AhoLab Speaker Diarisation System for Albayzin 2010

Iker Luengo, Eva Navas, Ibon Saratxaga, Inmaculada Hernáez, Daniel Erro

Dpt. of Electronics and Telecommunications
University of the Basque Country
{iker.luengo, eva.navas, ibon.saratxaga, inma.hernaez, daniel.erro}@ehu.es

Abstract

This paper presents the speaker diarisation system presented by Aholab Signal Processing Laboratory to the Albayzin Speaker Diarization Challenge 2010. The system was built to run on-line, without any recording of the audio data to produce its output. As a result, the whole process must be done in a single iteration, which prevents the use of many optimisation processes that are usually implemented in diarisation systems. In order to minimise the reduction of the accuracy in the output and to maximise computational efficiency, the applied algorithms were carefully selected and some new modifications were implemented.
Index Terms: speaker diarisation, BIC, on-line audio processing

1. Introduction

The aim of speaker diarisation is to detect speaker changes in an audio recording, and to identify which of the resulting speech segments come from the same speaker. That is, the task is to detect 'who speaks when'.

The process is usually divided into different subtasks, each of them dealing with a specific subproblem. Typically the subtasks include speech detection, speaker segmentation, clustering and resegmentation of the audio stream. Usually these subtasks are executed one after another, as a series of processing steps, i.e., each subtask is applied to the whole audio recording before starting the next one. This architecture implies having the whole audio recording available for several processings, and also that no result can be obtained until the recording is finished.

Against this off-line architecture, we propose an on-line algorithm for speaker diarisation, which is capable of performing the whole process in a single iteration. Such algorithm can work with direct audio input or audio streaming, without needing to record it. Furthermore, it is more efficient in terms of execution time and memory requirements. Nevertheless, some accuracy loss is expected, as the system can only rely on past audio samples to make decisions, and not future ones. Also the impossibility of using multi-pass or iterative algorithms may provide suboptimal results.

2. Speaker diarisation: a short overview

2.1. Speech detection

This step is necessary in order to discard audio segments without speech, thus making the following steps easier and less error prone. Depending on which environment the system is going to operate, non-speech segments can contain a large variety of acoustic events: silence, noise, music, applause, shouting, etc. The most common approach is to make a Viterbi segmentation with GMM models trained on labelled data, although sometimes more elaborate models such as multistate HMM are used.

It is possible to use only two models (one for speech and another one for non-speech), although it is convenient to train more specific models when different acoustic events are expected. Often models for noise, music, clean speech, speech+noise and speech+music are used [1], but it is also possible to train separate models for male and female speech or to differentiate wideband and narrowband speech [2]. This way the detection accuracy is improved, assuming there is enough training data.

2.2. Speaker change detection

This is a critical step in the diarisation process. Once the segments without speech are discarded, the locations of the speaker changes must be located. Almost every diarisation system performs this step calculating a distance metric between two adjacent audio windows. If the distance is larger than a given threshold, a speaker change is assumed between both windows. The differences among specific algorithms lie in the choice of the distance metric and the windowing scheme.

One of the most widely used metric for change detection is the Bayesian information criterion (BIC) [3]. BIC is a likelihood criterion penalised by the model complexity, and is usually used for model selection. Therefore, it can be used to estimate if the windowed audio is better modelled with two different distributions (one for each window) or a single one (combining both windows), thus effectively detecting changes in the audio stream. Nevertheless, this algorithm is computationally very expensive. That is why some implementations prefer to use other metrics, which may be less accurate but faster. Examples of these metrics are Hotlling's T^2 [4], the Gaussian divergence [2] or the generalised likelihood ratio (GLR) [5]. It is also possible to use these faster metrics as a first approach, and then refine the detected changes with BIC [6].

Regarding the windowing, the simplest way to implement the algorithm is to use two adjacent fixed-size sliding windows. The distance metric between both windows is calculated, and the peaks in the distance function define the locations of the change points. A more elaborated windowing scheme that is usually applied together with the BIC metric uses a growing window. Every audio frame inside the window is a possible change point, and the BIC value for each of them is calculated. If the highest BIC value inside the window is greater than zero, a change point is detected at the position of that maximum, and the window is reset to its original size. If not, the window grows a fixed length and the process starts again. Generally, the growing window scheme provides better accuracy results, but is also computationally more expensive. Some implementations reduce this computation cost avoiding to search for changes in

improbable places and defining a limit to the window length [7, 8].

2.3. Clustering

Once the boundaries among speakers are known, it is necessary to detect which speech segments belong to the same speaker. This step is usually implemented as a clustering process, the bottom-up clustering being the most common method. The distance between all cluster pairs is calculated, and the pair with least distance is selected and combined. Then, the distance matrix is updated and another pair is selected until the stop criterion is met. Many distance metrics can be used: BIC, GLR, Euclidean distance between GMM, etc.

2.4. Cluster recombination

Although this step is not absolutely necessary, it may improve the final result [1]. The idea here is to under-cluster the segments in the first clustering process, thus having more clusters than speakers, but at the same time ensuring that the clusters have a reasonable amount of speech. A GMM model is created for every cluster by MAP adaptation from a UBM. Finally, the GLR is used in order to identify which clusters to recombine. A new model is created for the newly recombined cluster, and the process is repeated until the stop criterion is met. It has been shown that feature normalisation is necessary to get any improvement with this technique [2].

2.5. Resegmentation

Now that each cluster has a reasonable amount of information, it is possible to train new models for each speaker, and use them in combination with the non-speech models in order to make a new Viterbi segmentation. This way it is possible to refine the segmentation boundaries. This process can be repeated iteratively for increased accuracy.

3. Proposed algorithm description

Figure 1 shows a schematic diagram of the proposed diarisation algorithm. The algorithm is based on an efficient implementation of a BIC change detector and an on-line speaker clustering. The change detector works with MFCC features without derivatives, whereas the speech detector appends first and second derivatives to this parametrisation. Furthermore, the BIC algorithm uses voiced frames only, discarding any unvoiced segment. In the following sections, each step in the algorithm will be explained with more detail.

3.1. Speech detection

A separate GMM model was trained for music, noise, clean speech, speech+music and speech+noise, using the development recordings and the audio segmentation labels provided by the contest organisation. These models are used in a Viterbi segmentation in order to detect audio segments with and without speech.

As the process ought to be on-line, an on-line Viterbi algorithm as described in [9] was implemented. This modified algorithm keeps track of the active paths and efficiently detects whether they converge in some point or not. In the case of all the paths converging in a point, the segmentation decision up to that point can be extracted, without loosing any accuracy and without needing to wait until the whole file has been processed. Furthermore, the part of the trellis that contained the consolidated path can be erased from memory, as it is not needed anymore.

Development experiments showed that the addition of first and second derivatives of MFCC provides slightly better segmentation results.

3.2. Speaker segmentation

For the initial speaker change detection, a growing window architecture and BIC metric are used. The growing window provides better results than a fixed-size sliding window, but the computational cost is also larger. In order to reduce the time of computation as much as possible, the solution described in [8] is used:

- No speaker change is searched in the first and last 2 seconds of the window.
- The window grows 2 seconds every time that no change is detected.
- Once the window reaches 20 seconds, instead of growing, it becomes a sliding window.
- For each window, a speaker change is searched every 250 ms. If a change is located, the search is refined to 50 ms.
- Once a change is found, the window size is reset to 5 seconds.

This solution provides the accuracy of a growing-window algorithm accuracy, while keeping the window size and the amount of calculation to a minimum. Furthermore, the calculation of the BIC values is also optimised by using a buffer of cumulative sums as described in work made by Cettolo and Vescovi [8].

Development results showed that discarding unvoiced frames and using only voiced ones decreased the diarisation error in a 12%. Therefore, only voiced frames were used for the speaker change detection. Similarly, it was confirmed that the use of feature derivatives was not convenient for this task.

3.3. Clustering

An on-line clustering that uses BIC metric was implemented, following the description given in [7]. Every time the speaker segmentation algorithm detects a new boundary, the newly extracted speech segment is immediately given to the clustering process. This process computes the BIC of this new segment against all known clusters, and selects the one with lowest BIC value. If this value is under a given threshold, the segment is assigned to that cluster and the cluster statistics are updated. If not, a new cluster is formed with the new segment.

This on-line clustering is theoretically suboptimal, since the clustering is performed without the information of forthcoming segments. However, in practice the on-line clustering may give better results than the bottom-up off-line clustering. The reason is that the speaker segmentation algorithm over-segments the speech, providing false speaker boundaries. Therefore, two consecutive segments are more likely to belong to the same speaker than segments far apart. The on-line clustering makes the clustering decision more locally, thus reinforcing the combination of adjacent segments. This algorithm not only does the clustering on-line, but it also provides better results.

3.4. Voiced unvoiced detection

As described before, the speaker change detection step uses only voiced frames, discarding the unvoiced ones. In order to

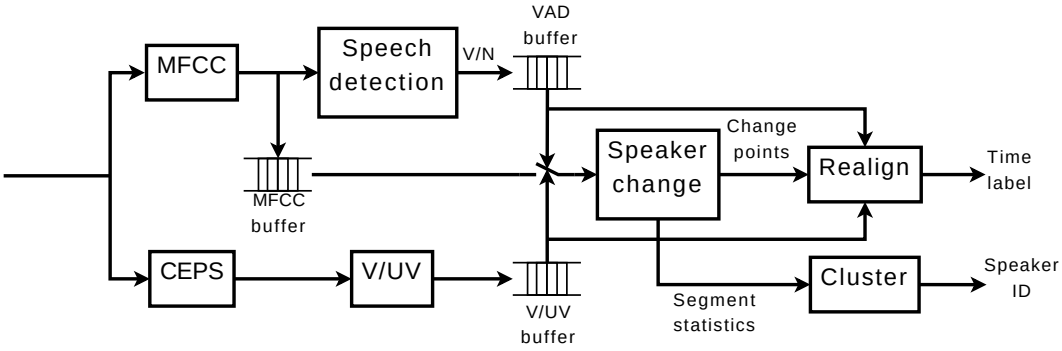

Figure 1: Schematic diagram of the diarisation algorithm.

make the voiced/unvoiced (VUV) estimation, the PTHCDP algorithm described in [10] was used. This algorithm uses cepstrum transformation and dynamic programming in order to estimate the F_0 curve and the VUV information. The algorithm was modified in order to use the on-line Viterbi algorithm, so that partial paths can be extracted with the VUV information every time the active paths converge.

3.5. System integration

The whole algorithm is meant to run on-line in a single iteration. But the clustering subprocess must wait until a speaker change is detected, and the speaker change subprocess is idle until the speech detector and the VUV detector make a decision. Both detectors make decisions asynchronously, depending on their own Viterbi algorithm and the convergence of their paths. Meanwhile, new feature frames are being generated from the audio input. For the synchronisation of all these subprocesses, a series of buffers and control points must be used.

Every time a new frame is collected, it is parametrised and delivered to the speech detection and VUV detection algorithms. Furthermore, the MFCC parametrisation is saved in a buffer for later use. In order to deal with the asynchronous behaviour of the speech detection and VUV detection algorithms, their outputs are directly stored in the corresponding buffer as soon as path convergence is detected. Now, let N be the number of speech decisions available in the VAD buffer, and M the number of VUV decisions available in the corresponding buffer. Then, we have a total of $min\{N, M\}$ new frames with all decisions made, so that they can be further processed. These frames are extracted from the MFCC buffer and the ones without speech and the voiced ones are discarded. The rest (if any) are provided to the speaker change algorithm.

As these frames are not needed anymore, they are deleted from the MFCC buffer, in order to save memory. At the same time, the time realignment process gets some information from the speech detection and VUV decisions for later use, and these decisions are also deleted from the buffers.

Whenever the speaker change algorithm finds a new boundary, it outputs the necessary statistics for the clustering process, together with the time instant in which this change happens. As this algorithm takes only voiced frames as input, this time mark does not consider non-speech or unvoiced frames. Therefore, a time realignment is necessary in order to convert the detected change times into absolute times. Finally, the clustering process outputs a unique speaker label, and the time realignment system outputs the corresponding time labels.

4. Analysis of the results

The presented algorithm was submitted as the primary system to the Albayzin Speaker Diarisation Challenge 2010. Nevertheless, two other systems were also presented as reference. The first one was an off-line version of the main algorithm, in which each subprocess (speech detection, VUV decision, speaker change detection and the clustering of the segments) was executed one after the other. The second one was an off-line version in which unvoiced frames were also considered for the speaker change detection algorithm, thus not needing a VUV decision step. This section compares the three systems in terms of diarisation accuracy and execution speed. See Table 1 for the error rates and Table 2 for the execution speed of each one of these systems.

The main difference between the on-line and off-line versions is post-processing. The algorithms used in both cases are the same, but the off-line architecture allows us to post-process the outcome of each step before going into the next one, which is not possible in the on-line system. For example, speech detection labels were post-processed in order to discard silences shorter than 500 ms, before feeding them to the speaker change detection algorithm. This provides a better estimation of speech activity, which is reflected in a much lower missed speaker error rate (see Table 1). However, the results from the speaker change detection or the clustering systems are not post-processed. As a result, the speaker error rate does not change much between the on-line and off-line architectures, and the small difference is due to the speech detection labels being more accurate.

Table 1 also shows the effect of discarding unvoiced frames for the speaker change detection step. When unvoiced frames were used in the off-line system, the speaker error rate increased a 16%. As the speech detection algorithm always uses both voiced and unvoiced frames, there is no difference in the missed speaker and false alarm error rates.

It is also interesting to compare the systems in terms of execution speed. Table 2 shows the average CPU time required in order to process one hour of speech. For the off-line implementations, the time required for each step is also shown. These measures were made on a quad-core Intel Xeon 2.27 GHz computer with 6 GB memory. Nevertheless, these values are not meant to be an absolute measure of the complexity of each system, but can be used in order to see which one is faster.

The on-line architecture makes a single iteration over the speech data, whereas the off-line systems must perform several iterations and post-processings. As a result, there is a significant difference in the processing speed. The off-line architec-

Error time (in %)	Off-line w/ unvoiced	Off-line	On-line
Missed speaker	2.8	2.8	4.9
F-alarm speaker	1.2	1.2	1.5
Speaker error	26.9	23.1	23.9
Diarisation error	31.00	27.17	30.38

Table 1: Error rates for each system on the Albayzin Speaker Diarisation Challenge 2010.

CPU time (in seconds)	Off-line w/ unvoiced	Off-line	On-line
Speech Detection	36.0	36.0	–
VUV detection	–	81.0	–
Diarisation	47.1	44.2	–
Total	83.1	161.2	126.1

Table 2: Mean CPU time for one hour speech for each one of the systems.

ture needs 161 CPU seconds to process one hour of speech, whereas the on-line systems needs 126 seconds. This means that the on-line implementation is a 22% faster, mainly due to the reduction in the number of iterations.

It is outstanding that half of the execution time of the off-line system without unvoiced frames is spent in the VUV detection step. When unvoiced frames are not discarded, this step is unnecessary, and the total execution time decreases to 83 seconds. Although we have not the corresponding measurements, is is expected that an on-line system without VUV detection would be even faster.

5. Conclusions

Most of the current speaker diarisation systems rely on an off-line architecture, in which several processing steps are performed over the same audio recording, one after the other. In this paper an on-line diarisation system has been described. This algorithm requires a single iteration in order to process the audio, and can be used with direct audio input or audio streaming. This features makes it suitable for applications where the recording of the signal is not a possibility.

As all the processing must be done in a single iteration, it is not possible to post-process the outcome of each step before going into the next one. Therefore, a certain increase of the error rate is unavoidable. The results show that the on-line architecture has indeed a 12% increase in the total diarisation error rate when compared to the off-line system. This increase is mostly due to a higher missed speaker error rate, which in turn occurs because it is not possible to post-process the speech detection labels.

Nevertheless, making the system run on a single iteration has its advantages in terms of speed. It has been shown that the on-line architecture is a 22% faster than the same algorithms running in an off-line fashion.

The possibility of including or discarding unvoiced frames in the speaker change detection algorithm has also been studied. On the one hand, discarding these frames provided a significant reduction in the speaker error rate. On the other hand, in order to discard these frames, a VUV detection step is mandatory, which increases the computational cost of the system. For example, the VUV detection algorithm that was used in the experiments [10] takes half of the total execution time. If the diarisation algorithm must run on devices with severe processing restrictions, faster VUV detection algorithms should be used, or even the whole VUV detection step can be avoided if the speaker change detection is performed with both voiced and unvoiced frames.

6. Acknowledgements

This work has been partially supported by the Spanish Ministry of Science and Innovation (Buceador Project, TEC2009-14094-C04-02) and The Basque Government (Berbatek, IE09-262).

7. References

[1] D. A. Reynolds and P. Torres-Carrasquillo, "The MIT Lincoln Laboratory RT-04F diarization systems: Applications to broadcast audio and telephone conversations," in *NIST Rich transcription Workshop*, Palisades, NY, USA, Nov. 2004.

[2] R. Sinha, S. E. Tranter, M. J. F. Gales, and P. C. Woodland, "The cambridge university march 2005 speaker diarisation system," in *Interspeech*, Lisbon, Portugal, Sep. 2005, pp. 2437–2440.

[3] S. S. Chen and P. S. Gopalakrishnan, "Speaker, environment and channel change detection and clustering via the bayesian information criterion," in *DARPA speech recognition workshop*, 1998, pp. 127–132.

[4] B. Zhou and J. Hansen, "Unsupervised audio stream segmentation and clustering via the bayesian information criterion," in *ICSLP*, Beijing, China, Sep. 2000, pp. 714–717.

[5] S. Meignier, D. Moraru, C. Fredouille, J. F. Bonastre, and L. Besacier, "Step-by-step and integrated approaches in broadcast news speaker diarization," *Computer Speech and Language*, vol. 20, pp. 303–330, 2006.

[6] P. Delacourt and C. J. Wellekens, "DISTBIC: A speaker-based segmentation for audio data indexing," *Speech Communication*, vol. 32, pp. 111–126, 2000.

[7] A. Tritschler and R. A. Gopinath, "Improved speaker segmentation and segments clustering using the bayesian information criterion," in *Eurospeech*, Budapest, Hungary, Sep. 1999, pp. 679–682.

[8] M. Cettolo and M. Vescovi, "Efficient audio segmentation algorithms based on the bic," in *International Conference on Acoustics, Speech, and Signal Processing (ICCASP'03)*, vol. 6, april 2003, pp. 537–5340.

[9] R. Šrámek, "The on-line Viterbi algorithm," Master's thesis, Comenius University, Bratislava, 2007.

[10] I. Luengo, I. Saratxaga, E. Navas, I. Hernáez, J. Sánchez, and I. Sainz, "Evaluation of pitch detection algorithms under real conditions," in *ICASSP*, Honolulu, USA, Apr. 2007, pp. 1057–1060.