for most RT09s diarization systems that all apply similar techniques. The analysis revealed that three components caused most errors: speech activity detection, the inability to handle overlapping speech, and robustness of the merging component to cluster impurity.**

***Index Terms*—Rich transcription, speaker diarization, system analysis.**

I. Introduction

SPEAKER diarization is the task of determining: "Who spoke when?". Speaker diarization systems *segment* and *cluster* speech on the basis of speaker characteristics. Being able to group all speech from one particular speaker is a useful pre-processing step for various speech processing tasks. For example, speaker diarization information can be used to improve automatic speech recognition (ASR) performance (feature normalization or model adaptation [1]), for a meeting summarization application it is important to track *who* said *what* to *whom*, and a dialog act tagger needs utterance boundary information and can exploit speaker change information to model interruptions. Speaker diarization is also useful as an initial step for tracking people across recordings, making it possible, for example, to search for quotations of a specific person in multimedia collections.

Since 2002 the U.S. National Institute of Standards and Technology (NIST) has organized evaluations of speaker diarization technology in the broadcast news domain and since 2004 these evaluations have also been performed in the meeting domain [2]. We have participated in the 2006, 2007, and 2009 Rich Transcription (RT) evaluations[1] with a system based on a hidden Markov model architecture and Gaussian mixture models that are trained using only the speech in the data under evaluation. Although we have made numerous adjustments, the framework of our system is still very similar to the system initially proposed in [3].

After participating in the NIST rich transcription evaluation in 2006 we performed an analysis of our system based on a series of "oracle" experiments that enabled us to determine efficiently which component needed our attention most in future research efforts [4]. Oracle experiments are "cheating" experiments where the system or part of the system can make use of whatever knowledge is available [5]. For our analysis in 2006, oracle experiments were used to assess the performance of separate system components. We first replaced all system components with an oracle variant and then, one by one, placed the original components back into the system. By measuring the difference in system performance at each step, we determined the performance of each individual component. In our original paper we did not investigate to what extent the performance of one component influences that of others [4]. In this study, we extend our analysis in order to measure this dependency and to obtain more detailed and more reliable information about the components in our system.

This paper is organized as follows. First, in the remainder of this section we will describe the procedures during the NIST RT evaluations, the overall framework of our diarization system and that of the other systems that were entered into the most recent NIST RT evaluation. Next, in Section II we will describe our analysis method. In Section III we will summarize the results of our initial study and we will discuss the most important changes made to our system due to the first analysis. In Section IV, we will investigate the shortcomings of our most recent system and finally we will use the same oracle setup to investigate why the results at RT09s were poorer than expected.

A. NIST Rich Transcription Evaluation Series

Since 2004, NIST has conducted competitive evaluations of speaker diarization systems using recordings from multi-party meetings and lectures, as part of the Rich Transcription (RT) evaluations, the speaker diarization task must be performed with little knowledge of the characteristics of the audio or of the talkers in the recording. There are several conditions in which diarization systems are evaluated. The primary evaluation condition allows the use of audio recorded from multiple distant

[1]NIST did not organize the RT evaluation in 2008.

Manuscript received August 27, 2010; revised December 27, 2010 and April 29, 2011; accepted April 29, 2011. Date of current version January 13, 2012. This paper is based on research carried out for the BATS project within the research program IM-Pact funded by IBBT and ICTRegie. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gerald Friedland.

M. Huijbregts and D. A. van Leeuwen are with the Centre for Language and Speech Technology, Radboud University Nijmegen, The Netherlands (e-mail: m.huijbregts@let.ru.nl; d.vanleeuwen@let.ru.nl).

C. Wooters is with Next IT Corporation, Spokane, WA 99201 USA (e-mail: cwooters@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2011.2162318

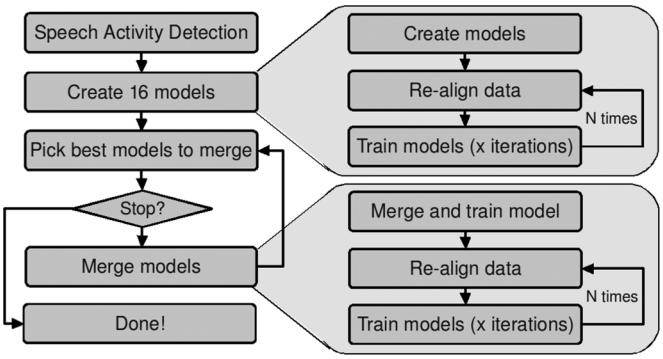


Fig. 1. Schematic representation of the speaker diarization algorithm. The steps for creating the initial 16 models and merging the models each consist of a number of training and re-segmentation iterations.

microphones (MDM) [2]. In this paper, we focus solely on the primary MDM condition in the meeting domain.

The metric used to evaluate the performance of each speaker diarization submission is called diarization error rate (DER) [2]. The diarization error rate is the sum of two error rates: the speech activity detection (SAD) error and the speaker classification error. The SAD error is the percentage of speech and non-speech that is misclassified. The speaker classification error is the time-weighted error due to misclassifying speakers. Note that if two speakers are talking simultaneously, both should be classified correctly. The speaker classification error consists of regular classification errors and of all overlapping speech errors.

Another metric, mainly used for system analysis, is *hypothesis speaker purity* (or cluster purity). The entire set of speech segments from one single hypothesis speaker might contain multiple reference speakers. The hypothesis speaker purity is the time-weighted fraction of the most occurring reference speaker for the particular hypothesis speaker. We use hypothesis speaker purity in our oracle experiments to determine which two clusters should be merged (see Section II).

Note that it is possible that multiple hypothesis speakers use the same reference speaker for purity calculation. If the purity of a hypothesis speaker is a hundred percent, this means that all speech of this speaker can be mapped to one single reference speaker and no noise of other speakers is present. However, this does not mean that the hypothesis speaker does not contribute to the DER as the reference speaker might be mapped to another hypothesis speaker.

B. Agglomerative Model-Based Speaker Diarization

The system described in this paper is based on a system that was originally proposed in [3]. Our system is model-based, but the models are created on the audio that is being processed and not on a pre-defined training set. It uses an agglomerative clustering algorithm: speech data is first partitioned into a large number of clusters and these clusters are merged pairwise until, presumably, the correct number of clusters is reached.

Fig. 1 presents the five steps of the algorithm. In the first step, speech activity detection, all non-speech audio is removed from the data and only speech is used in the remainder of the system. Second, during initialization the speech data is randomly distributed over a number of clusters and by iteratively training models for these clusters and re-segmenting the speech data, speaker models are created. In the third step, a distance metric is used to determine which two models are most similar and in the fourth step, if the two models are *not* regarded to be from the same speaker, the optimum number of clusters is reached and the process finishes. Finally, if the two models *are* indeed regarded to be from the same speaker, the two models are merged into one single model and the system continues at the third step.

In order to find the two models that are most similar, the Bayesian information criterion (BIC) distance metric is used [6]. This metric compares the sum of likelihoods of two models M_i and M_j each on their own data with the likelihood of a third model M_{ij} that is trained and evaluated on the combined data of M_i and M_j. If the combined likelihood of models M_i and M_j is smaller than the likelihood of M_{ij}, the two models are considered to be trained on speech from a single speaker.

The first to apply BIC to segmentation and clustering were Chen and Gopalakrishnan [7]. The need for a tunable model complexity parameter was later made superfluous in [3] by keeping the complexity the same before and after merging, effectively done by keeping the number of Gaussians in M_{ij} the same as the sum of Gaussians in M_i and M_j. For our system, we have adopted the approach in [3].

C. State-of-the-Art Diarization Systems

The agglomerative clustering algorithm described above is not only the basis for our system, it is also used in most other systems that were entered into the most recent NIST Rich Transcription evaluation. For example, the IIR-NTU [8] submission for RT09s applies HMM based agglomerative clustering. The system uses two feature streams: time delay of arrival (TDOA) features and MFCC features. The MFCC features are used for SAD and both feature streams are applied in the agglomerative clustering step. Before clustering is started, the IIR-NTU system performs a smart initialization step in which the audio is clustered in nine groups using the TDOA information.

Also the UPC [9] and UPM [10] systems apply agglomerative clustering using an HMM topology, Viterbi, and BIC. The UPC system incorporates a model for determining the prior probability of a speaker talking after other speakers have been speaking. The UPM system applies "frame purification" to filter out feature vectors that are not helpful for classification.

The LIA-Eurecom submission was the only system in RT09s that did not apply agglomerative clustering [11]. Instead this system starts with a single cluster for the entire recording and splits this cluster iteratively into multiple speaker clusters. Similar to the other systems, the LIA-Eurecom systems applies GMMs in an HMM topology for segmentation and clustering and the models are also refined iteratively.

Because the basis of all systems in RT09s is similar, we are convinced that the findings of our system analysis will be, to greater or lesser extent, applicable to the other systems. Further, even for systems that do not apply the same techniques as ours, the analysis method itself can be interesting. For every system that can be divided in a concatenation of separate components, the oracle-based analysis approach can be applied.

II. Oracle-Based System Analysis

The analysis described in this paper is based on oracle experiments. The term "oracle" stems from the fact that (part of) the system can make use of whatever knowledge is available. Even the optimal system output, the reference transcripts, may be used. In a sense, the system is an oracle that knows everything [5].

A. Oracle-Based Experiments in Other Studies

Oracle-based analysis is commonly used in various research fields and also for analysis of speaker diarization systems; oracle experiments have been conducted before. In [12], oracle experiments were used to determine the impact of overlapping speech in speaker diarization. It was shown that, with perfect overlap detection, the system could be improved in two ways: by skipping the overlap regions during clustering and by assigning the overlap regions to two speakers using an effective algorithm called "nearest-2." By applying the perfect overlap input, the performance was close to that of the oracle optimum overlap assignment. In the same research, oracle experiments were used to show that the extent of dimensionality reduction of location features should be variable for each recording.

In [13] oracle experiments were used to analyze existing and new stopping criteria. Both the speech/non-speech classification and speaker change detection components were replaced by oracle components and only the clustering and stopping components were tested. This study showed that, with perfect input, the proposed stopping component based on information change rate, outperformed the conventional BIC based stopping criterion.

The two papers used oracle experiments in order to focus on a specific problem in speaker diarization. By using oracle components as input for the component under test, the errors in each experiment could be attributed purely to the tested components. This advantage can also be regarded as a disadvantage because in the oracle setting, the component is not tested to be robust for non-perfect input. In this paper, we do not focus on one single component of the system. Instead, we will use the oracle technique to analyze each component under perfect conditions (top-down analysis, see Section II-D) and under actual input conditions (bottom-up, see Section II-E).

B. Oracle Based System Analysis of Our Diarization System

For our analysis, oracle experiments are used to assess the performance of separate system components. For each of these tests, the components that are *not* tested are replaced by an experimental setup that performs optimally by using knowledge of the reference transcription (the oracle knowledge) and the components that *are* tested are left unchanged.

In our initial set of experiments, at first all components are replaced by the oracle setup and the DER is measured. Then, one at a time and in a top-down approach, the actual components are placed back into the system. The increase in DER after replacing a component is attributed to shortcomings of that particular component. By replacing the components top-down, each newly replaced component receives its normal input, but its output is further processed by the oracle.

In our second set of experiments, we replace the components in a bottom-up approach. This way, the investigated component receives perfect oracle input and the output is further processed by the actual system.

Note that because of the iterative nature of the diarization algorithm (the loop in Fig. 1), it is not possible to perform the experiments completely top-down or bottom-up. We have chosen the order to replace the oracle components with the actual components in such a manner that this effect is minimized. Before discussing these experimental setups, we will first describe how the reference transcripts are used.

C. Reference Transcripts

For scoring the output of diarization systems, a reference transcript is manually created labeling the fragments where each speaker is talking [2]. In the oracle experiments, these reference transcripts are used in three different ways. In the first experiments of the series, the transcripts are used as input for the diarization system. One way of doing this is to replace the SAD output with the transcription. In this case the IDs of all speakers in the transcripts are replaced with one ID ("speech") and all overlap regions are replaced by single regions. Another method for using the reference transcripts as input, is to replace the initial clustering with a perfect clustering obtained from the reference transcript.

The reference transcripts are also used to take merging decisions. Instead of performing BIC, the oracle merge component scores a segmentation with the NIST scoring tools[2] and determines for each cluster which speaker it represents most (which speaker is classified by that cluster the longest period of time). The *purity* of each cluster is then calculated as discussed in Section I-B. The oracle merge component will then decide to merge the two clusters with the same representing speaker that has the highest purity.

Third, the system is modified so that an intermediate hypothesis segmentation file can be printed after each merging iteration for monitoring purposes. From these files we can calculate what is the best possible system performance, the diarization error rate if the system does not make any more mistakes, in two steps. First, for each speaker cluster we determine which speaker it represents the most. After labeling all IDs with the true speaker identities (an oracle approach for perfectly merging clusters with the same reference speaker identity) we use the reference transcripts for scoring.

D. Top-Down Analysis

The basis of our analysis consists of six oracle experiments. In the first experiment, all algorithm steps are replaced by an experimental setup that uses oracle information, and at each following experiment one of the components is placed back into the system (top-down). This procedure is depicted in Fig. 2.

Experiment 1, Optimal Models: If the algorithm would do a perfect job, the HMM would contain exactly one model per speaker and each model is trained on all the available speech of its speaker. Even if the algorithm would not make a single mistake and these optimal models were created, the system is not

[2]http://www.nist.gov/speech/tools.

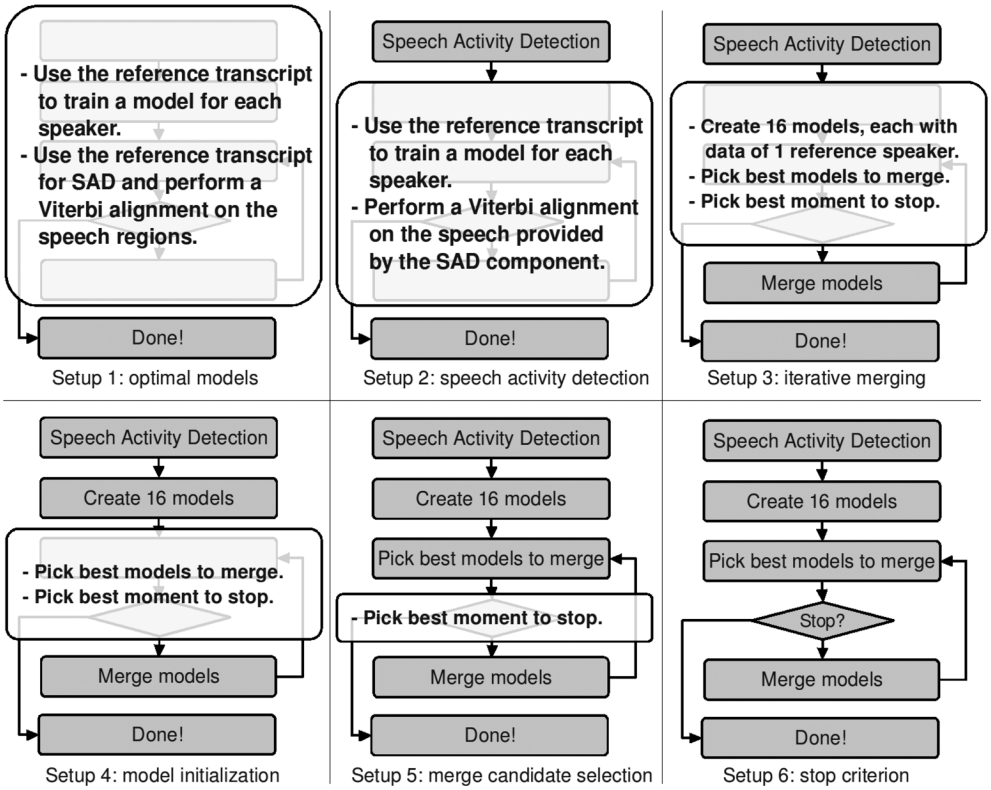


Fig. 2. The six experimental setups. The bigger light gray boxes represent the oracle components that replace the actual components. The oracle components perform their task using the reference transcript. In each experiment, one component is placed back into the system.

expected to have a perfect diarization score because the system is not able to model overlapped speech and because the models, with their limited number of Gaussians, might not be able to classify all speech perfectly. In order to test the system on these limitations, in the first experiment the reference transcription is used instead of the SAD component. For each speaker in the reference, one model is trained using all speech from that particular speaker. The total number of Gaussians in the system is the same as normal (80) and they are distributed over the clusters based on the amount of speech that is available for each speaker. After the models are created, a Viterbi pass is performed to find the final system result. This experiment gives us the error due to overlapping speech (further referred to as error A: the *overlapping speech* error) and the error due to imperfect modeling and segmenting the data (error B: the *modeling/segmentation* error).

Experiment 2, Speech Activity Detection: In the second experiment, the actual SAD component is placed back into the system. The models are still trained directly on the speech from the reference transcription, but the final segmentation is performed with the actual SAD segmentation. Compared to experiment 1 (the DER scored on this experiment subtracted by the error of experiment 1), this gives us the error that can be blamed on the SAD component (further referred to as error C: the *speech activity detection* error).

Experiment 3, Iterative Merging: Next it is tested what the influence of performing the actual merging iterations is on the final result. For this test, the reference transcript is used to create 16 initial models. Each model is created with speech of only one speaker, but because now 16 models are needed, the speech of each speaker is cut up in pieces so that multiple models can be trained for each speaker. The data are distributed such that each model is trained on an average amount of data (a person that spoke a lot in a meeting, will have a high number of initial models). The normal model initialization and iterative merging procedure are used, but the decisions about which models to merge and when to stop are performed by the oracle setup (as described in Section II-C). Compared to experiment 2, this gives us the error due to the procedure of creating the final models by merging the smaller initial models together (error D: the *iterative merging* error).

Experiment 4, Model Initialization: Instead of creating the initial models with use of the reference transcript, in this experiment the initial models are created normally by dividing the speech data randomly. However, the merging and stop decisions are still performed by the oracle setup. Therefore, compared to experiment 3, this gives us the error due to the shortcomings of the systems model initialization method (error E: *non-perfect initial clusters* error).

Experiment 5, Merge Candidate Selection: The oracle merge candidate selection component is now replaced by the actual candidate selection component-based on BIC. Compared to experiment 4, this gives us the error due to the shortcomings of the BIC selection component (error F: *combining wrong models* error).

Experiment 6, Stop Criterion: Finally, the remaining oracle setup that is able to decide when to stop merging perfectly, is replaced by the actual component that decides when to stop based on BIC. Compared to experiment 5, this gives us the error due to incorrect stop decisions (error G: *stop clustering too early/late* error).

E. Bottom-Up Analysis

By applying the oracle-analysis described above, it is assumed that the performance of each component is mostly independent of the performance of others. However, changing one component *is* likely to have an impact on other components. Therefore, blaming a single component for the increase of DER between two experiments, might sometimes give a slightly distorted picture. The first two blame classes, overlapping speech and SAD, do not suffer from this problem because the amount of overlapping speech is fixed (for a given test set) and SAD is the first component in the system and it does not need input from any other component.

A way to investigate the dependencies between the remaining components is to test each component with input of varying quality. The results of such experiments would probably provide very valuable information and possibly give a nuance the analysis results, but performing such a set of experiments is very elaborate. Also, creating a set of inputs with varying quality is not always straightforward. For example, if two SAD segmentations have the same SAD error it is not necessarily true that the diarization system performs equally well using both segmentations. Because of these two difficulties we restrict the possible input for each component to either perfect (or actually as good as we can do) or real input.

With the six oracle experiments described above, we have not yet tested the bottom components with perfect input. We only tested the string of components top-down, but not yet bottom-up. Therefore, in order to provide a better understanding of the last five blame classes, in a second series of experiments, we start with all components replaced by the oracle components and we replace the actual components bottom-up.

In the previous set of top-down experiments, we could blame the increase in DER when replacing an oracle component for the real component entirely on that one component, because the components following the evaluated component were all oracle components and therefore were not affected by the errors of the evaluated component. In this set of experiments, any errors made by a replaced component *will* influence the behavior of the components that follow and therefore we cannot blame the increase in DER on the component that is placed back, but we have to blame it on that one component and all components that use its input. If we want to specify where the error lies exactly, for each group of components that influence each other we have to do another top-down analysis.

Experiment 1, Cheat All: In the first experiment of this series we use oracle components for all four steps (SAD and overlap are not taken into account in this series). This experiment is not the same as the "optimal models" experiment in the first experiments series, because here we are not bypassing any of the components. In the optimal models experiment, we train models for each speaker directly. Now, we train 16 perfect models and we actually merge these models until we reach the (cheated) stopping point. For this experiment we now also use an oracle component for the re-segmentation step. In the previous series we either used the actual component ("Iterative merging") or we bypassed it completely. Now we cheat by simply not performing re-segmentation at all. Because the initial segmentation is already perfect, re-segmenting is not needed. When two clusters are merged, the training data of the two clusters is simply combined.

Experiment 2, Stop Criterion: The first bottom-up experiment is to replace all components with their oracle variant, except for the stop criterion component. This experiment will teach us how well the stop criterion is performing with "perfect" input.

Experiment 3, Merge Candidate Selection: Next, the component that is responsible for the selection of models to merge is also placed back into the system. The diarization error of this experiment minus the error of the first experiment can be blamed on both the stopping criterion and on the component for selecting merge candidates.

To find out which part of the error to blame on which of the two components, it is needed to perform an experiment where the stopping criterion component is replaced by its oracle variant. This oracle component is implemented as follows: we simply merge until only one cluster is left. We then calculate the DER for each possible number of clusters and we pick the number of clusters with the lowest error rate.

Experiment 4, Performing Re-Segmentation: In this experiment, we replace the oracle re-segmentation component (not doing segmentation at all) with the actual Viterbi segmentation step. If the models represent their data well enough, placing back the segmentation component will not influence the DER much. The increase in DER is shared between the three non-oracle components. If it needs to be specified which component is to blame exactly, two additional experiments need to be performed: one where the merging and stopping components are both replaced by the oracle components and one where only the stopping component is replaced by the oracle variant.

Experiment 5, Non-Perfect SAD and Initialization: Finally, we replace the perfect initialization step with the original step and we replace the perfect SAD segmentation with the actual segmentation. Note that for the previous experiments we trained the models on the perfect non-overlapping speech regions, but at the very end we perform one single Viterbi segmentation on the actual SAD segmentation. Because of this, we are able to compare the results of the previous experiments directly to the results of this experiment.

Again, in order to divide the increase in error over all the non-oracle components, a top-down analysis is needed of the segmentation, merging and stopping components in which the oracle variants are used and top-down replaced with the real components.

F. Discussion

In this section, we have described a system analysis method based on oracle experiments. With six components in the system, there may be 2^6 possible oracle experiment by replacing any of the components by their oracle counterpart. However, not all of these combinations are feasible or even make sense. We can investigate cumulative effects by adding components successfully, this can be performed top-down or bottom-up and if both are done, a total of eleven experiments are needed. Performing only the top-down analysis provides a good impression of the performance of each system component.

However, because the components do not work independently of each other, combining the top-down and bottom-up analysis provides a better picture.

An even better understanding can be formed when the bottom-up analysis is combined with a top-down analysis of all components that use the output of the component under evaluation. This type of analysis is a bit more complicated than the simple top-down and bottom-up approaches and it also requires some additional experiments (six experiments in our case).

III. Summary of the 2006 Analysis

In this section, we summarize the results of the top-down analysis that we performed in 2006 and we discuss how we adjusted our current system largely based on the results of this analysis. An in-depth description can be found in [4].

A. Top-Down Analysis

In our previous work [4], we have performed the six top-down oracle experiments on our 2006 system. In the first experiment, the entire algorithm to create the HMM topology was bypassed and the models were created directly. At each following experiment in a top-down manner, one step of the algorithm was placed back into the system (see Section II-D).

The error analysis showed that three factors contributed most to the total DER: the lack of being able to model overlapped speech, the speech activity detection itself and the initialization of the 16 clusters. Unfortunately, because each component is treated as a black box in the the error analysis, the analysis did not provide information on why these components performed suboptimal or how they could be improved. We therefore needed to investigate the three components further. In the remainder of this section we summarize our attempts to improve the three weakest components of our system. These studies are described in-depth in [1] and [14].

Cluster Initialization: One of the parameters in cluster initialization is choosing the number of initial clusters. In the original system, the number of initial clusters is fixed to 16, but for recordings of varying length, keeping the number of initial models fixed will result in models trained on too little data for short recordings and models trained on too much data for long recordings. Models trained on too little data tend to get overtrained and this might prevent models from the same speaker to be selected for merging. Models that are trained on high quantities of data might be so general that all models become similar and are all merged together. In order to prevent these two kinds of mistakes, after the analysis we changed cluster initialization so that the number of initial models was determined on basis of the amount of speech in the recording. We found that our system performed best with one initial model for each 40 seconds that the recording contains speech (using 5 Gaussians in each initial GMM).

We have not yet investigated other methods for cluster initialization improvement. One interesting method to investigate is that of IIR-NTU used at RT09s where the initial models are only trained on part of the data that is believed to be clean [8].

Speech Activity Detection: In our 2006 system, speech activity detection was done with the RT06s SAD component of ICSI [15]. This component used a two step algorithm. First, a silence-based setup was used to find all segments with low energy. It was assumed that silence was the only form of non-speech in the meetings and that this first step was able to find enough representative speech and silence segments to use in the second step. In this second step, the segments were used to train a model-based system with two states: one for speech and one for non-speech. The HMM was used to re-segment the data and using the new segmentation, new Gaussian mixture models (GMMs) were trained. After a number of iterations the final speech/non-speech segmentation was obtained.

The appealing feature of the original SAD algorithm was that it did not apply any models or parameters that needed tuning on an in-domain training set. Unfortunately, the energy-based method used to generate an initial segmentation for training the speech and silence models did not always work well. When audible non-speech is expected to be present in the audio, a bootstrap classification based on silence will not be sufficient. Our new approach addresses the problem by applying a model-based classification component to create the bootstrap classification. After the initial classification step, three models are trained on the audio to be processed: a model trained on silence, a model trained on audible non-speech, and a model trained on speech. Each of these models is trained on the data to be segmented. By applying the three models, the system is able to perform high-quality SAD.

The downside of our new method is that the models for the bootstrapping classifier need to be trained on external data. Fortunately, the initial segmentation does not need to be perfect. We have shown that it is even possible to use bootstrapping models trained on Dutch broadcast news data for use in English meeting recordings.

Overlapping Speech: In [12], it is shown that overlapping speech is a problem in two ways. First, because our system is not able to output overlapping speech segments. The Viterbi segmentation only outputs the one most likely speaker at each time. Second, during the training process of the speaker models, the overlapping speech segments act as noise. Because the overlapping speech is not at all similar to the speech of the individual speakers, it only degrades the models and with that, the final segmentation.

In [14], we have attempted to improve our system by trying to avoid to use overlapping speech segments during the model training phase and by assigning multiple speakers to these overlapping regions. In line with our approach for diarization, we tried to detect overlapping speech segments without the use of models trained on a development set. Instead, we generated overlapping speech models during a first diarization run and applied these models in a second iteration. In [14], we show that by applying this two-pass approach, it is possible to obtain a modest gain in system performance (0.33% DER absolute),[3] but we feel that this very modest system improvement does not justify inclusion of this component in our detailed analysis in

[3]This result is obtained on the same test set as used in Section IV where we show that the total error due to overlapped speech is ten times as high.

TABLE I
THE 27 CONFERENCE MEETINGS OF THE TEST SET

Meeting ID		
AMI20041210-1052	AMI20050204-1206	CMU20050228-1615
CMU20050301-1415	CMU20050912-0900	CMU20050914-0900
CMU20061115-1030	CMU20061115-1530	EDI20050216-1051
EDI20050218-0900	EDI20061113-1500	EDI20061114-1500
ICSI20000807-1000	ICSI20010208-1430	NIST20030623-1409
NIST20030925-1517	NIST20051024-0930	NIST20051102-1323
NIST20051104-1515	NIST20060216-1347	TNO20041103-1130
VT20050304-1300	VT20050318-1430	VT20050408-1500
VT20050425-1000	VT20050623-1400	VT20051027-1400

TABLE II
RESULTS OF THE ORACLE EXPERIMENTS WITH- AND WITHOUT APPLYING DELAY FEATURES

Oracle experiment	MFCC % DER	MFFC&delay % DER
1. Optimal models	5.70	4.98
2. Speech Activity Detection	8.90	8.18
3. Iterative merging	10.09	8.66
4. Model Initialization	10.89	9.19
5. Merge Candidate Selection	14.24	11.20
6. Stop Criterion	16.50	12.66

this work. Although we feel that our approach is promising, we clearly need to perform more research to improve our overlap detection system. Note that a number of other research institutes currently also investigate the overlapping speech problem [12], [16].

Delay Feature Stream: One improvement, adding a delay feature stream, was not obtained by performing the analysis but by comparing our system to that of others (see Section I-C). The delays between microphones with which a sound is recorded are a by-product of the beam forming toolkit and can be used as a second feature stream. The probability density function of each state is then modeled by two GMMs: one for the MFCC stream and one for the delay stream. In the original study in [17], this decreased DER by 21% relative. We adopt this approach for the system that we analyze in the following section.

IV. ORACLE-BASED ANALYSIS

The performance of our system on the NIST 2009 rich transcription evaluation was poorer than expected. Instead of analyzing our system on the evaluation set directly, we decided to first analyze its behavior on a bigger test set, composed from recordings of earlier NIST evaluations. In this section we discuss the evaluation on this set. Both the top-down and bottom-up series of oracle experiments are performed as described in Section II-D. Table I contains the meetings from which we have compiled the test set. The reference segmentations were obtained by forced alignment of the reference speech transcriptions in order to avoid inconsistencies in the placement of segment boundaries.

A. Top-Down Analysis

In Table II, the results of the six oracle experiments are listed for our most recent system with- and without the use of delay features. The SAD error on our test set is 6.7%, consisting of 4.8% missed speech and 1.9% false alarms.

As can be seen in Table II, the delay features improve the system considerably for each experiment. The improvement in the real system, without the use of oracle components, is 3.84% DER absolute (experiment 6).

The six experiments from Table II were used to perform the blame assignment. For example, the error due to the inability to model overlapping speech (error A) is the missed speech error from experiment 1 where a perfect (one-speaker) segmentation was used. Note that this error of 3.5% (error A) is just the fraction of overlapping speech. The remaining part of the error in experiment 1 is due to imperfect modeling (error B). The SAD error (error C) is obtained by subtracting the overlapping speech error (error A) from the SAD error of the actual component (6.7%) in experiment 2.

In this manner we have assigned part of the overall diarization error rate to each of the system components (see Section II-D). In Table III the contribution of each component to the total DER is listed. Solely relying on these top-down analysis results it seems that overlapping speech and speech activity detection are still responsible for a large fraction of the error. New compared to our previous analysis is that, especially for the system setup without delay features, a high percentage of the DER is due to errors in combining models. Twenty percent of the total error is due to model merging errors.

TABLE III
BLAME ASSIGNMENT ON THE TEST SET

Error name	MFCC % DER	MFCC&delay % DER
A. Overlapping speech	3.50 (21.21)	3.50 (27.65)
B. Modeling/segmentation	2.20 (13.33)	1.48 (11.69)
C. Speech Activity Detection	3.20 (19.39)	3.20 (25.28)
D. Iterative merging	1.19 (7.21)	0.48 (3.79)
E. Non-perfect initial clusters	0.80 (4.85)	0.53 (4.19)
F. Combining wrong models	3.35 (20.30)	2.01 (15.88)
G. Stop clustering too early/late	2.26 (13.70)	1.46 (11.53)
System DER (sum)	16.5 (100.0)	12.66 (100.0)

B. Bottom-Up Analysis

Because the component for model merging seems to function suboptimal, it is interesting to analyze the system in the bottom-up approach described in Section II-E. For the system set-up that does not apply the delay features, we have performed the five oracle experiments for this analysis and listed the results in Table IV. What stands out is that the stopping criterion experiment resulted in a very high error rate of 42%. We experienced that testing the stopping criterion on its own is difficult. For each recording, at one point the BIC score is negative for a pair of clusters that should be merged according to the oracle merging component. If the system is actually stopped at this point, the DER is very high. Normally, the merging component provides the pair of clusters with the highest BIC score—this is not necessarily the same as found by the oracle, but also not necessarily wrong. This will lead to higher BIC scores and to a lower probability of premature stopping. This shows that it is hard and maybe even less meaningful to separate the merging

TABLE IV
RESULTS OF THE BOTTOM-UP ORACLE EXPERIMENTS ON THE TEST SET WITH A SYSTEM SETUP WITHOUT DELAY FEATURES

Oracle experiment	% DER
1. Cheat all	9.21
2. Stopping criterion	42.0
3. Merge candidate selection	10.22
4. Perform re-segmentation	15.75
5. Non-perfect initialization and SAD	16.50

and stopping components. The results of the remainder of experiments on the test set are more according to expectations. The difference in DER between each step is similar to the differences observed in the top-down analysis.

We observe that when given perfect input, the step for selecting models to merge does not perform poor at all. Both the stopping criterion and the models selection component perform well with perfect input, but when re-segmentation is switched on, the performance drops. Note that in the fourth experiment when re-segmentation is performed, the initial segmentation is still perfect. This means that somehow the perfect segmentation is corrupted during the iterations of re-training the models and re-segmenting the data. We have investigated three hypotheses that can cause this performance drop: 1) it is possible that the perfect segmentation consists of so many short speaker segments that it is impossible for the system to generate a segmentation that is close enough to the truth; or 2) the initial models are not trained well enough to generate a good initial segmentation so that the system starts essentially with an almost random segmentation after the first Viterbi run (as is the case normally without cheating); or 3) the merge candidate selection and stopping components are not robust enough to handle non-perfect input.

1) Impossible Segmentation: If it is true that with the chosen HMM topology and the restricted number of Gaussians for each GMM it is not possible to generate a segmentation close enough to the truth for the remaining steps to work with, then the optimal segmentations that where generated in the second experiment of the first series would have to be very poor input for the system. In the second experiment, the actual SAD segmentation was used to train a GMM for each speaker in the recording with as many Gaussians that would optimally be used for each speaker in the normal system setup. The final Viterbi segmentation in that experiment is generated with all the limitations that the system has due to its topology or number of Gaussians and it is as close as the system can get in generating the perfect segmentation.

In the third experiment of our bottom-up analysis, we have replaced the reference segmentation by the near-perfect segmentation output of the second experiment from the top-down analysis. If the first hypothesis is true, the model combining and stopping step should fail on this segmentation and the output of this experiment should be poor. However, the result of this experiment for our test set was 10.93% DER. This is similar to the DER of the third experiment (10.22%). In fact, the speaker error was 4.10% in both experiments and the difference is completely due to the increase in SAD error.

2) Poor Initial Modeling: It is very easy to test the hypothesis if the first models are too weak to generate good initial segmentations. If this would be the case, the segmentation after the first few Viterbi iterations would be so poor that even if all clusters would be assigned to the actual speakers (perfect merging and stopping), the diarization error rate would be very high.

TABLE V
BLAME ASSIGNMENT ON RT09S. FRACTION OF THE DER, IN %, ABSOLUTE, AND RELATIVE IN PARENTHESES

Error name	MFCC	MFCC&delay
A. Overlapping speech	5.6 (18.08)	5.6 (21.04)
B. Modeling/segmentation	2.34 (7.55)	1.36 (5.11)
C. Speech Activity Detection	5.3 (17.11)	5.3 (19.92)
D. Iterative merging	3.85 (12.43)	1.32 (4.96)
E. Non-perfect initial clusters	1.43 (4.62)	0.02 (0.08)
F. Combining wrong models	6.83 (22.05)	9.38 (35.25)
G. Stop clustering too early/late	5.63 (18.17)	3.63 (13.64)
System DER (sum)	30.98 (100.0)	26.61 (100.0)

We stored the segmentation after each Viterbi iteration and relabeled and scored these segmentations as described in Section II-E. The DER after the first Viterbi run (and with perfect merging and stopping) is 9.82%. This is only 0.61% above the optimal result of 9.21%. After ten iterations, the best possible DER is 11.26%, which is still reasonable. We therefore deduce that modeling and re-segmenting is going well and that the problem must lie in the robustness of the component for picking clusters to merge.

3) Robustness Problem: We observe that the merging and stopping components seem to work fine for (near-)perfect input, but that the performance drops as soon as the input is corrupted by imperfect segmentation. With the reference input, the DER is 10.22% and with the near-perfect input in the experiment above ("Impossible segmentation") the DER was 10.94% while with linear segmentation and even when perfect initial models are created, the DER is 16.5% and 15.75%, respectively.

In the previous two experiments, we could not prove that the input is unfit to be used by the merging component. We therefore conclude that this component is just not robust enough for the actual input and that the system performance can only be improved by either developing (near-)perfect initial components or by developing a more robust merging/stopping component.

In the next section, we try to specify if it is the merging component that is not robust, if it is the stopping component or if both components are not robust against corrupted input.

V. RT09S EVALUATION

The performance of our system on the RT09s evaluation set is 30.98% DER for the setup without delay features and 26.61% DER for the setup that does apply delay features. In Table V, we have conducted the top-down blame assignment on the RT09s data.

What stands out from this analysis is that there is a lot more overlapping speech in this set than we have observed in the test set, that the SAD component is not performing as well as before and that the merging and stopping components are not performing well. The fact that we observe a lot more overlapping speech indicates that the meetings are less structured and perhaps more informal than before (more interruptions, people arguing) and therefore more difficult to process.

All the overlapping speech and the increase in false alarms of the SAD component (the sum of errors A and C; 10.9% for

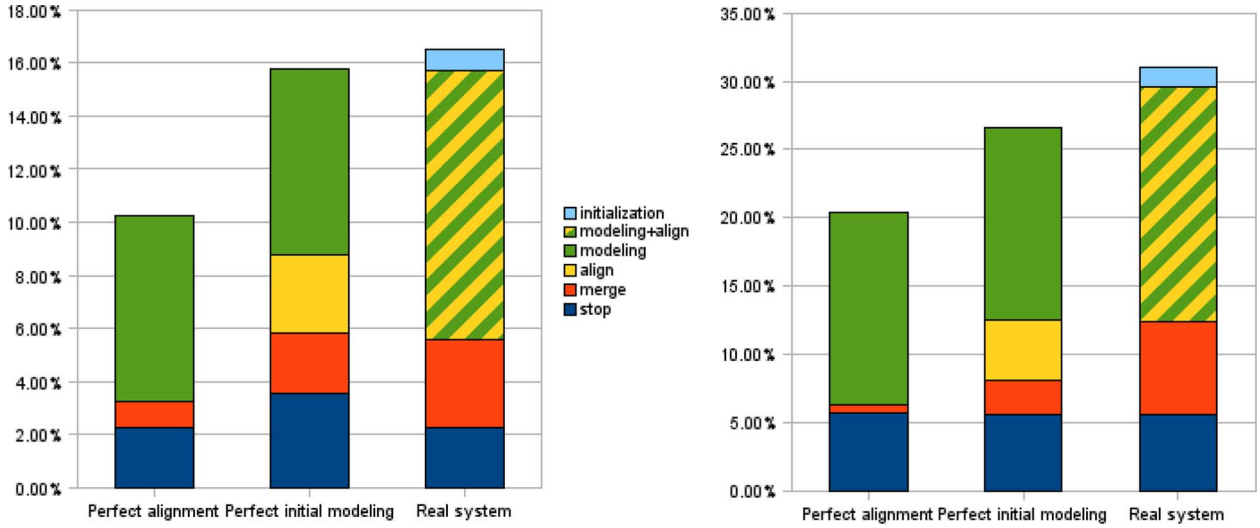


Fig. 3. Performance of the stopping criterion and merging component of the system in three cases of Table IV (perfect segmentation, perfect initial modeling and for the real system) obtained by a combination of top-down and bottom-up experiments. Left: the test set; right: the RT09s evaluation set).

RT09s against 6.7% for the test set) act as noise for the other components. This might explain why the merging and stopping component are not performing well. In order to test this hypothesis we have performed an experiment on the system without delay features and without any oracle components, except that for training the models we have filtered-out all overlapping speech and SAD false alarms. If the increase in error is due to these two sources of noise, the DER should drop considerably in this experiment. For the test set, this was only true in part. The DER dropped from 16.5% to 14.83%. For the RT09s evaluation set the DER was reduced from 30.98% to 27.51%.

A. Top-Down, Bottom-Up Analysis

For both the test set and the RT09s evaluation set, we have performed a bottom-up analysis where we also performed a top-down analysis for each bottom-up experiment so that we could observe the growth in error for each component for each bottom-up step (see Section II-E). Fig. 3 contains the results of this analysis for the system without delay features. We have focussed on the stopping criterion and merging component in three cases: one where the perfect initial segmentation is used throughout the entire process (cf. Table IV, line 3; Fig. 3, 1st bar), one where we started with the perfect segmentation but re-segmented the data during the process (line 4; second bar) and finally the case where we ran the system without cheating (line 5; third bar). Note that the error region "modeling" in this graph also contains the errors due overlapping speech and SAD.

Fig. 3 shows that, as we have noted before, Viterbi segmentation is performing well and that the stopping criterion and model selection for merging, even with perfect input, are not performing well. It is surprising that for both sets, the stopping criterion performs poorly even if the perfect segmentation is used. In this case, the clusters are very pure and it should not be very difficult to determine the optimal stopping point. The merging selection component is not robust against corrupted input. With perfect input the component performs very well, but unfortunately, the performance drops for the full system.

VI. Discussion

In this paper, we have analyzed our speaker diarization system using oracle components. For each component an oracle variant has been developed and in a set of experiments we have replaced one or more components with their oracle variant and measured the performance of the remaining components. Although the oracle analysis method has its shortcomings, we gained a good insight of the performance of individual system parts. In this section, we first discuss our findings of the analysis method itself and then summarize the results of the experiments.

A. Oracle Based Component Tests

The golden rule of performing experiments is to change only one aspect of the experimental setup between experiments. Applying this rule for experiments on a reasonably complex application such as a speaker diarization system, the number of experiments needed grows combinatorically. For the RT09s evaluation for example, we have stored the logging of 398 experiments and performed at least as many that we did not log. For benchmarks we will probably stick to the tactics of trying every possible system setting to tune the system as well as possible, but we do feel that for obtaining better insights during development it is better to also test components in an oracle-based experimental setup.

The oracle-based analysis as described in this paper has two advantages over testing the entire system. The total analysis is done with only 17 experiments.[4] After these experiments, it is clear for each component how it performs under perfect conditions and how robust it is against errors of other components. This information will help to focus the remaining experimental efforts towards the components that need most attention.

Another advantage of the oracle setup is that it can be used to speed up individual experiments. When testing a new component it is not needed to run the entire system, but instead oracle components can be used for the part of the system that is not

[4]Six for the top-down analysis, five for the bottom-up analysis and six for the combined bottom-up/top-down analysis.

tested. Later, the component robustness can be tested by placing back one or more other components.

We also acknowledge a number of disadvantages of the method. The first shortcoming of the method is that although we test the components with input of various quality to measure their robustness, in the end the system performance will depend on one particular set of components and the interaction between these components. It is possible that a component change that seems to work fine during analysis has a negative effect in the actual system. The system is more than the sum of its components. We believe that this might especially be true for major component changes but we have not yet experienced this ourselves.

The analysis is reasonably straightforward once an oracle is created for each component; it is possible to automate the entire analysis and run it overnight to have a clear picture of the system performance every day. Unfortunately, developing the entire oracle environment and creating an oracle for each component is quite elaborate. This is only a minor problem if the architecture of the system does not change often, but when the architecture *does* change, it is likely that also the oracles need to be developed again.

Finally, the error analysis can be a little bit misleading because it is not necessarily true that the component that is performing the least is also the component that can be improved most easily. After a lot of effort on the development of our SAD component for example, we still measure high errors due to misclassified speech and non-speech.

B. Component Performance

In this paper, we have analyzed our most recent speaker diarization system on a test set of 27 recordings and on the RT09s evaluation set. The analysis showed on both sets that speaker modeling, the iterative retraining process and initialization perform reasonably well (errors B, D, and E). Judging solely on the basis of the top-down analysis of the test set, it seemed that the lack of handling overlapping speech and misclassifications in speech activity detection are responsible for a large part of the diarization error rate (errors A and C). The overlapping speech problem was shown even more clearly in an additional test where we ran the system without training the models on the overlapping regions (Section V). This experiment showed that on top of the errors made because of the lack of assigning speech to more than one speaker, another 1.67% DER (absolute) is lost because of training errors due to overlapping speech.

In the bottom-up analysis of the test set and also in the top-down and bottom-up analysis on the RT09s evaluation set it became clear that also the merging step and stopping criterion were responsible for a large part of the DER (errors F and G). Without combining the bottom-up and top-down analysis it is hard to tell which of the two components is most to blame for the poor performance. In Section V-A, we combined both methods and discovered that the merging component performs well with pure clusters, but that it starts to make more errors if the input quality degrades. The stopping criterion is not affected much by the input quality. Especially on the RT09s evaluation set it performs poorly on both perfect input and on the actual input (5% DER absolute).

In Section IV-B, we further investigated the role of the SAD and resegmentation steps in the poor performance of the merging component. The experiments show that the merging component performs well when SAD and resegmentation are done perfectly, but also when SAD is close to perfect, that is, when overlapping speech errors are included in SAD and minimum duration constraints are enforced (the "Impossible segmentation" experiment). However, merging performance drops for the actual SAD input. Although the top-down analysis shows that the SAD component can be improved considerably and that overlapping speech detection can potentially improve the system, future research is needed to tell to what extend the merging component will negate improvements to SAD or overlap detection.

ACKNOWLEDGMENT

The authors would like to thank the SARA Computing and Networking Services (www.sara.nl) for their support in using the Lisa Compute Cluster.

REFERENCES

[1] M. Huijbregts, "Segmentation, diarization and speech transcription: surprise data unraveled" Ph.D. dissertation, Univ. of Twente, Enschede, The Netherlands, Nov. 2008 [Online]. Available: http://doc.utwente.nl/60130/

[2] J. G. Fiscus, J. Ajot, and J. S. Garofolo, "The rich transcription 2007 meeting recognition evaluation," in *Multimodal Technologies for Perception of Humans*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2008.

[3] J. Ajmera, H. Bourlard, I. Lapidot, and I. McCowan, "Unknown-multiple speaker clustering using HMM," in *Proc. Int. Conf. Spoken Lang. Process. (ICSLP)*, Denver, CO, 2002.

[4] M. Huijbregts and C. Wooters, "The blame game: Performance analysis of speaker diarization system components," in *Proc. Interspeech*, Antwerp, Belgium, Aug. 2007.

[5] R. V. Binder, *Testing Object-Oriented Systems: Models, Patterns, and Tools*. Boston, MA: Addison-Wesley, 1999.

[6] G. Schwartz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.

[7] S. 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*, Herndon, VA, 1998.

[8] N. T. Hieu, "Speaker diarization in meetings domain," Ph.D. dissertation, Nanyang Technol. Univ., Singapore, 2009.

[9] J. Luque, X. Anguera, A. Temko, and J. Hernando, "Speaker diarization for conference room: The UPC RT07s evaluation system," in *Multimodal Technol. Percept. Humans*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2008.

[10] J. M. Pardo, R. Barra, and B. Martinez, "The UPM RT09 meetings evaluation system," presented at the NIST RT09 Workshop, May 2009 [Online]. Available: http://www.itl.nist.gov/iad/mig/tests/rt/2009/workshop/upmrt09-presentation-2009-5-29-2.pdf

[11] S. Bozonnet, N. W. D. Evans, and C. Fredouille, "The LIA-Eurecom RT'09 speaker diarization system: Enhancements in speaker modelling and cluster purification," in *Proc. ICASSP*, Dallas, TX, Mar. 2010, pp. 4958–4961.

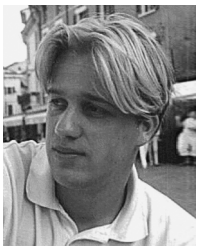
[12] S. Otterson, "Use of speaker location features in meeting diarization," Ph.D. dissertation, Univ. of Washington, Seattle, 2008.

[13] K. J. Han, S. Kim, and S. S. Narayanan, "Strategies to improve the robustness of agglomerative hierarchical clustering under data source variation for speaker diarization," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 8, pp. 1590–1601, Nov. 2008.

[14] M. Huijbregts, D. van Leeuwen, and F. de Jong, "Speech overlap detection in a two-pass speaker diarization system," in *Proc. Interspeech*, Brighton, U.K., Sep. 2009.

[15] X. Anguera, C. Wooters, and J. Pardo, "Robust speaker diarization for meetings: ICSI RT06s evaluation system," in *Proc. Mach. Learn. Multimodal Interact. (MLMI)*, Berlin, Germany, Oct. 2007, vol. 4299, Ser. Lecture Notes in Computer Science, Springer-Verlag.

[16] K. Boakye, B. Trueba-Hornero, O. Vinyals, and G. Friedland, "Overlapped speech detection for improved speaker diarization in multiparty meetings," in *Proc. ICASSP*, Las Vegas, NV, 2008, pp. 4353–4356.
[17] J. M. Pardo, X. Anguera, and C. Wooters, "Speaker diarization for multiple distant microphone meetings: Mixing acoustic features and inter-channel time differences," in *Proc. Interspeech*, 2006.



Marijn Huijbregts received the M.S. (Ir.) degree from University of Twente, Enschede, The Netherlands, in 2001 and the Ph.D. degree in automatic speech recognition for multimedia retrieval from University of Twente, Enschede, The Netherlands, in 2008, with the main focus of this study was on robust speaker diarization for which he visited the International Computer Science Institute in Berkeley, CA.

He is a Postdoc Researcher at Radboud University, Nijmegen, The Netherlands. After the M.S. degree, he worked in industry on the development of Bluetooth headsets and carkits. At the Centre for Language and Speech Technology, Radboud University Nijmegen, The Netherlands, he is currently performing research on speaker retrieval for large broadcast television archives within the research program IM-Pact funded by IBBT and ICTRegie.

David A. van Leeuwen received the Ir. (M.S.) degree from the Delft University of Technology, Delft, The Netherlands, in 1984 and the Dr. (Ph.D.) degree from the University of Leiden, Leiden, The Netherlands, in 1993.

He was with TNO Human Factors in 1994 and has been with Radboud University, Nijmegen, The Netherlands, since 2008. He has worked in various areas in speech technology, with special interests in evaluation of systems. Examples of evaluations he organized are the EU FP5 SQALE project (1995), evaluating large-vocabulary speech recognition systems in four languages, the NFI-TNO Forensic Speaker Recognition Evaluation with 11 international participants in 2003, and the N-Best evaluation of Dutch speech recognition systems in 2008. He has also successfully participated in various NIST Rich Transcription, Speaker and Language Recognition Evaluations. In 2008 he was appointed professor at Radboud University and he coauthored the EU FP7 Marie Curie ITN project "Bayesian Biometrics for Forensics" (BBfor2), which he is currently leading. In recent years, he has been focussing on speaker diarization and automatic speaker and language recognition, with special interest in highly accurate and efficient systems, forensic application scenarios, and calibration.

Chuck Wooters received the B.A. degree in linguistics, the M.A. degree in linguistics, and an interdisciplinary Ph.D. degree in speech recognition, all from the University of California at Berkeley, in 1986, 1988, and 1993, respectively.

He is the Chief Technology Officer at Next IT Corporation, Spokane, WA. He has held research positions in government, industry, and academia. His industrial research has included the development of real-time speech recognition interfaces for surgical robots and other medical devices, and applications for visualization of large quantities of textual data. The majority of his career was spent at the International Computer Science Institute (ICSI), Berkeley, CA. While at ICSI, his research was focused on speech recognition and speaker diarization. In addition to his current duties at Next IT, he also holds an Adjunct Research Faculty position at the Human Language Technology Center of Excellence (HLT-COE), Johns Hopkins University, Baltimore, MD.

Dr. Wooters serves on the IEEE Speech and Language Technical Committee.