INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Developing On-Line Speaker Diarization System

Dimitrios Dimitriadis[1], *Petr Fousek*[2]

IBM, [1]Yorktown Heights, USA, [2]Prague, Czech Republic

dbdimitr@us.ibm.com, petr_fousek@cz.ibm.com

Abstract

In this paper we describe the process of converting a research prototype system for Speaker Diarization into a fully deployed product running in real time and with low latency. The deployment is a part of the IBM Cloud Speech-to-Text (STT) Service. First, the prototype system is described and the requirements for the on-line, deployable system are introduced. Then we describe the technical approaches we took to satisfy these requirements and discuss some of the challenges we have faced. In particular, we present novel ideas for speeding up the system by using Automatic Speech Recognition (ASR) transcripts as an input to diarization, we introduce a concept of active window to keep the computational complexity linear, we improve the speaker model using a new speaker-clustering algorithm, we automatically keep track of the number of active speakers and we enable the users to set an operating point on a continuous scale between low latency and optimal accuracy. The deployed system has been tuned on real-life data reaching average Speaker Error Rates around 3% and improving over the prototype system by about 10% relative.

Index Terms: speaker diarization, speaker recognition, i-vectors, speech recognition

1. Introduction

Speaker diarization is a task of determining "who spoke when" in a multi-speaker environment. Speaker diarization (or just diarization in this paper) is an essential component of many consumer-demanded tasks processing large volumes of audio, such as information retrieval or audio analytics, thus drawing attention of the research world. However, commercial availability of systems including diarization is still limited. One of the reasons is that the performance of these systems largely varies depending on the application scenario. Diarization on recordings from call centers, broadcast news or meetings pose different challenges [1]. An on-line system[1], which would work reliably across all scenarios, still remains an open technical problem. Diarization systems presented in the literature appear to work relatively well when specific conditions are met, such as low noise levels or no speaker overlap.

The research community has been actively working on improving the overall robustness by using better acoustic representations (or embeddings) such as the i-vectors [2, 3, 4], by improving the speaker models using better clustering approaches, like the Agglomerative Clustering or Variational Bayes Systems [5, 6], etc. Unstable performance on short audio or on initial parts of audio can be alleviated by introducing a warm-up period [7] or by presetting the number of speakers [8].

In this paper we focus on a two-stage on-line approach, where we first perform an unsupervised audio segmentation into homogeneous segments hypothesized to have been uttered by a single speaker and then assign them a speaker label. We have adopted a number of existing technologies such as the Bayesian Information Criterion (BIC), the i-vector features, Within Class Covariance Normalization (WCCN) [2, 9, 10], etc. Herein, we propose improvements to some of them, such as a novel mechanism to update previously seen cluster results, based on the prefix method [11]. The updates are due to the additional acoustic information seen in the meantime.

In Section 2 we overview our diarization pipeline and then we focus at specific parts requiring attention from a development point of view. This means revisiting the unsupervised audio segmentation, speaker clustering and its continuity, and the mechanism of providing results when considering limited CPU resources and low-latency requirements. Section 3 presents experiments relevant to optimizing the system for deployment. This includes measuring the amount of audio required for accurate diarization, comparing the performance of different speaker clustering approaches and sampling from the trade-off between low-latency and optimal accuracy. Finally, we summarize what we have learnt from converting a research prototype to a deployed diarization system in Section 4.

[1]A system running at real time and with low latency on any input

2. System Description

Diarization works essentially in two steps: First, it attempts to find homogeneous audio segments likely been spoken by a single speaker. Then, a speaker label is assigned to each such segment by matching the audio to a set of models representing the speakers (or their clusters). Each of these steps is rather simple when working offline, but requires revisiting the algorithms to run efficiently on-line.

An example of a system integrating Diarization and ASR is shown in Fig. 1. Audio is end-pointed by a Speech Activity Detector (SAD), then augmented with speaker labels provided by the Diarization Component and routed to the appropriate speaker-specific ASR system.

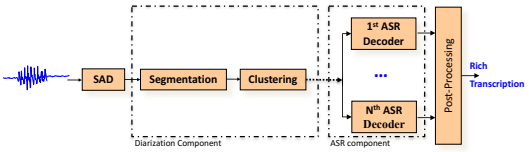

Figure 1: *Diarization and ASR systems in tandem. Given a speaker label, the audio is routed into one of the speaker-specific ASRs.*

2.1. Speech Activity Detection

Speech Activity Detector is one of the key components for the diarization systems [12]. The quality of the i-vector features carrying the speaker information may drop significantly when extracted from audio containing non-speech, ultimately degrading the diarization performance. Here we investigate two dif-

http://dx.doi.org/10.21437/Interspeech.2017-166

ferent SAD approaches, a neural-network based SAD [13] and a SAD based on the output of a speech recognizer.

Initially, we attempted to improve the performance of an ASR system by implementing a pipeline as in Fig. 1. The speaker labels from the diarization system were used to route the audio into one of N identically initialized, tandem ASR systems. The hypothesis was that each ASR system would adapt to the acoustics specific to the particular speaker or cluster thereof. The number of speaker clusters and ASR engines was fixed to the maximum number of speakers expected to be present in each of the recordings. A neural-network SAD was trained with in-domain data. Since the primary goal of the SAD was to optimize ASR performance and not diarization, there was a bias towards labeling more non-speech audio as speech rather than rejecting speech. The end-pointed audio was passed to the Segmenter module detecting potential speaker turns within the contiguous audio segments coming from the SAD. This turn detector was based on the BIC algorithm [9]. Finally, the resulting segments were eventually assigned a speaker label by the Clustering module before being routed to the respective ASR engine.

Nonetheless, we were unable to improve the ASR performance. The problem was that the Segmenter was generating either too many false positives or it ignored the true speaker turns. Many of the false positives were the result of the SAD passing through chunks of non-speech, ultimately triggering the split points. Even in the case of the Segmenter parameters being tuned for reasonable performance, it still did not generalize across different tasks. Finally, some split points were inserted in the middle of words, ruining any ASR attempts.

This experiment motivated us to revisit the overall architecture by switching the order of the ASR and Diarization modules. This approach brings many benefits but the main drawback is the ASR cannot improve when using the diarization results, unless a feedback loop is introduced. Consequently, issues with channel differences or different audio levels from speakers [11] remain unaddressed. On the other hand, a scenario with multiple speaker-dependent ASR systems can still be implemented on top of this system. The rest of the paper will follow this ASR→Diarization architecture, Fig. 2.

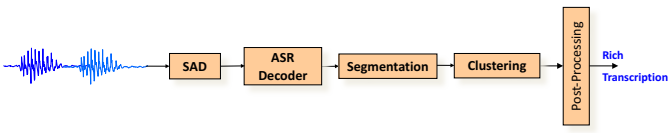

Figure 2: *Diarization system using ASR as an advanced Speech Activity Detector. Word boundary positions of the transcript are used as candidate speaker turn points.*

2.2. Speaker Segmentation

The ASR system is not just an additional layer on top of the SAD but it introduces an important new functionality. It provides accurate end-pointing around words, thus eliminating most of false-positives from the model-based SAD because it seldom emits words in non-speech segments and finally, it significantly reduces the possible speaker-turn points only to the boundaries between words[2], which also accelerates the segmentation. Under these constraints, the new Segmenter performance and efficiency has been much improved.

As mentioned earlier, the goal of the speaker segmenter is to process contiguous audio segments, marking possible speaker turn points. Given the standard cepstral acoustic features and the ASR output as inputs the process is the following: for each word boundary within the contiguous segment, the segment is split into two sub-segments (left to the boundary, right to the boundary). Two Gaussians are fit to features from the respective sub-segments and then compared. If they are different enough, it marks a possible speaker-turn point. We use standard cepstra and not i-vectors for this step because the segments are relatively short. The comparison of the Gaussians is done using the BIC algorithm, as mentioned above. If a sub-segment is shorter than about 2s, the T^2-criterion is used instead [9].

2.3. Speaker Clustering

With the possible speaker turns detected, the next step is to assign a speaker label to each audio segment. In more detail: one i-vector per homogeneous audio segment is obtained [10, 2] and post-processed by WCCN, unit-length normalization and a smoothed PCA transform [10, 14]. The Universal Background Model (UBM) for the i-vector is obtained as detailed in [10].

Two different clustering approaches were explored: an agglomerative (bottom-up) clustering [10, 11] and more recently, a top-down approach based on the "X-means" algorithm [15]. Both approaches are primarily designed for offline processing when the entire audio is available beforehand. On-line mode in contrast requires incremental results as new audio comes in. A brute-force approach would be to re-run the clustering from scratch with every new input segment but that would be expensive, and bring an issue of temporal discontinuity: the clustering algorithms examined here provide un-ordered clusters, i.e. the labels obtained from two subsequent runs are unrelated. The greedy algorithm [10], where the clustering is run only once after a warm-up period and then the existing clusters are only updated, does not suffer from this issue and it is computationally very efficient. On the other hand, it was shown [11] to be significantly less accurate than re-clustering. It is also sensitive to the initial conditions, it does not converge to the off-line solution and it assumes the number of speakers fixed. Because of these drawbacks we decided to build on the re-clustering idea.

X-means clustering is a variation of k-means algorithm that refines cluster assignments by repeatedly attempting cluster subdivision, while keeping the most meaningful splits based on a given stopping criterion (herein, BIC is used). Our X-means implementation uses the first few PCA components even though it starts from the same features as the bottom-up clustering. In case of bottom-up clustering, the stopping criterion determining the final number of clusters can be a threshold of the clustering error loss. Alternatively, one could find a knee of the tradeoff curve between the number of clusters and the clustering error loss and stop clustering when the knee is reached.

Table 3 compares the performance of the bottom-up clustering vs X-means, with X-means showing superior performance in most cases[3]. One of the reasons is that the bottom-up clustering used the simple thresholding as a stopping criterion, which proved to be rather unstable. Also, the bottom-up approach may be more prone to clustering errors due to the individual noisy i-vectors. On the other hand, X-means models the clusters by

[2]We assume that no speaker turn happens intra-word.

[3]Some of these cases are presented in Table 3. In general, the X-means algorithm outperforms on average the bottom-up clustering based on our extensive internal benchmarking and testing.

Gaussians so it evaluates distances between smoothened distributions. In terms of complexity, X-means approach turns out to be computationally less intensive since the number of iterations needed remains low (most often 2 or 3) and it is linear with the number of segments. On the contrary, bottom-up clustering complexity is $n^2 \log(n)$. X-means thus seems to be a more viable clustering approach.

The complexity of the clustering sets an upper bound on the number of training samples we can process in real-time. For long audio files, clustering cannot be re-run on the complete history of segments. The history must be limited to the N latest segments and this is herein called the *"active window"*[4]. The trade-off between the active window length and accuracy is further discussed in Section 3. Limiting the history means that the solution doesn't converge anymore to the off-line clustering, while causing a new phenomenon of "speaker leaving", i.e. when a certain speaker is no longer present within the active window, as discussed below.

Significant speedup was obtained when switching to the ASR→Diarization pipeline which only requires running the clustering when there is a new hypothesis from the ASR. This typically happens once per sentence[5]. Another speed-up can be gained by initializing the bottom-up clustering by k-means and then continuing with the clustering, but this is again suboptimal w.r.t. the off-line solution.

2.4. Speaker Label Updates

Here we extend the algorithm of "prefix label estimation and reconciliation" [11] to support more than two speakers while addressing two particular challenges of on-line processing. The first one is how to deal with a varying number of speakers over time, i.e. when a new speaker comes in or another leaves. The second challenge is balancing the tradeoff between delivering results as fast as possible while maintaining the highest possible accuracy.

Briefly, the prefix method works as follows: At any given time, the clustering is performed on all the i-vectors accumulated so far (to the limits of the active window). The more audio processed, the more accurate the speaker representation by the clusters is. As mentioned before, clusters from two different clusterings are unordered and so are the speaker labels. A solution is the reconciliation algorithm [11] which considers two parallel label sequences obtained with two cluster sets (previous and current) on the same portion of the audio. The algorithm examines all possible permutations of the current labels and picks the permutation which gives the lowest Hamming distance between both label sequences. In other words, it permutes the current labels so that they are the most similar to the previous ones. For example, on a six-word sentence let the previous labels be "113332" and current labels be "223331". We see that by swapping current labels "1" and "2" we reach the original sequence. For simplicity, this example does not consider changes in individual labels due to differences in clusters. Realistic label sequences include such changes, and are usually long. According to our experience the least-Hamming-distance reconciliation always works adequately well.

Two special cases occur when a speaker leaves the active window or when a new speaker enters it. Assuming that the stopping criterion for clustering works ideally, i.e. the number of clusters equals to the number of speakers, a new cluster appears or correspondingly is lost. In case a cluster is lost, its label disappears and the question is whether to reuse that label should a new cluster ever appear or leave that label reserved and unused. Considering that the only information available is within the active window, there is no way to tell if a new speaker has been seen before or not. Therefore we decided that unallocated labels are allowed to be re-assigned to future clusters. This maintains the number of active labels low even on long audio files. A drawback of allowing cluster labels to be recycled is that a label used to represent a particular speaker, when reused, can represent a completely different speaker. On the other hand, these two speakers appear at minimum tens of minutes apart from each other, so in practice this does not represent an issue.

The second challenge is the latency-accuracy tradeoff. In order to serve equally well users whose application requires low-latency labels and those who prefer sparing latency for higher precision, we have introduced the concept of label updates. At any given time the on-line diarization system delivers results in two parts: the first part contains the incremental speaker labels for the new audio. These are low-latency results and may possibly be noisy. The second part carries possible updates to any previously produced labels that lay within the active window. Thus, the end users have a choice between running with the latest results or keep applying updates before using possibly more accurate labels later.

3. Experiments

For the system evaluations and benchmarking 3 different test sets are used, all manually annotated: an internal IBM call-center (IBM-CC) set, the LDC CallHome set [16] after "collapsing[6]" the stereo audio and an internal call center database in Spanish (SP-CC). Two different CallHome subsets have been created based on the number of speakers, i.e. the original set has 120 sentences with 11 out of them containing more than two speakers (CH-120). The second CallHome subset (CH-109) is created with the remaining 109 sentences. The length of the audio files is 10min. The audio quality of this set is particularly good, with low noise and limited overlapping speech. The SP-CC set is more challenging with a lot of background typing-, babbling- and ringing noises. The number of speakers is strictly two and the length of the files ranges from one to several minutes. Finally, the IBM-CC contains 71 customer-agent interactions of various audio lengths. Although the audio is manually annotated, a significant chunk of audio contains ring tones, pre-recorded messages and music and remained un-transcribed. That part of the audio makes this set particularly challenging because the speaker clustering algorithm tends to classify these noises as separate speakers. Worse, the algorithm includes the corresponding i-vectors in the speaker clusters, skewing their statistics and widening their variance. After all, this is one of the reasons why algorithms using thresholds as the stopping criterion, e.g., the bottom-up approach, cannot perform efficiently. All experiments are focused on the speaker clustering error ignoring the SAD-related errors since these errors depend solely on the SAD and the ASR performance (thus mentioned only once in Table 2). That part of the system remains always the same for all the experiments (there are no changes of the ASR configuration), thus these rates cannot change.

First, the tradeoff between diarization accuracy and the length of the available audio was investigated. All audio files were truncated in sequence to 5, 10, 30, 60, 90 and 120s.

[4] N is in the order of hundreds, representing about 15min of audio.

[5] The speaker clustering is run only on the finalized ASR hypothesis.

[6] The stereo audio has been merged to single-channel audio.

Table 1: *Diarization Performance as a function of the audio length.*

Speaker Clustering Error (in frames %)						
Task	5.0	10.0	30.0	60.0	90.0	120.0
SP-CC	18.75	10.76	9.56	5.21	4.01	5.17
CH-109	14.34	13.25	7.06	3.94	4.91	3.46
CH-120	14.58	13.33	7.26	3.89	4.91	3.76
IBM-CC	10.22	1.25	5.15	2.07	2.04	1.92

As seen in Table 1 the clustering error stabilizes after 60s. The outlier seen for the 10s (IBM-CC) case can be attributed to either to single speaker being present in the truncated audio or the SAD letting through too few frames.

The next experiment compares the diarization performance of two extreme use cases, the lowest-latency results with no updates ("No Updates" column) vs. fully-updated results, i.e. those available after the audio has left the *active window* ("Updates" column), shown in Table 2. Applying all updates significantly improves the diarization accuracy. Herein, the active window length is about 15min.

Table 2: *Performance with low-latency results (No Updates) or with updates applied over the active window length (Updates).*

Diarization Performance (in frames %)				
	Missed Sp.	FAlarm	Spk. Cluster. Err	
Task	Rate	Rate	No Updates	Updates
SP-CC	21.71	0.92	8.65	6.28
CH-109	10.82	0.55	4.86	3.02
CH-120	10.91	0.51	4.82	3.24
IBM-CC	18.45	4.93	4.73	3.36

The influence of the active window length on the real-time factor of the diarization is shown in Fig. 3. With the active window no longer than about 20min the diarization runs under $0.1\times$RTF, i.e. ten times faster than at real-time speed. Note that if diarization is run in batch mode, i.e. delivering results once every 15min, the CPU usage is negligible because the clustering runs only once every 15min.

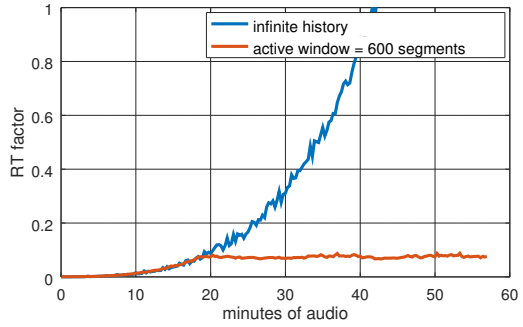

Figure 3: *Comparing speed of diarization with and without limits on the length of the active window.*

Two different UBMs, based on PLP and MFCC features, have been trained for the i-vector extraction. The models are trained according to [10] with the sole differences of: 1. the frame length equals to 20ms (vs. 32ms) to match the sampling rate of the ASR feature extraction process, 2. the PLP model does not use a mel-filterbank but the features are directly mapped from the log-magnitude domain to the acoustic nonlinearity, 3. the number of PLP coefficients equals to 12, and 4. the mel-filterbank used for the MFCC-based model contains 24 filters spanning the frequency range 200-3500Hz.

The two first columns of Table 3 compare the diarization performance of i-vectors extracted using PLP or MFCC, respectively. We sticked to the better performing MFCCs and performed another experiment comparing bottom-up vs. X-means clustering using MFCC-based i-vectors (compare columns 2 and 3) showing mixed results. When X-means system was constrained to produce at least 2 clusters its accuracy increased (compare columns 3, 4), however in reality we have no prior information about the number of speakers.

Table 3: *Diarization Performance for different UBMs and clustering algorithms.*

Speaker Clustering Errors (in frames %)				
	Clustering Configuration			
Task	Bottom-up PLP	Bottom-up MFCC	X-Means MFCC	X-Means† MFCC
SP-CC	8.52	4.57	5.93	3.95
CH-109	4.08	2.14	1.79	1.29
CH-120	4.50	2.33	1.75	1.30
IBM-CC	3.34	3.49	4.16	3.52

†Constrain system to find 2 or more speakers.

4. Conclusions

Turning a research prototype into a deployed system requires satisfying multiple, often contradictory, requirements. The deployed system have to be stable and robust to a wide range of input conditions yet being simple to configure and use. In order to do so, many decisions must be made automatically without exposing them to the user. At the same time the system must run with limited CPU and memory resources and still deliver accurate results at low latency.

We have shown that SAD together with the speaker turn detector are critical parts of the system. Large improvements in speed and accuracy of the diarization process were obtained when using the ASR system as an input to the diarization module (even though ASR cannot benefit from the speaker information in this architecture). We introduced X-means as an alternative speaker clustering approach and compared its performance to the baseline bottom-up approach on different tasks. We have also shown that MFCC features are preferred over PLP for i-vector extraction. Finally, we introduced an on-line re-clustering approach capable to deal with variable number of speakers.

The deployed diarization system augments automatic speech transcript with speaker labels, without adding on the latency of the ASR results. It runs on-line yet allows to reach the upper bound of the accuracy by enabling updates of previously released results. It is robust to non-speech acoustic events and it automatically keeps track of the number of speakers. Finally, the system is available for multiple languages and sampling rates.

5. Acknowledgements

The authors would like to thank W. Zhu, K. Church and J. Pelecanos for their contributions. The diarization system could not be deployed without their help.

6. References

[1] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals, "Speaker Diarization: A Review of Recent Research," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 356–370, February 2012.

[2] S. H. Shum, N. Dehak, R. Dehak, and J. R. Glass, "Unsupervised Methods for Speaker Diarization: An Integrated and Iterative Approach," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2015 – 2028, Oct. 2013.

[3] S. H. Shum, N. Dehak, and J. R. Glass, "On the Use of Spectral and Iterative Methods for Speaker Diarization," in *INTERSPEECH*, 2008.

[4] G. Dupuy, M. Rouvier, S. Meignier, and Y. Esteve, "i-Vectors and ILP Clustering Adapted to Cross-show Speaker Diarization," in *INTERSPEECH*, 2012.

[5] D. Reynolds, P. Kenny, and F. Castaldo, "A Study of New Approaches to Speaker Diarization," in *INTERSPEECH*, 2009, pp. 1047 – 1050.

[6] M. Senoussaoui, P. Kenny, T. Stafylakis, and P. Dumouchel, "A Study of the Cosine Distance-Based Mean Shift for Telephone Speech Diarization," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 217 – 227, Jan. 2014.

[7] H. Aronowitzh, "Trainable Speaker Diarization," in *INTERSPEECH*, 2007.

[8] H. Aronowitz, Y. Solewicz, and O. Toledo-Ronen, "Online Two Speaker Diarization," in *EEE Odyssey - The Speaker and Language Workshop*, 2012.

[9] B. Zhou and J. H. L. Hansen, "Efficient Audio Stream Segmentation via the Combined T^2 Statistic and Bayesian Information Criterion," *IEEE Transactions on Speech, and Audio Processing*, vol. 13, no. 4, pp. 467–474, July 2005.

[10] W. Zhu and J. Pelecanos, "Online Speaker Diarization Using Adapted i-Vector Transforms," in *ICASSP*, 2016, pp. 5045 – 5049.

[11] K. Church, W. Zhu, J. Vopicka, J. Pelecanos, D. Dimitriadis, and P. Fousek, "Speaker diarization: A perspective on challenges and opportunities from theory to practice." in *ICASSP*, 2017.

[12] X. Anguera, M. Aguilo, C. Wooters, C. Nadeu, and J. Hernando, "Hybrid Speech/Non-speech Detector Applied to Speaker Diarization of Meetings," in *IEEE Odyssey - The Speaker and Language Workshop*, 2006.

[13] S. Thomas, G. Saon, M. V. Segbroeck, and S. S. Narayanan, "Improvements to the IBM Speech Activity Detection System for the DARPA RATS Program," in *ICASSP*, 2015.

[14] L. Lu, Y. Dong, X. Zhao, J. Zhao, C. Dong, and H. Wang, "Analysis of Subspace Within-class Covariance Normalization for SVM based Speaker Verification," in *INTERSPEECH*, 2008.

[15] D. Pelleg and A. Moore, "X-means: Extending K-means with Efficient Estimation of the Number of Clusters," in *17th Intern. Conf. on Machine Learning*, 2000, pp. 727–734.

[16] A. Canavan, D. Graff, and G. Zipperlen, "CALLHOME American English Speech – LDC97S42," Linguistic Data Consortium, Tech. Rep., 1997.