IMPROVED SPEAKER SEGMENTATION AND SEGMENTS CLUSTERING USING THE
BAYESIAN INFORMATION CRITERION

Alain Tritescher and Ramesh Gopinath

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
email: alain@us.ibm.com

ABSTRACT

Detection of speaker, channel and environment changes in
a continuous audio stream is important in various applica-
tions (e.g., broadcast news, meetings/teleconferences etc.).
Standard schemes for segmentation use a classifier and hence
do not generalize to unseen speaker / channel / environ-
ments. Recently S. Chen introduced new segmentation and
clustering algorithms, using the so-called BIC. This paper
presents more accurate and more efficient variants of the
BIC scheme for segmentation and clustering. Specifically,
the new algorithms improve the speed and accuracy of seg-
mentation and clustering and allow for a real-time imple-
mentation of simultaneous transcription, segmentation and
speaker tracking.

1. INTRODUCTION

The segmentation of continuous audio is useful as a pre-
processor for further classification of the segments for speaker
identification/verification, noise rejection, music removal etc.
In automatic transcription applications such a segmentation
scheme allows the creation and use of speaker / channel / envi-
ronment-specific acoustic models for improved transcrip-
tion accuracy. In several of these applications clustering
of segments from the same speaker / channel / environ-
ment is also useful. Segmentation and clustering can be
used in conjunction in speaker tracking applications. To-
gether they can be used to increase the amount of adapta-
tion data for unsupervised adaptation of acoustic models in
transcription applications. In general they allow specialized
processing of the audio for specific speakers / channels / envi-
ronments. This paper presents improvements (both speed and
accuracy) to algorithms for segmentation and clustering based
on the Bayesian Information Criterion (BIC) intro-
duced recently in [1]. These improvements have allowed us
to create an application that concurrently segments, tran-
scribes, identifies and tracks speakers in broadcast news au-
dio in real-time.

The paper is organized as follows: Section 2 briefly re-
views the BIC, which is the key concept used in both the
segmentation and clustering algorithms. Section 3 describes
the new version of the segmentation algorithm and Sec-
ton 4 describes improvements to the clustering algorithm.
Section 5 describes how these new algorithms are incorpo-
rated in a real-time transcription, segmentation and speaker
identification and tracking system for broadcast news.

2. THE BAYESIAN INFORMATION CRITERION

BIC is an asymptotically optimal Bayesian model-selection
scheme used to decide which of $p$ parametric models best
represents $n$ data samples $x_1, \ldots, x_n, x \in \mathbb{R}^d$. Each model
$M_j$ has a number of parameters, say $k_j$. We assume that
the samples $x_i$ are independent.

According to the BIC theory [3], for sufficiently large $n$,
the best model of the data is the one which maximizes

$$BIC_j = \log L_j(x_1, \ldots, x_n) - \frac{1}{2} \lambda k_j \log n$$ (1)

with $\lambda = 1$, and where $L_j$ is the maximum likelihood of the
data under model $M_j$ (i.e., the likelihood of the data with
maximum likelihood values for the $k_j$ parameters of $M_j$).

In the particular case where there are only two models
we have a simple test for model selection: choose the model
$M_1$ over $M_2$ if $\Delta BIC = BIC_{1} - BIC_{2}$ is positive.

Note that BIC can also be viewed as a penalized maxi-
mum likelihood technique [3, 1].

3. SEGMENTATION USING BIC

3.1. BIC for segmentation

In this paper standard 24-dimensional mel-cepstral feature
vectors generated at 10ms intervals from the continuous au-
dio stream form the data samples (or frames). The audio
stream is from a Broadcast news source sampled at 16KHz
with 16-bit PCM. The basic problem is to identify all pos-
sible frames where there is a segment boundary. Without
loss of generality consider a window of consecutive data
samples $\{x_1, \ldots, x_n\}$ in which there is at most one segment
boundary. In this case the basic question of whether or not
there is a segment boundary at frame $i$ can be cast as a
model selection problem between the following two models:
model $M_1$ where $\{x_1, \ldots, x_n\}$ is drawn from a single full-
covariance Gaussian, and model $M_2$ where $\{x_1, \ldots, x_n\}$
is drawn from two full-covariance Gaussians, with $\{x_1, \ldots, x_i\}$
drawn from the first Gaussian, and $\{x_{i+1}, \ldots, x_n\}$ drawn
from the second Gaussian. Since $x_i \in \mathbb{R}^d$, model $M_1$ has
$k_1 = d + \frac{d(d+1)}{2}$ parameters, while model $M_2$ has twice as
many parameters ($k_2 = 2k_1$).

It is straightforward to show [1] that the $i^{th}$ frame is a
good candidate for a segment boundary if the expression :

$$\Delta BIC_i = -\frac{n}{2} \log |\Sigma| + \frac{i}{2} \log |\Sigma_j| + \frac{n-i}{2} \log |\Sigma_i|$$

$$+ \frac{1}{2} \lambda(d + \frac{d(d+1)}{2}) \log n$$

is positive. This is equivalent to the number of parameters
in the second model being larger than the number of parame-
ters in the first model, by the difference in the BIC values.
is negative, where $|\Sigma_w|$ is the determinant of the covariance of the whole window (i.e., all $n$ samples), $|\Sigma_s|$ is the one of the covariance of the first subdivision of the window, and $|\Sigma_i|$ is the determinant of the covariance of the second subdivision of the window.

Applying this test for all possible values of $i$ and choosing $i_0$ with the most negative $\Delta BIC_i$, we can detect a segment boundary in the window: if $\Delta BIC_{i_0} < 0$, then $x_{i_0}$ corresponds to a segment boundary. If the test fails then more data samples are added to the current window and the process repeated with this new window of data samples.

3.2. Improvements in speaker segmentation

For the above segmentation scheme to be useful in real-time applications it several enhancements are necessary. Moreover, the original algorithm presented in [1] has a high (25%) miss rate on small segments (say segments during 2 seconds or less). In applications where segmentation is followed by clustering missing segments is a more severe error than introducing spurious segments since clustering can take care of eliminating spurious segment boundaries from the segmentations. This is particularly true in applications like the one described in Section 5 - concurrent transcription, segmentation and speaker identification. Indeed, even without clustering, in applications like speaker identification, spurious boundaries (assuming no speaker identification errors) lead to consecutive segments being labeled the same - which is tolerable. However, missed boundaries lead to two problems. Firstly, one of the speakers cannot be identified. Secondly, the other speaker will also be poorly identified since that speaker’s audio data is “corrupted” by data from the missed speaker. In the previous version of the segmentation algorithm, the missed turns of less than 2 seconds by themselves constituted 74.9% of the errors. Therefore, particular attention has been given to the problem of the small segments and the error rate on missed turns has been reduced by more than 10%.

Two main improvements are presented here: in order to improve the accuracy, especially on small segments, a new windows selection scheme has been designed; in order to improve speed, a more effective way to compute the BIC tests has been found.

3.2.1. Variable window scheme

The choice of the windows on which the segment boundaries detection is performed is very important. The reason of the sensitiveness of the algorithm to this choice comes from the clear trade-off that has to be done: if the window chosen contains too many vectors, some boundaries are likely to be missed. If on the other hand the window chosen is too small, lack of information will result in poor representation of the data by the Gaussians.

The segmentation algorithm in [1] adds a fixed amount of data to the current window if no segment boundary has been found. Such a scheme does not take advantage of the ‘contextual’ information to improve the accuracy: the same amount of data is added, whether or not a segment boundary has just been found, or no boundary has been found for a long time.

The improved algorithm makes it possible to consider a relatively small amount of data in areas where new boundaries are very likely to occur, and to increase the window size more generously when boundaries are not very likely to occur.

Initially, a window of vectors of a small size is considered (typically 100 frames of speech).

If no segment boundary is found on the current window, the size of the window is increased by $\Delta N_i$ frames. If no boundary found on this new window, the number of frames is increased by $\Delta N_{i+1}$, with $\Delta N_i = \Delta N_{i+1} + \delta_i$, where $\delta_i = 2\delta_{i+1}$; until a segment boundary is found or the window extension has reached a maximum size (in order to avoid accuracy problems if a boundary occurs). This ensures an increasing window size which is pretty slow when the window is still small, and is faster when the window gets bigger.

When a segment boundary is found in a window, the next window begins after the detected boundary, using the minimal window size.

3.2.2. More efficient BIC tests

Although the scheme just explained also speeded up the overall processing time, the big gain came from a second improvement, consisting in better selecting the locations where BIC tests are performed.

It is in fact possible to arbitrarily eliminate some of the BIC tests in the window, when they correspond to locations where the detection of a boundary is very unlikely.

First, the BIC tests are not performed at the very borders of each window, since they necessarily represent one Gaussian with very little data (this apparently small gain is repeated over segments detections and actually has no negligible performance impact).

There is a second case in which some computations can be avoided. Assume the current window is large (this happens quite often in broadcast news data). If all the BIC tests are performed, the BIC computations at the beginning of the window will have been done several times, with some new information added each time. Now, if no segment boundary has been found in the first 5 seconds, say, in a window size 10 seconds, it is quite unlikely that a boundary will be hypothesized in the first 5 seconds with an extension of the current 10 second window. This suggests that one can decrease the number of BIC computations by ignoring BIC computations in the beginning of the current window. In fact, the maximum number of BIC computations is now an adjustable parameter, tweaked according to the speed/accuracy level required.

By doing so, the algorithm has now a behavior very desirable in the real-time, which is that it is possible to know the maximum time it takes before having some feedback on the segmentation information. Because even if no segment boundary has been found yet, if the window is big enough one knows that there is no segment is present in the first frames. This information can be used to do some other processings on this part of the speech signal.

3.2.3. The penalty weight used in the BIC

A parameter has been introduced in the BIC formula, the penalty weight $\lambda$, in order to compensate for the differences between the theory and the practical application of the criterion.

In our experiments we found that the best value of $\lambda$ that gives a good tradeoff between miss rate and false-alarm
rate is 1.3. For a more comprehensive study of the effect of λ on the segmentation accuracy for the Broadcast News transcription task see the discussion in [2].

In principle, this means that the factor λ is task-dependent and has to be re-tuned for every new task. However, in practice we have applied the algorithm to different types of data (Broadcast News, Telephony data,...), and there is no appreciable change in performance by using the same value of λ in these different tasks.

3.3. Experimental Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>False Alarms</th>
<th>Missed turns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>&lt; 2 secs</td>
<td>≥ 2 secs</td>
</tr>
<tr>
<td>Original</td>
<td>4.1 %</td>
<td>33.4 %</td>
</tr>
<tr>
<td>Improved</td>
<td>9.2 %</td>
<td>24.7 %</td>
</tr>
</tbody>
</table>

Table 1: Segmentation results on the HUB4 1997 BN data

We see that we have been able to reduce the error rate for the missed segment boundaries of almost 11%, and of even more than 12% for the small segments. This is done at the expense of an increase in the false alarms rate though, but still very limited.

In terms of speed, segmentation takes a small fraction (< 10%) of real-time on a Pentium 200 NT workstation.

4. CLUSTERING USING THE BIC

4.1. Using the BIC for clustering

The use of the Bayesian Information Criterion is more straightforward in the clustering scheme than it is in the segmentation scheme. The problem is here more how to make it efficient.

Let’s assume we have a set of clusters \( C_1, \ldots, C_k \). In an offline scheme, the problem to solve is to try to merge one of the clusters with another one, leading to a new set of clusters \( C'_1, \ldots, C'_{k'} \), where one of the new clusters is the merge between two previous clusters.

In order to see whether it is good to merge two clusters \( C_i \) and \( C_j \), two models are built: the first model, say \( M'_1 \), is a Gaussian model computed with the data of \( C_i \) and \( C_j \), which leads to \( BIC'_1 \). The second model, \( M_2 \), keeps two different Gaussians, one for \( C_i \) and one for \( M'_1 \), and gives \( BIC_2 \). We see that we are back in the same situation as in the segmentation scheme, and that therefore it is better to keep two distinct models if \( \Delta BIC = BIC_1 - BIC_2 < 0 \). If this difference of BIC is positive, the two clusters are merged and we have the desired new set of clusters.

In [1], it has been shown how to practically implement the clustering in a bottom-up fashion, i.e. starting with all the initial segments, and building a tree of clusters by merging the closest nodes of the tree (the measure of similarity being the BIC). However, this is a scheme which only works in an offline way, and we now focus on what issues appear in the online scheme.

4.2. Clustering the segments online

The online clustering involves first the clusters found in the previous iterations (or calls to the clustering procedure), and the new segments to cluster. We shall refer to the former by \( C_i \) to \( C_K \), and to the latter by \( S_1 \) to \( S_M \).

When the new segments are added, the merging gains \( g(C_i, S_j) \) for all possible \((i, j)\) pairs, and the merging gains \( g(S_i, S_j) \) are computed, with \( g(X_i, X_j) = BIC(merging \ X_i \ and \ X_j) - BIC(keeping \ X_i \ and \ X_j \ separated) \), where \( X_i \) generically represents a cluster or a new segment.

Then, the largest likelihood gain \( \Delta BIC_{MAX} \) is looked at. If \( \Delta BIC_{MAX} > 0 \), the two elements of the pair \( \{C_i, S_j\} \) or \( \{S_i, S_j\} \) leading to \( \Delta BIC_{MAX} \) are merged together, possibly creating a new cluster if the pair is of the form \( \{S_i, S_j\} \). On the other hand, if \( \Delta BIC_{MAX} < 0 \), the two components are kept distinct, leading to one new cluster if the pair is \( \{C_i, S_j\} \), or two new clusters if the pair is \( \{S_i, S_j\} \).

This process is repeated until the \( M \) new segments have been clustered.

This is theoretically a suboptimal algorithm in comparison to the offline bottom-up clustering, since the maxima we are considering for the \( \Delta BIC \) values can be local in the online scheme, as opposed to the global maxima found in the online version. However, practically the opposite happens. Part of the reason can be that the optimal segments merges are those corresponding to segments close in time. The online algorithm would make it easier to associate such segments to the same cluster.

In order to reduce the influence of the non-reliable small segments to cluster, only the segments with sufficient data are clustered; the others are gathered in a separate 'garbage' cluster.

4.3. Experimental results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Purity</th>
<th>Number of Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline</td>
<td>96.7 %</td>
<td>172</td>
</tr>
<tr>
<td>Online</td>
<td>98.58 %</td>
<td>149</td>
</tr>
</tbody>
</table>

Table 2: Clustering performed on the BIC segments on HUB4 1997 BN data

The clustering algorithm described works efficiently in both offline and online (where the cluster information is needed in a very short time) modes. Experimental results show that the algorithm described is superior in terms of speed, cluster purity and number of clusters on the 1997 DARPA Hub4 Broadcast News Evaluation data.

5. TRANSCRIPTION, SEGMENTATION AND SPEAKER IDENTIFICATION

The segmentation and clustering algorithms in this paper have been implemented in a real-time application for the Broadcast News transcription and Speaker Identification. The application runs on a Windows platform and uses a set of tools collectively referred to as Viavoice Technologies for Broadcast Speech Transcription and Speaker Identification.
The transcription engine is IBM's ViaVoice for Broadcast News and is accessed by the application via IBM's SMAPI calls. The application also obtains the audio from ViaVoice and gives it to the segmentation/clustering module that returns segment/cluster information with a confidence score. The resulting segments and clusters are provided to the Speaker Identification engine for identification and verification.

Speaker identification is done via the standard SVAPI calls to IBM's Speaker ID engine provided as part of the ViaVoice toolkit. The ID engine uses a pre-enrolled pool of speakers for identification. The audio and segment/cluster information from the application are used to identify the speakers in each segment from the pre-enrolled pool. Some of the standard techniques used for speaker identification at IBM is given in [5].

Another interesting application of the segmentation is described in [4].

6. CONCLUSION

This paper describes improvements in speed and accuracy and enhancements to BIC-based algorithms for segmentation of continuous audio and clustering of similar segments. These improvements and enhancements were used to create an application for automatic transcription, segmentation, speaker identification and tracking of broadcast news. The application uses commercially available components for transcription (IBM ViaVoice '98 for Broadcast News transcription) and speaker identification (IBM ViaVoice Identified).

7. REFERENCES


