

A FAST, ACCURATE AND STREAM-BASED SPEAKER SEGMENTATION AND CLUSTERING ALGORITHM

An Vandecatseye, Jean-Pierre Martens

ELIS, Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Gent (Belgium)
{avdecats,martens}@elis.rug.ac.be

Abstract

In this paper a new pre-processor for a free speech transcription system is described. It performs a speech/non-speech partition, a segmentation of the speech parts into speaker turns, and a clustering of the speaker turns. It works in a stream-based mode, and it is aiming for a high accuracy with a low delay and processing time. Experiments on the Hub4 Broadcast News corpus show that the newly proposed pre-processor is competitive with and in some respects better than the best systems published so far. The paper also describes attempts to raise the system performance by supplementing the standard MFCC features with prosodic features such as pitch and voicing evidence.

1. Introduction

All free speech transcription systems seem to comprise a pre-processor to detect long non-speech segments, to locate changes of speaker or acoustic conditions in the speech parts, and to label the intervals between successive change points.

The architecture of our pre-processor is very similar to that of e.g. [3, 4], but our algorithms are designed for stream-based (online) processing and optimized so as to achieve a high accuracy in combination with a low delay and computational load. We will mainly discuss algorithmic details which have contributed to an increased accuracy and/or a reduced complexity. We will also describe our attempts to improve the system accuracy by including prosodic features in the acoustic feature vector. Finally, we will pay attention to evaluation methodologies and parameter fixing.

All our experiments were performed on the DARPA Broadcast News corpus. It consists of (i) the BN96 train data, (ii) the BN96 development data and (iii) the BN97 evaluation data. The first two parts are used for training and parameter tuning¹, the third part is considered as the test set. The correct frame labels (speech/non-speech, acoustics and speaker identity) can be derived from the STM-files supplied with the corpus.

The acoustic front-end comprises (i) a cepstral feature extractor producing 12 MFCCs and a log-energy per frame of 10 ms and (ii) a prosodic feature extractor adding a pitch and a voicing evidence to each frame. The pitch extractor is AMPEX [8]. It was chosen for its excellent performance and its robustness against noise.

The rest of this paper is organized as follows. The first three sections describe the main building blocks of our pre-processor and their performance in combination with the standard acoustic features. The next section reviews our first attempts to raise that

performance by introducing prosodic features, and the paper ends with a short conclusion.

2. Speech/non-speech segmentation

The goal of the speech/non-speech segmenter is to detect long intervals of non-speech that can be discarded for further processing. In this paper, long means longer than 2 s (200 frames).

2.1. The GMM-HMM core

The core of the speech/non-speech segmenter is a looped HMM comprising an initial and a final state and five acoustic states, each modeled by a GMM. There are GMMs for *speech*, *speech+music*, *speech+other*, *music* and *other* frames, and each GMM comprises 64 Gaussian mixtures with diagonal covariance matrices.

Based on the observation that a lot of the frames are expected to be either speech or speech+other, a transition probability of 0.35 is attached to the transitions from the initial to the two speech states. The remaining 0.3 is equally divided over the other transitions leaving the initial state. There are no probabilities on the transitions leaving the acoustic states. The only free parameter of the HMM automaton is an inter-segment penalty P_s which is used to control the average length of the generated segments.

The frames are processed by a time-synchronous Viterbi algorithm. The last N_{buf} frames are kept in a ring buffer. If at a certain moment the likelihood of the best path to one acoustic state is larger than 10^5 times the highest likelihood of the paths to the other acoustic states, then only this dominant path is maintained and all frames of the buffer are labeled as speech or non-speech. If the buffer is full before this happens, the system searches for the frame where the likelihood dominance of one path is maximal, and it labels all the buffered frames preceding this frame. The smaller N_{buf} is, the smaller the maximum decision delay is, but the more likely it is to get a sub-optimal labeling.

2.2. The S/NS post-processor

The GMM-HMM core tends to produce short non-speech intervals at pauses between words. Since these pauses can be handled properly by a recognizer, there is no need to generate them here. We therefore eliminate all short (<2 s) non-speech segments. Another problem is caused by the fact that noises in long pauses may create very short 'speech' segments which disturb the detection of these long pauses. Therefore, very short (<0.2 s) speech segments between sufficient long (>0.4 s) non-speech segments have to be eliminated first. These two post-processing operations can cause a maximal delay of 2.6 s on top of the delay emerging from the buffer size. However by assigning preliminary labels

¹ Some less important parameters were not really tuned but were given values on the basis of common sense and some quick verification on a few development files

to the frames in the buffer (namely that of the dominant acoustic state), and by basing the decision on these labels, this extra delay can be avoided. The loss in performance caused by using those preliminary labels, turns out to be insignificant.

2.3. Evaluation

Methodology. As in [1] we compute the percentages of correctly classified speech and non-speech frames. Obviously, short non-speech segments appearing in the correct segmentation are relabeled to speech prior to the evaluation.

Settings. Since adding accelerations did not raise the performance, only 26 acoustic features were used (12 MFCCs + log-energy + derivatives).

System performance. In Table 1 we have listed the correctly classified speech and non-speech frames for different choices of P_s . From the results on the development data we derived an

P_s	sp (%)	non-sp (%)	sp (%)	non-sp (%)
0.063	98.35	60.15	99.18	77.47
0.125	98.56	59.30	99.42	76.59
0.250	98.85	57.60	99.50	76.44
0.500	99.15	53.34	99.62	69.18
1.000	99.29	51.15	99.62	69.18
	development data		test data	

Table 1: Correctly classified speech and non-speech frames as a function of P_s . In bold are the results for the 'optimal' P_s which was derived from the development data.

'optimal' $P_s = 0.25$. We did so as follows. Starting with $P_s = 1$ we checked whether going to $P_s/2$ would yield a larger gain in correctly classified non-speech frames than a loss in correctly classified speech frames. If so, P_s was divided by 2, else it was selected as the 'optimal' value. For $P_s = 0.25$ we get 99.5% of the speech and 76.4% of the non-speech frames correct in the test data. Apparently, the HTK system [1] performs a little better : on the test data it reaches 99.82% speech and 70.4% non-speech correct. However, this is accomplished with GMMs comprising 1024 mixtures whose means and variances are adapted to the file under test. Such an adaptation does not comply with our aim to build a stream-based system. Without adaptation, the HTK system removes 59% of the non-speech while maintaining 99.75% of the speech frames.

Maximum delay. The performance seems to saturate as soon as N_{buf} becomes larger than 300 (=3 s). We therefore used this value in all further tests.

Computational load. On a PC with a 866 MHz processor the algorithm takes about 7.5% of real-time.

3. Speaker segmentation

The task of this module is to look for changes in circumstances in the course of the speech parts. Since most of these changes are speaker changes, we talk about speaker segmentation and we refer to the segments between changes as speaker turns. In analogy to [2], we propose a system consisting of two stages :

1. A boundary generation stage to generate candidate change points at places of maximum differences between the statistical distributions of the acoustic vectors in two fixed length windows (N_w frames) at both sides of that place.

2. A boundary elimination stage to remove some of the boundaries on the basis of a statistical analysis of the formed speaker turns.

The second stage is activated whenever a predefined number (N_{smax}) of speech frames is delivered by the speech/non-speech segmentation, or whenever the end of a speech segment is detected.

3.1. The boundary generation stage

To reduce the CPU-time, the frames are grouped in blocks of 10 frames, and only maxima at block boundaries are pursued. To avoid obvious dependencies of the distance on the size d of the acoustic vector and the length N_w of the considered windows, we introduce the normalized log-likelihood ratio (NLLR) for full covariance gaussian distributions as our metric. If Σ_{left} , Σ_{right} and Σ_{both} are the ML (Maximum Likelihood) covariance matrices for $(n - N_w, n)$, $(n, n + N_w)$ and $(n - N_w, n + N_w)$, the NLLR at frame n is given by

$$NLLR(n) = \frac{2 \log |\Sigma_{both}| - \log |\Sigma_{left}| - \log |\Sigma_{right}|}{2d}$$

It represents the LLR per frame and per acoustic feature.

3.2. The boundary elimination stage

When there are enough speech frames available, the analysis of candidate boundaries n_i (if any) is started. The aim is to eliminate n_i if there is no apparent advantage in modeling the acoustics in (n_{i-1}, n_i) and (n_i, n_{i+1}) by two distinct distributions. The Bayesian Information Criterion (BIC) [7] states that the quality of model \mathcal{M} to represent $\mathbf{x}_1, \dots, \mathbf{x}_N$ is given by

$$BIC(\mathcal{M}) = \log \mathcal{L}(\mathbf{x}_1, \dots, \mathbf{x}_N | \mathcal{M}) - \frac{\lambda}{2} k(\mathcal{M}) \log N$$

with \mathcal{L} representing the likelihood. The complexity term $k(\mathcal{M})$ is equal to the number of free parameters of model \mathcal{M} . Theoretically, λ should be equal to 1, but in practice it is varied to control the number of eliminated boundaries.

To decide on n_i , the BICs for two ML models of interval (n_{i-1}, n_{i+1}) are computed : a single-gaussian model \mathcal{M}_1 and a double-gaussian model \mathcal{M}_2 (one gaussian for (n_{i-1}, n_i) and one for (n_i, n_{i+1})). From these values we derive

$$\Delta BIC(n_i) = \frac{BIC(\mathcal{M}_2) - BIC(\mathcal{M}_1)}{(n_{i+1} - n_{i-1} + 1) \cdot d}$$

as the BIC-difference per frame and per feature. The boundary elimination stage now selects the n_i with the smallest associated $\Delta BIC(n_i)$, it rejects n_i if $\Delta BIC(n_i)$ is negative, and it repeats this process on the new segment configuration until no n_i can be eliminated anymore. By normalizing our ΔBIC , we attain that the a priori chance of selecting a boundary is independent of the length of the segments surrounding that boundary. It was verified that a normalized ΔBIC does consistently outperform a non-normalized one.

3.3. Evaluation

Methodology. If n_{ri} and n_{cj} are real (correct) and computed change points respectively, they are linked to one-another if (i) n_{cj} is the computed boundary closest to n_{ri} and (ii) n_{ri} is the real boundary closest to n_{cj} and (iii) the distance between n_{ri} and n_{cj} is less than 1 s. If there is a gap between two real speaker

turns, the corresponding n_{ri} is allowed to be at any place in that gap.

From the formed links one can determine the *recall* (=percentage of real boundaries linked to a computed one) and *precision* (=percentage of computed boundaries linked to a real one). One can also compute a global error measure such as $E=(2D+I)/3N$ (D =deletions, I =insertions, N =number of real changes) expressing that deletions are worse than insertions.

Settings. The input vector consists of 12 MFCCs ($d = 12$). After some trials on the development data, N_w was set to 400 and N_{smax} to 1500. We also tried smaller block sizes, but as they did not yield larger recalls for large precisions, we stucked to blocks of 10 frames.

System performance. We have evaluated the system on two tasks: (i) detection of true speaker turns (515 in the test), and (ii) detection of all turns (speaker and/or acoustics) (624 in the test). The results on the test set are depicted in Figure 1 as a function of λ . For both tasks, the global error on the development data is

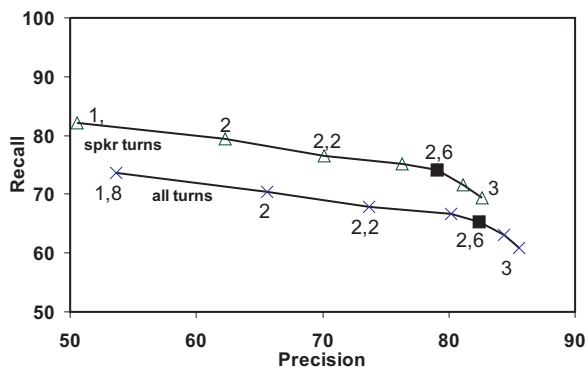


Figure 1: Recall and precision on the test set for speaker turn and all turn detection as a function of λ .

minimal for $\lambda = 2.6$. For this λ we obtain a (recall,precision) of (74.2%,79.1%) for the speaker and (65.4%,82.4%) for the all turn task (see filled squares on Figure 1).

Recently, Ajmera et al [5] proposed an HMM-based method for speaker segmentation. Using our evaluation software, they found a recall of 65% and a precision of 68% for the pure speaker change task.

In [7, 6] a one-stage speaker segmentation system based on BIC is proposed. Using an incompletely specified evaluation strategy (but mentioning discount of insertions in music regions), a (recall,precision) of (75.3%,90%) was reported for the all turn task. However, in [5], the same algorithm was implemented and evaluated with our software. This yielded only 66% recall and 71% precision. Apparently, our evaluation methodology is much stricter.

Maximum delay. The performance seems to saturate as soon as N_{smax} becomes as large as 1500. The maximum decision delay of the speaker segmentation is then equal to $N_{smax} + N_w = 1900$ frames. Since $N_w > N_{buf}$, the speech/non-speech segmentation causes no extra delay.

Computational load. On a PC with a 866 MHz processor the algorithm takes no more than 0.3% of real time.

4. Speaker clustering

The clustering of speaker turns is fully integrated in the speaker segmentation, meaning that it does not introduce any additional delay. Whenever a new set of speaker turns $\Omega_S = \{S_1, \dots, S_{N_S}\}$ is available, it is supplied to the clustering algorithm which then updates the cluster set, denoted as $\Omega_C = \{C_1, \dots, C_k, \dots, C_{N_C}\}$.

We have conceived a new algorithm and compared it with two existing on-line algorithms. All algorithms use BIC as a decision criterion: $\Delta BIC(C, S)$ measures the improvement per frame and per feature of a two-gaussian over a single-gaussian model for the frames of $C + S$. The complexity factor is proportional to some λ_c which is used to control the number of clusters being generated.

4.1. A basic algorithm

A simple strategy is to cluster the speaker turns S_i one by one in their order of appearance :

1. Calculate $\Delta BIC(C_k, S_i) \forall k \leq N_C$, determine the cluster with the minimal ΔBIC and call it C_m .
2. If $\Delta BIC(C_m, S_i) < 0$, then merge S_i with C_m , else add it as a new cluster to Ω_C .

4.2. The IBM algorithm

In [6] a more refined algorithm is proposed. It considers $\Omega_X = \Omega_C \cup \Omega_S$ as a new cluster set and it determines for every $S_i \in \Omega_S$ its closest neighbour - X with the minimal $\Delta BIC(X, S_i)$ - in Ω_X . The corresponding Δ is called the *log-distance* of S_i to Ω_X . The algorithm then selects the S_i with the lowest log-distance, and if it is negative, S_i is merged with the corresponding X . The process is repeated until no merges can be performed anymore. The remaining speaker turns in Ω_S are then also added to Ω_C .

4.3. A new clustering algorithm

It is known [7] that BIC is not working well if there is a large mismatch between the sizes (number of frames) of its two arguments. Therefore, we propose not to merge a speaker turn with a cluster whose size already exceeds a threshold N_{cmax} . Furthermore, instead of searching for merges first, we propose to give preference to the formation of new clusters. This leads to the following loop to be repeated until Ω_S is empty:

1. $\forall S_j \in \Omega_S$ and $\forall C_k \in \Omega_C$: calculate the 'log-distance' $ld_{jk} = \Delta BIC(S_j, C_k)$.
2. $\forall S_j \in \Omega_S$: record the minimal ld_{jk} , and call it ld_j , the log-distance of S_j to Ω_C .
3. Keep track of the $S_j \in \Omega_S$ with the minimal and the one with the maximal log-distance to Ω_C .
4. If the *maximal* log-distance is positive, then add the corresponding S as a new cluster to Ω_C and delete it from Ω_S . Else, select the S with the *minimal* log-distance and do the following: (i) establish the nearest cluster in Ω_C , (ii) if the size of this cluster is smaller than N_{cmax} , merge S with this cluster, and (iii) delete S from Ω_S .

4.4. Evaluation

Methodology. As in [4] we use the *average cluster purity* (P_{av}), defined as the mean of the percentages of correctly classified frames per speaker turn, as our first cluster quality measure. However, since cluster purities tend to increase when N_c becomes larger, they are always represented as a function of N_c . Furthermore, since some frame mis-classifications can be due to segmentation errors (e.g. boundary shifts), we also introduce

the notion of an *ideal clustering*. The latter assigns to all frames of a *computed* speaker turn the most frequently occurring *correct* label (based on STM file) in that turn. The ideal clustering yields an ideal number of clusters N_{ic} and a cluster purity P_{iav} .

Settings. The acoustic feature vector consists of 12 MFCCs. For the new algorithm, the maximum cluster size T_{cmax} is set to 1000 frames (10 seconds).

System performance. On Figure 2 we have depicted average cluster purities for the test set as a function of λ_c . Apparently,

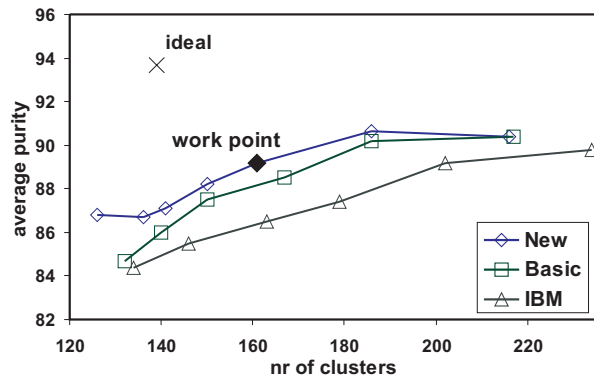


Figure 2: Average cluster purities for the three tested algorithms (varying λ_c). The cross represents the ideal clustering.

the newly proposed algorithm outperforms the basic algorithm which on its turn, a bit surprisingly, outperforms the IBM algorithm. In [4] a figure of 89% is mentioned for P_{av} , but this no information on the number of generated clusters is provided.

The 'optimal' $\lambda_c = 3.2$, was determined as the value yielding the best approximation of the ideal clustering on the development data. I.e., the value minimizing the deviation

$$D = |N_c - N_{ci}|/N_{ci} + (P_{iav} - P_{av})/(1 - P_{iav})$$

This point is marked as a black diamond on Figure 2.

Computational load. On a PC with a 866 MHz processor the algorithms takes only 0.4% of real time.

5. Adding prosodic features

In this section we discuss some first attempts to improve our system by including prosodic features (pitch and voicing evidence) in the acoustic feature vector.

First we investigated whether we could improve the speech/non-speech segmentation by taking the percentage of voiced frames in the generated segments into account in the S/NS post-processor. We were not successful.

Then we tried to improve the boundary elimination stage of the speaker segmentation module. Two strategies were investigated: (i) adding prosodic features to the cepstral feature vector, and (ii) considering the prosodic feature vector as an independent feature vector. In the latter approach we add a fraction (α) of the prosodic to the cepstral ΔBIC component. In the first approach, the pitch of an unvoiced frame is replaced by the mean pitch in the speech segment of interest. In the second approach, the prosodic component is based on the statistics of the voiced frames only. The λ -factor of the prosodic ΔBIC is optimized by performing a speaker clustering with the prosodic components

only, and by minimizing the deviation from the ideal clustering. The fraction α is then used to control the importance of the prosodic component. Our experiments showed that none of the two approaches could offer any improvement.

Since the average pitch is reckoned to be an important speaker characteristic, there is reason to believe that it can help to improve the quality of the clustering algorithm. Again, we tried the same two approaches we tested in connection with the speaker segmentation. By applying the second approach, some small improvement of the clustering can be attained, too small however to justify the implied extra work.

6. Conclusion

We have shown that it is possible to build a stream-based speaker segmentation and clustering system with a very low processing time (8.2% of real time on a 866 MHz processor) and an acceptable maximum delay (19 s).

Using well established evaluation methods, we could show that our system is competitive with and in some respects better than previously published systems performing the same tasks.

Finally, thus far, we were unable to improve any part of our system by adding prosodic features to the input vector. More sophisticated methods may be needed in this respect.

7. Acknowledgement

This research was supported by the Flemish Institute for the Promotion of the Scientific and Technical Research in the Industry (contract ADV/STWW/000151).

8. References

- [1] T. Hain, S.E. Johnson, A. Tuerk, P.C. Woodland and S.J. Young (1998), "Segment Generation And Clustering in the HTK Broadcast News Transcription System", *DARPA Broadcast News Transcription and Understanding Workshop*, 133-137.
- [2] P. Delacourt, D. Kryze and C.J. Wellekens (1999), "Detection of Speaker Changes in an Audio Document", *Proceedings Eurospeech*, 179-182.
- [3] J.L. Gauvain, L. Lamel and G. Adda (1998), "Partitioning and Transcription of Broadcast News Data", *Proceedings ICSLP-98*, 1335-1338.
- [4] M. Harris, X. Aubert, R. Haeb-Umbach and P. Beyerlein (1999). "A Study of Broadcast News Audio Stream Segmentation and Segment Clustering", *Proceedings Eurospeech*, 1027-1030.
- [5] J. Ajmera, I. McCowan, I. Lapidot (2002). "BIC revisited and applied to speaker change detection", *IDIAP Research Report 02-39*.
- [6] A. Tritschler and R. Gopinath (1999). "Improved Speaker Segmentation and Segments Clustering Using the Bayesian Information Criterion", *Proceedings Eurospeech-99*, 679-682.
- [7] S.S. Chen and P.S. Gopalakrishnan (1998). "Speaker, Environment and Channel Change Detection and Clustering via the Bayesian Information Criterion", *DARPA Broadcast News Transcription and Understanding Workshop*, 127-132.
- [8] L. Van Immerseel, J.P. Martens (1992). "Pitch and voiced/unvoiced determination with an auditory model", *J. Acoust. Soc. Am.* 91, 3511-3526.